ISO/JTC 1/SC 29 **N2203**

Date: 1998-05-15

ISO/IEC FCD 14496-3 Subpart 1

ISO/JTC 1/SC 29/WG11

Secretariat:

Information Technology - Very Low Bitrate Audio-Visual Coding

Part 3: Audio

Subpart 1: Main Document

Authors: **B. Grill, B. Edler, R. Funken, M. Hahn, K. Iijima, N. Iwakami, Y. Lee, T. Moryia, J. Ostermann, S. Nakajima, M. Nishiguchi, T. Nomura, W. Oomen, H. Purnhagen, E. Scheirer, N. Tanaka, A.P. Tan, R. Taori, MPEG-2 AAC Authors (IS -13818-7)**

Document type: International standard
Document:sub-type if applicable
Document:stage (20) Préparation
Document:language E

C:\ISOST18\BASICEN.DOT ISOSTD Basic Version 1.8 1996-10-30

ISO/JTC 1/SC 29 **N2203PAR**

Date: 1998-03-20

ISO/IEC FCD 0.1 14496-3 Subpart 2

ISO/JTC 1/SC 29/WG11

Secretariat:

Information Technology - Very Low Bitrate Audio-Visual Coding

Part 3: Audio

Subpart 2: Parametric Coding

Document type: International standard
Document:sub-type if applicable
Document:stage (20) Préparation
Document:language E

C:\ISOST18\BASICEN.DOT ISOSTD Basic Version 1.8 1996-10-30

FCD 0.1 14496-3 Subpart 2

Parametric coding

1 Introduction: Parametric Decoder Core

The parametric decoder core provides two sets of decoder tools. The HVXC (Harmonic Vector eXcitation Coding) decoder tools allow decoding of speech signals at 2 kbit/s and 4 kbit/s with scalable scheme. HVXC also provides variable bit rate decoding where a typical average bit-rate is around 1.2-1.7 kbps. The HILN (Harmonic and Individual Line plus Noise) decoder tools allow decoding of non-speech signals like music at bit rates of 4 kbit/s and higher. Both sets of decoder tools allow independent change of speed and pitch during decoding and can be combined to handle a wider range of signals and bit rates.

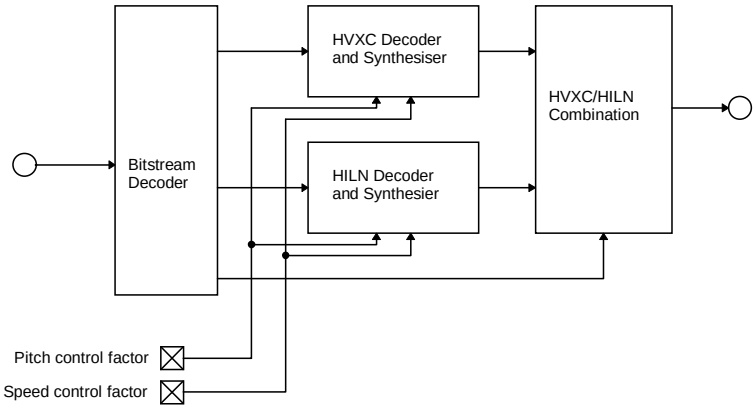


Figure: Block diagram of the parametric decoder core

The integrated parametric coder can operate in the following modes:

PARAmode	Description
0	HVXC only
1	HILN only
2	switched HVXC / HILN
3	mixed HVXC / HILN

PARAmodes 0 and 1 represent the fixed HVXC and HILN modes. PARAmode 2 permits automatic switching between HVXC and HILN depending on the current input signal type. In PARAmode 3 the HVXC and HILN decoders can be used simultaneously and their output signals are added (mixed) in the parametric decoder.

In „switched HVXC / HILN“ and „mixed HVXC / HILN“ modes both HVXC and HILN decoder tools are operated alternatively or simultaneously according to the PARAswitchMode or PARAmixMode of the current frame. To obtain proper time alignment of both HVXC and HILN decoder output signals before they are added, a FIFO buffer compensates for the time difference between HVXC and HILN decoder delay.

To avoid hard transitions at frame boundaries when the HVXC or HILN decoders are switched on or off, the respective decoder output signals fade in and out smoothly. For the HVXC decoder a 20 ms linear fade is applied when it is switched on or off. The HILN decoder requires no additional fading because of the smooth synthesis windows utilized in the HILN synthesizer. It is only necessary to reset the HILN decoder (numLine = 0) if the current bitstream frame contains no „HILNframe()“.

2 Bitstream Syntax

The syntax is described in pseudo-C code.

2.1 Decoder Configuration (Bitstream Header)

Decoder configuration for the parametric decoder core:

Parametric Base Layer -- Configuration

For the parametric core in unscalable mode or as base layer in scalable mode has the following ParametricSpecificConfig() is required:

```
ParametricSpecificConfig() {
        PARAconfig();
}
```

Parametric HVXC Enhancement Layer -- Configuration

The HVXC provides a 2kbps base layer plus 2kbps enhancement layer scalable mode. In this scalable mode the basic layer configuration must be as follows:

PARAmode = 0	HVXConly
HVXCvarMode = 0	HVXC fixed bit rate
HVXCrateMode = 0	HVXC 2 kbps

For the enhancement layer, there is no ParametricSpecificConfig() required:

```
ParametricSpecificConfig() {
}
```

Parametric HILN Enhancement Layer -- Configuration

To use HILN as core in an "T/F scalable with core" mode, additionally to the HILN basic layer an HILN enhancement layer is required. This enhancement layer has the following ParametricSpecificConfig():

```
ParametricSpecificConfig() {
        HILNenhaConfig();
}
```

Syntax	No. of bits	Mnemonic

PARAconfig()
{
PARAmode
if (PARAmode != 1) {
HVXCconfig()
}
if (PARAmode != 0) {
HILNconfig()
}
}
	2	**uimsbf**

Table: PARAmode

PARAmode	frameLength	Description
0	20 ms	HVXC only
1	see Section 2.1.2	HILN only
2	40 ms	HVXC/HILN switching
3	40 ms	HVXC/HILN mixing

2.1.1 HVXC decoder configuration

Syntax	No. of bits	Mnemonic
HVXCconfig()		

{		
HVXCvarMode	**1**	**uimsbf**
HVXCrateMode	**2**	**uimsbf**
}		

Table: HVXCvarMode

HVXCvarMode	Description
0	HVXC fixed bit rate
1	HVXC variable bit rate

Table: HVXCrateMode

HVXCrateMode	HVXCrate	Description
0	2000	HVXC 2 kbit/s
1	4000	HVXC 4 kbit/s
2	3850	HVXC doubleframe 3.85 kbit/s
3	(reserved)	

Table: HVXC Constants

NUM_SUBF1	NUM_SUBF2
2	4

2.1.2 HILN decoder configuration

Syntax	No. of bits	Mnemonic
HILNconfig()		
{		
HILNquantMode	**3**	**uimsbf**
HILNmaxNumLine	**8**	**uimsbf**
HILNframeLengthBase	**1**	**uimsbf**
HILNframeLength	**12**	**uimsbf**
HILNcontMode	**2**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
HILNenhaConfig()		
{		
HILNenhaQuantMode	**2**	**uimsbf**
}		

Table: frameLength for HILN (PARAmode==1)

HILNframeLengthBase	frameLength
0	HILNframeLength * (1/48000) seconds
1	HILNframeLength * (1/44100) seconds

Table: linebits

MaxNumLine	0	1	2..3	4..7	8..15	16..31	32..63	64..127	128..255
linebits	0	1	2	3	4	5	6	7	8

Table: HILNquantMode

HILNquantMode	fmin	fmax	fbits	abits	dfbits	dabits	noisebw
0	30 Hz	4 kHz	10	4	6	4	4 kHz
1	30 Hz	4 kHz	10	6	6	4	4 kHz

2	20 Hz	20 kHz	11	5	6	4	8 kHz
3	20 Hz	20 kHz	11	6	6	4	8 kHz
4	20 Hz	20 kHz	11	5	6	4	20 kHz
5	20 Hz	20 kHz	11	6	6	4	20 kHz
6	(reserved)						
7	(reserved)						

Table: HILNcontMode

HILNcontMode	additional decoder line continuation (see Section 4.4.3.1)
0	harmonic lines <-> individual lines and harmonic lines <-> harmonic lines
1	mode 0 plus individual lines <-> individual lines
2	(reserved)
3	(reserved)

The number of frequency enhancement bits (fEnhabits[i]) in HILNenhaFrame() is calculated as follows:

- individual line: fEnhabits[i] = max(0,fEnhabitsBase[lineFreq[i]]+fEnhabitsMode)
- harmonic line: fEnhabits[i] = max(0,fEnhabitsBase[harmFreq]+fEnhabitsMode+fEnhabitsHarm[i])

For a continued individual line, the lineFreq value of its initial „new line“ is used to calculate fEnhabits[i].

Table: fEnhabitsBase

lineFreq (HILNquantMode: 0,1)	lineFreq (HILNquantMode: 2,3)	harmFreq	fEnhabitsBase
0.. 588	0.. 954	0..1177	0
589.. 733	955..1159	1178..1467	1
734.. 878	1160..1365	1468..1757	2
879..1023	1366..1570	1758..2047	3
	1571..1776		4
	1777..1981		5
	1982..2047		6

Table: fEnhabitsMode

HILNenhaQuantMode	0	1	2	3
fEnhabitsMode	-3	-2	-1	0

Table: fEnhabitsHarm

i	0	1	2..3	4..7	8..9
fEnhabitsHarm[i]	0	1	2	3	4

Table: HILN constants:

amin	amax	dfmax	damax
1	32768	0.15	4

tmbits	atkbits	decbits
4	4	4

tmEnhbits	atkEnhbits	decEnhbits	phasebits
3	2	2	5

2.2 Bitstream frame

Transmission of parametric audio bitstreams in AL-PDUs

Each scaleable layer of an MPEG-4 parametric audio bitstream is transmitted in an Elementary Stream
In an **AL-PDU payload** the following dynamic data for parametric audio has to be included:

Parametric Base Layer -- Access Unit payload

```
alPduPayload {
        PARAframe();
}
```

Parametric HVXC Enhancement Layer -- Access Unit payload

To parse and decode the HVXC enhancement layer, information decoded from the HVXC base layer is required.

```
alPduPayload {
        HVXCenhaFrame();
}
```

Parametric HILN Enhancement Layer -- Access Unit payload

To parse and decode the HILN enhancement layer, information decoded from the HILN base layer is required.

```
alPduPayload {
        HILNenhaFrame();
}
```

Syntax	No. of bits	Mnemonic

PARAframe()
{
if (PARAmode == 0) {
if(HVXCvarMode ==0)
HVXCframe(HVXCrate)
else
HVXCvarframe()
}
else if (PARAmode == 1) {
HILNframe()
}
else if (PARAmode == 2) {
switchFrame()
}
else if (PARAmode == 3) {
mixFrame()
}
}		

Syntax	No. of bits	Mnemonic

switchFrame()
{
PARAswitchMode
if (PARAswitchMode == 0) {
HVXCdoubleframe(HVXCrate)
}
else {
HILNframe()
}
}
	1	**uimsbf**

One of the following PARAswitchModes is selected in each frame:

PARAswitchMode	Description
0	HVXC only
1	HILN only

Syntax	No. of bits	Mnemonic

mixFrame()
{
PARAmixMode
if (PARAmixMode == 0) {
HVXCdoubleframe(HVXCrate)
}
else if (PARAmixMode == 1) {
HVXCdoubleframe(2000)
HILNframe()
}
else if (PARAmixMode == 2) {
HVXCdoubleframe(4000)
HILNframe()
}
else if (PARAmixMode == 3) {
HILNframe()
}
}
	2	**uimsbf**

One of the following PARAmixModes is selected in each frame:

PARAmixMode	Description
0	HVXC only
1	HVXC 2 kbit/s & HILN
2	HVXC 4 kbit/s & HILN
3	HILN only

Syntax	No. of bits	Mnemonic

HVXCdoubleframe(rate)
{
if(rate >=3000){
HVXCframe(4000)
HVXCframe(rate * 2 - 4000)
}
else{
HVXCframe(2000)
HVXCframe(rate * 2-2000)
}
}		

2.2.1 HVXC bitstream frame

HVXCframe

Syntax	No. of bits	Mnemonic

HVXCframe(rate)
{
if(rate >= 2000){
idLsp1()		

```
        idVUV()
        idExc1()
    }
    if(rate >= 3700){
        idLsp2()
        idExc2(rate)
    }
}
```

Syntax	No. of bits	Mnemonic
HVXCenhaFrame()		
{		
idLsp2()		
idExc2(4000)		
}		

Syntax	No. of bits	Mnemonic
idLsp1()		
{		
LSP1	**5**	**uimsbf**
LSP2	**7**	**uimsbf**
LSP3	**5**	**uimsbf**
LSP4	**1**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
idLsp2()		
{		
LSP5	**8**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
idVUV()		
{		
VUV	**2**	**uimsbf**
}		

Table: VUV

VUV	Description
0	Unvoiced Speech
1	Mixed Voiced Speech-1
2	Mixed Voiced Speech-2
3	Voiced Speech

Syntax	No. of bits	Mnemonic
idExc1()		
{		
if(VUV!=0){		
Pitch	**7**	**uimsbf**
SE_shape1	**4**	**uimsbf**
SE_shape2	**4**	**uimsbf**
SE_gain	**5**	**uimsbf**
}		
else{		
for(sf_num=0;sf_num<NUM_SUBF1;sf_num++){		
VX_shape1 [sf_num]	**6**	**uimsbf**
VX_gain1 [sf_num]	**4**	**uimsbf**
}		

```
    }
}
```

Syntax	No. of bits	Mnemonic
idExc2(rate)		
{		
if(VUV!=0){		
SE_shape3	**7**	**uimsbf**
SE_shape4	**10**	**uimsbf**
SE_shape5	**9**	**uimsbf**
if(rate>=4000){		
SE_shape6	**6**	**uimsbf**
}		
}		
else{		
for(sf_num=0;sf_num<NUM_SUBF2-1;sf_num++){		
VX_shape2[sf_num]	**5**	**uimsbf**
VX_gain2[sf_num]	**3**	**uimsbf**
}		
if(rate>=4000){		
VX_shape2[3]	**5**	**uimsbf**
VX_gain2[3]	**3**	**uimsbf**
}		
}		
}		

idLsp1(), idExc1(), idVUV() are treated as base layer in case of scalable mode.
idLsp2(), idExc2() are treated as enhancement layer in case of scalable mode.

HVXCvarframe

Syntax	No. of bits	Mnemonic
HVXCvarframe()		
{		
idvarVUV()		
idvarLsp1()		
idvarExc1()		
}		

Syntax	No. of bits	Mnemonic
idvarVUV()		
{		
VUV	**2**	**uimsbf**
}		

Table: VUV

VUV	Description
0	Unvoiced Speech
1	Background Noise
2	Mixed Voiced Speech
3	Voiced Speech

Syntax	No. of bits	Mnemonic
idvarLsp1()		
{		
if(VUV != 1)		

{		
LSP1	**5**	**uimsbf**
LSP2	**7**	**uimsbf**
LSP3	**5**	**uimsbf**
LSP4	**1**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
idvarExc1()		
{		
if(VUV != 1)		
{		
if(VUV != 0)		
{		
Pitch	**7**	**uimsbf**
SE_Shape1	**4**	**uimsbf**
SE_Shape2	**4**	**uimsbf**
SE_Gain	**5**	**uimsbf**
}		
else		
{		
for(sf_num=0;sf_num<NUM_SUBF1;sf_num++)		
{		
VX_gain1[sf_num]	**4**	**uimsbf**
}		
}		
}		
}		

2.2.2 HILN bitstream frame

Syntax	No. of bits	Mnemonic
HILNframe()		
{		
HILNbasicFrame()		
}		

Syntax	No. of bits	Mnemonic
HILNbasicFrame()		
{		
prevNumLine = numLine		
/* prevNumLine is the number of lines in the previous frame */		
/* prevNumLine = 0 for the first bitstream frame */		
numLine	**linebits**	**uimsbf**
envFlag	**1**	**uimsbf**
if (envFlag) {		
envTmax	**tmbits**	**uimsbf**
envRatk	**aktbits**	**uimsbf**
envRdec	**decbits**	**uimsbf**
}		
for (k=0; k<prevNumLine; k++) {		
prevLineContFlag[k]	**1**	**uimsbf**
}		
i = 0		
for (k=0; k<prevNumLine; k++)		

if (prevLineContFlag[k])		
lineContFlag[i++] = 1		
while (i<numLine)		
lineContFlag[i++] = 0		
harmFlag	**1**	**uimsbf**
if (harmFlag) {		
HARMbasicPara()		
}		
noiseFlag	**1**	**uimsbf**
if (noiseFlag) {		
NOISEbasicPara()		
}		
INDIbasicPara()		
}		

Syntax	No. of bits	Mnemonic
HARMbasicPara()		
{		
numHarmTransCod	**6**	**uimsbf**
addHarmAmplBits	**1**	**uimsbf**
harmPred	**1**	**uimsbf**
harmEnv	**1**	**uimsbf**
harmFreq	**11**	**uimsbf**
harmFreqStretch	**5**	**uimsbf**
harmTransAmpl[0]	**6**	**uimsbf**
numHarmTrans = numHarmTransCod + 1		
habits = 4 + (addHarmAmplBits ? 1 : 0)		
for (i=1; i<numHarmTrans; i++) {		
harmTransAmpl[i]	**habits**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
NOISEbasicPara()		
{		
numNoiseParaCod	**2**	**uimsbf**
noiseEnvFlag	**1**	**uimsbf**
noiseNorm	**6**	**uimsbf**
numNoisePara = numNoiseParaCod + 4		
for (i=0; i<numNoisePara; i++) {		
noisePara[i]	**3**	**uimsbf**
}		
if (noiseEnvFlag) {		
noiseEnvTmax	**tmbits**	**uimsbf**
noiseEnvRatk	**aktbits**	**uimsbf**
noiseEnvRdec	**decbits**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
INDIbasicPara()		
{		
for (i=0; i<numLine; i++) {		
if (envFlag) {		
lineEnvFlag[i]	**1**	**uimsbf**
}		

```
        if (!lineContFlag[i]) {
            lineAmpl[i]                    abits       uimsbf
            lineFreq[i]                    fbits       uimsbf
        }
        else {
            lineAmplDelta[i]               dabits      uimsbf
            lineFreqDelta[i]               dfbits      uimsbf
        }
    }
}
```

Syntax	No. of bits	Mnemonic
HILNenhaFrame()		
{		
if (envFlag) {		
envTmaxEnha	**tmEnhbits**	**uimsbf**
envRatkEnha	**atkEnhbits**	**uimsbf**
envRdecEnha	**decEnhbits**	**uimsbf**
}		
if (harmFlag) {		
HARMenhaPara()		
}		
INDIenhaPara()		
}		

Syntax	No. of bits	Mnemonic
HARMenhaPara()		
{		
for (i=0; i<min(numHarmTrans,10); i++) {		
harmFreqEnha[i]	**fEnhbits[i]**	**uimsbf**
harmPhase[i]	**phasebits**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
INDIenhaPara()		
{		
for (i=0; i<numLine; i++) {		
lineFreqEnha[i]	**fEnhbits[i]**	**uimsbf**
linePhase[i]	**phasebits**	**uimsbf**
}		
}		

3 HVXC decoder tools

3.1 Overview

An efficient coding scheme for Linear Predictive Coding (LPC) residuals is used based on harmonic and stochastic vector representation. Vector quantization (VQ) of the spectral envelope of LPC residuals with a weighted distortion measure is used when the signal is voiced. Vector excitation coding (VXC) is used when the signal is unvoiced. This algorithm is called Harmonic Vector Excitation Coding (HVXC). The major algorithmic features are:

- Weighted VQ of variable dimension spectral vector.

- A fast harmonic synthesis algorithm by IFFT.
- Interpolative coder parameters for speed/pitch control

Also, functional features include:

- As low as 33.5 ms of total algorithmic delay is supported
- 2.0-4.0 kbps scalable mode is supported
- Variable bit rate coding for rates less than 2.0 kbps is supported

3.1.1 Framing structure and Block diagram of the decoder

Fig. 3.1.1 shows the overall structure of the HVXC decoder. The HVXC decoder tools allow decoding of speech signals at 2 kbit/s and higher, up to 4 kbit/s. HVXC decoder tools also allow decoding with variable bit rate mode at an average bit rate of around 1.2-1.7 kbps. The basic decoding process is composed of four steps; de-quantization of parameters, generation of excitation signals for voiced frames by sinusoidal synthesis (harmonic synthesis) and noise component addition, generation of excitation signals for unvoiced frames by codebook look-up, and LPC synthesis. Spectral post-filtering is also used to enhance the synthesized speech quality.

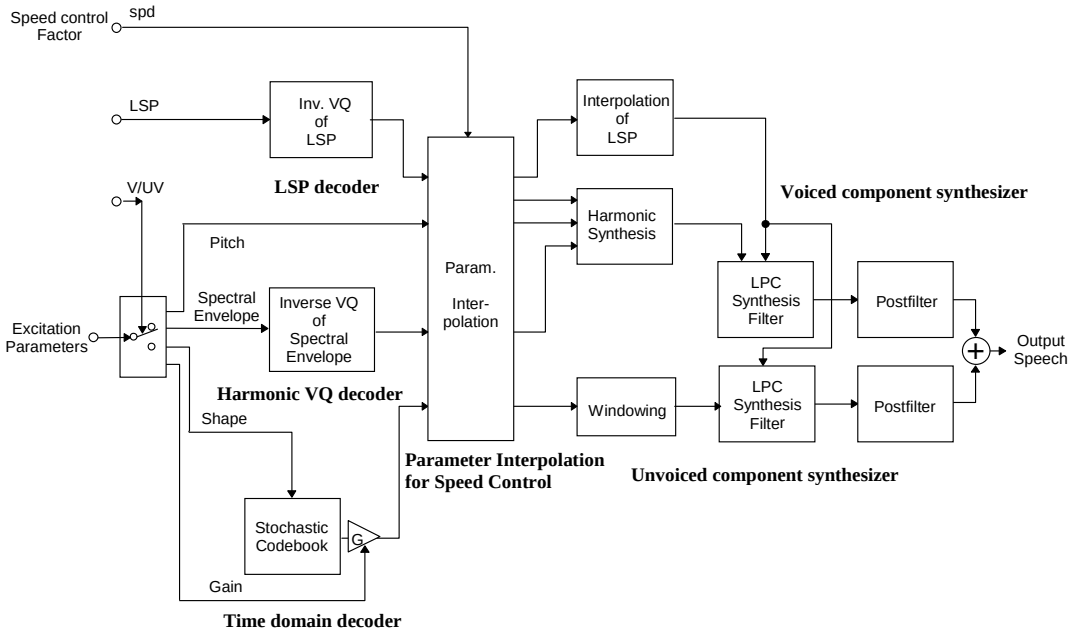


Fig.3.1.1 Blockdiagram of the HVXC decoder

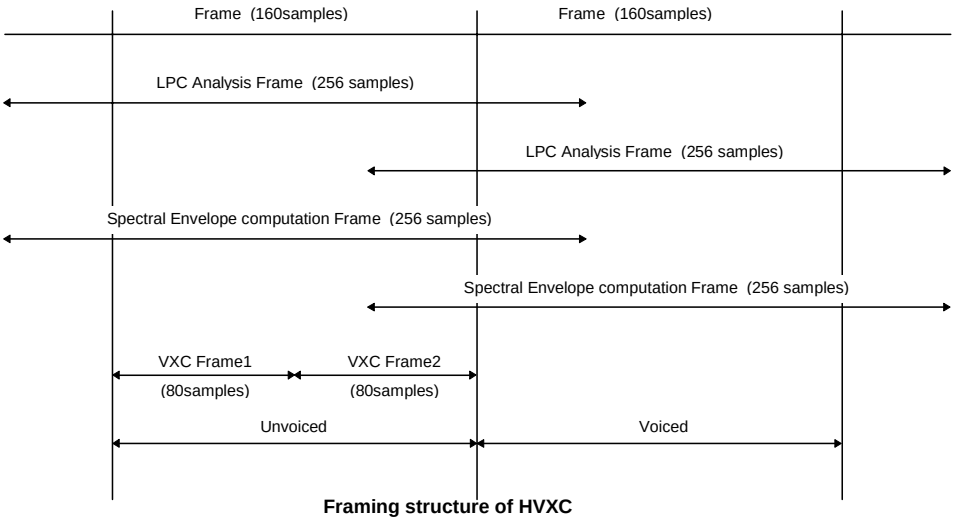
3.1.2 Delay Mode

HVXC coder/decoder supports low/normal delay encode/decode mode, allowing any combinations of delay mode at 2.0-4.0 kbps with scalable scheme. Figure below shows the framing structure of each delay mode. Frame length is 20 ms for all the delay modes. For example, use of low delay encode and low delay decode mode allows total coder/decoder delay of 33.5 ms.

In the encoder, algorithm delay could be selected to be either 26 ms or 46 ms. When 46 ms delay is selected, one frame look ahead is used for pitch detection. When 26 ms delay is selected, only current frame is used for pitch detection. For both cases, syntax is common, all the quantizers are common, and bitstreams are compatible. In the decoder , algorithm delay could be selected to be either 10 ms or 7.5 ms. When 7.5 ms delay is selected, the decoder frame interval is shifted by 2.5 ms compared with 10 ms delay mode. In this case, excitation generation

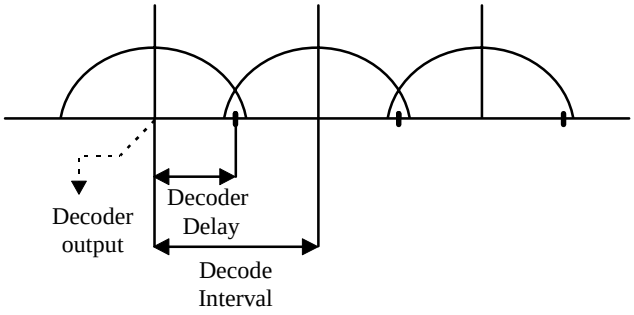
and LPC synthesis phase is shifted by 2.5 ms. Again, for both cases, syntax is common, all the quantizers are common, and bitstreams are compatible.

In summary, any independent choice of encoder/decoder delay from the following combination is possible:
Encoder delay: 26 ms or 46 ms
Decoder delay: 10 ms or 7.5 ms

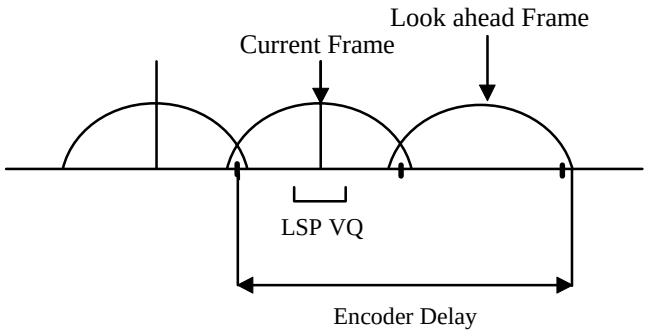


Framing structure of HVXC

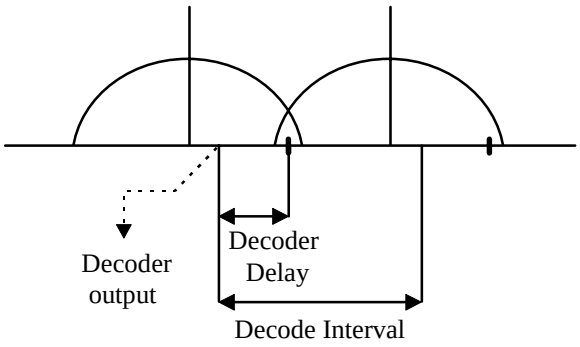
Normal Delay Decoder



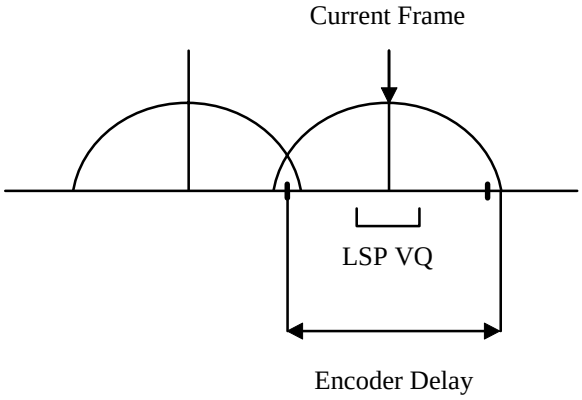
Normal Delay Encoder



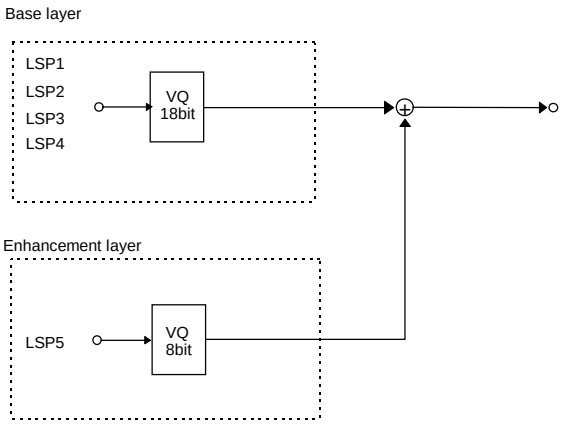
Low Delay Decoder



Low Delay Encoder



3.2 LSP decoder



3.2.1 Tool description

As for LSP parameters, a multistage quantizer structure is used and the output vectors from each stage have to be summed up to obtain the LSP parameters.

When the bitrate is 2 kbps, the LSPs of the current frame, which are coded by split and two-stage vector quantization, are decoded with two-stage decoding processes. In the case of 4 kbps, a 10-dimensional vector quantizer, which has an 8 bit codebook, is added to the bottom of the LSP quantizer scheme of 2.0 kbps coder. The bitrate of LSPs is increased from 18bits/20msec to 26bits/20msec.

LSP parameters are then linearly interpolated to be updated every 2.5 ms for voiced segments. Interpolated LSPs are converted to direct form of linear predictive coefficients α_n.

1st stage	10 LSP VQ	5 bits
2nd stage	(5+5) LSP VQ	(7+5+1) bits
3rd stage	10 LSP VQ	8 bits

3.2.2 Definitions

Definition of constants

LPCORDER : LPC analysis order (=10)

L_VQ : number of LSP_VQ indices (=4 (2 kbps) =5 (4 kbps))

Definition of variables

qLsp[i] : quantized LSP parameters

idLSP : LSP indices packed by following IdLsp structure

```
typedef struct
{
   int                  nVq1[L_VQ],
}
IdLsp;
```

nVq1[i] : LSP VQ indices

lsp_tbl[][][]	look-up tables for the first stage decoding process
d_tbl[][][]	look-up tables for the second stage decoding process of the VQ without interframe prediction
pd_tbl[][][]	look-up tables for the second stage decoding process of the VQ with interframe prediction.
vqLsp[][]	look-up table for the enhancement layer
sign	sign of code vector for the second stage decoding process
idx	unpacked index for the second stage decoding process
lsp_predict[]	the LSPs predicted from *lsp_previous[]* and *lsp_first[]*
lsp_previous[]	the LSPs decoded at the previous frame
lsp_current[]	the LSPs decoded at the current frame
lsp_first[]	the LSPs decoded at the first stage decoding process
ratio_predict	prediction ratio for predicting *lsp_predict[]*
ratio_sub	interpolation ratio for calculating *lsp_subframe[][]*
dim[][]	dimensions for the split vector quantization

3.2.3 Decoding process

The decoding process of the LSP parameters for the base layer (2.0 kbps) is the same as that of the narrowband CELP. The decoding process is as described below.

Converting indices to LSPs

The LSPs of the current frame (*lsp_current[]*), which are coded by split and two-stage vector quantization, are decoded with a two-stage decoding process. The dimension of each vector is described in the tables below. The **nVq1[0]** and **nVq1[1], nVq1[2]** represent indices for the first and the second stage respectively.

Dimension of the first stage LSP vector

Split Vector Index: i	Vector Dimension: dim[0][i]
0	10

Dimension of the second stage LSP vector

Split Vector Index: i	Vector Dimension: dim[1][i]
0	5
1	5

In the first stage, the LSP vector of the first stage *lsp_first[]* is decoded simply by looking up the table *lsp_tbl[][][]* . (The table *lsp_tbl[][][]* is shown in Annex C)

```
for(i=0;i<dim[0][0];i++){
   lsp_first[i] = lsp_tbl[0][nVq1[0]][i];
}
```

In the second stage, there are two types of decoding processes, namely, decoding process of VQ without interframe prediction and VQ with interframe prediction. The **nVq1[3]** indicates which process should be selected.

Decoding process for the second stage

LPC Index: nVq1[3]	Decoding process
0	VQ without interframe prediction
1	VQ with interframe prediction

Decoding process of VQ without interframe prediction

In order to obtain LSPs of the current frame *lsp_current[]* , the decoded vectors in the second stage are added to the decoded first stage LSP vector *lsp_first[]*. The MSB of the **nVq1[]** represents the sign of the decoded vector, and the remaining bits represent the index for the table *d_tbl[][][]*. (The table *d_tbl[][][]* is shown in Annex C)

```
sign = nVq1[1]>>6;
idx = nVq1[1]&0x3f;
if(sign==0) {
   for(i=0;i<dim[1][0];i++){
      lsp_current[i] = lsp_first[i] + d_tbl[0][idx][i];
   }
}else {
    for(i=0;i<dim[1][0];i++) {
      lsp_current[i] = lsp_first[i] - d_tbl[0][idx][i];
   }
}
sign = nVq1[2]>>4;
idx = nVq1[2]&0x0f;
if(sign==0) {
   for(i=0;i<dim[1][1];i++){
      lsp_current[dim[1][0]+i] = lsp_first[dim[1][0]+i] + d_tbl[1][idx][i];
   }
}else {
    for(i=0;i<dim[1][1];i++) {
      lsp_current[dim[1][0]+i] = lsp_first[dim[1][0]+i] - d_tbl[1][idx][i];
   }
}
```

Decoding process of VQ with interframe prediction

In order to obtain LSPs of the current frame *lsp_current[]* , the decoded vectors of the second stage are added to the LSP vector *lsp_predict[]* which are predicted from the decoded LSPs of the previous frame *lsp_previous[]* and the decoded first stage LSP vector *lsp_first[]*. As with the decoding process of VQ without interframe prediction, the MSB of the **nVq1[]** represents the sign of the decoded vector, and the remaining bits represent the index for the table *pd_tbl[][][]*. (The table *pd_tbl[][][]* is shown in Annex C)

```
for(i=0;i<LPCORDER;i++){
   lsp_predict[i]=(1-ratio_predict)*lsp_first[i]
                     + ratio_predict*lsp_previous[i]
}
```

where *ratio_predict* = 0.7

```
sign = nVq1[1]>>6;
idx = nVq1[1]&0x3f;
if(sign==0) {
   for(i=0;i<dim[1][0];i++){
      lsp_current[i] = lsp_predict[i] + pd_tbl[0][idx][i];
   }
}else {
    for(i=0;i<dim[1][0];i++) {
      lsp_current[i] = lsp_predict[i] - pd_tbl[0][idx][i];
   }
}
sign = nVq1[2]>>4;
idx = nVq1[2]&0x0f;
if(sign==0) {
   for(i=0;i<dim[1][1];i++){
      lsp_current[dim[1][0]+i]
          = lsp_predict[dim[1][0]+i] + pd_tbl[1][idx][i];
   }
}else {
    for(i=0;i<dim[1][1];i++) {
      lsp_current[dim[1][0]+i]
          = lsp_predict[dim[1][0]+i] - pd_tbl[1][idx][i];
   }
}
```

Stabilization of LSPs

The decoded LSPs *lsp_current[]* are stabilized in order to ensure stability of the LPC synthesis filter which is derived from the decoded LSPs. The decoded LSPs are arranged in ascending order, having a minimum distance between adjacent coefficients.

```
for(i=0;i<LPCORDER;i++) {
   if(lsp_current[i] < min_gap) lsp_current[i] = min_gap;
}
for(i=0;i<LPCORDER-1;i++) {
   if(lsp_current[i+1]-lsp_current[i] < min_gap) {
      lsp_current[i+1] = lsp_current[i]+min_gap;
   }
}
for(i=0;i<LPCORDER;i++) {
   if(lsp_current[i] > 1-min_gap) lsp_current[i] = 1-min_gap;
}
for(i=LPCORDER-1;i>0;i--) {
   if(lsp_current[i]-lsp_current[i-1] < min_gap) {
      lsp_current[i-1] = lsp_current[i]-min_gap;
   }
}

for(i=0;i<LPCORDER;i++){
   qLsp[i] = lsp_current[i];
}
```

where min_gap = 4.0/256.0

Storing the coefficients

After the LSP decoding process, the decoded LSPs have to be stored in memory, since they are used for prediction at the next frame.

```
for (i=0;i<LPCORDER;i++) {
     lsp_previous[i] = lsp_current[i];
}
```

It must be noted that the stored LSPs *lsp_previous[]* must be initialized as described below when the whole decoder is initialized.

```
for (i=0;i<LPCORDER;i++) {
     lsp_previous[i] = (i+1) / (LPCORDER+1);
}
```

Decoding process for the enhancement layer

For the enhancement layer (4.0 kbps), additional code vectors and the base layer's LSPs are summed up as follows.

```
for(i = 0; i < LPCORDER; i++)
{
    qLsp[i] += vqLsp[nVq1[4]][i];
}
```

After the calculation, the LSPs are stabilized again.

```
for(i = 0; i < 2; i++)
{
   if(qLsp[i + 1] - qLsp[i] < 0)
   {
       tmp = qLsp[i + 1];
       qLsp[i + 1] = qLsp[i];
       qLsp[i] = tmp;
   }

   if(qLsp[i + 1] - qLsp[i] < THRSLD_L)
   {
       qLsp[i + 1] = qLsp[i] + THRSLD_L;
   }
}


for(i = 2; i < 6; i++)
{
   if(qLsp[i + 1] - qLsp[i] < THRSLD_M)
   {
       tmp = (qLsp[i + 1] + qLsp[i]) / 2.0;
       qLsp[i + 1] = tmp + THRSLD_M / 2.0;
       qLsp[i] = tmp - THRSLD_M / 2.0;
   }
}


for(i = 6; i < LPCORDER - 1; i++)
{
   if(qLsp[i + 1] - qLsp[i] < 0)
   {
       tmp = qLsp[i + 1];
       qLsp[i + 1] = qLsp[i];
       qLsp[i] = tmp;
   }
```

```
   if(qLsp[i + 1] - qLsp[i] < THRSLD_H)
   {
       qLsp[i] = qLsp[i + 1] - THRSLD_H;
   }
}
```

where, THRSLD_L=0.020, THRSLD_M=0.020 and THRSLD_H=0.020

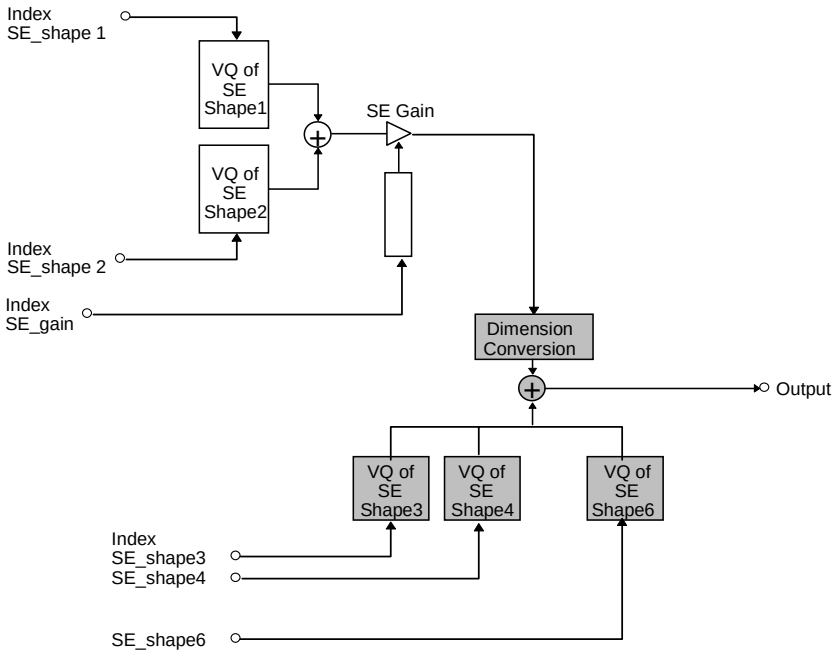
3.2.4 Tables

See LSP quantizer table of Annex C

3.3 Harmonic VQ decoder

3.3.1 Tool description

The decoding process consists of two steps in the case of 2 kbps, that is, inverse vector quantization of residual vectors and dimension conversion. In the case of 4 kbps, additional inverse-quantizers are used . The operations of each step are given below.



Vector Quantization of Spectral envelope

3.3.2 Definitions

vqdim0: vector dimension of harmonic spectral quantization (=44)
qedvec[ ]: quantized harmonic spectral vector in the fixed dimension
g0[ ]: SE_gain codeword
cb0[ ]: SE_shape1 codeword
cb1[ ]: SE_shape2 codeword
idx_g: SE_gain index
idx_s0: SE_shape1 index

idx_s1: SE_shape2 index
send : number of harmonics in the current frame
send1 : fixed number of harmonics = 44
OVER_R: over sampling rate in the 1st step of dimension conversion (=8)
w0f: w01/R . w01 is original fundamental frequency where SAMPLE*R represents 2*π (R = 8, SAMPLE=256).
w0: target fundamental frequency after the dimension conversion where SAMPLE*R represents 2*π
JISU: order of 1st over sampling filter in up sampled domain
re[ ]: input of the 1st step of the dimension conversion
rel[ ]: output of the 1st step of the dimension conversion (also input of the 2nd step)
f_coef[ ]: over sampling filter coefficients expressed as $coef[i]$
inint(x): nearest integer of x
am2deci[ ]: output of the 2nd step of the dimension conversion

schm4k->num_vq: number of vector quantizers for additional stage for 4 kbps (=4)
schm4k->vqdim[j]: dimension of jth vector quantizer at the additional stage (=2,4,4,4)
id4k[j]: received index of jth vector quantizer at the additional stage
send2: number of harmonics of current frame
schm4k->dim_tot: sum of dimensions of vector quantizers at the additional stage (=14)
target[i]: reconstructed vector due to the vector quantizers at the additional stage
qam2[i]: reconstructed harmonic spectral vector for 2 kbps mode
qam4[i]: reconstructed harmonic spectral vector for 4 kbps mode
cb4k[0][idx][l]: l-th component of the idx-th codeword of SE_shape3
cb4k[1][idx][l]: l-th component of the idx-th codeword of SE_shape4
cb4k[2][idx][l]: l-th component of the idx-th codeword of SE_shape5
cb4k[3][idx][l]: l-th component of the idx-th codeword of SE_shape6

3.3.3 Decoding process

For the quantization of harmonic spectral magnitudes, 2.0 kbps decoder uses a combination of two-stage shape vector quantizer and a scalar gain quantizer, where each of the shape codebooks is 4 bits and gain codebook is 5 bits respectively. The dimension of the shape codebooks is fixed (=44). Inverse vector quantization is first carried out to obtain the fixed dimension spectral vector. When the bit-rate is 2 kbps, the first two-stage VQ and gain quantizer are used (SE_shap1, SE_shape2, and SE_gain). The two shape vectors are added and multiplied by gain. In order to obtain the spectral vector of original dimension, dimension conversion is then applied to the fixed dimension spectral vector. For 4.0 kbps mode, the de-quantized harmonic magnitudes with fixed dimension (=44) are first converted to the dimension of original harmonic vector, which varies frame by frame. The output of additional stage with a split VQ scheme composed of four vector quantizers (SE_Shape3, SE_Shape4, SE_Shape5, SE_Shape6) is added to the dimension converted quantizer output of 2.0 kbps scheme.

2.0 kbps two-stage VQ	4bit shape + 4bits shape + 5bits gain			
dimension	44			
4.0 kbps split VQ	7bits	10bits	9bits	6bits
dimension	2	4	4	4

How dimension conversion is carried out is explained below. The number of points which composes the spectral envelope varies depending on pitch values, since the spectral envelope is a set of the estimates of the magnitudes at each harmonic. The number of harmonics ranges from about 8 to 60. In order to obtain the variable number of harmonic magnitudes, the decoder has to convert a fixed-dimension codevector to a variable dimension vector. The number of points, which represent the shape of the spectral envelope, should be modified without changing the shape. For this purpose, a dimension converter for a spectral envelope by a combination of low pass filter and 1st order linear interpolator is used. An FIR low pass filter with 7 sets of coefficients, each set consisting of 8

coefficients, is used for the first stage 8-times over-sampling. The 7 sets of the filter coefficients are obtained by grouping 8 every coefficients from a windowed sinc, $coef[i]$, with the offsets of 1 through 7, where

$$coef[i] = \frac{\sin \pi(i-32)/8}{\pi(i-32)/8}(0.5 - 0.5\cos 2\pi i / 64) \qquad 0 \le i \le 64$$

This FIR filtering allows decimated computation, in which only the points used at the next stage are computed. They are the left and right adjacent points of the final output of the dimension converter.
At the second over-sampling stage, 1st order linear interpolation is applied to obtain the necessary output points. In this way, variable-dimension spectral vectors from fixed-dimension (= 44) spectral vectors are obtained. Pitch modification can be done just by modifying the target fundamental frequency w0 and number of harmonics correspondingly according to a pitch modification factor *pch_mod* before the dimension conversion. The modified target fundamental frequency for pitch change is computed as ; w0 * *pch_mod* .

Inverse vector quantization(2 kbps):

```
for(l=0;l<vqdim0;l++)
   re[l]=g0[idx_g]*(cb0[idx_s0][l]+cb1[idx_s1][l]);
```

1st step of dimension conversion:

```
for(i=0; i<=send1;i++){
   for(p=0;p<OVER_R;p++){
      if((i*OVER_R+p+1)*w0f > w0*ii){
         rel[i*OVER_R+p]=rel[i*OVER_R+p+1]=0.;
         for(j=1;j<JISU;j++){
            if(i+j-(JISU-1)/2 >= 0){
               rel[i*OVER_R+p] += f_coef[j * OVER_R - p] * re[i+j-(JISU-1)/2];
               rel[i*OVER_R+p+1] += f_coef[j * OVER_R - (p+1)] * re[i+j-(JISU-1)/2];
            }
         }
         ii++;
      }
   }
}
```

2nd step of dimension conversion:

```
ii=0;
for(i=0;i<=(send1+1)*R-1;i++){
   if(i==0) lb=0;
   else  lb=ub;
   ub=inint((float)(i+1) * w0f);
   bw=ub-lb;
   idx=0;
   for(j=lb;j<ub;j++){
      if(j == inint((float)ii * w0)){
         am2deci[j]=rel[i]*(1.-(float)idx/(float)bw)+rel[i+1]*((float)idx/(float)bw);
         ii++;
      }
      idx++;
      if(ii > send ) break;
   }
   if(ii > send ) break;
}
```

Inverse vector quantization(4 kbps):

```
target[0]=0.;
k=1;
for(j=0;j<schm4k->num_vq;j++){
   idx=id4k[j];
   for(l=0;l< schm4k->vqdim[j];l++){
      target[k]=cb4k[j][idx][l];
      k++;
```

```
   }
}
if(schm4k->dim_tot < send2){
   for(i=(schm4k->dim_tot)+1;i<=send2;i++)
      target[i]=0.;
}

for(i=0;i<=send2;i++)
   qam4[i]=qam2[i]+target[i];
```

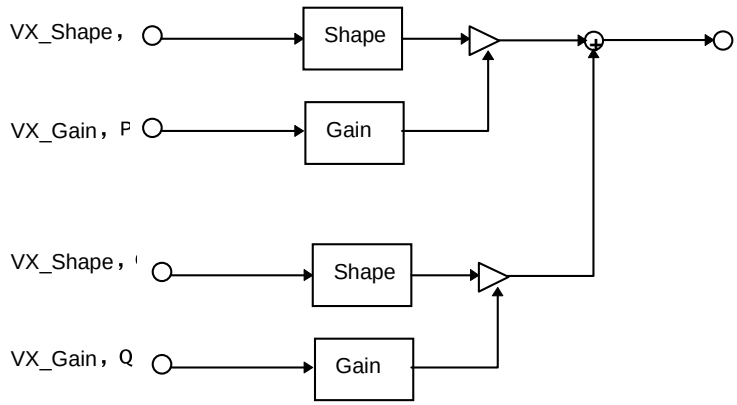
3.3.4 Tables

See harmonic VQ table of Annex C

3.4 Time domain Decoder

3.4.1 Tool description

For unvoiced segments of the speech, a scheme that is the same as VXC (Vector Excitation Coding) is used. The time domain decoder generates an excitation waveform for unvoiced portion by table look up using transmitted indices. The shape vector and gain are updated every 10 ms and they are multiplied. In the case of 2 kbps, only the output of the first stage is used. In the case of 4 kbps, the shape output of the second stage is multiplied by the gain output of the second gain quantizer and added to the excitation waveform of the first stage. The shape and gain of the second stage are updated every 5 ms.



1st stage	80dimension 6bits shape + 4bits gain
2nd stage	(40dimension 5bits shape + 3bits gain) x 2

3.4.2 Definitions

Definition of constant

DimShape : VXC frame length(=80)

DimShape2 : 2nd stage VXC frame length(=40)

Definition of variables

res[i] : i-th component of VXC decoder output

cbL0_g[i] : i-th entry of VXC gain codebook

cbL0_s[i][j] : j-th component of i-th entry of VXC shape codebook

cbL1_g[i] : i-th entry of 2nd stage VXC gain codebook

cbL1_s[i][j] : j-th component of i-th entry of 2nd stage VXC shape codebook

idCelp : VXC indices packed by following IdCelp structure
typedef struct
{
  int idSL0[2], idGL0[2];
  int idSL1[4], idGL1[4];
}IdCelp;
idSL0[i] : 1st stage VXC shape index for i-th sub-frame
idGL0[i] : 1st stage VXC gain index for i-th sub-frame
idSL1[i] : 2nd stage VXC shape index for i-th sub-frame
idGL1[i] : 2nd stage VXC gain index for i-th sub-frame

3.4.3 Decoding process

```
for(i = 0; i < DimShape; i++)
   res[i] = cbL0_g[idCelp->idGL0[0]] * cbL0_s[idCelp->idSL0[0]][i];
for(i = 0; i < DimShape; i++)
   res[i + DimShape] = cbL0_g[idCelp->idGL0[1]] *cbL0_s[idCelp->idSL0[1]][i];
```

When the bit rate is 4 kbps :

```
for(i = 0; i < 4; i++){
   for(j = 0; j < DimShape2; j++){
      res[j+DimShape2*i] += cbL1_g[idCelp->idGL1[i]]*cbL1_s[idCelp->idSL1[i]][j];
   }
}
```

3.4.4 Tables

See stochastic codebook table of Annex C

3.5 Parameter Interpolation for Speed Control

3.5.1 Tool description

The scheme for Speed Control is explained in this chapter. The decoder has a scheme for parameter interpolation to generate the time domain and harmonic synthesizer input parameters at any arbitrary time instant. By this scheme, a sequence of parameters in modified intervals are computed and applied to the source decoder. In this way, decoder output in a modified time scale is obtained. Since parameters used are easily interpolated, and thus additional complexity for the time scale modification is very small.

3.5.2 Definitions

"Parameter Interpolation" block computes the parameters in the modified time scale by interpolating the received parameters. The operation of this block is described below.
The operation of this block is basically linear interpolation and replacement of parameters.
Let us denote the arrays of original parameters as:

Pitch: $pch[n]$

V / UV: $vuv[n]$

LSP: $lsp[n][k]$ $\cdots$ $0 \le k < K$ K: LSP order

Spectral magnitude: $am[n][l]$ $\cdots$ $0 \le l < L$ L: number of harmonics

VXC excitation: $vex[n][j]$ $\cdots$ $0 \le j < J$ J: frame interval (= 160)

and interpolated parameters as:

Pitch: $mdf\_pch[m]$

V / UV: $mdf\_vuv[m]$

LSP: $mdf\_lsp[m][k]$ ⋯ $0 \le k < K$ K: LSP order

Spectral magnitude: $mdf\_am[m][l]$ ⋯ $0 \le l < L$ L: number of harmonics

VXC excitation: $mdf\_vex[m][j]$ ⋯ $0 \le j < J$ J: frame interval (= 160)

where n and m are the time indices (frame number) before and after the time scale modification. The frame intervals are both 20 ms.

3.5.3 Speed control process

Let us define the ratio of speed change as spd :

$$spd = N_1 / N_2 \tag{3.5.1}$$

where N_1 is the duration of the original speech and N_2 is the duration of the speed controlled speech. Therefore,

$0 \le n < N_1$ and $0 \le m < N_2$.

Basically, the time scale modified parameters are expressed as:

$$mdf\_param[m] = param[m \times spd] \tag{3.5.2}$$

where $param$ are: pch, vuv, lsp and am. In general, however, $m \times spd$ is not an integer number. We therefore define:

$$\begin{cases} fr_0 = \lceil m \times spd \rceil - 1 \\ fr_1 = fr_0 + 1 \end{cases} \tag{3.5.3}$$

to generate parameters at a time index $m \times spd$ by linearly interpolating the parameters at time indices fr_0 and fr_1.

In order to execute linear interpolation, let us define:

$$\begin{cases} left = m \times spd - fr_0 \\ right = fr_1 - m \times spd \end{cases} \tag{3.5.4}$$

Then Eq.(3.5.2) can be approximated as:

$$mdf\_param[m] = param[fr_0] \times right + param[fr_1] \times left \tag{3.5.5}$$

where $param$ are: pch, vuv, lsp and am.

For $lsp[n][i]$ and $am[n][l]$, this linear interpolation is applied with the indices i and l being fixed.

As for the parameter vex :

$vex[n][j]$ has excitation signals for UV frames by codebook look-up.

J samples from $vex[n][j]$ centering around the time $m \times spd$ are taken and the energy over the J samples are computed. Gaussian noise consisting of J samples is then generated and its norm is adjusted so that its energy be equal to that of J samples taken from $vex[n][j]$. This gain adjusted Gaussian noise sequence is used for $mdf\_vex[m][j]$.

Principal operation of time scale modification can be expressed by the Eq.(3.5.5), however, the V/UV decisions at fr_0 and $fr1$, prior to the interpolation, have to be considered.

The interpolation and replacement strategies adapted to V/UV decisions are described below. In the explanation below, full voiced and mixed voiced (V/UV !=0) are grouped as Voiced, and only V/UV ==0 case is regarded as Unvoiced. In the case of variable rate coding, background noise mode (V/UV=1) is also treated as Unvoiced.

- When V/UV decisions at fr_0 and fr_1 are Voiced - Voiced:

  New V/UV $mdf\_vuv[m]$ is obtained as follows;
  $tmp\_vuv = vuv[fr_0] \times right + vuv[fr_1] \times left$
  if ($tmp\_vuv > 2$)
  $mdf\_vuv[m] = 3$
  else if ($tmp\_vuv > 1$)
  $mdf\_vuv[m] = 2$
  else if ($tmp\_vuv > 0$)
  $mdf\_vuv[m] = 1$

  New pitch $mdf\_pch[m]$ is obtained as follows;
  if($0.57 < pch[fr_0] / pch[fr_1]$ && $pch[fr_0] / pch[fr_1] < 1.75$)
  $mdf\_pch[m] = pch[fr_0] \times right + pch[fr_1] \times left$
  else
  if($left < right$)
  $mdf\_pch[m] = pch[fr_0]$
  else
  $mdf\_pch[m] = pch[fr_1]$

  All the other parameters are interpolated by Eq.(3.5.5).

- When V/UV decisions at fr_0 and fr_1 are Unvoiced - Unvoiced:
  All the parameters are interpolated by Eq.(3.5.5) except for mdf_vex. $mdf\_vex[m][j]$ is generated by Gaussian noise having the same energy as that of J samples taken from $vex[n][j]$ centering around the time $m \times spd$.

- When V/UV decisions at fr_0 and fr_1 are Voiced - Unvoiced:
  If $left < right$
  All the parameters at time fr_0 are used instead of computing the parameters at $m \times spd$.
  If $left \geq right$
  All the parameters at time fr_1 are used instead of computing the parameters at $m \times spd$.
  $mdf\_vex[m][j]$ is also generated by Gaussian noise having the same energy as that of J samples from $vex[fr_1][j]$. $(0 \leq j < J)$

- When V/UV decisions at fr_0 and fr_1 are Unvoiced - Voiced:
  If $left < right$
  All the parameters at time fr_0 are used instead of computing the parameters at $m \times spd$.
  $mdf\_vex[m][j]$ is also generated by Gaussian noise having the same energy as that of J samples from $vex[fr0][j]$. $(0 \leq j < J)$
  If $left \geq right$
  All the parameters at time fr_1 are used instead of computing the parameters at $m \times spd$.

In this manner, one obtain all the necessary parameters for the HVXC decoder. Just by applying these modified parameters, $mdf\_param[m]$, to the source decoder in the same way as usual (normal) decoding process, one obtain the time scale modified output.
Apparently, when $N_2 < N_1$, speed up decoding is executed, and when $N_2 > N_1$ speed down decoding is executed. Power spectrum and pitch are not affected by this speed control, thus we can obtain good quality speech for the speed control factor of about $0.5 < spd < 2.0$.

3.6 Voiced Component Synthesizer

3.6.1 Tool description

The voiced component synthesizer consists of these steps ; harmonic excitation synthesis, noise component addition, LPC synthesis, and postfilter. An efficient harmonic excitation synthesis method is first used to obtain periodic excitation waveform from harmonic magnitude envelope. Noise component is added to the periodic waveform and voiced excitation signal is obtained, which is then fed into LPC synthesis filter and postfilter to generate voiced speech signal. The configuration of the postfilter is not normative and described in Annex A.

3.6.2 Definitions

$send$: *the number of harmonics between* 0 *and* 4000 *Hz*

$Am[i]$: *the harmonic magnitude obtained as* $qam2[i](2kbps)$ *or* $qam4[i](4kbps)$ *in the* sec *tion* 3.3.3. $(1 \le i \le send)$

$Am\_noise[i]$: *the noise magnitudes* $(1 \le i \le send)$

$Am\_h[i]$: *the modified harmonic magnitudes* $(1 \le i \le send)$

3.6.3 Synthesis process

Harmonic excitation is generated as shown below. The generation algorithm can generate harmonic excitation for both normal delay mode and low delay mode. In the description below, ipc_decDelayMode == DM_SHORT means low delay decode is selected; otherwise normal delay mode is selected.

For low delay mode, constant LD_LEN is used. The parameter LD_LEN means decode interval shifting, and this case it is set to 20. The waveform synthesized by this function covers from N = -160+LD_LEN [sample] to N = 0+LD_LEN[sample]. N=0 represents the center of the current frame. If this is set to 0, the frame shifting is 0, and the function generates identical results as that of the normal delay mode. This is shown in the figure below. During the period from N=0 to N=LD_LEN, pitch value, harmonic magnitude, and LPC parameters are not interpolated and hold. If LD_LEN=0, decoder delay is 10 ms and if LD_LEN=20, decoder delay is 7.5 ms.

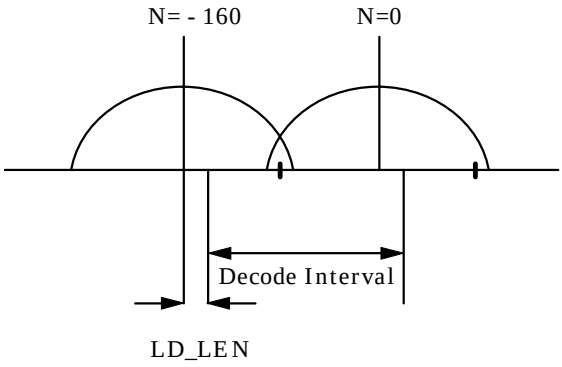


Fig. 3.6.1

Excitation generation:

First excitation generation algorithm is shown followed by excitation generation process. Voiced excitation, $v'(n)$, can be obtained by a fast synthesis method composed of an IFFT and sampling rate conversion, in which harmonic magnitudes and fundamental frequency are linearly interpolated.

Suppose at the k^{th} frame we have a spectrum with M_1 harmonics, with the magnitude of each being A_m $(0 \le m < M_1)$. The pitch lag expressed in terms of the number of samples is $2M_1$. Appending zeros to this array of $A_m(n)$ yields a new array with 2^b components ranging from 0 to π. The number b can be arbitrarily chosen so that $M \le 2^b$. The same processing is done on the array of phase data.

The phase data used here are generated from those of the previous frame assuming that the fundamental frequency is linearly interpolated. 2^{b+1}-point IFFT is applied to these arrays of magnitude and phase data with the constraint that the results be real numbers. Now we have an over-sampled version of the time domain waveform over a one-pitch period. Let this be $w_1(i)$ $(0 \le i \le 2^{b+1})$. 2^{b+1} points are used to express the one-pitch period of the waveform, whereas the actual pitch is $2M_1$ since the over-sampling ratio ov_1 is .

$$ov_1 = 2^b / M_1 .$$

Similarly, one can get another one-pitch period of the waveform at the $k+1^{th}$ frame, which has an over-sampling ratio of.

$$ov_2 = 2^b / M_2 .$$

where the pitch lag is $2M_2$. Let this waveform be $w_2(i)$ $(0 \le i \le 2^{b+1})$. Here the function $f(n)$ is defined, which maps the time index n from the original sampling version to the over-sampled version under the condition that the fundamental frequency is linearly interpolated, to be.

$$f(n) = \int_0^n (ov_1 \frac{N_0 - t}{N_0} + ov_2 \frac{t}{N_0})\, dt .$$

The number of over-sampled data needed to reconstruct a waveform of length N_0 at the original sampling rate is at most L:

$$L = nint(f(N_0)) = nint(\frac{N_0}{2}(ov_1 + ov_2)) ,$$

where $nint(x)$ returns the nearest integer of x. Cyclically extending $w_1(i)$ and $w_2(i)$, we obtain waveforms $\tilde{w}_1(l)$ and $\tilde{w}_2(l)$ of length L:

$$\tilde{w}_1(l) = w_1(\bmod(l, 2^{b+1})) \qquad (0 \le l \le L),$$

$$\tilde{w}_2(l) = w_2(\bmod(offset + l, 2^{b+1})) \quad (0 \le l \le L),$$

where

$$offset = 2^{b+1} - \bmod(L, 2^{b+1}),$$

and $\bmod(x, y)$ returns the remainder of x divided by y. These two waveforms, $\tilde{w}_1(l)$ and $\tilde{w}_2(l)$, from the spectra of the k^{th} and $k+1^{th}$ frames have the same "pseudo" pitch (= 2^{b+1}) and they are aligned. So, simply adding these two waveforms using appropriate weights produces the result $w(l)$:

$$w(l) = \frac{L-1}{L}\tilde{w}_1(l) + \frac{1}{L}\tilde{w}_2(l) \qquad (0 \le l \le L),$$

where each A_m is linearly interpolated between the adjacent frames. Lastly, $w(l)$ has to be re-sampled so that the resulting waveform can be expressed at the original uniform sampling rate. This operation brings the waveform back from the "pseudo" pitch domain to the real pitch domain as well. In principle, the re-sampling operation is just

$$v'(n) = w(f(n)) \qquad (0 \le l \le L),$$

Usually $f(n)$ does not return integer values. So, $v'(n)$ is obtained by linearly interpolating $w(\lceil f(n) \rceil)$ and $w(\lfloor f(n) \rfloor)$. For a more general formulation, a higher order interpolation could be used. $\lceil x \rceil$ and $\lfloor x \rfloor$ denote the smallest integer greater than or equal to x, and the largest integer less than or equal to x, respectively. Typically b=6 is used. By the operation above, fundamental frequencies of the previous frame and current frame are linearly interpolated.

At the onset of the voiced signal, a different shape of the window is used as shown below.

```
for(i = 0; i < (int) ( FRM * 5.0 / 16.0); i++)
   c_dis[i]=1.0;
for(i=(int) (FRM * 5.0 / 16.0); i < (int) (FRM * 11.0 / 16.0); i++)
   c_dis[i]=((FRM * 11.0 / 16.0) - (float) i ) / (FRM * 6.0 / 16.0);
for(i=(int) (FRM * 11.0 / 16.0); i < FRM; i++)
   c_dis[i]=0.0;
for(i=0;i<FRM; i++)
   win[i]=1-c_dis[i];
```

where the frame interval FRM=160. win[i] could be used for windowing the on-set of voiced waveform. The shape of c_dis[i] and win[i] are shown in figure 3.6.2. Constants are set as: HM_UP=HM_DOWN=60, HM_FLAT=50. The position of the windows are shifted right by LD_LEN samples when low delay decode mode is selected. The windows shown in the figure 3.6.2 are also used for a voiced frame which is adjacent to a voiced frame when the ratio of fundamental frequency change | (w02-w01)/w02 | is more than WDEVI, where w01 and w02 are fundamental frequencies of previous frame and current frame respectively, WDEVI=0.1. When the ratio of fundamental frequency change is more than WDEVI, it is regarded that pitch contour is discontinuous. In such case, fundamental frequencies w01 and w02 are not interpolated. Independently synthesized periodic waveform with fixed fundamental frequency w01 using harmonic magnitudes of the previous frame, and synthesized periodic waveform with fixed fundamental frequency w02 using harmonic magnitudes of the current frame are added using the weighting windows shown in figure 3.6.2.

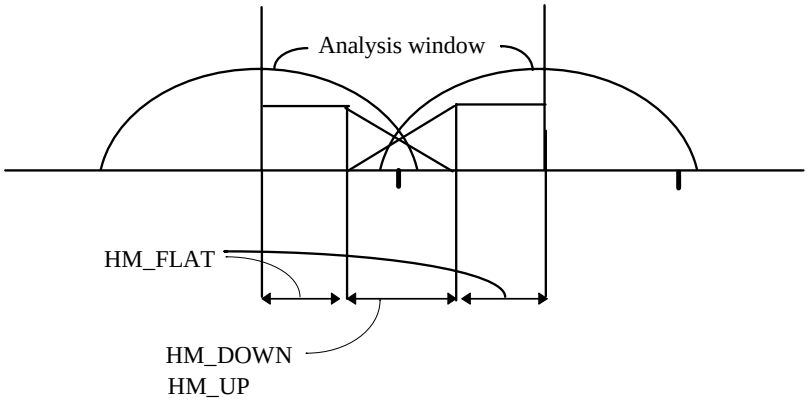


Fig 3.6.2

Noise component addition:
A voiced excitation signal is composed of harmonic components described above and noise components. The amount of noise to be added is controlled by V/UV flag. Harmonic magnitudes are also slightly modified by the information. The parameter control of noise and harmonic magnitude is described below.

When V/UV is 0:
Do nothing. Just VXC decoder works.

When V/UV is 1:

$$Am\_noise[i] = \begin{cases} 0 & (1 \le i < send * B\_TH1) \\ Am[i] * AN1 & (send * B\_TH1 \le i \le send) \end{cases}$$

$$Am\_h[i] = \begin{cases} Am[i] & (1 \le i < send * B\_TH1) \\ Am[i] * AH1 & (send * B\_TH1 \le i \le send) \end{cases}$$

When V/UV is 2:

$$Am\_noise[i] = \begin{cases} 0 & (1 \le i < send * B\_TH2) \\ Am[i] * AN2 & (send * B\_TH2 \le i < send * B\_TH2\_2) \\ Am[i] * AN2\_2 & (send * B\_TH2\_2 \le i \le send) \end{cases}$$

$$Am\_h[i] = \begin{cases} Am[i] & (1 \le i < send * B\_TH2) \\ Am[i] * AH2 & (send * B\_TH2 \le i < send * B\_TH2\_2) \\ Am[i] * AH2\_2 & (send * B\_TH2\_2 \le i \le send) \end{cases}$$

When V/UV is 3:

$$Am\_noise[i] = \begin{cases} 0 & (1 \le i < send * B\_TH3) \\ Am[i] * AN3 & (send * B\_TH3 \le i \le send) \end{cases}$$

$$Am\_h[i] = \begin{cases} Am[i] & (1 \le i < send * B\_TH3) \\ Am[i] * AH3 & (send * B\_TH3 \le i \le send) \end{cases}$$

Constant values:

2 kbps:

B_TH1	AN1	AH1	B_TH2	AN2	AH2	B_TH2_2	AN2_2	AH2_2	B_TH3	AN3	AH3
0.5	0.4	0.8	0.5	0.3	0.9	0.85	0.5	0.5	0.7	0.2	1.0

4 kbps:

B_TH1	AN1	AH1	B_TH2	AN2	AH2	B_TH2_2	AN2_2	AH2_2	B_TH3	AN3	AH3
0.5	0.3	0.9	0.5	0.3	0.9	0.85	0.5	0.5	0.8	0.2	1.0

Noise Component Generation:

For the noise component addition for voiced excitation, white Gaussian noise is first generated. It is then colored and gain controlled by Am_noise[i] parameter derived above, and weighted overlap&add is used to generate continuous noise components.

Let ns[i] (0<=i<SAMPLE) be samples of Gaussian noise with zero mean and unit variance. Hamming window of length SAMPLE ham[i] (0<=i<SAMPLE) is then multiplied to ns[i] and obtain wns[i],

```
wns[i]=ns[i] * ham[i]
```

SAMPLE point FFT of wns[i] is then computed and real part re[i] (0<=i<SAMPLE) and imaginary part im[i] (0<=i<SAMPLE) are obtained. Let rms[i] and ang[i] be,

```
rms[i]=sqrt(re[i]*re[i] + im[i]*im[i])  (0<=i<SAMPLE/2)
ang[i]=arctan (im[i]/re[i])            (0<=i<SAMPLE/2)
```

Let nh be a number of harmonics from 0 to 4000 Hz (fs/2) and w01 be a fundamental frequency, where $2*\pi$ is expressed as SAMPLE. (SAMPLE=256)

```
nh=floor(pitch/2),
w01=SAMPLE/pitch,
```

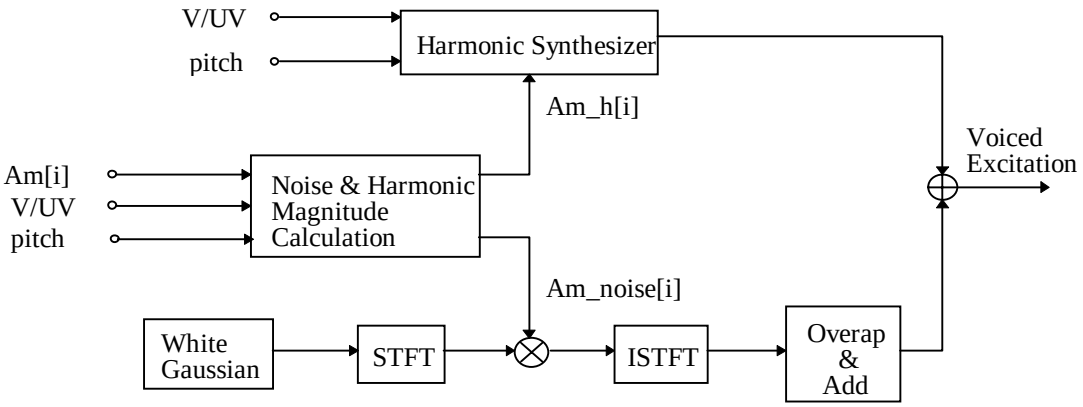
where pitch is a pitch lag value of the current frame. Let gain scaled noise magnitude am1[i] be,

```
am1[i]=SCALEFAC*Am_noise[i]
SCALEFAC=10
```

Then colored and gain adjusted noise component is obtained as,

```
for(i=0;i<=nh;i++){
   if (i==0)
      lb=0;
   else
      lb=ub+1;
   if(i==nh)
      ub=SAMPLE-1;
   else
      ub=inint((float)i * w01 + w01/2.);
   if(ub >= SAMPLE/2)
      ub=SAMPLE/2 ;
   bw=ub-lb+1;
   s=0.;
   for(j=lb;j<=ub; j++)
      s=s+rms[j]*rms[j];
   s=sqrt(s/(float)bw);
   for(j=lb;j<=ub;j++)
      rms[j]=am1[i]*rms[j]/s;
   }
```

where inint(x) returns nearest integer of x.
Here, gain controlled and colored noise spectral sequence rms[i] is obtained. Together with original phase ang[i] (0<=i<SAMPLE/2), SAMPLE point inverse FFT is conducted with a constraint that the result is real numbers.



Let the result of the IFFT be cns[i] (0<=i<SAMPLE). In order to combine the noise component of the previous frame, weighted overlap and add is used.

First, (FRM*2-SAMPLE)/2 of zeros are padded to both sides (beginning and ending) of cns[i] (0<=i<SAMPLE) to form a new sequence cns_z[i] (0<=i<FRM*2). In the same manner, let us define ham_z[i] (0<=i<FRM*2) where (FRM*2-SAMPLE)/2 of zeros are padded to both sides of the SAMPLE point Hamming window ham[i] (0<=i<SAMPLE). FRM is the frame interval (=160). When a frame is Unvoiced, cns_z[i]=0 (0<=i<FRM*2) Let us denote the cns_z[i] of the previous frame cns_z_p[i]. Now previous and current frame combined noise componet s_uv[i] (0<=i<FRM) is obtained from the cns_z[i] and cns_z_p[i].

```
s_uv[i]=(cns_z_p[FRM+i]*ham_z[FRM+i]+cns_z[i]*ham_z[i])
   /(ham_z[FRM+i]*ham_z[FRM+i]+ham_z[i]*ham_z[i])          (0<=i<FRM)
```

In the case of Low delay decode mode, s_uv[i] generation is slightly modified, where ham_z[i] is formed by using center of HAMLD samples of ham[i] (0<=i<SAMPLE), and (FRM*2-HAMLD)/2 of zeros are padded to the beginning (left side) and (FRM*2-HAMLD)/2 + LD_LEN of zeros are padded to the end (right side) of extracted HAMLD non-zero samples. (HAMLD=240, LD_LEN=20)

```
s_uv[i]=(cns_z_p[FRM+i+LD_LEN]*ham_z[FRM+i+LD_LEN]+cns_z[i+LD_LEN]*ham_z[i+LD_LEN])

/(ham_z[FRM+i+LD_LEN]*ham_z[FRM+i+LD_LEN]+ham_z[i+LD_LEN]*ham_z[i+LD_LEN])
  (0<=i<FRM)
```

Noise component s_uv[i] is added to harmonic excitation waveform to generate voice excitation.

Harmonic excitation generation process:

The voiced excitation generation by the harmonic synthesis shown above could be realized by the operations below.

```
static void ifft_am(float *rms, float *ang, float *aryd)
{
    int    i;
    float re[SAMPLE/2],im[SAMPLE/2];

    for(i=0; i <= SAMPLE/4 ; i++) {
     re[i] = rms[i] * cos(ang[i]);
     im[i] = rms[i] * sin(ang[i]);
    }
    for(i=SAMPLE/4+1; i< SAMPLE/2 ; i++) {
     re[i]=re[SAMPLE/2-i];
     im[i]=(-1)*im[SAMPLE/2-i];
    }
```

```
    IPC_ifft(re,im,7);

    for(i=0; i < SAMPLE/2; i++)
     aryd[i]=SAMPLE/2*re[i];

}


static void modu2pai(float x)
{
   float y;

   if (x >= 0.) {
      y = x - floor(x/(2*M_PI))*(2*M_PI);
      if (y > M_PI)
         y = y - 2*M_PI;
      return(y);
   }
   else {
      y = x - ceil(x/(2*M_PI))*(2*M_PI);
      if (y < -M_PI)
         y = y + 2*M_PI;
      return(y);
   }
}


void IPC_vExt_fft(float *pch, float (*am)[3], int *vuv, float *sv)
{
    int i,s;
    float w01,w02,w0s,fs;
    int send1,send2,send,sendm,wdevif;
    float am2[SAMPLE/2];
    static int old_old_vuv=0;
    float aw,dw;
    int phrst;

    float phai2[SAMPLE/2];
    float ovsr1,ovsr2,ovsrc;
    float lp1,lp2,lp12,ilp12;
    float lp1r,lp2r,lp12r;
    static float wave1[SAMPLE/2],wave2[SAMPLE/2];

    float out[2000],out2[2000],out3[2000];
    float c_dis_lp12[2000];

    int st;
    float ffi,fi,ffim,ffip;
    float sv1[FRM],sv2[FRM];
    float pha1[SAMPLE/2];
    static float pha2[SAMPLE/2];

    int iflat = 0,iflat2 = 0;

    float  b_th1;
    float  b_th2;
    float  b_th3;

    float  gv_th1;
    float  gv_th2;
    float  gv_th3;

    float  b_th2_2;
    float  gv_th2_2;
```

Fundamental frequency w01 of the beginning of the decode interval boundary is computed:

```
    w01 = (float)(2.*M_PI/pch[0]);
```

Fundamental frequency w02 of the ending of the decode interval boundary is computed:

```
    w02 = (float)(2.*M_PI/pch[1]);
```

Normalized fundamental frequency w0s (in the oversampled domain)
of the entire range the decode intervals computed:

```
w0s = (float)(2.*M_PI/(SAMPLE/2.));
```

Number of harmonics corresponding to w01:

```
send1= (int)(pch[0]/2.);
```

Number of harmonics corresponding to w02:

```
send2= (int)(pch[1]/2.);
```

When the ratio of fundamental frequency change is greater than WDEVI (=0.1),
wedvif is set to 1, otherwise set to 0.

```
if(fabs((w02-w01)/w02) > WDEVI )
 wdevif=1;
else
 wdevif=0;
```

If the V/UV of the frame centered around the beginning boundary of the decode interval is
UV, and that of the previous frame is also UV, then phase initialization flag phrst is set
to 1, otherwise 0.

```
if(vuv[0] ==0 && old_old_vuv ==0)
phrst=1;
 else
phrst=0;
```

Constants for harmonic magnitude adjustment for noise component addition are set.

```
if(ipc_decMode == DEC2K)   /* 2kbps decoding mode */
{
 b_th1 = B_TH1_2K;
 b_th2 = B_TH2_2K;
 b_th3 = B_TH3_2K;

 gv_th1 = GV_TH1_2K;
 gv_th2 = GV_TH2_2K;
 gv_th3 = GV_TH3_2K;

 b_th2_2 = B_TH2_2;
 gv_th2_2 = GV_TH2_2;
}
else    /* 4kbps decoding mode */
{
 b_th1 = B_TH1_4K;
 b_th2 = B_TH2_4K;
 b_th3 = B_TH3_4K;

 gv_th1 = GV_TH1_4K;
 gv_th2 = GV_TH2_4K;
 gv_th3 = GV_TH3_4K;

 b_th2_2 = B_TH2_2;
 gv_th2_2 = GV_TH2_2;
}
```

Harmonic magnitude am2[] at the ending boundary of decode interval and initial phase
value pha1[] are set.
Phase value pha2[] at the ending boundary of decode interval are computed using
pha1[], w01, w02 and frame interval.
In the case of phase reset is happening, random phase uniformly distributing between
0-0.5π is used.

```
for(i=0; i<SAMPLE/2; i++){

 am2[i]=am[i][2];
 pha1[i] = pha2[i] ;

 if(phrst==1)
```

```
 {
     pha2[i] = 0.5 * M_PI * random() / (float) RND_MAX;
 }
 else
 {
     if(ipc_decDelayMode == DM_SHORT)
     {
      pha2[i] += (float)i * ((float)(FRM-LD_LEN) * (w01+w02) * 0.5 + w01 *
         (float)LD_LEN) ;
     }
     else
     {
      pha2[i] += (w01+w02)*(float)i*(float)FRM/2.0;
     }
 }

 pha2[i] = modu2pai( pha2[i] );
}
```

When the V/UV of the encoder frame centered around the ending boundary of the decode interval UV, harmonic magnitudes of that point am2[] are set to 0.

```
if(vuv[1]==0){
 for(i=0; i<SAMPLE/2; i++)
  am2[i]=0.;
}
```

Harmonic magnitudes are adjusted for noise component addition.

```
for(i=0;i<SAMPLE/2;i++)
{
 if(vuv[1]==1)
 {
  if(i >= (float)send2 * b_th1)
  {
   am2[i]=am2[i] * gv_th1;
  }
 }
 else if(vuv[1]==2)
 {
  if(i >= (float)send2 * b_th2 && i < (float)send2 * b_th2_2)
  {
   am2[i]= am2[i] * gv_th2;
  }
  else if(i>= (float)send2 * b_th2_2)
  {
   am2[i]= am2[i] * gv_th2_2;
  }
 }
 else if(vuv[1]==3)
 {
  if(i >= (float)send2 * b_th3)
  {
   am2[i]=am2[i] * gv_th3;
  }
 }
}

for(i=send2+1;i<SAMPLE/2;i++)
 am2[i]=0.;
```

Maximum number of harmonics “sendm” and minimum number of harmonics “send” during the decode interval are set.

```
if(send1 <= send2){
 send=send1;
 sendm=send2;
}
else {
```

```
      send=send2;
      sendm=send1;
    }
    s = FRM;
    fs = (float)s;

    aw=(w01+w02)*fs/2.;
    dw=(w02-w01)/(2.*fs);
```

When the change of the fundamental frequency is small enough and continuous pitch interpolation is possible, the voiced excitation synthesis below is executed.

```
    if(wdevif == 0){
```

One pitch period of waveform in the oversampled domain in the previous time is copied to wave1[].

```
   for(i=0;i<SAMPLE/2;i++)
       wave1[i]=wave2[i];
```

over sampling ratio ovsr1 and ovsr2 at the beginning and ending boundaries of decode interval are computed.

```
   ovsr1= (float)(SAMPLE/2.)/pch[0];
   ovsr2= (float)(SAMPLE/2.)/pch[1];
```

A number of samples representing the waveforms in oversampled domain are computed, where lp1 corresponds to am1[] and lp2 am2[].

```
   lp1=ceil(fs*ovsr1);
   lp1r=floor(fs*ovsr1 + 0.5);
   lp2=ceil(fs*ovsr2);
   lp2r=floor(fs*ovsr2 + 0.5);
```

A number of samples representing the pitch interpolated waveform for the decoding interval in the oversampled domain using w01 and w02 are computed,

```
   if(ipc_decDelayMode == DM_SHORT)
   {
       lp12=ceil((float)(FRM-LD_LEN)*(ovsr1*0.5+ovsr2*0.5)+(float)LD_LEN*ovsr1);
       lp12r=floor((float)(FRM-LD_LEN)*(ovsr1*0.5+ovsr2*0.5)+(float)LD_LEN*ovsr1
              + 0.5);

       iflat=(int)floor(ovsr1 * (float)LD_LEN + 0.5);
       iflat2=(int)floor(ovsr2 * (float)LD_LEN + 0.5);
   }
   else
   {
       lp12=ceil(fs*(ovsr1*0.5+ovsr2*0.5));
       lp12r=floor(fs*(ovsr1*0.5+ovsr2*0.5) + 0.5);
   }
```

One pitch period of waveform wave1[] is cyclically extended to have sufficient length in the over sampled domain. Time domain waveform out[] due to harmonic magnitudes am1[] is obtained.

```
   if(ipc_decDelayMode == DM_SHORT)
   {
       for(i=0;i<=lp12+iflat2+10;i++)
        out[i]=wave1[(i)%(SAMPLE/2)];
   }
   else
   {
       for(i=0;i<lp12;i++)
        out[i]=wave1[(i)%(SAMPLE/2)];
   }
```

One pitch period of time domain waveform wave2[] in the over sampled domain due to the

harmonic magnitudes am2[] are computed by IFFT.

```
ifft_am(am2,phai2,wave2);
```

One pitch period of waveform wave2[] is cyclically extended to have sufficient length in the over sampled domain. Time domain waveform out2[] due to harmonic magnitudes am2[] is obtained.

```
st=SAMPLE/2-(((int)lp12r)%(SAMPLE/2));

if(ipc_decDelayMode == DM_SHORT)
{
    for(i=0; i<lp12+iflat2+10; i++)
     out2[i]=wave2[(st+i)%(SAMPLE/2)];
}
else
{
    for(i=0; i<lp12; i++)
     out2[i]=wave2[(st+i)%(SAMPLE/2)];
}
```

out3[] is obtained by weighted overlap and add using out[] and out2[].
Here triangular windows are used for the weighting.

```
if(ipc_decDelayMode == DM_SHORT)
{
    for(i=0;i<iflat;i++)
     c_dis_lp12[i]=1.0;

    ilp12=1./(lp12-(float)iflat);
    for(i=iflat;i<(int)lp12;i++)
             c_dis_lp12[i]=((float)lp12 - (float)i)*ilp12;

    for(i=(int)lp12;i<(int)lp12+iflat2;i++)
   c_dis_lp12[i]=0.;

    for(i=0;i<lp12+iflat2;i++)
     out3[i]=out[i] * c_dis_lp12[i] + out2[i] * (1. - c_dis_lp12[i]);
}
else
{
    ilp12=1./lp12;
    for(i=0;i<lp12;i++)
     out3[i]=out[i]*(lp12-(float)i)*ilp12 + out2[i]*(float)i*ilp12;
}
```

out3[] is re-sampled to obtain the waveform sv[] that is represented in the original real sampling rate. The re-sampling operation where the sampling rate is linearly interpolated is shown below.

```
if(ipc_decDelayMode == DM_SHORT)
{
    sv[0]=out3[iflat];
    ffi=0;
    for(i=1;i<s;i++){
     if( i< (s- LD_LEN ))
      ovsrc=ovsr1 * ((float)(s- LD_LEN -i)/(float)(s-LD_LEN))
             + ovsr2 * (float)i/(float)(s-LD_LEN);
     else
      ovsrc=ovsr2;
     ffi=ffi+ovsrc;
     ffim=floor(ffi);
     ffip=ceil(ffi);
     if(ffim == ffip)
      sv[i]=out3[(int)ffip + iflat];
     else
      sv[i]=(ffi-ffim)*out3[(int)ffip+iflat]+(ffip-ffi)*out3[(int)ffim+iflat];

    }
```

```
}
else
{
    sv[0]=out3[0];
    ffi=0.;
    for(i=1;i<s;i++){
     fi=(float)i;
     ovsrc=ovsr1 * ((float)(s-i)/(float)s)
           + ovsr2 * (float)i/(float)s;
     ffi=ffi+ovsrc;

     ffim=floor(ffi);
     ffip=ceil(ffi);
     if(ffim == ffip)
      sv[i]=out3[(int)ffip];
     else
      sv[i]=(ffi-ffim)*out3[(int)ffip]+(ffip-ffi)*out3[(int)ffim];
    }
  }
 }
```

When the change of the fundamental frequency is large and continuous pitch interpolation is not possible, the voiced excitation synthesis below is executed.

```
 else
 {
```

One pitch period of waveform in the oversampled domain in the previous time is copied to wave1[].

```
for(i=0;i<SAMPLE/2;i++)
    wave1[i]=wave2[i];
```

over sampling ratio ovsr1 and ovsr2 at the beginning and ending boundaries of decode interval are computed.

```
ovsr1= (float)(SAMPLE/2.)/pch[0];
ovsr2= (float)(SAMPLE/2.)/pch[1];
```

A number of samples representing the waveforms in oversampled domain are computed, where lp1 is corresponding for am1[] and lp2 is corresponding for am2[].

```
lp1=ceil(fs*ovsr1);
lp2=ceil(fs*ovsr2);
lp2r=floor(fs*ovsr2 + 0.5);
```

One pitch period of waveform wave1[] is cyclically extended to have sufficient length in the over sampled domain. Time domain waveform out[] due to harmonic magnitudes am1[] is obtained.

```
if(ipc_decDelayMode == DM_SHORT)
{
    iflat=(int)floor(ovsr1 * (float)LD_LEN + 0.5);
    iflat2=(int)floor(ovsr2 * (float)LD_LEN + 0.5);

    for(i=0;i<= lp1 + iflat +10;i++)
     out[i]=wave1[(i)%(SAMPLE/2)];
}
else
{
    for(i=0;i<lp1;i++)
     out[i]=wave1[(i)%(SAMPLE/2)];
}

for(i=0;i<SAMPLE/2;i++)
    phai2[i]= modu2pai ( pha2[i] );
```

One pitch period of time domain waveform wave2[] in the over sampled domain due to the

harmonic magnitudes am2[] are computed by IFFT.

```
ifft_am(am2,phai2,wave2);
```

One pitch period of waveform wave2[] is cyclically extended to have sufficient length in the over sampled domain. Time domain waveform out2[] due to harmonic magnitudes am2[] is obtained.

```
st=SAMPLE/2-(((int)lp2r)%(SAMPLE/2));

if(ipc_decDelayMode == DM_SHORT)
{
    for(i=0; i<=lp2 + iflat2 +10; i++)
     out2[i]=wave2[(st+i)%(SAMPLE/2)];
}
else
{
    for(i=0; i<lp2; i++)
     out2[i]=wave2[(st+i)%(SAMPLE/2)];
}
```

out[] and out2[] are re-sampled to obtain the waveform sv1[] and sv2[] that are represented in the original real sampling rate.
The re-sampling operation is shown below.

```
if(ipc_decDelayMode == DM_SHORT)
{
    sv1[0]=out[iflat];
    ffi=0.;
    for(i=1;i<s;i++){
     fi=(float)i;
     ffi=ffi+ovsr1;
     ffim=floor(ffi);
     ffip=ceil(ffi);
     if(ffim == ffip)
      sv1[i]=out[(int)ffip+iflat];
     else
      sv1[i]=(ffi-ffim)*out[(int)ffip+iflat]+(ffip-ffi)*out[(int)ffim+iflat];
    }

    sv2[0]=out2[iflat2];
    ffi=0.;
    for(i=1;i<s;i++){
     fi=(float)i;
     ffi=ffi+ovsr2;
     ffim=floor(ffi);
     ffip=ceil(ffi);
     if(ffim == ffip)
      sv2[i]=out2[(int)ffip+iflat2];
     else
      sv2[i]=(ffi-ffim)*out2[(int)ffip+iflat2]
             +(ffip-ffi)*out2[(int)ffim+iflat2];
    }
```

sv[] is obtained by weighted overlap and add using sv1[] and sv2[].
Here trapezoid window c_dis[] shown in the figure 3.6.2 is used for the weighting.

```
    for(i=0;i<s;i++)
   sv[i]=sv1[i]*c_dis[i+LD_LEN]+sv2[i]*(1.-c_dis[i+LD_LEN]);
}
else
{
    sv1[0]=out[0];
    ffi=0.;
    for(i=1;i<s;i++){
     fi=(float)i;
     ffi=ffi+ovsr1;
     ffim=floor(ffi);
     ffip=ceil(ffi);
```

```
        if(ffim == ffip)
          sv1[i]=out[(int)ffip];
        else
          sv1[i]=(ffi-ffim)*out[(int)ffip]+(ffip-ffi)*out[(int)ffim];
      }

      sv2[0]=out2[0];
      ffi=0.;
      for(i=1;i<s;i++){
        fi=(float)i;
        ffi=ffi+ovsr2;
        ffim=floor(ffi);
        ffip=ceil(ffi);
        if(ffim == ffip)
          sv2[i]=out2[(int)ffip];
        else
          sv2[i]=(ffi-ffim)*out2[(int)ffip]+(ffip-ffi)*out2[(int)ffim];
      }
```

sv[] is obtained by weighted overlap and add using sv1[] and sv2[].
Here trapezoid window c_dis[] shown in the figure 3.6.2 is used for the weighting.

```
      for(i = 0; i < s; i++)
            sv[i] = sv1[i] * c_dis[i]+ sv2[i] * (1.0 - c_dis[i]);
      }
  }

  old_old_vuv=vuv[0];

}
```

LPC synthesis:
In this decoder, independent LPC synthesis filters are used for voiced and unvoiced signals
The voiced excitation signal obtained above is fed into the LPC synthesis filter.

LPC synthesis filter:

$$H(z) = \frac{1}{\sum_{n=0}^{P} \alpha_n z^{-n}}$$

where α_n are linear predictive coefficients converted from de-quantized and linearly interpolated LSPs, which are updated every 2.5 ms ($P = 10$). When low delay decode mode is selected, decode frame interval is shifted by 2.5 ms and interpolation of LSPs is carried out for the first 17.5ms of decode frame interval shown in Fig. 3.6.1, and the latest LSPs are used for the last 2.5 ms without interpolation. Output of the LPC synthesis filter is fed into the postfilter described in section 3.1 of Annex A.

3.7 Unvoiced Component Synthesizer

3.7.1 Tool description

The unvoiced component synthesizer is composed of LPC synthesis filter and post filter operation. For unvoiced segments, VXC (CELP) scheme is used. LPC coefficients of only the current frame are used for two sub-frames without any interpolation either in the encoder and decoder.

3.7.2 Definitions

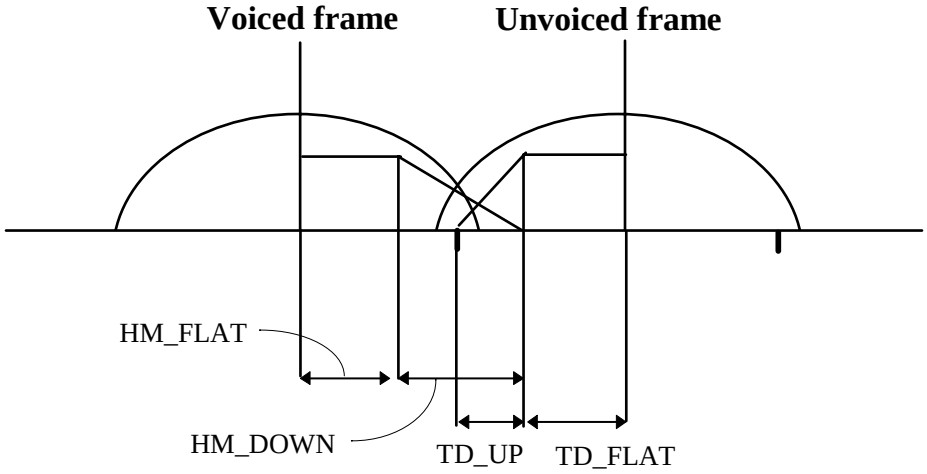
3.7.3 Synthesis process

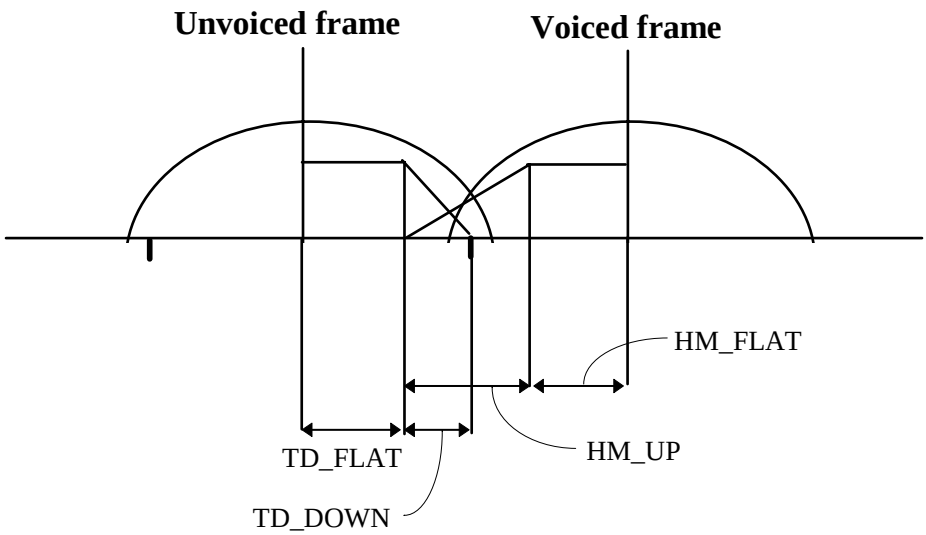
The unvoiced excitation signal generated is windowed to be smoothly connected to voiced signal. The figures below show the window shape for excitation waveform where V/UV is changed fron voiced to unvoiced and unvoiced to voiced. The parameters in the figure are set as:TD_UP=30, TD_FLAT=50, HM_DOWN=60, HM_FLAT=50, TD_DOWN=30, HM_UP=60. These windows for an unvoiced frame are used only when an unvoiced frame is placed adjacent to voiced or mixed voiced frame. The position of the windows are shifted right by LD_LEN samples when low delay decode mode is selected. Unvoiced excitation is then fed into LPC synthesis filter. As explained above, independent LPC synthesis filters are used for voiced and unvoiced.

LPC synthesis filter:

$$H(z) = \frac{1}{\sum_{n=0}^{P} \alpha_n z^{-n}}$$

where α_n are linear predictive coefficients converted from de-quantized LSPs, which are updated every 20 ms ($P = 10$). When normal decode mode is selected, LSP coefficients are updated at the middle of decode frame interval. If low delay decode mode is selected , decode frame interval is shifted by 2.5 ms and LSP update happens at 7.5 ms point from the beginning of the decode interval in Fig 3.6.1. Output of the LPC synthesis filter is fed into the postfilter described in section 3.1 of Annex A.





3.8 Bit Allocations

Shown below is bit allocations of 2.0 and 4.0 kbps HVXC. The bit allocations are common for both normal and low delay mode.

	V	Common	UV
LSF1		18bits/20msec	
LSF2		8bits/20msec	
V/UV		2bits/20msec	
pitch	7bits/20msec		
harmonic1 shape	4+4bits/20msec		
harmonic1 gain	5bits/20msec		
harmonic2 split	32bits/20msec		
VXC1 shape			6bits/10msec
VXC1 gain			4bits/10msec
VXC2 shape			5bits/5msec
VXC2 gain			3bits/5msec
Total(1) 2kbps	40bits/20msec		40bits/20msec
Total(1&2)4kbps	80bits/20msec		80bits/20msec

3.9 Variable rate decoder

This chapter describes tools for variable rate decoding with HVXC core. This tool allows HVXC to operate at variable bit rates, where average bit rate is reduced to 1.2 - 1.7 kbps with typical speech material. The major part of the algorithm is composed of "background noise interval detection", where only the mode bits are received during the "background noise mode", and unvoiced frame is received with certain period of time for background noise generation.

idVUV is a parameter that has the result of V/UV decision and defined as;

$$\text{idVUV} = \begin{cases} 0 & \text{Unvoiced speech} \\ 1 & \text{Background noise interval} \\ 2 & \text{Mixed voiced speech} \\ 3 & \text{Voiced speech} \end{cases}$$

Using the background noise detection method, variable rate coding is carried out based on fixed bit rate 2kbps HVXC. For idVUV=2,3, the same decoding method as fixed bit rate mode is used.

Mode(idVUV)	Back Ground Noise(1)	UV(0)	MV(2),V(3)
V/UV LSP Excitation	2bit/20msec 0bit/20msec 0bit/20msec	2bit/20msec 18bit/20msec 8bit/20msec (gain only)	2bit/20msec 18bit/20msec 20bit/20msec (Pitch&harmonic spectral parameters)
Total	2bit/20msec 0.1kbps	28bit/20msec 1.4kbps	40bit/20msec 2.0kbps

In the decoder, two sets of LSP parameters of previously transmitted ones are holded.

prevLSP1: previously transmitted LSP parameters
prevLSP2: previously transmitted LSP parameters before prevLSP1

For “Background noise” frame, VXC decoder is used in the same manner as UV frame. However, when the "Background noise" mode is selected, no LSP parameters are sent. LSP parameters generated by linearly interpolating prevLSP1 and prevLSP2 are used for LPC synthesis, and the same gain index as the previous frame is used for excitation generation of VXC decoding. During the period of "Background noise" mode, UV frames are inserted every N (=9) frames to transmit background noise parameters. This UV frame may or may not be a real UV frame of the beginning of speech bursts. Whether or not the frame is real UV is judged by transmitted gain index. If the gain index is smaller than or equal to that of previous one+2, then this UV frame is regarded as "Background noise interval", and therefore the previously transmitted LSP (=prevLSP1) is used to keep the smooth variation of LSP parameters, otherwise; currently transmitted LSPs are used as real UV frame. The gain indices are sorted according to the magnitudes. If "Background noise" mode is selected again, then linearly interpolated LSPs using prevLSP1 and prevLSP2 are used. For both UV and BackGroundNoise mode (idVUV=0,1), gain normalized gaussian noise is used in place of stochastic shape codevector for VXC decoding.

Fig. 3.10.1 shows an example. Suppose that Frame 0 and Frame 1 are unvoiced and Frame 2…Frame 9 are background noise mode. Then during decoding of Frame 2…Frame 9, prevLSP1&2 are set as: prevLSP2=LSP(0) and prevLSP1=LSP(1). For decoding of Frame i (2<=i<=9), LSP parameters are generated as

LSP(i)= (prevLSP2*(2*BGN_INTVL -1 -2*bgnCnt) + prevLSP1*(1+2*bgnCnt))/(2*BGN_INTVL)

where, LSP(i) is an LSP vector of Frame i composed of 10 LSP coefficients. Maximum background noise interval BGN_INTVL=8. Frame count of continuous number of bacground noise frames is given by bgnCnt. In this example, bgnCnt=0 for Frame 2, bgnCnt=1 for Frame3, ….bgnCnt=7 for Frame 9. As for gain index of VXC decoding during Frame2…Frame9, the gain index of Frame 1 is used. When parameters of Frame 10 are received, prevLSP1&2 are updated as: prevLSP2=LSP(1), prevLSP1=LSP(10). For decoding of Frame 10, gain index is first checked whether or not it is larger than that of Frame 1’s index value + 2. If yes, Frame 10 is decoded as usual UV frame; otherwise it is decoded as background noise frame and LSP(1) is used instead of LSP(10) while received gain index is used for both cases.

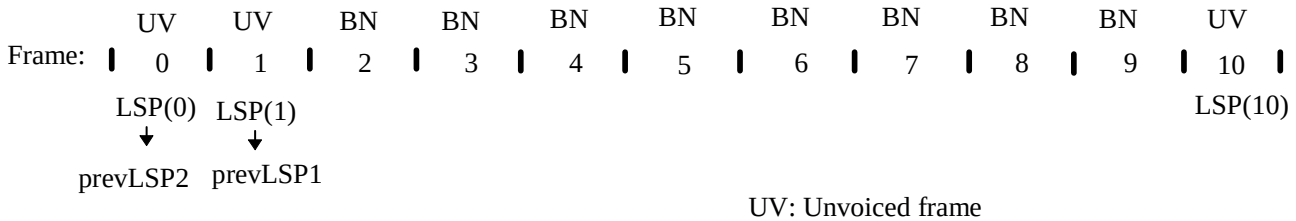


Fig. 3.10.1

UV: Unvoiced frame
BN: Bacground noise frame

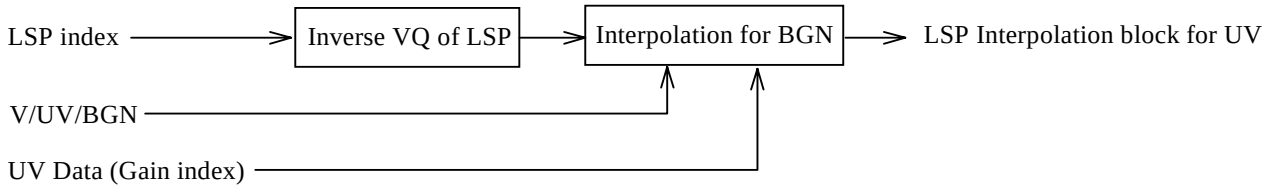


Fig. 3.10.2 Background noise decoder for variable rate decoding

4 HILN Decoder Tools

The Harmonic and Individual Lines plus Noise (HILN) decoder utilises a set of parameters which are encoded in the bitstream to describe the audio signal. Three different signal models are supported:

signal model	description	parameters
harmonic lines	group of sinusoidal signals with common fundamental frequency	fundamental frequency and amplitudes of the spectral lines
individual lines	sinusoidal signals	frequency and amplitude of the individual spectral lines
noise	spectrally shaped noise signal	spectral shape and power of the noise

The HILN decoder first reconstructs these parameters from the bitstream with a set of decoding tools and then synthesises the audio signal based on these parameters using a set of synthesiser tools:

- harmonic line decoder
- individual line decoder
- noise decoder
- harmonic and individual line synthesiser
- noise synthesiser

The HILN decoder tools reconstruct the parameters of the harmonic and individual lines (frequency, amplitude) and the noise (spectral shape) as well as possible envelope parameters from the bitstream.

The HILN synthesiser tools reconstruct one frame of the audio signal based on the parameters decoded by the HILN decoder tools for the current bitstream frame.

The HILN decoder supports a wide range of frame lengths and sampling frequencies. By scaling the synthesiser frame length with an arbitrary factor, speed change functionality is available at the decoder. By scaling the line frequencies with an arbitrary factor, pitch change functionality is available at the decoder.

The HILN decoder can operate in two different modes, as basic decoder and as enhanced decoder. The basic decoder which is used for normal operation only evaluates the information avaliable in the bitstream elements HILNbasicFrame() to reconstruct the audio signal. To allow large step scalability in combination with other codec cores (e.g. the T/F core) the additional bitstream elements HILNenhaFrame() need to be transmitted and the HILN decoder must operate in the enhanced mode which exploits the information of both HILNbasicFrame() and HILNenhaFrame(). This mode reconstructs an audio signal with well defined phase relationships which can be combined with a residual signal coded at higher bit rates using an enhancement codec (e.g. T/F codec with scalability tools). If the HILN decoder is used in this way as a core for a scalable coder no noise signal must be synthesised for the signal which is given to the enhancement decoder.

Although Section 2 describes a combined syntax for basic and enhancement data, it might be more useful to transmit the corresponding elements in separate transport streams and to control the decoding and synthesis process according to the availability of enhancement data.

4.1 Harmonic line decoder

4.1.1 Tool description

This tool decodes the parameters of the harmonics lines transmitted in the bitstream.

4.1.2 Definitions

harmFlag:	flag indicating harmonic line data in current frame
numHarmTransCod:	number of harmonic line group amplitudes-1
numHarmTrans :	number of harmonic line group amplitudes
addHarmAmplBits:	number of harmonic line amplitude bits - 4
harmPred:	flag indicating continuation of harmonic lines
harmEnv:	flag enabling envelope for harmonic lines
harmFreq:	coded harmonic lines fundamental frequency
harmFreqStretch:	coded harmonic lines frequency stretching
harmTransAmpl[i]:	coded harmonic line group amplitudes
harmFreqEnha[i]:	coded harmonic line i frequency enhancement
harmPhase[i]:	coded harmonic line i phase
habits:	number of harmonic line group amplitudes bits

4.1.3 Decoding Process

If the „harmFlag“ is set and thus HARMbasicPara() data and in enhancement mode HARMenhaPara() data is available in the current frame, the parameters of the harmonic lines are decoded and dequantised as follows:

4.1.3.1 Basic decoder

First the number of group amplitudes is derived:

numHarmTrans = numHarmTransCod+1

Next the fundamental frequency and stretching of the harmonic lines are dequantised:

hFreq = exp((harmFreq+0.5)/2048 * log(4000/30)) * 30
hStretch = (harmFreqStretch/32-0.5) * 0.002

Then the transmitted line amplitudes are dequantised:

```
haoff = (addHarmAmplBits) ? 10 : 5

amplInt = harmTransAmpl[0]
hTransAmpl[0] = exp(log(2)/3 * (amplInt+0.5))
for (i=1; i<numHarmTrans; i++) {
        amplInt += harmTransAmpl[i]-haoff
        hTransAmpl[i] = exp(log(2)/3 * (amplInt+0.5))
}
```

These transmitted amplitudes are valid for single harmonic lines or groups of neighboring lines. The transmitted amplitudes are mapped and spread to the harmonic line amplitudes according to the following algorithm:

```
for (i=0; i<numHarmTrans; i++)
        for (k=0; k<w[i]; k++)
                hLineAmpl[j0[i]+k] = hTransAmpl[i] / sqrt(w[i])
```

The values of the group width w[i] and the harmonic line start index j0[i] is given in the harmonic line grouping table in Section 4.1.4.

The total number of harmonic lines is calculated as follows:

```
harmNumLine = j0[numHarmTrans-1]+w[numHarmTrans -1]-1
```

Finally the frequencies of the harmonic lines are calculated:

```
for (i=0; i<harmNumLine; i++)
        hLinefreq[i] = hFreq * (i+1) * (1 + hStretch*(i+1))
```

The harmEnv and harmPred flags require no further dequantisation; they are directly passed on to the synthesiser tool.

4.1.3.2 Enhanced decoder

In this mode, the harmonic line parameters decoded by the basic decoder are refined and also line phases are decoded using the information contained in HARMenhaPara() as follows:

For the first maximum 10 harmonic lines i

```
i = 0 .. min(numHarmTrans,10)-1
```

the enhanced harmonic line parameters are refined using the basic harmonic line parameters and the data in the enhancement bitstream:

```
hLineAmplEnh[i] = hLineAmpl[i]
hLineFreqEnh[i] = hLineFreq[i] * (1+((harmFreqEnh[i]+0.5)/(2^fEnhbits[i])-0.5)*(hFreqRelStep-1))
```

where hFreqRelStep is the ratio of two neighbouring fundamental frequency quantiser steps:

```
hFreqRelStep = exp(log(4000/30)/2048))
```

For both line types the phase is decoded from the enhancement bitstream:

```
hLinePhaseEnh[i] = 2*pi*(harmPhase[i]+0.5)/(2^phasebits)-pi
```

4.1.4 Tables

Table: harmonic line grouping

transmitted amplitude index: i	group width: w[i]	harmonic line start index: j0[i]
0, 1, 2, 3, 4, 5, 6, 7, 8, 9	1	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
10, 11, 12, 13, 14, 15	2	10, 12, 14, 16, 18, 20
16, 17, 18, 19, 20	3	22, 25, 28, 31, 34
21, 22, 23, 24	4	37, 41, 45, 49
25, 26, 27	5	53, 58, 63
28, 29	6	68, 74

30	7	80

4.2 Individual line decoder

4.2.1 Tool description

The individual line basic bitstream decoder reconstructs the line parameters *frequency*, *amplitude*, and *envelope* from the bitstream. The enhanced bitstream decoder reconstructs the line parameters *frequency*, *amplitude*, and *envelope* with finer quantisation and additionally reconstructs the line parameters *phase*.

4.2.2 Definitions

numLine:	number of individual lines in current frame	
envFlag:	flag indicating envelope data in the current frame	
prevNumLine:	number of individual lines in previous frame	
envTmax:	coded envelope parameter in current frame:	time of maximum
envRatk:		attack rate
envRdec:		decay rate
prevLineContFlag[k]:	flag indicating line k in previous frame is continued in current frame	
lineContFlag[i]:	flag indicating line i in current frame has continued from previous frame	
lineEnvFlag[i]:	flag enabling envelope for individual line i	
lineAmpl[i]:	coded amplitude of individual line i	
lineFraq[i]:	coded frequency of individual line i	
lineAmplDelta[i]:	coded amplitude delta of individual line i	
lineFreqDelta[i]:	coded frequency delta of individual line i	
envTmaxEnha:	envelope enhancement parameter in current frame:	time of maximum
envRatkEnha:		attack rate
envRdecEnha:		decay rate
lineFreqEnha[i]:	coded individual line i frequency enhancement	
linePhase[i]:	coded individual line i phase	
linebits:	number of bits for numLine	
abits:	number of individual line amplitude bits	
fbits:	number of individual line frequencs bits	
dabits:	number of individual line amplitude delta bits	
dfbits:	number of individual line frequency delta bits	
tmbits:	number of envTmax bits	
atkbits:	number of encRatk bits	
decbits:	number of envRdec bits	
tmEnhbits:	number of envTmax bits	
atkEnhbits:	number of encRatk bits	
decEnhbits:	number of envRdec bits	
fEnhbits[i]:	number of lineFreqEnha[i] and harmFreqEnha[i] bits	
phasebits:	number of linePhase and harmPhase bits	
fmin:	minimum frequency of individual lines (in Hz)	
fmax:	maximum frequency of individual line (in Hz)	
amin:	miminum amplitude of individual line	
amax:	maximum amplitude of individual line	
dfmax:	maximum frequency delta of individual line	
damax:	maximum amplitude delta of individual line	

4.2.3 Decoding Process

4.2.3.1 Basic decoder

The basic decoder reconstructs the line parameters from the data contained in HILNbasicFrame() and INDIbasicPara() in the following way:

For each frame, first the number of individual lines encoded in this frame is read from HILNbasicFrame():

```
numLine
```

Then the frame envelope flag is read from HILNbasicFrame():

```
envFlag
```

If envFlag = 1 then the 3 envelope parameters t_max, r_atk, and r_dec are decoded from HILNbasicFrame():

```
t_max = (envTmax+0.5)/(2^tmbits)
r_atk = tan(pi/2*max(0,envRatk-0.5)/(2^atkbits-1))/0.2
r_dec = tan(pi/2*max(0,envRdec-0.5)/(2^decbits-1))/0.2
```

These envelope parameters are valid for the harmonic lines as well as for the individual lines. Thus the envelope parameters envTmax, envRatk, envRdec must be dequantised if present, even if numLine == 0.

For each line k of the previous frame

```
k = 0 .. prevNumLine-1
```

the previous line continuation flag is read from HILNbasicFrame():

```
prevLineContFlag[k]
```

If prevLineContFlag[k] = 1 then line k of the previous frame is continued in the current frame. If prevLineContFlag[k] = 0 then line k of the previous frame is not continued.

In the current frame, first the parameters of all continued lines are encoded followed by the parameters of the new lines. Therefore, the line continuation flag and the line predecessor are determined before decoding the line parameters:

```
i=0
for (k=0;k<prevNumLine;k++)
    if (prevLineContFlag[k])
        linePred[i] = k
        lineContFlag[i++] = 1
while (i<numLine)
    lineContFlag[i++]=0
```

For each line i of the current frame

```
i = 0 .. numLine-1
```

the line parameters are decoded from INDIbasicPara() now.

If envFlag = 1 then the line envelope flag is read from INDIbasicPara():

```
lineEnvFlag[i]
```

If lineContFlag[i] = 0 then the parameters of a new line are decoded from INDIbasicPara():

```
ampl[i] = exp((lineAmpl[i]+0.5)/(2^abits)*log(amax/amin))*amin
freq[i] = exp((lineFreq[i]+0.5)/(2^fbits)*log(fmax/fmin))*fmin
```

If lineContFlag[i] = 1 then the parameters of a continued line are decoded from INDIbasicPara() based on the amplitude and frequency parameters decoded in the previous frame without refinement:

```
ampl[i] = prev_ampl[linePred[i]]*
    exp((lineAmplDelta[i]+1)/(2^dabits)-0.5)*log(damx*damax)/envStart
freq[i] = prev_freq[linePred[i]]*
    (1+((lineFreqDelta[i]+1)/(2^dfbits)-.5)*dfmax*2)
```

where envStart is the value of the amplitude envelope at the frame start boundary:

```
if (envFlag==1  && lineEnvFlag[i]==1)
    envStart = max(0.1,1-t_max*r_atk)
```

```
else
        envStart = 1
```

The dequantised line parameters are stored for decoding the line parameters of the next frame:

```
prevNumLine = numLine
prev_ampl[k] = ampl[k]* envEnd
prev_freq[k] = freq[k]
```

where envEnd is the value of the amplitude envelope at the frame end boundary:

```
if (envFlag==1  && lineEnvFlag[i]==1)
        envEnd = max(0.1,1-(1-t_max)*r_dec)
else
        envEnd = 1
```

If the decoding process starts with an arbitrary frame of a bitstream all individual lines which are marked in the bitstream as to be continued from previous frames which have not been decoded are to be muted.

4.2.3.2 Enhanced decoder

The enhanced decoder refines the line parameters obtained from the basic decoder and also decodes the line phases. The additional information is contained in bitstream element INDIenhaPara() and evaluated in the following way:

First, all operations of the basic decoder have to be carried out in order to allow correct decoding of parameters for continued lines.

If envFlag = 1 then the enhanced parameters t_maxEnh, r_atkEnh, and r_decEnh are decoded using the envelope data contained in HILNbasicFrame() and HILNenhaFrame():

```
t_maxEnh = (envTmax+(envTmaxEnh+0.5)/(2^tmEnhbits))/(2^tmbits)
if (envRatk==0)
        r_atkEnh = 0
else
        r_atkEnh = tan(pi/2*(envRatk-1+(envRatkEnh+0.5)/(2^atkEnhbits))/(2^atkbits-1))/0.2
if (envRdec==0)
        r_decEnh = 0
else
        r_decEnh = tan(pi/2*(envRdec-1+(envRdecEnh+0.5)/(2^decEnhbits))/(2^decbits-1))/0.2
```

For each line i of the current frame

```
i = 0 .. numLine-1
```

the enhanced line parameters are obtained by refining the parameters from the basic decoder with the data in INDIenhaPara():

```
amplEnh[i] = ampl[i]
if (fEnhbits[i]!=0)
        freqEnh[i] = freq[i] * (1+((lineFreqEnh[i]+0.5)/(2^fEnhbits[i])-0.5)*(freqRelStep-1))
else
        freqEnh[i] = freq[i]
```

where freqRelStep is the ratio of two neighbouring frequency quantiser steps, i.e. for new lines:

```
freqRelStep = exp(log(fmax/fmin)/(2^fbits))
```

and for continued lines:

```
freqRelStep = 2*dfmax/(2^dfbits)*prev_freq[linePred[i]]/freq[i].
```

For a continued line, the number of frequency enhancement bits fEnhbits[i] is determined by the lineFreq value of its initial „new line“.

For both line types the phase is decoded from the enhancement bitstream:

```
phaseEnh[i] = 2*pi*(linePhase[i]+0.5)/(2^phasebits)-pi
```

4.2.4 Tables

4.3 Noise decoder

4.3.1 Tool description

This tool decodes the noise parameters transmitted in the bitstream.

4.3.2 Definitions

noiseFlag	flag indicating noise data in current frame	
numNoiseParaCod:	number of noise parameters - 4	
numNoisePara:	number of noise parameters	
noiseEnvFlag:	flag indicating noise envelope data	
noiseNorm:	coded noise normalisation factor	
noisePara[i]:	coded noise parameters	
noiseEnvTmax:	coded noise envelope parameter in current frame:	time of maximum
noiseEnvRatk:		attack rate
noiseEnvRdec:		decay rate

4.3.3 Decoding Process

4.3.3.1 Basic decoder

If the „noiseFlag“ is set and thus NOISEbasicPara() data is available in the current frame, the parameters of the „noise“ signal component are decoded and dequantised as follows:

First the noise scaling factor is calculated:

```
nNorm = exp(log(2)/2 * (noiseNorm+1)) / 8
```

Then the noise parameters are dequantised:

```
nPara[0] = (noisePara[0]+0.5) * nNorm
for (i=1; i<numNoisePara; i++)
    nPara[i] = (noisePara[i]-4+0.5) * 2*nNorm
```

If noiseEnvFlag == 1 then the noise envelope parameters noiseEnvTmax, noiseEnvRatk and noiseEnvRdec are dequantised in the same way as described in the individual line decoder.

4.3.3.2 Enhanced decoder

Since there is no enhancement data for noise components, there is no specific enhanced decoding mode for noise parameters. If noise is to be synthesised with enhancement data present for the other components, the basic noise parameter decoder can be used. However it has to be noted that if the HILN decoder is used as a core in a scalable coder no noise signal must be synthesised for the signal which is given to the enhancement decoder.

4.3.4 Tables

4.4 Harmonic and Individual Line synthesizer

4.4.1 Tool description

This tool synthesises the audio signal according to the harmonic and individual line parameters decoded by the corresponding decoder tools. It includes the combination of the harmonic and individual lines, the basic synthesiser and the enhanced synthesiser.

4.4.2 Definitions

HILNcontMode	additional decoder line continuation
totalNumLine	total number of lines to be synthesised (individual and harmonic)
freq[i]:	frequency of individual line i (in Hz)
hLineFreq[i]:	frequency of harmonic line i (in Hz)
ampl[i]:	amplitude of individual line i
hLineAmpl[i]:	amplitude of harmonic line i
prev_freq[k]:	frequency of individual line k in previous frame (in Hz)
prev_ampl[k]:	amplitude of individual line k in previous frame
prev_phi[i]:	end phase of individual line k in previous frame (in rad)
env(t):	amplitude envelope
a(t):	momentary amplitude of line being synthesised
phi(t):	momentary frequency of line being synthesised
x(t):	synthesised output signal

4.4.3 Synthesis Process

4.4.3.1 Combination of harmonic and individual lines

For the synthesis of the harmonic lines the same synthesis technique as for the individual lines is used.

If no harmonic component is decoded for the following steps numHarmLine has to be set to zero.

Otherwise the parameters of the harmonic lines are appended to the list of individual line parameters as decoded by the individual line decoder:

```
for (i=0; i<numHarmLine; i++) {
        freq[numLine+i] = hLineFreq[i]
        ampl[numLine+i] = hLineAmpl[i]
        linePred[numLine+i] = (harmPred) ? prevNumLine+i+1 : 0
        lineEnvFlag[numLine+i] = harmEnv
}
```

Thus the total number of line parameters passed to the „individual line“ synthesiser is:

```
totalNumLine = numLine + numHarmLine
```

Depending on the value of HILNcontMode it is possible to connect lines in adjacent frames in order to avoid phase discontinuities in the case of transitions to and from harmonic lines (HILNcontMode == 0) or additionally from individual lines to individual lines for which the continue bit lineContFlag in the bitstream was not set by the encoder (HILNcontMode == 1).

For each line i = 0 .. totalNumLine-1 of the current frame that has no predecessor, the best-fitting line j of the previous frame having no successor and with the combination meeting the requirements specified by HILNcontMode as described above is determined by maximising the following measure q:

```
df = freq[i] / prev_freq[j]
df = max(df, 1/df)
```

```
da = ampl[i] / prev_ampl[j]
da = max(da, 1/da)
q = (1 - (df-1)/(dfCont-1)) * (1 - (da-1)/(daCont-1))
```

where dfCont = 1.05 and daCont = 4 are the maximum relative frequency and amplitude changes permitted. For additional line continuations determined in this way, the line predecessor information is updated:

```
linePred[i] = j+1
```

If there is not at least one predecessor with df < dfCont and da < daCont linePred[i] remains unchanged.

For the enhanced synthesiser, the enhanced harmonic (up to maximum of 10) and individual line parameters are combined as follows:

```
for (i=0; i<min(10,numHarmLine); i++) {
    freqEnh[numLine+i] = hLineFreqEnh[i]
    amplEnh[numLine+i] = hLineAmplEnh[i]
    phaseEnh[numLine+i] = hLinePhaseEnh[i]
    linePred[numLine+i] = (harmPred) ? prevNumLine+i+1 : 0
    lineEnvFlag[numLine+i] = harmEnv
}
```

Thus the total number of line parameters passed to the enhanced „individual line“ synthesiser, if the HILN decoder is used as a core in a scalable coder, is:

```
totalNumLine = numLine + min(10,numHarmLine)
```

Since phase information is available for all of these lines , no line continuation is introduced for the enhanced synthesiser.

4.4.3.2 Basic synthesiser

The basic synthesiser reconstructs one frame of the audio signal. Since the line parameters encoded in a bitstream frame are valid for the middle of the corresponding frame of the audio signal, the Individual Lines Synthesiser generates the one-frame long section of the audio signal that starts in the middle of the previous frame and that ends in the middle of the current frame.

In the following, the calculation of the synthesised output signal

```
x(t) = x((n+0.5)*(T/N))        0 <= t < T        0 <= n < N
```

is described. T is the frame length in seconds, N is the number of samples in a frame and N/T is the sampling frequency in Hz. Some parameters of the previous frame (names starting with „previous“) are taken out of a frame to frame memory which has to be reset before decoding the first frame of a bitstream.

First the envelope function previousEnv(t) and env(t) of the previous and current frame are calculated according to the following rules:

If envFlag = 1 then the envelope function env(t) is derived from the envelope parameters t_max, r_atk, and r_dec. With T being the frame length, env(t) is calulated for -T/2 <= t <= 3/2*T:

```
env(t) = max(0,1-(t_max-t/T)*r_atk)        for t/T <= t_max
env(t) = max(0,1-(t/T-t_max)*r_dec)        for t/T = t_max
```

If envFlag = 0 then a constant envelope function env(t) is used:

```
env(t) = 1
```

Accordingly previousEnv(t) is calculated from the parameters previousT_max, previousR_atk, previousR_dec and previousEnvFlag.

The envelope parameters transmitted in case of envFlag == 1 are valid for the harmonic lines as well as for the individual lines. Thus the envelope functions always must be generated, even if all lineEnvFlag[i] == 0.

Before the synthesis is performed, the accumulator x(t) for the synthesised audio signal is cleared:

```
x(t) = 0        for 0 <= t <T
```

The lines i continuing from the previous frame to the current frame

```
all i=0 .. totalNumLine-1 with lineContFlag[i] = 1
```

are synthesised as follows for 0 <= t <T:

```
k = linePred[i]
ap(t) = previousAmpl[k]
if previousEnvFlag[k] = 1 then
        ap(t) *= previousEnv(t+T/2)
ac(t) = ampl[k]
if envFlag[i] = 1 then
        ac(t) *= env(t-T/2)
short_x_fade =
        (previousEnvFlag && !(previousR_atk < 5 && (previousT_max > 0.5 || previousR_dec < 5)))
        || (envFlag && !(r_dec < 5 && (t_max < 0.5 || r_atk < 5)))
if short_x_fade = 1 then
        a(t) = ap(t)                                    for t < 7/16*T
        a(t) = ap(t) + (ac(t)-ap(t))*(t/T-7/16)*8       for 7/16*T < t < 9/16*T
        a(t) = ac(t)                                    for t > 9/16*T
else
        a(t) = ap(t) + (ac(t)-ap(t))*t/T
phi[i](t) = previousPhi[k]+
        2*pi*previousFreq[k]*t+
        2*pi*(freq[i]-previousFreq[k])/(2*T)*t^2
x(t) += a(t)*sin(phi[i](t))
```

The lines i starting in the current frame

```
all i=0 .. totalNumLine-1 with lineContFlag[i] = 0
```

are synthesised as follows for 0 <= t <T:

```
if (envFlag && !(r_dec < 5 && (t_max < 0.5 || r_atk < 5))) then
        fade_in(t) = 0                                  for t < 7/16*T
        fade_in(t) = 0.5 - 0.5*cos((8*t/T-7/2)*pi)      for 7/16*T < t < 9/16*T
        fade_in(t) = 1                                  for t > 9/16*T
else
        fade_in(t) = 0.5-0.5*cos(t/T*pi)
a(t) = fade_in(t)*ampl[i]
if envFlag[i] = 1 then
        a(t) *= env(t-T/2)
start_phi[i] = random(2*pi)
phi[i](t) = start_phi[i] + 2*pi*freq[i]*t
x(t) += a(t)*sin(phi[i](t))
```

random(x) is a function returning a random number with uniform distribution in the interval

```
0 <= random(x) < x.
```

The lines k ending in the previous frame

```
all k=0 .. previousTotalNumLine-1 with prevLineContFlag[k] = 0
```

are synthesised as follows for 0 <= t <T:

```
if (previousEnvFlag && !(previousR_atk < 5 && (previousT_max > 0.5 || previousR_dec < 5)))
        fade_out(t) = 1                                 for t < 7/16*T
        fade_out(t) = 0.5 + 0.5*cos((8*t/T-7/2)*pi)     for 7/16*T < t < 9/16*T
        fade_out(t) = 0                                 for t > 9/16*T
else
        fade_out(t) = 0.5+0.5*cos(t/T*pi)
a(t) = fade_out(t)*previousAmpl[k]
if previousEnvFlag[k] = 1 then
        a(t) *= previousEnv(t+T/2)
phi(t) = previousPhi[k]+2*pi*previousFreq[k]*t
x(t) += a(t)*sin(phi(t))
```

Parameters needed in the following frame are stored in the frame to frame memory:

```
previousEnvFlag = envFlag
previousT_max = t_max
previousR_atk = r_atk
```

```
previousR_dec = r_dec
previousTotalNumLine = totalNumLine
all i=0 .. totalNumLine-1
        previousFreq[i] = freq[i]
        previousAmpl[i] = ampl[i]
        previousPhi[i] = fmod(phi[i](T),2*pi)          (2*pi modulus of the end phase of line i)
```

Due to the phase continuation of this decoder implementation, the speed of the decoded signal can be changed by simply changing the frame length without any other modifications. The relation of the encoder frame length and the selected decoder frame length directly corresponds to a speed factor.
In a similar way, the pitch of the decoded signal can be varied without affecting the frame length and without causing phase discontinuities. The pitch change is performed by simply multiplying a factor to each frequency parameter before it is used in the synthesis.

4.4.3.3 Enhanced synthesiser

The enhanced synthesiser is based on the basic synthesiser but evaluates also the line phases for reconstructing one frame of the audio signal. Since the line parameters encoded in a bitstream frame and the corresponding enhancement frame are valid for the middle of the corresponding frame of the audio signal, the Individual Lines Synthesiser generates the one frame long section of the audio signal that starts in the middle of the previous frame and and ends in the middle of the current frame.
Some parameters of the previous frame (names starting with „previous“) are taken out of a frame to frame memory which has to be reset before decoding the first frame of a bitstream.

First the envelope functions previousEnv(t) and env(t) of the previous and current frame are calculated according to the following rules:

If envFlag = 1 then the envelope function env(t) is derived from the envelope parameters t_maxEnh, r_atkEnh, and r_decEnh. With T being the frame length, env(t) is calulated for -T/2 <= t <= 3/2*T:

```
        env(t) = max(0,1-(t_maxEnh-t)/T*r_atkEnh)          for t <= t_maxEnh
        env(t) = max(0,1-(t-t_maxEnh)/T*r_decEnh)          for t >= t_maxEnh
```

If envFlag = 0 then a constant envelope function env(t) is used:

```
        env(t) = 1
```

Accordingly previousEnv(t) is calculated from the parameters previousT_maxEnh, previousR_atkEnh, previousR_decEnh and previousEnvFlag.

The envelope parameters transmitted in case of envFlag == 1 are valid for the harmonic lines as well as for the individual lines. Thus the envelope functions always must be generated, even if all lineEnvFlag[i] == 0.

Before the synthesis is performed, the accumulator x(t) for the synthesised audio signal is cleared:

```
        x(t) = 0        for 0 <= t < T
```

All lines i in the in the current frame

```
        all i =0 .. totalNumLine-1
```

are synthesised as follows for 0 <= t < T:

```
        if (envFlag && !(r_decEnh < 5 && (t_maxEnh < 0.5 || r_atkEnh < 5))) then
                fade_in(t) = 0                                  for t < 7/16*T
                fade_in(t) = 0.5 - 0.5*cos((8*t/T-7/2)*pi)      for 7/16*T < t < 9/16*T
                fade_in(t) = 1                                  for t > 9/16*T
        else
                fade_in(t) = 0.5-0.5*cos(t/T*pi)
        a(t) = fade_in(t)*amplEnh[i]
        if envFlag[i] = 1 then
                a(t) *= env(t-T/2)
        phi(t) = 2*pi*freqEnh[i]*(t-T)+phaseEnh[i]
        x(t) += a(t)*sin(phi(t))
```

The lines k in the previous frame

```
        all k=0 .. previousTotalNumLine-1
are synthesised as follows for 0 <= t < T:
        if (previousEnvFlag &&
                        !(previousR_atkEnh < 5 && (previousT_maxEnh > 0.5 || previousR_decEnh < 5)))
                fade_out(t) = 1                                 for t < 7/16*T
                fade_out(t) = 0.5 + 0.5*cos((8*t/T-7/2)*pi)     for 7/16*T < t < 9/16*T
                fade_out(t) = 0                                 for t > 9/16*T
        else
                fade_out(t) = 0.5+0.5*cos(t/T*pi)
        a(t) = fade_out(t)*previousAmpl[k]
        a(t) = fade_out(t)*previousAmplEnh[k]
        if previousEnvFlag[k] = 1 then
                a(t) *= previousEnv(t+T/2)
        phi(t) = 2*pi*previousFreqEnh[k]*t+previousPhaseEnh[i]
        x(t) += a(t)*sin(phi(t))
```

Parameters needed in the following frame are stored in the frame to frame memory:

```
previousEnvFlag = envFlag
previousT_maxEnh = t_maxEnh
previousR_atkEnh = r_atkEnh
previousR_decEnh = r_decEnh
previousTotalNumLine = totalNumLine
all i=0 .. totalNumLine-1
        previousFreqEnh[i] = freqEnh[i]
        previousAmplEnh [i] = amplEnh[i]
        previousPhaseEnh [i] = phaseEnh [i]
```

4.4.4 Tables

4.5 Noise synthesizer

4.5.1 Tool description

This tool synthesises the noise part of the output signal based on the noise parameters decoded by the noise decoder.

4.5.2 Definitions

numNoisePara:	number of noise parameters
noisebw:	noise bandwidth (in Hz)
N:	frame length for noise synthesis in samples
M:	length of IDCT-generated spectrum
nPara[n]:	decoded noise parameters
noiseWin[i]:	window for noise overlap-add
noiseEnv[i]	envelope for noise component
n[i]	synthesised noise samples

4.5.3 Synthesis Process

4.5.3.1 Basic synthesiser

If noise parameters are transmitted for the current frame, a noise signal with a spectral shape as described by the noise parameters decoded from the bitstream is synthesised and added to the audio signal generated by the harmonic and individual line synthesiser.

To calculate the uniformly sampled spectral shape of the noise, an inverse DCT (Discrete Cosine Transform) is applied to the noise parameters to obtain the spectral shape of the noise:

```
nPara[0] *= sqrt(0.5)
for (j=0; j<min(N,M), j++) {
        y[j] = 0
        for (i=0; i<numNoisePara, i++)
                y[j] += nPara[i]*sqrt(2/M)*cos(pi*i*(j+0.5)/M)
}
for (j=0; j<M, j++)
        y[j] = max(0,y[j])
for (j=M; j<N, j++)
        y[j] = 0
```

where N is the frame length in samples and M is the length of the IDCT generated spectrum. M is calculated as follows:

```
M = N * noisebw / (f_sample/2)
```

where f_sample = N/T is the sampling frequency of the signal being synthesised.

A vector p[j] of N random values equally distributed in the interval [0,2*pi[is used to obtain the complex frequency domain representation of the noise:

```
for (j=0; j<N; j++) {
        re[j] = y[j]*cos(p[j])
        im[j] = y[j]*sin(p[j])
        re[2*N-1-j] = re[j]
        im[2*N-1-j] = -im[j]
}
```

Now the real time domain noise signal n[i] is calculated by applying an inverse ODFT (Odd Discrete Fourier Transform) of length 2*N to re[j], im[j]. If the noiseEnvFlag is set, the noise envelope noiseEnv[i] is generated in the same way as for the „individual line“ synthesiser (see Section 4.4.3.2) and the noise signal is windowed with noiseEnv[i]:

```
for (i=0; i<2*N; i++) {
        n[i] = 0
        for (j=0; j<2*N; j++) {
                pf = pi*i*(j+0.5)/N
                n[i] += re[j]*cos(pf)-im[i]*sin(pf)
        }
        n[i] /= 512
        if(noiseEnvFlag)
                n[i] *= noiseEnv[i]
}
```

For smooth cross-fade of the noise signal at boundary between two adjacent frames, the following window is used for this overlap-add operation:

```
noiseWin[i] = 0                                 if i < N*3/8
noiseWin[i] = sin(pi/2 * (i+0.5)/(N*2/8))       if N*3/8 < i < N*5/8
noiseWin[i] = 1                                 if i > N*5/8
prev_noiseWin[i] = noiseWin[N-i]
```

Finally the noise signal is added to the previously synthesised signal x[i] and the second half of the generated noise signal is stored in a frame to frame memory for overlap-add:

```
for (i=0; i<N; i++) {
        x[i] += n[i]*noiseWin[i] + prev_n[i]*prev_noiseWin[i]
        prev_n[i] = n[N+i]
}
```

Pitch and speed change functionalities are implemented similar as in the basic line synthesiser: To change the pitch of the noise, the noise bandwidth noisebw must be multiplied by the pitch change factor. To change the speed, an correspondingly increased or decreased frame length N is used for the synthesis.

4.5.3.2 Enhanced synthesiser

Since there is no enhancement data for noise components, there is no specific enhanced synthesiser mode for noise components. If noise is to be synthesised with enhancement data present for the other components, the basic noise synthesiser decoder can be used. However it has to be noted that if the HILN decoder is used as a core in a scalable coder no noise signal must be synthesised for the signal which is given to the enhancement decoder.

4.5.4 Tables

5 Integrated parametric coder

The integrated parametric coder can operate in the following modes:

PARAmode	Description
0	HVXC only
1	IL only
2	switched HVXC / HILN
3	mixed HVXC / HILN

PARAmodes 0 and 1 represent the fixed HVXC and HILN modes. PARAmode 2 permits automatic switching between HVXC and HILN depending on the current input signal type. In PARAmode 3 the HVXC and HILN coders can be used simultaneously and their output signals are added (mixed) in the decoder.

The integrated parametric core uses a frame length of 40 ms and a sampling rate of 8 kHz and can operate at 2025 bit/s or any higher bitrate. Operation at 4 kbit/s or higher is suggested.

5.1 Integrated Parametric Decoder

For the „HVXC only“ and „HILN only“ modes the parametric decoder is not modified.

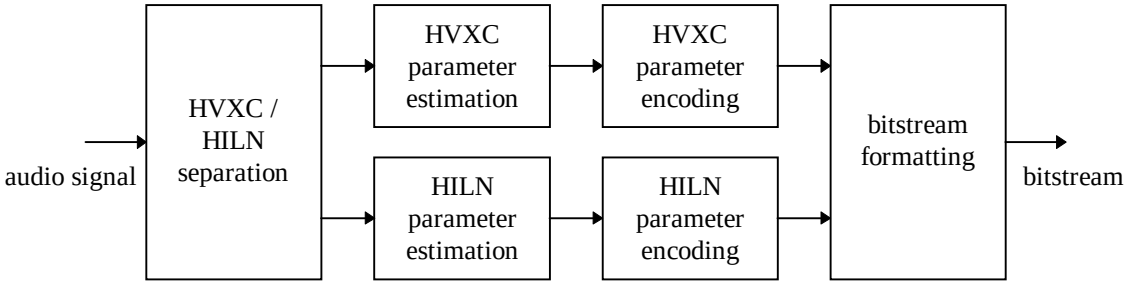
In „switched HVXC / HILN“ and „mixed HVXC / HILN“ modes both HVXC and HILN decoder tools are operated alternatively or simultaneously according to the PARAswitchMode or PARAmixMode of the current frame. To obtain proper time alignment of both HVXC and HILN decoder output signals before they are added, the difference between HVXC and HILN decoder delay has to be compensated with a FIFO buffer:
If HVXC is used in the low delay decoder mode, its output must be delayed for 100 samples (i.e. 12.5 ms).
If HVXC is used in the normal delay decoder mode, its output must be delayed for 80 samples (i.e. 10 ms).

To avoid hard transitions at frame boundaries when the HVXC or HILN decoders are switched on or off, the respective decoder output signals are faded in and out smoothly. For the HVXC decoder a 20ms linear fade is applied when it is switched on or off. The HILN decoder requires no additional fading because of the smooth synthesis windows utilised in the HILN synthesiser. It is only necessary to operate the HILN decoder with no new components for the current frame (i.e. force numLine = 0, harmFlag = 0, noiseFlag = 0) if the current bitstream frame contains no „HILNframe()“.

Annex A Informative Tool Descriptions

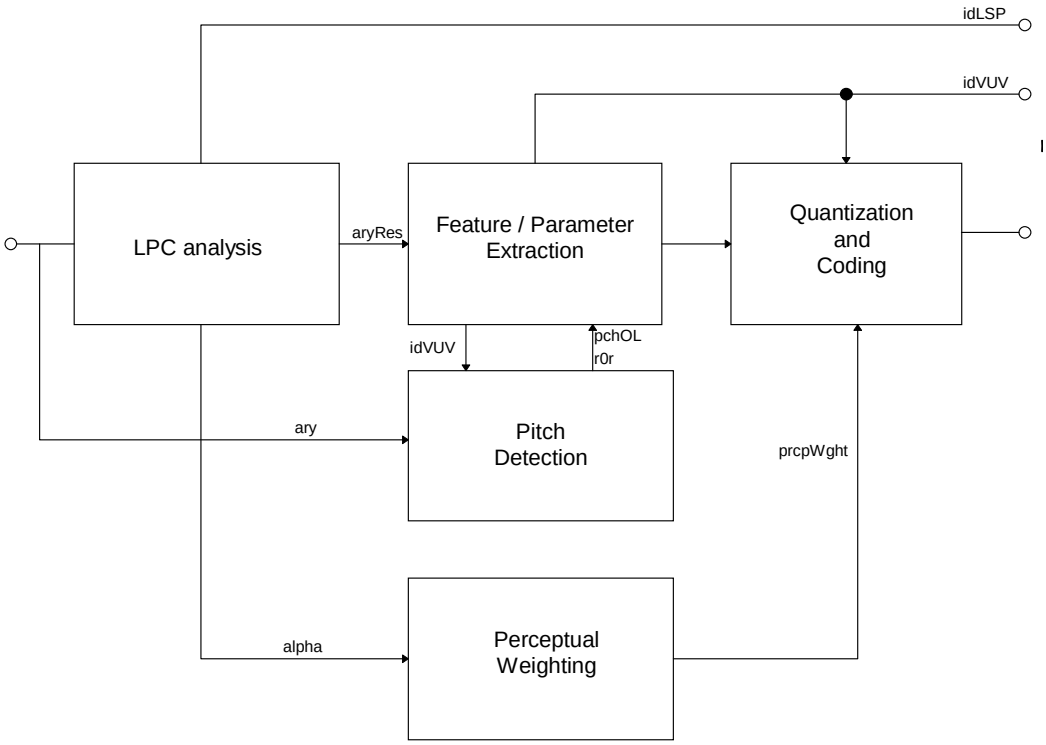
1 Parametric Encoder Core

The following figure shows a general block diagram of a parametric encoder. First the input signal is separated into the two parts which are coded by HVXC and by HILN tools. This can be done manually or automatically. Currently automatic switching between speech and music signals is supported (see Section 5), allowing the use of HVXC for speech and HILN for music. For both HVXC and HILN parameter estimation and parameter encoding can be performed. A common bitstream formatter allows operation either in HVXC only, HILN only or also in combined modes, i.e. switched or mixed mode.

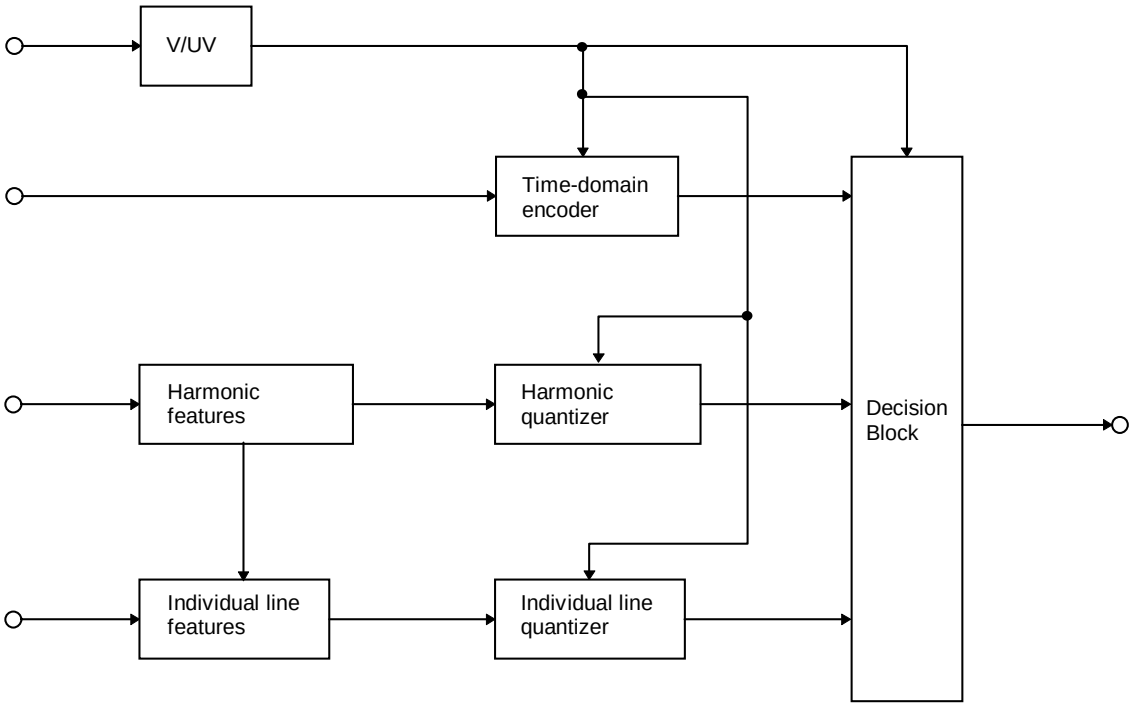


General block diagram of parametric encoder

Block diagram of the parameter based encoder core



More detailed diagram for the Feature/Parameter Extraction and Quantization&Coding Module



2 HVXC Encoder Tools

2.1 Overview of Encoder tools

Fig.2.1 shows the overall structure of the encoder. Speech input at a sampling rate of 8kHz is formed into frames with a length and interval of 256 and 160 samples, respectively. LPC analysis is carried out using windowed input data over one frame. LPC residual signals are computed by the inverse filtering of input data using quantized and interpolated LSP parameters. The residual signals are then fed into the pitch and spectral magnitude estimation block, where the spectral envelopes for LPC residual are estimated in the same manner as in the MBE coder except that only a two-bit V/UV decision is used per frame. The spectral envelope for voiced segment is then vector quantized with weighted distortion measure. For unvoiced segment, closed loop search for the vector excitation coding is carried out. Detailed configurations are described below.

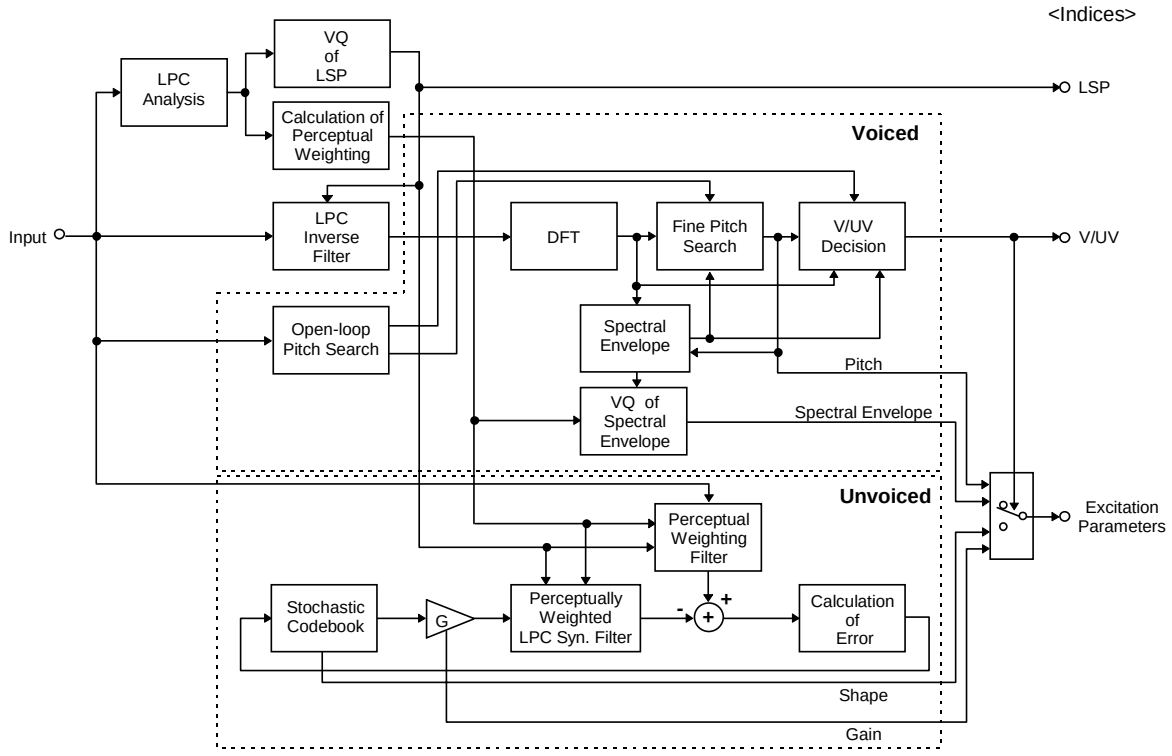


Fig.2.1 Blockdiagram of the HVXC encoder

2.2 Normalization

2.2.1 Tool description

The normalization process is composed of three operations, that is, LPC analysis, LSP parameter quantization, and inverse filtering. These operations are described below.

2.2.2 Definition

2.2.3 Normalization process

2.2.3.1 LPC analysis

10th order LPC coefficients are computed for every frame, using Hamming-windowed input signals by autocorrelation method.

2.2.4 LSP quantization

The same LSP quantizer as that of the narrowband CELP is used.

LPC coefficients are first converted to Line Spectral Pair (LSP) parameters. LSP parameters are then quantized with a Vector Quantization (VQ) respectively. In case of the base layer, there are two methods for quantizing the LSPs as described in decoding section; a two-stage VQ without interframe prediction, and a combination of the VQ and the inter-frame predictive VQ. At the encoding process, both methods are tried to quantize the LSPs and determined which method should be applied by comparing the quantization error. The quantization error is calculated as a weighted euclidean distance.

In the case of enhancement layer, a 10-dimensional vector quantizer, which has 8 bits codebook, is added to the bottom of the current LSP quantizer scheme of 2.0kbps coder. The bit rate of LSPs is increased from 18bits/20msec to 26bits/20msec.

The encoding process of the base layer is as follows.

The weighting coefficients (*w[]*) are,

$$w[i] = \begin{cases} \dfrac{1}{lsp[0]} + \dfrac{1}{lsp[1] - lsp[0]} & (i = 0) \\ \dfrac{1}{lsp[i] - lsp[i-1]} + \dfrac{1}{lsp[i+1] - lsp[i]} & (0 < i < Np - 1) \\ \dfrac{1}{lsp[Np-1] - lsp[Np-2]} + \dfrac{1}{1.0 - lsp[Np-1]} & (i = Np - 1) \end{cases}$$

where *Np* is the LP analysis order and *lsp[]*s are the converted LSPs.

```
w_fact = 1.;
for(i=0;i<4;i++) w[i] *= w_fact;
for(i=4;i<8;i++) {
    w_fact *= .694;
    w[i] *= w_fact;
}
for(i=8;i<10;i++) {
    w_fact *= .510;
    w[i] *= w_fact;
}
```

The first stage quantizer is the same for each quantization method. The LSPs are quantized by using a vector quantizer and corresponding index is stored in nVq1[0] . In order to carry out delayed decision, plural indices are stored as candidates for the second stage. The quantization error in the first stage *err1*[] is given by:

$$err1[n] = \sum_{i=0}^{\dim-1} \left\{ \left(lsp[sp+i] - lsp\_tbl[n][m][i] \right)^2 \cdot w[sp+i] \right\} \qquad n = 0$$

where *n* is the split vector number, *m* is the index of the candidate split vector , sp is the starting LSP order of the n-th split vector and dim is the dimension of the n-th split vector. (*lsp_tbl[][][]* is shown in Annex C)

Starting order and dimension of the first stage LSP vector

Split vector number: n	Starting LSP order: sp	Dimension of the vector: dim
0	0	10

In the second stage, above-mentioned two quantization methods, which are two-split vector quantizer, are applied respectively. Total quantization errors in the second stage are calculated for all combinations of the first stage candidates and the second stage candidates and the one which has the minimum error is selected. As a result, indices of the first stage are determined and corresponding indices and signs for the second stage are stored in nVq1[1] and nVq1[2]. The flag which indicates the selected quantization method is also stored in nVq1[3]. The quantization error in the second stage *err2_total* is given by:

VQ without interframe prediction:

$$err2\_total = err2[0] + err2[1]$$

$$err2[n] = \sum_{i=0}^{\dim-1} \left\{ \left(lsp\_res[sp+i] - sign[n] \cdot d\_tbl[n][m][i] \right)^2 \cdot w[sp+i] \right\} \qquad n = 0,1$$

$$lsp\_res[sp+i] = lsp[sp+i] - lsp\_first[sp+i]$$

where *lsp_first*[] is the quantized LSP vector of the first stage, *n* is the split vector number, *m* is the index of the candidate split vector , sp is the starting LSP order of the n-th split vector and dim is the dimension of the n-th split vector. (*d_tbl[][][]* is shown in Annex C)

VQ with interframe prediction:

$$err2\_total = err2[0] + err2[1]$$

$$err2[n] = \sum_{i=0}^{\dim-1} \left\{ \left(lsp\_pres[sp+i] - sign[n] \cdot pd\_tbl[n][m][i] \right)^2 \cdot w[sp+i] \right\} \qquad n = 0,1$$

$$lsp\_pres[sp+i] = lsp[sp+i] - \left\{ (1 - ratio\_predict) \cdot lsp\_first[sp+i] + ratio\_predict \cdot lsp\_previous[sp+i] \right\}$$

where *lsp_first*[] is the quantized LSP vector of the first stage, *n* is the split vector number, *m* is the index of the candidate split vector , sp is the starting LSP order of the n-th split vector, dim is the dimension of the n-th split vector and ratio_predict=0.7. (*pd_tbl[][][]* is shown in Annex C)

Starting order and dimension of the second stage LSP vector

Split vector number: n	Starting LSP order: sp	Dimension of the vector: dim
0	0	5
1	5	5

The quantized LSPs *lsp_current[]* are stabilized in order to ensure stability of the LPC synthesis filter which is derived from the quantized LSPs. The quantized LSPs are arranged in ascending order, having a minimum distance between adjacent coefficients.

```
for(i=0;i<LPCORDER;i++) {
   if(lsp_current[i] < min_gap) lsp_current[i] = min_gap;
}
for(i=0;i<LPCORDER-1;i++) {
   if(lsp_current[i+1]-lsp_current[i] < min_gap) {
      lsp_current[i+1] = lsp_current[i]+min_gap;
   }
}
for(i=0;i<LPCORDER;i++) {
   if(lsp_current[i] > 1-min_gap) lsp_current[i] = 1-min_gap;
}
for(i=LPCORDER-1;i>0;i--) {
   if(lsp_current[i]-lsp_current[i-1] < min_gap) {
      lsp_current[i-1] = lsp_current[i]-min_gap;
   }
}

for(i=0;i<LPCORDER;i++){
   qLsp[i] = lsp_current[i];
}
```

where min_gap = 4.0/256.0

After the LSP encoding process, the current LSPs have to be stored in memory, since they are used for prediction at the next frame.

```
for (i=0;i<LPCORDER;i++) {
     lsp_previous[i] = lsp_current[i];
}
```

It must be noted that the stored LSPs *lsp_previous[]* must be initialized as described below when the whole of the encoder is initialized.

```
for (i=0;i<LPCORDER;i++) {
     lsp_previous[i] = (i+1) / (LPCORDER+1);
}
```

The third stage for the enhancement layer (4.0kbps) has 10-dimensional VQ structure.
The error between the quantized version of the base layer and the original version is quantized.

After the quantization, the LSPs of the enhancement layer qLsp[] are stabilized again.

```
for(i = 0; i < 2; i++)
{
   if(qLsp[i + 1] - qLsp[i] < 0)
   {
       tmp = qLsp[i + 1];
       qLsp[i + 1] = qLsp[i];
       qLsp[i] = tmp;
   }

   if(qLsp[i + 1] - qLsp[i] < THRSLD_L)
   {
       qLsp[i + 1] = qLsp[i] + THRSLD_L;
   }
}


for(i = 2; i < 6; i++)
{
   if(qLsp[i + 1] - qLsp[i] < THRSLD_M)
   {
       tmp = (qLsp[i + 1] + qLsp[i]) / 2.0;
       qLsp[i + 1] = tmp + THRSLD_M / 2.0;
       qLsp[i] = tmp - THRSLD_M / 2.0;
   }
}


for(i = 6; i < LPCORDER - 1; i++)
{
   if(qLsp[i + 1] - qLsp[i] < 0)
   {
       tmp = qLsp[i + 1];
       qLsp[i + 1] = qLsp[i];
       qLsp[i] = tmp;
   }

   if(qLsp[i + 1] - qLsp[i] < THRSLD_H)
   {
       qLsp[i] = qLsp[i + 1] - THRSLD_H;
   }
}
```

where, THRSLD_L=0.020, THRSLD_M=0.020 and THRSLD_H=0.020

1st stage	10 LSP VQ	5bits
2nd stage	(5+5)LSP VQ	(7+5+1)bits
3rd stage	10LSP VQ	8bits

2.2.5 LPC inverse filter

LSPs are converted to alpha parameters to form a LPC inverse filter in a direct form. LPC residual signals are then computed by inverse filtering the input signal. Only the current frame's quantized LPC is used without any interpolation for the inverse filtering to compute the LPC residual signal. The residual signal is 256 point Hamming windowed to compute the power spectrum. The transfer function of the inverse filter is,

$$A(z) = \sum_{n=0}^{P} \alpha_n z^{-n}$$

where $P = 10$ and α_0 =1. LPC coefficients α_n stay unchanged during the computation of residual samples of one frame length (256 samples).

2.3 Pitch Estimation

2.3.1 Tool description

To obtain the first estimation of the pitch lag value, the autocorrelation values of the LPC residual signals are computed. Based on the lag values which give the peaks in autocorrelation, open loop pitch is estimated. Pitch tracking is carried out in the process of pitch estimation so that the estimated pitch gets more reliable.

2.3.2 Pitch estimation process

In the low delay mode encoder, pitch tracking is conducted using only current and past frames to keep encoder delay 26 ms. When the normal delay mode is used, pitch tracking uses one frame ahead and the encoder delay becomes 46 ms.

2.3.3 Pitch tracking

For the low delay mode of HVXC, pitch tracking algorithm does not use the pitch value of the future (look ahead) frame. The pitch tracking algorithm operates based on reliable pitch "rblPch", V/UV decision of the previous frame "prevVUV", and past/current pitch parameters.
The basic operation is as follows:

- When prevVUV != 0 and rblPch !=0, pitch is tracked based on rblPch. However, the pitch value of the previous frame has higher priority.
- When prevVUV = 0 and rblPch !=0, pitch is tracked based on rblPch.
- When prevVUV !=0 and rblPch = 0, tracking is conducted based on the pitch of the previous frame.
- When prevVUV = 0 and rblPch = 0, current pitch parameter is simply used.

prevVUV !=0 represents Voiced, and prevVUV = 0 represents Unvoiced status respectively.
By this strategy, pitch tracking is carried out without any look ahead. Source code of the pitch tracking is shown below.

```
typedef struct
{
   float   pitch;          /* pitch */
   float   prob;           /* 1st peak devided by 2nd peak of autocorrelation */
   float   r0r;            /* 1st peak of autocorrelation  - modified */
   float   rawR0r; /* 1st peak of autocorrelation - raw */
   float   rawPitch;       /* pitch with no tracking */
}
NgbPrm;

static int NearPitch(
float   p0,
float   p1,
float   ratio)
{
```

```
    return((p0 * (1.0 - ratio) < p1) && (p1 < p0 * (1.0 + ratio)));
}

static float TrackingPitch(
NgbPrm *crntPrm,          /* current parameter */
NgbPrm *prevPrm,          /* previous  parameter */
int                *scanlimit,        /* Number of autocorrelation peaks */
float              *ac,               /* Autocorrelation */
int        peakPos[PEAKMAX],     /* Position of autocorrelation peaks */
int        prevVUV)              /* V/UV of ptrevious frame */
{
   float   kimete;
   float   pitch;
   int             i;
   static float    prevRawp= 0.0;
   int             st0, st1, st2;
   float   stdPch;
   static float    rblPch = 0.0;
   static float    prevRblPch = 0.0;

   rblPch = global_pitch;
   if(prevVUV != 0 && rblPch != 0.0)
   {
          st0 = Ambiguous(prevPrm->pitch, rblPch, 0.11);
          st1 = Ambiguous(crntPrm->pitch, rblPch, 0.11);
          if(!(st0 || st1))
          {
             if(NearPitch(crntPrm->pitch, prevPrm->pitch, 0.2))
                    pitch = crntPrm->pitch;
             else if(NearPitch(crntPrm->pitch, rblPch, 0.2))
                    pitch = crntPrm->pitch;
             else if(NearPitch(prevPrm->pitch, rblPch, 0.2))
             {
                    if(crntPrm->r0r > prevPrm->r0r && crntPrm->prob > prevPrm->prob)
                       pitch = crntPrm->pitch;
                    else
                       pitch = prevPrm->pitch;
             }
             else
                    pitch = GetStdPch2Elms(crntPrm, prevPrm, scanlimit, peakPos, ac);
          }
          else if(!st0)
          {
             if(NearPitch(prevPrm->pitch, crntPrm->pitch, 0.2))
                    pitch = crntPrm->pitch;
             else if((gpMax * 1.2 > crntPrm->pitch) &&
                    NearPitch(prevPrm->rawPitch, crntPrm->pitch, 0.2))
                    pitch = crntPrm->pitch;
             else
                    pitch = prevPrm->pitch;
          }
          else if(!st1)
          {
             if((crntPrm->rawPitch != crntPrm->pitch) &&
                    NearPitch(crntPrm->rawPitch, prevPrm->rawPitch, 0.2))
                    pitch = crntPrm->rawPitch;
             else
                    pitch = crntPrm->pitch;
          }
```

```
            else
            {
               if(NearPitch(prevPrm->pitch, crntPrm->pitch, 0.2))
                     pitch = crntPrm->pitch;
               else
                     pitch = rblPch;
            }
      }
      else if(prevVUV == 0 && rblPch != 0.0)
      {
            st1 = Ambiguous(crntPrm->pitch, rblPch, 0.11);
            if(!st1)
               pitch = crntPrm->pitch;
            else
               pitch = rblPch;
      }
      else if(prevVUV != 0 && rblPch == 0.0)
      {
            st1 = Ambiguous(crntPrm->pitch, prevPrm->pitch, 0.11);
            if(!st1)
               pitch = GetStdPch2Elms(crntPrm, prevPrm, scanlimit, peakPos, ac);
            else
            {
               if(prevPrm->r0r < crntPrm->r0r)
                     pitch = crntPrm->pitch;
               else
                     pitch = prevPrm->pitch;
            }
      }
      else
            pitch = crntPrm->pitch;

      crntPrm->pitch = pitch;
      prevRblPch = rblPch;
      prevRawp = pitch;
      return(pitch);
}
```

2.4 Harmonic magnitudes extraction

2.4.1 Tool description

Harmonic magnitudes extraction consists of two steps, that are fine pitch search and estimation of spectral envelope. The operation of each step is described below.

2.4.2 Definition

2.4.3 Harmonic magnitudes process

2.4.3.1 Fine pitch search

Using the open loop integer pitch lag, fractional pitch value is estimated here. The step size of the fraction is 0.25. This is carried out by minimizing the error between the synthesized spectrum and original spectrum. Here, pitch value and spectral magnitudes are estimated simultaneously. This parameter is transmitted to the decoder as “Pitch”.

2.4.3.2 Estimation of spectral envelope

256 point DFT is applied to the LPC residual signals to obtain a original spectrum. Using the original spectrum, estimation of the spectral envelope is carried out in the part of the above mentioned fine pitch search. The spectral envelope is a set of spectral magnitudes estimated at each harmonic. This magnitude estimation is carried out by computing an optimal amplitude A_m using pre-defined basis function $E(j)$, and original spectrum $X(j)$. Let a_m and b_m be the indices of DFT coefficient of lower and higher boundary of the m-th band respectively, covering m-th harmonic band. The amplitude estimation error ε_m is defined as:

$$\varepsilon_m = \sum_{j=a_m}^{b_m} \left(|X(j)| - |A_m||E(j)| \right)^2$$

Solving

$$\frac{\partial \varepsilon_m}{\partial |A_m|} = 0$$

gives

$$|A_m| = \frac{\sum_{j=a_m}^{b_m} |X(j)||E(j)|}{\sum_{j=a_m}^{b_m} |E(j)|^2}$$

This value is defined as spectral magnitude. Basis function $E(j)$ can be obtained by DFT of 256 point Hamming window.

2.5 Perceptual weighting

2.5.1 Tool description

Frequency response of a perceptual weighting filter is computed which is for the use of weighted vector quantization of the harmonic spectral envelope. Here, a perceptual weighting filter is derived from liner predictive coefficients α_n. The transfer function of the perceptual weighting filter is;

$$w(z) = \frac{\sum_{n=0}^{P} \alpha_n A^n z^{-n}}{\sum_{n=0}^{P} \alpha_n B^n z^{-n}}$$

where $A = 0.9$ $B = 0.4$ could be used. In this tool, the frequency response of the LPC synthesis filter $h(z)$ is also computed so that it could be incorporated where,

$$h(z) = \frac{1}{\sum_{n=0}^{P} \alpha_n z^{-n}}$$

In this tool, frequency response of $w(z)h(z)$ is computed and outputed as an array *per_weight, which could be used as diagonal components of the weighting matrices WH.

2.6 Harmonic VQ encoder

2.6.1 Tool description

The harmonic VQ encoding process consists of two steps, that is, dimension conversion, and vector quantization of residual vectors. The operations of the each step are given below

2.6.2 Encoding process

2.6.2.1 Dimension converter

The number of points which composes the spectral envelope varies depending on pitch values, since the spectral envelope is a set of the estimates of the magnitudes at each harmonic. The number of harmonics ranges from about 8 to 60.
In order to vector quantize the spectral envelope, the coder has to convert them to a constant number for a fixed-dimension VQ. A band-limited interpolation is used for the sampling frequency conversion to obtain the fixed-dimension spectral vectors. The number of points, which represent the shape of the spectral envelope, should be modified without changing the shape. For this purpose, a dimension converter for a spectral envelope by a combination of low pass filter and 1st order linear interpolator is used. An FIR low pass filter with 7 sets of coefficients, each set consisting of 8 coefficients, is used for the first stage 8-times over-sampling. The 7 sets of the filter coefficients are obtained by grouping 8 every coefficients from a windowed sinc, $coef[i]$, with the offsets of 1 through 7, where

$$coef[i] = \frac{\sin \pi(i-32)/8}{\pi(i-32)/8}(0.5 - 0.5\cos 2\pi i / 64) \qquad 0 \le i \le 64$$

This FIR filtering allows decimated computation, in which only the points used at the next stage are computed. They are the left and right adjacent points of the final output of the dimension converter.
At the second over-sampling stage, 1st order linear interpolation is applied to obtain the necessary output points. In this way, we get fixed-dimension (= 44) spectral vectors.

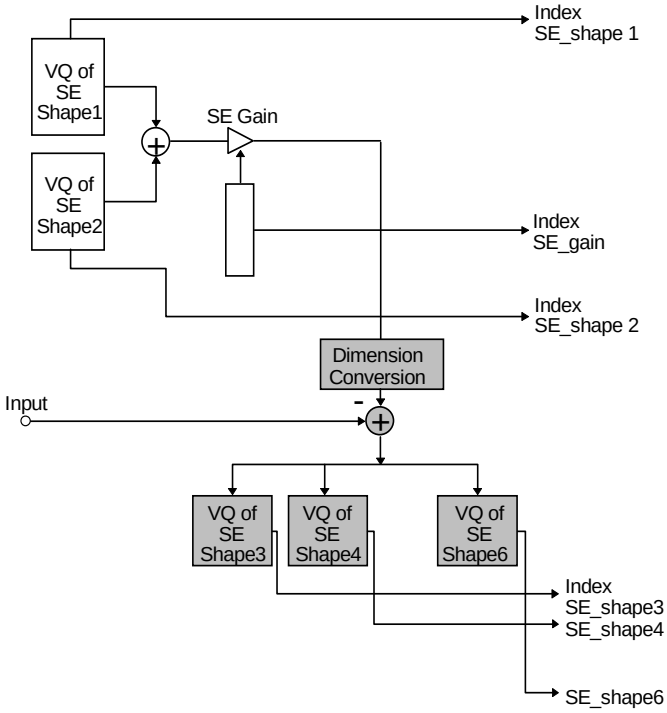
2.6.2.2 Vector quantization

A fixed-dimension (= 44) spectral vector is then quantized. In order to reduce the memory requirements and search complexity while maintaining a high performance, a Multi-Stage (2-stage) Vector Quantization (MSVQ) scheme is employed for the spectral shape together with a scalar quantizer for the gain, as shown in the figure below. The weighted distortion measure below is used for the codebook search of both shape and gain.

$$D = \left\| \mathbf{WH}(\mathbf{x} - g(\mathbf{s}_1 + \mathbf{s}_2)) \right\|^2$$

where $\boldsymbol{x}$ is a source vector, $\mathbf{s}_1$ is the output of Spectral Envelope (SE) shape1 codebook, $\mathbf{s}_2$ is the output of SE_shape2 codebook, and g is the output of the SE_gain codebook. The diagonal components of the matrices $\mathbf{H}$ and $\mathbf{W}$ are the magnitudes of the frequency response of the LPC synthesis filter and the perceptual weighting filter, respectively.

For 4.0kbps scheme, additional vector quantizers are added to the bottom of the quantizer for 2.0kbps scheme after the dimension conversion. For the quantization of harmonic spectral magnitudes, the 2.0kbps coder uses a combination of two stage shape vector quantizer and a scalar gain quantizer, where each of the shape codebooks and gain codebook is 5 bits respectively. The dimension of the shape codebooks is fixed (=44).
For 4.0kbps mode, the quantized harmonic magnitudes with fixed dimension (=44) is first converted to the dimension of original harmonic vector, which varies frame by frame. The difference between the quantized/dimension converted harmonic vector and the original harmonic vector is computed. This difference is then quantized with a split VQ scheme composed of four vector quantizers.



Vector Quantization of Spectral envelope

2.7 V/UV decision

2.7.1 Tool description

A V/UV decision is made per 20ms frame. The decision is made based on: similarity of the shape of the synthesized spectrum and original spectrum, signal power, maximum autocorrelation of LPC residual signals normalized by residual signal power, and number of zero crossing.

2.7.2 Encoding process

The V/UV decision is composed of three different modes, that is, Unvoiced, Mixed Voiced, and Fully Voiced. In order to send out the information of the three modes, two bits are used for V/UV. First, a decision is made whether or not the current frame is Unvoiced. When this decision is Voiced, then the voicing strength is evaluated from the value of the normalized maximum peak of the autocorrelation of LPC residual signal. Let us denote the value r0r.

Shown below is the decision rule :

When the first decision is Unvoiced	V/UV is 0			- Unvoiced
When the first decision is Voiced	V/UV is 1	for	r0r < TH2	- Mixed Voiced 1
	V/UV is 2	for	TH2<= r0r < TH1	- Mixed Voiced 2
	V/UV is 3	for	TH1 <= r0r	- Fully Voiced

Example:
TH1=0.7
The number 0,1,2,3 is send out to the decoder using the two bits.

2.8 Time domain encoder

2.8.1 Tool description

When a speech segment is unvoiced, Vector Excitation Coding (VXC) algorithm is used. Fig.2.8.1 shows the overall structures of the VXC encoder. The operation of the VXC is described below.

2.8.2 Encoding process

LPC analysis is carried out first, and LPC coefficients (α) are then converted to LSP parameters in the same manner as in voiced case. LSP parameters are quantized, and quantized LSPs are converted to alpha parameters ($\hat{\alpha}$).

Perceptually weighted LPC synthesis filter $H(z)$ is expressed as:

$$H(z) = A(z)W(z)$$

where $A(z)$ is a transfer function of the LPC synthesis filter, and $W(z)$ is a perceptual weighting filter derived from LPC coefficients.

Let $x_w(n)$ be perceptually weighted input signal. Subtracting zero-input response (of both 1st and second stage when bit-rate is 6kbps) $z(n)$ from $x_w(n)$, we obtain reference signal, $r(n)$, for the analysis-by-synthesis procedure of the VXC. Optimal shape and gain vectors are searched using the distortion measure E:

$$E = \sum_{n=0}^{N-1} \left(r(n) - g \times syn(n) \right)^2$$

where $syn(n)$ is zero-state response of $H(z)$, driven by a excitation input by only a shape vector $s(n)$, which is an output of VX_shape codebook. g is a gain, which is an output of VX_gain coodbook. N is vector dimension of VX_shape codebook. ($N = 80$ at 2kbps, $N = 40$ at 6kbps)

The codebook search process for the VXC consists of two steps, that are:

1. Search $s(n)$ that maximize

$$E_s = \frac{\sum_{n=0}^{N-1} r(n) \times syn(n)}{\sqrt{\sum_{m=0}^{N-1} syn(m)^2}}$$

2. Search g that minimize

$$E_g = \left(g_{ref} - g \right)^2$$

where

$$g_{ref} = \frac{\sum_{n=0}^{N-1} r(n) \times syn(n)}{\sum_{m=0}^{N-1} syn(m)^2}$$

Quantization error $e(n)$ is computed as:

$$e(n) = r(n) - g \times syn(n)$$

When the bit-rate is 4kbps, one more stage is used for the quantization of unvoiced segments, and $e(n)$ is used as reference input to the second stage VQ.
The operation of the second stage VQ is the same as that of the first stage VQ.

2.0kbps coder uses 6bits shape and 4bits gain codebooks for the unvoiced excitation for every 10msec. The 4kbps scheme adds 5bits shape and 3bits gain codebooks for every 5 msec at the bottom of the current quantizer.

1st stage	80 dimension 6bits shape + 4bits gain
2nd stage	(40dimension 5bits shape + 3bits gain) x 2

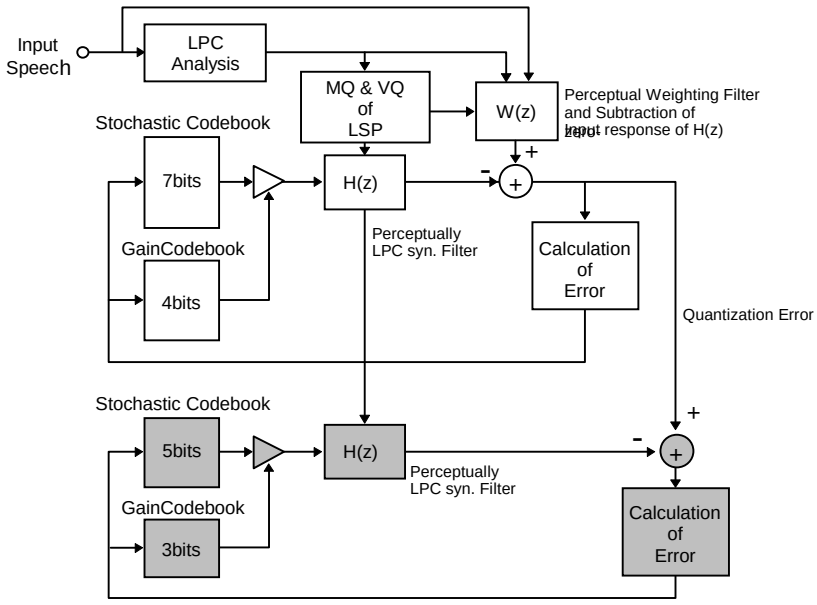


Fig. 2.8.1 Vector Excitation Coding (VXC) for unvoiced signals

2.9 Variable rate encoder

This chapter describes a tool for variable rate coding with the HVXC core of the VM. This tool allows HVXC to operate at variable bit rates. The major part of the algorithm is composed of "background noise interval detection", where only the mode bits are transmitted during the "background noise mode", and unvoiced frame is inserted with certain period of time to send parameters for background noise generation

In the encoding algorithm, minimum level tracker has temporal minimum level in order to adjust a threshold value by which a decision is made about whether or not the input segment is speech.

Minimum level tracking

Parameters are defined as follows:
lev: r.m.s. of a speech frame
vCont: number of continuous voiced frames
cdLev: candidate value of minimum level
prevLev: r.m.s. of a previous frame
gmlSetState: number of frames in which the candidate value is set
gmlResetState:number of frames in which the candidate value is not set after setting of minimum level
gml:minimum level

The minimum level is tracked according to the algorithm shown below.

```
if(vCont > 4)
{
        cdLev = 0.0;
        gmlSetState = 0;
        gmlResetState++;
}
else if(lev < MIN_GML)
{
        *gml = MIN_GML;
        gmlResetState = 0;
        gmlSetState = 0;
}
else if(*gml > lev)
{
        *gml = lev;
        gmlResetState = 0;
        gmlSetState = 0;
}
else if((lev < 500.0 && cdLev * 0.70 < lev && lev < cdLev * 1.30) || lev < 100.0)
{
        if(gmlSetState > 6)
        {
                *gml = lev;
                gmlResetState = 0;
                gmlSetState = 0;
        }
        else
        {
                cdLev = lev;
                gmlResetState = 0;
                gmlSetState++;
        }
}
else if((lev < 500.0 && prevLev * 0.70 < lev && lev < prevLev * 1.30) || lev < 100.0)
{
        cdLev = lev;
        gmlResetState = 0;
        gmlSetState++;
}
else if(gmlResetState > 40)
{
        *gml = MIN_GML;
        gmlResetState = 0;
```

```
            gmlSetState = 0;
    }
    else
    {
            cdLev = 0.0;
            gmlSetState = 0;
            gmlResetState++;
    }
```

As shown above, the minimum level is held during appropriate period of time and updated. The minimum level is always set higher than a predetermined value MIN_GML.

Background noise detection based on the minimum level

The reference level refLev is computed as,

$$refLev = A \times \max(lev, refLev) + (1.0 - A) \times \min(lev, refLev) \qquad (1)$$

Usually A is set to 0.75. Background noise detection is carried out using the refLevel derived from the equation (1).

For the frames where "voiced" decision is made:

```
if refLev < B *gml and contV < 2
idVUV = 1
```

For the frames where "unvoiced" decision is made:

```
if refLev < B *gml ... condition-1
        if bgnCnt < 3
                bgnCnt++
        else
                if bgnIntvl < 8
                        idVUV=1
                        bgnIntvl++
                else
                        bgnIntvl=0
        else
                bgnCnt=0
```

where B is a constant. In this case we set B=2.0
Each parameter is defined below.

countV : number of consecutive voiced frames
bgnCnt: number of frames which satisfies the condition-1
bgnIntvl: number of frames where "Background noise" mode is declared

idVUV is a parameter that has the result of V/UV decision and defined as;

$$\text{idVUV} = \begin{cases} 0 & \text{Unvoiced speech} \\ 1 & \text{Background noise interval} \\ 2 & \text{Mixed voiced speech} \\ 3 & \text{Voiced speech} \end{cases}$$

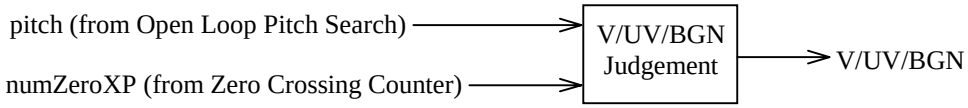
If the current frame is declared to be "Voiced", it is checked whether or not the previous frame is "Voiced". If the previous frame is "Voiced", "backgroundnoise" mode is not selected; otherwise, "background noise" mode is selected.

If the current frame is declared to be "Unvoiced", "background noise" mode is selected only after the condition-1 is satisfied for four consecutive frames. When "background noise" mode is selected for consecutive N frames, then the last frame is replaced with "Unvoiced" mode in order to transmit speech parameters which represents characteristics of the time varying back ground noise.
At this moment, N is set to 9.

Variable rate encoding:
Using the background noise detection method described above, variable rate coding is carried out based on fixed bit rate 2kbps HVXC.

Mode(idVUV)	Back Ground Noise(1)	UV(0)	MV(2), V(3)
V/UV LSP Excitation	2bit/20msec 0bit/20msec 0bit/20msec	2bit/20msec 18bit/20msec 8bit/20msec (gain only)	2bit/20msec 18bit/20msec 20bit/20msec
Total	2bit/20msec 0.1kbps	28bit/20msec 1.4kbps	40bit/20msec 2.0kbps

If the current frame or the previous frame is "Background noise" mode, differential mode in LSP quantization is inhibited in the encoder because LSP parameters are not sent during "Background noise" mode and inter frame coding is not possible.



Block Diagram for variable rate encoding

3 HVXC Decoder tools

3.1 postfilter

3.1.1 Tool description

The basic operation of the post-filter is to enhance the spectral formants and suppress the spectral valley.
One can use a single postfilter after voiced and unvoiced synthesized speech are added.
Alternatively, one can use independent postfilters for voiced and unvoiced speech respectively.
Use of independent postfilters for voiced and unvoiced signals are recommended.
Each of the voiced speech and unvoiced speech obtained is fed into the independent postfilters.

3.1.2 Definitions

Pf_v : *Transfer function of spectral shaping filter for voiced speech.*

Pf_{uv} : *Transfer function of spectral shaping filter for unvoiced speech*

r_{adj} : *gain adjustment factor for spectral shaping.*

$S(n)$: *LPC sythesis filter $H(z)$ output.*

$S_{pf}(n)$: *Spectral shaping filter $Pf_v(z)$ output.*

$S_{pf}'(n)$: *Spectral shaped and gain adjusted output.*

$S_{pf}'\_prev(n)$: *$S_{pf}'(n)$ computed using previous frame's α_n and r_{adj}.*

$vspeech(n)$: *Postfiltered voiced speech.*

$uvspeech(n)$: *Postfiltered unvoiced speech.*

3.1.3 Processing

Each of the operation of the postfilter consists of three steps, that is, spectral shaping, gain adjustment and smoothing process.

Voiced speech:

Spectral shaping:

$$Pf_v(z) = \frac{\sum_{n=0}^{P} \alpha_n \gamma^n z^{-n}}{\sum_{n=0}^{P} \alpha_n \beta^n z^{-n}} \left(1 - \delta z^{-1}\right)$$

Output of the LPC synthesis filter s(n) is first fed into the spectral shaping filter $Pf_v(z)$, where α_n are linear predictive coefficients converted from de-quantized and linearly interpolated LSPs, which are updated every 2.5 ms. $p = 10, \gamma = 0.5, \beta = 0.8, \delta = -0.15 \times \alpha_1$ where the value of δ is limited to the range of $0 \le \delta \le 0.5$.

Gain adjustment:

The output of the spectral shaping filter, $S_{pf}(n)$, is then gain adjusted so that the frame gain of the input and output of the spectral shaping is unchanged. When low delay decode mode is selected, decode frame interval is shifted by 2.5 ms and interpolation of LSPs is carried out for the first 17.5ms of decode frame interval shown in Fig. 3.6.1, and the latest LSPs are used for the last 2.5 ms without interpolation. Gain adjustment is done once every 160 sample frame while LSPs are updated every 2.5ms. Gain adjustment factor r_{adj} is computed as follows:

$$r_{adj} = \sqrt{\frac{\sum_{n=0}^{159} \{s(n)\}^2}{\sum_{n=0}^{159} \{s_{Pf}(n)\}^2}}$$

Spectral shaped and gain adjusted output $s_{pf}'(n)$ is obtained as:

$$s_{Pf}'(n) = r_{adj} s_{Pf}(n) \quad (0 \le n \le 159)$$

Smoothing process:

Spectral shaped and gain adjusted output $s_{Pf}'(n)$ are then fed to smoothing process to avoid discontinuity due to parameter change at the beginning of each frame. Postfiltered voiced output $vspeech(n)$ is obtained as follows

$$vspeech(n) = \begin{cases} \left(1 - \frac{n}{20}\right) s_{Pf\_prev}'(n) + \frac{n}{20} s_{Pf}'(n) & (0 \le n \le 19) \\ s_{Pf}'(n) & (20 \le n \le 159) \end{cases}$$

where $s_{Pf\_prev}'(n)$ is a $s_{Pf}'(n)$ computed using previous frame's α_n and r_{adj}.

Unvoiced speech:

Spectral shaping:

$$Pf_{uv}(z) = \frac{\sum_{n=0}^{P} \alpha_n \gamma^n z^{-n}}{\sum_{n=0}^{P} \alpha_n \beta^n z^{-n}} \left(1 - \delta z^{-1}\right)$$

Similarly the transfer function of the spectral shaping filter for unvoiced speech is given as $Pf_{uv}(z)$, where α_n are linear predictive coefficients converted from de-quantized LSPs, which are updated every 20 ms. When normal decode mode is selected, LSP coefficients are updated at the middle of decode frame interval. If low delay decode mode is selected , decode frame interval is shifted by 2.5 ms and LSP update happens at 7.5 ms point from the beginning of the decode interval in Fig 3.6.1. $p = 10, \gamma = 0.5, \beta = 0.8, \delta = 0.1$. **Gain adjustment** and **Smoothing process** same as the voiced part described above is carried out to obtainn postfiltered unvoiced speech $uvspeech(n)$.

Output of each of the postfilter, $vspeech(n)$ and $uvspeech(n)$ are added to generate postfiltered speech output. The use of postfilter is required , however, the configuration and constants of the postfilters described here is one example and not normative, and they could be modified.

3.2 Post processing

3.2.1 Tool description

Output of the postfilter is fed into the post processing part. Post processing is composed of three filters, those are, high pass filter, high frequency emphasis filter, and low pass filter. High pass filter is used to remove unnecessary low frequency components, high frequency emphasis is used to increase the brightness of the speech, and low pass filter is used to remove unnecessary high frequency components. The filter configurations and constants described here is one example and not normative, and they could be modified.

3.2.2 Definitions

$HPF(z)$:	Transfer function of high pass filter.
$Emp(z)$:	Transfer function of high frequency emphasis filter.
$LPF(z)$:	Transfer function of low pass filter.

3.2.3 Processing

The three filters below are applied to the output of the postfilter.

High Pass Filter:

$$HPF(z) = G_{inv}\frac{1+A_1z^{-1}+B_1z^{-2}}{1+C_1z^{-1}+D_1z^{-2}}\cdot\frac{1+A_2z^{-1}+B_2z^{-2}}{1+C_2z^{-1}+D_2z^{-2}}$$

High Frequency Emphasis:

$$Emp(z) = GG\frac{1+AA\ z^{-1}+BB\ z^{-2}}{1+CC\ z^{-1}+DD\ z^{-2}}$$

Low Pass Filter:

$$LPF(z) = GL\frac{1+AL\ z^{-1}+BL\ z^{-2}}{1+CL\ z^{-1}+DL\ z^{-2}}$$

3.2.4 Tables

G_{inv}	1.100000000000000
A_1	-1.998066423746901
B_1	1.000000000000000
C_1	-1.962822436245804
D_1	0.9684991816600951
A_2	-1.999633313803449
B_2	0.9999999999999999
C_2	-1.858097918647416
D_2	0.8654599838007603

AA	0.551543
BB	0.152100
CC	0.89
DD	0.198025
GG	1.226

AL	-2. * 1. * cos((4.0/4.0)* π)
BL	1.
CL	-2. * 0.78 * cos((3.55/4.0)* π)
DL	0.78*0.78
GL	0.768

4 HILN Encoder Tools

The basic principle of the „Harmonic and Individual Lines plus Noise“ (HILN) encoder is to analyse the input signal in order to extract parameters describing the signal. These parameters are coded and transmitted as a bitstream. In the decoder the output signal is synthesised based on the parameters extracted and transmitted by the encoder.

The encoder consist of two main parts: „Parameter Extraction“ and „Parameter Coding“. In the encoder, the input signal is divided into consecutive frames and for each frame a set of parameters describing the signal in this frame is extracted and coded. Due to this parametric description, a wide range of bitrates, sampling rates and frame lengths are possible. Typically a frame length of 32 ms is used. For input signals with 8 kHz sampling rate typically a bitrate of 6 kbit/s is used. For signals with greater bandwidth, a higher bitrate is recommended.

The „Parameter Extraction“ and „Parameter Coding“ is described in detail in the following sections.

4.1 HILN Parameter Extraction

Since different parameter sets and different synthesis techniques can be applied, the input signal of the encoder has to be split up in an appropriate way. This is performed by the *Separation* unit. Depending on the most appropriate synthesis technique, a parameter set is derived for each part of the input signal in the *Model Based Parameter Estimation* unit. The two units *Separation* and *Model Based Parameter Estimation* can be regarded as the analysis stage which produces a parametric description of the input signal. The separation of the input signal is enhanced by feeding back the signals which are generated in the *Synthesis* unit from all the parameters of previously separated parts. The *Separation* and the *Model Based Parameter Estimation* additionally receive data from a synthesis model independent *Pre-Analysis*. Prior to transmission, the parameters are fed through the *Quantization and Coding* unit, which is controlled by a *Psychoacoustic Model*. This *Psychoacoustic Model* processes the input signal in order to derive information about the relevancy of synthesis parameters. In addition, the synthesized signal is fed into the *Psychoacoustic Model*, which thus is allowed to assist the *Model Based Parameter Estimation*.

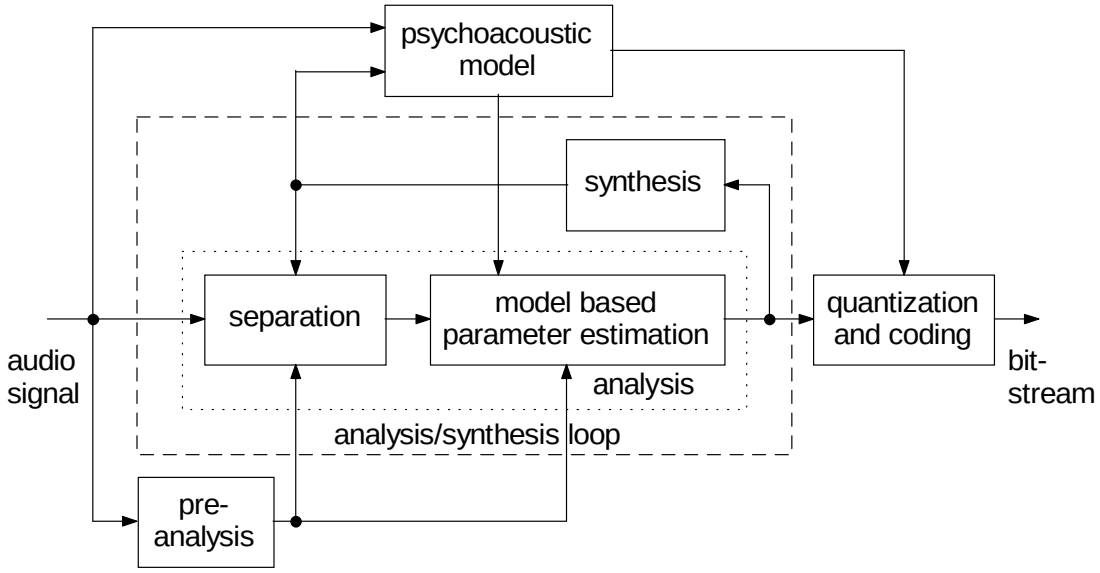


Figure: Block diagram of the HILN encoder.

In the parameter extraction, the input signal is separated into three different parts: „harmonic lines“, „individual lines“ and „noise“.

For each of these parts parameters describing the signal are extracted. These are basically:

- harmonic lines: fundamental frequency and amplitudes of the harmonic components
- individual lines: frequency and amplitude of each individual line
- noise: spectral shape of the noise

Additionally parameters for amplitude envelopes and for continuation of spectral lines from one frame to the next can be determined.

The signal separation and parameter estimation is implemented in three steps: First the fundamental frequency of the harmonic part of the signal is estimated. Then the parameters of the relevant spectral lines are estimated and these lines are classified as „individual lines“ or „harmonic lines“ depending on the frequency with respect to the fundamental frequency. After all relevant spectral lines are extracted, the remaining residual signal is assumed to be noise-like and its spectral shape is described by a set of parameters.

The harmonic line extraction of the HILN tools could also be utilised in an integrated parametric core utilising both the HVXC speech coding tools as well as the HILN coding tools simultaneously. If the input signal is e.g. a speech signal mixed with background music, the HILN encoder can be used to extract only those individual spectral lines that do not belong to the harmonic part of the signal. These individual lines are encoded by the HILN tools and the remaining signal - consisting of the harmonic signal part and noise - then is encoded by the HVXC parametric speech codec tools. In the decoder, the audio signal is reconstructed by adding the output of the „individual line“ synthesiser and the HVXC decoder.

4.1.1 Fundamental frequency estimation

A „cepstrum“-based fundamental frequency estimation technique is employed by the HIN tools. First the input signal is windowed with a Hanning window of twice the frame length centered around the current frame. For the windowed signal, the magnitude spectrum is calculated and the logarithm is applied to the magnitude spectrum. Then the log spectrum is multiplied with the window

$$w(f) = (1+\cos(2*pi*f/fs))/2 \qquad 0 <= f <= fs/2$$

and zero-padding is used to virtually double the sampling frequency before the cepstrum is calculated. Finally the local maxima in the cepstrum are determined and the largest maximum within the permitted „pitch lag“ search range is identified. The fundamental frequency is calculated from the „pitch lag“ (period of the fundamental frequency) of the largest maximum.

The fundamental frequency determined by this cepstrum-based technique is used as an initial (coarse) estimate in the following line parameter estimation.

4.1.2 Harmonic and individual line parameter estimation

The estimation of harmonic and individual line parameters is based on an „Analysis/Synthesis Loop“ described in the following sections.

In a first step the parameters of all harmonic lines are estimated. This is done by performing the regression-based high accuracy frequency estimation for all integer multiples of the coarse fundamental frequency as initial estimates. Based on the accurate frequencies of the harmonic lines, a fine estimate of the fundamental frequency hFreq and the so-called „stretching“ hStretch is calculated which minimises the total error between the real harmonic line frequencies and those calculated according to

$$hLinefreq[i] = hFreq * (i+1) * (1 + hStretch*(i+1)) \qquad i = 0 .. harmNumLine-1$$

where the total number of harmonic lines is determined by the bandwidth w of the signal and the current fundamental frequency hFreq:

$$harmNumLine = floor(w/hFreq)$$

The harmonic envelope flag is set if using the current amplitude envelope for all harmonic lines results in a lower residual error than if no envelope is used. If the relative change of the fundamental frequency between the previous and the current frame is less that 15%, the harmonic continuation flag is set.

In the second step the relevant spectral lines are extracted from the input signal by means of the „Analysis/Synthesis Loop“. This loop utilises a psychoacoustic model to extract the spectral lines in order of their subjective relevance. If the frequency of an extracted spectral line is close to the frequency of a harmonic line as calculated from hFreq and hStretch, this extracted line is classified as harmonic line. Otherwise it is classified as individual line. The „Analysis/Synthesis Loop“ is terminated if the requested number of individual lines was extracted or if the remaining signal components cannot be properly modeled by spectral lines. The ratio between

the number of harmonic lines extracted and the total lines extracted is passed on to the encoder as measure of „relevance“ of the harmonic lines.

If less than three extracted lines were classified as „harmonic line“, these lines are added to the list of „individual lines“ and numHarmLine is set to 0. Finally all harmonic lines that were not extracted by the „Analysis/Synthesis Loop“ are also removed from the residual signal. This residual signal is then passed to the noise parameter estimation.

4.1.2.1 Pre-Analysis

The pre-analysis module determines the signal amplitude envelope which is used in the analysis/synthesis loop.

4.1.2.2 Analysis/Synthesis based on single spectral lines

The Individual Line encoder is based on the model of single spectral lines, which can be generated with the help of sine wave generators. The according *Model Based Parameter Estimation* for the i-th line in the loop consists of the following steps:

- calculation of the deviation between FFT spectra of input and synthesized signals
- selection of the most relevant FFT line with center frequency $f_{i,m}$
- high resolution frequency estimation in the surrounding of $f_{i,m}$
- amplitude and phase estimation, selection of envelope information
- synthesis with the determined parameters
- calculation of the residual error signal by subtraction of the synthesized signal from the input signal

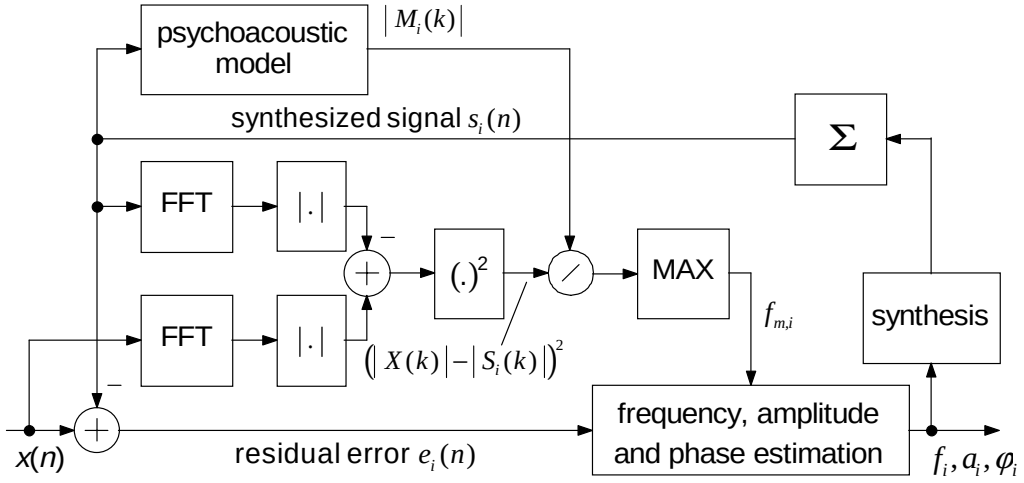


Figure 1: Analysis/Synthesis Loop based on the synthesis method „single spectral lines“.

The FFT line to be processed is determined by calculating the deviation between input spectrum and synthesized spectrum and searching the maximum ratio of the square of this deviation and the masking threshold derived from the signal synthesized from the previously determined spectral lines.
Based on the center frequency $f_{i,m}$ of the selected FFT line a frequency estimation is performed in order to obtain a frequency parameter of higher accuracy than the FFT resolution (Figure 2).

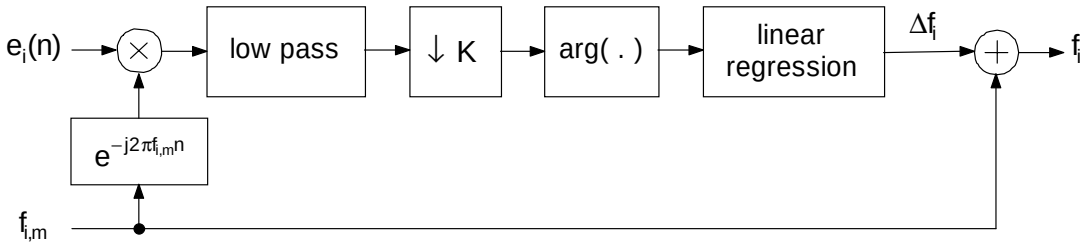


Figure 2: High accuracy frequency estimation.

For this frequency estimation, first the spectrum of the residual error signal is shifted in a way that the center frequency $f_{i,m}$ of the selected FFT line becomes zero. The complex output values of this operation are fed into lowpass filter and sampling rate reduction, which are realized by applying time-shifted versions of the window function determined in the *Pre-Analysis*. The slope of the regression line for the phase values of the obtained complex samples gives a frequency offset, which is added to $f_{i,m}$ in order to give a high resolution frequency parameter f_i. In the current implementation the time shift of the window function ranges from -0.32 to 0.32 times the frame length. The time shift step size is 0.08 times the frame length and thus 9 data points are used for the linear regression.
Based on f_i the amplitude and phase of the spectral line are calculated. This is realized by calculating a complex correlation coefficient of the residual error signal and a complex harmonic signal of frequency f_i. The absolute value of the correlation coefficient gives the amplitude parameter a_i and the argument gives the phase parameter φ_i. If the *Pre-Analysis* has generated envelope parameters, a second set of parameters $a_{i,e}$ and $\varphi_{i,e}$ is generated by correlation of the residual error signal and a complex harmonic signal of frequency f_i multiplied with the according envelope.
In the *Separation* unit, a new residual error signal is generated by subtracting the signal synthesized from the parameters f_i, a_i and φ_i. If the parameters $a_{i,e}$ and $\varphi_{i,e}$ are also available, a second signal is synthesized accordingly and the parameter set leading to the lowest residual error variance is selected.

4.1.2.3 Psychoacoustic Model

The psycoacoustic model calculates the masked threshold for the synthesised signal components in the analysis/synthesis loop.

4.1.3 Noise parameter estimation

The noise parameters are used to model the spectral shape of the residual signal. First the magnitude spectrum of the Hanning-windowed residual signal is calculated. Then a DCT is applied to the magnitude spectrum. Since only the global shape of the spectrum of the residual signal is of interest, only the first 7 DCT coefficients are passed on to the encoder.

Additionally a new set of envelope parameters is calculated for the residual signal. Thus also the temporal shape of the residual signal can be modeled. The ratio of residual signal power to input signal power is calculated and passed to encoder as a measure of „relevance“ of the noise-like signal component.

4.2 HILN parameter encoder

The extracted parameters of the harmonic, individual line and noise parts of the signal are quantised and encoded to generate the bistream output of the HILN encoder.

The allocation of the bits available in a frame to the parameters for the three parts of the signal is determined by the harmonic and noise component „relevance“ measures calculated during the parameter estimation.

4.2.1 Harmonic line parameter quantisation

The number of bits available for the harmonic line parameters depends on the „relevance“ measure of the harmonic signal component. If this measure is low, the number of harmonic lines encoded can be less than the number of lines extracted. This corresponds to a bandwidth limitation of the harmonic signal.

The fundamental frequency is quantised with 11 bits on a logarithmic scale ranging from 30 Hz to 4 kHz. The „stretching“ parameter is quantised with 5 bits on a uniform scale ranging from -0.001 to +0.001.

Prior to quantisation, the harmonic line amplitudes are grouped according to the Table given for the harmonic line dequantiser. This algorithm is used for grouping:

```
for (i=0; i< numHarmTrans; i++) {
        hTransAmpl[i] = 0
        for (k=0; k<w[i]; k++)
                hTransAmpl[i] += hLineAmpl[j0[i]+k] / sqrt(w[i])
}
```

All harmonic amplitude parameters are quantised with a logarithmic quantiser having a step size of 2 dB. The first parameters hTransAmpl[0] is quantised directly with 6 bits and for all following parameters only the difference of the logarithmic amplitude is quantised with 4 or 5 bits - depending on the addHarmAmplBits flag.

Since this coding scheme limits the maximum difference between neighbouring amplitude parameters, special care has to be taken to ensure that the large harmonic amplitude parameters are properly reconstructed if some differences exceed the range of the quantiser.

4.2.2 Individual line parameter quantisation

In the *Quantization and Coding* unit, the parameters are processed in the order as they are obtained from the *Analysis/Synthesis Loop*, since this order corresponds to the relevancy with respect to the reproduction of the sound. This unit is able to generate two bitstreams, one *basic bitstream* which allows the generation of the basic quality audio signal, and an *enhancement bitstream* which can be used in applications where a difference signal between input and decoder output is needed, e.g. for scalability. The *basic bitstream* mainly contains the frequency and amplitude parameters, while the *enhancement bitstream* contains the phase parameters and information for finer quantization of frequency and envelope parameters.
For each frame of the input signal, a constant number of bits according to the desired bit rate is transmitted. The first bit in each frame is the *envelope bit*, which indicates whether envelope parameters are used or not. If this bit is set, the 6 envelope parameters follow, and for each transmitted line an additional *line envelope bit* is transmitted, which indicates, whether a constant amplitude or the envelope is to be used for the synthesis of the corresponding line.
Since the human auditory system is not very sensitive to phase changes, only the frequency and amplitude information of the spectral lines are coded and transmitted in the *basic bitstream* to obtain a signal with the basic audio quality. But in this case, it is necessary to provide information for the decoder which enables it to generate a signal free of phase discontinuities at frame boundaries. Therefore the first processing stage detects lines which continue from one frame to another. If a line is to be continued from the previous frame, only the frequency and amplitude changes are quantized and transmitted instead of the absolute frequency and amplitude values. For this purpose the frequency and amplitude parameters of the *i*-th line in the current frame *m* are compared with those of the *k*-th line in the previous frame *m*-1 for all possible combinations of *i* and *k*. The line continuation is used, if the relative frequency change

$$q_f(i,k) = \frac{\left| f_i(m) - f_k(m-1) \right|}{f_i(m)}$$

does not exceed a given threshold $q_{f,max}$ and if the ratio of amplitudes

$$q_a(i,k) = \begin{cases} a_i(m)/a_k(m-1) & \text{if } a_i(m) \geq a_k(m-1) \\ a_k(m-1)/a_i(m) & \text{if } a_i(m) < a_k(m-1) \end{cases}$$

lies within the interval $[1...q_{a,max}]$. If there is more than one possibility to continue a line from the previous frame, that line in the previous frame is selected, for which the following similarity criterion reaches its maximum:

$$Q = \frac{q_{f,max} - q_f(i,k)}{q_{f,max}} \cdot \frac{q_{a,\max} - q_a(i,k)}{(q_{a,\max} - 1)\, q_a(i,k)}$$

For each line, a *continuation bit* is transmitted in the bitstream, which indicates whether the line is continued from the previous frame or not. If this bit is set, the relative frequency change is transmitted after uniform quantization and the amplitude ratio is transmitted after quantization with a logarithmic characteristic. If the continuation bit is not set, the absolute frequency and amplitude values are both quantized with logarithmic characteristic.
Since the transmission of absolute frequency and amplitude values requires more bits per line than for the relative values, the number of lines transmitted per frame is varied in order to obtain a constant bit rate for the *basic bitstream*.
Since the *basic bitstream* does not contain phase information, it is not useful to calculate a residual error signal by subtracting the corresponding decoder output signal from the input signal. In order to enable scalability modes, in which the residual signal is transmitted in a bitstream of higher bit rate, an additional *enhancement bitstream* is generated. It is constructed in the following way:

- if the envelope parameters are transmitted in the *basic bitstream*, additional bits for finer quantization of the 4 time parameters t_1, t_2, t_3, and t_4 are transmitted
- if a line is starting, i.e. not continued from the previous frame, and its frequency exceeds a given threshold, additional bits for finer quantization of the absolute frequency are transmitted
- for each line the phase parameter is transmitted after uniform quantization

The number of bits per frame in the *enhancement bitstream* can vary, this has to be taken into account in the calculation of available bits for the coding of the residual error.

If a line is continued from the previous frame, the line that has been transmitted in the same position in the previous frame is used as predecessor. To increase the probability that the most relevant lines of the previous frame can be continued in the current frame, the lines that were transmitted in the previous frame are reordered according to their amplitudes before the current frame is encoded. Since this reordering must be done in exactly the same way in both the encoder and decoder, the quantized amplitudes as reconstructed by the decoder have to be used in the encoder as well. Since the position of a continued line in the current frame is determined by the position of its predecessor in the previous frame and not by its position in the list of most relevant lines found by the *Analysis/Synthesis Loop*, a bit allocation algorithm is used which ensures that the *N* lines transmitted in the current frame are always the *N* most relevant lines found by the *Analysis/Synthesis Loop*.
The system delay of the encoder is 1.5 times the frame length. This delay results from the length of the frame itself plus an additional delay of (0.5) times the frame length caused by the shifted overlapping window used for the frequency estimation.

4.2.3 Noise parameter quantisation

The number of noise parameters that are quantised and encoded depends on the „relevance“ measure of the noise signal component. If it is very low, no noise parameters are transmitted. For higher values of this measure at least 4 and as many as 7 noise parameters (i.e. DCT coefficients) are quantised.

First the maximum of the DCT noise parameters to be encoded is determined. This value is with 6 bits and a quantiser step size of 3 dB. Then all DCT noise parameters are divided by this maximum to normalise them to the range [-1, +1]. The normalised noise parameters are now quantised with 3 bits on a uniform scale ranging from 0 to 1 for the first parameter and from -1 to 1 for all following parameters.

If the noiseEnvFlag is set then also the additional set of noise envelope parameters is quantised in the same way as described in the current WD.

5 Music/Speech mixed encoder tool

In MPEG-4 Audio the parametric core is utilised for coding natural audio signals at very low bitrates ranging from 2 kbit/s to about 8 kbit/s. The parametric core provides two sets of tools suited for coding speech and non-speech audio signals respectively:

- The Harmonic Vector Excitation (HVXC) tools are suited for coding speech signals at 2 kbit/s and 4 kbit/s.
- The Harmonic and Individual Lines plus Noise (HILN) tools are suited for coding non-speech audio signals at bitrates of about 4kbit/s and above.

In the HVXC only or HILN only mode, the encoding mode is selected manually during encoding and the selected mode is used for all of the audio signal being encoded.

In this section, an integrated parametric core which utilises the HVXC and HILN tools alternatively or simultaneously is described. This integrated core automatically selects the coding tools that are suited best for the actual input signal characteristics. In case of a speech signal the HVXC tools are used and for music the HILN tools are used. This selection is done based on the decision of an automatic speech/music classification tool. For signals which are a mixture of speech and music, it is also possible to use the HVXC and HILN tools simultaneously.

5.1 Music/Speech Classification Tool

This is a tool for the parametric speech coder core, which enables automatic music/speech identification for the parametric speech/audio core (HVXC and HILN). The tool makes decisions using internal parameters of the HVXC.

The features of input sequences used for the identification are: behavior of "pitch strength" and "frame energy". In general, speech has higher "pitch strength" and frequent and higher energy change than music.

This music/speech classification tool can be applied in two ways:

- The first 5 seconds of the signal to be encoded are analysed by the classification tool and then either HVXC or IL are selected to encode the signal according to the speech/music decision.
- The classification tool is operated continuously and its current speech/music decision is used to select HVXC or IL for the current frame. In this application the decision delay of 5 sec has to be taken into account.

5.1.1 Frame Energy

Frame Energy P is computed as:

$$P = \sum_{n=0}^{159} s(n)^2$$

where s(n) is the input signal.

In this case, frames with energy levels higher than a pre-determined minimum level are used (ex. > -78dB). A short-term average frame energy is defined as

$$Pav = \sum_{t=0}^{3} P\{t\} / 4$$

which is computed from the last four Frame energies.

A difference between frame energy and short term average frame energy is computed as:

$$Pd[frm] = |P - Pav| / Pav$$

Pd[frm] is kept for around 250 frames (5 seconds).

5.1.2 Pitch Strength

In HVXC, maximum autocorrelation of LPC residual (r0r) is computed during pitch detection process. r0r are kept for around 250 frames.

5.1.3 Music/Speech decision

Mean and variance of frame energies and r0rs are computed respectively as:

$$Pd(av) = \sum_{frm=0}^{249} Pd[frm] / 250$$

$$Pd(va) = \sqrt{\sum_{frm=0}^{249} \left(Pd[frm] - Pd(av) \right)^2} / 250$$

$$r0r(av) = \sum_{frm=0}^{249} r0r[frm] / 250$$

$$r0r(va) = \sqrt{\sum_{frm=0}^{249} \left(r0r[frm] - r0r(av) \right)^2} / 250$$

Speech data has higher variances than music data in the same range of mean value of r0r. The matrix is classified to three areas.

(1)speech $r0r(va) \geq 0.153\, r0r(av) + 0.113$

(2)unknown $0.07\, r0r(av) + 0.137 < r0r(va) < 0.153\, r0r(av) + 0.113$

(3)music $0.07\, r0r(av) + 0.137 \geq r0r(va)$

If mean and variance are included in the area (1), the data is classified as speech. If they are in the area (3), the data is classified as music.

If the mean and variance exist in the area (2), the mean and variance of (differential) frame energy Pd are used additionally. Speech data has larger means and variances of Pd than music data. Speech and music data is separated into the following two areas.

(1)speech $Pd(va) \geq -0.5 Pd(av) + 0.8$

(2)music $Pd(va) < -0.5 Pd(av) + 0.8$

Using the above two criteria, speech and music are separated.

5.1.4 Integrated parametric coder

The integrated parametric coder can operate in the following modes:

PARAmode	Description
0	HVXC only

1	HILN only
2	switched HVXC / HILN
3	mixed HVXC / HILN

PARAmodes 0 and 1 represent the fixed HVXC and HILN modes. PARAmode 2 permits automatic switching between HVXC and HILN depending on the current input signal type. In PARAmode 3 the HVXC and HILN coders can be used simultaneously and their output signals are added (mixed) in the decoder.

The integrated parametric core uses a frame length of 40 ms and a sampling rate of 8 kHz and can operate at 2025 bit/s or any higher bitrate. Operation at 4 kbit/s or higher is suggested.

5.1.5 Integrated Parametric Encoder

For the „HVXC only“ and „HILN only“ modes the parametric encoder is not modified. The „switched HVXC / HILN“ and „mixed HVXC / HILN“ modes are described below.

5.1.6 Switched HVXC / HILN mode

Because the speech/music classification tool is based on the HVXC encoder, the HVXC encoder is operated continuously for every frame. The bitstream frame generated by the HVXC encoder and the input audio signal are stored in two FIFO buffers to compensate for the 5 sec delay of the speech/music decision. If a frame is classified as „speech“ then PARAswitchMode is set to 0 and the HVXC bitstream frame available at the bitstream FIFO output is transmitted. In case of a „music“ decision, then PARAswitchMode is set to 1 and the output of the signal FIFO buffer is encoded by the HILN encoder and this HILN bitstream frame is transmitted. If HVXC is used for a frame, the HILN encoder is reset (prevNumLine = 0).

5.1.7 Mixed HVXC / HILN mode

To operate the parametric codec in „mixed HVXC / HILN“ mode, speech and music components of the input signal have to be separated. If both components are already available separately (e.g. speech and background music) encoding is straightforward.

Annex B
(informative)
Patent Holders

List of patent holders

The user’s attention is called to the possibility that - for some of the processes specified in this part of ISO/IEC 14496 - conformance with this part of ISO/IEC 14496 may require use of an invention covered by patent rights. By publication of this part of ISO/IEC 14496, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. However, each company listed in this annex has undertaken to file with the Information Technology Task Force (ITTF) a statement of willingness to grant a license under such rights that they hold on reasonable and non-discriminatory terms and conditions to applicants desiring to obtain such a license.

Information regarding such patents can be obtained from the following organizations.

The table summarizes the formal patent statements received and indicates the parts of the standard to which the statement applies. Three “N”s in the row corresponding to a company mean that the statement from the company did not mention whether part 1, part 2 or part 3 is the object of the statement. The list includes all organizations that have submitted informal statements. However, if no "X" is present, no formal statement has yet been received from that organization.

Company	ISO/IEC 14496 -2	ISO/IEC 14496 -3	ISO/IEC 14496 -1	ISO/IEC 14496 -7

Annex C VQ Tables

VQ quantizer tables listed below are given in a separate file w2203pvq.

LSP quantizer table - 2k	CbLsp
LSP quantizer table - 4k	CbLsp4k
harmonic VQ table - 2k	CbAm
harmonic VQ table - 4k	CbAm4k
stochastic codebook table - 2k	CbCelp
stochastic codebook table - 4k	CbCelp4k

Annex D Parametric system layer

```
PARAbitstream() {
        PARAconfig()
        while (1) {
                PARAframe()
        }
}
```

Organisation of the document

MPEG-4 Audio comprises 6 subparts:

Subpart 1: Main

Subpart 2: Parametric Coding

Subpart 3: CELP Coding

Subpart 4: Time/Frequency Coding

Subpart 5: Structured Audio

Subpart 6: Text to Speech

For reasons of managability of large documents, the CD is divided in several files:

1. w2203: The master file, containing general introduction and description of tools for other functionalities (this file)
2. w2203par: Contains the description of Subpart 2.
3. w2203cel: Contains the description of Subpart 3.
4. w2203tft: Contains the tool descriptions of Subpart 4.
5. w2203tfs: Contains the syntax part of Subpart 4.
6. w2203tfa: Contains the informative part of Subpart 4.
7. w2203sa: Contains the description of Subpart 5.
8. w2203tts: Contains the description of Subpart 6.

For each of the five coding schemes as well as for the tools for other functionalities, there is a section containing a normative part (description of the syntax and the decoding process) and an informative annex with encoder and interface description. In addition there are two files containing very large tables. These are:

w2203tvq Twin-VQ vector quantizer tables (Subpart 4).

w2203pvq Parametric coder vector quantizer tables (Subpart 2).

1 Introduction

MPEG-4 audio coding integrates the worlds of speech and high quality audio coding as well as the worlds of sound synthesis and the representation of natural audio. The sound synthetis part is comprised of tools for the realisation of symbolically defined music and speech. This includes MIDI and Text-to-Speech systems. Furthermore, tools for effects processing and 3-D localisation of sound are included, allowing the creation of artificial sound environments using artificial and natural sources.

Synthetic audio is described by first defining a set of 'instrument' modules that can create and process audio signals under the control of a script or score file. An instrument is a small network of signal processing primitives that can emulate the effects of a natural acoustic instrument. A script or score is a time-sequenced set of commands that invokes various instruments at specific times to contribute their output to an overall music performance. Other instruments, serving the function of effects processors (reverberators, spatialisers, mixers), can be similarly invoked to receive and process the outputs of the performing instruments. These actions can not only realise a music composition but can also organise any other kind of audio, such as speech, sound effects and general ambience. Likewise, the audio sources can themselves be natural sounds, perhaps emanating from an audio channel decoder, thus enabling synthetic and natural sources to be merged with complete timing accuracy.

TTS is becoming a rather common interface and plays an important role in various multi-media application areas. For instance, by using TTS functionality, multi-media contents with narration can be easily composed without recording natural speech sound. Moreover, TTS with FA/AP/MP functionality would possibly make the contents much richer. In MPEG-4 activity, common interfaces for TTS and TTS for FA/AP/MP are proposed. The proposed MPEG-4 TTS functionality; Hybrid/Multi-Level Scalable TTS, can be considered as a superset of the conventional TTS framework. This extended TTS can utilize prosodic information of natural speech in addition to input texts and can generate much higher quality of synthetic speech. The interface and its bitstream format is strongly scalable; for example, if some parameters of prosodic information are not available, it then generates the missing parameters by rule. Still the basic idea for the scalable MPEG-4 TTS interface is it can fully utilize all the provided information according to the level of user's requirements. The functionality of this extended TTS thus ranges from conventional TTS to natural speech coding and its application areas, from simple TTS to AP with TTS and MP dubbing with TTS.

MPEG-4 standardises natural audio coding at bitrates ranging from 2 kbit/s up to 64 kbit/s. The presence of the MPEG-2 AAC standard within the MPEG-4 tool set will provide for compression of general audio. For the bitrates from 2 kbit/s up to 64 kbit/s, the MPEG-4 standard normalises the bitstream syntax and decoding processes in terms of a set of tools. In order to achieve the highest audio quality within the full range of bitrates and at the same time provide the extra functionalities, three types of coder have been defined. The lowest bitrate range between about 2 and 6 kbit/s, mostly used for speech coding at 8 kHz sampling frequency, is covered by parametric coding techniques. Coding at the medium bitrates between about 6 and 24 kbit/s uses Code Excited Linear Predictive (CELP) coding techniques. In this region, two sampling rates, 8 and 16 kHz, are used to support a broader range of audio signals (other than speech). For the bitrates typically starting at about 16 kbit/s, time to frequency coding techniques are applied. The audio signals in this region typically have bandwidths starting at 8 kHz.

A number of functionalities are provided to facilitate a wide variety of applications which could range from intelligible speech to high quality multichannel audio. Examples of the functionalities are speed control, pitch change, error resilience and scalability in terms of bitrate, bandwidth, error robustness, complexity, etc. as defined below. These functionalities are applicable to the individual coding schemes (parametric, CELP and t/f) as well as across the coding schemes.

- The speed change functionality allows the change of the time scale without altering the pitch during the decoding process. This can, for example, be used to implement a "fast forward" function (data base search) or to adapt the length of an audio sequence to a given video sequence.
- The pitch change functionality allows the change of the pitch without altering the time scale during the encoding or decoding process.This can be used for example for voice alteration or Karaoke type applications.

Bitrate scalability allows a bitstream to be parsed into a bitstream of lower bitrate such that the combination can still be decoded into a meaningful signal. The bit stream parsing can occur either during transmission or in the decoder.

- Bandwidth scalability is a particular case of bitrate scalability, whereby part of a bitstream representing a part of the frequency spectrum can be discarded during transmission or decoding.
- Encoder complexity scalability allows encoders of different complexity to generate valid and meaningful bitstreams.
- Decoder complexity scalability allows a given bitstream to be decoded by decoders of different levels of complexity. The audio quality, in general, is related to the complexity of the encoder and decoder used.
- Error robustness provides the ability for a decoder to avoid or conceal audible distortion caused by transmission errors.

To allow for smooth transitions between the bitrates and to allow for bitrate and bandwidth scalability, a general framework has been defined. This is illustrated in figure 1.

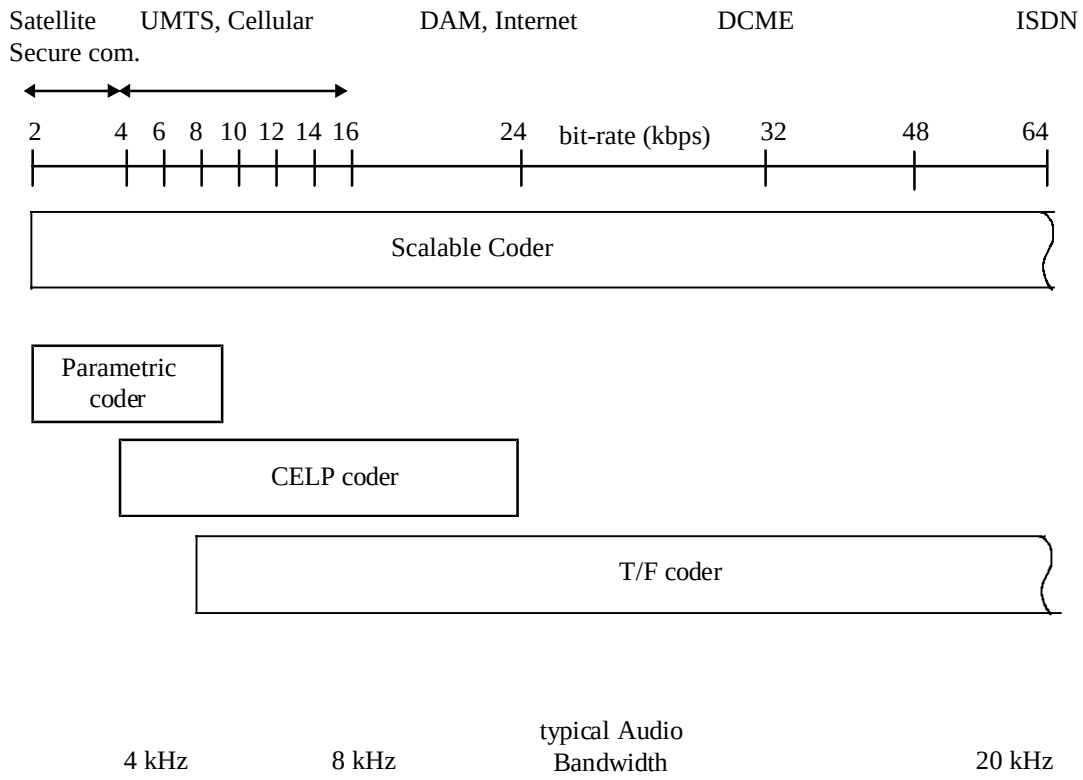


Figure 1

Starting with a coder operating at a low bitrate, by adding enhancements both the coding quality as well as the audio bandwidth can be improved. These enhancements are realised within a single coder or alternatively by combining different techniques.

Additional functionalities are realised both within individual coders, and by means of additional tools around the coders. An example of a functionality within an individual coder is pitch change within the parametric coder.

1.1 Glossary

For the purposes of this International Draft Standard, the following definitions apply. If specific to a part, this is noted in square brackets.

1. alias: Mirrored signal component resulting from sampling.

2. analysis filterbank: Filterbank in the encoder that transforms a broadband PCM audio signal into a set of spectral coefficients.

3. ancillary data: Part of the bitstream that might be used for transmission of ancillary data.

4. audio buffer: A buffer in the system target decoder (see ISO/IEC 13818-1) for storage of compressed audio data.

5. Bark: The Bark is the standard unit corresponding to one critical band width of human hearing.

6. backward compatibility: A newer coding standard is backward compatible with an older coding standard if decoders designed to operate with the older coding standard are able to continue to operate by decoding all or part of a bitstream produced according to the newer coding standard.

7. bitrate: The rate at which the compressed bitstream is delivered to the input of a decoder.

8. bitstream; stream: An ordered series of bits that forms the coded representation of the data.

9. bitstream verifier: A process by which it is possible to test and verify that all the requirements specified in ISO/IEC 13818-7 are met by the bitstream.

10. block companding: Normalising of the digital representation of an audio signal within a certain time period.

11. byte aligned: A bit in a coded bitstream is byte-aligned if its position is a multiple of 8-bits from either the first bit in the stream for the Audio_Data_Interchange_Format (see 1.1) or the first bit in the syncword for the Audio_Data_Transport_Stream Format (see 1.2).

12. byte: Sequence of 8-bits.

13. centre channel: An audio presentation channel used to stabilise the central component of the frontal stereo image.

14. channel: A sequence of data representing an audio signal intended to be reproduced at one listening position.

15. coded audio bitstream: A coded representation of an audio signal.

16. coded representation: A data element as represented in its encoded form.

17. compression: Reduction in the number of bits used to represent an item of data.

18. constant bitrate: Operation where the bitrate is constant from start to finish of the coded bitstream.

19. CRC: The Cyclic Redundancy Check to verify the correctness of data.

20. critical band: This unit of bandwidth represents the standard unit of bandwidth expressed in human auditory terms, corresponding to a fixed length on the human cochlea. It is approximately equal to 100 Hz at low frequencies and 1/3 octave at higher frequencies, above approximately 700 Hz.

21. data element: An item of data as represented before encoding and after decoding.

22. de-emphasis: Filtering applied to an audio signal after storage or transmission to undo a linear distortion due to emphasis.

23. decoded stream: The decoded reconstruction of a compressed bitstream.

24. decoder: An embodiment of a decoding process.

25. decoding (process): The process defined in ISO/IEC 13818 part 7 that reads an input coded bitstream and outputs decoded audio samples.

26. digital storage media; DSM: A digital storage or transmission device or system.

27. discrete cosine transform; DCT: Either the forward discrete cosine transform or the inverse discrete cosine transform. The DCT is an invertible, discrete orthogonal transformation.

28. downmix: A matrixing of n channels to obtain less than n channels.

29. editing: The process by which one or more coded bitstreams are manipulated to produce a new coded bitstream. Conforming edited bitstreams must meet the requirements defined in part 7 of ISO/IEC 13818.

30. emphasis: Filtering applied to an audio signal before storage or transmission to improve the signal-to-noise ratio at high frequencies.

31. encoder: An embodiment of an encoding process.

32. encoding (process): A process, not specified in ISO/IEC 13818, that reads a stream of input audio samples and produces a valid coded bitstream as defined in part 7 of ISO/IEC 13818.

33. entropy coding: Variable length lossless coding of the digital representation of a signal to reduce statistical redundancy.

34. FFT: Fast Fourier Transformation. A fast algorithm for performing a discrete Fourier transform (an orthogonal transform).

35. filterbank: A set of band-pass filters covering the entire audio frequency range.

36. flag: A variable which can take one of only the two values defined in this specification.

37. forward compatibility: A newer coding standard is forward compatible with an older coding standard if decoders designed to operate with the newer coding standard are able to decode bitstreams of the older coding standard.

38. Fs: Sampling frequency.

39. Hann window: A time function applied sample-by-sample to a block of audio samples before Fourier transformation.

40. Huffman coding: A specific method for entropy coding.

41. hybrid filterbank: A serial combination of subband filterbank and MDCT.

42. IDCT: Inverse Discrete Cosine Transform.

43. IMDCT: Inverse Modified Discrete Cosine Transform.

44. intensity stereo: A method of exploiting stereo irrelevance or redundancy in stereophonic audio programmes based on retaining at high frequencies only the energy envelope of the right and left channels.

45. joint stereo coding: Any method that exploits stereophonic irrelevance or stereophonic redundancy.

46. joint stereo mode: A mode of the audio coding algorithm using joint stereo coding.

47. low frequency enhancement (LFE) channel: A limited bandwidth channel for low frequency audio effects in a multichannel system.

48. main audio channels: All single_channel_elements (see clause 3.2.1) or channel_pair_elements (see clause 3.2.1) in one program.

49. mapping: Conversion of an audio signal from time to frequency domain by subband filtering and/or by MDCT.

50. masking: A property of the human auditory system by which an audio signal cannot be perceived in the presence of another audio signal.

51. masking threshold: A function in frequency and time below which an audio signal cannot be perceived by the human auditory system.

52. modified discrete cosine transform (MDCT): A transform which has the property of time domain aliasing cancellation. An analytical espression for the MDCT can be found in clause B 2.3.1.2.

53. M/S stereo: A method of removing imaging artefacts as well as exploiting stereo irrelevance or redundancy in stereophonic audio programmes based on coding the sum and difference signal instead of the left and right channels.

54. multichannel: A combination of audio channels used to create a spatial sound field.

55. multilingual: A presentation of dialogue in more than one language.

56. non-tonal component: A noise-like component of an audio signal.

57. Nyquist sampling: Sampling at or above twice the maximum bandwidth of a signal.

58. padding: A method to adjust the average length of an audio frame in time to the duration of the corresponding PCM samples, by conditionally adding a slot to the audio frame.

59. parameter: A variable within the syntax of this specification which may take one of a range of values. A variable which can take one of only two values is a flag or indicator and not a parameter.

60. parser: Functional stage of a decoder which extracts from a coded bitstream a series of bits representing coded elements.

61. polyphase filterbank: A set of equal bandwidth filters with special phase interrelationships, allowing for an efficient implementation of the filterbank.

62. prediction error: The difference between the actual value of a sample or data element and its predictor.

63. prediction: The use of a predictor to provide an estimate of the sample value or data element currently being decoded.

64. predictor: A linear combination of previously decoded sample values or data elements.

65. presentation channel: An audio channel at the output of the decoder.

66. program: A set of main audio channels, coupling_channel_elements (see clause 3.2.1), lfe_channel_elements (see clause 3.2.1), and associated data streams intended to be decoded and played back simultaneously A program may be defined by default (see clause 3.5.1) or specifically by a program_configuration_element (see clause 3.2.1). A given single_channel_element (see clause 3.2.1) , channel_pair_element (see clause 3.2.1), coupling_channel_element, lfe_channel_element or data channel may accompany one or more programs in any given bitstream..

67. psychoacoustic model: A mathematical model of the masking behaviour of the human auditory system.

68. random access: The process of beginning to read and decode the coded bitstream at an arbitrary point.

69. reserved: The term "reserved" when used in the clauses defining the coded bitstream indicates that the value may be used in the future for ISO/IEC defined extensions.

70. Sampling Frequency (Fs): Defines the rate in Hertz which is used to digitise an audio signal during the sampling process.

71. scalefactor: Factor by which a set of values is scaled before quantization.

72. scalefactor band: A set of spectral coefficients which are scaled by one scalefactor.

73. scalefactor index: A numerical code for a scalefactor.

74. side information: Information in the bitstream necessary for controlling the decoder.

75.spectral coefficients: Discrete frequency domain data output from the analysis filterbank.

76. spreading function: A function that describes the frequency spread of masking effects.

77. stereo-irrelevant: A portion of a stereophonic audio signal which does not contribute to spatial perception.

78. stuffing (bits); stuffing (bytes): Code-words that may be inserted at particular locations in the coded bitstream that are discarded in the decoding process. Their purpose is to increase the bitrate of the stream which would otherwise be lower than the desired bitrate.

79. surround channel: An audio presentation channel added to the front channels (L and R or L, R, and C) to enhance the spatial perception.

80. syncword: A 12-bit code embedded in the audio bitstream that identifies the start of a adts_frame() (see 1.2, Table 6.4).

81. synthesis filterbank: Filterbank in the decoder that reconstructs a PCM audio signal from subband samples.

82. tonal component: A sinusoid-like component of an audio signal.

83. variable bitrate: Operation where the bitrate varies with time during the decoding of a coded bitstream.

84. variable length coding: A reversible procedure for coding that assigns shorter code-words to frequent symbols and longer code-words to less frequent symbols.

85. variable length code (VLC): A code word assigned by variable length encoder (See variable length coding)

86. variable length decoder: A procedure to obtain the symbols encoded with a variable length coding technique.

87. variable length encoder: A procedure to assign variable length codewords to symbols.

1.2 Symbols and abbreviations

The mathematical operators used to describe this Recommendation | International Standard are similar to those used in the C programming language. However, integer division with truncation and rounding are specifically defined. The bitwise operators are defined assuming twos-complement representation of integers. Numbering and counting loops generally begin from zero.

1.2.1 Arithmetic operators

+	Addition.
–	Subtraction (as a binary operator) or negation (as a unary operator).
++	Increment.
– –	Decrement.
*	Multiplication.
^	Power.
/	Integer division with truncation of the result toward zero. For example, 7/4 and –7/–4 are truncated to 1 and –7/4 and 7/–4 are truncated to –1.
//	Integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example 3//2 is rounded to 2, and –3//2 is rounded to –2.
DIV	Integer division with truncation of the result towards –∞.
| |	Absolute value. | x | = x when x > 0 | x | = 0 when x == 0 | x | = –x when x < 0
%	Modulus operator. Defined only for positive numbers.
Sign( )	Sign. Sign(x) = 1 when x > 0 Sign(x) = 0 when x == 0 Sign(x) = –1 when x < 0
INT ( )	Truncation to integer operator. Returns the integer part of the real-valued argument.
NINT ( )	Nearest integer operator. Returns the nearest integer value to the real-valued argument. Half-integer values are rounded away from zero.
sin	Sine.
cos	Cosine.
exp	Exponential.

$\sqrt{}$	Square root.
$\log_{10}$	Logarithm to base ten.
$\log_e$	Logarithm to base e.
$\log_2$	Logarithm to base 2.

1.2.2 Logical operators

||	Logical OR.
&&	Logical AND.
!	Logical NOT

1.2.3 Relational operators

>	Greater than.
>=	Greater than or equal to.
<	Less than.
<=	Less than or equal to.
==	Equal to.
!=	Not equal to.
max [,...,]	the maximum value in the argument list.
min [,...,]	the minimum value in the argument list.

1.2.4 Bitwise operators

A twos complement number representation is assumed where the bitwise operators are used.

&	AND
|	OR
>>	Shift right with sign extension.
<<	Shift left with zero fill.

1.2.5 Assignment

=	Assignment operator.

1.2.6 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit stream.

bslbf	Bit string, left bit first, where "left" is the order in which bit strings are written in ISO/IEC 11172. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. ’1000 0001’. Blanks within a bit string are for ease of reading and have no significance.
L, C, R, LS, RS	Left, center, right, left surround and right surround audio signals
rpchof	Remainder polynomial coefficients, highest order first. (Audio)
uimsbf	Unsigned integer, most significant bit first.
vlclbf	Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written.
window	Number of the actual time slot in case of block_type==2, 0 <= window <= 2. (Audio)

The byte order of multi-byte words is most significant byte first.

1.2.7 Constants

π 3,14159265358...

e 2,71828182845...

1.3 Method of describing bit stream syntax

The bit stream retrieved by the decoder is described in 1 (Syntax). Each data item in the bit stream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bit stream depends on the value of that data element and on data elements previously decoded. The decoding of the data elements and the definition of the state variables used in their decoding are described in 3.3.. The following constructs are used to express the conditions when data elements are present, and are in normal type:

Note this syntax uses the 'C'-code convention that a variable or expression evaluating to a non-zero value is equivalent to a condition that is true.

```
while ( condition ) {
  data_element
  . . .
}
```

If the condition is true, then the group of data elements occurs next in the data stream. This repeats until the condition is not true.

```
do {
  data_element
  . . .
} while ( condition )
```

The data element always occurs at least once. The data element is repeated until the condition is not true.

```
if ( condition) {
  data_element
  . . .
}
```

If the condition is true, then the first group of data elements occurs next in the data stream

```
else {
  data_element
  . . .
}
```

If the condition is not true, then the second group of data elements occurs next in the data stream.

```
for (expr1; expr2; expr3) {
  data_element
  . . .
}
```

Expr1 is an expression specifying the initialisation of the loop. Normally it specifies the initial state of the counter. Expr2 is a condition specifying a test made before each iteration of the loop. The loop terminates when the condition is not true. Expr3 is an expression that is performed at the end of each iteration of the loop, normally it increments a counter.

Note that the most common usage of this construct is as follows:

```
for ( i = 0; i < n; i++) {
  data_element
  . . .
}
```

The group of data elements occurs n times. Conditional constructs within the group of data elements may depend on the value of the loop control variable i, which is set to zero for the first occurrence, incremented to one for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the {} may be omitted when only one data element follows.

data_element [ ]	data_element [ ] is an array of data. The number of data elements is indicated by the context.
data_element [n]	data_element [n] is the n+1th element of an array of data.
data_element [m][n]	data_element [m][n] is the m+1,n+1 th element of a two-dimensional array of data.
data_element [l][m][n]	data_element [l][m][n] is the l+1,m+1,n+1 th element of a three-dimensional array of data.
data_element [m..n]	data_element [m..n]is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that clause x.x.x implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bit stream. Actual decoders must include a means to look for start codes in order to begin decoding correctly.

Definition of bytealigned function

The function bytealigned ( ) returns 1 if the current position is on a byte boundary, that is the next bit in the bit stream is the first bit in a byte. Otherwise it returns 0.

Definition of nextbits function

The function nextbits ( ) permits comparison of a bit string with the next bits to be decoded in the bit stream.

2 Technical Overview

to be added

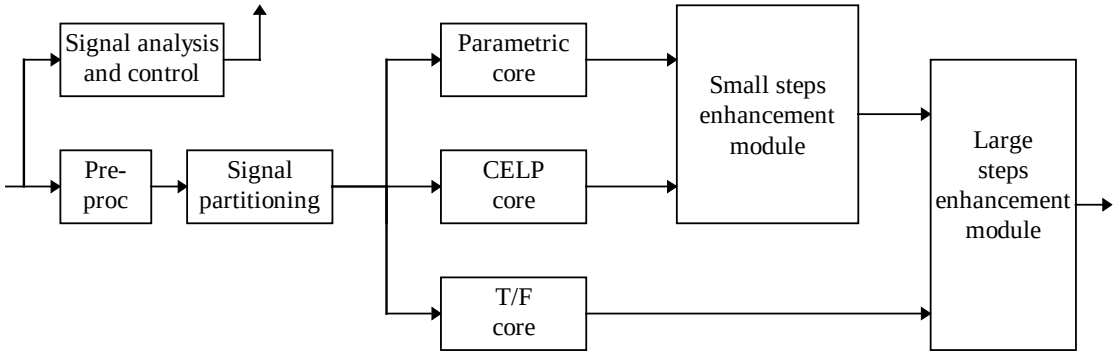


Figure 2

2.1 MPEG-4 Audio Object Profiles and Levels

2.1.1 Object Profiles

Profile	Hierarchy	Tools supported:	Object Profile ID
reserved			0
AAC Main	contains AAC LC	13818-7 main profile PNS	1
AAC LC		13818-7 LC profile PNS	2
AAC SSR		13818-7 SSR profile PNS	3
T/F		13818-7 LC PNS LTP	4
T/F Main scalable	contains T/F LC scalable	13818-7 main PNS LTP BSAC tools for large step scalability (TLSS) core codecs: CELP, TwinVQ, HILN	5
T/F LC scalable		13818-7 LC PNS LTP BSAC tools for large step scalability (TLSS) core codecs. CELP, TwinVQ, HILN	6
TwinVQ core		TwinVQ	7
CELP		CELP	8
HVXC		HVXC	9
HILN		HILN	10
TTSI		Text-To-Speech Interface	11
Main Synthetic	contains Wavetable Synthesis	all structured audio tools	12
Wavetable Synthesis		SASBF MIDI	13
reserved			14
reserved			15

2.1.2 Combination Profiles

[make reference to systems 7.3.3.3 DecoderConfigDescriptor, profileAndLevelIndication Values !]

Combination Profile	Hierarchy	Audio Object Profiles supported:

Main	Contains Scalable, Speech and Low Rate Synthetic	AAC Main, LC, SSR T/F, T/F Main Scalable, T/F LC Scalable TwinVQ core CELP HVXC HILN Main Synthetic TTSI
Scalable	Contains Speech	T/F LC Scalable AAC-LC or/and T/F CELP HVXC TwinVQ core HILN Wavetable Synthesis TTSI
Speech		CELP HVXC TTSI
Low Rate Synthesis		Wavetable Synthesis TTSI

2.1.3 Complexity Units

Complexity units are defined to give an approximation of the decoder complexity in terms of processing power and RAM usage required for processing MPEG-4 Audio bitstreams in dependence of specific parameters.

The approximated processing power is given in “Processor Complexity Units” (PCU), specified in integer numbers of MOPS. The approximated RAM usage is given in “RAM Complexity Units” (RCU), specified in (mostly) integer numbers of kWords (1000 words). The RCU numbers do not include working buffers which can be shared between different objects and/or channels. Total complexity estimates for single-object / single-channel decoders which also include ROM requirements are given in Section 2.2.

If a level of a profile is specified by the maximum number of complexity units, a flexible configuration of the decoder handling different types of objects is allowed under the constraint that both values for the total complexity for decoding and sampling rate conversion (if needed) do not exceed this limit.

The following table gives complexity estimates for the different object profiles:

Object Profile	Parameters	PCU (MOPS)	RCU (kWords)	Remarks
AAC Main	fs = 48 kHz	5	5	1)
AAC LC	fs = 48 kHz	3	3	1)
AAC SSR	fs = 48 kHz	4	3	1)
T/F	fs = 48 kHz	4	3	1)
T/F Main Scalable	fs = 48 kHz	6	5	1), 2)

T/F LC Scalable	fs = 48 kHz	5	3	1), 2)
TwinVQ core	fs = 24 kHz	2	3	1)
CELP	fs = 8 kHz	1	1	
CELP	fs = 16 kHz	2	1	
CELP	fs = 8/16 kHz (bandwidth scalable)	4	1	
HVXC	fs = 8 kHz	2	1	
HILN	fs = 8 kHz, ns = 40	5	2	1), 3)
HILN	fs = 8 kHz, ns = 90	10	2	1), 3)
TTSI		-	-	4)
Wavetable Synthesis	fs= ???, nt = ???	under study	under study, 5)	
Main Synthetic		under study	under study	
Sampling Rate Conversion	rf = 2, 3, 4, 6	2	(.5)	

Definitions:

- fs = sampling frequency
- ns = max. number of sinusoids to be synthesized
- nt = max. number of tones to be synthesized simultaneously
- rf = ratio of sampling rates

Notes:

1) PCU proportional to sampling frequency
2) Includes core decoder
3) RCU proportional to sampling frequency
4) The complexity for speech synthesis is not taken into account
5) Size of wavetable

2.1.4 Levels within the Combination Profiles

Levels for Main Combination Profile

Four levels are defined by complexity units:

1. PCU < 40, RCU < 20
2. PCU < 80, RCU < 64
3. PCU < 160, RCU < 128
4. PCU < 320, RCU < 64000

Levels for Scalable Combination Profile

Four levels are defined by configuration, the fourth level is defined by complexity units:

1. 24kHz 1 channel or 1 object (all object profiles) or 2 objects (overlay of a speech object and one low complexity SA or HILN object at 16 kHz)
2. 24kHz 2 channels or 2 objects
3. 48kHz 2 channels or 2 objects (48 kHz t/f or speech coding)

4. 48 kHz/24 kHz 5 channels or multiple objects, max. one integer factor sampling rate conversion for a maximum of two channels. Flexible configuration is allowed. PCU < 30, RCU < 19.

Levels for Speech Combination Profile

Only one level is defined for one speech object.

Levels for Low Rate Synthesis Combination Profile

Two levels are defined by the maximum number of tones to be synthesized simultaneously, the size of the sample RAM and by the maximum number of TTS objects which can be transmitted:

1. max. 8 tones to be synthesized simultaneously, 64 kWords of sample RAM
   max. 2 TTS objects
2. max. 24 tones to be synthesized simultaneously, 256 kWords of sample RAM
   max. 8 TTS objects

2.2 Complexity Figures for the Natural Audio Coding Algorithms

Overview of the complexity of the MPEG-4 Natural Audio Coding Algorithms

Complexity	AAC	AAC-LC	BSAC	Twin VQ	HILN	HVXC	NB-CELP	WB-CELP
Memory Requirements for 1 Audio Channel and Minimum Word Length								
RAM (Words)	4256	2232	4346	4240	3000	1500	650	830
ROM (Words)	3545	3545	3618	43000	4000	7700	2300	1000
min. Word Length	>=20	>=20	>=20	>=20	16	16	16	24 (16)
Computational Complexity calculated for a Sampling Rate of								
kHz	48	48	48	48	8	8	8	16
MOPS/ MIPS	5	2.5	5.7	3	4 typ. 10 max	5.1	1.5	3

The above numbers relate to a decoder for one audio channel. For a multi-channel system the numbers given for the RAM size and the computational complexity have to be multiplied with the number of audio channels to get a first estimation. The figures given for AAC have been calculated during the MPEG-2 AAC standardization process. The Wide-Band CELP figures are derived from an implementation on a Motorola DSP 56002. The figures given for the computational complexity may have been calculated with different methods, and may therefore not be directly comparable, but only give a first impression. Also, all the figures are not guaranteed to be the absolute maximum.

3 Interface to MPEG-4 Systems

3.1 Syntax

3.1.1 Audio DecoderSpecificInfo

In case that DecoderConfigDescriptor() (see ISO/IEC 14496-1 MPEG4 Systems) is used for MPEG-4 audio decoders, the array specificInfoByte[] shall contain AudioSpecificInfo() defined as follows:

```
aligned (8) class AudioSpecificConfig() {
        uint(4) objectProfile; /* see also Clause 2.1.1, „Object Profiles“ in this document; remark :
                                  composition profile and  level are already signaled by the
                                 profileAndLevelIndication value in DecoderConfigDescriptor */
        uint(4) samplingFrequencyIndex;
        if (samplingFrequencyIndex==0xf){
                uint(24) samplingFrequency;/* in Hz */
        }

        uint(4) channelConfiguration;

        if (objectProfile<8) { /* this is T/F  (AAC/TwinVq)*/
                TFSpecificConfig tfConfig(  uint(4) samplingFrequencyIndex);
        }
        if (objectProfile==8) { /* this is Celp */
                CelpSpecificConfig celpConfig( uint(4) samplingFrequencyIndex);
        }
        if (objectProfile==9 or 10 ) { /* this is Parametric , HVXC or HILN*/
                ParametricSpecificConfig paramConfig( );
        }
        if (objectProfile==11) { /* this is TextToSpeech */
                TTSSpecificConfig ttsConfig( );
        }
        if (objectProfile==12 or 13) { /* this is structured audio , Main Synthetic or Wavetable
                                          Synthesis*/
                StructuredAudioSpecificConfig strucConfig( );
        }

}
```

3.1.2 ParametricSpecificConfig

Defined in subpart 2.

3.1.3 CelpSpecificConfig

Defined in subpart 3.

3.1.4 TFSpecificConfig

Defined in subpart 4.

3.1.5 StructuredAudioSpecificConfig

Defined in subpart 5.

3.1.6 TTSSpecificConfig

Defined in subpart 6.

3.1.7 Payloads

The actual payloads for each configuration are defined in the corresponding subparts. These are the basic entities to be carried by the systems transport layer.

3.2 Semantics

3.2.1 objectProfile

A four bit field indicating the object profile. This is the master switch which selects the actual bit stream syntax of the audio data. In general, different object profiles use a different bit stream syntax. The interpretation of this field is given in the Object Profiles table above.

3.2.2 SamplingFrequencyIndex

A four bit field indicating the sampling rate used.

Value	samplingFrequencyIndex
0x0	96000
0x1	88200
0x2	64000
0x3	48000
0x4	44100
0x5	32000
0x6	24000
0x7	22050
0x8	16000
0x9	12000
0xa	11025
0xb	8000
0xc	reserved
0xd	reserved
0xe	reserved
0xf	escape value

3.2.3 channelConfiguration

A four bit field indicating the channel configuration

value	number of channels	channel to speaker mapping
0	-	defined in audioDecderSpecificConfig
1	1	center front speaker

2	**2**	**left, right front speakers**
3	**3**	**center front speaker,** **left, right front speakers**
4	**4**	**center front speaker,** **left, right center front speakers,** **rear surround speakers**
5	**5**	**center front speaker,** **left, right front speakers,** **left surround, right surround rear speakers**
6	**5+1**	**center front speaker,** **left, right front speakers,** **left surround, right surround rear speakers,** **front low frequency effects speaker**
7	**7+1**	**center front speaker** **left, right center front speakers,** **left, right outside front speakers,** **left surround, right surround rear speakers,** **front low frequency effects speaker**
8-15	**-**	**reserved**

4 Tools for Other Functionalities

This section describes the functionalities that are applicable across the natural audio coding schemes (parametric, CELP and t/f). Functionalities that are applicable to the individual coding schemes can be found in the corresponding file.

4.1 Speed change

4.1.1 Tool description

PICOLA (Pointer Interval Controlled OverLap Add) speed control tool supports speed change functionality for mono-channel signals sampled at 8kHz or 16kHz. The speed control is achieved by replacing a part of the input signal with an overlap-added waveform or by inserting the overlap-added waveform into the input signal.

4.1.2 Definitions

InputSignal[]: This array contains the input signal from the decoder.

NumInputSignal: This field contains the number of samples of the input signal InputSignal[].

OutputSignal[]: This array contains the speed-changed signal and is of a fixed length.

SpeedControlFactor: This field contains the speed control ratio. (SpeedControlFactor = 1.0 indicates normal speed.)

DecodeFlag: This field contains the flag which indicates whether the input signal of the next frame is required for speed change processing.

DecodeFlag	Description
0	Input signal of the next frame is not required
1	Input signal of the next frame is required

OutputFlag: This field contains the flag which indicates whether the output array OutputSignal[] contains speed-changed samples, the size of which is less than the fixed output array size, the outstanding difference being filled by the next speed-changed signal.

OutputFlag	Description
0	Speed-changed signal has size less than the fixed-size output array
1	Speed-changed signal has size equal to the fixed-size output array and therefore be outputted.

4.1.3 Speed control process

The block diagram of the speed controller is shown in Figure 1. The input signal InputSignal[], which is an output from the decoder with a given frame length NumInputSample, is stored in the buffer memory. Adjacent waveforms with the same length are extracted in pairs from the memory buffer and the pair with the minimum distortion between the two waveforms is selected. The selected waveforms are overlap-added. The speed control is achieved by replacing a part of the input signal with the overlap-added waveform or by inserting the overlap-added waveform into the input signal. The speed controller outputs the speed changed signal with a certain fixed length frame. Normally, the frame length is to be the same length as that of the associated decoder. In the case where the frame length of the associated decoder is variable, the maximum possible frame length is applied. Since the number of samples of the

output signal differs from that of the input signal, the input and the output are controlled using flags, namely, the DecodeFlag and OutputFlag. In the case where the samples of the input signal in the buffer are not enough to carry out the speed change, DecodeFlag is set to 1 and the decoded signal from the associated decoder is inputted. On the other hand, in the case where samples of the speed-changed signal has size equal to the fixed-size output array OutputSignal[], OutputFlag is set to 1 and the speed-changed signal is outputted. Details of the processing are described below.

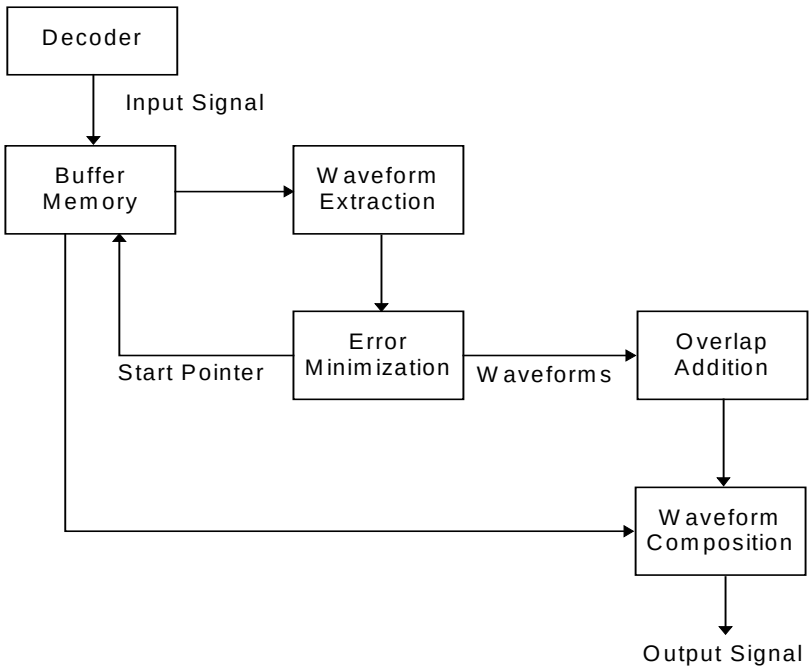


Figure 1. Block Diagram of the Speed Controller

4.1.3.1 Time scale compression (High speed replay)

The compression principle is shown in Figure 2. P0 is the pointer which indicates the starting sample of the current processing frame in the memory buffer. The processing frame has a length of LF samples and comprises adjacent waveforms each of length LW samples. The average distortion per sample between the first half of the processing frame (waveform A) and the second half (waveform B) is calculated as shown below.

$$\mathrm{D}(\mathrm{LW}) = \frac{1}{\mathrm{LW}} \sum_{n=0}^{\mathrm{LW}-1} \{x(n) - y(n)\}^2 \quad (\mathrm{PMIN} \le \mathrm{LW} \le \mathrm{PMAX})$$

where, D(LW) is the average distortion between the two waveforms when the waveform length is LW, $x(n)$ is the waveform A, $y(n)$ is the waveform B, PMIN is the minimum length of the candidate waveform and PMAX is the maximum length of the candidate waveform. Typically PMIN=32 and PMAX=160 for 8kHz sampling rate, PMIN=80 and PMAX=320 for 16kHz sampling.

The length LW which minimizes the distortion D(LW) is selected, and corresponding waveforms A and B are determined. If the cross-correlation between the selected waveforms A and B is negative, the pointer P0 is shifted forward by the frame length of the decoder and the length LW is determined again with the processing frame starting from the updated pointer P0. After the waveform length LW is

determined, the waveform A is windowed by a half triangular window with a descending slope, and the waveform B is windowed by a half triangular window with an ascending slope. The overlap-added waveform C is obtained by linearly adding the windowed waveform A and waveform B. Then, the pointer P0 moves to the point P1. The distance L from the beginning of waveform C to the pointer P1 is given by;

$$L = LW \bullet \frac{1}{SpeedControlFactor - 1} \qquad (1 < SpeedControlFactor \le 2)$$

L samples from the beginning of waveform C are outputted as the compressed signal. If L is greater than LW, the original waveform D which follows the waveform B is outputted. Therefore the length of the signal is shortened from LW+L samples to L samples. The updated pointer P1 indicates the starting sample P0' of the next processing frame.

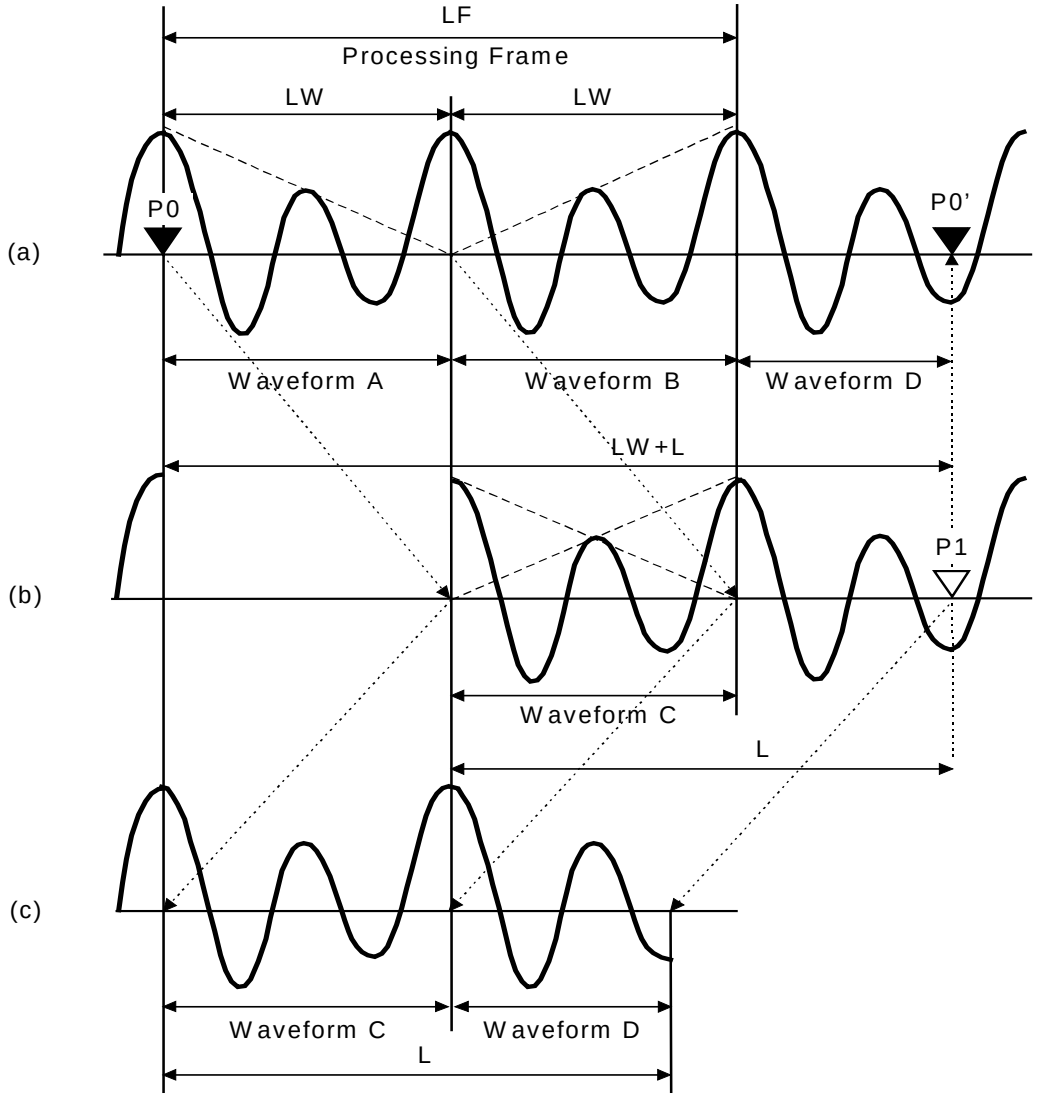


Figure 2. Principle of Time Scale Compression

(a) Orginal signal; (b) Overlap-added waveform; (c) Compressed signal

4.1.3.2 Time scale expansion (Low speed replay)

The expansion principle is shown in Figure 3. P0 is the pointer which indicates the starting sample of the current processing frame in the memory buffer. The processing frame has a length of LF samples and includes adjacent waveforms each of length LW samples. After the waveform length LW is determined using the same method as described in the time scale compression, the first half of the processing frame (waveform A) is outputted without any modification. Next, the first half (waveform A) is windowed by a half triangular window with an ascending slope, and the second half (waveform B) is windowed by a half triangular window with a descending slope. The overlap-added waveform C is obtained by linearly adding the windowed waveform A and waveform B. Then, the pointer P0 moves to the point P1. The distance L from the beginning of waveform C to the pointer P1 is given by;

$$L = LW \bullet \frac{SpeedControlFactor}{1 - SpeedControlFactor} \qquad (0.5 \le SpeedControlFactor < 1)$$

L samples from the beginning of waveform C are outputted as the expanded signal. If L is greater than LW, the original waveform B is repeated as the output. The length of the signal is therefore expanded from L samples to LW+L samples. The updated pointer P1 indicates the starting sample P0' of the next processing frame.

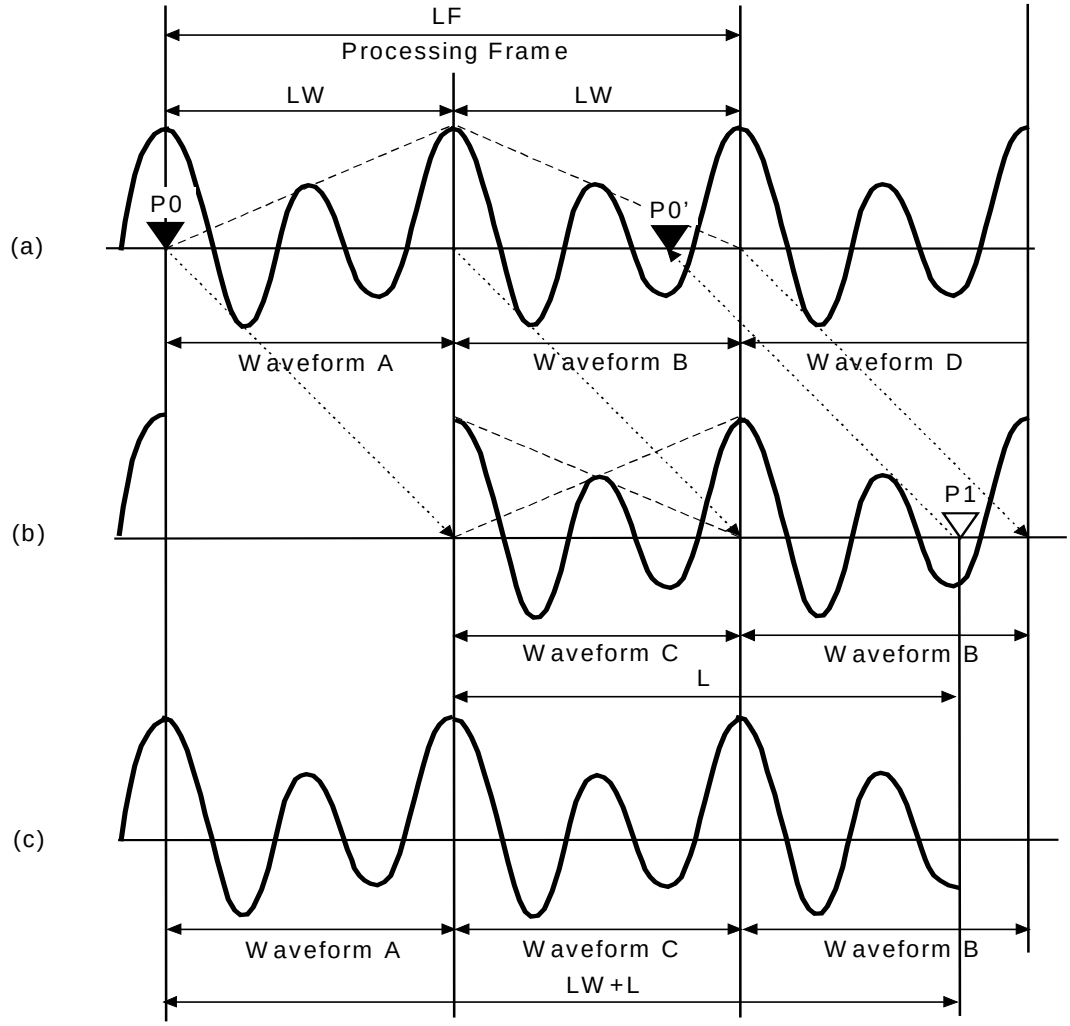


Figure 3. Principle of Time Scale Expansion

(a) Orginal signal; (b) Overlap-added waveform; (c) Expanded signal

Annex A (Informative)

1 PICOLA functional interface description

```
void mod_picola(
   float InputSignal[],                    /* input */
   int NumInputSample,                     /* input */
   float OutputSignal[],                   /* output */
   float SpeedControlFactor,               /* input */
   int *DecodeFlag,                        /* output */
   int *OutputFlag                         /* output */
)
```

Annex B (Informative)

2 Patent statements

(Table of organizations indicating to possess patent/intellectual property related to this specification to be added)

WG11 requests companies who believe they hold rights on patents that are necessary to implement MPEG-4 parts 1-2-3-5-6 to deliver a statements on company letterhead of compliance with ISO policy concerning use of patented items in International Standards. The patent statement can take a form similar to the statement given below:
<Company name> is pleased that the standardisation in relation to "Very low bitrate audio-visual coding" (known as MPEG-4) has reached Committee Draft level as documents ISO/IEC JTC1/SC29/WG11 N1901, N1902, N1903, N1905, N1906.
<Company name> hereby declares that it is prepared to license its patents, both granted and pending, which are necessary to manufacture, use, and sell implementations of the proposed MPEG-4 Systems, Visual, Audio, Reference Software and DMIF standards or combinations thereof.
<Company name> hereby also declares that it is aware of the rules governing inclusion of patented items in international standards, as described by Section 5.7, part 2 of the ISO/IEC Directives, and in particular that it is willing to grant a license to an unlimited number of applicants throughout the world under reasonable terms and conditions that are demonstrably free of any unfair competition.
This statement is intended to apply to the following parts of the proposed MPEG-4 standard (use the ones which apply):
Systems, Visual, Audio, Reference Software, DMIF".

I.

Annex C

VQ codebooks for HVXC

1. CbAm

VQ codebook for harmonic spectral vector for 2kbps

/* SE_Gain */
/* dim= 1 x 32 codewords */

index	codeword	indes	codeword
0	1.4277322769e+01f	16	6.6570297852e+03f
1	2.4427261353e+01f	17	5.5529902344e+03f
2	5.1531036377e+01f	18	4.4632456055e+03f
3	3.8094165802e+01f	19	5.0344755859e+03f
4	1.5629663086e+02f	20	3.1603977051e+03f
5	1.2007367706e+02f	21	3.4228183594e+03f
6	6.9551498413e+01f	22	4.0258488770e+03f
7	9.1061431885e+01f	23	3.6935373535e+03f
8	1.1360032959e+03f	24	1.3403870850e+03f
9	9.4679547119e+02f	25	1.5429980469e+03f
10	6.1508343506e+02f	26	2.0011907959e+03f
11	7.6936199951e+02f	27	1.7811741943e+03f
12	2.0661782837e+02f	28	2.8798361816e+03f
13	2.7738925171e+02f	29	2.6719782715e+03f
14	4.8511575317e+02f	30	2.2384204102e+03f
15	3.7061630249e+02f	31	2.4639086914e+03f

/* SE_Shape1 */
/* dim=44 * 16 codevectors */

index	codeword			
0	5.8819748461e-02f	9.3861304224e-02f	4.7118034214e-02f	1.3120673597e-02f
	4.2679972947e-02f	5.6458037347e-02f	2.1073838696e-02f	1.4127705945e-03f
	1.9078921527e-02f	3.2528404146e-02f	2.9394164681e-02f	2.2569352761e-02f
	3.1202243641e-02f	5.2047207952e-02f	5.0222467631e-02f	3.5644743592e-02f
	3.0297558755e-02f	3.6113645881e-02f	2.5096289814e-02f	1.8328050151e-02f
	2.6043191552e-02f	3.6892917007e-02f	4.1834302247e-02f	4.1009154171e-02f
	4.4162660837e-02f	4.6920023859e-02f	5.2744042128e-02f	4.4227186590e-02f
	3.5669751465e-02f	3.5182271153e-02f	4.0188916028e-02f	2.9835734516e-02f
	1.6929663718e-02f	1.2898490764e-02f	1.9803266972e-02f	2.4855496362e-02f
	3.2566908747e-02f	4.3261632323e-02f	5.6911692023e-02f	6.3303709030e-02f
	5.1197323948e-02f	2.3190757260e-02f	1.8529446796e-02f	7.1556223556e-03f
1	4.8871569335e-02f	7.9412519932e-02f	6.1238162220e-02f	5.5874615908e-02f
	7.0684976876e-02f	5.5689472705e-02f	2.0528776571e-02f	-6.7783694249e-04f
	4.1763554327e-03f	2.1712809801e-02f	3.1134236604e-02f	2.8364202008e-02f
	2.1726056933e-02f	3.3196389675e-02f	4.5456487685e-02f	6.4699374139e-02f
	7.3621042073e-02f	6.7012704909e-02f	4.4837214053e-02f	3.4835200757e-02f
	4.1962504387e-02f	5.5134207010e-02f	6.7201323807e-02f	6.7834749818e-02f
	5.6158803403e-02f	4.7373812646e-02f	4.3273460120e-02f	4.2160362005e-02f
	4.0031205863e-02f	3.9065551013e-02f	4.0149483830e-02f	2.7852423489e-02f
	1.0785039514e-02f	6.7186630331e-03f	1.5545932576e-02f	2.5005336851e-02f
	3.0253017321e-02f	3.0891941860e-02f	3.5267263651e-02f	4.1896276176e-02f
	4.2022921145e-02f	5.3294818848e-02f	9.1553628445e-02f	8.5433095694e-02f

	2.4683391675e-02f	4.2231369764e-02f	4.1787624359e-02f	4.2044684291e-02f
	4.9441475421e-02f	5.0555076450e-02f	4.2335528880e-02f	2.8047479689e-02f
	1.9379884005e-02f	2.3474236950e-02f	3.4558840096e-02f	3.4197043628e-02f
	2.9653949663e-02f	4.5857202262e-02f	4.8962634057e-02f	4.9172062427e-02f
	4.8764143139e-02f	5.6099772453e-02f	5.0366304815e-02f	3.6228150129e-02f
2	3.7603646517e-02f	4.8473093659e-02f	4.8064518720e-02f	3.7903234363e-02f
	3.2496560365e-02f	3.2952822745e-02f	3.7241499871e-02f	5.1853429526e-02f
	8.4321834147e-02f	1.0171223432e-01f	1.1665632576e-01f	1.2236715853e-01f
	9.9999070168e-02f	7.2181835771e-02f	4.0153615177e-02f	2.0574832335e-02f
	1.5615364537e-02f	1.5869993716e-02f	2.6966281235e-02f	3.5173915327e-02f
	2.4136606604e-02f	-5.1817572676e-03f	-6.3364054076e-03f	5.3371610120e-03f
	3.2316930592e-02f	6.9475047290e-02f	8.6175821722e-02f	8.4221631289e-02f
	7.9035773873e-02f	6.8688094616e-02f	5.3259685636e-02f	3.3728487790e-02f
	2.0533466712e-02f	1.5299044549e-02f	1.7260372639e-02f	1.7390608788e-02f
	1.7463522032e-02f	3.1245855615e-02f	3.0911922455e-02f	2.5266775861e-02f
	2.7406129986e-02f	3.0088895932e-02f	2.1932294592e-02f	1.4222909696e-02f
3	1.6033075750e-02f	2.4194544181e-02f	2.4819886312e-02f	2.4123618379e-02f
	2.2787833586e-02f	2.6162726805e-02f	2.6893839240e-02f	2.3479567841e-02f
	2.2096801549e-02f	1.9024293870e-02f	1.3828465715e-02f	7.0211756974e-03f
	2.2266115993e-03f	5.4535889067e-03f	1.8315145746e-02f	2.4675985798e-02f
	3.0890822411e-02f	3.3887583762e-02f	3.8129080087e-02f	4.0101163089e-02f
	2.8312398121e-02f	-3.0101649463e-03f	-1.1731998529e-03f	1.0505731218e-02f
	1.0645846277e-01f	6.7329242826e-02f	-1.6714345664e-02f	7.3007727042e-03f
	2.2047113627e-02f	1.8778838217e-02f	2.0771019161e-02f	1.7934981734e-02f
	1.9775275141e-02f	2.8688205406e-02f	3.7169270217e-02f	2.7402881533e-02f
	1.2317134999e-02f	2.3753659800e-02f	3.0001468956e-02f	3.8586121053e-02f
	3.9731174707e-02f	3.5777512938e-02f	2.1823013201e-02f	9.0754339471e-03f
4	1.1811972596e-02f	2.4254757911e-02f	3.6848545074e-02f	3.7516880780e-02f
	3.8546051830e-02f	4.2317461222e-02f	4.4197242707e-02f	4.5351982117e-02f
	4.6806633472e-02f	4.1953504086e-02f	2.9400527477e-02f	1.5827732161e-02f
	3.6057170946e-03f	7.8869936988e-03f	2.1630072966e-02f	2.5109922513e-02f
	3.3328097314e-02f	3.9614472538e-02f	4.9577750266e-02f	5.1395148039e-02f
	3.2411795110e-02f	4.0923268534e-03f	-4.9198260531e-03f	-6.5661720000e-03f
	1.6975034028e-02f	4.1352070868e-02f	6.4901754260e-02f	7.9033359885e-02f
	7.2041116655e-02f	3.6708444357e-02f	-1.0403243825e-02f	-3.2840065658e-02f
	-2.1601937711e-02f	3.3114261460e-03f	2.2614018992e-02f	2.5267221034e-02f
	1.6067504883e-02f	2.2008022293e-02f	2.0423235372e-02f	1.6338530928e-02f
	2.1665964276e-02f	3.0732197687e-02f	2.8864432126e-02f	1.7907647416e-02f
5	1.7633078620e-02f	2.2661305964e-02f	2.5312885642e-02f	2.1312126890e-02f
	1.5247132629e-02f	1.5420299023e-02f	1.8361732364e-02f	1.6553647816e-02f
	1.5191568062e-02f	1.6205305234e-02f	1.9334279001e-02f	1.6149401665e-02f
	1.2179457583e-02f	1.3218896464e-02f	1.3571953401e-02f	1.1827865615e-02f
	1.3693536632e-02f	1.1865335517e-02f	2.2464603186e-02f	3.1910575926e-02f
	2.2024041042e-02f	-5.6170350872e-03f	-9.4753624871e-03f	-1.1609563371e-03f
	1.0118631646e-02f	3.6039318889e-02f	7.6038688421e-02f	1.1855419725e-01f
	1.2439072132e-01f	8.1049792469e-02f	2.4128470570e-02f	-6.0141538270e-03f
	1.1604151689e-02f	4.4392492622e-02f	6.3117690384e-02f	5.3042437881e-02f
	3.3345483243e-02f	3.0971633270e-02f	2.8364934027e-02f	2.5159498677e-02f
	3.1332012266e-02f	4.0923949331e-02f	3.7316050380e-02f	2.3052547127e-02f
6	2.3418258876e-02f	3.1808927655e-02f	3.5763289779e-02f	2.6295870543e-02f
	1.9919270650e-02f	2.3661365733e-02f	2.9869837686e-02f	3.3885229379e-02f
	3.5865496844e-02f	3.7956643850e-02f	2.6621466503e-02f	1.1694314890e-02f
	-1.3196260261e-04f	1.6740674619e-03f	1.4299917035e-02f	2.3926476017e-02f
	3.1406264752e-02f	3.0109817162e-02f	3.8054041564e-02f	4.1092451662e-02f
	2.5878285989e-02f	-8.0648791045e-03f	-1.7479123548e-02f	-1.0938938707e-02f
	4.3306499720e-02f	8.6021497846e-02f	8.7245382369e-02f	6.1799056828e-02f
	3.3690650016e-02f	1.4516823925e-02f	3.2722528558e-03f	-9.9198147655e-04f
	1.2129316106e-02f	3.4726873040e-02f	5.7010982186e-02f	5.4310474545e-02f
	3.4216213971e-02f	2.8946742415e-02f	2.1106116474e-02f	1.5583798289e-02f
	2.2401792929e-02f	2.5670088828e-02f	2.0390128717e-02f	1.5215812251e-02f
7	2.0550219342e-02f	3.1793337315e-02f	3.4282941371e-02f	3.1439457089e-02f
	3.0268855393e-02f	3.3411063254e-02f	3.5523779690e-02f	3.3361945301e-02f
	2.8115876019e-02f	2.2591777146e-02f	1.5990726650e-02f	9.0388664976e-03f
	4.2094485834e-03f	1.0867130943e-02f	2.8406267986e-02f	3.9541911334e-02f
	4.4806934893e-02f	3.8689646870e-02f	4.0346574038e-02f	4.2935628444e-02f
	3.1997796148e-02f	3.0310042202e-03f	-5.0504799001e-03f	1.9110098947e-04f
	6.1082910746e-02f	1.1011497676e-01f	1.0341765732e-01f	5.9664249420e-02f
	4.2849544436e-02f	7.4377849698e-02f	1.1801311374e-01f	1.0613727570e-01f
	6.2655277550e-02f	3.7145569921e-02f	4.2601726949e-02f	5.6263964623e-02f
	6.9558031857e-02f	9.2821739614e-02f	8.8654749095e-02f	7.4373811483e-02f
	6.9542765617e-02f	6.8817518651e-02f	6.2202688307e-02f	5.3755577654e-02f
8	5.4700423032e-02f	6.2831953168e-02f	6.7061342299e-02f	6.3850417733e-02f
	6.3165843487e-02f	6.5290436149e-02f	7.2861030698e-02f	7.6138064265e-02f
	7.5420312583e-02f	6.4289174974e-02f	5.7370644063e-02f	4.5673359185e-02f

	3.9198007435e-02f	3.9754133672e-02f	5.5165186524e-02f	6.8919144571e-02f
	8.0096468329e-02f	8.5268713534e-02f	9.3134567142e-02f	9.6661575139e-02f
	8.1678152084e-02f	5.3439568728e-02f	4.9137350172e-02f	3.1697809696e-02f
	3.8420464844e-02f	7.7857807279e-02f	8.7948091328e-02f	7.5073793530e-02f
	7.0671856403e-02f	6.7001931369e-02f	6.0827746987e-02f	5.7909548283e-02f
	5.6665752083e-02f	5.6151337922e-02f	5.7596098632e-02f	5.0867322832e-02f
	4.9803815782e-02f	7.0196703076e-02f	7.8416034579e-02f	8.0167070031e-02f
	8.4792912006e-02f	7.6703198254e-02f	6.0424931347e-02f	4.8945486546e-02f
9	5.2403375506e-02f	6.6278316081e-02f	7.7262736857e-02f	8.7728902698e-02f
	1.0316040367e-01f	1.2786401808e-01f	1.3337166607e-01f	1.0862018913e-01f
	7.6222136617e-02f	5.5518288165e-02f	4.1951432824e-02f	3.0695341527e-02f
	2.3415615782e-02f	2.4426858872e-02f	3.4665416926e-02f	4.3070361018e-02f
	5.5287018418e-02f	7.5118437409e-02f	1.1355710030e-01f	1.6842924058e-01f
	1.8229193985e-01f	1.3081254065e-01f	7.0660121739e-02f	2.2412663326e-02f
	2.2335371003e-02f	4.8364449292e-02f	5.7229641825e-02f	6.0332540423e-02f
	6.4586319029e-02f	6.6814132035e-02f	5.8615058661e-02f	5.2371211350e-02f
	5.2903410047e-02f	5.5516991764e-02f	6.1046056449e-02f	5.2505835891e-02f
	3.7680771202e-02f	5.2805595100e-02f	6.2148202211e-02f	6.6147819161e-02f
	6.9502726197e-02f	7.2346821427e-02f	6.1051461846e-02f	4.8195082694e-02f
10	4.7433760017e-02f	5.2788671106e-02f	5.1999382675e-02f	5.0198711455e-02f
	5.2395604551e-02f	5.5790595710e-02f	5.9964872897e-02f	5.8963406831e-02f
	5.8805327863e-02f	5.4320957512e-02f	3.3696003258e-02f	2.2846685722e-02f
	4.2909730226e-02f	1.0647840053e-01f	1.7445056140e-01f	2.0022581518e-01f
	1.9361107051e-01f	1.5712690353e-01f	1.0923520476e-01f	6.1707310379e-02f
	2.8488021344e-02f	-6.4317951910e-03f	-1.0690449737e-02f	-5.8550904505e-03f
	1.7727492377e-02f	3.7350386381e-02f	5.1462803036e-02f	7.1507707238e-02f
	1.0140767694e-01f	1.2004347891e-01f	1.1460112035e-01f	8.7861843407e-02f
	6.2203314155e-02f	4.6546161175e-02f	4.1290409863e-02f	3.2635774463e-02f
	2.8707578778e-02f	5.1554739475e-02f	7.1465060115e-02f	8.5711009800e-02f
	9.7041286528e-02f	1.0464775562e-01f	9.3846425414e-02f	8.1467039883e-02f
11	8.0551490188e-02f	8.7732598186e-02f	8.6113564670e-02f	7.6208010316e-02f
	6.2575966120e-02f	5.6754224002e-02f	5.8409158140e-02f	6.2651827931e-02f
	6.4884789288e-02f	6.1761405319e-02f	5.9512097389e-02f	4.8387810588e-02f
	4.1640341282e-02f	4.4278290123e-02f	5.4952085018e-02f	6.3185855746e-02f
	6.7893177271e-02f	6.9451905787e-02f	8.0154784024e-02f	8.8171586394e-02f
	8.3153806627e-02f	6.1147753149e-02f	5.3778670728e-02f	4.3326616287e-02f
	4.0364358574e-02f	6.9392733276e-02f	4.7434836626e-02f	2.2801719606e-02f
	2.5718430057e-02f	5.0013735890e-02f	7.2132200003e-02f	8.2452550530e-02f
	7.6382443309e-02f	6.7744299769e-02f	6.7147806287e-02f	6.0207460076e-02f
	4.7333352268e-02f	6.2260188162e-02f	7.5620360672e-02f	8.1184223294e-02f
	7.5513362885e-02f	7.5355529785e-02f	6.8073995411e-02f	6.1538677663e-02f
12	5.7830538601e-02f	5.6865684688e-02f	5.8483913541e-02f	5.7368561625e-02f
	5.5988781154e-02f	6.1915632337e-02f	6.5816864371e-02f	6.8111591041e-02f
	6.6529579461e-02f	5.3584717214e-02f	4.3755196035e-02f	3.2064981759e-02f
	2.5044754148e-02f	2.6641104370e-02f	3.7416994572e-02f	4.1116293520e-02f
	3.7952277809e-02f	3.5305988044e-02f	4.4744879007e-02f	5.4820153862e-02f
	7.2098992765e-02f	1.1404184997e-01f	1.8387553096e-01f	1.8528544903e-01f
	5.7107888162e-02f	5.4888118058e-02f	5.7970746420e-03f	4.7652497888e-02f
	9.3716062605e-02f	3.6349788308e-02f	-4.3968958780e-03f	2.0093087107e-02f
	3.8630887866e-02f	3.5486537963e-02f	2.6472328231e-02f	1.9216997549e-02f
	1.8381707370e-02f	3.1321909279e-02f	3.0055094510e-02f	2.7434976771e-02f
	3.6957971752e-02f	4.1335891932e-02f	3.1953182071e-02f	2.5468677282e-02f
13	2.7038693428e-02f	3.7809468806e-02f	4.1285607964e-02f	4.2780894786e-02f
	3.9417959750e-02f	3.8220454007e-02f	4.2868603021e-02f	4.3441738933e-02f
	3.7647411227e-02f	3.4852787852e-02f	3.0444566160e-02f	1.9364051521e-02f
	5.6729717180e-03f	3.1449880917e-03f	1.7938503996e-02f	2.6688387617e-02f
	3.9262577891e-02f	5.0142459571e-02f	6.3333943486e-02f	6.3696458936e-02f
	5.1719944924e-02f	1.6124179587e-02f	5.0704530440e-03f	7.9158721492e-03f
	2.9456902295e-02f	5.9918854386e-02f	8.0512829125e-02f	9.2115439475e-02f
	1.0516671091e-01f	1.0463880002e-01f	8.6369782686e-02f	5.9955984354e-02f
	4.9864243716e-02f	6.3410110772e-02f	9.5254153013e-02f	1.2300701439e-01f
	1.3081574440e-01f	1.2616425753e-01f	9.5623619854e-02f	7.1546003222e-02f
	6.6854029894e-02f	6.9707997143e-02f	6.4478017390e-02f	6.0643393546e-02f
14	6.4357183874e-02f	7.0302851498e-02f	6.9613456726e-02f	6.3198678195e-02f
	5.5283885449e-02f	5.2881922573e-02f	5.7896248996e-02f	6.7727655172e-02f
	7.9541660845e-02f	7.9558596015e-02f	7.6944433153e-02f	6.5805949271e-02f
	5.5649552494e-02f	5.5435992777e-02f	6.6283158958e-02f	7.5325563550e-02f
	8.0668196082e-02f	8.3046242595e-02f	9.2973299325e-02f	9.9515043199e-02f
	8.5281021893e-02f	4.6252630651e-02f	3.3056970686e-02f	2.7986641973e-02f
	4.0577519685e-02f	7.9788267612e-02f	8.7299197912e-02f	6.9627106190e-02f
	5.0393719226e-02f	3.7108283490e-02f	3.4224815667e-02f	4.4347092509e-02f
	6.3847832382e-02f	7.5548335910e-02f	7.2322390974e-02f	5.5542118847e-02f
	4.2693156749e-02f	5.9285275638e-02f	7.2072334588e-02f	7.8878879547e-02f
	9.2471212149e-02f	1.0410591960e-01f	1.0047107935e-01f	9.0327091515e-02f

15	8.5730113089e-02f	7.9587087035e-02f	6.3908971846e-02f	5.1930986345e-02f
	4.4641312212e-02f	4.5633178204e-02f	5.3232021630e-02f	6.1188496649e-02f
	7.7726230025e-02f	8.3959192038e-02f	7.4758298695e-02f	5.4894741625e-02f
	3.6559864879e-02f	3.2735075802e-02f	4.3763387948e-02f	5.1399320364e-02f
	5.1693774760e-02f	5.0045587122e-02f	5.9883374721e-02f	7.3056742549e-02f
	6.7932091653e-02f	4.1426528245e-02f	3.2348286361e-02f	2.1336564794e-02f

/* SE_Shape2 */
/* dim=44 * 16 codevectors */

index	codeword			
0	3.8706362247e-02f	7.2990626097e-02f	7.5231507421e-02f	5.9537094086e-02f
	6.9145523012e-02f	7.2555713356e-02f	5.2657146007e-02f	3.3242393285e-02f
	2.7920771390e-02f	3.1365152448e-02f	3.5927828401e-02f	4.7156255692e-02f
	5.4359123111e-02f	4.6460285783e-02f	6.1409343034e-02f	7.3193587363e-02f
	5.7434201241e-02f	4.4028025120e-02f	5.1478073001e-02f	6.0767289251e-02f
	5.9352658689e-02f	4.9155827612e-02f	3.8960408419e-02f	4.3170157820e-02f
	4.7791708261e-02f	5.1973942667e-02f	5.4617501795e-02f	5.9349846095e-02f
	6.9081991911e-02f	6.6679917276e-02f	5.9687253088e-02f	6.3100852072e-02f
	6.9456934929e-02f	6.9135591388e-02f	5.8942925185e-02f	4.8410013318e-02f
	4.1769035161e-02f	3.9985764772e-02f	3.1373497099e-02f	3.1174277887e-02f
	4.9212012440e-02f	7.8576855361e-02f	7.9990953207e-02f	5.7072967291e-02f
1	5.5484507233e-02f	1.0093380511e-01f	8.3452947438e-02f	5.1292255521e-02f
	8.5428223014e-02f	1.3923546672e-01f	1.1544527113e-01f	7.3407284915e-02f
	7.0194594562e-02f	7.7750638127e-02f	8.2068979740e-02f	7.9910092056e-02f
	7.8697979450e-02f	6.5324291587e-02f	6.3150055707e-02f	7.0677608252e-02f
	6.8730130792e-02f	6.4008466899e-02f	6.8564571440e-02f	7.4436940253e-02f
	7.3772847652e-02f	6.9107316434e-02f	7.1665905416e-02f	7.9596959054e-02f
	8.6811721325e-02f	8.2727611065e-02f	7.4204713106e-02f	7.3417976499e-02f
	7.1524374187e-02f	7.3802322149e-02f	7.3060497642e-02f	7.9214364290e-02f
	8.9975997806e-02f	9.8077870905e-02f	1.0727620870e-01f	1.0809461772e-01f
	1.0416465998e-01f	1.1431089789e-01f	1.1454233527e-01f	1.0222646594e-01f
	9.3370839953e-02f	1.0473866016e-01f	8.6701139808e-02f	5.1155142486e-02f
2	2.2572457790e-02f	5.6805346161e-02f	1.0488257557e-01f	1.4198201895e-01f
	1.4484642446e-01f	1.3089640439e-01f	1.1805525422e-01f	9.9233835936e-02f
	7.1837514639e-02f	4.8670921475e-02f	3.5636518151e-02f	4.4414546341e-02f
	7.0559695363e-02f	7.9148665071e-02f	8.4304831922e-02f	8.2300446928e-02f
	6.9804303348e-02f	6.5054200590e-02f	6.9092057645e-02f	7.4492998421e-02f
	6.8367697299e-02f	6.0354534537e-02f	6.2587723136e-02f	6.9613106549e-02f
	6.8079143763e-02f	6.2400545925e-02f	5.8752343059e-02f	5.9182912111e-02f
	6.1796065420e-02f	6.8422377110e-02f	8.0459110439e-02f	9.3372814357e-02f
	1.0539379716e-01f	1.0390836746e-01f	9.0056695044e-02f	7.6026119292e-02f
	6.4753800631e-02f	5.8991476893e-02f	4.9729228020e-02f	4.5603331178e-02f
	6.0888845474e-02f	8.8226355612e-02f	8.0370634794e-02f	5.7824049145e-02f
3	6.7203538492e-03f	2.9254691675e-02f	7.2735577822e-02f	1.1968076229e-01f
	1.4931136370e-01f	1.6118887067e-01f	1.6263423860e-01f	1.5781964362e-01f
	1.4002835751e-01f	1.2222293764e-01f	1.0758341104e-01f	1.0917364806e-01f
	1.1760831624e-01f	1.0191807151e-01f	9.6628487110e-02f	9.3745961785e-02f
	9.0448878706e-02f	8.4572479129e-02f	9.0673096478e-02f	9.7116120160e-02f
	9.1625668108e-02f	8.4917679429e-02f	8.0454923213e-02f	8.4654025733e-02f
	9.3122549355e-02f	9.1688968241e-02f	8.7778761983e-02f	8.4576502442e-02f
	8.2850515842e-02f	8.7590374053e-02f	9.5837078989e-02f	1.0765032470e-01f
	1.2293945998e-01f	1.2516519427e-01f	1.1241103709e-01f	1.0059371591e-01f
	9.1731742024e-02f	8.8746219873e-02f	8.3748273551e-02f	8.6221352220e-02f
	1.0733599216e-01f	1.3765008748e-01f	1.2150909752e-01f	8.7062753737e-02f
4	5.6150142103e-02f	1.1155796051e-01f	1.1954480410e-01f	7.1470871568e-02f
	2.1016081795e-02f	1.5891348943e-02f	3.9042443037e-02f	5.8757707477e-02f
	5.4319556803e-02f	3.6753036082e-02f	2.3346025497e-02f	2.7783064172e-02f
	3.4745741636e-02f	2.2894755006e-02f	2.1850267425e-02f	2.8336251155e-02f
	3.1278837472e-02f	3.0013794079e-02f	3.1160833314e-02f	3.5724785179e-02f
	3.3537276089e-02f	2.6618346572e-02f	2.2854270414e-02f	2.9976347461e-02f
	3.8834948093e-02f	3.8354214281e-02f	3.5426922143e-02f	3.1536396593e-02f
	2.8617581353e-02f	2.8230568394e-02f	3.2673768699e-02f	4.5456364751e-02f
	5.5265519768e-02f	5.3952883929e-02f	4.4147871435e-02f	3.6736268550e-02f
	3.3460836858e-02f	2.7243627235e-02f	1.4796694741e-02f	8.3728376776e-03f
	2.2343823686e-02f	5.1126047969e-02f	5.6937962770e-02f	5.5966418236e-02f
5	3.9643403143e-02f	8.1140279770e-02f	9.2448562384e-02f	7.8608423471e-02f
	5.5948685855e-02f	4.1359551251e-02f	4.0713753551e-02f	5.0717510283e-02f
	6.9980926812e-02f	9.7598358989e-02f	1.2106382102e-01f	1.2034637481e-01f
	9.0608052909e-02f	4.5446041971e-02f	3.4212715924e-02f	4.2682409286e-02f
	4.6938654035e-02f	4.5344885439e-02f	5.3507074714e-02f	6.2605492771e-02f
	6.0620311648e-02f	5.2731964737e-02f	5.2676618099e-02f	6.1694148928e-02f
	7.2479620576e-02f	7.3951460421e-02f	7.3033429682e-02f	7.2181493044e-02f

	6.7661106586e-02f	7.1738228202e-02f	7.2957105935e-02f	7.6371036470e-02f
	8.3036832511e-02f	8.2807041705e-02f	7.0938996971e-02f	5.8039572090e-02f
	5.4765589535e-02f	5.5595654994e-02f	5.0787240267e-02f	3.8547512144e-02f
	4.8532806337e-02f	7.6936498284e-02f	6.8060472608e-02f	4.1970644146e-02f
6	2.9097380117e-02f	6.0783542693e-02f	9.3657515943e-02f	1.1780882627e-01f
	9.9183216691e-02f	6.4551152289e-02f	3.8557775319e-02f	2.5910379365e-02f
	2.3990513757e-02f	2.9282517731e-02f	4.1784778237e-02f	5.5893603712e-02f
	6.5700493753e-02f	4.8495396972e-02f	4.0120333433e-02f	3.7728376687e-02f
	3.3264040947e-02f	3.1073169783e-02f	3.9877511561e-02f	5.0224550068e-02f
	4.8018299043e-02f	3.9955433458e-02f	3.9441648871e-02f	4.4570297003e-02f
	4.5898962766e-02f	4.2628306895e-02f	4.4162441045e-02f	4.3730895966e-02f
	3.8452282548e-02f	3.6206915975e-02f	4.0576506406e-02f	4.8898220062e-02f
	6.0252428055e-02f	6.0313124210e-02f	4.8443078995e-02f	3.9951995015e-02f
	3.4930154681e-02f	3.5448398441e-02f	3.1672414392e-02f	3.4144412726e-02f
	6.5202005208e-02f	1.2354435027e-01f	1.3055154681e-01f	9.0947292745e-02f
7	2.7572141960e-02f	6.4806818962e-02f	1.0038803518e-01f	1.1764846742e-01f
	1.0950426012e-01f	8.8794387877e-02f	7.6011784375e-02f	8.6720772088e-02f
	1.0681661218e-01f	1.1980085820e-01f	1.0690146685e-01f	8.9794509113e-02f
	7.8151136637e-02f	5.4277040064e-02f	4.9921169877e-02f	5.5518355221e-02f
	5.8136064559e-02f	5.9430684894e-02f	7.3612406850e-02f	8.5090495646e-02f
	8.3840191364e-02f	7.1488454938e-02f	6.3060939312e-02f	5.8577693999e-02f
	6.1905216426e-02f	6.6401906312e-02f	7.1843564510e-02f	8.0152198672e-02f
	7.9502537847e-02f	7.6400928199e-02f	7.0105545223e-02f	7.7218644321e-02f
	9.0097382665e-02f	9.6982680261e-02f	9.7282163799e-02f	9.4035781920e-02f
	8.2511022687e-02f	7.2979368269e-02f	6.2213540077e-02f	6.2729232013e-02f
	7.5849249959e-02f	9.6873670816e-02f	8.0736257136e-02f	5.2927505225e-02f
8	7.3545061052e-02f	1.2888546288e-01f	1.0550788790e-01f	5.3969461471e-02f
	4.1674967855e-02f	7.3834180832e-02f	1.1738254875e-01f	1.1707971990e-01f
	7.7369011939e-02f	5.9448834509e-02f	7.1564652026e-02f	9.7164817154e-02f
	1.0626997799e-01f	7.7785313129e-02f	6.1773065478e-02f	6.8501845002e-02f
	7.6232999563e-02f	7.3954001069e-02f	7.8446097672e-02f	8.5335403681e-02f
	8.4762319922e-02f	7.3200292885e-02f	6.6689379513e-02f	6.4248047769e-02f
	6.5616674721e-02f	6.8610660732e-02f	7.4283622205e-02f	7.5482107699e-02f
	7.6617516577e-02f	8.1293962896e-02f	8.7007276714e-02f	9.7466453910e-02f
	1.1315532774e-01f	1.1647962034e-01f	1.0017292947e-01f	8.6254857481e-02f
	7.8061200678e-02f	7.1598038077e-02f	6.1614274979e-02f	5.4236598313e-02f
	6.7423835397e-02f	9.7347311676e-02f	8.8979527354e-02f	5.2700813860e-02f
9	3.3805582672e-02f	6.9729052484e-02f	9.1367661953e-02f	9.0377435088e-02f
	8.2413487136e-02f	8.1832021475e-02f	8.9285664260e-02f	9.6965841949e-02f
	9.0433552861e-02f	7.7089324594e-02f	6.5413534641e-02f	7.3764905334e-02f
	8.9196287096e-02f	8.0944366753e-02f	7.8773580492e-02f	8.0819264054e-02f
	7.4580185115e-02f	7.3159672320e-02f	8.6987555027e-02f	1.0832354426e-01f
	1.1938677728e-01f	1.1934114993e-01f	1.1672365665e-01f	1.0829072446e-01f
	8.9212000370e-02f	6.6442221403e-02f	5.1822472364e-02f	5.2195400000e-02f
	5.7408746332e-02f	7.2611682117e-02f	9.2485398054e-02f	1.2534233928e-01f
	1.5412391722e-01f	1.6207909584e-01f	1.3924449682e-01f	1.0470146686e-01f
	6.6691070795e-02f	4.7201212496e-02f	3.9681032300e-02f	4.3567605317e-02f
	6.5298214555e-02f	9.4449795783e-02f	8.2501679659e-02f	5.3902659565e-02f
10	4.8961296678e-02f	1.0521986336e-01f	1.3708667457e-01f	1.3649751246e-01f
	1.0006598383e-01f	7.4411183596e-02f	7.1618899703e-02f	8.0776132643e-02f
	7.7145263553e-02f	7.2762742639e-02f	6.3263364136e-02f	7.3825359344e-02f
	8.5945509374e-02f	8.0621585250e-02f	7.9135924578e-02f	8.0727092922e-02f
	7.6561644673e-02f	7.2861753404e-02f	8.1293068826e-02f	9.1111838818e-02f
	8.5633359849e-02f	7.7586971223e-02f	7.5633987784e-02f	8.1931807101e-02f
	9.5486573875e-02f	1.0435543954e-01f	1.0501424223e-01f	9.9639542401e-02f
	8.7607070804e-02f	8.2942113280e-02f	8.5371844471e-02f	7.9755656421e-02f
	8.3965733647e-02f	9.1229692101e-02f	1.0330295563e-01f	1.3499264419e-01f
	1.7367357016e-01f	1.9138227403e-01f	1.8229748309e-01f	1.3575275242e-01f
	9.1723747551e-02f	8.3105213940e-02f	6.8250790238e-02f	4.4135343283e-02f
11	5.0678752363e-02f	1.0295594484e-01f	1.2747347355e-01f	1.1097738147e-01f
	7.5022920966e-02f	6.4382776618e-02f	9.5980927348e-02f	1.4541450143e-01f
	1.6140109301e-01f	1.3157613575e-01f	8.5914716125e-02f	6.8473413587e-02f
	8.0014474690e-02f	8.2100607455e-02f	9.0031147003e-02f	9.6752375364e-02f
	9.6454642713e-02f	8.9624367654e-02f	9.1973006725e-02f	9.5993250608e-02f
	9.4336368144e-02f	8.5323274136e-02f	8.2597263157e-02f	8.5424177349e-02f
	8.3329349756e-02f	7.8489668667e-02f	7.5004741549e-02f	7.7952772379e-02f
	8.0848120153e-02f	8.4944054484e-02f	8.8886335492e-02f	1.0391312093e-01f
	1.1861022562e-01f	1.2023038417e-01f	1.1000689864e-01f	9.9737919867e-02f
	9.0197488666e-02f	8.4633164108e-02f	7.7712871134e-02f	7.6428145170e-02f
	9.1144204140e-02f	1.1691223830e-01f	9.9759243429e-02f	6.1518128961e-02f
	7.1107685566e-02f	1.0198251903e-01f	5.5477257818e-02f	2.6764094830e-02f
	3.6523252726e-02f	5.0641078502e-02f	4.8425164074e-02f	3.2875560224e-02f
	2.7992824093e-02f	3.2850336283e-02f	3.9351645857e-02f	4.3503489345e-02f
	5.0081767142e-02f	3.6236539483e-02f	3.1667202711e-02f	3.5004470497e-02f

	3.6414261907e-02f	3.7266802043e-02f	4.7006540000e-02f	5.2110664546e-02f
12	4.6211581677e-02f	3.6575816572e-02f	3.4310445189e-02f	3.9176344872e-02f
	4.3630409986e-02f	4.4992618263e-02f	4.0209054947e-02f	3.5170663148e-02f
	3.7787709385e-02f	4.7971140593e-02f	5.2886690944e-02f	6.0913782567e-02f
	6.4439773560e-02f	5.9719312936e-02f	4.6340830624e-02f	4.0867593139e-02f
	3.7135388702e-02f	3.2559096813e-02f	2.0773991942e-02f	2.5727422908e-02f
	5.4446887225e-02f	9.6700564027e-02f	8.6165823042e-02f	3.8739793003e-02f
	7.0733830333e-02f	9.5370665193e-02f	4.1769202799e-02f	3.8947705179e-02f
	1.0411918908e-01f	1.0418128222e-01f	5.5299911648e-02f	4.2688995600e-02f
	3.9719246328e-02f	4.3510835618e-02f	6.5251894295e-02f	8.8830925524e-02f
	8.1258825958e-02f	5.5699277669e-02f	5.0580784678e-02f	6.2724605203e-02f
	6.0044020414e-02f	5.5649843067e-02f	6.6323854029e-02f	7.5363725424e-02f
13	7.4311643839e-02f	6.2426831573e-02f	6.2092084438e-02f	6.0429688543e-02f
	6.7231416702e-02f	7.3580779135e-02f	7.7451758087e-02f	7.8391827643e-02f
	8.0791518092e-02f	8.0324009061e-02f	7.5925827026e-02f	6.9355510175e-02f
	7.5381651521e-02f	7.9297393560e-02f	6.5655201674e-02f	5.7774394751e-02f
	5.0197795033e-02f	4.2525831610e-02f	2.8420466930e-02f	1.9136168063e-02f
	4.1596818715e-02f	1.0648939013e-01f	1.6417750716e-01f	1.7307174206e-01f
	5.6198593229e-02f	1.1380554736e-01f	1.3920806348e-01f	1.1666043848e-01f
	7.1563348174e-02f	5.1009245217e-02f	5.8051686734e-02f	6.8593293428e-02f
	6.4346000552e-02f	4.8834037036e-02f	3.7655897439e-02f	5.3460288793e-02f
	8.9279972017e-02f	1.0571532696e-01f	1.0469762981e-01f	9.4223447144e-02f
	7.8351385891e-02f	6.6081300378e-02f	6.9504633546e-02f	8.0036111176e-02f
14	7.7518336475e-02f	6.6689766943e-02f	6.4562387764e-02f	6.6546119750e-02f
	6.4414262772e-02f	6.3479207456e-02f	6.4499184489e-02f	6.8517610431e-02f
	6.9911785424e-02f	7.7615454793e-02f	8.3714760840e-02f	8.9376308024e-02f
	9.7193241119e-02f	9.5563635230e-02f	8.3775185049e-02f	7.0126205683e-02f
	5.7045780122e-02f	5.0486944616e-02f	4.2355980724e-02f	4.0009293705e-02f
	6.0175243765e-02f	1.1014249921e-01f	1.1538860202e-01f	8.3244211972e-02f
	2.7824740857e-02f	6.3298113644e-02f	9.9403604865e-02f	1.1743813008e-01f
	1.0541018099e-01f	7.9558864236e-02f	5.7742845267e-02f	4.8049386591e-02f
	4.1546121240e-02f	4.2710237205e-02f	5.3325593472e-02f	8.3362765610e-02f
	1.0777246207e-01f	1.0401544720e-01f	1.0091172904e-01f	1.0539283603e-01f
	1.0535249859e-01f	1.0320958495e-01f	9.8641633987e-02f	8.9673303068e-02f
15	7.5271598995e-02f	6.2084432691e-02f	6.3357584178e-02f	7.3057331145e-02f
	8.5113592446e-02f	9.1620981693e-02f	8.7552756071e-02f	8.2570925355e-02f
	7.9608388245e-02f	7.9005375504e-02f	8.7157271802e-02f	1.0537154227e-01f
	1.2238393724e-01f	1.2732574344e-01f	1.1667739600e-01f	9.3778163195e-02f
	6.5241351724e-02f	4.5343209058e-02f	3.1513817608e-02f	3.1813796610e-02f
	5.6809503585e-02f	9.9267236888e-02f	9.2696867883e-02f	6.2752395868e-02f

2. CbAm4k

VQ codebook for harmonic spectral vector quantization for 4kbps.

/* SE_Shape3 */
/* dim=2x128 codewords */

index	codeword	
0	-9.330278e+00f	1.061832e+00f
1	-1.492120e+01f	-2.622407e+00f
2	-1.900595e+00f	-7.610120e+00f
3	-3.441202e+00f	-3.577140e+00f
4	2.710095e+00f	-1.017617e+01f
5	2.931478e+00f	-6.087423e+00f
6	-4.666350e+00f	-1.694503e-01f
7	-5.307190e-01f	-1.796010e-01f
8	4.581445e-01f	5.514712e-01f
9	7.915068e-01f	-3.515388e+00f
10	9.874204e+00f	-1.054440e+01f
11	7.185859e+00f	-4.573974e+00f
12	1.883653e-01f	-1.623154e+01f
13	-1.950054e+01f	-2.563769e+01f
14	-2.781315e+01f	-1.208438e+01f
15	-3.783871e+01f	-3.179733e+01f
16	-1.357486e+01f	-1.628080e+01f
17	-2.513922e+01f	3.266418e+00f
18	-1.631194e+01f	-7.922872e+00f
19	-1.653433e+01f	3.635027e+00f

20	-6.067080e+00f	-1.184124e+01f
21	-6.535157e+00f	-2.653723e+01f
22	1.929753e+01f	-3.680296e+01f
23	7.749988e+00f	-3.761107e+01f
24	9.959105e+00f	-1.678411e+01f
25	-4.877574e+00f	5.101901e+00f
26	-6.106386e+01f	7.439591e+01f
27	-3.656976e+01f	2.985260e+01f
28	-5.946972e+01f	1.370801e+02f
29	-2.180651e+01f	1.777025e+02f
30	8.370714e+00f	7.181134e+01f
31	-4.053744e+01f	1.101968e+02f
32	1.282547e+01f	4.682676e+01f
33	1.148900e+01f	2.345230e+01f
34	-1.550038e+01f	2.164598e+01f
35	4.731580e+00f	2.090409e+01f
36	-2.702352e+01f	1.993958e+01f
37	-5.698683e+01f	3.089530e+01f
38	-1.119594e+02f	5.631562e+01f
39	-8.300320e+01f	3.276011e+01f
40	-9.705156e+01f	8.933459e+01f
41	-7.938153e+01f	2.220908e+02f
42	-2.366370e+01f	4.197907e+02f
43	-6.310874e+01f	4.001344e+02f
44	-6.204693e+01f	2.778516e+02f
45	-3.907400e+01f	2.183506e+02f
46	8.714683e+01f	1.656750e+02f
47	2.370267e+01f	1.425619e+02f
48	-3.834114e+00f	1.251659e+02f
49	-1.895855e+01f	9.247249e+01f
50	-3.303055e+00f	4.702967e+01f
51	-1.113869e+01f	6.266771e+01f
52	2.144364e+00f	3.224986e+01f
53	-6.454165e+00f	2.528232e+01f
54	4.275454e+00f	1.313530e+01f
55	-8.864230e+00f	1.484505e+01f
56	-1.040869e+01f	4.023464e+01f
57	-2.168141e+01f	5.589467e+01f
58	-2.392366e+01f	3.691694e+01f
59	-4.436973e+01f	5.498315e+01f
60	-1.449710e+01f	3.019523e+01f
61	-1.010045e+01f	7.361667e+00f
62	-6.747651e+00f	-4.614012e+01f
63	-8.345326e+00f	-5.205293e+00f
64	-3.626784e+01f	-6.445258e+01f
65	-7.917755e+01f	-6.871539e+01f
66	-1.693471e+02f	-1.707075e+01f
67	-1.773848e+02f	-1.270122e+02f
68	-9.773572e+01f	-1.461007e+01f
69	-7.357993e+01f	3.526001e+00f
70	-4.965025e+01f	1.262963e+01f
71	-5.648425e+01f	-1.376295e+01f
72	-3.824075e+01f	-3.148426e+00f
73	-3.403748e+01f	9.539351e+00f
74	-4.027715e+00f	1.035430e+01f
75	-1.708537e+01f	1.185532e+01f
76	-2.007979e+00f	1.718235e+01f
77	1.935987e-01f	8.960434e+00f
78	8.661563e+00f	2.804276e+00f
79	-4.136063e-01f	4.116927e+00f
80	2.219103e+01f	9.406723e+00f
81	2.907092e+01f	4.419312e+00f
82	5.144141e+01f	-2.646189e+01f
83	4.198582e+01f	-1.215980e+01f
84	6.010605e+01f	-1.494482e+01f
85	7.151018e+01f	-2.900623e+01f
86	9.791963e+01f	1.244136e+01f
87	9.330734e+01f	-1.662056e+01f
88	6.341529e+01f	2.788360e+00f
89	4.337169e+01f	1.653579e+00f

90	1.300677e+01f	-4.108841e+00f
91	2.147639e+01f	-1.962553e+00f
92	1.816353e+01f	-8.455014e+00f
93	1.942810e+01f	-1.930163e+01f
94	-1.476922e+00f	-6.915480e+01f
95	4.642614e+00f	-2.533368e+01f
96	2.267755e+01f	-1.021017e+02f
97	-5.465425e+01f	-2.118657e+02f
98	4.114766e+01f	-5.645194e+01f
99	7.544187e+01f	-1.406515e+02f
100	2.998307e+01f	-2.806185e+01f
101	2.797319e+01f	-1.376450e+01f
102	5.208605e+01f	1.533336e+01f
103	3.514661e+01f	1.650218e+01f
104	7.573270e+01f	3.889431e+01f
105	5.697143e+01f	4.354917e+01f
106	2.351386e+01f	2.414765e+01f
107	3.836670e+01f	3.775769e+01f
108	9.905634e+00f	7.630831e+00f
109	4.381948e+00f	8.813215e-01f
110	4.059433e+00f	5.612900e+00f
111	1.909887e+00f	2.500592e+00f
112	1.439434e+01f	1.336010e+01f
113	2.194865e+01f	3.560208e+01f
114	7.621645e+01f	8.225101e+01f
115	3.429607e+01f	6.667796e+01f
116	1.397948e+02f	9.610065e+01f
117	1.557423e+02f	-3.970766e+00f
118	1.191250e+02f	-4.364631e+01f
119	1.901227e+02f	-5.155558e+01f
120	7.658812e+01f	-5.967710e+01f
121	3.363451e+01f	-5.372894e+00f
122	9.363360e+00f	-5.223809e-01f
123	1.601680e+01f	3.522246e+00f
124	4.217764e+00f	-2.024498e+00f
125	1.655762e+00f	-6.364616e-01f
126	-2.120250e+00f	1.710167e+00f
127	-8.462719e-01f	-1.689207e+00f

/* SE_Shape4 */
/* dim=4x1024 codewords */

index	codeword			
0	-1.364384e+02f	-5.245039e+01f	-5.029283e+01f	-1.846552e+01f
1	-1.445289e+01f	-3.624968e+01f	-1.981857e+01f	-1.460481e+01f
2	-1.299906e+01f	-1.671952e+01f	-2.561983e+01f	1.041228e+00f
3	-2.478106e+00f	-6.855657e-01f	-2.661671e+00f	1.236647e+00f
4	-1.596030e+01f	-2.411508e+01f	4.306259e+00f	8.641845e+00f
5	2.379335e+00f	2.301139e+01f	3.618788e+01f	6.056512e+01f
6	-9.528793e+00f	-3.647632e-02f	-3.337528e+01f	2.952796e+00f
7	-1.512855e+01f	-6.847051e+00f	1.189933e+01f	-1.562475e+01f
8	-3.023451e+01f	-4.650233e+00f	-7.240145e-01f	1.400420e+01f
9	4.839326e+01f	1.415856e+01f	-4.687170e+01f	6.165537e+01f
10	1.119807e+01f	1.252406e+01f	-4.392911e+00f	1.185895e+00f
11	3.159623e+00f	8.051858e+00f	-1.219139e+01f	8.082780e+00f
12	7.653646e+01f	-3.183190e+01f	6.952997e+01f	5.520950e+01f
13	5.641226e+01f	1.061417e+01f	8.461655e+01f	2.602136e+01f
14	4.420422e+00f	-4.395473e+00f	9.149106e+00f	3.813422e+01f
15	1.106854e+01f	5.233653e+00f	1.511094e+01f	4.045953e+00f
16	-2.415439e+02f	-7.630371e+00f	-3.983972e+01f	9.655096e+01f
17	-8.070139e+01f	-4.810519e-01f	6.841217e-01f	2.531219e+01f
18	-1.147429e+02f	2.331063e+01f	-2.730755e+01f	1.383179e+00f
19	-1.252475e+01f	7.038949e+00f	-2.132626e+01f	-8.217557e+00f
20	-8.331569e+01f	-6.820316e-01f	-3.673656e+01f	2.942174e+00f
21	-8.723468e+00f	7.811459e+00f	1.077449e+01f	1.319319e+01f
22	-5.799712e+01f	1.743526e+01f	-6.709693e+01f	1.409284e+01f
23	-9.693032e+01f	7.158996e+00f	2.251706e+01f	-3.195248e+01f
24	-7.693613e+01f	-6.474833e+00f	-2.913046e+01f	2.165680e+01f

25	7.577949e+00f	-2.688006e+00f	-1.251871e+01f	-6.103900e-01f
26	-1.663193e+01f	4.009673e+00f	1.638231e+00f	-1.556418e+00f
27	2.813078e+00f	6.526221e+00f	1.383008e+00f	1.543341e+01f
28	4.058404e+00f	-1.344226e+01f	1.266024e+01f	1.826480e+01f
29	6.848901e+00f	-2.997821e+00f	1.220647e+01f	4.058192e+00f
30	-9.275431e+00f	1.124766e+00f	-5.466308e+00f	3.648460e+00f
31	-1.704063e+01f	5.319377e+00f	1.104139e+01f	3.435214e+00f
32	-5.258579e+01f	4.318124e+01f	-2.113438e+01f	-2.529253e+01f
33	-1.516301e+00f	6.995528e-01f	1.016399e+00f	-7.206601e+00f
34	2.677489e-01f	1.384137e+01f	-1.539023e+02f	2.286378e+01f
35	3.558169e-02f	1.111842e+01f	-3.277153e+01f	4.942796e+00f
36	-6.726787e+00f	-4.899942e+00f	-3.928260e+00f	-2.684797e+01f
37	-2.259289e+01f	-9.837343e+00f	2.329112e+01f	-5.143919e+01f
38	1.041503e+00f	3.104128e+01f	-2.995096e+01f	3.311337e+00f
39	8.188571e-01f	-1.874124e+00f	2.969519e+01f	8.541806e+00f
40	-8.432916e+00f	3.625626e+00f	-2.273713e+00f	7.080631e+00f
41	-1.647968e+00f	-1.689554e+01f	9.840540e+00f	-8.863089e+00f
42	5.104390e+01f	3.928497e+01f	-1.093151e+01f	2.765937e+00f
43	7.543349e+00f	6.280970e+00f	-3.152267e+00f	5.181215e+00f
44	2.210004e+01f	1.919289e+01f	1.593593e+01f	6.876768e+01f
45	2.528874e+01f	1.624230e+01f	6.702363e+01f	4.260561e+01f
46	-1.191514e+01f	6.263915e+00f	-1.528244e+01f	-5.174061e+01f
47	3.631498e+00f	1.120965e+01f	1.202130e+01f	1.424202e+01f
48	-1.810129e+02f	1.026435e+02f	-3.063002e+01f	2.053101e+00f
49	-5.536498e+01f	2.238407e+01f	-1.888661e+01f	-3.248638e+01f
50	-3.955809e+01f	2.794468e+01f	-1.888197e+01f	-3.696971e+00f
51	-2.484796e+01f	5.076927e+01f	1.887568e+01f	-1.483922e+01f
52	-3.814856e+01f	3.748082e+01f	-2.964822e+01f	-9.238855e+00f
53	-8.180192e-01f	1.602494e+01f	1.914709e+01f	-6.949176e+00f
54	1.101730e+01f	1.091229e+01f	2.343170e+01f	-1.471074e+01f
55	-4.407196e+00f	5.090756e+01f	1.706512e+02f	-2.900731e+01f
56	-5.394787e+01f	2.721749e+01f	9.624456e+00f	-1.927935e+01f
57	-1.884629e+01f	-1.492471e+01f	-3.104690e+01f	-2.292073e+01f
58	1.302283e+00f	1.112015e+01f	-1.663091e+00f	2.115508e+00f
59	2.489330e+00f	2.539989e+01f	-6.583021e+00f	8.081338e+01f
60	2.511405e-01f	4.815754e+00f	1.869065e-01f	5.738816e+00f
61	4.690960e+00f	1.061526e+01f	7.124635e+00f	-9.537401e+00f
62	-4.940240e+00f	4.921678e+00f	2.173046e+01f	4.852204e+00f
63	-7.682037e+00f	1.437732e+01f	4.045655e+01f	5.271817e+01f
64	-5.582238e+01f	3.809919e+00f	1.701020e+01f	-3.842353e+01f
65	4.335150e-01f	7.173625e-01f	5.716141e+00f	-1.483467e+00f
66	-1.887343e+01f	1.375183e+00f	-2.224252e+01f	1.485839e+01f
67	4.440818e+00f	3.797212e+00f	-6.636334e+00f	-3.661210e+00f
68	1.161608e+00f	2.516729e+01f	-6.612689e+00f	3.902880e+01f
69	5.355930e+00f	-5.289291e+00f	4.502131e+01f	-2.627135e+01f
70	-7.977249e+01f	-8.232895e+00f	-1.162735e+02f	3.295274e+01f
71	-1.441911e+01f	1.024908e+01f	-1.576822e+01f	1.072019e+01f
72	1.712986e+01f	4.179140e+01f	-5.814886e+00f	4.088091e+01f
73	-2.911865e+00f	-1.509857e+01f	-1.424130e+01f	-1.891752e+01f
74	3.493282e+00f	5.736425e+00f	2.427680e+00f	8.170983e-01f
75	-2.174843e+00f	-2.318189e+00f	-1.540144e+01f	6.063041e+00f
76	6.041290e+01f	6.993306e+01f	-3.003879e+01f	1.685289e+02f
77	6.605937e+00f	-1.792933e+01f	5.167519e+01f	8.996784e+00f
78	8.896298e+00f	5.102569e+00f	-2.127043e+01f	5.257586e+01f
79	1.238322e+01f	-7.333014e+00f	2.483858e+01f	1.446348e+00f
80	-7.349715e+01f	-2.143580e+01f	-1.509891e+01f	5.480368e+01f
81	-2.055109e+00f	-6.509520e-01f	-1.644456e+01f	-1.147320e+01f
82	-3.475819e+01f	2.349188e+01f	-6.625946e+01f	1.429955e+00f
83	-1.197055e+00f	1.922565e+00f	-8.884270e+00f	-2.468740e+00f
84	-6.344799e+01f	2.527560e+01f	-5.604682e+01f	-3.014129e+01f
85	-7.675814e+00f	-2.276068e+01f	1.039212e+01f	-7.174798e+00f
86	-8.910252e+01f	1.345742e+02f	-1.572299e+02f	3.291177e+01f
87	-2.604298e+01f	3.734848e+01f	-4.586042e+01f	3.797831e+00f
88	4.485603e+00f	-1.853267e+01f	-4.241403e+00f	1.104383e+01f
89	-6.504644e+00f	-4.541655e+01f	-3.721288e+01f	-2.076201e+01f
90	-3.380043e+00f	7.383989e+00f	-4.793192e+00f	-1.331655e+00f
91	3.027233e+00f	-2.887969e+00f	-5.145234e+00f	-2.647340e+00f
92	-1.985596e+00f	-3.623952e+01f	2.455942e+00f	6.063980e+01f
93	-7.175469e+00f	9.048038e+00f	1.783442e+01f	-1.844755e+01f
94	-2.123359e+01f	3.778030e+01f	-4.079358e+01f	3.962847e+01f

95	5.302966e-02f	1.130616e+00f	1.285206e+01f	1.289455e+01f
96	-6.031443e+00f	1.062359e+01f	-2.598964e+00f	-1.179437e+01f
97	-4.690765e+01f	-3.282092e+00f	3.942076e+01f	-1.992904e+01f
98	-1.633853e+01f	1.430039e+01f	-4.476931e+01f	6.340025e+00f
99	-5.362904e+00f	-9.219887e-01f	3.769846e+00f	-1.075939e+00f
100	-7.666621e+01f	2.776053e+01f	4.013647e+01f	-5.836412e+01f
101	-6.604718e+01f	-1.045560e+01f	1.874482e+02f	-9.877850e+01f
102	-1.820392e+01f	7.968429e-02f	-2.106621e+01f	-7.976777e+00f
103	-4.791987e+01f	-1.975986e+01f	2.743540e+01f	-3.477007e+01f
104	-1.343250e+01f	-1.735530e+01f	2.851516e+00f	-1.793200e+01f
105	2.300427e+00f	-1.049200e+02f	2.025416e+01f	-7.292495e+01f
106	1.123309e+01f	9.307873e-01f	9.826490e+00f	-2.068632e+00f
107	1.426521e+00f	-2.604758e+01f	-2.016302e+00f	-1.185248e+01f
108	3.566378e+01f	5.230002e+00f	1.590986e+01f	5.840561e+00f
109	1.108218e+02f	-8.324540e+01f	9.517146e+01f	-1.112273e+02f
110	1.020885e+00f	-3.804494e+00f	3.306611e+00f	-5.192638e+00f
111	2.075151e+01f	-4.624358e+01f	3.415155e+01f	1.637074e+00f
112	-6.573619e+01f	-7.319780e+00f	-4.839073e+01f	-3.879312e+01f
113	-2.660107e+01f	5.437686e+00f	-1.091847e+02f	-4.340979e+01f
114	-6.588006e+00f	3.632881e+00f	-1.447060e+01f	2.822483e+00f
115	9.133369e+00f	-3.843127e+00f	-2.748470e+00f	-1.813885e+01f
116	-1.630414e+01f	9.061513e+00f	1.494428e+00f	2.949062e+01f
117	-1.346091e+01f	-3.059415e+01f	4.014598e+01f	-7.611642e+01f
118	-3.284071e+01f	6.700513e-01f	-7.987959e+01f	4.425238e+01f
119	-7.951221e-01f	3.695445e+00f	4.022542e+01f	2.107323e+01f
120	-2.743797e+01f	-2.264292e+01f	-5.985368e+01f	-4.651924e+01f
121	3.379884e+01f	3.971936e+00f	-1.562857e+02f	-6.777834e+01f
122	-2.706210e+00f	-1.185812e+00f	-6.892154e+00f	-5.375547e+00f
123	2.990784e+01f	2.851436e+00f	-5.598698e+00f	-4.791245e+01f
124	3.804678e+00f	-6.670058e+00f	-7.543237e+00f	7.871152e+00f
125	3.068575e+00f	-4.805424e+01f	2.243452e+01f	-2.454581e+01f
126	2.673559e+00f	8.312793e-01f	-5.612770e-01f	1.928470e+01f
127	-1.018784e-01f	-1.734256e+01f	8.015221e+01f	4.912214e+01f
128	-7.029315e+01f	-3.396966e+00f	-9.215202e+00f	-1.757423e+00f
129	-5.111988e+00f	3.115451e+00f	2.493005e-01f	1.722551e+00f
130	6.221227e+00f	1.361065e+01f	-7.017313e+01f	1.080508e+01f
131	3.967444e-01f	1.946329e+00f	-3.983733e+00f	2.962061e+00f
132	4.981498e+00f	1.207967e+01f	-1.111744e+01f	-5.853736e+00f
133	9.254358e-02f	4.333448e+00f	1.001746e+01f	-8.329934e+00f
134	-2.473052e+01f	2.337822e+00f	-6.680203e-01f	4.417503e+01f
135	-8.905037e+00f	-4.899843e+00f	1.554351e+00f	-4.582358e-01f
136	-2.373562e+00f	-5.077421e+00f	6.082286e+00f	-1.197413e+01f
137	-1.124587e+01f	-9.164964e+00f	-1.421781e+01f	1.095421e+01f
138	6.229926e+01f	2.746668e+01f	2.874645e+00f	-3.131371e+01f
139	1.453926e+01f	1.169905e+01f	-6.108858e+00f	3.269911e+01f
140	2.071638e+01f	1.253891e+01f	5.166270e+00f	2.076083e+01f
141	5.729747e+01f	-1.734672e+01f	-1.642786e+01f	-6.970825e+00f
142	8.925461e+00f	6.834633e+00f	-8.100564e+00f	-2.623063e+01f
143	1.055552e+01f	4.434007e+00f	-1.143398e+00f	-1.100516e+01f
144	-1.250868e+02f	4.272156e+01f	-1.881339e+01f	1.232183e+01f
145	-2.228594e+01f	4.071894e+01f	9.753891e+00f	-8.282732e+00f
146	-3.278488e+01f	-1.447801e+01f	-8.555849e+00f	3.017109e+01f
147	3.988797e-01f	5.820558e+00f	-9.059520e+00f	1.800231e-01f
148	-4.913155e+01f	8.712673e+00f	-4.711021e+01f	4.488874e+01f
149	-1.049373e+00f	3.121366e+01f	7.601249e+01f	-2.527371e+01f
150	-1.888543e+02f	-6.553314e+00f	-2.751895e+01f	7.131720e+01f
151	-5.291015e+01f	7.235156e-01f	-4.696699e+01f	2.729982e+01f
152	-1.752879e+01f	7.456645e+00f	-3.887018e+00f	1.220160e+01f
153	-5.084573e+01f	-4.632529e+01f	-6.089034e+01f	-3.183292e+01f
154	9.781092e+00f	8.025480e+00f	1.510336e+00f	-4.972064e+00f
155	-1.018678e+01f	-4.683022e+00f	-6.294638e+00f	-3.824615e+00f
156	-3.112019e+00f	7.266704e-01f	-5.110612e+00f	8.982461e+00f
157	4.078359e+00f	2.427756e+01f	7.437158e+00f	-3.409283e+01f
158	-5.107452e+01f	-4.662268e-01f	-5.428381e+00f	-1.750780e+00f
159	3.076901e-01f	-2.362309e+00f	-6.159365e+00f	5.071189e+00f
160	-2.165909e+01f	1.132642e+02f	-4.600035e+01f	-6.259497e+00f
161	-2.934526e+01f	1.479210e+01f	-4.959803e+00f	-1.142350e+01f
162	5.331546e+01f	1.310906e+02f	-1.732934e+02f	4.101749e+01f
163	-1.000669e+01f	2.899943e+01f	-6.179684e+01f	-1.546127e+01f
164	3.981433e+01f	4.304431e+01f	3.809388e+00f	-6.635926e+01f

165	1.299529e+01f	5.647130e+00f	6.504213e+00f	-1.683990e+01f
166	1.177363e+01f	-8.083778e+00f	-8.775627e+01f	-1.191059e+01f
167	-3.444707e+00f	-7.519234e+00f	-1.113360e+01f	3.028802e+00f
168	2.104631e+01f	5.629979e+01f	-4.830819e+00f	1.988586e+01f
169	-1.442214e+01f	4.659875e+00f	-5.054874e+01f	8.280144e+00f
170	1.488777e+02f	1.897744e+02f	1.099179e+01f	-3.835738e+01f
171	6.497350e+01f	2.055889e+01f	-5.038729e+01f	-4.478764e+00f
172	8.368832e+00f	7.273671e+00f	1.012722e+01f	-5.151683e+01f
173	2.862624e+01f	3.519567e+00f	-3.759276e-01f	-1.977781e+01f
174	4.866736e+01f	5.684104e+01f	-1.359630e+01f	-1.470820e+02f
175	1.611762e+00f	1.534478e+00f	-3.790378e+00f	-4.604877e+01f
176	-1.703138e+02f	2.100686e+02f	2.387944e+01f	-3.577670e+01f
177	-1.226857e+02f	9.784676e+01f	-3.382297e+01f	-6.534503e+01f
178	-5.407039e+00f	4.451594e+01f	-2.765289e+01f	-7.728964e+01f
179	-2.508013e+01f	2.465974e+01f	-5.400587e+00f	-1.971823e+01f
180	-5.671938e+01f	5.354270e+01f	7.699358e+00f	-1.833650e+01f
181	-3.416914e+01f	7.245964e+00f	-2.100003e+01f	3.567788e+00f
182	-4.084135e+01f	8.593622e-01f	-2.095057e+01f	1.275056e+01f
183	-1.631933e+01f	2.516509e+01f	2.364433e+01f	2.276656e+00f
184	-2.225595e+01f	8.088742e+01f	-3.504063e+01f	5.657921e-01f
185	-8.914178e+01f	2.153607e+01f	-1.429584e+02f	-8.033673e+01f
186	4.566643e+01f	5.556253e+01f	2.821412e+01f	-3.871313e+01f
187	-6.890999e+00f	9.559111e+00f	-6.557432e+01f	-1.110234e+01f
188	-6.875913e+00f	4.019174e+00f	-6.069849e+00f	-1.143891e+01f
189	-2.436351e+01f	2.432122e+01f	1.045277e+01f	-8.411496e+01f
190	-9.498792e-01f	-4.213404e+00f	-1.404884e+01f	-3.196807e+01f
191	9.203326e-01f	1.152253e+01f	5.661743e+00f	-8.115625e-01f
192	-1.823661e+02f	-1.243622e+00f	7.826379e+01f	3.376929e+01f
193	-5.051270e+01f	1.523163e+01f	1.812113e+01f	-3.296370e+01f
194	-5.827927e+01f	1.805878e+01f	-5.517637e+00f	1.378874e+01f
195	2.929571e+01f	-4.126551e+01f	-3.114973e+01f	3.915338e+00f
196	-5.823452e+01f	6.536344e+00f	-1.257728e+01f	3.202623e+01f
197	8.875800e+00f	-1.542554e+01f	1.245588e+01f	-7.185780e-01f
198	-6.673718e+01f	1.363886e+02f	-8.770198e+01f	4.210150e+01f
199	-1.731568e+01f	3.459954e+01f	-1.178726e+01f	1.773182e+01f
200	-7.558144e+01f	-2.149077e+01f	2.170759e+01f	-1.694419e+01f
201	-2.228052e+01f	-8.436509e+00f	-3.878604e+01f	3.249622e+00f
202	-3.313989e+00f	3.706884e+00f	-4.626763e+00f	1.054091e+00f
203	2.307668e+01f	2.106004e+00f	-5.879541e+01f	5.416113e+01f
204	2.513297e+01f	2.423261e+01f	6.439422e+00f	3.246154e+01f
205	1.712485e+01f	5.334249e+01f	4.648046e+01f	-2.719465e+01f
206	2.321403e+00f	2.756384e+01f	-4.143797e+01f	3.302007e+01f
207	-3.027530e+00f	5.355422e+00f	-1.842946e+01f	-3.311219e+00f
208	-3.091694e+01f	-1.115365e+01f	6.871840e+00f	1.773173e+01f
209	-2.222453e+01f	-3.428719e+01f	9.459258e-01f	-5.984766e+01f
210	-4.692722e+01f	5.979692e+01f	-6.974178e+01f	7.171124e+01f
211	1.119312e-01f	9.897666e-01f	-3.442615e+01f	-3.163636e-01f
212	-6.347123e+01f	4.582921e+01f	4.269857e+00f	6.134549e+01f
213	-3.669745e+01f	-7.546339e+01f	8.495920e+01f	1.210883e+01f
214	-1.754275e+02f	1.064665e+02f	-1.015302e+02f	2.097872e+02f
215	-1.350057e+02f	4.454966e+00f	-6.787295e+01f	3.018392e+01f
216	7.846435e+00f	1.572743e+01f	-2.464054e+01f	-3.582338e+01f
217	1.277370e+01f	-6.232156e+01f	-5.415674e+01f	-8.968465e+01f
218	-7.500269e+00f	1.117709e+01f	-8.346478e+00f	1.032481e+01f
219	2.850878e+01f	1.674574e+01f	-3.200722e+01f	2.839867e+00f
220	-8.896938e+00f	3.361208e+01f	6.939054e+00f	-1.233075e+01f
221	5.326577e+00f	6.082571e+01f	9.778513e+01f	-9.985917e+01f
222	-3.030131e+01f	6.312125e+01f	-3.338861e+01f	8.317265e+01f
223	-7.933629e+00f	-1.227354e+01f	4.051698e+01f	6.396040e+00f
224	-5.448136e+01f	1.197895e+01f	9.322082e-01f	-1.236984e+01f
225	-3.810226e+01f	2.165925e+01f	-3.028893e+01f	-5.339388e+01f
226	-1.168318e+01f	4.340295e+01f	-4.231737e+01f	2.956805e+01f
227	5.085625e+00f	4.497290e-01f	3.451980e+00f	-1.675972e+01f
228	4.420306e+00f	-1.301856e+01f	-9.836760e+00f	-2.033964e+01f
229	2.996057e+01f	5.087288e+01f	6.068042e+01f	-6.699461e+01f
230	-1.941310e+01f	1.881030e+01f	-2.514719e+01f	1.290587e+01f
231	-1.310282e+01f	9.216829e+00f	7.067560e+00f	-1.031713e+01f
232	-1.642498e+01f	1.504787e+01f	-3.923534e+01f	-5.632610e+01f
233	-8.609097e+01f	-1.900084e+01f	-1.327302e+02f	-5.077298e+01f
234	2.634005e+01f	6.159839e+01f	3.454085e+01f	6.128467e+00f

235	4.707952e+00f	-2.253467e+01f	-4.193263e+01f	-7.752820e+00f
236	2.050854e+01f	1.228599e+01f	7.663282e+00f	-2.301759e+01f
237	4.805712e+01f	4.623794e+00f	4.227706e+01f	-1.016092e+01f
238	-1.365608e+00f	1.453584e+01f	7.642492e+00f	-2.366271e+01f
239	3.096721e+00f	-8.564740e+00f	5.784921e+00f	7.943358e-01f
240	-8.218154e+01f	6.208510e+01f	-4.504518e+01f	-3.137702e+01f
241	-7.214978e+01f	3.619879e+01f	-1.027165e+02f	-2.151495e+02f
242	-8.633247e+00f	6.583620e+00f	-2.287796e+01f	2.528210e+01f
243	1.273039e+01f	3.011594e+01f	-7.804057e+01f	-2.763880e+01f
244	-1.665050e+01f	1.605340e+01f	-1.077351e+01f	2.452008e+01f
245	-1.903834e+01f	3.371863e+01f	6.645950e+01f	-7.711543e+01f
246	-1.363200e+02f	4.763965e+01f	2.690875e+01f	5.228906e+01f
247	-2.546481e+01f	1.141561e+01f	-1.223216e+01f	1.854151e+01f
248	4.866745e+01f	-1.050607e+01f	-1.425744e+02f	-1.273686e+02f
249	-8.395813e+00f	-1.698979e+02f	-2.407271e+02f	-2.015320e+02f
250	-3.837341e+00f	-1.716437e+01f	-4.656874e+01f	-6.054681e+01f
251	9.825644e+01f	-5.582764e+00f	-1.339953e+02f	-8.366324e+01f
252	1.980105e+01f	-1.704742e+00f	-1.793546e+01f	-3.977437e+01f
253	-2.764871e-02f	-1.366155e+01f	-8.006773e+01f	-1.266143e+02f
254	-2.996475e+01f	1.001527e+01f	4.287861e+00f	1.239964e+01f
255	2.666461e+01f	-2.968126e+01f	3.441442e+01f	-3.020520e+00f
256	-4.233699e+01f	-1.484702e+01f	4.059062e-02f	-1.633027e+01f
257	-1.111410e+01f	-9.731126e-01f	9.527723e+00f	-5.516987e+00f
258	4.515641e-01f	-5.674265e+00f	-1.094341e+00f	4.350498e+00f
259	2.104685e+00f	-3.616603e+00f	3.736672e+00f	1.159341e+01f
260	-8.995392e+00f	-2.120466e+01f	-2.188073e+01f	-1.192191e+01f
261	-2.355909e+00f	-1.107746e+00f	7.939836e+01f	-7.251102e+00f
262	-4.932747e-02f	-2.962788e-01f	1.727751e+00f	-1.541557e+00f
263	2.943852e+00f	-2.330070e+00f	7.634018e+00f	-2.765083e+00f
264	3.637004e+00f	-5.431600e+00f	6.324958e-01f	-9.760778e-01f
265	1.394120e+01f	-1.461755e+01f	2.673202e+01f	4.860985e+00f
266	2.040424e+01f	-3.099823e+00f	2.925377e+00f	1.543499e+01f
267	1.884806e+01f	-2.715036e-01f	-2.391444e+01f	3.604598e+01f
268	5.081792e+00f	-3.447244e+01f	3.655263e+01f	-9.087006e+00f
269	5.381700e+01f	-4.665833e+01f	1.370708e+02f	8.807584e+01f
270	8.118089e-01f	-7.748569e+00f	7.426198e+00f	-4.496917e+01f
271	5.169230e+01f	1.629703e+01f	1.916335e+01f	1.060153e+01f
272	-1.406776e+02f	-2.270749e+01f	3.397583e+00f	3.485284e+01f
273	-2.200432e+01f	-5.236344e+00f	-1.773632e+01f	7.316194e+00f
274	-2.466597e+01f	-2.022205e+00f	1.275651e-01f	-3.437297e+00f
275	-1.292198e+00f	5.853935e-01f	-1.525512e-02f	4.286126e+00f
276	-2.049102e+01f	4.227305e+00f	-9.154531e+00f	6.982212e+00f
277	-1.325410e+01f	-1.821595e+00f	5.288197e+01f	9.430734e+00f
278	-1.187919e+01f	6.400764e+00f	-6.699371e+00f	6.088468e-01f
279	-3.319604e+01f	1.299779e+01f	1.823846e+01f	1.888519e+01f
280	-1.640248e+01f	-3.403337e+00f	7.819552e+00f	8.250827e+00f
281	5.181305e-01f	-2.207756e+00f	5.031233e+00f	3.844315e+00f
282	-2.061355e-01f	1.609183e-01f	-5.825494e-01f	-2.495552e-01f
283	-3.001649e+00f	7.474183e+00f	4.865119e+00f	2.963460e+00f
284	6.400288e+00f	-2.124607e+01f	2.727135e-02f	5.301356e+01f
285	4.605424e+00f	4.347178e+00f	2.799114e+01f	-1.365209e+01f
286	7.999937e+00f	-6.428239e+00f	-8.075500e-02f	8.420396e+00f
287	1.138389e+00f	1.163593e+00f	9.876581e-01f	1.113594e+00f
288	5.591957e+00f	3.754222e+00f	-1.195820e+01f	-1.615310e+01f
289	-3.113197e+00f	-1.115794e+00f	3.156817e+01f	-3.188639e+01f
290	2.832300e+00f	5.729441e+00f	-5.245057e+01f	1.641788e+01f
291	-1.688206e+00f	1.411052e+00f	-1.129727e+00f	-7.011593e-01f
292	3.273468e+01f	-2.995054e+01f	-4.554807e+01f	-1.071842e+02f
293	-4.351562e+00f	5.844672e+01f	1.236382e+02f	-1.432800e+02f
294	-3.223483e+01f	1.172027e+00f	-5.028826e+00f	-4.088419e+01f
295	-7.117505e+00f	1.537117e+01f	1.508588e+01f	-4.666476e+01f
296	1.327628e+01f	-1.443850e+01f	2.731545e-01f	2.223950e+00f
297	9.589542e+01f	-6.422327e+01f	2.061943e+01f	-4.823240e+00f
298	-4.484424e+01f	-2.695908e+01f	-3.454128e+01f	-5.427109e+01f
299	1.061797e+01f	-8.258651e+00f	1.037549e+01f	-5.256356e+00f
300	2.594402e+01f	1.250810e+01f	5.222567e+01f	-2.394504e+01f
301	9.371501e+01f	1.059311e+00f	2.431561e+02f	-5.692429e+01f
302	-7.950269e+01f	-8.142823e+00f	1.068885e+01f	-1.562585e+02f
303	1.247188e+01f	-2.938435e+01f	5.110986e+01f	-3.395209e+01f
304	-6.051830e+01f	3.939079e+01f	9.411986e-01f	-1.887713e+01f

305	1.062171e+01f	-9.523574e+00f	3.908717e+00f	-1.624663e+01f
306	-2.193580e+00f	5.836875e+00f	2.070272e+00f	-4.974146e+00f
307	-3.228108e-01f	3.910665e+00f	-5.038945e-01f	-1.580646e+00f
308	-2.383173e+01f	-2.453479e+01f	3.611456e+01f	-6.541574e+00f
309	1.693035e+00f	-6.557175e+01f	4.143530e+00f	-1.048087e+02f
310	-1.036818e+00f	-2.239783e+00f	1.592637e+00f	-8.338949e-01f
311	-2.952086e+01f	9.114791e+00f	2.859642e+01f	-7.822580e+00f
312	3.074123e+00f	1.918763e+01f	-7.135770e+00f	3.524028e-01f
313	1.666495e+01f	1.997736e+00f	6.387029e+00f	2.860560e+00f
314	5.002252e-01f	-1.036329e+01f	-4.509731e+00f	-2.495539e+00f
315	-5.070354e+00f	2.398696e+00f	4.284402e+00f	1.008904e+01f
316	1.690337e+00f	-1.020579e+00f	1.703887e+01f	-1.054835e+01f
317	1.654246e+01f	-2.521169e+00f	5.420544e+01f	-5.150176e+01f
318	-2.119298e+01f	-2.158412e+01f	8.099245e-01f	-2.879742e+01f
319	-1.384991e+01f	-2.766740e+01f	2.373925e+01f	-7.315295e+00f
320	-1.336582e+02f	-8.226297e+01f	1.325215e+01f	-5.058432e+01f
321	-1.999098e+01f	-4.021140e+01f	3.671985e+01f	-1.005309e+01f
322	-9.526601e+00f	-5.382011e+01f	8.892178e+00f	-2.549077e+01f
323	3.898500e-01f	-2.825384e+00f	-1.506877e+00f	-5.686819e+00f
324	-2.142993e+01f	-1.817011e-01f	1.925709e+01f	-5.506707e+01f
325	-1.929650e+01f	-2.124934e+01f	1.807067e+02f	-9.147254e+01f
326	-2.556558e+00f	-5.206636e+00f	-2.516588e+01f	1.024313e+01f
327	3.912559e+00f	-2.519847e+01f	2.930760e+01f	-5.537000e+01f
328	-1.605634e+01f	-5.149089e+01f	-1.281055e+01f	-2.957010e+00f
329	5.703425e+01f	-3.726262e+01f	4.378047e+01f	-4.614997e+01f
330	-3.087193e+00f	-6.292802e+00f	1.544594e+00f	2.510291e+00f
331	1.985336e+01f	-3.563247e+00f	2.263825e+01f	-1.059652e+01f
332	-1.572959e+00f	1.488619e+01f	7.517023e+01f	4.656979e+00f
333	4.933411e+01f	-3.549252e+01f	2.676606e+02f	-3.943716e+01f
334	3.123707e+01f	1.730937e+01f	1.375933e+01f	5.334971e+00f
335	8.288226e+01f	2.817309e+01f	9.561280e+01f	-3.428425e+01f
336	-5.234233e+01f	-4.243114e+01f	-1.493310e+00f	2.116121e+01f
337	-8.883437e-01f	-2.338993e+01f	2.661225e+01f	7.872527e+00f
338	3.917457e+01f	8.673862e+00f	-1.567403e+01f	1.966639e+01f
339	8.394983e+00f	-4.367273e-01f	1.298718e+00f	2.332638e+00f
340	1.015396e+01f	-2.636494e+01f	1.688221e+01f	8.170255e+00f
341	-1.321133e+01f	-7.700809e+01f	1.051726e+02f	-1.372562e+01f
342	-5.574287e+01f	7.006753e+00f	-7.137306e+01f	1.420670e+01f
343	1.075656e+01f	-6.160712e+01f	2.831907e+00f	3.077424e-01f
344	8.036528e+01f	-1.556703e+01f	-3.651667e+01f	2.024620e+01f
345	1.796745e+00f	-9.879107e+00f	2.813387e+01f	-1.073788e+01f
346	1.222661e+01f	-4.897011e+00f	-2.411628e+00f	-2.044617e+00f
347	2.423739e+00f	-9.954109e-01f	6.344218e-01f	-1.997399e+00f
348	-9.926308e-01f	-9.692344e+01f	1.802800e+01f	4.967787e+00f
349	1.661234e+01f	3.271424e+00f	7.752914e+01f	-1.213209e+02f
350	-1.472457e+01f	-9.397635e+00f	1.087097e+00f	7.848398e+00f
351	1.531066e+01f	5.261401e+00f	3.593806e+01f	-1.421995e+01f
352	-2.074474e+00f	-4.701074e+00f	-2.211484e+01f	-6.486559e+01f
353	5.977266e+00f	-3.059424e+01f	1.243887e+02f	-1.316213e+02f
354	1.709133e+00f	1.541384e+00f	-2.526852e+00f	-8.404149e+00f
355	-2.812556e+01f	1.444074e+01f	2.206622e+01f	-3.818631e+01f
356	-9.606567e+00f	1.913277e+01f	1.628632e+01f	-2.191588e+02f
357	-4.742059e+01f	-6.004596e+01f	2.426472e+02f	-3.652139e+02f
358	4.253900e+00f	-7.718038e+00f	3.140039e+01f	-5.439022e+01f
359	-3.480640e+01f	5.597046e+00f	1.442441e+02f	-1.589188e+02f
360	6.903464e+01f	-6.833533e+01f	1.380096e+00f	-3.642692e+01f
361	2.093134e+02f	-1.543404e+02f	1.422983e+02f	-1.229340e+02f
362	3.490446e+00f	-7.848567e+00f	1.339885e+01f	-9.380313e+00f
363	4.356404e+01f	-4.943150e+01f	6.373597e+01f	-7.372006e+00f
364	-2.242620e+00f	2.363227e+00f	1.811335e+02f	-8.698129e+01f
365	1.208371e+02f	-1.134491e+02f	2.682845e+02f	-2.781379e+02f
366	1.465303e+01f	-4.762909e+01f	3.995422e+01f	-2.393050e+01f
367	7.128372e+01f	-1.515600e+02f	2.248420e+02f	-3.752792e+01f
368	-7.254371e+00f	-6.205770e+00f	1.368396e+01f	6.664135e+00f
369	-3.019798e+01f	-5.570705e+01f	3.870132e+01f	-3.780846e+01f
370	8.293087e-01f	-1.369847e+00f	-5.428192e-01f	1.541628e+00f
371	2.914141e+00f	-4.652065e+00f	3.429955e+00f	-1.689716e+01f
372	3.913741e+00f	-2.893460e+01f	9.192861e+01f	-4.850806e+00f
373	-8.455902e+00f	-1.900804e+02f	1.744550e+02f	-1.617185e+02f
374	-1.717477e+00f	-1.101012e+01f	1.331833e+01f	-2.928749e+00f

375	-3.156922e-01f	-3.709767e+01f	1.888053e+01f	-1.065270e+02f
376	7.870759e-01f	-9.670293e+00f	3.141125e+00f	-9.112347e+00f
377	1.855414e+01f	-4.800977e+01f	3.072922e+01f	-8.101409e+01f
378	4.290566e+00f	8.868722e-01f	-9.718518e-01f	1.137770e+00f
379	-4.554711e+00f	-3.291445e+01f	-3.098261e+00f	-3.751968e+00f
380	2.570423e+01f	-4.400358e+01f	6.252388e+01f	-6.295003e+00f
381	9.159167e+01f	-2.088931e+02f	5.388553e+01f	-1.099316e+02f
382	3.251133e+00f	-2.043773e+01f	4.538823e+00f	-8.949266e-01f
383	-5.319481e+00f	-1.272107e+02f	2.844507e+01f	1.430711e+01f
384	-4.306971e+00f	-4.238111e+00f	-1.634487e+00f	9.508975e+00f
385	-1.829651e-02f	-5.635989e+00f	7.904145e+00f	-2.497068e+00f
386	2.009468e+01f	6.890540e+00f	1.851679e+00f	5.280002e+00f
387	-1.055699e+01f	4.180188e+00f	-4.372098e+01f	4.455614e+01f
388	1.612135e+01f	-1.361035e+01f	4.144374e-01f	-7.659850e+01f
389	-5.266716e+01f	-2.004326e+01f	5.942376e+01f	2.647123e+00f
390	-3.093310e+00f	3.063833e+01f	-1.466189e+01f	-1.456706e+01f
391	-6.729136e+00f	5.948622e+00f	6.313396e+00f	-3.727629e+00f
392	1.405434e+01f	2.180688e+01f	-5.657064e+00f	2.415584e+01f
393	-4.786229e+00f	-1.670917e+00f	-2.628248e+01f	5.480457e+01f
394	1.173743e+02f	2.493141e+01f	-2.563304e+01f	3.874641e+01f
395	2.065721e+01f	4.445803e+01f	-5.993733e+01f	1.461265e+02f
396	2.950293e+01f	1.451982e+01f	2.597657e+01f	-5.107552e+01f
397	-3.881274e+01f	5.008808e+01f	5.637462e+01f	-5.518165e+00f
398	5.516830e+00f	8.160178e+01f	8.616129e+01f	-8.891920e+01f
399	4.239742e+00f	1.636732e+01f	1.901920e+00f	1.130635e+01f
400	-6.938871e+00f	-5.361866e+01f	-2.609818e+01f	2.967550e+01f
401	1.186089e+01f	-2.003815e+01f	3.999254e+01f	-1.556683e+01f
402	4.779575e+00f	1.476940e+01f	-1.324330e+01f	2.562427e+01f
403	-2.026271e+00f	3.018617e+00f	6.496635e+00f	1.616645e+00f
404	2.433960e+01f	3.263271e+01f	7.172142e+01f	-5.122205e+00f
405	1.550800e+01f	-7.535391e+01f	2.406361e+02f	-9.239929e+00f
406	-4.279413e+01f	1.551900e+01f	-3.609276e+01f	5.931329e+01f
407	-4.491935e+00f	1.280949e+01f	6.510652e+01f	-2.804217e+01f
408	-1.228014e+00f	1.520215e+01f	-4.210004e+00f	7.827713e+00f
409	-8.821086e+00f	2.604311e+01f	2.531644e+01f	-2.531382e+01f
410	1.456450e+01f	-3.025596e+00f	-1.673524e+01f	1.655199e+01f
411	-9.664453e+00f	2.405569e+01f	-1.981051e+01f	2.932251e+01f
412	-1.734855e+01f	2.221209e+01f	6.298586e+00f	-3.535759e+01f
413	-6.335560e+00f	7.015432e+01f	1.233713e+02f	-1.291291e+02f
414	-1.580743e+01f	4.816009e+00f	4.943432e+00f	-2.168502e+01f
415	-3.243454e-01f	3.353358e+01f	4.075500e+01f	-7.346035e+00f
416	-1.353716e+01f	-2.366930e+00f	6.246557e+00f	-8.563261e+01f
417	1.444983e+01f	1.909878e+01f	4.292479e+00f	-1.809886e+01f
418	-1.766911e+01f	8.252804e+01f	-2.627814e+01f	-3.745637e+01f
419	-1.299793e+01f	1.459759e+01f	-5.399992e+00f	4.770630e+00f
420	9.978232e+01f	2.791992e+00f	-1.951453e+01f	-2.610208e+02f
421	1.811604e+01f	3.385145e+01f	-1.345550e+01f	-1.082997e+02f
422	-3.298956e+01f	7.200571e+01f	-3.182132e+01f	-1.468429e+02f
423	-4.879557e+01f	-1.833121e+01f	-1.033310e+01f	-5.143038e+01f
424	1.571514e+01f	4.947201e+01f	9.140464e+00f	-4.456269e+01f
425	1.043287e+01f	-1.138545e+01f	2.042072e+01f	-1.692118e+01f
426	-2.223107e+00f	3.351655e+01f	-1.549696e+01f	-8.332043e+01f
427	-2.340801e+01f	5.804561e+00f	-3.094010e+01f	2.158077e+01f
428	5.594800e+01f	1.123502e+02f	7.225366e+01f	-1.977648e+02f
429	8.724786e+01f	1.939756e+01f	7.225141e+01f	-1.111740e+02f
430	-9.932788e+01f	1.285685e+02f	2.310413e+01f	-2.679491e+02f
431	-6.520186e+01f	5.677448e+00f	-3.399880e+01f	-1.179790e+02f
432	-7.350035e+01f	8.443907e+01f	-8.544090e+00f	3.212865e+01f
433	-3.852507e+01f	-6.510648e+00f	-6.490796e+00f	-1.686561e+01f
434	-1.581413e+01f	1.620759e+01f	-1.147920e+01f	-1.211709e+01f
435	4.618664e+00f	1.977408e+00f	2.324445e+00f	-5.373918e+00f
436	6.077433e+01f	2.666435e+01f	-6.231514e+00f	-7.690803e+01f
437	3.503428e+01f	4.997464e+01f	9.006268e+01f	-1.767684e+01f
438	-6.585789e+00f	6.189503e+00f	-1.918143e+01f	-2.546400e+01f
439	9.797235e+00f	3.513052e-01f	9.761535e+00f	-1.371822e+01f
440	1.512284e+01f	9.689066e+01f	-9.495427e+00f	1.177964e+01f
441	3.950151e+00f	1.885789e+01f	3.020031e+01f	-7.823894e+01f
442	-2.079835e+01f	1.148478e+01f	-3.167937e-01f	-1.506680e+01f
443	7.014849e+00f	-3.686678e+00f	9.248325e-01f	-7.169533e+00f
444	1.051680e+01f	3.336847e+01f	1.303980e+01f	-7.701772e+01f

445	2.290351e+01f	1.616450e+02f	1.397829e+02f	-1.775781e+02f
446	-1.076573e+02f	7.422704e+01f	-1.220930e+00f	-7.131834e+01f
447	4.215919e+01f	-7.195023e+00f	6.703672e+01f	-3.315619e+01f
448	-4.686045e+01f	-1.280709e+01f	2.513502e+01f	3.355340e+01f
449	-1.622197e+01f	-1.669920e+01f	2.784143e+01f	4.811460e+01f
450	5.023046e+00f	3.122398e+00f	-8.585894e+00f	1.819677e+01f
451	2.870848e+00f	1.574091e+00f	-1.628366e+01f	9.853959e+00f
452	2.054036e+01f	-2.661060e+01f	8.027773e-01f	-2.943704e+00f
453	4.122062e+01f	-9.073490e+01f	8.774212e+01f	-2.721096e+01f
454	-3.317490e+00f	1.476908e+01f	-8.258421e+01f	6.891264e+01f
455	-1.634003e+01f	-1.436659e+01f	2.158062e+01f	1.444229e+01f
456	-1.119620e+01f	1.886837e-01f	-3.171735e-02f	-6.059623e+00f
457	-9.840658e+00f	3.603443e+01f	2.732906e+01f	-4.397656e+01f
458	2.291438e+01f	2.029326e+01f	2.336518e+00f	7.966622e+00f
459	-2.805850e+00f	-6.303710e+00f	-1.523161e+00f	5.334238e+01f
460	2.667533e+00f	9.719355e+00f	4.486612e+01f	-3.741927e+01f
461	5.336722e+01f	1.178521e+02f	1.736633e+02f	-1.022076e+02f
462	-1.763592e+01f	-2.136992e+00f	4.119778e+00f	-3.190218e+01f
463	-1.065662e+00f	1.896643e+01f	4.085415e+01f	-1.136046e+01f
464	-5.728620e+01f	-7.071723e+01f	-7.885262e+01f	1.394004e+02f
465	3.077403e+00f	-1.211988e+02f	1.368623e+02f	7.106351e+01f
466	-1.357660e+01f	7.484928e+00f	-7.498009e+01f	9.820176e+01f
467	7.924682e+00f	-3.414455e+01f	2.622110e+01f	1.562070e+00f
468	4.023158e+01f	-1.185262e+02f	9.678011e+01f	5.477391e+01f
469	-5.128909e+01f	-2.501539e+02f	2.691713e+02f	-2.556378e+01f
470	-2.049165e+01f	-6.106120e+01f	-9.722127e+01f	1.370337e+02f
471	-9.313046e+01f	-1.104646e+02f	6.431312e+01f	3.609277e+01f
472	-5.107183e+00f	-2.720469e+01f	3.902756e+00f	3.423293e+01f
473	3.583994e+01f	2.604219e+01f	7.102895e+01f	-9.212521e+01f
474	-6.382868e+00f	-2.691912e+00f	-4.191895e+01f	3.637857e+01f
475	-7.426709e+00f	-9.163099e+00f	3.908282e+01f	-2.287922e+01f
476	-1.406348e+01f	-6.821640e-02f	1.030378e+02f	-7.003305e+01f
477	-2.738800e+00f	2.768921e+00f	2.926173e+02f	-1.707727e+02f
478	-5.672770e+00f	1.709496e+01f	1.086244e+01f	3.725180e+01f
479	7.659022e+00f	2.906890e+01f	1.977121e+02f	-4.312979e+01f
480	-4.690416e-01f	-3.209879e+01f	-2.151103e+01f	-2.469377e+01f
481	2.244858e+01f	9.469790e+00f	-1.407788e+01f	-9.920224e+01f
482	-6.564308e+00f	1.230836e+01f	2.802306e-02f	2.094978e+00f
483	6.703163e+00f	1.656538e+01f	-3.785893e-01f	-1.452693e+01f
484	6.253483e+01f	-1.937788e+01f	-2.517318e+01f	-1.388232e+02f
485	6.041099e+01f	4.520276e+01f	1.493713e+02f	-2.242399e+02f
486	-3.068370e+01f	-1.392866e+01f	-1.822093e+01f	-3.780522e+01f
487	-9.072470e+00f	7.364742e+01f	6.605718e+01f	-6.122920e+01f
488	1.424084e+01f	3.196169e+01f	2.011089e+00f	-3.095317e+01f
489	1.210985e+02f	-1.217729e+01f	-1.729840e+01f	-4.087893e+01f
490	-6.278011e-01f	5.754099e+00f	7.068807e+00f	-2.470455e+01f
491	1.396013e+01f	-7.763794e+00f	-1.118907e+01f	-1.074099e+01f
492	1.186175e+02f	5.206248e+01f	4.674505e+01f	-1.100414e+02f
493	2.136846e+02f	-2.356897e+01f	2.012665e+02f	-5.148911e+01f
494	-7.414952e+00f	2.156318e+01f	4.298984e+01f	-1.538220e+02f
495	3.379526e+01f	-2.575916e+01f	5.402335e+01f	-1.460367e+01f
496	-5.257572e+01f	-3.146810e+01f	-2.457557e+01f	9.562953e+00f
497	-1.127376e+01f	-2.347442e+01f	3.421125e+01f	-7.069389e+01f
498	-1.043137e+01f	4.503954e-02f	-1.200073e+01f	1.732812e+01f
499	2.537802e+00f	-2.246646e+00f	-2.640296e+01f	-1.131004e+01f
500	3.288895e+01f	-9.622323e+00f	4.725278e+01f	-5.887745e+00f
501	5.681990e+01f	-1.868427e+01f	2.460396e+02f	-4.490611e+01f
502	-2.007582e+01f	2.698408e+01f	-4.952689e+01f	6.775954e+01f
503	3.474986e+01f	-1.520015e+01f	4.861251e+01f	2.986283e+01f
504	-3.099471e+01f	1.239172e+00f	-5.726177e+01f	-4.343630e+01f
505	-2.328212e+01f	-3.147305e+01f	-1.375312e+02f	-1.802631e+01f
506	-3.324767e+00f	-4.511247e+00f	-2.892406e+00f	-6.238190e+00f
507	4.184345e+01f	-5.300082e+01f	2.385303e+01f	5.772085e-01f
508	3.128453e+01f	2.092512e+01f	1.579939e+01f	-2.140039e+01f
509	4.304324e+00f	-1.083049e+01f	8.852475e+01f	-3.909897e+01f
510	3.048393e+01f	-3.726225e+01f	3.465858e+01f	3.573012e+01f
511	7.702124e+01f	-1.731066e+02f	2.035811e+02f	2.434030e+01f
512	-1.324538e+02f	6.693506e+01f	4.118715e+00f	6.169838e+01f
513	1.679245e+01f	1.735733e-01f	8.314161e-01f	8.487957e+01f
514	-4.391596e+01f	3.673513e+01f	-9.908871e+01f	9.457977e+00f

515	-1.160614e+01f	2.212174e+01f	-5.307086e+00f	-4.948171e+00f
516	2.949734e+00f	-3.581507e+01f	3.104855e+01f	1.337450e+02f
517	-3.496254e+00f	-3.557597e+01f	1.705503e+02f	1.188636e+02f
518	6.922087e+00f	6.489554e+00f	3.518541e+00f	3.995038e+01f
519	-8.223513e+01f	-3.499005e+01f	2.618531e+01f	2.823101e+01f
520	7.405940e+00f	-1.439287e+01f	4.799878e+01f	1.357872e+02f
521	1.758828e+02f	-5.172195e+01f	5.008850e+01f	1.436737e+02f
522	3.857748e+01f	3.107783e+01f	-3.756261e+01f	3.312200e+01f
523	2.653239e+01f	-1.030429e+01f	9.523264e-01f	6.269112e+01f
524	1.093450e+02f	-1.205489e+02f	1.348202e+02f	2.515242e+02f
525	1.792166e+02f	-1.435439e+02f	2.174966e+02f	8.706808e+01f
526	5.538205e+01f	-1.444197e+01f	-1.078822e+00f	1.392515e+02f
527	5.361964e+01f	-3.398829e+01f	6.420087e+01f	-1.722211e+00f
528	-1.696716e+02f	1.426236e+02f	-3.128430e+01f	2.772169e+02f
529	-7.183598e+01f	8.137905e+01f	1.240362e+01f	1.019299e+02f
530	-6.179289e+01f	6.165622e+01f	-1.439240e+01f	7.615507e+01f
531	-9.057697e+01f	2.030451e+01f	1.228551e+01f	1.310544e+01f
532	-3.015379e+01f	9.787966e+01f	9.758345e+00f	1.123044e+02f
533	-5.134047e+01f	7.251836e+01f	6.226471e+01f	5.664281e+01f
534	-8.965919e+01f	2.862096e+01f	-1.201933e+01f	1.936544e+00f
535	-1.948891e+02f	7.430205e+01f	1.320043e+02f	4.420006e+01f
536	-1.017108e+01f	4.684011e+01f	1.025091e+01f	1.359086e+02f
537	7.037966e+01f	6.747733e+00f	-4.356126e+01f	4.855159e+01f
538	-4.400116e+00f	2.524936e+00f	-4.001873e-01f	2.454993e+01f
539	-9.162676e+00f	6.792388e+00f	1.007019e+01f	3.424553e+01f
540	5.245608e+00f	-4.892959e+01f	1.907192e+01f	8.129413e+01f
541	6.505585e+01f	-9.268250e+00f	6.919937e+01f	1.248285e+01f
542	1.938480e+01f	-1.367059e+01f	7.129734e+00f	1.844953e+01f
543	-3.081289e+01f	8.040442e+01f	4.294506e+01f	2.116449e+00f
544	-6.075689e+01f	7.209251e+01f	-1.040092e+02f	3.269313e+01f
545	1.175057e+00f	1.526615e+01f	-2.598910e+01f	2.049989e+01f
546	-4.979608e+01f	1.078994e+02f	-2.544138e+02f	6.949106e+01f
547	-7.832911e+01f	3.704396e+01f	-7.953941e+01f	1.460774e+01f
548	8.788995e+00f	4.059314e+01f	3.023145e+01f	3.571615e+01f
549	2.156585e+01f	-6.178663e+00f	4.641425e+01f	1.794503e+01f
550	-4.170870e+01f	4.736156e+01f	-8.879990e+01f	4.114155e+01f
551	-1.332707e+01f	2.278458e+01f	9.917879e+01f	3.011320e+01f
552	-5.035087e+00f	2.785377e+01f	2.926025e+01f	5.029836e+01f
553	-1.150210e+01f	1.096912e+01f	4.450495e+01f	8.867838e+01f
554	3.794104e+00f	1.013845e+02f	-5.693449e+01f	2.790562e+00f
555	-2.250740e+01f	5.974445e+01f	1.607032e+01f	1.555546e+01f
556	-4.563849e+00f	4.049814e+01f	1.625074e+02f	1.170957e+02f
557	8.647525e+01f	-8.539795e+01f	1.993176e+02f	1.226163e+02f
558	-1.172520e+01f	3.096998e+01f	1.651977e+01f	1.567937e+01f
559	1.318220e+01f	3.298785e+01f	6.888927e+01f	1.077794e+02f
560	-1.609080e+02f	1.463501e+02f	-4.356193e+01f	9.040104e+01f
561	-4.254871e+01f	3.020395e+01f	-7.156489e+00f	1.255086e+01f
562	9.237153e+00f	6.159613e+01f	-6.591632e+01f	-2.440261e+01f
563	-3.247116e+01f	1.724962e+01f	8.516510e+01f	2.188443e+01f
564	-2.652324e+01f	5.337551e+01f	2.299432e+01f	4.042635e+01f
565	6.453709e+01f	8.788883e+01f	1.272164e+02f	-8.136165e-01f
566	1.355491e+00f	8.502541e+01f	9.446342e+01f	-1.391914e+01f
567	-9.835211e+01f	1.216394e+02f	2.933503e+02f	1.223392e+01f
568	-7.494526e+01f	1.835042e+01f	-1.146169e+02f	3.260764e+01f
569	-1.688256e+01f	8.704553e+00f	-1.002956e+01f	5.711757e+01f
570	1.338541e+00f	1.208847e+01f	-6.497414e+00f	5.721006e+01f
571	-7.136398e+00f	7.660382e+01f	1.004949e+02f	1.399631e+02f
572	-5.183258e+00f	1.276906e+00f	1.029973e+01f	7.173253e+01f
573	8.131329e+00f	-1.833745e+00f	5.485801e+01f	3.282408e+00f
574	-1.136446e+01f	7.221272e+01f	5.606171e+01f	5.170399e+01f
575	-4.593139e+01f	8.080035e+01f	2.180268e+02f	5.440265e+01f
576	-3.250035e+01f	2.905584e+00f	2.418200e+01f	2.936612e+01f
577	-1.482085e+01f	-3.623267e+01f	7.802638e+00f	-1.738306e+01f
578	2.422929e+00f	-1.251183e+01f	-1.926798e+01f	8.277201e+00f
579	1.176706e+01f	-4.246539e+01f	-3.796022e+01f	-5.778217e+00f
580	5.232240e+01f	-4.083886e+01f	-8.827531e+00f	1.115151e+02f
581	-3.494143e+00f	-4.948751e+01f	8.912379e+01f	-1.029095e+02f
582	-1.768457e-01f	-5.682303e+00f	-2.818602e+01f	3.948042e+01f
583	-1.158690e+01f	-1.710530e+01f	1.877142e+01f	-3.999368e+01f
584	-2.887593e-01f	1.655600e+01f	-2.628887e+01f	1.445902e+02f

585	3.500855e+01f	-5.858514e+01f	-1.131978e+01f	2.162531e+01f
586	-7.159005e+00f	-1.604995e+01f	6.748738e+00f	4.447649e+01f
587	2.333647e+01f	-1.805039e+01f	-5.358218e+01f	7.861269e+00f
588	7.641125e+01f	-1.216720e+02f	-2.729175e+01f	3.253617e+02f
589	-4.509085e+01f	-1.997913e+02f	8.240949e+01f	1.282353e+02f
590	3.236950e+01f	1.875263e+01f	-9.896556e+00f	1.560374e+02f
591	-1.079539e+01f	-7.371985e+01f	-2.642347e+00f	2.332956e+01f
592	-8.893890e+01f	6.279792e+00f	2.191141e+01f	9.372643e+01f
593	-8.856963e+00f	2.873086e+01f	-1.462998e+00f	2.603564e+01f
594	-2.730403e+00f	-2.456888e+00f	-3.782933e+00f	2.243099e+01f
595	2.233877e+01f	-4.459153e-01f	-6.676023e+00f	-2.186212e+00f
596	-1.669250e+01f	-1.903957e+01f	-7.486783e+00f	9.509269e+01f
597	-4.369968e+00f	-5.759187e+00f	1.972721e+01f	-1.268405e+01f
598	-3.727077e+01f	3.365496e+01f	-3.660106e+01f	2.169342e+01f
599	-4.938757e+01f	2.896924e+01f	2.432996e+01f	2.689671e+01f
600	7.147046e+01f	-7.244958e+01f	-4.319358e+01f	3.652143e+01f
601	-1.497505e+00f	-7.577834e+01f	-3.357653e+01f	-1.007243e+01f
602	7.872799e+00f	-1.189171e+01f	-3.483538e+00f	-6.981084e-01f
603	-7.721202e+00f	-1.521004e+01f	1.083080e+01f	2.769260e+00f
604	9.429761e+01f	-1.882617e+02f	-2.397583e+00f	2.032479e+02f
605	1.626658e+01f	-3.111600e+01f	2.255600e+01f	5.751735e+01f
606	5.666544e+01f	-4.202782e+01f	-1.149615e+00f	5.020944e+01f
607	2.662613e+01f	-9.129365e+00f	8.788309e+01f	6.862697e+00f
608	-4.875762e+00f	1.363286e+01f	-2.174384e+01f	1.150052e+00f
609	1.087799e+01f	-4.067959e+01f	2.921849e+01f	-6.541483e+01f
610	-1.019043e+01f	-1.785134e+01f	-1.337112e+02f	3.791756e+01f
611	-1.717970e+01f	1.719966e+01f	1.545373e+00f	-1.376534e+01f
612	2.628613e+01f	-3.561285e+01f	1.732816e+01f	-2.839317e+01f
613	2.876343e+01f	-4.493835e+01f	8.031085e+01f	-1.459015e+02f
614	-1.498503e+01f	3.250471e+01f	-2.072002e+01f	-2.687050e+01f
615	-1.318204e+01f	1.001220e+02f	7.121446e+01f	-1.172166e+02f
616	2.588269e+01f	-1.206868e+01f	-9.985402e+00f	6.600942e-01f
617	9.896849e+01f	-1.306661e+02f	4.304852e+01f	-4.971230e+00f
618	8.875648e+00f	1.836581e+01f	-2.297161e+01f	-1.467474e+01f
619	3.105870e+01f	-4.082000e+01f	-2.889599e+00f	-1.967733e+01f
620	3.277287e+01f	-1.114657e+02f	1.665104e+01f	4.037086e+01f
621	1.233802e+02f	-1.863446e+02f	2.710405e+02f	-1.223438e+01f
622	3.457119e+01f	-1.945728e+01f	-1.646727e+00f	1.956390e+01f
623	3.158605e+01f	-4.178214e+01f	1.238023e+02f	-3.669991e+01f
624	-3.077055e+01f	4.699765e+01f	-2.841250e+01f	2.030783e+01f
625	9.419230e+00f	-2.182541e+01f	-4.896216e+00f	-7.793738e+00f
626	1.996350e+01f	1.074746e+01f	-2.342288e+01f	1.130534e+00f
627	1.198775e+02f	5.693146e-01f	2.244793e+01f	-5.130892e+00f
628	-6.982616e+01f	-3.736679e+01f	8.064318e+01f	9.367391e+01f
629	-5.948353e+00f	-3.560788e+01f	4.786756e+01f	-2.691214e+01f
630	-1.582199e-01f	1.336584e+01f	3.044120e+01f	2.104574e+01f
631	3.477216e+01f	7.435226e+01f	1.656265e+02f	6.363906e+00f
632	-5.215322e+00f	-9.123239e+00f	-3.413515e+01f	-5.218996e+00f
633	-7.506986e+01f	-8.389351e+01f	-3.624172e+01f	-2.635443e+01f
634	4.493553e+00f	2.371679e+00f	9.732565e+00f	4.605050e+00f
635	1.080104e+01f	1.469317e+00f	8.283530e+01f	4.209332e+01f
636	3.938502e+01f	-7.018873e+01f	2.774764e+01f	2.695159e+01f
637	1.744025e+01f	-1.949167e+02f	1.561661e+02f	1.016101e+01f
638	3.288594e+01f	-3.143294e-02f	6.883347e+01f	5.438435e+01f
639	5.324142e+01f	-4.976404e+01f	3.057836e+02f	8.935638e+01f
640	-3.418599e+01f	4.122675e+01f	-9.941812e+00f	-4.309741e+01f
641	-5.907380e+00f	6.801458e+00f	-9.647188e+00f	1.187326e+01f
642	-6.015008e+00f	6.819878e+01f	-1.275437e+02f	-1.375766e+02f
643	1.724411e+01f	-1.986115e+00f	-4.198265e+01f	-2.264639e+01f
644	4.607205e+00f	-5.968939e+00f	1.888238e+01f	2.008500e+01f
645	2.440482e+01f	4.807918e+00f	2.292846e+01f	2.237110e+01f
646	1.399966e+01f	5.306919e+00f	-3.527641e+01f	-4.800431e+01f
647	-8.708187e+00f	7.335740e-01f	8.404271e+00f	1.739331e+01f
648	6.240478e+01f	7.641223e+00f	-1.101703e+00f	-2.287003e+01f
649	3.997194e+01f	6.148356e+00f	-1.972559e+01f	5.332725e+01f
650	1.512299e+02f	1.448944e+02f	-9.359822e+01f	-4.979952e+01f
651	2.157206e+01f	5.791470e+01f	-3.541057e+01f	1.958220e+01f
652	3.809779e+01f	1.201499e+01f	6.840080e+01f	1.132443e+02f
653	1.985629e+02f	4.148804e+00f	5.494341e+01f	2.265434e+01f
654	4.645415e+01f	1.677235e+01f	-5.458559e+00f	-2.487264e+00f

655	3.460564e+01f	-5.420698e+00f	4.493125e+00f	8.991035e+00f
656	-5.637494e+01f	4.266245e+01f	-6.566439e+01f	8.902560e+01f
657	-1.451242e+01f	1.351668e+01f	-9.723356e-02f	1.516706e+01f
658	4.558960e+00f	5.244886e+00f	-7.749979e+01f	-7.657774e+01f
659	1.993906e+01f	-1.586769e+01f	2.420889e+00f	-1.687672e+01f
660	-7.168363e+00f	8.924122e+00f	-1.141982e+01f	2.825212e+01f
661	-4.766348e+00f	2.792422e+01f	3.782442e+00f	8.973313e+00f
662	-3.233113e+01f	3.417500e+01f	-1.892772e+01f	4.290440e+01f
663	-6.535619e+01f	7.055284e+00f	1.859326e+01f	1.564925e+01f
664	2.108429e+00f	9.003813e+00f	-4.237339e+00f	2.451527e+01f
665	-2.368505e+01f	-7.170089e+01f	-2.085383e+01f	-3.944243e+00f
666	2.345321e+01f	3.645747e+01f	-2.520285e+01f	-1.721897e+01f
667	1.984758e+00f	1.010539e+00f	2.402391e+00f	7.886534e+00f
668	1.190016e+01f	1.824377e+01f	1.833447e+01f	1.810231e+01f
669	3.788043e+01f	2.974007e+01f	2.811974e+01f	1.092218e+01f
670	4.172004e+00f	2.387792e-02f	-3.952451e+00f	7.630627e+00f
671	-1.001433e+01f	1.538656e+01f	4.175638e+00f	-4.585331e+00f
672	-2.287842e+01f	9.066156e+01f	-7.794315e+01f	-1.352516e+02f
673	-4.000587e+01f	5.376698e+00f	-3.176727e+01f	-3.425835e+01f
674	1.964819e+01f	2.009793e+02f	-3.156604e+02f	-1.487206e+02f
675	-8.675649e+00f	9.703707e+01f	-1.015607e+02f	-2.610002e+01f
676	2.267632e+01f	2.988729e+01f	-2.856794e+01f	-4.487603e+01f
677	-9.497101e+00f	-1.093178e+01f	5.081721e+00f	-1.349082e+01f
678	8.298846e+01f	8.432370e+01f	-1.542493e+02f	-1.004258e+02f
679	1.232216e+01f	2.556110e+01f	-1.795818e+01f	-3.690431e+01f
680	2.547219e+01f	1.170200e+02f	-1.121492e+00f	-4.084301e+01f
681	-7.994299e+00f	4.058447e+01f	-2.162321e+01f	-7.862995e+00f
682	1.351650e+02f	3.175757e+02f	-6.109109e+01f	-1.194583e+02f
683	2.510471e+01f	1.758947e+02f	-4.691559e+01f	-4.184299e+01f
684	-1.438172e+01f	4.715656e+01f	2.488607e+01f	1.841790e+00f
685	1.854158e+01f	-1.392844e+01f	2.434599e+01f	4.487618e+01f
686	-8.353729e+00f	1.252510e+02f	2.907210e+01f	-1.328502e+02f
687	-5.188773e-01f	5.765542e+01f	-3.269815e+00f	-8.404601e+00f
688	-1.893661e+01f	6.592638e+01f	-6.285763e+01f	-4.119950e+01f
689	-2.200151e+01f	1.959695e+01f	-2.910393e+01f	-6.875514e+00f
690	1.172625e+02f	1.517899e+01f	-2.245068e+02f	-1.622411e+02f
691	4.540557e+01f	5.221880e+01f	-6.879441e+01f	-9.025370e+01f
692	-2.112530e-01f	6.508348e+00f	-3.103070e+00f	-1.581080e+01f
693	5.769729e+00f	2.794693e+01f	9.261814e+00f	7.091106e-01f
694	3.051217e+01f	-5.761347e+00f	-4.445641e+01f	-6.676211e+01f
695	2.692282e+01f	1.425102e+02f	9.871372e+01f	7.614314e+00f
696	-5.985685e+01f	6.847592e+01f	-3.459780e+01f	2.466585e+01f
697	-2.264522e+01f	-8.892400e+01f	-6.759137e+01f	4.448546e+01f
698	3.244344e+01f	7.758714e+01f	-8.399716e+01f	-2.076788e+01f
699	-2.088205e+01f	3.650559e+01f	3.887870e+00f	1.204043e+01f
700	-7.026672e+00f	2.254978e+01f	-1.732586e+01f	4.410594e+00f
701	-1.010293e+01f	-1.848022e+01f	-1.006922e+01f	1.573011e+00f
702	-2.814080e+00f	4.439491e+01f	-1.202272e+00f	-2.928211e+01f
703	-1.969506e+01f	4.159978e+01f	5.524605e+01f	5.450475e+01f
704	-8.687387e+01f	-2.556090e+01f	-1.996338e+01f	-1.312210e+00f
705	2.968735e+01f	-3.594611e+01f	-3.702018e+01f	-1.299556e+01f
706	3.209087e+01f	-1.836816e+01f	-6.419936e+01f	-2.970847e+01f
707	1.191986e+02f	-7.866699e+01f	-1.490345e+02f	2.007144e+01f
708	1.054978e+01f	-1.449298e+01f	-7.246943e+00f	3.540858e+01f
709	-1.700039e+00f	-2.580526e+01f	1.229472e+01f	-3.724393e+01f
710	-3.196374e+01f	1.436016e+01f	-1.002639e+02f	5.569424e+01f
711	2.204428e+01f	-2.610426e+00f	-5.137908e+01f	9.519783e+00f
712	3.114646e+01f	7.394151e-01f	-1.102246e+01f	2.476493e+01f
713	1.475815e+01f	-4.938419e+01f	-7.581536e+01f	2.129209e+01f
714	1.803427e+01f	2.487227e+01f	-6.365866e+01f	1.909500e+01f
715	1.485247e+02f	-1.495795e+01f	-1.482918e+02f	7.322524e+01f
716	6.856821e+01f	-4.389209e+01f	-6.429262e+01f	1.234642e+02f
717	1.681183e-01f	-3.659481e+01f	2.683502e+01f	2.477735e+01f
718	1.909363e+01f	-1.294971e+01f	-2.743396e+01f	4.425002e+01f
719	1.674998e+01f	1.328632e+01f	-1.036409e+02f	2.913704e+01f
720	-1.549621e+02f	-1.009961e+02f	-3.852526e+01f	7.045767e+01f
721	-1.799400e+00f	-3.632102e+01f	-1.598709e+01f	4.395170e+01f
722	-2.196762e+01f	-3.020142e+01f	-6.717113e+01f	6.003951e+01f
723	1.293511e+02f	-2.785736e+01f	-5.540552e+01f	-5.658658e+01f
724	-5.744523e+01f	-3.625014e+00f	-1.877765e+01f	4.384721e+01f

725	-1.051910e+01f	-1.309656e+01f	1.533924e+01f	-6.056771e+00f
726	-3.420113e+01f	-4.058830e+01f	-6.130788e+01f	1.316666e+02f
727	1.202284e+01f	7.463804e+00f	-3.418734e+01f	1.488603e+01f
728	-2.040201e+01f	-2.300564e+01f	-4.989170e+00f	3.111719e+01f
729	-8.457126e+01f	-1.551687e+02f	-5.432684e+01f	4.528644e+01f
730	-2.067588e+01f	-1.164697e+01f	-2.083741e+01f	4.561675e+01f
731	6.301585e+00f	-2.525336e+01f	-6.080367e+01f	2.821207e+01f
732	4.060559e+01f	-3.783954e+01f	-3.256785e+01f	4.525366e+01f
733	-4.047721e+01f	-4.355825e+01f	1.123731e+01f	-2.759612e+01f
734	-1.900935e+01f	1.285299e+01f	-3.847023e+01f	5.553098e+01f
735	-2.386966e+00f	2.359607e+01f	1.024443e+01f	-5.438670e+00f
736	-2.102680e+01f	4.756369e+00f	-4.273252e+01f	-4.205120e+01f
737	-6.155722e+00f	-9.052700e+00f	-1.091449e+01f	-8.254757e+00f
738	-4.775027e+01f	6.261572e+01f	-1.687977e+02f	-7.462393e+01f
739	6.297286e+01f	-1.441816e+01f	-2.424700e+01f	-8.135682e+01f
740	-3.353405e+00f	-2.571037e-01f	5.070043e+00f	-1.052775e+01f
741	-3.456714e+01f	-9.634171e+00f	2.384632e+01f	-7.663716e+01f
742	3.081483e+00f	1.697663e+00f	-6.015709e+01f	-2.811456e+01f
743	-7.470695e+00f	9.528769e+00f	5.579212e-01f	-3.356922e+01f
744	-9.649081e-01f	1.208077e+01f	-1.380898e+01f	-2.050679e+01f
745	-1.719909e+01f	-6.557355e+01f	-9.308704e+01f	-1.857477e+01f
746	5.622174e+01f	9.379105e+01f	-1.022290e+02f	-7.367463e+01f
747	1.891843e+01f	2.616319e+01f	-4.092460e+01f	4.378316e+01f
748	1.772411e+01f	-3.890635e+01f	-1.006467e+00f	1.512140e+01f
749	8.419118e+01f	-2.921475e+01f	9.103231e+01f	-4.236228e+01f
750	-1.792336e+01f	5.595153e+01f	-2.074405e+01f	-2.094756e+01f
751	5.963704e+00f	-3.668378e+00f	1.148626e+01f	-1.586078e+01f
752	-3.759831e+01f	-2.005268e+01f	-1.836153e+01f	1.301835e+01f
753	-1.117114e+00f	-3.470609e+01f	-1.001216e+02f	-2.057817e+01f
754	9.504166e+01f	-6.187603e+01f	-8.802662e+01f	-1.649348e+01f
755	2.182206e+02f	2.756000e+01f	-1.500531e+02f	-4.312000e+01f
756	-5.273233e+00f	-3.216387e+00f	1.858440e+01f	2.122498e+01f
757	-1.897845e+01f	-9.157072e+00f	-3.859215e+00f	-1.287125e+01f
758	-1.006869e+01f	2.203850e+01f	-6.940772e+01f	7.316298e+01f
759	7.761732e+01f	-9.614064e+00f	-7.837193e+00f	1.445539e+01f
760	-3.035535e+01f	-8.395415e+01f	-6.668788e+01f	-9.265003e+01f
761	-2.182724e+01f	-2.808022e+02f	-1.652848e+02f	-8.148761e+01f
762	8.459266e-01f	1.128841e+01f	-3.188462e+01f	-1.929439e+01f
763	4.620980e+01f	-8.350466e+01f	-6.697655e+01f	-5.275536e+01f
764	4.258047e+00f	-4.672506e+01f	-1.372055e+01f	-1.890357e+01f
765	-1.705400e+01f	-6.659692e+01f	-3.352232e+01f	-4.446761e+01f
766	1.354901e+01f	2.054241e+01f	1.463595e+01f	-3.304188e+00f
767	2.005803e+01f	3.969085e+01f	1.273515e+02f	-2.151401e+01f
768	-2.768731e+01f	2.192357e+01f	1.020171e+01f	1.308240e+01f
769	-2.530553e+01f	-1.920673e+01f	7.919061e+01f	2.200243e+01f
770	5.738914e+00f	2.162897e+00f	-2.873176e+01f	2.914064e+01f
771	2.405375e+01f	4.464695e+00f	8.758464e+00f	-2.965664e+00f
772	5.744418e+00f	-4.934349e+01f	7.709294e+01f	-3.960596e+01f
773	-1.569737e+01f	-1.192908e+02f	2.787154e+02f	-7.418080e+00f
774	7.943162e+00f	-1.785791e+00f	4.772649e+00f	1.459797e+01f
775	-7.284598e+00f	-5.395229e+01f	5.284584e+01f	-1.232359e+01f
776	2.990790e+01f	-1.655663e+01f	2.484006e+01f	1.786429e+01f
777	1.318371e+02f	-3.439078e+01f	1.282451e+02f	5.549863e+01f
778	8.888863e+01f	2.677984e+01f	5.229078e-01f	7.591013e+01f
779	9.692883e+01f	5.620651e+01f	2.668667e+01f	1.512826e+01f
780	1.182132e+02f	-9.778113e+01f	1.650855e+02f	5.154694e+01f
781	1.862310e+02f	-1.204073e+02f	3.094156e+02f	2.680133e+01f
782	6.197972e+01f	3.738727e+00f	1.343672e+01f	5.930763e+01f
783	1.823352e+02f	-2.559928e+01f	1.408731e+02f	2.765486e+01f
784	-4.199327e+01f	2.600017e+01f	2.918049e+01f	1.309890e+02f
785	-1.151137e+01f	1.219965e+01f	4.331160e+01f	-9.185301e+00f
786	-6.359554e+00f	-8.666116e+00f	-1.502041e+01f	3.043577e+01f
787	-3.335657e+01f	-2.566159e+01f	4.867952e+00f	1.228191e+01f
788	6.147111e+00f	2.123772e+01f	4.903119e+01f	3.002020e+01f
789	-4.685352e+01f	6.573169e+01f	2.136317e+02f	-3.663751e+01f
790	-1.863419e+01f	1.718433e+01f	1.799129e+00f	4.642439e+01f
791	-1.393831e+02f	-2.633188e+01f	7.290552e+01f	-2.070424e+01f
792	2.336415e+01f	-4.584238e+01f	1.719092e-01f	1.712473e+01f
793	1.503782e+01f	8.289335e+00f	2.780900e+01f	4.665477e+01f
794	5.158942e+00f	3.138076e+01f	-1.898593e+00f	1.563417e+01f

795	1.252478e+01f	1.039489e+01f	8.384733e+00f	7.290882e+00f
796	8.149488e+01f	-1.149853e+02f	2.431959e+01f	1.552979e+02f
797	5.309566e+01f	-2.342405e+01f	1.464157e+02f	4.704935e+01f
798	1.355413e+01f	-3.378127e+01f	-3.589083e+01f	5.062108e+01f
799	3.493534e+01f	-3.549863e+01f	1.802167e+01f	-1.738651e+00f
800	1.655334e+00f	1.964788e+01f	-3.871947e+01f	-2.363826e+00f
801	8.576201e-01f	-6.430894e+01f	2.235762e+01f	-4.363389e+01f
802	4.628930e+01f	2.021932e+01f	-1.419929e+02f	4.048859e+01f
803	3.261779e+00f	-3.621087e+00f	-8.350210e+00f	-1.408727e+01f
804	2.723528e+01f	-1.622667e+01f	1.879206e+01f	-2.955630e+01f
805	4.279874e+01f	-1.779919e+01f	8.458615e+01f	-1.740235e+02f
806	1.563372e+01f	-4.376210e-01f	-1.866151e+01f	-1.674219e+01f
807	7.679243e+00f	1.764167e+00f	4.262370e+01f	-1.440146e+02f
808	4.760362e+01f	-2.824437e+01f	-1.599418e+01f	-1.258467e+01f
809	1.125298e+02f	-3.826447e+01f	3.367866e+01f	-4.870871e+01f
810	7.971504e+00f	4.539820e+01f	-2.587281e+01f	9.929165e+00f
811	2.231303e+01f	-5.981276e+00f	7.602949e+00f	-6.737984e+00f
812	3.653819e+01f	-7.939540e+00f	2.442206e+01f	2.865491e+01f
813	2.572804e+01f	-7.670291e+01f	1.440385e+02f	-4.460611e+01f
814	-2.412106e+01f	1.463260e+01f	2.578654e+00f	-3.360564e+01f
815	5.429477e+01f	-2.163717e+01f	2.675753e+01f	-2.099842e+01f
816	-6.532527e+01f	2.613231e+01f	6.796822e+00f	6.309983e+01f
817	-5.957546e+00f	1.038128e+00f	1.223842e+01f	-1.610839e+01f
818	8.193707e+00f	2.391448e+01f	-3.058087e+01f	1.661432e+01f
819	2.923354e+00f	7.461344e+00f	2.237512e+01f	2.256000e+00f
820	-3.822870e+01f	3.382700e+01f	1.649283e+02f	3.133902e+01f
821	-6.620045e+01f	-5.311700e+00f	8.984978e+01f	-5.747396e+01f
822	-1.930665e+01f	-9.006734e+00f	2.422767e+01f	-9.428704e-01f
823	-2.354346e+01f	4.763775e+01f	9.236687e+01f	-4.489233e+01f
824	-1.332966e+01f	7.848492e+01f	-4.281089e+01f	3.261721e+01f
825	1.230018e+01f	-5.192026e+01f	-1.631666e+01f	-6.316978e+01f
826	-3.236750e+00f	2.120987e+01f	2.903601e-01f	5.090530e+00f
827	-5.182761e+00f	7.846030e-02f	2.414229e+01f	3.448927e+01f
828	2.824051e+01f	-2.182227e+01f	7.429716e+01f	1.441461e+01f
829	1.018105e+01f	-1.607008e+02f	-5.109291e+00f	-4.968994e-02f
830	-2.350919e+00f	-2.194157e+00f	1.673405e+01f	2.952935e-01f
831	-3.405037e+01f	-4.525275e+01f	1.045560e+02f	1.148525e+01f
832	-2.432734e+01f	-3.193792e+01f	2.480840e+01f	5.834962e+01f
833	-8.208164e+01f	-9.511692e+01f	6.617106e+01f	-7.874486e+01f
834	-2.092052e+00f	-1.349614e+01f	-1.292960e+01f	2.383340e+01f
835	-2.704547e+01f	-2.539488e+01f	2.758351e+01f	-2.579905e+01f
836	-4.097847e+01f	-9.674759e+01f	6.462234e+01f	-4.952875e+01f
837	-1.078055e+02f	-2.624438e+02f	2.011038e+02f	-1.997315e+02f
838	-4.276978e+01f	2.109215e+00f	-3.334932e+01f	2.937229e+01f
839	-8.298761e+01f	-7.621107e+01f	1.157998e+02f	-1.103885e+02f
840	3.758508e+01f	-2.054067e+01f	-3.983710e+01f	3.888942e+01f
841	1.053182e+02f	-2.056008e+01f	6.831894e+00f	-1.095161e+02f
842	1.420826e+01f	4.836735e+00f	-7.736090e+00f	1.153876e+01f
843	3.777174e+01f	5.920525e+00f	-1.598548e+01f	-7.408976e+00f
844	-8.075408e+00f	-6.357864e+01f	6.167083e+01f	1.241133e+02f
845	4.799581e+01f	-1.200705e+02f	9.182138e+01f	-6.006325e+01f
846	-5.954858e+00f	7.038715e-01f	-5.790468e+00f	4.340525e+01f
847	4.643439e+01f	-3.434743e+01f	1.823754e+01f	-1.542746e+01f
848	-8.056193e+01f	-1.389048e+02f	2.351175e+01f	1.710303e+02f
849	-4.415148e+01f	-3.132471e+01f	6.103727e+00f	4.658950e+01f
850	2.646888e+01f	-5.319800e+01f	-7.796865e+01f	9.609572e+01f
851	4.545875e+00f	-3.108286e+01f	-1.256057e+01f	1.816563e+01f
852	-3.055820e+01f	-8.358926e+01f	7.639609e+01f	5.447409e+01f
853	-4.430750e+01f	-7.407861e+01f	8.570035e+01f	-7.752283e+01f
854	-2.717723e+01f	2.656626e+01f	-1.035096e+02f	1.211826e+02f
855	-3.204031e+01f	3.543600e-01f	1.356996e+01f	-4.121263e+00f
856	1.462762e+02f	-1.629672e+02f	4.979584e+00f	1.103755e+02f
857	1.813507e+01f	-7.801218e+01f	-1.395607e+01f	-3.662534e+01f
858	4.155019e+01f	-1.212471e+01f	8.088035e+00f	6.044754e+01f
859	8.861943e+00f	-4.491681e+01f	-3.187201e+00f	-7.575119e-02f
860	-1.670837e-01f	-2.994097e+02f	1.045740e+02f	9.792484e+01f
861	-7.197754e+01f	-1.532460e+02f	3.307798e+01f	-8.419086e+01f
862	1.449431e+01f	-7.059361e+01f	3.134989e+01f	3.885006e+01f
863	8.258032e+01f	-1.012373e+02f	2.757929e+01f	3.387559e+01f
864	3.398041e+01f	-6.830631e+01f	-2.087918e+01f	-7.694563e+01f

865	3.246582e+01f	-2.186573e+02f	1.172295e+02f	-1.638100e+02f
866	-1.100929e+01f	-5.121780e+00f	-2.155409e+01f	-2.161070e+01f
867	-3.811477e+01f	-2.739832e+01f	4.981980e+01f	-1.449438e+02f
868	5.342378e+01f	-6.345115e+01f	1.689675e+02f	-1.119798e+02f
869	9.602051e-01f	-1.898181e+02f	1.989813e+02f	-3.619860e+02f
870	-3.988692e+01f	-2.872766e+00f	-1.269458e+01f	-1.204617e+02f
871	-6.063855e+01f	5.561716e+01f	7.929016e+01f	-3.057763e+02f
872	9.171507e+01f	-5.883910e+01f	-3.243724e+01f	-1.286652e+02f
873	1.907596e+02f	-2.040385e+02f	1.062597e+02f	-1.750176e+02f
874	-9.521598e+00f	-1.981121e+01f	-6.605047e+00f	-4.274698e+01f
875	4.255152e+01f	-7.141331e+01f	3.434788e+01f	-5.175042e+01f
876	1.191142e+02f	-4.384454e+01f	3.738463e+01f	-9.220525e+01f
877	9.113657e+01f	-2.180482e+02f	2.014158e+02f	-2.310953e+02f
878	1.726182e+01f	-6.445826e+00f	4.577249e+00f	-2.478416e+01f
879	3.000737e+01f	-7.890555e+01f	4.512649e+01f	-1.147188e+02f
880	-4.850410e+01f	-9.399017e+00f	1.014715e+02f	3.047922e+01f
881	5.439991e+00f	-8.001514e+01f	2.249002e+01f	-5.734633e+01f
882	3.927570e+00f	-8.132877e+00f	-3.980527e+00f	2.240579e+01f
883	1.191013e+01f	-2.376749e+01f	1.042589e+01f	-1.952318e+01f
884	-7.034888e+01f	-1.379250e+02f	2.702361e+02f	1.023775e+02f
885	2.405563e+00f	-1.559290e+02f	2.180431e+02f	-1.110229e+02f
886	-8.231149e+00f	-2.081702e+00f	7.771729e+01f	8.447720e+01f
887	-2.117809e+01f	-4.219151e+01f	5.489133e+01f	-5.590174e+01f
888	-9.937498e+00f	-7.646420e+01f	3.256841e+01f	-2.575710e+01f
889	-3.377592e+00f	-2.089105e+02f	-1.697136e+01f	-2.061761e+02f
890	-4.458464e+00f	-2.064615e+01f	-1.646459e+00f	-5.430994e+00f
891	3.598014e+01f	-9.946160e+01f	3.442486e+01f	-4.252667e+01f
892	7.640167e+01f	-2.246599e+02f	1.689923e+02f	6.801297e+01f
893	1.242609e+01f	-3.928396e+02f	6.638480e+01f	-1.537965e+02f
894	5.132978e+00f	-8.230289e+01f	7.979272e+01f	-2.631989e+01f
895	3.078100e+01f	-2.729869e+02f	1.947574e+02f	5.480072e+00f
896	-2.799760e+01f	-4.730762e+00f	-1.793505e+01f	1.025499e+02f
897	-1.752084e+00f	-1.008342e+01f	4.972482e+00f	1.337730e+01f
898	1.009280e+02f	4.410386e+01f	-3.090244e+01f	1.159987e+00f
899	8.682887e+00f	3.759020e+01f	-1.300923e+01f	6.528017e+01f
900	-1.092584e+01f	-1.614863e+01f	-8.958755e+00f	1.592620e+01f
901	-1.715158e+01f	-3.242145e+01f	8.285250e+01f	-2.912034e+01f
902	-1.389204e+01f	2.942801e+01f	-2.509853e+01f	1.072711e+02f
903	-4.301907e+00f	7.724734e+00f	5.678146e-01f	1.117320e+01f
904	3.453416e+01f	1.371431e+02f	-4.298758e+01f	8.070879e+01f
905	3.357521e+01f	4.010500e+01f	-2.202576e+00f	4.918327e+01f
906	2.279385e+02f	1.796524e+02f	-2.060144e+01f	1.343851e+02f
907	9.838353e+01f	1.746644e+02f	-1.447610e+01f	1.635639e+02f
908	2.801592e+01f	1.230933e+01f	3.468154e+01f	1.197100e+01f
909	3.723420e+01f	2.024563e+00f	1.598909e+02f	1.941511e+01f
910	3.969309e+01f	1.838511e+02f	6.145607e+01f	5.628458e+01f
911	2.607072e+01f	8.247676e+01f	5.580140e+01f	1.539209e+01f
912	-4.554895e+01f	-1.824279e+02f	-5.575848e+01f	1.585562e+02f
913	-1.458893e+01f	-5.080788e+01f	1.747289e+00f	3.375986e+01f
914	1.753119e+01f	-2.450553e+00f	-9.074928e+01f	9.838161e+01f
915	-6.360537e-01f	5.042346e+00f	-1.339492e+01f	2.057966e+01f
916	-1.052980e+02f	-4.067219e+01f	-5.481996e+00f	9.032536e+01f
917	1.831553e+01f	1.704221e+00f	4.757848e+01f	-2.088168e+01f
918	-1.591674e+01f	5.121291e+01f	-7.000573e+01f	1.927318e+02f
919	-2.527100e+01f	2.105769e+01f	-1.331635e+01f	3.876060e+01f
920	-4.532313e+01f	8.373482e+00f	-1.413524e+01f	8.107645e+01f
921	-2.435918e+00f	3.836693e+00f	-5.798696e+00f	1.310498e+01f
922	4.380313e+01f	4.899327e+01f	-4.883399e+01f	5.336309e+01f
923	-4.980548e-01f	5.279466e+01f	2.328701e+01f	6.443145e+01f
924	-4.423166e+00f	-1.092303e+01f	7.251961e+00f	8.440732e+01f
925	-1.283354e+01f	1.293243e+01f	2.905103e+01f	-1.383844e+01f
926	4.347485e+00f	4.221582e+01f	-3.842675e+01f	3.919870e+01f
927	-4.567650e+00f	9.623812e+00f	1.207033e+01f	7.418797e-01f
928	4.343932e+01f	7.134294e+01f	4.388439e+00f	-4.132276e+00f
929	-2.101004e+00f	1.599172e+01f	-6.849517e+00f	-8.455407e+00f
930	7.304959e+01f	2.006082e+02f	-1.456714e+02f	-4.234155e+01f
931	-1.132936e+01f	1.051540e+02f	1.851726e+00f	-2.098751e+01f
932	1.182452e+01f	-5.153357e+00f	2.063510e+00f	-7.432684e+01f
933	-2.249662e+01f	-1.587951e+01f	8.651531e+00f	-9.167120e+01f
934	4.757087e+01f	4.730294e+01f	-2.398663e+01f	-2.345738e+01f

935	-2.944503e+00f	-2.121625e+01f	-8.592415e-01f	-3.301051e+01f
936	-5.809109e+01f	4.834226e+01f	-5.387604e+00f	2.054737e+01f
937	1.981356e+01f	-3.852367e+01f	-3.919827e+00f	-4.096904e+01f
938	2.610822e+01f	1.946264e+02f	-5.082562e+01f	7.041968e+01f
939	3.016208e+01f	6.223083e+01f	1.029608e+01f	5.595626e+01f
940	5.653005e+01f	3.117116e+00f	1.776895e+01f	-5.069824e+01f
941	2.349273e+00f	-1.280223e+01f	1.738795e+01f	-3.042604e+01f
942	-2.741161e+01f	4.211334e+01f	-5.737218e+00f	-6.821308e+01f
943	-3.974626e+00f	-9.524748e+00f	1.775658e+00f	-2.409856e+01f
944	-4.035726e+01f	8.790712e+00f	1.969059e+01f	1.130087e+02f
945	-1.381554e+01f	1.962326e+00f	-4.584522e+00f	1.914155e+01f
946	-1.239851e+01f	3.769103e+01f	-1.036908e+02f	-5.320386e+01f
947	8.939028e+00f	7.000800e+00f	-1.829186e+01f	-1.449995e+00f
948	-1.815824e+01f	2.915338e+00f	2.138418e+01f	1.261184e+01f
949	3.214741e+00f	-1.828209e+01f	2.545336e+01f	-1.192453e+01f
950	1.243644e+01f	1.052582e+01f	-4.815324e+01f	6.559429e+01f
951	2.303979e+01f	3.458112e+01f	1.393319e+00f	2.014394e+00f
952	-9.741342e+01f	2.127838e+02f	-4.403180e+01f	1.250183e+02f
953	-1.281060e+00f	6.827779e+01f	-1.121834e+01f	3.792794e+01f
954	-1.797575e+01f	9.068700e+01f	-4.230067e+01f	6.291307e+01f
955	2.848770e+00f	3.803514e+01f	-7.395258e+00f	-1.693666e-01f
956	-1.416765e+01f	4.939830e+01f	-8.476540e+01f	7.132703e+01f
957	-4.534632e+01f	1.199923e+01f	2.495244e+01f	-7.785877e+01f
958	-1.773323e+01f	2.762570e+01f	2.099282e+00f	1.469839e+00f
959	3.620496e+00f	-1.193903e+01f	8.626517e+00f	4.930952e+00f
960	-7.298593e+01f	-6.710960e+00f	-1.165048e+01f	1.441259e+02f
961	-1.198211e+01f	-3.671595e+01f	2.725596e+00f	1.483689e+01f
962	-3.174024e+01f	-5.967281e+01f	-6.020750e+01f	5.907795e+01f
963	-1.052593e+01f	-1.570269e+01f	-8.629033e+01f	3.707693e+01f
964	-2.424240e+01f	1.756507e+01f	-3.863182e+01f	3.050188e+01f
965	-5.261983e+01f	-6.006301e+01f	1.238116e+02f	-8.669578e+01f
966	-3.913574e-01f	7.259784e+01f	-1.866658e+02f	1.772961e+02f
967	-3.437796e+01f	9.181270e+00f	-2.618374e+01f	3.512183e+01f
968	-1.155222e+01f	2.302230e+00f	-1.977294e+01f	5.599016e+01f
969	5.604462e+01f	-4.315877e+01f	-9.239677e+00f	-3.356361e+01f
970	2.199024e+01f	6.823550e+01f	-2.496532e+01f	7.178606e+01f
971	6.520518e+01f	2.798703e+01f	-3.052851e+01f	3.473392e+01f
972	9.726257e+00f	-1.769550e+00f	-8.242507e+00f	3.457922e+01f
973	-3.963404e+01f	2.661633e+01f	4.168720e+01f	-3.309643e+01f
974	-5.904578e+00f	1.068306e+01f	-1.914852e+01f	8.601739e+01f
975	1.026949e+01f	2.508734e+01f	-1.291526e+01f	-3.002816e+00f
976	-1.987645e+02f	-2.027293e+02f	1.471570e+01f	3.078370e+02f
977	1.166489e+01f	-8.063556e+01f	4.721591e+01f	1.691607e+02f
978	-1.204971e+01f	-1.963677e+02f	-1.530434e+02f	2.348594e+02f
979	6.878704e+00f	-6.473373e+01f	-4.436267e+01f	5.489179e+01f
980	-1.099679e+02f	-2.445468e+01f	2.317781e+01f	1.886992e+02f
981	-1.582926e+01f	-5.178455e+01f	1.164595e+02f	-8.350600e+00f
982	-6.173004e+01f	-3.057202e+01f	-1.549427e+02f	3.851968e+02f
983	-6.983607e+01f	1.252710e+01f	-2.993007e+01f	1.431717e+02f
984	-6.455817e+01f	-7.985500e+01f	2.732176e+01f	1.545720e+02f
985	-3.389105e+00f	-1.737665e+01f	8.641815e+00f	1.793043e+01f
986	-1.082667e+02f	-4.230530e+01f	-9.343213e+01f	9.824146e+01f
987	-1.671752e+01f	-8.321855e+00f	-3.322777e+01f	2.803246e+01f
988	3.784456e+00f	-8.419405e+01f	1.767135e+01f	7.743160e+01f
989	-5.121458e+01f	2.314851e+01f	1.058561e+02f	-1.060212e+02f
990	5.171604e+00f	5.652861e+01f	-1.605891e+02f	1.381751e+02f
991	-7.460727e+00f	3.657645e+01f	7.743556e+00f	2.313348e+01f
992	-5.493473e+01f	-5.676268e+01f	1.801532e+01f	-7.411072e-01f
993	2.306760e+01f	-2.067780e+01f	5.038124e+01f	-6.544560e+01f
994	8.032934e+00f	7.179582e+01f	-4.396809e+01f	9.931790e+00f
995	9.717121e+00f	-2.473879e+01f	-1.486118e+01f	-3.630994e+01f
996	3.638663e+01f	-7.411973e+00f	3.391744e+01f	-6.686446e+01f
997	-1.724561e+01f	2.994366e+01f	7.154828e+01f	-2.532613e+02f
998	2.283850e+01f	4.587559e+01f	-6.379286e+01f	2.264895e+01f
999	6.928734e+00f	-1.676284e+01f	-2.729474e+01f	-1.699161e+02f
1000	-1.423502e+01f	-1.059721e+01f	-1.126292e+01f	-5.592815e+01f
1001	6.551392e+01f	-2.878760e+01f	9.464539e+00f	-1.746977e+02f
1002	-8.054388e+00f	5.076021e+01f	-6.935185e+00f	2.150099e+01f
1003	3.306339e+01f	-9.456460e+00f	1.349558e-01f	-4.643474e+01f
1004	1.108146e+01f	-6.862247e-01f	1.216288e+01f	-3.612302e+01f

1005	9.055611e+01f	-5.146319e+01f	1.239097e+01f	-1.519000e+02f
1006	-7.393864e+00f	-3.992108e+00f	1.982524e+01f	-3.321019e+01f
1007	2.257887e+00f	-5.489872e+01f	-1.862823e+01f	-4.236289e+01f
1008	-1.802795e+02f	-7.061139e+01f	-3.673163e+01f	7.671021e+01f
1009	-3.259672e+01f	-2.710078e-01f	-1.773940e+01f	4.089658e+01f
1010	-9.618672e+00f	-3.806740e+01f	-9.396459e+00f	9.597201e+01f
1011	6.016460e+01f	-4.094289e+01f	-1.085322e+02f	3.031608e+01f
1012	1.623755e+01f	-4.422408e+01f	6.261455e+01f	1.042913e+02f
1013	-2.336336e+01f	-1.602995e+01f	6.543449e+01f	-4.103353e+01f
1014	1.155188e+01f	7.104852e+01f	-8.454871e+01f	2.138997e+02f
1015	8.585933e-01f	1.652802e+01f	-4.011930e+01f	2.972007e+01f
1016	-4.630685e+01f	3.170146e+01f	-4.765725e+01f	6.005632e+01f
1017	-5.831522e+00f	-1.042655e+02f	-2.752071e+01f	-1.285155e+01f
1018	-1.826684e+01f	-2.553606e+01f	-4.969689e+01f	2.659323e+01f
1019	-4.273684e-01f	-2.607804e+01f	-1.086372e+01f	3.341604e+00f
1020	1.295670e+01f	-2.975246e+01f	5.892236e+01f	7.497585e+01f
1021	-2.509461e+01f	-1.936021e+02f	1.012253e+02f	-6.859307e+01f
1022	2.677677e+01f	-7.885068e+00f	-4.971602e+01f	6.600209e+01f
1023	1.383562e+01f	-6.607554e+01f	9.244051e+01f	4.518500e+01f

/* SE_Shape5 */
/* dim=4x512 codewords */

index	codeword			
0	-7.224097e+01f	-3.415088e+02f	-2.299115e+02f	-1.744034e+02f
1	-2.534680e+00f	-8.452189e+01f	8.752303e+01f	4.117774e+00f
2	7.237285e+01f	-1.368329e+02f	1.104489e+02f	-3.564561e+01f
3	1.443610e+02f	-5.744830e+01f	4.372877e+02f	2.318452e+02f
4	1.959064e+01f	-1.440060e+02f	-5.479437e+01f	-9.330870e+01f
5	1.631901e+01f	1.279862e+01f	1.384809e+02f	-2.520641e+00f
6	4.287267e+01f	-6.124569e+01f	1.339805e+02f	-8.577949e-01f
7	-2.793676e+00f	-3.667525e+01f	1.860507e+02f	2.169447e+02f
8	-4.800562e+01f	-1.024903e+02f	-1.504208e+01f	-8.645311e+01f
9	-6.269104e+01f	3.641669e+01f	7.465507e+01f	3.056171e+01f
10	-6.933166e+00f	7.949766e+00f	7.609005e+01f	-1.528468e+01f
11	-2.276996e+01f	-5.166711e+01f	2.931263e+02f	-6.924191e+01f
12	-1.325288e+01f	-2.081453e+01f	-3.817722e+01f	-1.284207e+01f
13	-3.719302e+01f	4.500993e+01f	8.259805e+01f	4.819974e+01f
14	-5.318383e+01f	7.055368e+01f	3.185610e+01f	-1.789640e+01f
15	-1.313422e+02f	1.429318e+02f	2.367224e+02f	7.958197e+01f
16	5.395267e+01f	-1.080866e+02f	-6.613143e+01f	-1.557333e+01f
17	-1.595204e+01f	5.814026e+01f	-2.252008e+00f	1.147405e+02f
18	-4.164081e+01f	-1.227245e+01f	6.071333e+01f	6.675083e+01f
19	-3.153067e+01f	4.557214e+01f	1.979789e+02f	2.157673e+02f
20	5.288546e+00f	-3.179867e+01f	-3.397883e+01f	-1.869532e+01f
21	-2.131606e+00f	1.015456e+01f	6.321984e+00f	2.647363e+01f
22	3.029670e+01f	-4.451591e+01f	9.633846e+00f	2.840607e+01f
23	1.592112e+01f	6.653127e+01f	2.992664e+01f	1.879005e+02f
24	-2.526952e+00f	-1.481567e+01f	3.434562e+01f	2.688499e+01f
25	4.918232e+01f	-3.433073e+01f	6.526061e+01f	1.960139e+02f
26	-3.557672e+00f	-1.112320e-01f	8.785844e+00f	1.174452e+00f
27	1.353359e+01f	-1.566228e+01f	6.677322e+01f	3.630854e+01f
28	8.490892e-01f	4.311048e-01f	-6.172758e-01f	-9.782301e-01f
29	8.002045e+00f	-9.476031e+00f	-2.325300e+01f	3.149468e+01f
30	4.687713e+00f	2.598924e+01f	5.220760e-01f	2.929035e+00f
31	-6.343770e+01f	1.553681e+02f	1.570776e+01f	6.289817e+01f
32	-1.486321e+01f	-5.984330e+01f	-9.458195e+01f	-8.537872e+01f
33	-3.971132e+01f	7.518286e+01f	-1.913205e+01f	-1.877368e+01f
34	2.091882e+01f	-2.996994e+01f	1.332216e+02f	-9.442741e+01f
35	-2.555159e+01f	1.900131e+01f	1.717694e+02f	4.231897e+01f
36	1.108218e+02f	-1.194417e+02f	8.008218e+01f	-5.304565e+01f
37	1.307303e+01f	-4.308566e+01f	4.757629e+01f	2.153208e+00f
38	2.239965e+02f	-1.444678e+02f	2.584158e+02f	-1.042914e+01f
39	9.814158e+01f	-3.548009e+01f	1.074628e+02f	2.943421e+01f
40	-1.939095e+01f	-6.611009e+01f	-4.734568e+01f	-1.682286e+01f
41	-1.380464e+02f	2.547407e+02f	-6.818320e+01f	-1.990747e+01f
42	-7.462549e+01f	3.825741e+01f	-4.141285e+01f	-1.001897e+02f
43	-1.688903e+02f	4.756228e+01f	7.568243e+01f	-1.065248e+02f

44	3.343108e+01f	3.606890e+01f	4.908850e+01f	-8.135235e-01f
45	-4.946521e+01f	1.829455e+02f	8.058339e+01f	8.076106e+01f
46	7.071374e+01f	-6.934637e+01f	6.110162e+01f	-1.720418e+00f
47	-5.384036e+01f	4.764317e+01f	5.147501e+01f	6.526317e+01f
48	4.479188e+00f	-1.605094e+01f	-1.248164e+01f	-8.819100e+00f
49	-6.317758e+00f	7.358287e+00f	4.270894e+00f	9.629669e+00f
50	9.213771e+00f	-1.005811e+02f	1.262815e+01f	-3.760102e+00f
51	-2.418411e+01f	-1.463119e+01f	7.851159e+01f	7.979369e+01f
52	4.450998e+01f	-6.310098e+01f	2.729916e+01f	-6.437669e+01f
53	9.246837e+00f	-1.387984e+01f	1.757221e+01f	-1.082478e+01f
54	1.172888e+02f	-2.489718e+02f	8.006781e+01f	-3.493406e+01f
55	1.627668e+01f	-9.593005e+01f	-1.782783e+00f	3.854639e+01f
56	-4.137079e+00f	-3.500773e+01f	2.794730e+01f	-5.075122e+00f
57	-5.936260e+01f	4.451046e+01f	-4.620115e+01f	3.808519e+01f
58	1.158419e+01f	-5.250583e+00f	6.933043e+00f	-6.722890e+01f
59	-3.191115e+01f	3.854952e+01f	1.916814e+00f	-4.577449e+01f
60	-1.264834e+00f	-1.635877e+01f	1.323104e+01f	4.210530e+00f
61	-3.391734e+00f	3.986929e+01f	1.045820e+01f	5.532623e+01f
62	6.293894e+01f	-4.811902e+01f	-2.186943e-01f	-4.099194e+01f
63	-1.752476e+01f	1.364287e+01f	-2.054596e+00f	1.792832e+01f
64	7.071561e+01f	-1.422936e+02f	-7.218620e+01f	-1.798856e-01f
65	-2.955167e+01f	-3.957426e+01f	9.439397e+01f	1.223563e+00f
66	4.648792e+01f	6.407607e+00f	6.054667e+01f	-6.768115e+01f
67	1.636022e+02f	-1.130492e+02f	2.518087e+02f	1.851260e+01f
68	-2.879143e+01f	-5.044048e+01f	5.302629e+01f	-2.900628e+01f
69	-1.341742e+02f	-1.237346e+02f	2.820704e+02f	-1.095361e+02f
70	1.615991e+01f	-1.321822e+01f	2.401171e+01f	2.934057e+00f
71	-2.331807e+01f	-6.894558e+01f	9.753874e+01f	5.683086e+01f
72	1.562576e+01f	2.225604e+01f	-4.661019e+01f	-4.524290e+01f
73	-7.035220e+01f	8.564919e+00f	-1.277187e+02f	3.625515e+01f
74	1.910042e-01f	9.802090e+00f	-6.103825e+01f	-6.093312e+01f
75	-2.696470e+01f	1.506442e+01f	4.495411e+01f	-9.506558e+01f
76	3.697345e+00f	-3.365402e+01f	1.063137e+01f	1.555931e+01f
77	-4.344927e+01f	-9.241866e+01f	4.619951e+01f	-6.521626e+01f
78	-4.559556e+00f	-7.634112e+00f	4.784321e+00f	-8.512321e+00f
79	-3.375061e+01f	4.439852e+01f	4.818821e+01f	1.863087e+01f
80	1.921443e+01f	-1.751837e+01f	-6.005112e+00f	-8.508079e-02f
81	-1.092482e+00f	1.306226e+01f	2.446462e+01f	3.357338e+00f
82	5.835654e-01f	1.050529e+00f	4.001868e+01f	-8.844140e+00f
83	7.346025e+01f	6.506699e+01f	1.453645e+02f	-1.460647e+01f
84	-1.722713e+01f	-2.089215e+01f	-1.454264e+00f	-1.185815e+01f
85	-7.723459e+01f	-9.071578e+01f	5.967016e+01f	-6.039589e+01f
86	5.123377e+00f	-7.422465e+00f	5.301455e+00f	-8.380628e+00f
87	1.872406e+01f	2.295778e+01f	6.248983e+00f	1.766867e+01f
88	1.977208e+00f	-2.972606e+00f	-1.175285e+01f	-4.078745e+00f
89	-2.942713e+00f	-1.520415e+01f	6.202744e+00f	4.604567e+01f
90	3.850652e+01f	4.496532e+01f	-3.908395e+01f	-4.026641e+01f
91	-1.285768e+01f	8.987773e+00f	1.889578e+01f	-2.575549e+01f
92	6.071974e+01f	4.405277e+01f	-4.532100e+01f	-5.389964e+00f
93	4.156258e+01f	-7.434866e-01f	-6.491521e+01f	5.007715e+01f
94	3.944423e+00f	2.262858e+01f	-2.024760e+01f	2.498853e+00f
95	-1.906281e+00f	8.951632e+00f	-4.671535e+01f	-1.980983e+01f
96	5.930780e+01f	1.539171e+00f	-6.356366e+01f	-9.658236e+01f
97	-4.453417e+01f	-6.896203e+00f	1.354732e+02f	-8.891150e+01f
98	7.585672e+01f	2.059613e+01f	-1.046719e+01f	-3.156588e+02f
99	2.539343e+01f	-6.126494e+01f	5.102383e+01f	-7.952541e+01f
100	4.147787e+01f	-6.059567e+01f	6.076451e+01f	-2.818322e+00f
101	1.961121e+01f	-2.183078e+02f	2.779418e+02f	-2.354968e+01f
102	7.595151e+01f	-1.691409e+01f	3.655060e+01f	-5.627213e+01f
103	-2.808554e+01f	-8.980690e+01f	3.654807e+01f	5.321900e+01f
104	2.945167e+01f	3.108421e+01f	-1.684262e+02f	-1.826881e+02f
105	3.741101e+01f	8.907853e+01f	-1.531120e+01f	-1.999500e+01f
106	-6.635968e+01f	1.611540e+02f	-1.422391e+02f	-2.359958e+02f
107	-1.027115e+02f	6.335635e+01f	5.890118e+01f	-2.317287e+02f
108	-3.568023e+01f	-3.004601e+01f	-4.720082e+01f	-6.538943e+01f
109	-2.420128e+00f	-4.263517e+01f	9.142486e+01f	3.677592e+01f
110	2.252550e+01f	2.613908e+01f	2.377423e+00f	-1.089434e+02f
111	-7.869323e+01f	9.381803e+00f	2.965409e+01f	-4.895101e+01f
112	-7.763386e+00f	1.729950e+01f	-2.076704e+00f	-3.999288e+01f
113	-1.834761e+01f	-1.275474e+01f	3.761565e+01f	-2.555982e+01f

114	6.128716e+00f	-1.627836e+01f	8.417426e-01f	-1.574892e+02f
115	-2.691706e+00f	-7.347255e+00f	4.439051e+01f	-4.360759e+01f
116	-3.648065e+00f	-1.232642e+01f	1.357200e+01f	-2.670861e+01f
117	-4.811148e+00f	-5.413715e+01f	8.195444e+01f	-8.428111e+01f
118	4.136722e+01f	-6.868942e+01f	-2.783349e+01f	-4.582828e+01f
119	-1.465775e+01f	-1.421526e+01f	1.700677e+01f	-3.863719e+00f
120	3.953027e+01f	-7.618148e+00f	-9.368017e+01f	-4.748377e+01f
121	2.818595e+00f	1.737600e+00f	-7.094651e+00f	5.785191e+00f
122	5.421157e+01f	2.708056e+01f	-1.358517e+02f	-1.383537e+02f
123	-5.815253e+01f	3.657610e+01f	1.650299e+00f	-4.102374e+01f
124	2.294277e+01f	6.393948e+00f	-3.306164e+01f	-3.661114e+00f
125	2.204850e+01f	-2.616923e+00f	-1.899096e+00f	2.395342e+00f
126	1.809921e+01f	1.074701e+02f	-9.412917e+01f	3.141559e+01f
127	7.264172e+00f	1.246083e+01f	-8.145624e+00f	3.595879e+01f
128	-4.901291e+02f	-3.087985e+02f	-3.517769e+02f	1.048723e+02f
129	-1.033980e+02f	-9.325104e+01f	-1.052364e+02f	-5.960427e+01f
130	-1.751980e+02f	-6.563534e+01f	-1.028124e+02f	-1.294020e+02f
131	3.624958e+01f	-1.333738e+01f	1.169443e+02f	7.270349e+01f
132	-1.904479e+02f	-4.231232e+02f	-4.614496e+01f	-7.899152e+01f
133	-9.572653e+01f	-1.610434e+02f	-5.067430e+00f	8.905225e+01f
134	-9.578900e+01f	-1.386260e+02f	3.794329e+01f	1.771788e+01f
135	2.386354e+00f	2.843141e+00f	2.771185e+01f	4.281747e+01f
136	-5.200424e+01f	-1.219745e+02f	-1.202008e+02f	1.577073e+01f
137	-1.895511e+02f	1.576509e+02f	-1.173526e+02f	1.189042e+02f
138	-4.789680e+01f	4.342012e+01f	-6.776256e+01f	-1.131609e+02f
139	-7.712103e+01f	8.002439e+01f	4.012933e+01f	2.982221e+01f
140	-2.675811e+01f	-1.660393e+02f	-7.038818e+01f	4.560122e+01f
141	-1.433711e+02f	-8.409892e+00f	5.042733e+01f	1.984105e+01f
142	-6.167263e+01f	-4.350230e+01f	4.184977e+01f	-3.039065e+01f
143	-1.081915e+02f	8.749730e+01f	1.640191e+02f	-2.196589e+01f
144	-1.341916e+02f	-1.917522e+02f	-1.324093e+02f	2.541303e+01f
145	-8.551830e+01f	-3.651320e+01f	-3.613755e+01f	6.185310e+01f
146	-4.670377e+01f	-6.288307e+01f	-1.657188e+01f	-2.293704e+01f
147	2.772961e+01f	4.050206e+01f	5.079276e+01f	4.041062e+01f
148	3.146358e+01f	-1.874087e+02f	-8.208204e+01f	-2.928578e+01f
149	1.224387e+01f	-2.902534e+01f	-1.892225e+01f	1.812803e+01f
150	5.352742e+01f	1.244220e+01f	1.437704e+01f	9.475008e+01f
151	1.640437e+02f	8.009580e+01f	-5.553344e+01f	3.805836e+01f
152	-1.278948e+02f	-3.359999e+01f	-6.157497e+01f	2.465482e+01f
153	-6.306075e+01f	6.181715e-01f	-4.802540e+01f	1.146905e+02f
154	-1.394029e+01f	-8.081942e+00f	-9.246063e+00f	-2.001749e+01f
155	-4.473860e+01f	3.249331e+01f	-7.300407e+00f	-7.805927e+00f
156	-2.878723e+01f	-7.668082e+01f	-1.014817e+01f	3.043956e+01f
157	-3.234480e+01f	3.568585e-01f	-9.329520e+00f	3.517903e+00f
158	1.541468e+01f	6.101679e+00f	1.235321e+01f	9.334053e+00f
159	8.038906e+00f	1.600213e+01f	3.440883e+01f	1.270487e+01f
160	-1.750028e+02f	-1.510330e+02f	-1.521861e+01f	8.141479e+01f
161	-1.851013e+02f	7.911119e+01f	-3.646004e+01f	3.229775e+00f
162	-9.298784e+01f	2.874952e+00f	3.992886e-01f	-8.231313e+01f
163	-1.286975e+02f	3.162759e+00f	-9.445391e+00f	-4.998420e+01f
164	-3.100434e+01f	-1.294288e+02f	-2.223382e+01f	-8.245887e+00f
165	-8.093064e+01f	5.030370e+01f	2.946943e+01f	5.822795e+01f
166	9.476801e+01f	-2.803452e+01f	4.016521e+01f	1.992910e+01f
167	-9.013789e+00f	-1.098684e+00f	2.153182e+01f	1.567203e+01f
168	-1.580277e+02f	7.607675e+01f	1.739395e+01f	9.331100e+01f
169	-6.314642e+02f	4.512007e+02f	-1.256703e+02f	5.309097e+02f
170	-2.074674e+02f	1.750129e+02f	-1.211692e+02f	-1.612573e+02f
171	-3.300590e+02f	2.232878e+02f	1.833881e+01f	-8.801331e+01f
172	-7.154375e+01f	1.583570e+01f	1.375203e+02f	1.179287e+02f
173	-2.880376e+02f	1.315067e+02f	1.126987e+02f	2.679645e+02f
174	-8.105770e+01f	6.738242e+01f	2.013170e+00f	-1.418342e+01f
175	-1.239462e+02f	8.758101e+01f	3.717259e+01f	2.490352e+01f
176	-8.196072e+01f	-7.400150e+01f	-2.588659e+01f	-1.124186e+00f
177	-1.055565e+02f	-3.779158e+01f	-1.693435e+01f	7.904105e+00f
178	-4.118056e+01f	-1.171803e+01f	1.851546e+01f	5.403842e+00f
179	-1.044728e+02f	3.182006e+01f	-3.267234e+01f	2.439805e+01f
180	2.173420e+01f	-4.249213e+01f	-2.495996e+01f	-7.372384e+00f
181	-1.733268e+01f	1.447956e+00f	1.910183e+00f	2.832842e+01f
182	3.224969e+01f	-7.542976e+01f	-5.473665e+00f	-1.669638e+01f
183	2.273862e+01f	2.261948e+01f	-1.736998e+01f	1.491119e-01f

184	-1.398786e+02f	-6.762144e+01f	4.519868e+01f	3.888116e+01f
185	-2.557274e+02f	-4.002994e+01f	-5.963669e+01f	1.002754e+02f
186	-8.424302e+01f	1.949413e+01f	-2.604656e+01f	-3.653975e+01f
187	-9.074365e+01f	1.559456e+02f	-1.228115e+02f	-1.380371e+01f
188	-1.756622e+01f	4.662712e+01f	-1.750987e-01f	3.089990e+01f
189	-5.727877e+01f	6.691576e+01f	-2.523796e+00f	9.987911e+01f
190	-2.496159e+00f	1.337864e+01f	9.069791e+00f	-4.325504e+00f
191	-5.387091e+01f	1.828805e+01f	-4.369616e+01f	-6.117632e-01f
192	-1.220607e+02f	-2.790378e+01f	-1.188131e+02f	1.023907e+01f
193	-6.613209e+01f	-8.022907e+00f	-7.920473e+01f	-3.726114e+00f
194	-8.169948e+01f	8.963309e-02f	1.500424e+00f	-1.580413e+01f
195	3.091161e+01f	-1.111030e+01f	6.177293e+01f	3.707045e+00f
196	-6.765980e+01f	-1.475013e+02f	-1.228245e+02f	-1.025674e+02f
197	-8.022839e+01f	-2.758991e+01f	1.014192e+02f	-7.769528e+01f
198	3.582092e+01f	-3.257145e+01f	2.857075e+01f	5.305964e+01f
199	-6.291245e+00f	-8.471578e+00f	2.387695e+01f	-8.501721e+00f
200	-1.136961e+01f	1.668148e+01f	-1.449703e+02f	-8.888467e+01f
201	-8.491724e+01f	6.349139e+01f	-3.147471e+02f	3.810745e+01f
202	-2.511765e+01f	-1.309610e+01f	-2.388255e+01f	-2.931112e+01f
203	-7.560140e+01f	6.884189e+01f	-1.211752e+02f	-9.709624e+00f
204	-8.094465e+00f	-3.952005e+01f	-2.111503e+01f	-1.414320e+01f
205	-1.276885e+02f	3.300267e+00f	-9.889220e+01f	6.638647e+01f
206	3.947919e+00f	-3.529669e+00f	3.016468e+00f	9.377594e+00f
207	-5.346292e+01f	2.809430e+01f	2.783436e+01f	9.107659e+00f
208	-5.544914e+01f	-8.007348e+01f	-1.948053e+01f	3.957221e+01f
209	-2.054000e+01f	-9.548264e+00f	-2.208137e+01f	2.416983e+01f
210	-1.207221e+01f	-6.140471e+00f	2.168431e+00f	-1.682225e-02f
211	-1.374285e+01f	-1.716161e+01f	5.237946e+01f	1.536680e+01f
212	-1.670513e+01f	-5.166593e+01f	-3.310556e+01f	-5.853762e+01f
213	-6.452653e+01f	-1.802617e+01f	1.152278e+01f	-9.785408e+00f
214	3.946273e+01f	4.586205e+00f	6.733745e-01f	-1.105114e-01f
215	3.787770e+01f	1.958724e+01f	7.722115e+00f	4.177884e+00f
216	-4.507698e-01f	-9.149450e+00f	-2.796767e+01f	-2.169631e+01f
217	-1.923915e+01f	-4.181669e+00f	-1.514533e+02f	3.102444e+01f
218	1.451133e+01f	7.607017e+00f	-3.752773e+00f	5.052477e+00f
219	-8.679494e+00f	-4.509811e-01f	-3.299517e+01f	1.138781e+01f
220	2.326215e+01f	-1.991486e+01f	-3.363077e+01f	-3.042105e+01f
221	-3.021509e+00f	1.055051e+01f	-1.422632e+01f	8.219278e+00f
222	3.319254e+00f	1.180170e+00f	2.348810e+00f	2.477852e+00f
223	-4.658415e+00f	-4.865668e-01f	-7.492277e+00f	-1.816152e+00f
224	-4.824936e+01f	-4.249844e+01f	-3.251180e+01f	1.524245e+01f
225	-3.558498e+01f	1.467670e+01f	-1.583411e+00f	4.619221e+01f
226	7.103062e+01f	-9.319691e+01f	-1.602815e+00f	-4.684371e+01f
227	-1.461588e+01f	7.738719e+00f	2.945543e-02f	-1.504904e+01f
228	7.408744e+01f	-2.779816e+01f	1.368839e+01f	1.789202e+01f
229	2.106104e+01f	-7.421533e+01f	4.978619e+01f	-2.649717e+01f
230	2.687700e+02f	-1.276467e+02f	8.540343e+01f	1.405406e+02f
231	9.755909e+01f	2.486925e+01f	5.460705e+01f	-2.167683e+01f
232	-7.225999e+00f	3.967591e+01f	-5.098844e+01f	8.102224e+00f
233	-1.455736e+02f	1.158296e+02f	-7.325707e+01f	1.475374e+02f
234	-3.181041e+01f	4.144828e+01f	-5.391638e+01f	-4.755762e+01f
235	-1.343841e+02f	7.649548e+01f	-1.276844e+01f	-3.154128e+01f
236	-6.046935e+01f	-5.052032e+00f	1.766635e+01f	6.888904e+01f
237	-9.695324e+01f	-4.249137e+01f	3.004257e+00f	1.666785e+02f
238	6.988446e+01f	-2.306783e+00f	5.821824e+01f	6.690777e+01f
239	-2.239819e+01f	-1.575533e+01f	2.469466e+00f	3.772826e+01f
240	-2.557168e+01f	1.260773e-01f	1.229623e+01f	6.607617e+00f
241	-9.243539e+01f	-4.456417e+00f	4.117477e+01f	-4.966759e+00f
242	-6.936916e+00f	-4.487998e+01f	2.598589e+01f	-5.575389e+01f
243	-1.690167e+01f	6.997915e-01f	-1.083251e+01f	2.470072e+00f
244	6.200054e+00f	-1.347607e+01f	-5.221631e+00f	1.131096e+01f
245	5.409183e+00f	-2.165078e+00f	1.565600e+01f	-1.784852e+01f
246	5.225631e+01f	-1.022675e+02f	1.609757e+01f	3.140914e+01f
247	1.236804e+01f	-6.297240e+00f	5.098393e+00f	5.616476e+00f
248	-2.870804e+01f	-3.057013e+01f	1.078646e+01f	-6.054094e-01f
249	-9.252771e+01f	1.871356e+01f	-1.509309e+01f	5.678858e+01f
250	-3.483762e+01f	1.004655e+01f	-2.516129e+01f	-4.843069e+01f
251	-2.793391e+01f	3.554142e+01f	-4.033838e+01f	3.380660e+01f
252	6.848726e+00f	1.272950e+01f	2.042256e+00f	1.335002e+01f
253	-3.801345e+01f	-2.381242e+01f	-1.274181e+01f	3.064587e+01f

254	-3.006693e+01f	2.350847e+01f	-1.582671e+01f	-1.313860e+01f
255	-2.267754e+00f	6.131951e+00f	-2.540524e+00f	1.793340e+00f
256	-2.602792e+01f	-1.017641e+02f	-1.740470e+01f	-8.075488e+01f
257	1.409504e+01f	8.692205e+00f	4.688150e+01f	6.275372e+01f
258	3.565010e+01f	1.115568e+00f	2.252525e+01f	5.101844e+01f
259	-5.316019e+01f	-5.211581e+01f	1.607909e+02f	2.316667e+02f
260	-3.395222e+01f	-7.369822e+01f	1.058922e+01f	-1.157806e+01f
261	-1.864633e+01f	-3.656468e+01f	1.242685e+01f	2.172509e+01f
262	2.365805e+01f	-1.780664e+01f	-8.828355e-01f	2.055815e+01f
263	-3.891084e+01f	-2.904659e+01f	-6.920439e+00f	1.522751e+02f
264	-4.278527e+01f	-6.188411e+01f	-1.497046e+01f	-1.417290e+02f
265	2.827925e+01f	-3.149699e+01f	-1.497658e+01f	9.085215e+01f
266	-2.469408e+01f	-6.660832e+00f	4.655645e+00f	-3.247622e+01f
267	-7.566485e+01f	1.222958e+02f	5.261352e+01f	-1.356718e+01f
268	-4.130568e+01f	-2.131731e+01f	1.114909e+01f	-4.405708e+01f
269	4.796301e+01f	-3.529256e+01f	-2.480848e+01f	-2.856434e+00f
270	2.565193e+00f	8.882797e+00f	7.956729e+00f	5.368738e+00f
271	8.576624e+00f	3.643532e+01f	9.092934e+01f	2.965130e+01f
272	2.582210e+01f	-1.256673e+01f	-1.823667e+01f	4.481687e+01f
273	1.539836e+02f	-3.192010e+01f	-4.010284e+01f	2.054426e+02f
274	3.893365e+01f	-3.577870e+01f	8.854280e-01f	5.005868e+00f
275	2.229121e+00f	4.106913e+00f	3.694304e+01f	9.558260e+01f
276	2.182979e+01f	-5.753213e+01f	-1.346167e+01f	3.177657e+01f
277	6.229033e+01f	4.111803e+01f	-5.778986e+00f	4.262572e+01f
278	1.843930e+02f	-9.886469e+01f	1.036606e+01f	8.812613e+01f
279	8.737914e+01f	3.776595e+00f	-4.403338e+01f	1.485321e+01f
280	2.556648e+01f	-7.843745e+01f	-6.763878e+01f	5.986902e+01f
281	1.163002e+02f	-1.780007e+02f	-1.320695e+02f	2.702686e+02f
282	-2.275584e+01f	5.035579e+01f	-1.846773e+01f	4.890922e+00f
283	1.589596e+01f	-4.262065e+01f	-5.007692e+01f	8.718648e+01f
284	2.540721e+01f	-2.090868e+01f	-6.316955e+01f	1.194563e+01f
285	1.723335e+02f	-7.153433e+01f	-1.170531e+02f	8.860314e+01f
286	2.988006e+01f	-2.234478e+01f	-1.632739e+01f	1.862799e+01f
287	3.723506e+01f	1.113514e+01f	-2.079331e+01f	1.227221e+01f
288	1.642086e+01f	-7.863937e+01f	-6.661040e+01f	-3.107991e+01f
289	-6.279432e+00f	3.391553e+01f	2.471869e+01f	-1.672129e+01f
290	1.149549e+02f	-5.191133e+01f	-5.961166e+01f	-5.641650e+01f
291	-1.443805e+01f	2.844816e+01f	5.630069e+01f	6.115506e+01f
292	2.931646e+01f	-1.196476e+01f	1.524423e+01f	-4.050534e+00f
293	2.662226e+01f	-2.992776e+01f	3.713841e+01f	-1.652210e+01f
294	1.164263e+02f	-5.027757e+01f	1.393697e+02f	3.350730e+00f
295	1.853317e+01f	-9.099771e+00f	2.259193e+01f	1.854424e+01f
296	-2.441974e+01f	-3.413545e+02f	-2.050337e+02f	-1.559022e+02f
297	-2.253459e+00f	-2.786962e+01f	-3.149986e+01f	-1.009109e+02f
298	2.493520e+01f	-1.814977e+01f	-9.839252e+01f	-1.082730e+02f
299	3.076356e+01f	-2.089092e+01f	7.518166e+01f	-1.690462e+02f
300	-6.538629e+01f	-6.239236e+01f	-8.157248e+00f	-1.139168e+02f
301	5.210783e+00f	3.985172e+01f	2.115188e+01f	5.861368e+00f
302	3.182151e+01f	1.175033e+01f	1.777158e+01f	-2.726555e+01f
303	-1.201960e+01f	-6.799473e+00f	1.167974e+00f	-7.026115e+01f
304	1.686264e+01f	-1.257363e+01f	1.436683e+01f	-3.512646e+01f
305	4.213156e+01f	2.840898e+01f	1.361698e+01f	4.502953e+01f
306	4.572717e+00f	8.981042e+00f	-1.659353e+01f	-4.616721e+01f
307	-1.009113e+01f	-7.480584e+00f	5.273086e+00f	1.448309e+01f
308	3.193474e+00f	3.980138e-01f	1.457782e+01f	-2.083707e+00f
309	2.166581e+00f	-4.578209e+00f	-2.132811e+00f	-2.260280e+00f
310	1.004164e+01f	-5.295354e+01f	2.448281e+01f	-2.849890e+01f
311	1.152722e+00f	-8.438873e+00f	5.415828e+00f	2.716412e+00f
312	4.970164e+01f	-1.316138e+02f	1.756749e+01f	-8.540855e+01f
313	2.722350e-01f	-5.901179e+01f	-9.601443e+01f	1.098240e+02f
314	1.344854e+01f	-3.868587e+00f	-4.690299e+00f	-9.962342e+00f
315	1.212572e+01f	-9.070940e+00f	-1.480468e+01f	-2.427865e+01f
316	2.058902e+01f	4.272989e+01f	-3.921455e+00f	4.564357e+00f
317	5.316376e+01f	-5.492102e+01f	-4.884978e+01f	-3.253591e+01f
318	-8.838170e+00f	1.408442e+02f	-4.611798e+01f	2.721488e+01f
319	-5.391349e+00f	3.352518e+01f	-1.602914e+00f	-6.346127e+00f
320	5.610821e+01f	-3.042099e+00f	-1.461347e+01f	-3.384731e+01f
321	5.607257e+00f	1.015061e+01f	2.845134e+01f	-2.102522e+01f
322	2.299744e+02f	-5.662606e+01f	5.187247e+01f	-7.452612e+01f
323	-3.435854e+00f	5.164482e+01f	8.259259e+01f	-6.077316e+01f

324	1.808501e+00f	3.408920e+00f	3.053936e+00f	-2.009278e+01f
325	-2.850314e+01f	2.598798e+01f	9.711818e+01f	-6.086765e+01f
326	5.492112e+01f	-2.320849e-01f	-1.063772e+00f	-7.880479e+01f
327	-1.839178e+01f	-1.473681e+01f	-2.772500e+00f	1.498143e+01f
328	-8.826769e+00f	-1.957164e+01f	-2.259529e+01f	-4.043682e+01f
329	-4.329793e+01f	9.829152e+01f	7.881666e+00f	-6.349838e+01f
330	-6.408821e+00f	9.681027e+01f	8.274528e+00f	-7.370311e+01f
331	-2.497018e+01f	3.048679e+02f	-5.007944e+01f	-2.336102e+02f
332	6.444579e+01f	1.104570e+01f	-4.091908e+01f	6.494843e+00f
333	5.344052e+01f	3.253053e+01f	-1.098247e+02f	5.570633e+01f
334	7.177333e+00f	3.631745e+00f	-4.173658e+00f	-1.448554e+01f
335	-9.155081e+00f	5.778922e+01f	-4.387947e+01f	-6.785622e+01f
336	7.786528e+00f	-2.373410e+00f	-4.598256e+00f	-6.484489e-01f
337	4.670035e+01f	-4.360101e+00f	-6.945338e+00f	2.766690e+01f
338	6.024177e+01f	3.689769e+01f	2.324295e+01f	-4.617384e+01f
339	1.162078e+01f	1.639684e+01f	9.677499e+00f	-1.469520e+01f
340	4.644822e+01f	-1.056956e+01f	-9.587570e+01f	8.819092e+00f
341	1.109483e+01f	-6.207466e+00f	-7.740614e+01f	8.306777e+01f
342	2.752359e+01f	6.604212e+00f	2.738659e+00f	3.604771e+01f
343	-6.307344e+00f	-2.491698e+01f	-7.206456e+01f	-6.195260e-01f
344	7.888876e+01f	3.348632e+01f	-5.813511e+01f	-1.447907e+01f
345	7.065101e+01f	3.323522e+01f	-4.861838e+01f	1.397695e+02f
346	6.998343e+01f	2.117982e+02f	-1.470336e+00f	1.976722e+00f
347	-1.676264e+01f	1.062041e+02f	-7.770114e+01f	-9.975589e+01f
348	7.198685e+01f	5.570198e+01f	-2.156210e+02f	1.037994e+02f
349	1.523071e+02f	7.964714e+01f	-2.476784e+02f	1.978513e+02f
350	-4.930971e+01f	7.767579e+01f	-6.662063e+01f	1.879738e+01f
351	1.338881e+01f	-3.418016e+01f	-2.153308e+02f	5.824058e+01f
352	1.141565e+02f	6.489650e+01f	-1.915947e+01f	-2.913036e+01f
353	-1.896922e+01f	1.549328e+02f	1.252628e+02f	-2.804871e+01f
354	4.487964e+02f	1.064910e+02f	3.758545e+01f	-3.766768e+02f
355	4.768187e+01f	6.938548e+01f	-7.254021e+00f	-1.367574e+02f
356	2.960706e+01f	8.411428e-01f	3.908372e+01f	-5.494880e+01f
357	7.875262e+01f	-6.580510e+01f	1.666134e+02f	-6.763023e+01f
358	1.106083e+02f	-9.009404e+00f	5.661780e+01f	-4.226500e+01f
359	-2.057510e+01f	3.824987e+01f	4.077012e+01f	-3.667485e+01f
360	2.153434e+01f	-4.313694e+01f	-4.448294e+01f	-9.049099e+01f
361	6.460657e+01f	2.441247e+02f	1.665349e+02f	-1.328280e+02f
362	5.666193e+01f	-1.640425e+01f	3.337148e+01f	-1.712585e+02f
363	1.432708e+02f	2.210887e+02f	2.408583e+02f	-4.323946e+02f
364	4.203845e+00f	3.450055e+01f	-3.212078e+01f	-4.185099e+01f
365	1.819144e+00f	5.881392e+01f	2.030422e+01f	-4.604983e+01f
366	4.257468e+00f	4.861159e+01f	-8.396690e+01f	-8.466282e+01f
367	-1.466254e+02f	6.237473e+01f	7.631308e+01f	-1.811652e+02f
368	2.059127e+01f	2.240190e+00f	-1.714106e+01f	-4.121369e+01f
369	-4.866959e+00f	2.002646e+01f	2.267364e+01f	-3.285638e+01f
370	1.388026e+02f	1.034330e+01f	-1.165650e+02f	-8.161495e+01f
371	3.092070e+01f	-8.273679e+00f	-1.300411e+01f	-1.025407e+01f
372	-2.059791e+01f	1.333329e+01f	-4.546544e+01f	-1.594612e+01f
373	3.170298e+01f	-2.047462e+01f	1.323532e+01f	-3.691870e+01f
374	-4.405898e+00f	7.705063e+01f	-3.568735e+01f	5.888852e+01f
375	-7.339100e+00f	1.216151e+01f	-5.211148e+00f	-3.067947e+00f
376	9.749936e+00f	-1.569341e+01f	-5.633550e+01f	-4.050390e+01f
377	5.981884e+01f	6.188660e+01f	5.843682e+01f	-1.234486e+01f
378	-8.091006e+00f	3.886321e+01f	-6.008685e+01f	5.057122e+01f
379	1.078938e+01f	2.006543e+01f	8.287444e+01f	-1.069432e+02f
380	2.682469e+00f	8.168697e+01f	-1.975866e+02f	-9.879182e+00f
381	1.350290e+02f	7.750233e+01f	-3.703911e+01f	3.273562e+01f
382	-9.499564e+01f	1.856126e+02f	-2.468191e+02f	1.916900e+01f
383	-6.502234e+00f	5.233828e+01f	-5.430744e+01f	-2.201103e+01f
384	-5.861549e+01f	-8.976859e+01f	-2.001322e+02f	2.384332e+01f
385	-2.518359e+01f	-3.275378e+01f	-4.499232e+01f	8.904284e+00f
386	-4.993419e+01f	-2.259721e+01f	-3.575805e+01f	-1.361727e+01f
387	-3.056616e+01f	-2.637572e+01f	2.234506e+01f	6.181581e+01f
388	-9.063458e+01f	-1.956654e+02f	-1.099328e+01f	-8.607054e+00f
389	-3.399231e+01f	-2.874210e+01f	-2.046417e+01f	-1.029926e+01f
390	1.104077e+01f	-7.331593e+01f	3.974207e+01f	9.901445e+01f
391	1.060935e+02f	-7.273944e+01f	5.859950e+00f	-1.049359e-01f
392	-6.105939e+01f	-6.446678e+00f	-5.397833e+01f	-4.837263e+01f
393	-2.896922e+01f	8.427853e+01f	-8.831187e+00f	5.232608e+01f

394	3.971591e-03f	1.494663e+01f	-6.777686e+00f	-1.545365e+01f
395	-2.339570e+01f	9.232541e+01f	2.463146e+01f	1.841809e+01f
396	-4.437596e+01f	-3.865835e+01f	3.485452e+01f	4.046657e+00f
397	-4.255429e+01f	-1.276305e-02f	2.539373e+01f	-1.368413e+01f
398	1.472968e+01f	-1.112262e+01f	-8.682323e-01f	2.792291e+01f
399	-2.501247e+01f	1.279790e+01f	3.698242e+01f	-5.256403e+00f
400	-1.389474e+01f	-7.027615e+01f	-8.271126e+01f	4.317635e+01f
401	1.315270e+01f	-2.836778e+01f	-1.211120e+01f	6.106445e+01f
402	1.046332e+02f	3.174458e+01f	-3.594292e+01f	7.461868e+01f
403	8.430895e+01f	-2.794108e+01f	-7.651225e+00f	-1.864202e+01f
404	6.736282e+01f	-4.285775e+01f	-3.774602e+01f	5.162328e+01f
405	1.019541e+02f	-4.999983e+01f	-5.921380e+01f	9.522647e+01f
406	2.169423e+02f	-1.001284e+02f	-2.095526e+01f	2.605537e+02f
407	2.974816e+02f	-1.309525e+02f	-1.091436e+02f	7.046960e+01f
408	1.289371e+01f	2.608696e+01f	4.367154e+00f	5.600982e+01f
409	2.592863e+01f	4.939797e+01f	-6.204736e+01f	2.576559e+02f
410	2.171405e+01f	2.171992e+01f	2.042728e+01f	1.275903e+00f
411	-4.641140e+00f	1.832718e+01f	-1.895327e+01f	7.047016e+01f
412	3.734298e+01f	2.737974e+01f	-2.416939e+01f	2.992100e+01f
413	8.615932e+00f	2.518330e+01f	-2.986406e+01f	5.851981e+01f
414	1.478873e+02f	3.734939e+01f	4.309802e+01f	8.912696e+01f
415	1.009914e+02f	1.594883e+01f	-4.559251e+01f	-3.494229e+01f
416	-5.953759e+01f	-2.569496e+01f	-2.755647e+01f	4.977902e+01f
417	1.469401e+01f	3.511003e+01f	-6.909016e+01f	3.997887e+01f
418	6.646771e+00f	-2.701534e+01f	1.565337e+01f	-5.212066e+01f
419	-2.735971e+01f	1.937494e+01f	1.467782e+00f	-3.120099e+01f
420	-1.463965e+01f	-4.245603e+01f	4.982465e+00f	-2.171298e+01f
421	-1.896457e+01f	6.798770e+00f	3.297768e+01f	2.709388e+01f
422	2.719375e+01f	3.157749e+00f	1.763870e+01f	1.253050e+01f
423	1.118850e+01f	-2.211740e+01f	3.104160e+00f	-9.720916e+00f
424	6.742558e+00f	-5.690960e+01f	-4.344276e+01f	-4.544669e+01f
425	-3.909027e+00f	1.953661e+02f	-7.378949e+01f	1.799198e+02f
426	-6.931242e+01f	-2.052773e+01f	-1.034761e+02f	-5.509537e+01f
427	-7.227165e+01f	1.190814e+02f	-7.998154e+01f	-9.575904e+01f
428	-8.820415e+00f	3.344547e+01f	3.233073e+01f	3.039071e+01f
429	-1.180260e+02f	7.402422e+01f	1.111348e+02f	5.770362e+01f
430	-5.513233e+00f	5.671739e+00f	-1.519959e+01f	-1.701871e+01f
431	-1.765647e+01f	3.139987e+01f	1.269327e+01f	-1.071523e+01f
432	-3.625782e+00f	-1.983015e+01f	-4.719091e+00f	5.555747e+00f
433	-1.837827e+01f	-3.237542e+01f	-1.636524e+01f	3.138012e+01f
434	5.246282e+00f	-2.644405e+00f	1.131800e+01f	2.291970e+01f
435	-1.774247e-01f	-5.758407e-01f	-4.437962e+00f	1.616659e+01f
436	4.611825e+01f	6.613115e+01f	1.113547e+01f	1.678631e+01f
437	1.050041e+01f	2.575098e+00f	-3.000301e+00f	1.997723e+01f
438	5.698211e+01f	5.567982e+01f	4.031338e+01f	1.608200e+02f
439	8.200202e+01f	-7.576649e+01f	1.300961e+01f	3.377452e+01f
440	-7.568134e+00f	2.789545e+01f	1.420245e+01f	1.861786e+01f
441	-3.319102e+01f	1.010449e-01f	-3.828445e+01f	8.014675e+01f
442	-1.825926e+01f	-1.675169e+01f	-1.518025e+01f	-2.674865e-01f
443	-1.432635e+01f	3.396428e+01f	-2.030063e+01f	-7.664519e+00f
444	2.570308e+00f	2.029564e+02f	3.938926e+01f	1.003221e+02f
445	1.105423e+01f	5.759029e+01f	8.209068e+00f	3.323616e+01f
446	4.171705e+00f	5.092254e+01f	-1.601243e+01f	1.018476e+01f
447	2.331273e+01f	-4.661372e+00f	2.846769e+00f	-1.833670e+01f
448	1.060724e+00f	-6.488963e+01f	-4.054662e+01f	9.731126e+00f
449	-2.060492e+01f	2.944526e+01f	1.018842e+00f	3.852171e+00f
450	4.697958e+01f	-9.994888e+00f	1.940078e+01f	-1.659323e+01f
451	9.603244e+00f	1.003621e+02f	-2.962532e+01f	-1.035525e+01f
452	-9.048316e+01f	-3.564206e+01f	-2.110400e+01f	-4.354585e+01f
453	-3.339963e+01f	-1.701546e+01f	7.058804e+00f	-1.584693e+01f
454	2.101175e+01f	3.412477e+01f	-1.542087e+01f	2.996322e+01f
455	1.204226e+01f	1.533093e+01f	-7.238664e+00f	-6.091084e+00f
456	-7.699817e+00f	2.266672e+01f	-6.865942e+00f	1.339207e+01f
457	-8.309087e+01f	1.265453e+02f	-1.094725e+02f	6.410957e+01f
458	3.133123e+01f	6.610358e+01f	-3.404598e+01f	-4.977095e+01f
459	-2.556649e+01f	2.669519e+02f	-2.576074e+01f	-1.083193e+01f
460	3.727963e+01f	2.121676e+01f	-2.461484e+01f	-2.638809e+01f
461	-1.431086e+01f	1.416486e+01f	-2.453219e+01f	-1.991739e-01f
462	-2.621643e+01f	1.329469e+01f	4.881847e-01f	-8.542284e+00f
463	-4.516982e+01f	1.252521e+02f	-1.587382e+01f	4.382043e+01f

464	-8.698389e+00f	-6.482470e+00f	-1.180956e+01f	1.579420e+01f
465	-1.398431e+00f	-9.906530e-01f	-3.849259e-01f	4.480045e-01f
466	5.955465e+01f	2.292495e+01f	-7.089993e+00f	-1.064877e+01f
467	4.178613e+00f	4.072033e+00f	-1.910906e+01f	-1.085948e+01f
468	3.855243e+01f	-1.133252e+01f	-3.265862e+01f	-9.706882e+00f
469	1.585528e+01f	2.807898e+00f	-1.425390e+01f	1.022648e+01f
470	1.502367e+02f	1.805243e+01f	-1.001755e+02f	8.048560e+01f
471	7.737529e+01f	-6.012516e+01f	-4.406645e+01f	-7.221910e+01f
472	3.429519e+01f	-2.226274e+01f	-1.944472e+01f	-5.244543e+01f
473	-5.442449e+00f	1.115782e+01f	-7.519498e+01f	3.535616e+01f
474	9.245712e+01f	8.540282e+01f	1.953422e+01f	1.182165e+01f
475	-1.434468e+01f	8.619244e+01f	-1.097559e+01f	-1.749157e+01f
476	1.134411e+02f	2.085314e+00f	-1.695788e+02f	-8.259795e+01f
477	7.374146e+01f	9.699767e+01f	-1.132444e+02f	-2.603187e+00f
478	1.167885e+01f	4.790757e+00f	-3.979600e+01f	1.788536e+01f
479	-2.798210e+01f	-8.691462e+00f	-8.154750e+01f	-1.114957e+01f
480	1.331594e+01f	-4.245553e+01f	-8.831864e-01f	-1.051352e+00f
481	1.339613e+01f	9.960055e+00f	6.338765e+01f	-2.678702e+01f
482	1.578893e+02f	-5.787085e+01f	1.554902e+01f	-8.440359e+01f
483	2.898470e+01f	2.830502e+01f	1.021858e+01f	-4.373375e+01f
484	-7.422417e+00f	-8.613497e+00f	-1.147180e+01f	-9.072688e+00f
485	1.441960e+01f	-1.988312e+01f	4.842483e+01f	-2.498265e+01f
486	7.052316e+01f	1.535350e+01f	2.538188e+01f	1.059195e+01f
487	1.156391e+01f	5.096578e+00f	5.923004e+00f	-5.004314e+00f
488	9.787965e+00f	-9.973074e+00f	-1.994116e+01f	8.842867e-01f
489	-4.847986e+00f	7.091714e+01f	2.401624e+01f	2.446466e-01f
490	5.321981e+00f	2.886687e+01f	4.766403e+00f	-2.820832e+01f
491	-8.224569e+00f	1.803262e+02f	-2.631248e+01f	-1.048655e+02f
492	-3.430131e+00f	-6.175203e+00f	-3.247138e+00f	2.332392e+00f
493	-7.643304e+01f	-5.142905e+01f	2.782852e+01f	1.696459e+01f
494	-5.816037e+01f	1.068651e+01f	-3.343019e+00f	2.189435e+01f
495	-5.095869e+01f	3.798218e+01f	3.576060e+01f	-3.464448e+01f
496	-3.606019e+00f	1.551350e-01f	1.928955e-01f	6.108159e+00f
497	-1.192014e+01f	9.884865e+00f	1.365563e+01f	-6.151581e+00f
498	2.186580e+01f	1.218010e+01f	-8.050222e+00f	-1.252733e+01f
499	-8.477223e+00f	7.879566e-01f	1.828698e+00f	-6.044724e+00f
500	3.688790e+00f	8.066817e+00f	-1.003428e+00f	-1.028438e+00f
501	-1.279520e+00f	2.008733e+00f	3.367066e+00f	-2.322891e+00f
502	-7.806357e+00f	1.177796e+01f	-1.576581e+01f	3.738209e+01f
503	-3.973956e+00f	-2.096575e+01f	-9.842667e+00f	-4.446458e+00f
504	-1.367573e+00f	3.273621e+00f	-3.123500e+00f	-7.097918e+00f
505	-1.582913e+01f	9.579975e+00f	1.568060e+01f	1.072786e+01f
506	-3.495904e+01f	2.574827e+01f	8.499443e+00f	1.844386e+01f
507	-8.165817e+00f	3.882582e+01f	-1.577274e+01f	-3.508841e+01f
508	2.183497e+01f	4.884528e+01f	2.238374e+00f	-1.917012e+01f
509	9.629498e+00f	2.772281e+01f	-2.375984e+01f	-1.359516e+01f
510	-1.271007e+02f	1.028345e+02f	-1.212040e+02f	2.950373e+01f
511	-4.024986e+01f	1.184665e+01f	-2.486875e+01f	1.010818e+01f

/* SE_Shape6 */
/* dim=4x64 codewords */

index	codeword			
0	-7.715674e+01f	-1.001903e+02f	-8.446484e+01f	-7.506603e+01f
1	-3.248690e-02f	-5.372419e+01f	-3.526208e+01f	-3.732882e+02f
2	2.715981e+01f	-9.542826e+00f	-2.544565e+01f	-3.521582e+01f
3	-9.543247e+00f	-4.588637e+01f	-7.009737e+01f	-5.957597e+01f
4	-4.587965e+01f	6.254984e+01f	-3.433681e+00f	-3.053433e+00f
5	-4.909431e+01f	3.062190e+01f	9.696911e+01f	-9.727205e+01f
6	5.393697e+01f	4.862160e+01f	5.290791e+01f	-1.371891e+01f
7	-2.342088e+01f	1.257393e+01f	1.969771e+00f	-1.221188e+01f
8	-9.483003e+00f	-1.329470e+01f	-2.258810e+01f	-2.134610e+01f
9	-1.318463e+01f	6.910762e+00f	-3.348378e+01f	-9.452280e+01f
10	1.157537e+02f	-2.375080e+02f	-1.108919e+02f	-8.794429e+01f
11	-2.328881e+01f	-4.520446e+01f	-9.562418e+00f	-3.007779e+01f
12	9.080060e+00f	-3.644405e+00f	1.984743e+01f	-7.466264e+00f
13	-7.262711e+01f	-3.694774e+01f	5.443581e+01f	1.241948e+01f
14	8.167358e+01f	-7.590872e+01f	8.089388e+01f	2.270017e+00f

15	1.077513e+01f	6.706063e+00f	8.733265e+00f	6.792525e+00f
16	-5.354506e+00f	2.685910e+01f	-8.763814e+00f	9.523971e+00f
17	2.654242e+01f	-4.311794e+01f	3.930261e+01f	-7.635615e+01f
18	1.922930e+02f	-6.526617e+01f	-1.349328e+02f	9.321072e+01f
19	8.107768e+01f	6.627316e+01f	-4.454518e+01f	-3.550345e+00f
20	-9.555407e+01f	2.559018e+02f	4.243907e+01f	2.431019e+01f
21	-1.276582e+02f	1.204307e+02f	-2.241808e+01f	-2.996500e+01f
22	1.971344e+01f	3.726667e+01f	1.724132e+01f	1.132150e+01f
23	-7.205238e+00f	2.247461e+00f	8.574753e+00f	2.154328e+01f
24	3.120033e+01f	-7.375560e+01f	-4.165591e+01f	-2.019328e+01f
25	-1.635207e+01f	-7.580523e+00f	2.391256e+01f	-2.736161e+01f
26	4.151114e+02f	-1.662466e+02f	-3.494158e+02f	9.736285e+01f
27	9.470551e+01f	-3.539871e+01f	-7.809959e+00f	-4.028755e+01f
28	-2.561813e+00f	8.842011e+01f	1.626362e+00f	-3.058676e+01f
29	-6.618422e+01f	-2.964061e+00f	2.212437e+01f	-5.837995e+01f
30	5.654617e+01f	6.321455e+00f	-5.399224e+01f	2.948944e+01f
31	1.151316e+01f	-7.026150e+00f	-1.069256e+01f	-5.755084e+00f
32	1.760924e+01f	2.445458e+01f	-1.205589e+02f	-5.178760e+01f
33	5.221605e+01f	1.096689e+02f	-7.264160e+01f	-1.383182e+02f
34	-6.866194e-01f	5.295076e+00f	3.402568e+00f	-4.084459e+00f
35	4.420654e+01f	2.676651e+00f	1.901172e+01f	-3.742046e+01f
36	-1.167449e+01f	2.257275e+01f	3.639659e+01f	-5.827017e+00f
37	-1.217190e+02f	2.740261e+01f	-5.965681e+01f	6.886417e+01f
38	3.659688e+01f	-1.813419e+01f	1.254554e+02f	-3.755882e+01f
39	1.317821e+01f	4.288682e+00f	6.359964e+01f	4.423034e+01f
40	-4.201677e+01f	9.449666e+01f	-1.265763e+02f	-4.276317e+01f
41	-5.769096e+01f	-3.712167e+01f	-3.750058e+01f	-1.273010e+00f
42	1.748997e+01f	-2.979258e+01f	4.098346e+00f	-3.537130e+00f
43	-4.623678e+00f	-2.593299e+01f	5.345570e+01f	-1.912617e+00f
44	-3.909778e+01f	-1.839182e+01f	8.174182e+00f	4.595065e+01f
45	-2.645674e+02f	-1.695619e+02f	-1.117828e+02f	1.524306e+02f
46	6.563335e-01f	-1.422062e+00f	-7.233149e-01f	1.993203e+00f
47	9.140232e-01f	-1.100823e+02f	3.974484e+01f	1.519804e+02f
48	1.414145e+00f	4.164361e+01f	-2.357601e+01f	-4.852063e+01f
49	-3.243694e-02f	7.453350e+00f	-1.356703e+01f	-1.089271e+00f
50	6.046296e+01f	-2.331565e+01f	1.748271e+01f	4.001984e+01f
51	-2.765871e+01f	7.363817e+01f	1.113717e+02f	5.121661e+01f
52	2.222178e+01f	2.067046e+02f	-7.302510e+00f	-6.585908e+01f
53	-2.034584e+01f	5.322728e+01f	-3.061950e+01f	7.132930e+01f
54	3.412923e+01f	6.344651e+00f	-4.309694e+00f	2.467916e+01f
55	-3.386706e+01f	9.299767e+00f	3.486781e+01f	2.452423e+02f
56	-3.223902e+01f	1.782521e+01f	-5.735512e+01f	3.463223e+00f
57	-7.755984e+00f	-8.210009e+00f	1.674958e+00f	-4.798564e+00f
58	3.105626e+01f	-8.668492e+01f	-1.062224e+02f	1.378531e+02f
59	8.345006e+00f	1.381427e+01f	1.620355e+01f	7.401991e+01f
60	1.639190e+01f	1.749418e+01f	-8.716717e+00f	-1.631431e+01f
61	-3.533627e+01f	-1.048827e+02f	-3.684306e+00f	4.842657e+01f
62	-1.763505e+01f	-1.690411e+01f	-9.526247e+00f	1.201487e+01f
63	2.514880e+00f	-2.629004e+01f	-3.616854e+01f	5.187842e+01f

3. CbCelp

VQ codebook for stochastic excitation vector for 2kbps

/* 1st stage VXC gain codebook: cbL0_g[] */
/* dim=1 x 16 codewords */

index	codeword
0	1.63679657e+01
1	4.08766632e+01
2	1.91003250e+02
3	8.12259903e+01
4	8.14230530e+02
5	6.07095825e+02
6	3.76144836e+02
7	5.17214417e+02

8	4.69211279e+03
9	2.85045361e+03
10	1.78430249e+03
11	2.00917432e+03
12	1.03291040e+03
13	1.17178540e+03
14	1.57345483e+03
15	1.43213721e+03

/* 1st stage VXC shape codebook: cbL0_s[][] */
/* dim=80 x 64 codewords */

index	codeword				
0	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-7.446723e-02	0.000000e+00	8.365795e-02	0.000000e+00
	0.000000e+00	0.000000e+00	1.263896e-01	0.000000e+00	0.000000e+00
	0.000000e+00	1.628212e-01	0.000000e+00	3.162742e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.559331e-01	0.000000e+00
	0.000000e+00	0.000000e+00	4.265333e-01	-2.755800e-03	-6.673675e-02
	8.668371e-02	0.000000e+00	3.223823e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	1.327417e-01	-9.680346e-02	5.832724e-02	8.642182e-02
	0.000000e+00	0.000000e+00	-1.399930e-01	0.000000e+00	7.616382e-02
	-1.229509e-01	-9.438528e-03	1.069120e-01	2.070712e-01	0.000000e+00
	2.063423e-02	7.181489e-03	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	3.469223e-01	0.000000e+00	-2.492027e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-3.091645e-02
	-1.531175e-01	0.000000e+00	4.361209e-03	0.000000e+00	8.736080e-03
	-3.212416e-01	0.000000e+00	0.000000e+00	2.817580e-01	0.000000e+00
1	0.000000e+00	0.000000e+00	1.034455e-01	0.000000e+00	0.000000e+00
	0.000000e+00	7.150955e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.278887e-02
	0.000000e+00	3.145205e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-8.912478e-02	0.000000e+00	-4.173242e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	2.946955e-01	0.000000e+00	-1.507957e-01	0.000000e+00
	0.000000e+00	0.000000e+00	-2.664525e-01	0.000000e+00	3.264460e-01
	3.038466e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	2.929597e-01	0.000000e+00	-2.387601e-01
	5.487535e-02	0.000000e+00	0.000000e+00	0.000000e+00	2.071751e-01
	0.000000e+00	0.000000e+00	5.576105e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-1.297768e-01	0.000000e+00	1.807242e-01	5.194358e-01
2	0.000000e+00	-3.431119e-02	0.000000e+00	-1.030924e-01	1.638147e-02
	-8.418150e-02	1.003403e-01	1.787905e-01	-1.453511e-01	7.306241e-02
	0.000000e+00	3.961568e-02	-7.716341e-02	-1.080922e-01	7.019868e-02
	0.000000e+00	0.000000e+00	-1.400476e-01	9.123196e-03	-2.019143e-01
	3.612782e-01	-1.386515e-02	1.297576e-01	8.974585e-02	1.122697e-01
	-2.887599e-02	3.220217e-02	-1.979366e-02	2.592422e-01	6.677318e-02
	1.928561e-01	1.195134e-01	7.011453e-02	0.000000e+00	0.000000e+00
	-1.029146e-02	1.313047e-01	0.000000e+00	-2.092986e-01	-1.249356e-01
	2.170474e-02	3.296610e-03	0.000000e+00	-5.190199e-02	2.513182e-01
	4.727957e-02	0.000000e+00	4.197563e-02	-5.300336e-02	5.749315e-02
	-5.546883e-02	0.000000e+00	-3.709866e-02	-2.156807e-01	0.000000e+00
	0.000000e+00	2.180739e-01	2.365173e-01	1.272576e-01	0.000000e+00
	0.000000e+00	0.000000e+00	1.501998e-01	1.218029e-01	5.407676e-02
	-1.081901e-01	-3.947741e-02	0.000000e+00	0.000000e+00	-1.326761e-01
	0.000000e+00	2.329781e-02	0.000000e+00	-1.425056e-01	0.000000e+00
	-1.324701e-01	-2.016195e-01	-1.284797e-01	0.000000e+00	-1.964498e-01
3	0.000000e+00	0.000000e+00	3.275302e-02	-7.426704e-02	-2.077776e-01
	6.211038e-02	0.000000e+00	-1.198653e-01	0.000000e+00	-7.663831e-02
	-6.830758e-02	9.526969e-02	1.432645e-01	-3.714148e-01	0.000000e+00
	0.000000e+00	-4.017135e-02	0.000000e+00	1.684766e-02	0.000000e+00
	-5.346030e-02	8.102176e-02	2.116191e-02	-2.892020e-02	1.712772e-01
	1.179079e-01	-5.978651e-03	6.032893e-02	0.000000e+00	9.755966e-02
	-6.924803e-02	2.367248e-02	-6.733087e-02	1.846650e-02	-2.750713e-02
	-6.094126e-02	-4.257798e-02	1.530618e-02	-2.906141e-01	8.083130e-02
	1.203273e-01	-6.632135e-02	-1.975852e-02	9.922199e-03	-9.718367e-02
	-4.624523e-02	6.067814e-03	-3.409956e-01	-4.223726e-02	-2.033218e-01
	-4.826438e-02	0.000000e+00	0.000000e+00	-9.283136e-02	1.301494e-01
	1.326201e-01	2.268664e-01	1.526320e-01	-2.409160e-02	5.123970e-02

	4.177166e-02	1.999217e-02	-1.842601e-01	6.696353e-02	-5.069190e-02
	0.000000e+00	1.550389e-01	1.044889e-01	-5.895543e-02	2.980536e-03
	-8.966246e-02	3.611934e-01	1.961482e-01	0.000000e+00	0.000000e+00
	2.800390e-02	0.000000e+00	-3.606865e-02	0.000000e+00	-5.167208e-02
	-6.851541e-02	-3.233859e-02	4.064562e-03	0.000000e+00	-5.817794e-02
	4.623980e-02	-1.893821e-01	0.000000e+00	1.161268e-01	2.189767e-01
	8.413111e-02	-1.850753e-02	1.340682e-02	0.000000e+00	1.241090e-01
	0.000000e+00	1.777913e-01	8.692873e-02	0.000000e+00	-6.321253e-02
	-1.061696e-02	-2.948175e-01	-9.250643e-04	-2.070189e-01	-2.299177e-01
	-1.309335e-03	0.000000e+00	-6.099151e-03	-9.783129e-02	4.102601e-02
	1.039670e-01	0.000000e+00	5.899777e-02	-8.974618e-03	-1.185342e-01
4	-2.047010e-01	-4.924064e-02	1.538036e-02	0.000000e+00	-2.208408e-01
	-7.275250e-03	2.597088e-02	-3.559601e-02	7.058578e-02	-1.740188e-02
	1.992517e-01	0.000000e+00	-2.173503e-01	5.231891e-02	7.031193e-03
	7.973570e-02	1.742104e-01	1.394742e-01	7.130611e-02	3.097296e-01
	1.248135e-02	1.562396e-01	1.066017e-01	8.316639e-02	8.664023e-02
	1.392538e-01	-1.501391e-01	5.376713e-02	-4.729616e-02	8.132518e-03
	-6.807949e-02	0.000000e+00	0.000000e+00	0.000000e+00	1.016306e-01
	1.173370e-01	9.033468e-02	-1.578965e-01	0.000000e+00	-1.779995e-01
	2.399448e-01	6.592661e-02	0.000000e+00	6.199083e-03	-4.819654e-02
	0.000000e+00	4.072129e-02	2.351524e-01	1.326076e-01	1.229718e-01
	2.676435e-02	-1.119737e-01	3.838634e-02	0.000000e+00	9.558739e-02
	-1.885883e-02	0.000000e+00	-2.448671e-01	0.000000e+00	-1.915690e-01
	4.922582e-02	1.884156e-01	-4.166127e-02	-1.206685e-01	8.930360e-02
	-2.868485e-02	5.374127e-02	2.237625e-01	0.000000e+00	-2.765942e-01
	-2.369016e-01	1.777102e-02	1.507170e-02	-7.786537e-02	2.768026e-02
	-4.007258e-02	-4.676265e-02	8.270830e-02	1.375236e-01	3.351047e-02
5	-1.905837e-01	-6.958213e-02	-8.811536e-02	8.150120e-02	-1.122283e-01
	0.000000e+00	2.729283e-01	-1.139816e-01	1.086328e-01	0.000000e+00
	-1.072020e-01	-1.246174e-01	-3.395832e-01	1.685047e-02	-1.168743e-01
	5.961511e-02	8.628888e-02	0.000000e+00	-1.440817e-01	-6.333286e-02
	2.573951e-02	3.175903e-02	0.000000e+00	0.000000e+00	0.000000e+00
	1.336364e-01	-1.100685e-01	-1.718847e-01	0.000000e+00	1.058596e-01
	-9.086831e-03	-6.042259e-02	-6.605418e-02	5.789364e-02	-1.864227e-01
	-1.935646e-02	-9.367181e-03	0.000000e+00	1.101601e-01	0.000000e+00
	4.881159e-02	-6.971891e-02	-2.261834e-02	-6.555185e-04	6.606196e-02
	1.144709e-01	2.859802e-01	2.479818e-01	-3.528697e-01	-5.462653e-02
	0.000000e+00	-4.622316e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	8.668036e-02	-8.784793e-02
	0.000000e+00	-9.544450e-02	-3.009367e-01	-1.616730e-01	0.000000e+00
	-1.516886e-01	-5.298662e-01	0.000000e+00	-1.999967e-01	0.000000e+00
	0.000000e+00	-8.968464e-02	-5.654503e-02	0.000000e+00	0.000000e+00
	-6.374828e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
6	-5.947820e-02	0.000000e+00	5.765983e-02	0.000000e+00	0.000000e+00
	-1.535626e-02	-4.012959e-01	6.120697e-02	0.000000e+00	0.000000e+00
	1.403747e-02	0.000000e+00	0.000000e+00	-1.847287e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-5.791732e-02	0.000000e+00	0.000000e+00	7.174537e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-2.978670e-02	0.000000e+00	0.000000e+00	-3.242782e-02
	0.000000e+00	1.724256e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-3.570322e-02
	9.525254e-02	1.714142e-01	7.894193e-02	5.404997e-03	-3.337107e-01
	0.000000e+00	9.249155e-02	-1.006343e-01	-6.669622e-02	5.757630e-02
	1.433624e-01	-4.716781e-02	1.627346e-01	0.000000e+00	0.000000e+00
	-5.106444e-02	0.000000e+00	0.000000e+00	1.490715e-01	1.127041e-01
	-8.394412e-02	0.000000e+00	-1.390087e-01	-1.589523e-01	0.000000e+00
	-1.600331e-02	1.419130e-01	3.547226e-02	1.410345e-01	-2.307342e-02
	-5.574644e-03	0.000000e+00	-1.061710e-01	-1.638792e-01	-1.521554e-02
7	-3.723726e-01	0.000000e+00	0.000000e+00	0.000000e+00	7.145053e-02
	-1.616386e-01	0.000000e+00	1.070670e-01	0.000000e+00	0.000000e+00
	-5.652595e-02	2.644849e-01	-1.707063e-01	-3.475726e-02	-8.437679e-03
	-8.078579e-02	1.238485e-01	5.397616e-02	-3.922509e-02	0.000000e+00
	-1.281174e-01	0.000000e+00	0.000000e+00	0.000000e+00	3.091413e-01
	-1.246579e-01	-1.082035e-01	-2.643339e-01	0.000000e+00	1.080647e-01
	0.000000e+00	0.000000e+00	0.000000e+00	1.730555e-02	-6.952362e-02
	-5.086229e-02	-4.057032e-02	0.000000e+00	5.099443e-02	-1.424230e-01
	-9.409947e-02	-1.469928e-01	1.336108e-02	-5.422432e-02	-2.694872e-02
	0.000000e+00	-1.130124e-01	1.259848e-01	2.064474e-02	0.000000e+00
	-2.668635e-02	1.428174e-01	-1.467296e-01	6.034980e-02	-2.733297e-01
	3.675922e-02	2.491294e-01	0.000000e+00	0.000000e+00	-1.423630e-02
	-1.100824e-01	2.214209e-01	0.000000e+00	0.000000e+00	-1.882147e-02
	0.000000e+00	0.000000e+00	6.979715e-02	0.000000e+00	4.597319e-02
	-5.317160e-02	5.855310e-02	0.000000e+00	3.008893e-02	1.636985e-01
	-1.058337e-01	0.000000e+00	1.505763e-02	1.465906e-01	0.000000e+00

8	3.135809e-02	1.677612e-02	8.379304e-02	0.000000e+00	0.000000e+00
	3.526599e-03	1.996471e-01	0.000000e+00	1.463492e-02	5.582351e-02
	-1.047543e-01	3.049806e-02	-3.253802e-01	-2.486716e-01	-3.005802e-02
	-9.420968e-02	1.268605e-01	-9.437799e-03	9.871085e-02	-3.122336e-01
	1.635047e-01	-1.252551e-01	-9.630246e-02	0.000000e+00	1.367348e-01
	-6.722090e-02	0.000000e+00	1.531722e-01	0.000000e+00	2.123449e-01
	0.000000e+00	0.000000e+00	-1.616551e-01	0.000000e+00	-1.699343e-02
	-1.339488e-01	0.000000e+00	1.817555e-01	1.901571e-02	-1.109425e-01
	-1.201857e-01	1.050510e-02	-6.851291e-02	1.741884e-01	5.951470e-02
9	-5.204961e-02	1.105765e-01	-2.188344e-01	1.588331e-01	-2.188095e-02
	2.500322e-02	0.000000e+00	-1.647620e-01	-1.047136e-01	-3.047016e-01
	0.000000e+00	-2.730675e-02	-2.478165e-02	7.318392e-02	0.000000e+00
	-7.017808e-03	-2.096990e-01	0.000000e+00	1.771540e-01	0.000000e+00
	0.000000e+00	-3.240453e-02	1.995394e-01	7.789627e-02	-9.488788e-02
	-1.558849e-02	-2.529439e-01	3.691096e-02	1.247959e-01	9.687182e-02
	5.006663e-02	0.000000e+00	-4.195499e-02	2.106432e-01	4.513912e-02
	1.477155e-01	3.950630e-03	-1.110148e-01	0.000000e+00	-1.824765e-01
	0.000000e+00	0.000000e+00	6.278210e-02	5.043171e-02	1.082794e-01
	-1.377237e-01	5.041304e-02	0.000000e+00	-1.686943e-01	0.000000e+00
	0.000000e+00	6.781540e-02	-9.709130e-03	4.035054e-02	1.415920e-01
	2.568376e-01	-4.846834e-02	0.000000e+00	1.679232e-01	-2.054341e-01
	0.000000e+00	0.000000e+00	9.028047e-02	-9.281924e-02	0.000000e+00
	1.070664e-01	2.855498e-01	6.904217e-02	1.433496e-01	1.450918e-01
	1.835904e-02	1.389033e-02	7.927611e-02	0.000000e+00	5.040524e-02
	1.575074e-01	-4.381688e-02	0.000000e+00	3.517214e-02	-8.021181e-02
10	-1.184579e-01	1.401820e-01	0.000000e+00	-8.655734e-02	2.511532e-01
	-8.753932e-02	-1.170508e-01	-4.103570e-02	-1.390736e-02	-4.976107e-03
	1.433563e-01	4.847837e-02	5.022078e-02	1.826874e-01	5.367411e-02
	1.146083e-01	-6.150220e-03	6.433681e-02	0.000000e+00	4.553910e-02
	-5.938482e-02	-1.658500e-01	8.084041e-02	2.477319e-01	-5.047380e-02
	3.542646e-02	0.000000e+00	1.147855e-01	1.235729e-01	1.692290e-01
	-1.914313e-01	1.353696e-01	-2.313394e-02	-1.107694e-02	1.116495e-02
	5.985822e-02	0.000000e+00	1.540551e-01	6.066566e-02	2.275869e-02
	9.607898e-02	-1.117417e-02	9.613124e-02	2.281128e-01	1.702338e-01
	2.783363e-02	2.233101e-01	0.000000e+00	-4.691504e-02	6.188341e-03
	1.270348e-01	-1.250601e-01	3.099766e-02	-4.954688e-02	7.404431e-02
	-1.698069e-01	-9.051166e-02	0.000000e+00	0.000000e+00	2.440053e-01
	0.000000e+00	-9.645480e-02	2.398406e-02	0.000000e+00	-1.699868e-01
	3.639935e-02	1.181297e-01	0.000000e+00	-1.859562e-01	0.000000e+00
	-2.976804e-01	0.000000e+00	1.392341e-01	-2.330781e-01	0.000000e+00
	0.000000e+00	-5.866628e-02	0.000000e+00	0.000000e+00	1.364874e-02
11	0.000000e+00	-8.846001e-02	-4.184854e-02	-3.031504e-01	2.229595e-01
	-1.060365e-01	-9.141333e-02	-8.926305e-03	6.040913e-02	0.000000e+00
	0.000000e+00	4.821959e-02	0.000000e+00	0.000000e+00	-2.396788e-01
	-8.451044e-02	-1.831814e-01	0.000000e+00	-7.316098e-02	4.718650e-02
	-1.153773e-01	7.463606e-02	-1.065313e-01	2.400000e-01	9.668455e-03
	9.353368e-02	0.000000e+00	-1.115068e-01	-1.269049e-01	1.510192e-02
	0.000000e+00	-2.967163e-02	-5.458750e-02	-1.824451e-01	-8.701614e-02
	0.000000e+00	1.875530e-02	0.000000e+00	-1.955204e-01	9.239270e-02
	-6.860760e-02	0.000000e+00	-4.121834e-02	0.000000e+00	0.000000e+00
	-2.426863e-01	-1.378449e-02	-5.167505e-02	1.044290e-01	-8.337912e-02
	0.000000e+00	0.000000e+00	-2.596197e-01	-9.034151e-02	8.773266e-02
	3.310943e-01	-9.083990e-02	1.012295e-01	1.082746e-01	0.000000e+00
	2.942288e-02	1.144124e-01	5.067735e-02	1.996776e-02	-2.169889e-02
	0.000000e+00	-1.240306e-01	3.790487e-03	0.000000e+00	-9.548061e-02
	-4.977221e-02	2.115396e-01	-5.795059e-02	0.000000e+00	2.814568e-01
	4.715882e-02	-8.318082e-02	0.000000e+00	4.452052e-02	-9.054305e-02
12	5.226738e-02	0.000000e+00	-2.249037e-02	9.563882e-02	-5.413606e-02
	-1.527019e-01	0.000000e+00	0.000000e+00	4.495829e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	1.131425e-01	2.251066e-01
	0.000000e+00	-1.600841e-01	-4.750357e-02	6.946425e-02	4.601294e-02
	1.200875e-01	-1.549908e-01	2.435023e-01	-2.979402e-02	-1.110556e-01
	2.945571e-01	5.249560e-02	0.000000e+00	-9.806531e-02	3.094085e-02
	-1.140711e-01	0.000000e+00	-3.622640e-02	1.585257e-01	-1.986223e-02
	1.951823e-01	4.790216e-02	-5.336912e-02	-1.081680e-01	6.258853e-03
	9.125248e-02	-1.290061e-01	1.381504e-01	1.916832e-01	0.000000e+00
	5.363149e-02	1.610838e-02	-1.321937e-01	-1.656941e-01	-6.689768e-02
	1.010989e-02	-3.498878e-02	4.359725e-02	0.000000e+00	-7.006236e-02
	0.000000e+00	2.794867e-02	0.000000e+00	-1.425188e-02	-1.421728e-02
	-1.840855e-01	6.270842e-02	0.000000e+00	5.461627e-02	0.000000e+00
	-3.153941e-02	1.476120e-01	3.594700e-01	-1.236055e-01	1.581428e-01
	0.000000e+00	6.789200e-02	-2.385516e-01	0.000000e+00	-2.124092e-01
	0.000000e+00	-2.341898e-01	6.973508e-02	-8.445712e-02	-7.705665e-02
	0.000000e+00	3.052458e-02	0.000000e+00	0.000000e+00	0.000000e+00
	1.255758e-01	-1.253119e-02	0.000000e+00	0.000000e+00	0.000000e+00

	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.444075e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-1.684990e-02	-5.203775e-02	0.000000e+00	0.000000e+00	0.000000e+00
	7.606973e-02	1.503713e-01	-3.399794e-02	4.019326e-02	0.000000e+00
	2.156103e-01	0.000000e+00	0.000000e+00	0.000000e+00	1.465064e-01
13	0.000000e+00	0.000000e+00	-1.666505e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	6.422910e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.960863e-01
	0.000000e+00	0.000000e+00	0.000000e+00	3.079987e-02	0.000000e+00
	0.000000e+00	-1.321795e-01	0.000000e+00	0.000000e+00	1.377151e-01
	0.000000e+00	0.000000e+00	1.829775e-01	0.000000e+00	0.000000e+00
	-2.328607e-02	0.000000e+00	1.829032e-01	-3.940682e-02	0.000000e+00
	-2.142807e-01	2.501262e-01	0.000000e+00	1.618111e-01	0.000000e+00
	-1.258520e-01	-7.787670e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-3.460924e-02	0.000000e+00	1.073027e-01
	0.000000e+00	-1.255805e-01	1.398782e-01	3.199930e-02	-2.230761e-02
	0.000000e+00	0.000000e+00	1.196708e-02	-1.646960e-01	1.003426e-01
	0.000000e+00	1.392960e-01	-3.493728e-01	0.000000e+00	3.705410e-02
	0.000000e+00	-7.485031e-02	-2.644881e-02	0.000000e+00	6.547042e-02
	-3.106349e-02	0.000000e+00	3.977729e-02	0.000000e+00	3.676563e-02
	-5.913478e-02	-5.275781e-02	1.130835e-02	0.000000e+00	0.000000e+00
14	6.054890e-02	8.096541e-03	1.430169e-02	-1.399695e-01	-1.124085e-01
	-1.248292e-01	0.000000e+00	-2.398101e-02	1.757598e-01	-1.066129e-01
	2.338520e-01	-2.090404e-02	6.138080e-02	-4.838313e-02	0.000000e+00
	1.589346e-01	1.430519e-01	7.092372e-02	-7.580565e-02	-2.998621e-02
	0.000000e+00	-4.291384e-03	-2.903766e-01	-8.532753e-02	1.388037e-01
	1.037748e-01	-1.370212e-01	6.838220e-02	-5.292721e-03	9.792621e-03
	1.730431e-01	-5.377327e-03	-1.471375e-02	3.054197e-01	-3.966831e-02
	1.122662e-01	1.734881e-01	2.089176e-01	1.918045e-01	-2.239722e-01
	-1.842949e-03	1.479437e-01	-3.514411e-02	5.747592e-02	-2.571907e-01
	4.244782e-03	0.000000e+00	-1.087159e-01	-1.080279e-03	0.000000e+00
	-9.503651e-02	1.152865e-01	4.316902e-02	-3.571646e-02	-2.054230e-02
	1.046008e-01	-1.462350e-01	-1.149189e-01	-8.611986e-03	0.000000e+00
	-2.144747e-01	-3.526621e-03	0.000000e+00	0.000000e+00	2.856612e-02
	9.581376e-04	0.000000e+00	9.642459e-02	-9.059028e-03	-1.314269e-01
	9.437560e-03	2.108056e-01	-3.144892e-01	-2.579096e-01	1.687101e-02
	0.000000e+00	-9.140352e-02	-1.073301e-02	0.000000e+00	1.234619e-01
15	0.000000e+00	-3.159320e-02	0.000000e+00	3.145353e-02	0.000000e+00
	0.000000e+00	0.000000e+00	-5.775847e-02	1.648641e-01	1.778964e-02
	-2.258416e-01	2.856773e-01	-2.572417e-02	-4.742481e-02	-2.561272e-01
	-5.396117e-03	0.000000e+00	2.450048e-01	0.000000e+00	-7.927544e-02
	-1.338785e-01	0.000000e+00	-4.158016e-02	-4.921980e-02	0.000000e+00
	2.256816e-03	1.062408e-01	1.389951e-01	9.768847e-02	1.275418e-01
	8.814584e-02	0.000000e+00	1.618956e-02	0.000000e+00	-1.352719e-01
	-7.685195e-02	6.388885e-02	7.738536e-02	2.753483e-01	-1.710922e-01
	1.736569e-01	-1.907217e-01	-4.377383e-02	1.166700e-01	-3.164948e-02
	-4.449867e-02	9.055889e-02	1.625533e-02	1.202041e-01	-1.989173e-01
	-5.145200e-03	-6.122393e-02	1.806225e-02	1.121715e-01	1.091225e-01
	7.871468e-03	-1.125362e-01	2.421628e-01	9.648913e-03	-7.914937e-02
	1.128704e-02	-2.129306e-01	8.113680e-02	0.000000e+00	-4.533440e-02
	-9.125906e-02	0.000000e+00	-6.747673e-02	0.000000e+00	0.000000e+00
	-6.053391e-02	0.000000e+00	-1.408204e-01	-5.927842e-03	7.438395e-02
	9.761242e-02	1.494223e-01	-6.023963e-02	0.000000e+00	-4.428609e-02
16	-1.111302e-01	3.324184e-04	8.027700e-02	-1.152890e-01	0.000000e+00
	-1.450241e-01	-1.362006e-01	0.000000e+00	3.360389e-01	0.000000e+00
	3.337385e-02	4.269869e-02	-1.447241e-02	0.000000e+00	1.055005e-01
	-1.206725e-01	3.755829e-02	-2.126236e-01	1.829056e-01	0.000000e+00
	0.000000e+00	4.119036e-01	8.387204e-02	-2.149649e-01	-2.185033e-01
	0.000000e+00	1.291150e-01	2.180859e-01	0.000000e+00	-1.299124e-01
	-1.862020e-02	-1.725418e-02	3.961763e-02	-7.739465e-02	-3.897086e-02
	0.000000e+00	6.661487e-02	-4.743303e-02	3.596134e-02	-1.549692e-01
	-9.140582e-02	1.964418e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	2.547679e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	1.122786e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-1.984257e-01	-1.067540e-01	-7.514788e-02
	0.000000e+00	0.000000e+00	-2.604505e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-1.301781e-01	4.037084e-01	0.000000e+00	0.000000e+00	1.766727e-01
17	0.000000e+00	0.000000e+00	0.000000e+00	1.709537e-03	0.000000e+00
	0.000000e+00	-1.318666e-02	3.546191e-02	0.000000e+00	0.000000e+00
	9.488191e-02	0.000000e+00	0.000000e+00	3.112417e-02	1.888284e-01
	0.000000e+00	1.582230e-01	-4.498250e-01	3.104823e-02	1.134750e-01
	0.000000e+00	1.199584e-01	3.733751e-01	0.000000e+00	0.000000e+00
	1.233830e-01	1.243223e-01	0.000000e+00	-8.233973e-02	0.000000e+00

	0.000000e+00	0.000000e+00	1.417152e-01	0.000000e+00	-2.623514e-01
	0.000000e+00	1.688116e-01	-1.676710e-01	0.000000e+00	1.343091e-02
	0.000000e+00	0.000000e+00	0.000000e+00	-9.082860e-02	-1.440900e-01
	-1.558381e-01	0.000000e+00	0.000000e+00	1.694985e-01	0.000000e+00
	-2.198474e-01	0.000000e+00	0.000000e+00	-9.483727e-02	3.344159e-01
	8.631221e-02	-8.744366e-02	-1.450577e-01	-9.487721e-02	-1.706874e-01
	-9.788779e-02	-5.133312e-02	-3.348986e-02	-1.319246e-01	0.000000e+00
	0.000000e+00	0.000000e+00	1.302543e-01	0.000000e+00	-6.526400e-02
	3.311100e-02	0.000000e+00	-1.297837e-01	1.043136e-01	8.844133e-02
	0.000000e+00	8.166263e-02	-1.731240e-02	-1.463674e-01	5.440976e-03
18	0.000000e+00	4.301036e-02	0.000000e+00	1.841493e-01	1.113537e-03
	0.000000e+00	-1.080874e-02	2.126244e-01	-8.532459e-02	2.739208e-01
	2.162094e-01	0.000000e+00	9.268834e-02	8.659123e-02	0.000000e+00
	1.363128e-01	8.272618e-02	-1.616666e-01	-3.496521e-02	-1.916176e-01
	-8.637311e-02	1.804252e-01	-1.420782e-01	3.068520e-02	-8.357511e-02
	0.000000e+00	-9.773062e-03	3.678833e-02	-1.834702e-01	-1.778197e-01
	8.465145e-03	8.587882e-02	-2.556848e-02	1.094626e-01	3.874636e-02
	-2.927592e-02	-1.726960e-01	-2.597356e-02	-7.228072e-02	1.203510e-01
	0.000000e+00	-2.112415e-01	0.000000e+00	1.365712e-01	0.000000e+00
	3.505024e-01	1.432176e-01	-4.468819e-02	0.000000e+00	3.566141e-02
	-1.137072e-02	-8.109218e-02	-3.930318e-02	-9.755846e-02	0.000000e+00
	9.082303e-02	2.500676e-02	-1.562702e-01	0.000000e+00	8.860830e-03
	-1.851064e-01	1.071777e-01	0.000000e+00	-1.401498e-01	-1.122344e-01
	-1.531077e-01	2.405685e-01	7.780046e-02	-6.850366e-02	0.000000e+00
	0.000000e+00	-8.732510e-02	0.000000e+00	0.000000e+00	1.210968e-01
	0.000000e+00	-3.193131e-03	2.493715e-01	0.000000e+00	-5.564532e-03
19	-1.743298e-01	-5.285450e-02	-8.769317e-02	-1.550521e-03	-2.397181e-01
	1.626664e-01	-1.221369e-01	0.000000e+00	7.441352e-02	6.362223e-02
	3.899942e-03	-2.432545e-01	1.973500e-02	4.628554e-02	-3.293500e-02
	-8.367816e-02	2.604903e-01	1.513346e-01	-3.229359e-04	1.071280e-02
	6.844914e-02	9.575964e-03	9.767970e-02	0.000000e+00	-1.468418e-01
	1.484347e-01	-9.321898e-02	-2.115592e-02	-8.271987e-03	0.000000e+00
	0.000000e+00	0.000000e+00	1.912192e-01	-2.163121e-02	1.444020e-01
	-1.869704e-01	2.258942e-02	-1.005450e-01	1.686485e-02	3.449182e-02
	0.000000e+00	1.265406e-01	-1.885987e-01	6.532769e-02	9.884507e-02
	0.000000e+00	-1.509707e-01	-5.119377e-02	-8.328734e-02	0.000000e+00
	0.000000e+00	6.118804e-02	0.000000e+00	-5.088449e-02	-3.027601e-02
	-1.064343e-01	0.000000e+00	0.000000e+00	1.208422e-01	0.000000e+00
	-8.676070e-02	-1.444054e-02	7.356741e-02	1.486961e-01	2.633528e-01
	4.295070e-02	9.466460e-03	8.172152e-03	-6.033778e-02	-1.513955e-01
	0.000000e+00	1.409087e-02	-2.111613e-01	2.040424e-01	-1.934707e-01
	1.928846e-02	1.251277e-01	1.414914e-01	8.033654e-03	1.369082e-01
20	-1.519352e-02	-1.284745e-01	0.000000e+00	1.934293e-02	0.000000e+00
	-3.473866e-02	-2.520794e-02	-1.274821e-01	-1.052486e-01	-4.069012e-01
	-2.445286e-02	-1.001322e-02	8.382109e-02	-3.331394e-02	-1.388487e-01
	-3.735461e-01	1.400303e-01	-4.070785e-02	-2.932206e-02	-4.841131e-02
	-4.696700e-02	5.512613e-02	-7.814900e-03	0.000000e+00	0.000000e+00
	-5.830980e-02	0.000000e+00	-1.923080e-01	1.946200e-02	-8.631478e-02
	6.633691e-03	0.000000e+00	-1.041185e-02	-7.724683e-02	0.000000e+00
	2.137232e-01	7.734106e-02	3.936302e-02	-3.046806e-01	0.000000e+00
	1.048632e-01	-2.653142e-02	0.000000e+00	0.000000e+00	-3.520869e-02
	-1.257286e-01	-7.482710e-02	0.000000e+00	0.000000e+00	0.000000e+00
	-2.306185e-01	-7.709291e-03	1.608664e-01	-1.341210e-01	0.000000e+00
	4.221632e-02	5.077273e-02	-1.366954e-02	0.000000e+00	-1.982493e-01
	0.000000e+00	2.528824e-01	2.098448e-01	-6.767017e-02	-6.837550e-02
	-1.692620e-01	-1.269179e-01	9.442349e-02	2.714258e-03	5.883445e-02
	-4.826591e-02	-9.128623e-02	-2.061996e-01	0.000000e+00	2.488538e-01
	-1.951189e-01	5.804066e-02	-9.983543e-02	7.001870e-02	1.668746e-01
21	2.571811e-01	3.345241e-03	5.234505e-02	-1.358375e-02	2.482888e-01
	0.000000e+00	0.000000e+00	0.000000e+00	2.253912e-01	-4.736172e-02
	2.592065e-01	0.000000e+00	0.000000e+00	7.018156e-02	-2.102318e-02
	-1.016980e-01	-1.107739e-01	0.000000e+00	-4.203513e-03	3.352506e-02
	-1.218223e-01	-2.787777e-02	2.886978e-02	0.000000e+00	0.000000e+00
	0.000000e+00	-1.010319e-01	0.000000e+00	0.000000e+00	-1.659465e-01
	1.153894e-01	-1.265708e-01	0.000000e+00	-6.094101e-02	-2.191994e-01
	8.643431e-02	-5.758055e-02	-1.118129e-01	7.647143e-03	4.483539e-03
	0.000000e+00	8.982566e-02	5.374382e-02	-9.004497e-03	-1.191895e-01
	3.113542e-01	-1.678654e-02	-2.343062e-02	5.884172e-02	1.196362e-01
	5.615357e-02	0.000000e+00	0.000000e+00	-4.995218e-02	5.119399e-02
	-4.162973e-02	1.549152e-01	1.180959e-01	-1.219568e-02	-1.592638e-01
	-1.469312e-01	-7.649948e-03	-8.582183e-02	-6.274144e-02	1.686104e-01
	0.000000e+00	0.000000e+00	2.896473e-01	1.755413e-01	5.550749e-02
	-1.516775e-01	-2.078092e-01	2.755288e-02	0.000000e+00	0.000000e+00
	-1.211131e-02	-6.926627e-02	8.253777e-02	-9.207196e-02	2.037183e-01
22	6.101103e-03	0.000000e+00	1.559555e-02	0.000000e+00	-1.115257e-01

	3.665647e-02	-9.792650e-02	-6.448544e-02	1.434751e-01	-9.176196e-03
	0.000000e+00	1.032965e-02	9.677500e-02	-1.138917e-01	0.000000e+00
	0.000000e+00	1.800339e-02	2.878965e-02	-9.219384e-02	-4.468555e-02
	-9.045225e-02	-3.078079e-02	2.362066e-02	0.000000e+00	1.511604e-01
	2.924474e-02	1.567455e-01	0.000000e+00	9.718517e-02	-6.342785e-02
	2.841692e-01	1.056360e-01	-3.782579e-01	-1.674601e-01	4.588354e-02
	0.000000e+00	0.000000e+00	-1.328063e-01	-9.278624e-02	1.600208e-01
	0.000000e+00	3.977479e-02	1.584520e-01	-7.385105e-02	0.000000e+00
23	-2.033872e-01	-2.954464e-02	4.687919e-02	-1.945650e-01	-2.311004e-01
	0.000000e+00	0.000000e+00	-1.069994e-01	-1.106644e-01	1.915462e-01
	-1.447804e-01	-1.768089e-02	0.000000e+00	-6.137063e-02	0.000000e+00
	-1.974016e-01	-1.051958e-01	-2.333684e-01	0.000000e+00	-1.421298e-01
	-3.481893e-02	-3.819962e-02	-1.036749e-01	-1.971943e-01	0.000000e+00
	-1.370049e-01	2.039143e-03	0.000000e+00	2.050813e-02	6.836043e-02
	1.620574e-02	2.665693e-02	4.919855e-03	-2.346562e-02	2.885252e-01
	7.988900e-02	5.000554e-03	0.000000e+00	8.430111e-02	4.071794e-02
	0.000000e+00	-5.589191e-02	-1.826511e-01	0.000000e+00	1.240300e-01
	-1.288806e-01	0.000000e+00	-1.972733e-02	0.000000e+00	7.472074e-03
	-5.491357e-02	-7.805638e-02	1.648394e-01	-3.717156e-02	1.767755e-01
	-4.941719e-02	0.000000e+00	9.244929e-02	-7.186580e-02	1.122488e-01
	0.000000e+00	-1.113929e-01	-5.270017e-02	-7.245102e-02	0.000000e+00
	6.667792e-02	2.311812e-01	2.534244e-02	0.000000e+00	2.812384e-02
	-2.078793e-01	6.663837e-02	-1.260824e-01	-3.933493e-02	-2.745107e-01
	-1.102345e-01	2.036201e-01	7.187688e-02	-1.801592e-01	0.000000e+00
24	2.768485e-01	-5.426589e-02	1.054037e-01	0.000000e+00	2.206461e-01
	1.395267e-01	0.000000e+00	-1.615684e-01	7.346235e-02	1.222862e-01
	-1.402718e-02	0.000000e+00	-9.378122e-02	2.207879e-01	-6.380153e-02
	0.000000e+00	0.000000e+00	0.000000e+00	1.618378e-02	7.228883e-02
	1.053941e-02	-2.764513e-02	-1.580412e-01	-1.416054e-02	-1.448974e-01
	2.468120e-01	-1.613466e-01	-6.772158e-02	1.746352e-01	-4.395303e-02
	-1.048592e-01	-9.243996e-03	2.335437e-02	1.358646e-02	1.167344e-01
	0.000000e+00	0.000000e+00	4.745781e-02	-1.882347e-01	2.225801e-01
	-1.319828e-01	-4.920179e-03	-7.518771e-02	0.000000e+00	-1.085564e-01
	1.457614e-01	2.121899e-01	4.388111e-02	0.000000e+00	6.757072e-02
	9.576823e-02	2.488467e-01	1.099766e-01	6.889626e-03	0.000000e+00
	-1.308049e-01	-3.002361e-03	-4.020611e-02	0.000000e+00	0.000000e+00
	1.913417e-01	2.694649e-02	-5.608093e-02	1.325018e-01	5.230738e-02
	0.000000e+00	1.656550e-02	-2.363928e-01	0.000000e+00	-1.732607e-01
	0.000000e+00	0.000000e+00	1.845152e-02	3.880707e-02	-1.906400e-01
	1.471496e-01	0.000000e+00	-1.240457e-02	-1.647823e-01	0.000000e+00
25	0.000000e+00	0.000000e+00	0.000000e+00	4.969337e-02	0.000000e+00
	-2.944231e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	4.412249e-02	0.000000e+00	0.000000e+00	0.000000e+00	-2.704061e-02
	0.000000e+00	-1.791069e-01	1.657378e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.658176e-01	1.292367e-02
	0.000000e+00	0.000000e+00	-3.220622e-01	0.000000e+00	0.000000e+00
	-3.411179e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-7.780495e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	8.003307e-04	0.000000e+00	1.067928e-02	0.000000e+00	0.000000e+00
	-1.034487e-02	1.671776e-01	-1.650592e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.316753e-02
	-4.985107e-01	2.789020e-01	0.000000e+00	7.973339e-02	-4.874192e-01
	0.000000e+00	0.000000e+00	2.918302e-01	0.000000e+00	1.101740e-01
	0.000000e+00	-3.548299e-03	-1.001646e-01	0.000000e+00	0.000000e+00
26	-1.943853e-01	9.289961e-02	-4.925807e-02	1.534220e-01	0.000000e+00
	0.000000e+00	-5.734173e-03	-1.223073e-01	1.419700e-01	5.449676e-02
	0.000000e+00	0.000000e+00	-1.032271e-01	-2.442879e-03	-1.211700e-02
	2.357580e-02	1.539323e-01	-7.251933e-02	-1.216809e-01	9.112048e-02
	-7.897280e-02	0.000000e+00	-6.676657e-02	2.192769e-01	-7.853070e-02
	1.173405e-03	2.337760e-01	1.960734e-01	1.427804e-01	0.000000e+00
	-1.510202e-01	-2.319302e-01	0.000000e+00	0.000000e+00	1.473334e-01
	-1.262754e-01	-2.242199e-02	3.514803e-03	1.013120e-01	-6.971137e-02
	-1.387573e-02	1.358712e-01	0.000000e+00	0.000000e+00	-2.070520e-02
	7.683654e-02	-9.101964e-02	4.464331e-02	2.806062e-01	7.742046e-02
	3.727559e-02	1.104044e-01	1.616145e-01	1.584554e-01	4.747138e-02
	2.994230e-02	-6.985522e-02	0.000000e+00	-1.136675e-02	0.000000e+00
	-6.185979e-03	-1.549233e-02	1.045641e-02	1.746030e-01	-2.995573e-01
	-2.489278e-01	0.000000e+00	1.558460e-02	2.120860e-01	-3.887424e-02
	7.633126e-02	-6.632909e-03	-5.264083e-02	-2.512232e-01	-2.909316e-02
	0.000000e+00	-1.296611e-01	4.726502e-02	-4.765115e-02	-3.960366e-02
	6.227123e-02	-6.115421e-02	1.772012e-01	-1.985426e-04	0.000000e+00
	2.845233e-03	0.000000e+00	-2.452843e-02	0.000000e+00	6.547032e-02
	-9.449153e-02	-1.351965e-02	4.338376e-02	1.436002e-01	0.000000e+00

	-3.087789e-02	-3.266681e-02	-9.442008e-02	-1.016076e-01	1.085821e-01
	-1.783888e-01	8.450606e-02	2.039862e-01	4.196332e-01	-4.847599e-03
	-1.473677e-01	0.000000e+00	0.000000e+00	1.523364e-01	-1.923450e-02
	-1.203434e-02	-4.233756e-02	2.655249e-02	-1.614154e-01	0.000000e+00
27	5.193471e-02	1.844770e-01	-8.687869e-02	1.947364e-02	0.000000e+00
	3.657959e-03	8.909392e-02	6.730644e-02	1.717418e-01	-8.916063e-02
	0.000000e+00	1.185226e-01	3.711998e-02	-8.372331e-02	-2.599214e-01
	0.000000e+00	0.000000e+00	0.000000e+00	-7.261579e-02	1.168693e-01
	1.003183e-01	-1.683789e-01	0.000000e+00	-1.157689e-01	3.814610e-02
	0.000000e+00	-1.558778e-01	0.000000e+00	2.052883e-01	1.187179e-01
	0.000000e+00	2.716609e-01	9.086628e-02	0.000000e+00	-8.152492e-02
	1.152438e-02	6.506586e-02	-2.391154e-01	6.323361e-02	-2.196630e-01
	9.524021e-02	1.067758e-01	0.000000e+00	-2.139481e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	2.827629e-01	0.000000e+00
	-1.308892e-01	0.000000e+00	0.000000e+00	4.769953e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.079127e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	3.183510e-02	0.000000e+00	0.000000e+00	-5.913722e-02	-1.050104e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.012387e-02
28	1.065648e-01	1.963168e-01	3.825294e-01	-4.754090e-01	0.000000e+00
	1.185495e-01	0.000000e+00	0.000000e+00	1.897743e-01	-1.655077e-01
	1.061208e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-6.583502e-02	0.000000e+00
	0.000000e+00	-1.636285e-01	0.000000e+00	0.000000e+00	0.000000e+00
	4.178372e-02	0.000000e+00	1.519792e-01	0.000000e+00	0.000000e+00
	0.000000e+00	-3.009328e-02	0.000000e+00	0.000000e+00	0.000000e+00
	1.986235e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-1.884589e-01	6.171096e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	2.237388e-01	-9.160060e-04	2.357267e-02
	-1.020783e-01	-1.158738e-04	1.624502e-01	0.000000e+00	0.000000e+00
	-1.167449e-01	2.626511e-02	1.492835e-01	-9.281360e-02	0.000000e+00
	0.000000e+00	-1.768597e-01	1.222033e-01	8.611261e-02	-1.103476e-01
	-9.499669e-02	-5.612902e-02	0.000000e+00	-1.023128e-01	-6.477947e-02
	1.358912e-01	7.992765e-02	-7.216670e-02	0.000000e+00	-1.749471e-02
	0.000000e+00	2.692556e-01	1.296867e-01	1.262529e-01	3.501203e-01
29	1.721932e-01	0.000000e+00	1.383511e-01	-5.265588e-02	0.000000e+00
	0.000000e+00	4.405776e-02	2.408024e-02	5.966376e-02	0.000000e+00
	-2.866341e-02	8.038671e-02	1.778118e-01	-7.938990e-02	0.000000e+00
	1.172803e-01	1.042891e-01	-1.354791e-01	2.549725e-01	0.000000e+00
	-5.561799e-02	4.089552e-04	-5.883588e-02	7.680114e-02	2.038721e-01
	-5.211513e-02	0.000000e+00	0.000000e+00	0.000000e+00	7.017971e-02
	-3.085604e-02	0.000000e+00	-1.104811e-01	1.854449e-01	-2.934458e-01
	0.000000e+00	-6.806617e-02	0.000000e+00	8.742157e-02	5.453546e-02
	0.000000e+00	-7.858650e-02	-1.449723e-02	-2.933699e-01	5.135650e-02
	1.541001e-01	-7.176381e-02	0.000000e+00	-7.894326e-02	-1.020498e-03
	-1.705167e-01	-2.387415e-01	9.428056e-02	8.317553e-03	4.704671e-02
	-4.194044e-02	0.000000e+00	1.947631e-02	6.391195e-02	7.388619e-02
	-4.441126e-02	2.045547e-01	1.248566e-01	1.248314e-01	-2.107189e-01
	9.500907e-02	0.000000e+00	-5.820566e-02	-4.145451e-02	0.000000e+00
	-1.702583e-02	0.000000e+00	0.000000e+00	5.364237e-03	3.947363e-01
	-6.161966e-02	-1.294796e-01	0.000000e+00	0.000000e+00	5.656169e-03
30	-7.350562e-02	-1.256491e-02	0.000000e+00	-9.153253e-02	1.651394e-02
	0.000000e+00	-1.382404e-01	5.724306e-02	0.000000e+00	0.000000e+00
	-2.709804e-01	-2.631862e-02	-5.459896e-02	2.909796e-01	-1.059908e-02
	-1.304712e-01	9.861576e-02	0.000000e+00	5.765521e-03	-9.341256e-02
	-1.051419e-01	-6.097612e-02	-2.384960e-02	3.959643e-02	0.000000e+00
	1.136109e-01	-3.332754e-03	8.247378e-02	-4.251916e-02	-1.133671e-01
	1.153753e-01	-1.875303e-02	-2.017118e-01	0.000000e+00	2.377363e-02
	9.679881e-02	-1.911092e-01	-5.704860e-02	5.687258e-02	-1.136134e-01
	2.980186e-01	0.000000e+00	8.140460e-02	9.410392e-02	1.894892e-01
	-1.905698e-01	1.254690e-01	2.383226e-01	4.232296e-02	-4.957388e-02
	-2.839471e-01	-1.765860e-02	-2.213603e-02	-1.726387e-01	1.990642e-02
	9.296396e-02	3.428154e-02	-5.562755e-02	2.974755e-01	-1.910826e-01
	0.000000e+00	4.565392e-02	-7.177272e-02	0.000000e+00	-2.615683e-02
	-4.314982e-03	0.000000e+00	6.233544e-02	9.517921e-02	-1.046339e-02
	-4.591090e-02	-6.953862e-02	-8.183660e-02	-4.795807e-02	0.000000e+00
	1.665708e-01	-1.120687e-01	4.507969e-02	0.000000e+00	2.036847e-01
31	5.471038e-02	0.000000e+00	0.000000e+00	5.642816e-02	0.000000e+00
	-8.854070e-02	7.765918e-02	0.000000e+00	-2.859845e-01	2.925882e-02
	-1.946581e-01	0.000000e+00	0.000000e+00	0.000000e+00	-1.938060e-01
	3.486312e-02	-1.064576e-01	1.475572e-01	0.000000e+00	-1.277474e-01
	-9.939930e-02	0.000000e+00	5.605459e-02	-3.080950e-02	-2.063663e-01
	0.000000e+00	4.203537e-02	-1.606857e-01	-1.895805e-02	-1.835702e-01
	1.068181e-01	0.000000e+00	-1.316566e-01	0.000000e+00	1.235550e-03

	0.000000e+00	-7.951529e-02	-2.576212e-01	-1.667024e-01	9.901331e-02
	5.108171e-02	0.000000e+00	8.534033e-02	-1.213613e-01	-7.715867e-02
	0.000000e+00	-3.283459e-01	0.000000e+00	0.000000e+00	-1.895084e-01
	-9.563014e-02	2.310340e-04	-9.353452e-02	2.521645e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	1.762867e-02	0.000000e+00	0.000000e+00	-5.999607e-01	0.000000e+00
	4.144656e-02	0.000000e+00	0.000000e+00	-2.034636e-01	0.000000e+00
	1.458463e-03	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
32	0.000000e+00	1.715785e-01	0.000000e+00	-1.317531e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.786239e-01
	1.290465e-02	0.000000e+00	0.000000e+00	-1.327665e-01	0.000000e+00
	0.000000e+00	-1.196896e-01	1.735534e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.396934e-01
	0.000000e+00	-4.077047e-02	0.000000e+00	9.309571e-03	0.000000e+00
	-2.649266e-01	0.000000e+00	0.000000e+00	4.313061e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-3.249480e-02	-1.602778e-01	-6.333429e-02	1.124733e-01	9.284082e-02
	-7.874490e-02	1.697473e-02	-1.665857e-01	1.504903e-01	-5.782890e-02
	-1.103568e-01	4.952956e-03	-1.196616e-02	1.531437e-01	-4.913073e-02
	-3.093560e-03	8.516176e-02	2.481723e-01	1.167599e-03	-1.272678e-01
	2.011214e-01	1.812335e-01	0.000000e+00	1.507126e-01	0.000000e+00
	7.901020e-02	1.156127e-01	-6.001019e-03	0.000000e+00	1.571657e-01
	8.896735e-02	-1.495840e-01	0.000000e+00	-5.957299e-02	-1.932359e-01
33	-1.981662e-01	-1.408195e-01	1.547140e-01	0.000000e+00	1.903718e-01
	-4.179015e-02	6.507748e-03	-9.187204e-02	-1.767419e-02	2.273528e-02
	7.568693e-02	0.000000e+00	3.422445e-02	-2.475617e-01	0.000000e+00
	0.000000e+00	-2.176993e-01	-1.518162e-01	0.000000e+00	0.000000e+00
	1.622839e-01	4.078059e-02	0.000000e+00	-1.575357e-02	-1.531148e-01
	0.000000e+00	0.000000e+00	2.582483e-01	0.000000e+00	0.000000e+00
	1.722287e-02	0.000000e+00	1.401425e-01	-2.834442e-01	0.000000e+00
	-4.388519e-02	5.433398e-02	4.301994e-02	0.000000e+00	-3.685032e-02
	-9.643744e-02	3.270401e-02	8.717625e-02	1.620158e-01	-1.140804e-01
	0.000000e+00	0.000000e+00	-1.258063e-01	6.900290e-02	6.882042e-02
	1.657469e-01	5.041063e-03	7.091919e-03	0.000000e+00	-4.646786e-02
	9.035891e-02	2.444057e-01	5.124819e-02	0.000000e+00	8.036846e-02
	1.829495e-01	-1.535938e-01	6.333969e-02	1.416789e-01	1.108288e-01
	3.108543e-01	-8.629087e-02	1.137684e-01	0.000000e+00	-1.000967e-01
	-8.071551e-03	-8.489311e-02	8.383948e-02	-1.459242e-01	1.260528e-01
	1.209160e-01	4.720706e-02	-6.475083e-02	0.000000e+00	9.236762e-02
34	0.000000e+00	0.000000e+00	-1.694174e-01	5.529078e-02	-1.038176e-01
	-3.115317e-02	8.995946e-02	1.787062e-01	-1.145721e-01	1.808379e-01
	0.000000e+00	-2.538114e-02	-1.518067e-01	5.550284e-02	1.106704e-01
	1.473961e-01	2.312183e-02	0.000000e+00	0.000000e+00	0.000000e+00
	2.225666e-01	0.000000e+00	-2.252502e-01	-5.131952e-02	7.789297e-02
	6.918818e-02	-1.203212e-01	-9.330358e-02	-2.350129e-03	0.000000e+00
	-2.256729e-01	0.000000e+00	0.000000e+00	0.000000e+00	2.520077e-01
	-5.338857e-02	1.610254e-01	2.779112e-02	-4.008395e-02	1.939551e-01
	3.084698e-03	2.298620e-01	5.295537e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.943733e-01	0.000000e+00
	0.000000e+00	0.000000e+00	-2.472905e-01	-3.882475e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.639278e-01
	0.000000e+00	0.000000e+00	1.376100e-01	0.000000e+00	2.537211e-01
	0.000000e+00	0.000000e+00	0.000000e+00	2.099858e-01	0.000000e+00
	-7.282094e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
35	0.000000e+00	-3.007074e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-3.569115e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	4.316505e-01	0.000000e+00	0.000000e+00	0.000000e+00	1.035779e-01
	1.644590e-01	3.491507e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	1.980560e-01	0.000000e+00	0.000000e+00
	-3.410442e-01	-2.253416e-01	-2.880538e-02	-1.556330e-03	0.000000e+00
	1.674552e-01	-8.590104e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-9.544261e-02	8.764337e-02	-9.653688e-03
	-3.633391e-02	-1.784087e-01	0.000000e+00	0.000000e+00	7.684226e-02
	-1.161073e-01	-2.905554e-02	1.314691e-01	4.387024e-02	0.000000e+00
	-8.399946e-02	-1.329643e-01	0.000000e+00	2.263473e-01	-9.109813e-02
	-6.411707e-02	2.581551e-01	-8.637272e-02	7.599245e-02	0.000000e+00
	-8.052940e-03	-3.893694e-02	-3.256879e-02	-3.483802e-02	9.887691e-02
	2.568428e-01	0.000000e+00	-1.989061e-01	-1.173420e-02	4.198608e-04
36	1.047744e-01	-6.257062e-02	3.395021e-02	0.000000e+00	-9.896559e-02
	0.000000e+00	1.314746e-01	0.000000e+00	0.000000e+00	4.663994e-02

	9.161910e-02	0.000000e+00	-8.746310e-02	8.973657e-02	2.593786e-01
	0.000000e+00	0.000000e+00	0.000000e+00	-5.490051e-02	0.000000e+00
	1.607468e-01	-3.860441e-02	0.000000e+00	-1.397199e-01	1.348633e-01
	2.328396e-01	-2.338241e-01	8.512305e-02	3.279883e-01	6.882735e-02
	-1.980798e-01	4.811943e-03	0.000000e+00	-1.167096e-01	-1.555851e-01
	-1.061918e-01	2.621488e-01	-5.555580e-02	5.970274e-02	-1.250368e-01
	0.000000e+00	-1.020933e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	4.745946e-02	0.000000e+00	1.768680e-02
	-1.278193e-01	-1.403725e-01	-1.680588e-01	1.623146e-01	0.000000e+00
	7.354951e-02	1.264290e-01	7.295935e-02	-2.061014e-01	8.926743e-02
	1.131144e-01	-9.003063e-02	-1.289876e-02	-5.068947e-02	-3.042417e-02
	-1.389432e-01	2.839091e-01	1.360499e-01	0.000000e+00	9.208447e-02
	4.404191e-02	0.000000e+00	0.000000e+00	4.638354e-02	0.000000e+00
	0.000000e+00	-4.257557e-02	1.589819e-01	-1.500221e-01	1.111023e-02
37	-2.828580e-01	-2.348762e-01	1.621372e-01	-2.907762e-01	1.013477e-01
	6.765024e-02	-1.693189e-02	8.917216e-02	-1.571030e-01	-1.214169e-03
	0.000000e+00	0.000000e+00	0.000000e+00	-6.111805e-02	-9.762530e-02
	8.643147e-02	1.823930e-01	-1.372005e-02	0.000000e+00	1.366321e-01
	1.848180e-01	0.000000e+00	-3.911247e-02	-4.055479e-02	1.780939e-02
	0.000000e+00	2.946363e-02	-1.192290e-01	0.000000e+00	0.000000e+00
	0.000000e+00	-1.216554e-01	2.895767e-02	1.510354e-02	-1.066618e-01
	-2.996593e-02	-1.699082e-01	-1.666425e-01	3.498710e-02	1.534413e-01
	0.000000e+00	0.000000e+00	0.000000e+00	-3.076427e-01	-2.828831e-02
	-1.157164e-01	2.539262e-01	-9.839177e-03	1.868605e-01	-9.652212e-02
	0.000000e+00	-1.298957e-01	1.520612e-01	0.000000e+00	2.696069e-02
	1.443890e-01	-2.560296e-01	3.866190e-02	1.575008e-01	-6.384536e-02
	-1.505456e-02	0.000000e+00	4.025807e-02	0.000000e+00	0.000000e+00
	1.396234e-01	-3.074226e-01	0.000000e+00	7.165064e-02	-1.022926e-01
	8.836515e-02	0.000000e+00	-6.627984e-02	-2.345870e-01	0.000000e+00
	6.919119e-02	0.000000e+00	6.837422e-02	-1.672438e-05	1.389712e-01
38	3.312315e-02	-7.077093e-02	0.000000e+00	6.048851e-02	3.473295e-01
	0.000000e+00	0.000000e+00	0.000000e+00	7.075194e-02	0.000000e+00
	0.000000e+00	-7.568043e-02	-2.129591e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	4.872224e-03	-1.980266e-01	1.246747e-02
	0.000000e+00	0.000000e+00	2.088179e-01	-9.201540e-02	0.000000e+00
	9.524766e-02	-1.526404e-02	3.773394e-02	-9.325427e-02	0.000000e+00
	1.474800e-01	1.385337e-01	-7.676570e-03	1.862375e-01	8.729067e-02
	-6.250084e-02	8.532207e-02	-1.868594e-01	-1.169226e-01	0.000000e+00
	-2.069486e-01	-6.580359e-02	1.870817e-02	6.986667e-02	0.000000e+00
	0.000000e+00	-1.073892e-02	-2.812366e-01	-7.657860e-02	0.000000e+00
	3.512838e-02	0.000000e+00	3.564242e-02	0.000000e+00	2.004855e-01
	-2.531501e-02	0.000000e+00	0.000000e+00	0.000000e+00	-8.875913e-02
	2.464498e-01	0.000000e+00	1.774883e-01	1.569065e-01	0.000000e+00
	2.246130e-01	-2.177168e-02	-2.372741e-01	-8.412304e-03	1.283020e-03
	-2.699531e-02	0.000000e+00	1.945082e-01	1.217185e-01	0.000000e+00
	-4.463774e-02	9.458582e-02	0.000000e+00	3.443969e-02	-9.023874e-03
39	5.075217e-02	-2.881549e-01	-4.192947e-02	1.205460e-02	-8.494505e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-2.062027e-01
	-1.318434e-02	-8.396941e-02	8.192604e-03	7.442917e-02	8.093105e-04
	2.485074e-02	0.000000e+00	2.503018e-01	4.266745e-02	1.091002e-01
	1.759357e-01	0.000000e+00	1.805937e-01	-1.318367e-01	1.579972e-01
	-1.045582e-02	-1.089510e-01	0.000000e+00	0.000000e+00	-1.105174e-02
	0.000000e+00	-9.935433e-02	3.004021e-01	-1.174369e-01	0.000000e+00
	-1.823402e-01	2.382599e-02	-1.925151e-01	1.644587e-01	-1.720008e-02
	-6.505589e-02	0.000000e+00	3.934063e-02	0.000000e+00	1.047833e-01
	3.089454e-01	-2.917191e-01	0.000000e+00	-2.824003e-02	8.091069e-02
	0.000000e+00	3.916826e-02	-7.923549e-02	6.460930e-02	1.516750e-01
	0.000000e+00	-2.062795e-01	5.940737e-02	9.132210e-02	4.466302e-02
	0.000000e+00	1.034116e-01	-8.211226e-03	7.706052e-02	-4.787372e-02
	-1.092646e-01	-1.151051e-01	2.177804e-01	-6.014922e-02	1.898578e-03
	1.381714e-02	-3.393020e-01	4.645121e-02	-4.170154e-02	1.272961e-02
	1.605749e-02	1.422081e-01	-1.059638e-01	1.490378e-03	0.000000e+00
40	6.668815e-02	6.458632e-02	0.000000e+00	1.227015e-01	-1.633692e-01
	0.000000e+00	4.529190e-02	3.341499e-02	-1.706555e-01	-1.767923e-02
	7.055236e-02	0.000000e+00	1.535658e-01	-5.135086e-02	-4.116459e-02
	-1.934740e-02	-9.514686e-02	0.000000e+00	-3.752438e-02	-8.057177e-03
	0.000000e+00	-5.132289e-02	-1.272908e-01	1.616529e-01	3.110470e-01
	2.981113e-02	6.737767e-02	-3.154569e-02	0.000000e+00	-1.230242e-01
	-1.264473e-01	-9.975108e-03	5.510241e-02	1.356748e-03	-2.867838e-01
	0.000000e+00	0.000000e+00	9.032106e-02	-7.846995e-02	2.757774e-02
	0.000000e+00	-9.055331e-02	-1.846260e-01	-9.950212e-02	-1.253535e-01
	0.000000e+00	0.000000e+00	0.000000e+00	4.501720e-02	6.428702e-02
	0.000000e+00	2.305043e-02	0.000000e+00	0.000000e+00	-1.206849e-01
	-7.981162e-02	-2.945195e-02	-1.242012e-02	-2.160389e-02	5.569625e-02
	0.000000e+00	-1.784327e-01	1.057502e-01	1.280805e-02	-2.130417e-01

	0.000000e+00	0.000000e+00	-7.042856e-02	6.973646e-02	-9.000307e-02
	0.000000e+00	0.000000e+00	1.963545e-01	-2.209776e-01	-1.159162e-01
	1.089002e-01	3.598578e-02	0.000000e+00	1.508839e-01	0.000000e+00
41	0.000000e+00	-1.520191e-01	1.132788e-01	-8.138808e-02	-1.967588e-01
	1.621106e-01	-3.013043e-01	0.000000e+00	9.831732e-02	-1.960935e-01
	0.000000e+00	2.040954e-02	1.586663e-01	0.000000e+00	0.000000e+00
	1.258156e-02	-6.600765e-02	8.108480e-02	0.000000e+00	0.000000e+00
	4.872772e-02	6.734531e-02	6.629723e-02	-2.226854e-01	0.000000e+00
	1.666765e-01	-2.962752e-02	-1.062781e-01	2.287146e-02	0.000000e+00
	6.247909e-02	-9.318022e-02	0.000000e+00	0.000000e+00	-3.496602e-01
	-9.096877e-02	0.000000e+00	-1.378621e-01	1.181396e-01	-2.840278e-02
	1.432092e-01	3.074735e-01	6.850848e-02	-8.330212e-02	1.909310e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.642728e-01
	-2.254947e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-4.199278e-01	0.000000e+00	0.000000e+00	-1.119695e-01
	0.000000e+00	0.000000e+00	-5.013975e-02	0.000000e+00	0.000000e+00
	0.000000e+00	9.085873e-02	0.000000e+00	1.234282e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.948011e-01
	4.164939e-01	0.000000e+00	-1.975164e-01	0.000000e+00	3.974489e-01
42	1.255559e-01	0.000000e+00	0.000000e+00	-3.179975e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	9.725258e-02	1.632868e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	2.727188e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	1.521279e-01	0.000000e+00
	5.928686e-02	-2.478354e-01	1.280224e-01	0.000000e+00	0.000000e+00
	-2.204625e-01	0.000000e+00	-2.559606e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	7.364691e-03	-3.124810e-02
	1.298542e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	1.220730e-01	2.252594e-01	-7.993162e-02	-2.722844e-01	0.000000e+00
	-9.633626e-02	-2.244533e-02	1.172702e-01	-3.820555e-02	0.000000e+00
	2.135619e-01	1.075710e-02	1.308232e-01	-5.431930e-02	-8.699663e-02
	5.953823e-02	0.000000e+00	-3.470596e-02	7.298985e-02	0.000000e+00
	0.000000e+00	2.859825e-01	-1.130891e-02	0.000000e+00	-1.229326e-01
	-1.334958e-01	1.076912e-01	-3.414621e-02	7.740691e-02	2.114631e-02
	0.000000e+00	-1.538354e-01	-1.791569e-01	0.000000e+00	0.000000e+00
43	-9.360868e-02	3.598030e-02	0.000000e+00	-2.008224e-02	-1.604317e-01
	-1.529427e-01	6.756241e-02	-2.831907e-02	-3.816459e-02	3.048317e-02
	1.856279e-01	1.311494e-01	0.000000e+00	-1.676460e-01	-1.164622e-01
	1.631631e-01	0.000000e+00	-4.891640e-02	0.000000e+00	-6.938017e-02
	8.412501e-02	0.000000e+00	2.502332e-02	-1.766020e-02	3.416777e-02
	2.752765e-02	7.933568e-02	-1.772837e-01	1.419957e-01	-1.426049e-01
	1.970538e-01	-1.578509e-01	1.023560e-01	-2.452806e-03	1.041795e-01
	-2.912046e-01	1.977646e-01	9.972805e-02	8.903846e-02	5.978534e-02
	-6.342193e-02	-6.795794e-03	-1.045438e-01	-5.676429e-02	1.557632e-01
	0.000000e+00	0.000000e+00	-1.902108e-01	0.000000e+00	1.308436e-01
	1.643991e-01	7.194531e-02	4.356217e-02	-9.772570e-02	1.361725e-01
	-3.498207e-02	-2.768522e-03	1.331455e-02	1.008620e-01	-2.086498e-02
	1.473316e-01	1.177691e-01	1.206780e-01	3.084751e-01	-5.091523e-03
	-2.295596e-01	2.515759e-02	-1.903997e-01	-5.554314e-02	0.000000e+00
	7.272176e-02	1.097104e-01	4.312816e-03	-1.696811e-01	2.126838e-02
	-1.813055e-01	-9.788775e-02	-1.094248e-01	-8.174628e-02	1.055069e-01
44	6.255002e-02	0.000000e+00	-1.905984e-01	-1.638862e-01	-2.723354e-02
	0.000000e+00	1.944689e-01	-6.630454e-02	1.296156e-01	0.000000e+00
	-1.119981e-01	1.488806e-01	4.535420e-02	1.364502e-02	-3.266117e-03
	-9.916708e-02	7.744209e-02	3.081058e-02	-1.459658e-01	-3.985174e-02
	-8.794389e-02	-4.312829e-02	0.000000e+00	7.895293e-02	3.153557e-02
	2.088699e-01	-1.395333e-03	1.322010e-01	0.000000e+00	0.000000e+00
	-5.767224e-02	6.542353e-02	1.890431e-01	-8.147565e-02	1.391408e-01
	-5.726032e-02	0.000000e+00	8.630424e-02	-2.783643e-01	0.000000e+00
	-6.299796e-02	0.000000e+00	-2.671113e-01	-1.256430e-02	-3.489397e-02
	-1.606524e-01	1.581753e-01	-4.895433e-02	-5.312045e-03	1.242068e-01
	-1.094206e-01	9.347972e-03	5.340980e-02	0.000000e+00	5.162187e-02
	0.000000e+00	-1.655636e-02	-7.596710e-02	0.000000e+00	-3.022955e-02
	-3.886658e-02	-5.994639e-02	0.000000e+00	-9.850787e-02	0.000000e+00
	-4.177111e-02	5.655181e-02	0.000000e+00	-3.017956e-02	2.082475e-01
	8.832982e-02	0.000000e+00	4.860526e-02	-1.066447e-01	1.263485e-01
	0.000000e+00	0.000000e+00	-1.124461e-01	-1.656961e-01	-7.817247e-02
45	0.000000e+00	-8.067810e-03	4.878099e-03	1.209587e-01	-2.755680e-01
	4.452356e-02	-6.460030e-02	0.000000e+00	1.082772e-01	-7.125887e-02
	1.320068e-01	0.000000e+00	1.448180e-01	2.997617e-02	1.096025e-01
	0.000000e+00	1.959266e-01	0.000000e+00	-4.884703e-01	1.526499e-01
	-2.632180e-02	0.000000e+00	0.000000e+00	-2.216876e-02	-8.222510e-02
	1.417504e-01	-3.324101e-01	0.000000e+00	1.821565e-01	-2.275971e-02
	4.548564e-04	-1.144147e-01	4.777031e-02	1.165208e-02	-1.380926e-01
	-3.571294e-02	8.295597e-02	9.191137e-02	-3.556537e-02	-2.350108e-01

	-4.201951e-02	-8.026653e-03	5.420449e-02	0.000000e+00	9.562975e-02
	0.000000e+00	7.634479e-02	0.000000e+00	0.000000e+00	5.828185e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.600761e-02
	0.000000e+00	-3.483103e-01	0.000000e+00	2.055003e-01	-2.933359e-01
	0.000000e+00	-6.925607e-02	0.000000e+00	1.669357e-01	0.000000e+00
	-4.132408e-02	0.000000e+00	0.000000e+00	6.769610e-02	-2.897804e-01
	0.000000e+00	0.000000e+00	-1.595861e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
46	4.920345e-01	-6.387433e-02	0.000000e+00	-9.932903e-02	0.000000e+00
	0.000000e+00	-3.868692e-02	-1.336535e-01	0.000000e+00	1.118689e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.541134e-01
	-4.953759e-02	2.800939e-01	0.000000e+00	-4.090675e-03	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.975055e-02
	1.425458e-01	0.000000e+00	0.000000e+00	3.250622e-01	0.000000e+00
	8.300103e-02	1.653923e-02	0.000000e+00	0.000000e+00	1.292001e-01
	-1.191415e-01	0.000000e+00	1.233994e-01	0.000000e+00	0.000000e+00
	-1.035613e-01	-5.806269e-02	0.000000e+00	0.000000e+00	0.000000e+00
	-3.307760e-02	-2.993981e-01	-8.582290e-02	9.358567e-03	-5.142987e-02
	4.258276e-02	6.641512e-02	2.666645e-02	2.863951e-02	6.827300e-02
	0.000000e+00	0.000000e+00	8.012966e-02	-3.928255e-02	-6.421129e-02
	-3.638282e-02	1.667841e-01	4.091731e-02	-8.523482e-03	0.000000e+00
	3.039413e-01	1.350559e-01	-9.241425e-02	-1.901282e-01	-2.654417e-01
	0.000000e+00	-1.464065e-02	-9.104504e-02	7.037415e-02	5.484661e-02
	6.556957e-03	-1.265556e-01	-9.843203e-03	1.389682e-02	-1.438335e-01
47	0.000000e+00	-4.877615e-02	-3.825533e-01	0.000000e+00	-9.254570e-02
	-3.946958e-02	1.537872e-01	-1.284628e-01	-7.193904e-02	1.150979e-01
	9.747426e-02	-4.018207e-02	-1.860719e-01	-2.972821e-03	0.000000e+00
	1.010504e-01	0.000000e+00	8.723965e-02	-1.065299e-01	-2.886868e-02
	-7.147130e-02	-6.547166e-02	8.612391e-02	6.704640e-02	-2.396629e-02
	-3.971305e-02	-1.949116e-01	-1.396781e-01	0.000000e+00	1.854541e-01
	-7.995628e-02	-7.020791e-02	8.505736e-02	-1.761238e-01	1.095653e-02
	9.161524e-02	-8.207477e-02	-7.938825e-02	1.764563e-01	-5.423461e-02
	3.505232e-02	0.000000e+00	-2.753778e-02	-6.810401e-02	-2.101099e-01
	-1.413620e-01	0.000000e+00	1.026553e-02	-8.476777e-02	-1.496839e-01
	-1.135335e-01	1.734192e-03	5.965361e-03	1.977258e-01	1.418120e-01
	1.839702e-01	-1.191329e-01	1.057696e-01	0.000000e+00	9.632366e-02
	-5.744714e-02	3.810491e-02	-7.678081e-02	-3.247526e-03	-2.351286e-02
	7.429464e-02	-2.411511e-01	2.185401e-01	-5.956781e-02	-2.491617e-01
	0.000000e+00	-2.307274e-01	0.000000e+00	0.000000e+00	-1.662879e-01
	0.000000e+00	8.092455e-02	0.000000e+00	2.123960e-02	0.000000e+00
48	0.000000e+00	-6.196349e-02	-3.478627e-02	1.238028e-01	-7.042740e-02
	5.480728e-02	-1.223048e-01	0.000000e+00	0.000000e+00	0.000000e+00
	-5.327531e-02	0.000000e+00	4.153144e-02	0.000000e+00	4.394394e-02
	-2.519259e-01	7.964419e-02	-1.629194e-01	-1.413204e-02	-7.562454e-02
	-1.310475e-01	-1.164455e-01	-1.257513e-01	0.000000e+00	-1.996453e-01
	0.000000e+00	-1.323992e-01	-1.759002e-02	0.000000e+00	0.000000e+00
	2.268515e-01	0.000000e+00	1.541401e-01	-4.741695e-02	-2.461421e-01
	2.409432e-01	-8.350389e-02	2.156594e-01	8.826172e-03	9.131511e-02
	0.000000e+00	0.000000e+00	1.385888e-01	3.282316e-04	0.000000e+00
	8.547926e-02	1.010874e-01	0.000000e+00	1.079392e-01	-8.836321e-02
	9.920058e-02	-8.851971e-02	-3.713604e-02	0.000000e+00	-1.012336e-01
	-7.570609e-02	-6.761350e-02	-1.909379e-02	3.193292e-02	2.206706e-02
	-1.568276e-02	0.000000e+00	-3.541052e-02	0.000000e+00	2.140476e-01
	0.000000e+00	-1.569695e-01	-1.673265e-01	9.049392e-02	1.525559e-01
	-1.159484e-01	-2.019159e-01	-1.456010e-01	-1.316736e-01	-1.017100e-01
	2.379382e-01	-1.489881e-02	0.000000e+00	1.173380e-01	5.095095e-02
49	7.530547e-02	1.687470e-01	2.376345e-01	0.000000e+00	-6.036330e-02
	1.952356e-01	-6.642060e-02	0.000000e+00	1.592915e-02	-1.114395e-01
	1.991122e-02	-5.274303e-02	6.515624e-02	-1.091807e-01	0.000000e+00
	-2.172748e-03	0.000000e+00	-3.459142e-01	3.924379e-02	-2.725987e-01
	-8.884631e-02	-1.149053e-02	2.584950e-02	0.000000e+00	-3.524360e-02
	1.065364e-01	-3.197537e-02	-4.756852e-02	1.701344e-01	1.323159e-01
	2.491510e-01	0.000000e+00	0.000000e+00	0.000000e+00	-1.772144e-02
	0.000000e+00	-2.330155e-02	0.000000e+00	-1.135144e-02	-1.216899e-01
	-1.143795e-01	0.000000e+00	2.871709e-01	0.000000e+00	-7.115103e-02
	0.000000e+00	0.000000e+00	0.000000e+00	1.568075e-01	0.000000e+00
	0.000000e+00	2.556798e-01	0.000000e+00	1.709507e-02	2.506982e-02
	3.557639e-02	4.267294e-02	-2.904683e-02	-1.604445e-01	1.020352e-01
	0.000000e+00	1.062375e-01	-2.452267e-02	-3.521116e-02	0.000000e+00
	1.115194e-02	-1.648154e-01	0.000000e+00	1.847017e-01	-1.119977e-01
	1.252549e-01	0.000000e+00	-2.320662e-01	0.000000e+00	2.082404e-01
	0.000000e+00	-2.875159e-02	7.106405e-02	2.058978e-01	0.000000e+00
50	-2.808484e-01	1.151783e-01	0.000000e+00	1.493231e-01	-2.074827e-01
	-2.246211e-01	0.000000e+00	1.139908e-01	-1.635408e-01	1.260911e-01
	2.574382e-02	-2.330054e-01	0.000000e+00	6.889942e-02	1.027037e-01

	0.000000e+00	4.798528e-02	6.168497e-02	9.232922e-03	0.000000e+00
	-1.406209e-01	0.000000e+00	0.000000e+00	2.602600e-01	0.000000e+00
	0.000000e+00	-1.876672e-01	3.622944e-02	3.746134e-02	-4.301685e-02
	-1.729882e-02	-8.212273e-02	0.000000e+00	-3.977175e-02	4.512832e-02
	2.057877e-01	2.250667e-02	2.575577e-01	0.000000e+00	4.856120e-02
	2.876634e-02	-4.430689e-02	3.721910e-02	1.644377e-01	-6.070274e-03
	0.000000e+00	7.262789e-02	0.000000e+00	1.153622e-01	0.000000e+00
	0.000000e+00	-1.421224e-01	1.004128e-01	0.000000e+00	0.000000e+00
	-1.842079e-02	0.000000e+00	0.000000e+00	-4.489239e-01	0.000000e+00
	2.960883e-02	1.725781e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	3.022778e-01	-3.488636e-01
	0.000000e+00	-9.710362e-03	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	3.459739e-02	0.000000e+00	0.000000e+00	0.000000e+00
51	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	1.068987e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	2.541975e-02	0.000000e+00	0.000000e+00	0.000000e+00	2.023061e-01
	0.000000e+00	1.877677e-01	-6.860981e-02	0.000000e+00	-1.560858e-01
	0.000000e+00	1.107794e-01	0.000000e+00	0.000000e+00	2.792157e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-2.119771e-01
	0.000000e+00	2.885011e-01	5.049332e-01	-1.136731e-01	7.216760e-02
	0.000000e+00	-4.566885e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-9.319339e-02	3.691503e-02	1.682478e-03	0.000000e+00
	3.087858e-01	1.166439e-01	-3.551694e-02	1.428851e-01	4.531432e-02
	-9.391070e-02	5.728457e-02	-1.816126e-01	0.000000e+00	-1.034354e-01
	-1.106128e-01	6.579152e-02	0.000000e+00	-1.072009e-01	0.000000e+00
	-1.370023e-01	1.587921e-01	1.341498e-01	0.000000e+00	-5.025442e-02
	0.000000e+00	-1.351815e-02	-5.507684e-02	0.000000e+00	1.469002e-01
	2.415199e-01	-2.125266e-02	0.000000e+00	-2.970062e-02	0.000000e+00
52	0.000000e+00	-8.768816e-02	1.019517e-01	8.848771e-02	0.000000e+00
	1.325257e-01	-1.548926e-01	3.569217e-02	1.948032e-02	-7.971930e-02
	-3.269670e-02	0.000000e+00	0.000000e+00	-7.299514e-02	0.000000e+00
	1.193764e-01	-2.339162e-01	7.891051e-02	-9.816863e-03	-8.211696e-02
	-6.156543e-03	-3.167585e-01	-2.840154e-01	1.418730e-01	1.161465e-01
	-3.224837e-02	3.889057e-02	5.306800e-02	0.000000e+00	-1.805380e-02
	2.198027e-01	-4.650866e-02	1.745225e-01	0.000000e+00	-2.406427e-02
	1.811940e-01	2.265921e-01	1.616894e-01	-9.129041e-02	0.000000e+00
	5.062540e-02	-1.477434e-01	8.630074e-03	1.026925e-01	-7.086020e-02
	1.674602e-01	3.634279e-01	-8.738743e-02	0.000000e+00	8.254031e-02
	1.903207e-02	1.754314e-01	0.000000e+00	1.089289e-01	6.466761e-02
	9.008273e-02	9.011894e-02	0.000000e+00	8.890025e-02	2.819083e-01
	-4.260586e-02	-4.902538e-02	0.000000e+00	-2.034516e-02	1.269190e-02
	6.804974e-02	4.119891e-02	-1.489807e-01	-3.996515e-02	-3.762686e-02
	-4.009191e-02	-4.410508e-02	1.138910e-01	-2.045999e-01	7.224963e-02
	0.000000e+00	-6.537995e-02	-5.451707e-02	0.000000e+00	2.149384e-01
53	7.637168e-02	-1.050101e-01	1.201306e-01	2.136943e-01	1.167856e-02
	-9.436246e-03	1.563580e-01	9.974492e-02	7.466304e-02	0.000000e+00
	1.045039e-01	0.000000e+00	0.000000e+00	0.000000e+00	-1.730806e-01
	2.861208e-02	-1.037862e-01	0.000000e+00	-1.203540e-01	-2.593208e-02
	1.802314e-01	0.000000e+00	-6.072117e-02	-2.547254e-01	0.000000e+00
	0.000000e+00	5.593230e-03	2.539070e-01	2.748248e-01	1.313802e-01
	-3.465358e-02	0.000000e+00	-1.853435e-01	-1.269655e-01	-1.180983e-01
	5.381035e-02	0.000000e+00	-6.067958e-02	0.000000e+00	1.089413e-02
	7.614640e-02	0.000000e+00	0.000000e+00	3.293857e-03	0.000000e+00
	-1.799225e-01	6.416696e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-3.433305e-02	0.000000e+00	3.653136e-02
	0.000000e+00	0.000000e+00	4.453992e-02	0.000000e+00	2.170820e-01
	-1.505013e-01	0.000000e+00	1.028656e-01	5.605336e-02	0.000000e+00
	0.000000e+00	0.000000e+00	7.904502e-03	0.000000e+00	-5.242339e-01
	0.000000e+00	6.014569e-02	3.502533e-01	0.000000e+00	-1.120680e-02
54	-2.929333e-01	0.000000e+00	1.187018e-01	9.216112e-02	0.000000e+00
	0.000000e+00	-1.197163e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-8.040235e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.607070e-01
	-6.887005e-02	0.000000e+00	8.054108e-02	0.000000e+00	2.094070e-01
	0.000000e+00	-8.864045e-02	2.582112e-02	0.000000e+00	0.000000e+00
	2.348643e-01	-2.011252e-01	-8.956531e-03	0.000000e+00	0.000000e+00
	0.000000e+00	2.477035e-01	0.000000e+00	2.963738e-01	1.034480e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-9.849381e-02	0.000000e+00	-1.208002e-01
	2.048512e-03	2.895734e-01	-7.842791e-02	-7.553183e-02	-7.265659e-02
	-9.780355e-02	-1.431192e-01	-2.469239e-01	0.000000e+00	-6.969870e-02
	-1.776288e-03	0.000000e+00	0.000000e+00	0.000000e+00	-4.023734e-02
	1.549295e-01	8.530009e-02	1.460983e-01	0.000000e+00	-1.073542e-01

	-2.157120e-02	-1.572345e-02	5.785071e-02	-3.126928e-01	1.718468e-02
	1.982107e-01	1.072956e-01	1.810852e-01	1.367210e-01	2.978326e-02
55	3.801447e-01	0.000000e+00	-7.510878e-02	0.000000e+00	-5.931052e-02
	-1.443186e-02	1.640274e-01	0.000000e+00	1.023705e-01	1.892272e-02
	9.443846e-02	-7.141464e-02	-5.015804e-02	6.021805e-02	0.000000e+00
	0.000000e+00	9.277559e-05	-1.810490e-02	0.000000e+00	-2.103466e-02
	0.000000e+00	1.577345e-01	-7.034821e-02	-1.086769e-01	1.401259e-01
	-5.261180e-02	7.832369e-02	-8.645146e-02	0.000000e+00	2.343270e-01
	9.574553e-02	1.941422e-02	0.000000e+00	-2.734192e-02	-1.098980e-01
	-3.727613e-02	-2.644345e-02	1.945430e-01	9.359970e-02	-1.039261e-02
	2.882850e-02	-1.619509e-01	-1.459054e-01	2.076766e-01	-6.331697e-05
	-5.946287e-02	-2.165774e-01	-1.234339e-01	-2.906369e-02	5.799169e-02
	0.000000e+00	0.000000e+00	0.000000e+00	1.657857e-02	-6.569286e-02
	0.000000e+00	3.543243e-02	0.000000e+00	0.000000e+00	1.773393e-01
	-4.152264e-02	3.835343e-02	1.230158e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	2.425674e-02	2.033859e-01
	0.000000e+00	-7.528476e-02	1.251914e-01	0.000000e+00	-1.094812e-01
	-1.489452e-01	5.289363e-02	0.000000e+00	2.389582e-01	6.981569e-02
56	2.035366e-01	0.000000e+00	0.000000e+00	-3.559894e-01	-8.186134e-02
	2.927052e-02	-6.079797e-02	-2.302594e-01	-2.344794e-01	1.659924e-01
	-1.225434e-01	9.631468e-02	0.000000e+00	-3.916048e-02	-1.301445e-01
	9.960078e-03	2.178143e-01	4.759180e-02	0.000000e+00	-9.789499e-02
	-1.092736e-01	-6.222205e-02	-1.305612e-01	1.521790e-01	-5.421362e-02
	-3.095096e-02	9.076166e-02	-1.518058e-02	0.000000e+00	-5.081929e-03
	0.000000e+00	-3.798654e-02	-1.920830e-03	0.000000e+00	-1.076431e-01
	-4.249724e-02	-6.700293e-02	-7.291976e-02	2.335329e-01	1.354690e-01
	0.000000e+00	1.080481e-01	-6.920038e-02	-1.040329e-01	3.192245e-01
	2.380990e-02	3.074836e-02	4.229492e-03	7.749490e-03	-2.531069e-01
	1.240785e-01	1.037696e-01	2.429861e-01	0.000000e+00	-7.639162e-03
	0.000000e+00	6.586251e-02	0.000000e+00	1.353089e-01	1.866795e-01
	5.852406e-02	0.000000e+00	-6.292515e-02	0.000000e+00	2.266506e-01
	0.000000e+00	1.465640e-01	6.886859e-02	3.139782e-02	0.000000e+00
	2.628456e-02	-5.364807e-02	9.342380e-02	-3.269404e-01	8.396785e-02
	-1.039547e-01	0.000000e+00	-2.823242e-01	0.000000e+00	1.198443e-01
57	9.042360e-02	5.376550e-03	-4.636283e-03	-6.466695e-02	0.000000e+00
	0.000000e+00	0.000000e+00	6.541937e-03	2.024027e-01	6.113494e-02
	-9.633359e-02	-2.458888e-02	-8.714657e-02	0.000000e+00	1.787365e-01
	-1.592383e-01	-3.812350e-02	1.376544e-02	2.638291e-01	6.209390e-02
	-1.681168e-02	0.000000e+00	0.000000e+00	-3.584104e-02	-6.124058e-02
	2.085256e-02	1.824509e-01	3.509739e-02	0.000000e+00	0.000000e+00
	-1.607996e-01	-9.170739e-02	2.048712e-02	1.729365e-01	2.897966e-02
	0.000000e+00	-2.532687e-01	8.949380e-02	2.512013e-01	1.261828e-01
	1.022691e-01	0.000000e+00	0.000000e+00	1.103641e-02	0.000000e+00
	-4.376644e-01	4.085334e-01	2.019363e-01	2.341131e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.490186e-01	-1.499640e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.716305e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	1.227526e-01	0.000000e+00	1.428378e-01	0.000000e+00	0.000000e+00
	0.000000e+00	-4.562255e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	1.667123e-01	-8.709051e-02	0.000000e+00
58	-8.896443e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-4.479540e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.360053e-02	0.000000e+00
	0.000000e+00	0.000000e+00	3.999706e-01	0.000000e+00	0.000000e+00
	-2.875428e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	2.297444e-01	0.000000e+00	0.000000e+00	1.722873e-01
	0.000000e+00	0.000000e+00	0.000000e+00	-8.495298e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	-1.673863e-01
	0.000000e+00	-7.971311e-02	0.000000e+00	0.000000e+00	0.000000e+00
	-3.442798e-02	0.000000e+00	1.028946e-01	7.003727e-02	-1.011136e-01
	-2.195787e-01	3.906484e-02	0.000000e+00	4.369234e-02	1.275274e-02
	0.000000e+00	-9.040543e-02	-9.189906e-02	3.740579e-02	1.678972e-01
	1.311885e-01	2.954210e-01	2.130871e-01	2.006178e-03	2.103851e-01
	0.000000e+00	-4.997072e-02	1.026138e-01	1.628942e-01	3.485985e-02
	1.426050e-02	4.224144e-02	-8.742522e-02	0.000000e+00	0.000000e+00
	3.105399e-02	5.636607e-02	1.123696e-02	3.649539e-01	0.000000e+00
59	2.394766e-01	1.431959e-01	-1.396422e-01	-1.210067e-01	-7.398228e-02
	-3.123550e-02	1.293987e-01	-5.742235e-02	-3.685695e-02	0.000000e+00
	-1.024555e-01	0.000000e+00	-8.263728e-02	0.000000e+00	-5.460139e-02
	-1.745019e-01	-6.388923e-02	5.985523e-02	2.413030e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-1.008974e-01	4.038258e-02
	6.962130e-02	2.869666e-02	-2.024777e-01	1.225101e-01	0.000000e+00
	1.328314e-01	8.512402e-02	0.000000e+00	1.345113e-01	9.255354e-02
	2.343800e-01	1.045466e-01	1.844225e-01	4.546188e-02	1.197661e-01
	1.296896e-01	-8.739814e-02	0.000000e+00	-6.494566e-02	1.555836e-01

	2.590561e-02	0.000000e+00	4.379955e-02	-1.683158e-02	0.000000e+00
	-2.573920e-01	5.126064e-02	3.376083e-02	1.148626e-01	-6.033851e-02
	2.989334e-03	0.000000e+00	-1.816715e-01	0.000000e+00	4.531539e-01
	1.235184e-01	-9.693836e-02	0.000000e+00	3.954533e-03	0.000000e+00
	1.175746e-01	-1.003772e-01	-1.736763e-01	1.730280e-01	5.520669e-02
	-3.856250e-02	-1.004522e-02	0.000000e+00	-5.204875e-02	0.000000e+00
	-2.599811e-02	-3.400957e-01	1.066605e-01	-1.074685e-01	-8.923043e-02
60	1.408347e-01	-6.883649e-02	2.929338e-02	-2.150650e-02	-8.601908e-02
	8.106044e-03	-1.030644e-01	-6.153358e-02	0.000000e+00	0.000000e+00
	0.000000e+00	4.100881e-02	7.273409e-02	2.294396e-01	-2.963061e-02
	4.923420e-02	5.218219e-02	-9.542798e-02	-7.479808e-02	-6.338769e-02
	9.132087e-02	-3.093529e-01	0.000000e+00	-8.100138e-03	6.343322e-02
	0.000000e+00	-2.561238e-01	0.000000e+00	6.374730e-02	1.361893e-02
	-1.017762e-02	0.000000e+00	-7.973879e-02	0.000000e+00	-4.658870e-03
	1.020018e-01	0.000000e+00	1.495400e-01	-5.910677e-02	-2.226049e-01
	5.091452e-02	0.000000e+00	6.020838e-02	0.000000e+00	-7.184980e-03
	-2.447774e-01	0.000000e+00	-2.006153e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	1.388716e-01	3.102324e-02	7.409393e-02
	0.000000e+00	0.000000e+00	0.000000e+00	-2.736213e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-4.096488e-01	-3.750778e-01
	4.241358e-01	0.000000e+00	7.067843e-02	1.439583e-01	-2.088255e-02
	1.474565e-01	0.000000e+00	4.449963e-02	0.000000e+00	6.864868e-02
	0.000000e+00	2.456015e-01	0.000000e+00	-1.452500e-01	0.000000e+00
61	0.000000e+00	-8.282694e-02	0.000000e+00	0.000000e+00	0.000000e+00
	-1.049646e-01	0.000000e+00	0.000000e+00	1.746235e-01	0.000000e+00
	0.000000e+00	0.000000e+00	9.655548e-02	0.000000e+00	0.000000e+00
	4.256880e-02	0.000000e+00	1.843919e-02	0.000000e+00	2.735261e-02
	8.984960e-02	0.000000e+00	0.000000e+00	0.000000e+00	-1.653457e-01
	-1.253492e-01	2.771518e-02	0.000000e+00	0.000000e+00	5.931080e-03
	4.474748e-02	0.000000e+00	-1.507363e-01	0.000000e+00	0.000000e+00
	1.099346e-01	0.000000e+00	0.000000e+00	-3.049195e-02	0.000000e+00
	1.168897e-01	-2.915995e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	1.571529e-01	0.000000e+00	0.000000e+00	7.074015e-02
	2.162519e-01	8.868094e-02	1.819860e-01	-1.471652e-01	-2.690856e-02
	1.904176e-01	7.735190e-02	-2.765730e-01	-4.470648e-02	1.115199e-01
	-1.076602e-01	1.109735e-01	-3.985935e-02	0.000000e+00	-6.325374e-02
	1.922875e-01	0.000000e+00	0.000000e+00	1.715637e-01	-2.574809e-02
	-8.560698e-02	0.000000e+00	-1.742907e-01	0.000000e+00	3.853224e-02
	-5.149692e-02	0.000000e+00	9.145613e-02	-2.898123e-02	-7.076332e-02
62	9.941098e-02	-1.432218e-01	0.000000e+00	0.000000e+00	-8.671431e-02
	1.113627e-01	2.413890e-01	4.370816e-02	0.000000e+00	0.000000e+00
	-1.415828e-02	-1.061231e-01	0.000000e+00	-7.569744e-02	9.233590e-02
	0.000000e+00	0.000000e+00	-9.676182e-02	8.401380e-03	-3.371450e-02
	4.858223e-02	0.000000e+00	1.568638e-01	-9.964058e-02	-1.638148e-01
	-8.849542e-02	-4.501185e-03	-3.534418e-02	-1.904148e-02	-6.261102e-02
	0.000000e+00	-9.219410e-02	-3.005882e-01	-7.090165e-03	1.292289e-01
	-3.365550e-01	5.696057e-02	1.134427e-01	1.849011e-01	-2.166282e-01
	6.930564e-02	0.000000e+00	7.119548e-02	-9.502438e-02	7.400046e-02
	-3.348881e-02	0.000000e+00	-1.197538e-01	-2.448463e-01	0.000000e+00
	8.039620e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-1.905592e-01	0.000000e+00	-3.349173e-01	-8.646562e-02	0.000000e+00
	0.000000e+00	0.000000e+00	-1.666701e-01	0.000000e+00	6.353411e-02
	0.000000e+00	0.000000e+00	0.000000e+00	-1.042109e-01	-3.026687e-01
	4.704570e-02	1.401615e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	1.633197e-01	0.000000e+00	0.000000e+00	-2.514190e-01
63	0.000000e+00	0.000000e+00	-3.923432e-01	0.000000e+00	0.000000e+00
	0.000000e+00	1.187031e-01	-1.371803e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-6.247355e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-1.550324e-01	0.000000e+00	0.000000e+00
	-1.382801e-01	0.000000e+00	0.000000e+00	0.000000e+00	-3.800917e-01
	4.077822e-02	8.123716e-02	-1.132705e-01	-1.444002e-02	0.000000e+00
	-1.123355e-01	1.224579e-01	0.000000e+00	0.000000e+00	-8.722979e-02
	0.000000e+00	9.115663e-04	0.000000e+00	0.000000e+00	-4.342650e-02
	0.000000e+00	0.000000e+00	0.000000e+00	-2.817008e-01	0.000000e+00

4. CbCelp4k

VQ codebook for stochastic excitation vector for 4kbps

/* 2nd stage VXC gain codebook cbL1_g[] */
/* dim=1 x 8 codewords */

index	codeword
0	1.44915600e+01
1	1.75510605e+02
2	7.61595276e+02
3	4.10209412e+02
4	8.47799707e+03
5	1.28569202e+03
6	3.83491211e+03
7	2.15161890e+03

/* 2nd stage VXC gain codebook cbL1_s[][] */
/* dim=40 x 32 codewords */

index	codeword			
0	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-1.077979e-01	0.000000e+00
	1.211023e-01	0.000000e+00	0.000000e+00	0.000000e+00
	1.829602e-01	0.000000e+00	0.000000e+00	0.000000e+00
	2.356981e-01	0.000000e+00	4.578350e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-2.257270e-01
	0.000000e+00	0.000000e+00	0.000000e+00	6.174449e-01
	-3.989266e-03	-9.660737e-02	1.254824e-01	0.000000e+00
	4.666770e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
1	0.000000e+00	1.063604e-01	0.000000e+00	1.689429e-01
	0.000000e+00	0.000000e+00	-2.081321e-01	1.470501e-01
	0.000000e+00	0.000000e+00	-2.697644e-02	0.000000e+00
	0.000000e+00	-6.574296e-01	0.000000e+00	4.336085e-02
	2.527332e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	4.426727e-01
	-5.108956e-01	0.000000e+00	-1.422041e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	5.066639e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
2	-2.325047e-01	0.000000e+00	0.000000e+00	2.528855e-01
	0.000000e+00	-3.280043e-01	0.000000e+00	0.000000e+00
	-1.414937e-01	4.989364e-01	1.287747e-01	-1.304628e-01
	-2.164208e-01	-1.415533e-01	-2.546594e-01	-1.460450e-01
	-7.658714e-02	-4.996565e-02	-1.968267e-01	0.000000e+00
	0.000000e+00	0.000000e+00	1.943347e-01	0.000000e+00
	-9.737150e-02	4.940040e-02	0.000000e+00	-1.936326e-01
	1.556320e-01	1.319512e-01	0.000000e+00	1.218377e-01
	-2.582947e-02	-2.183749e-01	8.117738e-03	0.000000e+00
	6.416988e-02	0.000000e+00	2.747440e-01	1.661356e-03
3	0.000000e+00	0.000000e+00	5.970257e-02	0.000000e+00
	2.224945e-02	-1.607928e-01	-1.765844e-01	-2.114129e-01
	2.041869e-01	0.000000e+00	9.252306e-02	1.590439e-01
	9.178066e-02	-2.592694e-01	1.122957e-01	1.422945e-01
	-1.132558e-01	-1.622625e-02	-6.376582e-02	-3.827269e-02
	-1.747863e-01	3.571490e-01	1.711467e-01	0.000000e+00
	1.158395e-01	5.540339e-02	0.000000e+00	0.000000e+00
	5.834910e-02	0.000000e+00	0.000000e+00	-5.355879e-02
	1.999945e-01	-1.887233e-01	1.397633e-02	-3.558268e-01
	-2.954671e-01	2.039637e-01	-3.657877e-01	1.274924e-01
4	-2.811667e-01	0.000000e+00	-2.304393e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	1.595166e-01
	3.563524e-02	8.510894e-02	0.000000e+00	0.000000e+00
	0.000000e+00	-3.142986e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-4.705483e-01	-4.308379e-01
	4.871890e-01	0.000000e+00	8.118568e-02	1.653595e-01
	-2.398701e-02	1.693778e-01	0.000000e+00	5.111507e-02
	0.000000e+00	7.885419e-02	0.000000e+00	2.821133e-01
	0.000000e+00	-1.668432e-01	0.000000e+00	0.000000e+00
	-9.514023e-02	0.000000e+00	0.000000e+00	0.000000e+00
5	0.000000e+00	0.000000e+00	5.036851e-02	-1.142099e-01
	-3.195262e-01	9.551508e-02	0.000000e+00	-1.843322e-01
	0.000000e+00	-1.178565e-01	-1.050453e-01	1.465084e-01
	2.203161e-01	-5.711720e-01	0.000000e+00	0.000000e+00
	-6.177663e-02	0.000000e+00	2.590881e-02	0.000000e+00
	-8.221275e-02	1.245975e-01	3.254338e-02	-4.447429e-02
	2.633948e-01	1.813220e-01	-9.194137e-03	9.277552e-02

	0.000000e+00	1.500300e-01	-1.064916e-01	3.640420e-02
	-1.035433e-01	2.839830e-02	-4.230123e-02	-9.371718e-02
	-6.547761e-02	2.353827e-02	-4.469144e-01	1.243046e-01
6	0.000000e+00	0.000000e+00	2.308859e-01	0.000000e+00
	0.000000e+00	0.000000e+00	1.596062e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	9.550293e-02	0.000000e+00
	7.019963e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-1.989227e-01	0.000000e+00
	-9.314497e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
7	-4.193331e-02	0.000000e+00	-1.499508e-01	-3.065865e-01
	0.000000e+00	1.006688e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-2.386105e-01	0.000000e+00
	-4.193698e-01	-1.082687e-01	0.000000e+00	0.000000e+00
	0.000000e+00	-2.086975e-01	0.000000e+00	7.955482e-02
	0.000000e+00	0.000000e+00	0.000000e+00	-1.304886e-01
	-3.789894e-01	5.890870e-02	1.755045e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	2.045022e-01
	0.000000e+00	0.000000e+00	-3.148166e-01	0.000000e+00
	0.000000e+00	-4.912761e-01	0.000000e+00	0.000000e+00
8	0.000000e+00	-4.636846e-02	0.000000e+00	-1.393200e-01
	2.213806e-02	-1.137637e-01	1.356008e-01	2.416191e-01
	-1.964287e-01	9.873721e-02	0.000000e+00	5.353699e-02
	-1.042793e-01	-1.460768e-01	9.486714e-02	0.000000e+00
	0.000000e+00	-1.892616e-01	1.232917e-02	-2.728688e-01
	4.882346e-01	-1.873749e-02	1.753556e-01	1.212834e-01
	1.517223e-01	-3.902327e-02	4.351830e-02	-2.674933e-02
	3.503422e-01	9.023787e-02	2.606275e-01	1.615115e-01
	9.475341e-02	0.000000e+00	0.000000e+00	-1.390797e-02
	1.774464e-01	0.000000e+00	-2.828480e-01	-1.688391e-01
9	2.251326e-01	-1.048434e-01	0.000000e+00	-1.153322e-01
	-1.490898e-03	-2.491165e-01	-3.487895e-01	1.377392e-01
	1.215153e-02	6.873291e-02	-6.127291e-02	0.000000e+00
	2.845392e-02	9.337220e-02	1.079441e-01	-6.488267e-02
	2.988443e-01	1.824093e-01	1.823725e-01	-3.078499e-01
	1.388035e-01	0.000000e+00	-8.503559e-02	-6.056299e-02
	0.000000e+00	-2.487390e-02	0.000000e+00	0.000000e+00
	7.836885e-03	5.766903e-01	-9.002327e-02	-1.891632e-01
	0.000000e+00	0.000000e+00	8.263382e-03	-1.073881e-01
	-1.835672e-02	0.000000e+00	-1.337245e-01	2.412605e-02
10	0.000000e+00	0.000000e+00	-2.559755e-01	0.000000e+00
	1.760823e-01	2.212395e-01	9.682013e-02	5.862363e-02
	-1.315140e-01	1.832536e-01	-4.707699e-02	-3.725728e-03
	1.791800e-02	1.357346e-01	-2.807897e-02	1.982710e-01
	1.584873e-01	1.624019e-01	4.151292e-01	-6.851899e-03
	-3.089290e-01	3.385574e-02	-2.562297e-01	-7.474697e-02
	0.000000e+00	9.786504e-02	1.476423e-01	5.803956e-03
	-2.283477e-01	2.862185e-02	-2.439912e-01	-1.317321e-01
	-1.472580e-01	-1.100098e-01	1.419855e-01	8.417646e-02
	0.000000e+00	-2.564971e-01	-2.205492e-01	-3.664943e-02
11	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	4.332744e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	1.909482e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	-3.374553e-01	-1.815526e-01	-1.278013e-01
	0.000000e+00	0.000000e+00	-4.429387e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	-2.213892e-01	6.865721e-01
	0.000000e+00	0.000000e+00	3.004608e-01	0.000000e+00
	0.000000e+00	0.000000e+00	2.907348e-03	0.000000e+00
12	-1.754829e-01	2.076649e-01	0.000000e+00	-1.282256e-01
	3.720572e-01	-1.296803e-01	-1.733985e-01	-6.079009e-02
	-2.060229e-02	-7.371581e-03	2.123673e-01	7.181562e-02
	7.439682e-02	2.706323e-01	7.951257e-02	1.697801e-01
	-9.110906e-03	9.530824e-02	0.000000e+00	6.746141e-02
	-8.797239e-02	-2.456894e-01	1.197566e-01	3.669889e-01
	-7.477165e-02	5.248059e-02	0.000000e+00	1.700427e-01
	1.830603e-01	2.506950e-01	-2.835853e-01	2.005359e-01
	-3.427051e-02	-1.640932e-02	1.653971e-02	8.867367e-02
	0.000000e+00	2.282165e-01	8.986981e-02	3.371461e-02
	-2.259322e-01	8.057571e-02	0.000000e+00	0.000000e+00

	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-4.311268e-02	0.000000e+00	4.587314e-02	0.000000e+00
13	0.000000e+00	5.592965e-02	0.000000e+00	2.725941e-01
	-1.889875e-01	0.000000e+00	1.291704e-01	7.038731e-02
	0.000000e+00	0.000000e+00	0.000000e+00	9.925838e-03
	0.000000e+00	-6.582908e-01	0.000000e+00	7.552613e-02
	4.398200e-01	0.000000e+00	-1.407260e-02	-3.678421e-01
	0.000000e+00	1.490562e-01	1.157285e-01	0.000000e+00
	0.000000e+00	-2.403559e-01	-8.150406e-02	-1.325993e-01
	0.000000e+00	0.000000e+00	9.741563e-02	0.000000e+00
	-8.101166e-02	-4.820152e-02	-1.694508e-01	0.000000e+00
	0.000000e+00	1.923893e-01	0.000000e+00	-1.381291e-01
14	-2.299034e-02	1.171245e-01	2.367346e-01	4.192760e-01
	6.838050e-02	1.507127e-02	1.301064e-02	-9.606195e-02
	-2.410322e-01	0.000000e+00	2.243365e-02	-3.361835e-01
	3.248497e-01	-3.080189e-01	3.070857e-02	1.992121e-01
	2.252643e-01	1.279014e-02	2.179674e-01	-2.418915e-02
	-2.045404e-01	0.000000e+00	3.079529e-02	0.000000e+00
	-4.317952e-02	-2.129793e-01	-8.415943e-02	1.494561e-01
	1.233681e-01	-1.046373e-01	2.255625e-02	-2.213613e-01
	1.999735e-01	-7.684381e-02	-1.466436e-01	6.581554e-03
	-1.590079e-02	2.034994e-01	-6.528557e-02	-4.110763e-03
15	1.131641e-01	3.297746e-01	1.551521e-03	-1.691152e-01
	2.672528e-01	2.408254e-01	0.000000e+00	2.002690e-01
	0.000000e+00	1.049898e-01	1.536277e-01	-7.974234e-03
	0.000000e+00	2.088439e-01	1.182210e-01	-1.987692e-01
	0.000000e+00	-7.916138e-02	-2.567744e-01	-2.633259e-01
	-1.871229e-01	2.055860e-01	0.000000e+00	2.529685e-01
	0.000000e+00	0.000000e+00	0.000000e+00	9.325930e-02
	0.000000e+00	-5.525424e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	8.280446e-02
16	0.000000e+00	0.000000e+00	0.000000e+00	-5.074697e-02
	0.000000e+00	-3.361290e-01	3.110393e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-3.111891e-01	2.425379e-02	0.000000e+00	0.000000e+00
	-6.044125e-01	0.000000e+00	0.000000e+00	-6.401744e-02
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	4.070788e-01	-3.843808e-01	0.000000e+00	-3.721019e-02
	1.066112e-01	0.000000e+00	5.160967e-02	-1.044038e-01
	8.513179e-02	1.998530e-01	0.000000e+00	-2.718021e-01
	7.827752e-02	1.203296e-01	5.884977e-02	0.000000e+00
17	1.362593e-01	-1.081944e-02	1.015380e-01	-6.308033e-02
	-1.439714e-01	-1.516670e-01	2.869562e-01	-7.925501e-02
	2.501642e-03	1.820601e-02	-4.470779e-01	6.120597e-02
	-5.494761e-02	1.677305e-02	2.115798e-02	1.873790e-01
	-1.396222e-01	1.963782e-03	0.000000e+00	8.787096e-02
	8.510151e-02	0.000000e+00	1.616764e-01	-2.152618e-01
	0.000000e+00	-4.047184e-01	0.000000e+00	0.000000e+00
	-2.335877e-01	-1.178735e-01	2.847720e-04	-1.152904e-01
	3.108173e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
18	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	2.172906e-02	0.000000e+00	0.000000e+00	-7.395102e-01
	0.000000e+00	5.108694e-02	0.000000e+00	0.000000e+00
	-2.507887e-01	0.000000e+00	1.797698e-03	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	2.114873e-01	0.000000e+00	-1.623986e-02	0.000000e+00
	1.285463e-01	3.211446e-01	2.784739e-01	-3.962589e-01
	-6.134347e-02	0.000000e+00	-5.190680e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	9.733866e-02	-9.864979e-02	0.000000e+00
19	-1.071804e-01	-3.379402e-01	-1.815525e-01	0.000000e+00
	-1.703403e-01	-5.950190e-01	0.000000e+00	-2.245885e-01
	0.000000e+00	0.000000e+00	-1.007124e-01	-6.349786e-02
	0.000000e+00	0.000000e+00	-7.158682e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-6.679170e-02
	0.000000e+00	6.474974e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	3.210638e-01
	0.000000e+00	-1.486184e-01	0.000000e+00	0.000000e+00
	5.416056e-01	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
20	0.000000e+00	0.000000e+00	-1.225297e-01	0.000000e+00

	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	3.614725e-02	0.000000e+00	0.000000e+00
	-6.714751e-02	-1.192343e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	4.555875e-02	1.209992e-01
	2.229084e-01	4.343440e-01	-5.398043e-01	0.000000e+00
	1.233227e-01	1.458409e-01	0.000000e+00	1.557261e-01
	-1.274834e-01	1.431187e-01	-1.277092e-01	-5.357691e-02
	0.000000e+00	-1.460517e-01	-1.092227e-01	-9.754737e-02
	-2.754700e-02	4.607027e-02	3.183660e-02	-2.262584e-02
21	0.000000e+00	-5.108747e-02	0.000000e+00	3.088107e-01
	0.000000e+00	-2.264631e-01	-2.414053e-01	1.305574e-01
	2.200954e-01	-1.672812e-01	-2.913082e-01	-2.100615e-01
	-1.899681e-01	-1.467390e-01	3.432782e-01	-2.149482e-02
	0.000000e+00	1.692858e-01	7.350796e-02	1.086447e-01
	2.434547e-01	3.428400e-01	0.000000e+00	-8.708735e-02
	0.000000e+00	9.343626e-02	0.000000e+00	0.000000e+00
	7.132953e-02	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	8.078488e-02	0.000000e+00	-4.262873e-01
	0.000000e+00	2.515062e-01	-3.590057e-01	0.000000e+00
22	-8.476058e-02	0.000000e+00	2.043080e-01	0.000000e+00
	-5.057540e-02	0.000000e+00	0.000000e+00	8.285139e-02
	-3.546542e-01	0.000000e+00	0.000000e+00	-1.953130e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	6.021874e-01
	-7.817402e-02	0.000000e+00	-1.215660e-01	0.000000e+00
	0.000000e+00	-1.470843e-01	5.826187e-02	2.655403e-03
	0.000000e+00	4.873471e-01	1.840955e-01	-5.605531e-02
	2.255112e-01	7.151821e-02	-1.482164e-01	9.041049e-02
	-2.866337e-01	0.000000e+00	-1.632490e-01	-1.745768e-01
23	1.038367e-01	0.000000e+00	-1.691919e-01	0.000000e+00
	-2.162266e-01	2.506167e-01	2.117246e-01	0.000000e+00
	-7.931501e-02	0.000000e+00	-2.133528e-02	-8.692609e-02
	0.000000e+00	2.318481e-01	3.811834e-01	-3.354242e-02
	0.000000e+00	-4.687558e-02	0.000000e+00	0.000000e+00
	-1.383955e-01	1.609073e-01	1.396574e-01	0.000000e+00
	-5.509263e-01	5.142565e-01	2.541948e-01	2.946985e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-1.875827e-01	-1.887728e-02	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	3.419250e-01	0.000000e+00
24	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	1.545193e-01	0.000000e+00	1.798024e-01	0.000000e+00
	0.000000e+00	0.000000e+00	-5.742907e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	2.098553e-01	-1.096284e-01	0.000000e+00	-1.119873e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-7.672629e-02	1.561450e-01	2.802804e-02	2.072605e-01
	-3.429807e-01	-8.871547e-03	-1.055646e-01	3.114361e-02
	1.934104e-01	1.881531e-01	1.357228e-02	-1.940391e-01
	4.175462e-01	1.663702e-02	-1.364723e-01	1.946154e-02
25	-3.671429e-01	1.398991e-01	0.000000e+00	-7.816728e-02
	-1.573523e-01	0.000000e+00	-1.163459e-01	0.000000e+00
	0.000000e+00	-1.043748e-01	0.000000e+00	-2.428079e-01
	-1.022101e-02	1.282556e-01	1.683070e-01	2.576395e-01
	-1.038674e-01	0.000000e+00	-7.635974e-02	-1.916149e-01
	5.731683e-04	1.384166e-01	-1.987855e-01	0.000000e+00
	1.760847e-01	3.249262e-01	-1.152977e-01	-3.927576e-01
	0.000000e+00	-1.389606e-01	-3.237636e-02	1.691569e-01
	-5.510974e-02	0.000000e+00	3.080531e-01	1.551662e-02
	1.887064e-01	-7.835308e-02	-1.254886e-01	8.588114e-02
26	0.000000e+00	-5.006174e-02	1.052845e-01	0.000000e+00
	0.000000e+00	4.125165e-01	-1.631258e-02	0.000000e+00
	-1.773245e-01	-1.925615e-01	1.553396e-01	-4.925433e-02
	1.116559e-01	3.050258e-02	0.000000e+00	-2.219004e-01
	-2.584255e-01	0.000000e+00	0.000000e+00	-1.350262e-01
	5.189991e-02	0.000000e+00	-2.896770e-02	-2.314153e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-3.035030e-01	0.000000e+00	0.000000e+00	0.000000e+00
	-3.861304e-01	-6.062269e-01	0.000000e+00	0.000000e+00
27	0.000000e+00	0.000000e+00	0.000000e+00	-2.559641e-01
	0.000000e+00	0.000000e+00	2.148703e-01	0.000000e+00
	3.961714e-01	0.000000e+00	0.000000e+00	0.000000e+00
	3.278811e-01	0.000000e+00	-1.137058e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00

	-4.695378e-02	0.000000e+00	0.000000e+00	0.000000e+00
28	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-1.938826e-01	-2.661398e-01	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	-4.956191e-01
	0.000000e+00	0.000000e+00	-1.321518e-01	0.000000e+00
	0.000000e+00	-5.917736e-02	0.000000e+00	0.000000e+00
	0.000000e+00	1.072359e-01	0.000000e+00	1.456759e-01
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	2.299137e-01	4.915662e-01	0.000000e+00
	-2.331184e-01	0.000000e+00	4.690884e-01	1.481871e-01
	0.000000e+00	0.000000e+00	-3.753161e-02	0.000000e+00
29	-4.356876e-02	0.000000e+00	1.302136e-01	8.863248e-02
	-1.279597e-01	-2.778779e-01	4.943673e-02	0.000000e+00
	5.529285e-02	1.613864e-02	0.000000e+00	-1.144085e-01
	-1.162987e-01	4.733719e-02	2.124746e-01	1.660197e-01
	3.738566e-01	2.696627e-01	2.538827e-03	2.662433e-01
	0.000000e+00	-6.323817e-02	1.298582e-01	2.061434e-01
	4.411530e-02	1.804672e-02	5.345674e-02	-1.106370e-01
	0.000000e+00	0.000000e+00	3.929896e-02	7.133152e-02
	1.422042e-02	4.618508e-01	0.000000e+00	3.030587e-01
	1.812150e-01	-1.767178e-01	-1.531345e-01	-9.362490e-02
30	3.246434e-02	4.192480e-02	5.766831e-03	1.056628e-02
	-3.451064e-01	1.691787e-01	1.414878e-01	3.313068e-01
	0.000000e+00	-1.041585e-02	0.000000e+00	8.980227e-02
	0.000000e+00	1.844911e-01	2.545339e-01	7.979643e-02
	0.000000e+00	-8.579723e-02	0.000000e+00	3.090338e-01
	0.000000e+00	1.998373e-01	9.390101e-02	4.281033e-02
	0.000000e+00	3.583849e-02	-7.314811e-02	1.273816e-01
	-4.457769e-01	1.144885e-01	-1.417402e-01	0.000000e+00
	-3.849437e-01	0.000000e+00	1.634054e-01	1.232908e-01
	7.330823e-03	-6.321483e-03	-8.817215e-02	0.000000e+00
31	0.000000e+00	7.456178e-02	0.000000e+00	0.000000e+00
	0.000000e+00	3.067416e-01	-3.060968e-02	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	0.000000e+00	0.000000e+00	3.527414e-01	0.000000e+00
	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00
	-4.115891e-02	-1.271116e-01	0.000000e+00	0.000000e+00
	0.000000e+00	1.858140e-01	3.673089e-01	-8.304609e-02
	9.817926e-02	0.000000e+00	5.266669e-01	0.000000e+00
	0.000000e+00	0.000000e+00	3.578682e-01	0.000000e+00
	0.000000e+00	-4.070740e-01	0.000000e+00	0.000000e+00

5. CbLsp

LSP table for the first stage lsp_tbl[32*10]

index	codeword									
0	0.10300	0.14989	0.19987	0.27587	0.33391	0.40516	0.57367	0.65024	0.77318	0.87194
1	0.09076	0.13299	0.18705	0.25552	0.30774	0.37970	0.58437	0.70786	0.80028	0.88011
2	0.09389	0.13307	0.18760	0.29758	0.36841	0.43454	0.51052	0.59234	0.77804	0.86998
3	0.08907	0.12412	0.20053	0.27616	0.34487	0.40211	0.47443	0.66503	0.78114	0.87761
4	0.12008	0.18370	0.28088	0.37154	0.47208	0.56008	0.66317	0.73985	0.83849	0.88920
5	0.11431	0.17987	0.25469	0.33435	0.42930	0.50131	0.60974	0.69642	0.79900	0.86632
6	0.09577	0.14741	0.20412	0.29639	0.39958	0.45178	0.53964	0.61910	0.72064	0.85017
7	0.11169	0.17043	0.25658	0.34485	0.40996	0.46039	0.56135	0.67018	0.77457	0.85167
8	0.07769	0.10613	0.16178	0.35089	0.42281	0.48053	0.57963	0.65581	0.80152	0.86692
9	0.07371	0.09886	0.16971	0.39296	0.47935	0.53062	0.62109	0.68010	0.80375	0.86421
10	0.08809	0.15716	0.25875	0.34753	0.44291	0.52739	0.63654	0.72092	0.82270	0.88909
11	0.06801	0.12702	0.24257	0.35017	0.44592	0.51850	0.60995	0.69012	0.78708	0.86515
12	0.06367	0.12001	0.25644	0.37933	0.49027	0.58582	0.67909	0.75639	0.83812	0.89306
13	0.10801	0.18154	0.30436	0.41583	0.52348	0.62326	0.70630	0.77381	0.85497	0.90314
14	0.11438	0.18859	0.30436	0.38954	0.46172	0.52796	0.62960	0.70330	0.80446	0.87445
15	0.12221	0.20923	0.36796	0.44375	0.52931	0.60860	0.68012	0.74087	0.81860	0.87096
16	0.09030	0.14959	0.20668	0.27770	0.39582	0.43787	0.53805	0.69180	0.75736	0.83915
17	0.05064	0.09899	0.20415	0.30282	0.40418	0.48828	0.59750	0.69367	0.79709	0.87791
18	0.06459	0.08788	0.15133	0.32602	0.49708	0.57248	0.66942	0.74076	0.82696	0.88751
19	0.04952	0.09101	0.21602	0.32517	0.42791	0.52316	0.62468	0.71228	0.81302	0.88566
20	0.09014	0.12380	0.17228	0.28648	0.46364	0.51743	0.59630	0.68287	0.75559	0.84323
21	0.05582	0.07871	0.12661	0.27763	0.42691	0.49969	0.60141	0.68778	0.80612	0.89135

22	0.07310	0.10448	0.15484	0.25617	0.46444	0.56881	0.63116	0.70035	0.77251	0.84411
23	0.06577	0.08644	0.13718	0.23731	0.42461	0.56929	0.67376	0.73255	0.81938	0.87502
24	0.08707	0.14861	0.21890	0.31565	0.37331	0.45307	0.59975	0.67993	0.80774	0.87807
25	0.09009	0.12975	0.19984	0.27136	0.33292	0.50954	0.59597	0.66668	0.76876	0.84146
26	0.07441	0.10490	0.16399	0.23062	0.28810	0.45924	0.58244	0.68239	0.80383	0.88572
27	0.08376	0.11414	0.17907	0.24511	0.31428	0.51658	0.64531	0.71371	0.81804	0.86778
28	0.06751	0.09927	0.16469	0.27561	0.40987	0.48742	0.58012	0.62826	0.73967	0.87049
29	0.07101	0.10586	0.15943	0.23689	0.41370	0.46858	0.55855	0.68507	0.76958	0.86548
30	0.06451	0.09541	0.15034	0.28238	0.35920	0.43263	0.58271	0.66821	0.78879	0.88441
31	0.05728	0.07966	0.13123	0.22471	0.37678	0.49222	0.60355	0.69238	0.79404	0.89280

LSP table for the second stage of VQ with inter-frame prediction pd_tbl[5*64+5*16] ....Lower 5 LSPs

index	codeword				
0	0.01137	0.00753	0.01401	0.00793	0.01203
1	0.00178	0.00449	0.01001	0.01646	-0.01133
2	0.02748	-0.00897	-0.00086	0.00972	0.00684
3	0.01202	0.00572	0.00631	0.02909	0.03414
4	0.00563	0.01050	-0.00997	0.01944	0.01603
5	0.00809	0.01586	0.01928	-0.01774	-0.01602
6	0.01407	0.01965	0.00225	0.02852	-0.01604
7	-0.00166	0.00424	0.00382	-0.00098	0.01344
8	0.00076	0.00747	-0.01093	-0.00711	0.01736
9	-0.01009	-0.01480	0.00674	0.02374	0.01142
10	0.00871	0.02094	0.01662	0.00157	-0.02691
11	-0.00391	0.00255	-0.00069	0.01747	0.00272
12	0.00193	-0.00153	-0.01372	0.03399	0.03747
13	-0.00518	-0.00148	0.00238	0.01327	0.02242
14	-0.00191	-0.00338	-0.00193	0.03930	0.01529
15	-0.01823	0.01401	0.00233	0.01221	0.01209
16	0.00755	0.01369	0.00297	0.00537	0.02886
17	0.00812	0.02379	0.00909	0.01989	0.00644
18	0.02272	0.04214	-0.00952	0.00638	-0.00662
19	0.00751	0.00581	0.00728	0.02917	0.00562
20	0.02117	0.03542	0.02863	0.02455	0.02190
21	0.02425	0.05339	0.02422	-0.00792	-0.00893
22	0.02980	0.01727	0.00506	0.02436	0.00842
23	0.01371	0.01883	-0.01121	0.00581	0.00110
24	0.00288	0.00629	0.01364	-0.00462	-0.00153
25	-0.00091	-0.00530	-0.00832	0.00292	0.03010
26	0.00617	0.00325	0.01457	0.05868	0.00035
27	0.00302	-0.00481	0.00937	0.01367	0.00419
28	0.00601	0.00438	0.01518	0.02746	-0.03100
29	0.00790	0.00463	-0.00219	0.00414	0.00117
30	0.01296	0.00191	0.01714	0.00432	-0.01262
31	-0.00322	0.00303	0.02576	0.03290	-0.00608
32	0.00168	-0.00699	0.01356	-0.00678	0.00938
33	0.00304	-0.00438	-0.00357	0.00590	0.01328
34	0.01270	0.00182	-0.00818	0.02025	-0.01017
35	0.00057	-0.01619	0.03075	0.01332	0.00282
36	0.00326	0.00948	0.03010	-0.02649	0.00154
37	0.01587	0.02342	-0.01832	-0.01954	0.01335
38	0.00243	0.00653	0.02578	0.02927	0.01819
39	-0.00024	-0.01052	0.00668	0.03235	-0.00889
40	-0.01009	-0.01427	0.01049	-0.00086	0.02918
41	-0.00771	-0.01567	0.02398	0.04490	0.02703
42	-0.00003	0.01490	0.00500	0.00604	-0.00731
43	-0.01169	-0.00799	-0.00548	0.01787	0.04018
44	-0.00636	0.02574	0.03337	0.01051	-0.00518
45	-0.00023	0.00379	0.01916	0.01029	0.03609
46	-0.00989	-0.00046	0.01603	0.00867	0.00294
47	-0.01679	-0.00276	0.04931	0.01506	0.01310
48	0.00486	0.01702	0.00341	-0.00872	0.00463
49	0.01731	0.03487	-0.00412	0.00367	0.02471
50	0.02062	0.00744	0.00382	-0.01547	0.00210
51	0.00532	0.01751	0.06412	0.02532	0.00391
52	0.01731	0.02385	0.02296	-0.00096	0.01089
53	0.03421	0.02161	0.00489	-0.00076	-0.01064

54	0.01476	0.01416	0.02477	0.01874	-0.00356
55	0.01543	0.01035	0.00167	0.00116	-0.01215
56	-0.00425	0.01154	0.02399	-0.00140	0.01388
57	0.00346	-0.00645	-0.01219	0.02079	0.00860
58	0.00500	0.01038	-0.01428	0.00210	0.04960
59	-0.00411	0.03968	0.00788	-0.00152	0.00346
60	0.00462	0.00407	0.00676	-0.01988	0.01946
61	0.01508	0.00025	-0.01273	-0.00522	0.01985
62	0.00284	0.00240	0.00411	-0.00796	0.04516
63	-0.00116	-0.00114	0.03146	-0.00158	-0.01357

LSP table for the second stage of VQ with inter-frame prediction pd_tbl[5*64+5*16] ....Higher 5 LSPs

index	codeword				
0	0.03633	0.00132	-0.01137	0.00102	-0.00128
1	0.01003	-0.02086	-0.00661	0.02710	0.00942
2	0.00891	0.01267	-0.01112	0.02438	0.00917
3	-0.01511	-0.00141	0.00619	0.03746	0.01660
4	0.03757	0.02319	0.01311	0.02730	0.00974
5	0.01368	0.00288	0.00739	-0.00495	0.00965
6	0.01999	0.03368	-0.00238	-0.00474	-0.00177
7	-0.00848	0.01428	-0.00012	0.00644	0.02067
8	0.02299	-0.00864	0.01004	0.01426	-0.00628
9	-0.00302	-0.01473	0.02270	0.01032	-0.00061
10	0.00256	0.01006	0.00555	0.00464	-0.01039
11	-0.01654	0.01022	0.02937	0.00594	-0.00071
12	0.03618	0.01049	0.01452	-0.01096	-0.01551
13	0.01165	0.01624	0.03265	0.02361	0.00584
14	0.00733	0.01799	0.03020	-0.01505	-0.00906
15	-0.00336	0.04025	0.01523	0.01452	0.00800

LSP table for the second stage of VQ without inter-frame prediction d_tbl[5*64+5*16] ...Lower 5 LSPs

index	codeword				
0	-0.00085	0.00588	0.02161	0.01369	0.01759
1	-0.00661	0.02041	0.02040	-0.00935	0.01107
2	0.01665	0.00647	-0.00596	0.04066	-0.00901
3	0.01127	0.01409	0.03237	0.00486	0.00602
4	0.01742	0.02792	0.01048	0.00648	0.03164
5	-0.00018	0.00805	0.00248	0.00826	0.00375
6	0.02702	0.03438	-0.00978	-0.03154	0.01303
7	0.01518	0.00106	0.01720	0.01115	-0.00581
8	-0.01025	-0.01590	0.03903	0.02052	0.01042
9	-0.00484	-0.00882	0.06051	0.01705	-0.02246
10	0.00735	0.00146	0.03771	0.04057	0.00039
11	0.01281	0.02130	0.00074	0.01862	-0.03525
12	-0.00011	-0.00031	-0.00173	0.01444	0.03040
13	0.01428	0.02660	0.02892	-0.02538	0.00243
14	0.01339	0.01471	0.03287	0.01788	-0.02441
15	0.03830	0.03646	0.02013	0.00861	0.00005
16	-0.01281	0.01086	0.03773	0.01244	-0.00445
17	-0.02453	-0.00722	0.03763	0.04487	0.02624
18	-0.00073	0.00697	0.00787	0.03030	-0.01250
19	-0.00726	-0.01199	0.02102	-0.00170	0.02755
20	-0.00006	-0.00844	-0.01507	0.03910	0.01830
21	-0.01788	0.00209	0.00791	0.02746	0.01490
22	0.00791	0.00662	-0.00440	0.03861	0.04938
23	-0.00144	-0.00160	0.02361	0.03935	-0.03237
24	-0.02698	0.00446	0.01271	0.00973	0.03441
25	-0.01884	-0.03063	0.06711	0.04675	0.00268
26	-0.00978	0.02594	0.00772	0.01774	0.00027
27	-0.00556	0.02292	0.05214	0.02281	0.02260
28	-0.01224	-0.02077	-0.01295	0.05391	0.02916
29	-0.00262	0.02742	0.03289	-0.00799	-0.02515
30	-0.00348	0.00766	-0.02139	0.02563	0.01186
31	0.01340	0.02502	0.02640	0.03146	0.00707
32	0.01781	0.06670	0.03869	-0.00752	-0.01057
33	0.01196	0.02115	-0.00364	-0.03600	0.03278

34	0.04018	0.00048	-0.02473	-0.01329	0.00160
35	0.03160	0.01414	-0.00404	0.01348	-0.01119
36	0.04114	0.04938	-0.00435	-0.03144	-0.03398
37	0.02036	-0.00034	-0.01480	-0.00154	0.03298
38	0.02476	0.02556	-0.00407	-0.00840	0.00581
39	0.00948	-0.00607	0.01187	-0.01058	0.01704
40	0.00658	0.00958	0.02822	0.00586	0.04568
41	0.01209	0.01931	0.00338	0.02664	0.01417
42	0.02620	0.01534	0.01419	-0.01191	-0.02375
43	0.03212	0.03200	-0.03069	0.00280	0.00123
44	0.02186	0.00836	0.00604	0.01086	0.01918
45	0.02532	-0.01663	0.00406	0.02398	0.01014
46	0.00502	0.01099	0.01271	-0.00786	-0.01332
47	0.01314	0.02553	0.00945	0.00125	-0.00706
48	0.00171	-0.01210	0.01809	0.01470	0.00290
49	-0.00675	-0.00477	-0.02453	0.00802	0.05097
50	0.01574	0.00254	-0.01118	0.00128	0.00182
51	0.01183	0.01195	0.00648	-0.01775	0.01287
52	0.02227	0.00380	-0.03080	0.02376	0.01403
53	0.00125	0.00112	0.03293	-0.00211	-0.01646
54	0.00322	-0.00056	0.01870	0.03877	0.02745
55	-0.00823	0.00313	-0.00336	-0.01107	0.02738
56	-0.01054	-0.02156	0.00750	0.02332	0.03266
57	-0.00550	-0.01639	0.00565	0.03429	0.00147
58	0.00636	0.01084	-0.00243	-0.01312	0.05153
59	0.01254	0.01900	-0.03292	0.00851	0.03883
60	0.00129	0.00829	0.03862	-0.02956	0.01259
61	0.00458	0.02098	-0.01340	0.00285	0.01500
62	-0.01159	0.04855	0.01077	-0.01779	0.01000
63	-0.00525	-0.00568	0.00242	0.05888	-0.00739

LSP table for the second stage of VQ without inter-frame prediction d_tbl[5*64+5*16] ...Higher 5 LSPs

index	codeword				
0	0.03767	-0.00950	0.01454	0.00361	-0.00926
1	0.01192	-0.01155	0.02772	0.03391	0.00364
2	0.04425	0.02244	0.00908	-0.01321	-0.01666
3	0.01098	0.03216	0.03717	-0.01401	-0.01775
4	0.01297	0.00672	-0.00077	-0.00145	-0.00403
5	-0.02429	0.00960	0.02993	0.00864	-0.00022
6	0.03357	0.03846	0.02941	0.02699	0.00967
7	0.00878	0.02304	0.01422	0.02662	0.00670
8	0.00920	-0.01319	-0.01053	0.03995	0.01758
9	-0.02687	-0.00829	0.01519	0.04286	0.03036
10	0.04248	0.00641	-0.01687	0.02205	0.00791
11	0.00529	0.00368	0.01743	0.00306	0.00994
12	-0.00655	0.02008	-0.01495	0.02361	0.02189
13	-0.00965	0.03087	0.04685	0.03672	0.01655
14	0.01991	0.03615	-0.01073	-0.00325	0.00499
15	-0.01679	0.04588	0.01319	-0.00479	0.00709

6. CbLsp4k

VQ codebook for LSP quantization of enhancement layer for 4kbps.

/* dim=10 x 256 codevectors */

index	codeword			
0	3.652737e-03f -4.910919e-03f 2.120240e-03f	-3.723557e-03f -9.568315e-03f 1.168347e-02f	-2.935357e-03f -9.355669e-03f	1.266268e-02f 7.896985e-04f
1	3.007427e-03f 1.270376e-02f -4.784224e-02f	8.423534e-03f -5.538360e-03f -1.918005e-02f	5.517199e-03f -1.415260e-03f	3.321410e-02f -1.938517e-02f
2	5.082027e-03f -7.855622e-03f	6.259896e-03f -3.202555e-03f	4.261945e-03f 8.309573e-03f	5.254921e-03f 1.073399e-02f

	-9.154360e-04f	1.514070e-03f		
3	-5.990481e-03f -6.251054e-04f 3.939836e-03f	8.417413e-03f -3.478267e-03f -1.563623e-03f	2.724798e-03f -2.528251e-03f	1.489685e-02f -7.314433e-04f
4	5.453609e-03f 1.413430e-03f -5.866571e-03f	1.500243e-03f -1.628425e-03f -5.111536e-03f	8.433697e-04f 4.847159e-03f	3.889741e-03f -1.033116e-02f
5	5.567431e-03f -1.655188e-04f 7.247546e-03f	-6.411438e-03f 1.501932e-03f -6.823328e-04f	-3.294032e-03f -7.187506e-03f	1.761921e-02f -6.440210e-04f
6	9.241253e-03f -1.345458e-02f -1.236911e-02f	1.572035e-02f -2.643184e-02f -8.403739e-03f	1.900817e-02f -1.098501e-03f	7.498425e-03f -1.513230e-02f
7	1.167794e-02f -3.290477e-03f -3.400779e-03f	1.325267e-03f 5.998920e-03f -5.581064e-03f	4.260662e-03f -1.524323e-03f	1.386645e-03f 1.476417e-03f
8	5.356349e-03f 2.572577e-03f -2.693594e-02f	4.341042e-04f 2.274942e-03f 1.134504e-02f	-6.217861e-03f -1.852000e-03f	7.929876e-03f -1.107384e-03f
9	-2.298185e-03f 4.693033e-03f -5.585823e-03f	6.392882e-04f -2.074355e-03f -4.520547e-03f	-5.831057e-03f -4.913450e-03f	1.256655e-02f 1.085111e-02f
10	-3.485705e-03f 2.480264e-03f -2.815041e-02f	-2.991108e-03f -8.928425e-03f -1.238459e-02f	1.572291e-03f 1.525912e-02f	9.020573e-03f 1.770989e-02f
11	-1.898336e-02f 1.188616e-03f 2.102770e-02f	-1.177026e-03f -5.415046e-03f -7.912045e-03f	-4.034785e-03f 2.086671e-02f	6.197917e-03f 3.667164e-02f
12	3.346877e-04f 2.590174e-02f -6.698537e-03f	9.263028e-03f 9.555846e-03f -1.552193e-02f	-1.252652e-02f -3.693220e-03f	1.083689e-02f 3.538526e-03f
13	2.580827e-03f 2.120919e-02f -1.890209e-02f	-1.120983e-02f 9.698965e-04f 3.935190e-03f	-3.055851e-02f -1.340943e-03f	2.361929e-02f -1.060086e-02f
14	-3.168637e-03f 9.510157e-04f -9.232963e-03f	1.079626e-02f 5.960308e-03f -1.126543e-02f	8.896769e-03f 4.869651e-03f	2.163695e-03f 4.237728e-04f
15	-9.698352e-03f 6.001916e-03f 1.639849e-03f	-1.749234e-03f -1.087748e-02f -5.235735e-03f	-1.471459e-02f -8.160731e-03f	7.161541e-03f -2.751539e-03f
16	-8.032992e-04f 5.303411e-03f -2.453878e-03f	-3.852987e-03f 4.973330e-03f 6.407294e-03f	-2.037228e-04f -5.176039e-03f	8.652086e-03f -1.573176e-02f
17	-6.964818e-03f 2.121836e-03f -2.383874e-02f	4.619313e-03f -1.569262e-02f -3.492002e-03f	1.125945e-02f 4.249239e-03f	1.152072e-02f -1.298962e-02f
18	2.851867e-03f 1.081158e-02f 9.741174e-04f	-1.200155e-03f -3.773540e-03f 1.647823e-02f	-3.421347e-03f 3.143326e-03f	-2.188925e-04f 8.471098e-03f
19	-4.212699e-03f 6.155124e-03f 9.057648e-03f	-4.123807e-04f 9.659707e-03f 2.677987e-03f	6.566235e-03f 4.326944e-03f	-2.653321e-04f -9.567866e-03f
20	2.439607e-03f 1.947496e-03f -2.510375e-03f	-6.613013e-03f -9.687494e-03f 1.163572e-02f	1.135112e-02f -8.036747e-04f	3.136804e-03f 1.296527e-04f
21	-5.343918e-04f -2.567801e-03f 1.191627e-02f	-3.527944e-04f -4.583817e-04f -2.350958e-03f	-5.708543e-03f -3.297064e-03f	2.344187e-03f -4.281778e-03f
22	5.313205e-03f -7.768130e-03f -3.763157e-02f	-3.060121e-03f -1.816116e-02f -5.448145e-03f	3.451916e-03f -5.659431e-04f	3.572271e-03f 6.418265e-05f
23	4.536640e-03f -5.254188e-03f -2.616289e-03f	3.720265e-03f 3.040391e-04f -5.382307e-02f	-8.125322e-03f 2.546868e-03f	-9.654257e-04f -5.697106e-03f
24	1.930056e-03f -1.169625e-02f -3.106675e-03f	-4.884879e-03f 4.076919e-04f 1.229215e-02f	-5.432401e-03f 4.617881e-03f	-4.493898e-03f 2.846494e-03f
25	7.630727e-04f 1.196309e-02f -5.516066e-03f	-5.194896e-03f 5.050292e-04f -2.963855e-03f	-8.561001e-04f -1.020009e-02f	1.135338e-03f -4.390706e-03f
26	-9.012232e-03f 6.964354e-03f -2.521125e-02f	-8.066729e-03f 1.224982e-02f -7.764434e-03f	-4.476638e-03f -1.367829e-02f	-2.424281e-03f 1.333025e-02f

27	-1.231340e-02f 9.063985e-03f 3.585841e-03f	-9.839881e-04f 8.674999e-04f -6.049501e-04f	-5.437053e-04f 5.548672e-04f	-2.181440e-03f 8.796170e-03f
28	-5.835191e-04f 6.398769e-03f -4.046456e-03f	5.901638e-05f 1.761706e-03f 1.886682e-02f	-1.003937e-02f -1.033383e-02f	-8.384589e-04f -1.127180e-03f
29	6.943180e-03f 5.201217e-03f 6.993818e-03f	-1.228034e-02f 3.297588e-03f 5.403467e-03f	-2.376641e-02f -1.464098e-02f	-5.476645e-04f -1.815397e-02f
30	9.245361e-04f 5.378167e-03f -4.335742e-03f	9.502835e-04f -6.009482e-03f 1.099616e-02f	4.276267e-03f -1.429168e-02f	-2.597004e-03f -5.980256e-03f
31	-4.196643e-03f -8.324333e-03f -2.331149e-02f	-7.775502e-03f 2.460759e-03f -5.856800e-03f	-5.812760e-03f -1.255601e-03f	-1.126528e-02f -9.815509e-03f
32	5.219009e-03f -4.848321e-03f -1.027974e-02f	4.588355e-03f 6.300760e-03f 7.019409e-03f	-1.346122e-03f 2.198690e-03f	1.335562e-02f -2.758825e-03f
33	-6.550327e-03f -3.297690e-03f -3.800842e-03f	4.024782e-03f -2.986906e-03f -6.067349e-03f	-5.123823e-03f -3.326780e-03f	6.805215e-03f -1.079117e-02f
34	8.864013e-04f -1.064496e-03f -1.418880e-02f	1.006732e-03f 2.707334e-02f -1.058451e-02f	6.840052e-03f 4.086632e-03f	2.225737e-03f 1.428441e-02f
35	5.908180e-04f 7.669921e-03f -4.926024e-03f	9.655243e-04f 6.422146e-03f 7.150469e-03f	3.548276e-03f -5.714211e-04f	-1.740077e-03f 1.348061e-02f
36	4.785011e-03f -7.001920e-03f -4.994520e-03f	-1.878544e-03f -1.282223e-02f -2.581924e-03f	1.187175e-02f -9.249634e-03f	-1.402563e-03f -1.100614e-02f
37	4.581669e-03f 1.252587e-03f -1.177235e-02f	-3.169239e-03f 1.518512e-02f -2.234588e-03f	2.500002e-03f 4.241958e-03f	-7.694806e-04f 9.267900e-05f
38	3.359215e-03f -2.124306e-03f -7.455145e-03f	1.474975e-03f 4.617673e-04f 7.329806e-04f	6.296603e-03f -7.951679e-03f	-1.467316e-03f 8.812257e-03f
39	1.642256e-02f 1.804994e-02f -1.445575e-04f	1.361377e-03f 2.803421e-02f -6.925915e-03f	1.053976e-02f 1.152213e-02f	-1.230617e-02f 1.128512e-02f
40	-4.965399e-04f -4.296236e-03f -7.115822e-03f	1.894237e-02f -6.254675e-03f -1.350490e-02f	-5.062107e-03f 4.299899e-03f	1.806658e-02f 8.871094e-03f
41	4.753989e-03f 1.487601e-03f -4.273516e-03f	-2.136245e-03f -1.739816e-04f -1.266964e-02f	-4.543622e-03f 9.266555e-05f	6.128799e-03f 1.106707e-02f
42	3.289295e-03f -4.845134e-04f 9.288295e-03f	-2.743073e-05f 9.794218e-03f -7.643725e-04f	-4.322019e-03f 1.091470e-02f	1.264026e-02f 7.220844e-03f
43	-4.676359e-03f -8.502363e-03f -1.867017e-03f	-3.065087e-03f 3.981473e-03f -7.278460e-03f	-6.560667e-03f 8.444586e-03f	1.293418e-02f -5.541305e-03f
44	7.248946e-03f 7.185068e-03f 2.163633e-03f	3.667385e-03f -2.255985e-03f 5.295739e-03f	-6.084583e-03f -3.887213e-03f	4.100251e-03f 3.735730e-03f
45	4.301169e-03f 1.094010e-03f -1.866494e-02f	-4.279325e-03f 5.390837e-03f -1.940644e-02f	-2.250393e-02f -1.393955e-03f	3.999471e-03f 1.053176e-02f
46	1.502219e-03f -4.535594e-03f -1.203969e-02f	3.553818e-03f 1.354697e-02f -1.707431e-02f	-1.862778e-03f -3.938419e-03f	-1.333485e-03f -1.843958e-02f
47	8.735496e-03f 4.028604e-03f -2.899244e-03f	-7.703744e-03f 1.498923e-04f -5.787425e-03f	-3.452237e-04f -2.352069e-03f	-1.003456e-02f -1.267970e-02f
48	-3.071757e-03f -8.139105e-03f 1.328956e-03f	-1.528063e-02f 2.922330e-03f 4.631181e-03f	2.386456e-04f -7.244879e-03f	1.459874e-03f 4.484526e-03f
49	-1.424293e-04f 1.505187e-03f 8.882516e-03f	1.121354e-03f -9.658133e-04f 8.092896e-05f	1.196605e-02f 8.636725e-03f	-5.554850e-03f 2.411842e-03f
50	-4.814104e-03f 6.257660e-03f 8.043977e-03f	3.249655e-03f 7.166135e-03f 8.890442e-03f	9.763432e-03f -7.594627e-03f	4.678721e-03f 5.938945e-03f
	-7.351896e-04f	4.527810e-03f	2.232695e-02f	1.268362e-03f

51	2.247665e-02f 6.332500e-03f	9.523518e-03f 3.544931e-03f	-1.231988e-03f	8.451500e-03f
52	-6.693678e-03f -2.541699e-02f 2.154128e-02f	-2.094994e-02f -2.511156e-02f 3.189137e-03f	1.466342e-02f -2.883795e-02f	2.785460e-03f 8.784076e-03f
53	3.169312e-04f -8.366204e-03f 2.387933e-03f	-3.590467e-03f -2.206428e-02f -3.367410e-03f	-1.737606e-03f -3.366562e-03f	7.490725e-04f -1.179166e-02f
54	-1.000463e-02f 1.209800e-03f -2.712474e-03f	-8.101309e-03f -2.024245e-02f -1.217605e-02f	1.486096e-02f -2.370059e-03f	4.477646e-03f -2.105724e-03f
55	2.619521e-03f 2.397625e-03f 1.210525e-03f	3.262678e-03f 3.268025e-03f -1.148903e-03f	3.338823e-03f -3.789807e-03f	-1.010728e-02f -5.456019e-03f
56	1.781108e-03f -5.378151e-03f -1.222944e-02f	1.748406e-03f -7.266575e-03f -2.065578e-02f	1.754846e-03f 4.689460e-03f	1.892651e-03f -4.355156e-04f
57	1.952171e-02f 5.359528e-03f -5.329408e-03f	-2.184372e-04f -6.750782e-03f 3.017137e-03f	1.470817e-02f 3.672688e-04f	-9.406712e-04f -5.030805e-03f
58	3.378210e-03f -2.406230e-03f 3.649102e-03f	-8.008012e-03f 9.327831e-03f 3.070126e-03f	4.272059e-03f -1.876703e-03f	1.125698e-03f -1.025413e-02f
59	-4.000695e-03f -1.818168e-03f 4.899397e-03f	9.590422e-05f 2.557290e-03f -4.911112e-03f	6.651438e-03f -9.250555e-03f	-3.987135e-03f -6.597573e-03f
60	1.806250e-03f -4.500440e-03f 1.743263e-03f	-8.211662e-03f -7.834731e-03f 9.522632e-03f	3.422183e-03f -1.022606e-02f	2.466248e-04f 2.263320e-02f
61	3.792968e-03f -2.676394e-03f -7.882334e-04f	-5.870985e-03f -5.285670e-03f 3.115863e-03f	-9.668824e-04f -4.444628e-03f	-1.264334e-03f 2.425161e-05f
62	1.477234e-03f -1.560314e-03f -8.686257e-03f	-6.178742e-04f 8.950584e-03f 5.261796e-03f	1.855492e-02f -1.154117e-02f	-5.442784e-03f -5.555451e-03f
63	1.153779e-02f -1.917192e-02f -7.751277e-03f	-3.836777e-03f 4.882887e-03f -1.390134e-02f	3.825390e-03f -2.868351e-02f	-1.609805e-02f -1.085440e-02f
64	-3.796474e-03f -2.857565e-03f -8.942970e-03f	-1.301075e-02f -1.437400e-02f -4.038817e-03f	3.063430e-02f -5.574645e-03f	6.850234e-03f 2.959154e-03f
65	7.808479e-03f 3.206945e-02f -6.259714e-03f	-2.634109e-02f 1.693192e-02f 3.767237e-02f	4.068844e-02f 5.421588e-03f	5.313094e-02f -5.979769e-03f
66	-9.019999e-04f 8.444993e-04f -1.750796e-02f	-3.381927e-03f 1.561092e-03f -1.064411e-02f	1.024538e-02f -2.842569e-03f	-1.290770e-04f -3.369390e-03f
67	-4.947701e-03f 1.022994e-02f -2.136847e-03f	-2.921486e-03f 1.557420e-02f 1.757795e-02f	1.477167e-02f 1.052292e-02f	5.357224e-03f -2.888189e-02f
68	-1.088002e-03f -1.057155e-02f -7.407617e-03f	-6.627142e-04f 4.121159e-03f 3.677449e-03f	1.518418e-02f -4.728383e-03f	5.806171e-03f 6.810960e-04f
69	-1.696596e-03f 5.981191e-03f 4.185967e-03f	-1.113829e-02f -2.705377e-03f 1.198569e-02f	1.442984e-02f -1.048368e-02f	7.099913e-03f -5.870787e-03f
70	9.944247e-03f -1.460545e-03f 1.444231e-03f	1.312078e-02f -5.359000e-03f -6.734593e-03f	1.111332e-02f -2.145618e-03f	2.983248e-03f 1.444280e-02f
71	3.590251e-03f -3.462442e-03f 3.592758e-04f	5.033677e-04f 2.520362e-03f 1.719483e-02f	7.033519e-03f -4.293506e-03f	7.591726e-03f -3.843852e-03f
72	3.317157e-03f -6.588476e-03f -2.289806e-02f	6.773717e-03f 2.789388e-03f 7.927395e-03f	9.440879e-03f 4.589631e-03f	-4.687264e-03f 7.155508e-03f
73	-8.460711e-04f 1.598135e-03f -4.007360e-04f	-2.716979e-04f 8.676168e-03f 4.614720e-03f	8.788021e-03f 8.779903e-04f	2.110675e-02f 4.455623e-04f
74	-5.101791e-03f 2.474155e-03f -5.983723e-03f	1.479873e-03f 1.427604e-03f -8.433567e-03f	-3.538115e-03f -1.175528e-02f	-7.467103e-04f 3.859651e-02f
75	-9.313056e-03f 1.147418e-03f	-2.202025e-03f -3.387244e-03f	4.671892e-03f -8.628819e-04f	9.829534e-04f -6.948340e-03f

	8.803410e-03f	-6.292498e-03f		
	1.561238e-03f	5.481106e-03f	-2.633958e-03f	6.163511e-03f
76	6.616046e-03f	6.700101e-03f	-1.381128e-04f	-4.092984e-03f
	1.282249e-03f	-1.916181e-03f		
	8.215987e-04f	-2.910781e-03f	-9.832651e-03f	6.146809e-03f
77	1.685707e-02f	2.365627e-03f	1.692326e-03f	-1.268129e-02f
	-7.912889e-03f	2.091493e-03f		
	3.007293e-03f	7.177587e-03f	-2.644681e-03f	4.586885e-03f
78	2.387825e-03f	9.542564e-04f	4.717833e-03f	1.220110e-02f
	7.981248e-03f	3.629636e-03f		
	4.298180e-03f	-8.159025e-03f	-4.824093e-04f	-2.365997e-03f
79	1.443402e-03f	-1.282932e-02f	1.098509e-02f	3.758089e-03f
	1.356135e-02f	-4.514530e-03f		
	-6.769612e-04f	-1.727762e-03f	3.714987e-03f	-3.898517e-03f
80	-2.549338e-03f	-3.761156e-03f	8.887003e-03f	1.410497e-02f
	-8.664217e-04f	2.903544e-03f		
	-9.866845e-04f	-3.834414e-03f	1.270591e-02f	2.555730e-02f
81	2.022552e-02f	2.137707e-03f	2.311466e-02f	-5.480827e-03f
	-1.303191e-02f	-3.751276e-03f		
	-1.466219e-03f	-9.263150e-03f	-4.862572e-03f	-6.186845e-03f
82	-2.300912e-03f	2.629752e-05f	5.446638e-03f	-6.427285e-03f
	1.337930e-02f	1.616794e-02f		
	-3.868585e-03f	-5.603151e-03f	9.286169e-03f	7.025137e-03f
83	4.677228e-03f	5.045516e-03f	8.484001e-03f	-6.394735e-03f
	8.004981e-04f	1.067847e-04f		
	-1.674989e-03f	1.419860e-03f	2.677594e-03f	3.630918e-03f
84	-6.236864e-03f	-2.640378e-02f	2.705653e-03f	1.901610e-02f
	8.748479e-03f	3.738135e-03f		
	6.622282e-04f	-3.433154e-04f	2.628178e-04f	1.198161e-02f
85	9.386477e-03f	-2.960499e-03f	4.901425e-03f	-1.116236e-03f
	1.907657e-02f	-9.647131e-03f		
	-3.890410e-03f	3.495162e-03f	2.743093e-03f	-1.890167e-03f
86	9.803947e-04f	-1.466707e-02f	1.358645e-02f	-1.169555e-03f
	5.814509e-03f	3.583656e-03f		
	-1.229524e-03f	2.817739e-03f	-7.959671e-03f	-4.578121e-03f
87	9.118455e-03f	3.794858e-04f	5.459442e-03f	1.560436e-03f
	-1.195020e-03f	-1.311997e-02f		
	1.140458e-02f	2.806868e-03f	-1.834734e-03f	-2.504130e-02f
88	-3.268702e-02f	-6.601134e-03f	1.701849e-02f	2.752936e-02f
	-2.031868e-02f	3.096776e-03f		
	2.019510e-03f	-8.727133e-04f	8.793235e-03f	-1.941444e-04f
89	-7.305787e-03f	1.329055e-03f	2.819403e-03f	5.653743e-03f
	1.834889e-03f	-1.501772e-02f		
	1.297990e-03f	-5.127568e-03f	-8.875234e-03f	-1.711917e-02f
90	-4.423819e-03f	1.651548e-04f	1.193207e-02f	1.361834e-02f
	-2.054773e-02f	1.467692e-03f		
	-4.533740e-03f	5.103354e-03f	-3.235980e-03f	-3.337868e-03f
91	9.002754e-04f	9.557203e-04f	1.646606e-02f	-1.169197e-02f
	-3.593248e-03f	-1.280382e-02f		
	1.433295e-03f	2.060747e-03f	-7.271801e-03f	-2.818825e-03f
92	-1.076638e-02f	-1.981592e-03f	2.793604e-03f	1.113446e-02f
	-4.099952e-03f	2.333166e-03f		
	-2.755152e-03f	1.805468e-03f	-1.304548e-02f	2.815914e-04f
93	3.971184e-04f	-3.309465e-03f	1.034132e-02f	8.151849e-03f
	1.392269e-03f	8.734949e-03f		
	3.953810e-04f	9.894504e-04f	-5.035528e-03f	-2.980191e-03f
94	1.393229e-03f	-4.007698e-03f	-6.069160e-03f	-1.629844e-02f
	-5.166050e-03f	-1.317904e-03f		
	4.115699e-03f	-2.893366e-04f	-1.749452e-02f	1.403850e-03f
95	-5.916532e-03f	-2.379843e-02f	1.467787e-02f	-3.656255e-03f
	-2.474477e-02f	-1.148054e-02f		
	-4.425356e-03f	2.123684e-03f	1.116878e-02f	7.563566e-03f
96	5.676363e-04f	-4.268084e-03f	5.436841e-03f	9.760944e-03f
	-2.204822e-03f	-2.604242e-03f		
	4.615903e-03f	-3.641260e-04f	5.483156e-03f	1.333472e-02f
97	1.198365e-02f	-4.090365e-03f	-6.181048e-04f	-1.368374e-03f
	-1.458331e-02f	-7.889602e-03f		
	-1.136315e-02f	-2.866826e-04f	-2.057165e-03f	4.880846e-03f
98	1.446933e-02f	9.441224e-03f	-4.579787e-03f	-1.189617e-02f
	-3.542147e-02f	-4.345029e-03f		
	-8.908256e-03f	-4.168275e-03f	2.269481e-04f	3.423096e-03f
99	6.947558e-03f	4.497263e-03f	-2.783035e-03f	-1.890156e-03f
	-4.663287e-03f	-1.061505e-02f		

100	6.250840e-04f 4.557121e-03f -1.204765e-03f	-1.151767e-03f -3.011361e-03f 8.160173e-03f	1.289362e-02f 1.875621e-02f	-7.801888e-03f -6.158939e-03f
101	5.356279e-04f 6.055032e-03f 5.977912e-03f	-5.925554e-03f 5.458925e-03f -3.061780e-02f	6.480728e-03f -9.209208e-03f	-3.545455e-03f -1.713302e-04f
102	-7.031405e-03f -1.122026e-03f -2.380753e-03f	6.178479e-03f 8.589214e-03f 9.091722e-03f	2.768579e-03f 1.238128e-03f	-1.923700e-03f -3.604174e-03f
103	1.171870e-03f 8.296776e-03f -6.881003e-03f	-1.837033e-03f 2.512901e-03f 5.398510e-04f	-4.814577e-04f 1.413989e-02f	-9.032621e-03f 6.352842e-03f
104	1.576778e-03f -6.798749e-03f -4.836449e-04f	5.200060e-03f 3.448114e-03f -5.070876e-03f	-5.436710e-04f -1.490755e-02f	5.426778e-03f 1.747572e-03f
105	-5.215421e-03f -6.345541e-03f -9.332499e-03f	-3.977378e-03f 3.651820e-03f 9.406623e-03f	-1.837611e-03f -4.770854e-03f	5.479580e-03f -5.962035e-03f
106	-1.064628e-03f -1.842055e-03f -1.645009e-02f	-7.952652e-03f -1.086610e-02f -1.237778e-02f	3.631247e-03f -9.548634e-04f	8.729455e-03f -2.327376e-03f
107	-2.434592e-02f -1.424528e-02f -4.696528e-03f	-1.865899e-02f -4.188107e-03f -1.482153e-02f	6.686618e-03f 5.069447e-03f	1.481204e-02f 1.373214e-02f
108	-2.221314e-03f 7.379989e-03f 8.662346e-04f	1.410932e-02f 9.728472e-03f -8.168525e-03f	1.261800e-03f 1.913235e-02f	6.094280e-03f -1.482304e-02f
109	-4.372357e-03f 5.863091e-03f 1.653828e-04f	-1.735769e-03f 1.332675e-02f -3.590727e-03f	-5.715966e-03f 1.297880e-03f	-5.721986e-04f 1.123908e-03f
110	-6.890452e-03f 6.705485e-03f -9.210443e-03f	1.471589e-04f -6.068957e-03f 3.870675e-03f	6.505037e-03f 6.902003e-03f	-4.027538e-03f -5.908776e-03f
111	-7.630660e-03f -6.659314e-03f -8.678864e-03f	-8.518287e-03f 5.033774e-03f 7.462562e-03f	8.760021e-03f 1.275069e-02f	-9.885123e-04f -2.077635e-03f
112	-5.835616e-03f -4.760296e-03f -1.480192e-02f	-1.132843e-03f -8.536105e-03f 6.654873e-04f	2.124937e-03f -6.496619e-03f	-8.366280e-03f 2.814082e-03f
113	-9.821877e-04f 2.090477e-03f -1.696985e-02f	4.264982e-03f 2.955391e-03f -1.011088e-02f	6.088652e-04f 1.095093e-02f	9.837995e-03f 6.711218e-03f
114	-1.110748e-02f 6.296424e-03f 2.534050e-02f	-8.250327e-03f 6.094709e-03f 3.400851e-02f	3.149717e-03f 7.577754e-03f	2.511794e-04f -5.696883e-03f
115	-3.142475e-03f 6.941997e-03f 1.939965e-02f	6.319882e-03f 2.055704e-03f 1.881087e-02f	-1.987041e-04f 4.878204e-03f	-3.618158e-03f 3.764067e-03f
116	-2.089295e-03f -1.001415e-02f 1.198812e-02f	-3.349839e-03f -5.487581e-03f -5.929493e-04f	4.504871e-03f -3.088817e-03f	-9.847276e-04f 8.970303e-03f
117	9.570902e-04f 1.890317e-03f -7.990076e-03f	1.566858e-03f 3.328927e-03f 1.715328e-02f	-3.882535e-03f 6.419302e-03f	-3.472500e-03f -1.181835e-02f
118	-5.878435e-03f 3.427088e-03f -1.567122e-02f	3.586229e-03f -1.503075e-03f 8.872494e-03f	8.025705e-03f 9.069436e-04f	-5.391117e-03f -1.459295e-02f
119	2.705809e-03f 6.808136e-03f 3.935895e-03f	8.304176e-03f -4.047903e-03f 4.029136e-03f	-5.802255e-03f 2.862929e-04f	-2.458526e-02f -1.069763e-02f
120	-4.729689e-04f -2.498743e-02f -9.291148e-03f	-6.250156e-03f 5.916168e-04f -2.306988e-02f	-7.381983e-03f -1.198016e-02f	-8.996093e-03f 1.738222e-02f
121	6.555850e-03f -1.104411e-02f -3.682040e-03f	-4.315736e-04f 1.992889e-05f -1.715709e-04f	3.770165e-04f 4.129603e-04f	1.320119e-03f -9.792194e-03f
122	-7.402499e-03f -4.921407e-03f 9.882043e-03f	1.471881e-04f 1.152461e-03f 9.843392e-03f	-5.233563e-03f -4.351592e-03f	-5.411007e-03f -7.130635e-03f
123	-5.701188e-03f -2.242787e-03f 4.923654e-03f	-4.369586e-03f 3.586322e-03f 2.003899e-02f	-8.866201e-05f -1.186859e-03f	2.603499e-03f 1.121914e-02f
	-7.556915e-03f	4.557524e-03f	3.411263e-03f	-4.612558e-04f

124	-1.347571e-02f 1.387484e-03f	1.533404e-03f 2.049847e-03f	5.446113e-03f	4.850179e-03f
125	4.346351e-04f -1.139087e-02f -5.643114e-03f	6.802967e-03f 1.816490e-02f -9.911337e-03f	-1.273027e-02f 1.312108e-03f	-4.476177e-03f -6.613926e-03f
126	-1.724403e-03f -4.952252e-03f 6.949296e-03f	-1.701965e-03f 1.600046e-02f 5.370671e-03f	1.317123e-03f -5.244591e-03f	-9.935366e-03f -1.990009e-03f
127	4.013782e-04f -6.244740e-03f -1.308021e-02f	-1.967101e-04f 3.029080e-03f -1.108917e-02f	-2.041033e-03f -7.674530e-04f	-5.863588e-03f -5.887265e-04f
128	2.009635e-02f -1.767381e-03f -1.426612e-02f	-1.616427e-02f 1.370532e-02f -2.000061e-02f	-1.634870e-02f -1.028348e-02f	1.012120e-02f 3.172370e-02f
129	-1.283212e-03f 6.985568e-04f -8.058073e-03f	-2.877269e-03f -5.372681e-03f 3.575511e-04f	-4.985473e-03f 4.778471e-03f	1.025647e-02f -2.926942e-03f
130	2.327187e-03f 3.932593e-03f -1.683694e-03f	-9.726853e-03f 4.882387e-03f -7.491328e-04f	-4.845596e-03f 1.090868e-02f	1.140297e-03f 3.026093e-03f
131	-1.675478e-04f 3.863686e-03f 1.203345e-02f	4.346011e-03f -8.300460e-05f 5.704091e-03f	-3.669733e-03f 4.669054e-03f	3.787540e-03f -1.809716e-02f
132	9.645837e-04f -4.534004e-03f -1.867142e-02f	-8.251049e-03f 7.487620e-03f 3.976797e-04f	-2.059044e-03f -7.947177e-03f	7.187484e-03f 7.522218e-03f
133	-1.310627e-03f 3.370224e-03f 5.518240e-03f	2.974908e-04f 9.027549e-03f 9.640077e-03f	-9.215294e-03f -2.199570e-02f	3.499797e-03f -4.486226e-03f
134	5.372383e-03f -4.378232e-03f -1.900715e-03f	1.004509e-02f 6.883491e-04f -3.694575e-03f	9.488233e-03f 8.049537e-04f	8.842357e-04f -1.353789e-02f
135	-7.405158e-04f 3.498688e-03f 1.200173e-02f	1.102929e-02f 2.657408e-02f 1.181070e-02f	-1.671523e-03f 4.159486e-03f	-2.824078e-04f 5.849247e-03f
136	1.082905e-02f -3.188209e-03f -1.602523e-03f	-7.063797e-03f -2.386893e-03f 1.174584e-03f	-9.547434e-03f 4.540127e-03f	4.523777e-03f 1.011854e-02f
137	-4.373719e-04f -8.801198e-03f 2.164893e-02f	8.654565e-04f 1.110226e-02f 1.149692e-02f	-7.775592e-03f -3.135206e-03f	1.306263e-02f 1.629813e-03f
138	-1.409876e-03f 9.972697e-03f -1.148824e-02f	1.679593e-03f 2.631439e-03f 9.335315e-03f	-2.577483e-03f 1.052357e-02f	3.164723e-03f -2.301110e-03f
139	-8.867767e-03f 2.068160e-03f 2.051714e-02f	5.865880e-03f 2.218697e-03f 1.525336e-02f	-9.909803e-03f 1.065032e-02f	6.800980e-03f -1.646896e-03f
140	1.010600e-02f 5.003391e-03f 4.828895e-03f	1.026499e-03f 1.866356e-03f -4.067012e-04f	-4.410379e-03f -1.153865e-02f	8.672195e-03f -2.191056e-02f
141	4.524964e-03f -5.522647e-03f 8.950272e-03f	-1.669230e-02f 1.231121e-02f -5.977901e-04f	-1.637948e-02f -6.955485e-03f	1.137581e-02f -7.417289e-03f
142	-5.622047e-03f 7.301279e-03f -1.728730e-03f	1.231754e-02f -3.078053e-03f -3.285118e-03f	-1.938394e-03f -6.822500e-03f	-3.138165e-03f 2.083670e-03f
143	2.269233e-03f -8.931928e-04f 1.636184e-02f	-1.773863e-03f 1.761113e-03f 1.957928e-02f	-1.557281e-04f -9.610383e-03f	2.068389e-03f -3.364177e-03f
144	6.097324e-03f 5.159747e-03f 4.489202e-03f	-4.146446e-03f 4.916488e-03f -5.474790e-03f	-4.495471e-03f -2.946672e-03f	-6.768560e-03f 1.467899e-02f
145	2.511621e-03f -3.293215e-03f 9.614059e-03f	2.546622e-03f -7.851399e-03f 1.773115e-02f	4.100445e-03f 5.429499e-03f	1.344009e-03f -7.704205e-03f
146	9.167643e-05f 3.255055e-02f -4.371985e-03f	-7.245598e-04f -8.385970e-04f -7.219145e-03f	-3.256085e-03f -4.705970e-03f	-1.436779e-02f -1.452480e-03f
147	1.501950e-03f 1.744604e-02f 9.766145e-04f	1.941176e-03f -1.293741e-02f -2.658267e-03f	-3.102586e-03f 1.844106e-03f	-5.616851e-03f 1.944136e-03f
148	-4.867795e-03f 1.141799e-02f	-1.681389e-04f -3.207362e-04f	6.563957e-03f -3.241608e-02f	6.349938e-03f 1.014923e-02f

	1.692820e-02f	2.614240e-03f		
149	-1.029514e-03f -3.246950e-04f -5.196892e-04f	1.252921e-03f -9.435491e-03f -8.200826e-03f	2.202538e-03f -2.278429e-02f	-6.891731e-04f 2.419470e-02f
150	2.503258e-03f 7.690421e-03f 6.091185e-03f	-1.405898e-03f -6.993480e-03f 5.595274e-03f	1.455035e-03f -4.170123e-04f	-1.651529e-04f -1.270636e-02f
151	2.668220e-03f -7.978587e-05f 1.119817e-02f	9.318681e-03f 3.704762e-03f -5.074611e-03f	-2.093518e-04f -1.230244e-02f	-1.180648e-02f 1.574853e-03f
152	-1.877933e-03f -5.738072e-03f 9.189512e-04f	-4.458337e-03f -3.579115e-03f -4.441866e-05f	4.796065e-03f 3.049552e-03f	-4.656425e-03f -1.298617e-02f
153	-4.259479e-03f 1.940283e-03f 5.329833e-02f	-5.828690e-03f 9.366261e-03f 2.379910e-02f	-2.126029e-03f 3.307405e-03f	2.736130e-03f -9.878902e-03f
154	-1.453727e-04f 1.185485e-02f 1.367926e-02f	-4.885160e-03f 7.445680e-04f 7.745240e-03f	-3.880866e-03f 3.412077e-04f	-5.796530e-03f -1.924183e-03f
155	2.126873e-03f -1.357793e-03f 2.792752e-02f	1.493234e-03f 7.398484e-04f 2.335526e-02f	-5.044834e-03f -6.817926e-03f	-5.652982e-03f -1.140052e-02f
156	5.106198e-03f 4.944115e-03f 1.881001e-02f	-1.089109e-03f 2.496819e-03f 3.167433e-03f	-2.729246e-03f -1.365014e-03f	-5.287427e-03f 8.297749e-03f
157	-3.385242e-03f 9.327208e-04f 6.731312e-03f	2.007139e-03f 1.757545e-03f 2.578418e-03f	-1.283784e-02f -1.106829e-02f	7.398259e-04f 1.599873e-02f
158	1.050102e-02f 3.085984e-03f 7.385422e-03f	-2.174708e-03f -9.758797e-03f 1.069597e-02f	3.901874e-03f -3.693609e-03f	-7.012685e-03f 6.219022e-03f
159	-1.334340e-03f 4.109002e-03f -8.912928e-03f	-7.548498e-03f -6.427865e-03f 7.001471e-03f	-2.697914e-03f -2.732459e-03f	-6.636421e-03f -2.370820e-03f
160	1.820463e-03f -4.671470e-03f 3.849501e-03f	1.188937e-03f 9.103372e-03f -7.700901e-04f	-1.099726e-02f -4.212771e-03f	7.239681e-04f 4.940898e-03f
161	-9.754810e-03f -6.445055e-03f -2.845778e-03f	-7.974691e-04f -1.259495e-02f -9.934577e-03f	-3.203113e-02f 9.365228e-03f	-9.014598e-03f 9.944174e-03f
162	3.956845e-03f -2.602407e-03f -2.108247e-02f	1.092550e-02f 9.542481e-04f -3.082176e-03f	-4.391141e-03f -5.544065e-03f	-2.012141e-03f 9.808995e-03f
163	-2.981349e-03f 3.039923e-03f 1.810883e-02f	-2.470687e-03f -1.551408e-03f -2.214932e-02f	-7.158190e-03f 1.985945e-03f	-1.626892e-03f 9.349466e-04f
164	-6.465100e-04f 6.467958e-03f -1.956256e-02f	4.641646e-03f -6.941768e-04f -1.565438e-02f	-9.010836e-04f -4.291298e-03f	6.965185e-03f -9.221018e-03f
165	-1.394155e-03f 5.926352e-04f -1.361746e-02f	-3.832226e-03f 8.705970e-04f -6.628834e-03f	-1.237637e-02f 3.972409e-03f	1.343527e-03f -2.937693e-03f
166	4.659873e-03f 9.681660e-03f 2.760603e-03f	6.670085e-03f -1.868635e-02f -1.321711e-02f	3.209983e-03f -5.518669e-03f	-4.557226e-03f -5.971916e-03f
167	6.195265e-03f 1.094670e-03f -4.045138e-04f	6.693259e-03f 5.416763e-03f -1.906802e-03f	-1.061671e-03f 1.355314e-02f	-8.128346e-03f -2.283274e-03f
168	-2.641427e-03f -6.595279e-03f 2.605402e-03f	2.263038e-04f -3.081414e-03f -3.843668e-02f	-2.303206e-04f -2.740725e-03f	1.142500e-02f -4.481534e-03f
169	-6.099933e-03f -8.810377e-03f 6.767384e-03f	-1.320001e-02f -1.766431e-03f -9.149113e-04f	-1.751140e-02f 1.400047e-02f	6.502512e-03f -1.098847e-02f
170	-4.696758e-03f 4.989055e-03f -1.408448e-02f	-3.304598e-03f -4.841784e-03f -1.053598e-03f	2.135345e-03f -1.796347e-03f	3.915078e-03f 8.524349e-03f
171	-3.857836e-03f -1.081667e-02f 4.152765e-03f	7.959062e-04f 8.050891e-03f -4.438237e-03f	-1.868777e-03f 4.238438e-03f	7.035939e-03f 3.127550e-03f
172	7.175456e-03f 2.866419e-03f 6.956621e-03f	1.044835e-03f -1.782649e-03f 4.908226e-05f	3.165597e-03f 1.773497e-02f	6.018296e-03f -4.071055e-03f

173	5.216455e-03f 1.391328e-03f 1.874495e-04f	-5.455938e-04f 7.787799e-03f 1.122079e-02f	-1.075868e-02f 2.679134e-03f	-3.992029e-03f -3.910262e-03f
174	1.671591e-03f -2.453581e-03f 6.180472e-03f	2.558857e-03f 8.224363e-04f 7.392424e-04f	1.619036e-02f -5.073382e-04f	-3.730161e-03f -1.377831e-02f
175	-1.038429e-03f -1.581226e-02f 6.596997e-03f	2.746212e-03f 6.540696e-03f 6.433421e-03f	1.665192e-03f -5.454986e-04f	-6.623864e-03f -4.808191e-03f
176	-7.740358e-04f 1.897090e-03f 1.432920e-02f	-3.003001e-03f 3.890266e-03f 1.132637e-02f	-5.871153e-06f 1.026016e-02f	2.886696e-03f 3.531602e-03f
177	-9.458872e-03f -3.840393e-03f 1.922981e-03f	-3.318168e-03f 1.112080e-03f 9.854094e-04f	-5.438146e-03f 9.993481e-03f	-3.325660e-03f 9.166464e-04f
178	4.345672e-03f 1.263180e-02f -6.536259e-03f	9.171013e-04f 6.139069e-03f -8.607442e-03f	8.081214e-03f 2.540349e-03f	-4.564138e-03f 6.029630e-03f
179	-3.529347e-03f 4.928914e-03f -2.855207e-03f	-7.503077e-03f 4.220743e-03f 1.646062e-03f	7.770835e-03f -3.003082e-03f	-7.283617e-03f -3.021387e-04f
180	-4.740290e-03f -8.331764e-03f -3.724498e-03f	3.490780e-03f -1.305356e-02f -5.889904e-03f	1.024996e-02f -3.199067e-03f	-7.098404e-04f 4.492775e-04f
181	1.689262e-03f -4.934974e-03f -4.790756e-03f	2.602116e-03f 1.384021e-04f 1.594088e-02f	4.546674e-03f 5.742726e-03f	-8.373111e-03f 1.568993e-03f
182	4.647950e-03f 4.269918e-03f -4.090967e-03f	8.670906e-03f -1.362179e-02f -1.830455e-02f	2.342103e-02f 2.381012e-03f	-5.464273e-03f -5.854967e-03f
183	-3.724628e-03f -1.003696e-03f 2.116769e-03f	3.151410e-03f 1.942707e-03f -4.539477e-03f	1.638087e-02f 2.033717e-03f	-1.630834e-02f 1.857484e-02f
184	6.593342e-03f 5.173538e-03f -4.493344e-03f	-9.674336e-04f -2.156589e-02f 8.995987e-03f	-4.077476e-03f 8.640969e-03f	5.583808e-03f 7.503193e-03f
185	-2.030477e-04f 5.563662e-04f 3.644231e-04f	6.742311e-05f -5.748491e-04f 4.656328e-04f	-3.249331e-05f -5.355374e-06f	5.178882e-04f 7.157761e-04f
186	6.339733e-03f -6.179531e-04f 8.924466e-03f	-4.108027e-03f -1.703425e-03f 7.603307e-03f	9.277836e-03f 7.613539e-03f	7.785245e-03f 4.728320e-03f
187	2.269498e-04f -1.056320e-02f 3.191192e-03f	-2.392239e-03f -3.684518e-03f -9.512989e-04f	3.471490e-03f 1.130002e-02f	4.224307e-03f -2.423250e-02f
188	8.834446e-04f -2.098013e-03f 7.915112e-03f	-8.707259e-03f -7.649009e-03f 7.465514e-03f	9.633941e-03f 2.095269e-02f	4.780505e-03f 2.758903e-02f
189	4.999262e-05f -9.813153e-03f -2.520232e-03f	6.430664e-04f 1.080979e-02f 1.469078e-03f	2.414457e-03f -3.619887e-03f	2.435233e-03f 1.903166e-02f
190	1.512080e-02f -6.812869e-04f 1.216848e-02f	-1.451399e-03f 1.131907e-02f 7.412558e-03f	1.565742e-02f 1.644202e-02f	1.801945e-03f 1.577487e-02f
191	-7.040548e-04f -4.368359e-03f 2.118084e-03f	-5.256371e-03f 2.543690e-03f -1.073636e-02f	2.613425e-03f 3.782241e-03f	-7.744323e-03f 4.062086e-03f
192	-2.586948e-03f 4.410753e-03f 6.477655e-03f	-8.689556e-03f 1.317612e-03f -1.456456e-03f	2.268067e-03f 3.464204e-03f	6.532260e-04f 1.306682e-02f
193	-2.854732e-04f 6.811229e-03f 1.958587e-02f	-3.743561e-03f -5.145066e-03f 7.243095e-03f	1.603250e-03f 1.332821e-02f	3.051908e-02f 2.516475e-03f
194	-7.941837e-04f -5.456270e-03f -4.260652e-03f	7.425379e-03f -7.785540e-03f 4.619726e-04f	-6.983161e-03f -3.021155e-03f	3.510630e-04f -3.609280e-03f
195	1.617885e-03f 1.307084e-03f -3.720478e-03f	-4.543330e-04f 5.656063e-03f -1.692654e-02f	3.490402e-03f 3.176480e-03f	1.101711e-02f -5.386925e-03f
196	1.339924e-02f 3.693620e-03f 1.950782e-03f	-2.756446e-03f 3.741737e-03f -1.474861e-02f	6.183488e-04f -2.221686e-02f	1.625494e-02f -3.108618e-03f
	1.221647e-03f	-3.338271e-03f	5.314416e-03f	7.587672e-03f

197	2.806886e-03f 1.144202e-02f	3.934485e-03f -6.561669e-03f	-4.572839e-03f	3.317354e-03f
198	8.942259e-03f 9.362173e-03f 1.789632e-03f	5.816611e-03f -9.995839e-03f 1.488410e-02f	-3.685696e-03f -5.057485e-03f	1.515044e-02f -7.581743e-03f
199	6.186283e-03f 5.732404e-03f 1.370589e-02f	9.226641e-03f 4.144744e-03f 2.636809e-02f	-9.456077e-03f 1.026376e-02f	-2.228343e-03f 2.117148e-03f
200	-5.173255e-04f -2.138965e-02f 1.732320e-02f	-1.382302e-03f 2.967975e-03f 1.735175e-02f	-2.838757e-03f -2.063015e-03f	1.132654e-02f 6.838708e-03f
201	3.639430e-04f 1.062672e-03f -6.214810e-03f	2.238074e-03f 4.776641e-03f 2.001370e-03f	-6.147094e-03f -1.620411e-03f	2.941333e-02f 1.562529e-02f
202	-2.158268e-03f 3.855857e-03f 1.265091e-02f	-2.905785e-03f -8.698823e-03f 1.095882e-02f	-6.237764e-03f -1.600154e-03f	4.605827e-03f 9.986827e-03f
203	7.552677e-03f -4.423656e-03f 9.139054e-03f	5.642658e-05f -3.028379e-03f 6.194433e-03f	-1.122479e-02f 9.717856e-03f	7.167683e-03f -1.078198e-02f
204	9.534687e-04f -6.268092e-03f 6.910685e-03f	-4.524385e-03f 2.133547e-03f -1.672165e-03f	1.184639e-03f 2.828050e-04f	5.700747e-03f -7.576027e-03f
205	5.587663e-03f -5.123777e-03f 1.118184e-02f	1.249403e-03f -1.166972e-02f -3.695586e-03f	-1.405867e-02f 5.972990e-04f	4.856740e-03f 1.798762e-03f
206	-1.600761e-03f 5.197397e-03f -5.422369e-03f	5.124725e-03f 1.133720e-02f -6.967892e-03f	3.847980e-03f -1.209140e-02f	2.760361e-03f -5.171445e-03f
207	-2.587543e-03f -8.942887e-04f -1.971179e-03f	4.485243e-03f -1.065368e-02f 1.517567e-02f	2.151197e-04f 3.121157e-03f	1.132871e-04f 1.783662e-03f
208	-1.743579e-03f -4.220432e-03f -4.581736e-03f	-5.254102e-04f 1.144197e-02f 7.017686e-03f	3.153341e-03f 1.051422e-02f	1.630122e-03f -3.051846e-03f
209	8.684423e-04f 8.407292e-03f -3.877544e-03f	6.165525e-03f -4.846681e-03f 1.345447e-02f	7.560868e-03f 8.363402e-03f	8.230970e-03f 2.834938e-03f
210	-5.451650e-03f 5.236057e-03f -1.028101e-02f	-5.491594e-04f 1.042214e-02f -5.089497e-03f	-6.463978e-03f -5.848783e-03f	-1.375537e-02f -9.913163e-05f
211	7.321546e-04f 3.139449e-04f -1.189506e-02f	5.423307e-03f 7.984062e-03f -1.550004e-03f	-4.685918e-03f -3.058955e-03f	-2.481543e-03f 4.229893e-03f
212	-1.048727e-03f 5.521541e-03f 3.130422e-04f	-6.656242e-04f -1.022653e-02f -1.483878e-02f	2.465317e-03f -2.849140e-03f	3.673826e-03f 3.411459e-03f
213	1.115606e-03f -5.332213e-03f 7.936094e-03f	7.017450e-04f -5.806324e-03f -1.530203e-03f	-4.067371e-03f 2.033036e-03f	-1.003285e-02f 2.436597e-03f
214	7.041058e-03f 4.069921e-03f -1.372507e-02f	7.301904e-05f -7.601162e-03f 7.758863e-03f	2.323989e-03f 3.685917e-03f	-4.765281e-03f 5.657341e-03f
215	1.651264e-03f -8.104994e-03f 2.297167e-02f	-2.243988e-03f -9.388414e-03f 8.154996e-03f	-4.183309e-03f -2.533528e-03f	-1.595446e-02f -5.130938e-05f
216	4.671375e-04f -1.943462e-02f -2.004089e-02f	-1.462560e-03f 1.135505e-02f -1.020578e-02f	8.156076e-03f 2.202265e-03f	-1.184664e-02f 1.549079e-04f
217	-3.759946e-03f -9.484748e-04f -4.187244e-04f	-2.534319e-03f -4.377515e-03f -6.102782e-03f	-5.134874e-03f -1.376678e-02f	4.404813e-04f 4.844753e-03f
218	3.236813e-03f 3.584662e-03f -2.799418e-02f	1.301519e-03f -1.943646e-03f 3.395132e-03f	2.375495e-03f 2.638764e-03f	-1.038060e-02f -5.743115e-03f
219	1.579778e-03f -5.775390e-03f -9.860662e-03f	5.471562e-03f 4.563178e-03f 8.736962e-03f	-1.479010e-02f -6.328922e-03f	-1.645372e-02f -2.403495e-02f
220	8.023615e-03f -4.418287e-03f -3.474543e-03f	4.543775e-03f -4.456788e-03f -8.938221e-04f	-2.754634e-03f -7.908101e-03f	-9.218303e-03f 2.084426e-03f
221	2.454518e-03f -1.687993e-02f	7.652345e-03f -1.568754e-02f	-2.538912e-02f -1.049650e-02f	-8.178637e-03f 3.168479e-02f

	1.105713e-02f	1.060880e-03f		
222	1.234106e-02f 1.628911e-03f -2.420058e-02f	-6.957438e-03f 7.505491e-03f -3.525581e-02f	8.292550e-03f -1.589341e-02f	-8.070498e-03f -1.086891e-03f
223	2.924323e-03f 5.896509e-04f -4.233774e-03f	4.164691e-04f -1.072079e-02f -1.371477e-02f	-1.109954e-02f -1.206337e-02f	-7.406602e-03f 4.816261e-04f
224	7.045474e-03f -6.522211e-03f 6.612040e-03f	2.541373e-03f -1.113079e-02f 3.458895e-03f	-1.464331e-03f -8.191359e-03f	6.138415e-03f 1.485779e-02f
225	-3.693989e-03f 1.645154e-03f -1.144421e-03f	-4.622853e-03f -8.042960e-03f 4.433040e-03f	-4.280869e-03f 9.825301e-03f	7.630988e-04f -3.762208e-03f
226	-1.072490e-02f -6.109325e-03f -2.898886e-03f	1.479915e-03f 1.276393e-03f -5.966623e-03f	5.659275e-03f -1.620660e-02f	-4.304069e-04f -5.063098e-04f
227	1.537702e-03f -1.235016e-02f 1.244742e-02f	3.309934e-03f -4.358308e-03f 9.875230e-03f	-4.049235e-03f 1.228403e-02f	2.159129e-03f -5.957560e-03f
228	9.189809e-04f 1.052246e-02f 4.066999e-03f	2.388605e-03f -2.400716e-03f -2.555106e-03f	7.302065e-03f -8.264933e-04f	2.856177e-03f -2.009176e-03f
229	-4.117063e-03f 1.054326e-03f 4.755150e-03f	-8.044141e-03f 9.672686e-03f 5.706816e-03f	6.933914e-04f 5.034051e-03f	-6.251663e-03f -3.140959e-02f
230	-1.452558e-04f -4.130097e-05f 8.908890e-03f	1.999158e-03f -4.165500e-03f 1.602218e-02f	7.213169e-03f -8.618700e-03f	6.695970e-04f 9.211879e-03f
231	-3.138093e-03f 3.584889e-03f -2.688278e-03f	2.167731e-03f -3.667630e-03f 1.775102e-04f	-5.264122e-03f -3.547350e-03f	-1.177892e-02f 5.124674e-03f
232	-9.287096e-03f -1.241790e-02f 7.138685e-03f	2.464166e-03f -2.813124e-02f 9.062703e-03f	-3.283683e-03f -2.916337e-02f	1.924886e-02f 3.330583e-03f
233	-4.608391e-03f -4.636431e-03f 9.676142e-03f	1.230328e-03f -3.486980e-03f 1.126921e-02f	7.143751e-03f -3.163074e-04f	6.443089e-03f -5.138342e-03f
234	-1.160231e-02f 1.272242e-02f 7.745907e-03f	-7.906822e-03f -6.788409e-03f 1.442892e-04f	5.549992e-03f -2.095316e-02f	8.666742e-03f 1.618161e-02f
235	-9.347779e-03f -1.900776e-02f -1.855089e-02f	-4.224787e-03f -6.096506e-03f -1.698709e-03f	-1.427022e-03f -1.452094e-02f	6.101868e-03f -4.211964e-03f
236	-2.140165e-03f 2.327005e-04f -3.543316e-04f	1.018454e-03f -6.605003e-03f 1.117882e-02f	6.427939e-03f -4.593808e-03f	1.508304e-02f -1.420942e-02f
237	-2.996384e-03f -1.074687e-03f 1.751561e-02f	4.293223e-03f 1.146860e-02f -3.869521e-03f	-1.412075e-03f -5.492439e-04f	1.579988e-03f 6.540580e-03f
238	-1.775320e-02f 3.357234e-03f 8.117375e-03f	8.440915e-03f 3.511100e-03f 3.212914e-03f	1.217321e-02f -1.127433e-02f	2.490315e-03f -8.528232e-03f
239	-1.661806e-02f -1.162106e-02f -1.562507e-02f	3.728558e-03f -5.897354e-03f -8.563295e-03f	-3.135371e-04f 6.038056e-03f	-6.894804e-03f -1.767297e-04f
240	-4.525748e-03f -1.180225e-02f 1.279893e-02f	1.471726e-02f -4.118816e-03f -1.398923e-03f	3.271689e-03f -1.152821e-03f	-5.769737e-03f 5.504507e-03f
241	7.044734e-03f -7.355639e-03f 1.648913e-02f	3.276661e-03f 6.844633e-03f 1.361675e-02f	6.057112e-03f 2.956307e-03f	-1.072844e-03f 6.242455e-03f
242	1.259124e-03f -1.664025e-03f 6.860351e-03f	4.120835e-04f 9.208011e-03f -8.386478e-03f	-3.756579e-03f 1.056618e-02f	-4.613922e-03f -8.902084e-03f
243	1.933795e-03f -7.273558e-03f 8.182457e-03f	2.480058e-03f 1.121324e-03f 5.182913e-03f	-7.604521e-03f 1.138499e-02f	-2.056746e-02f 8.894619e-03f
244	-8.734879e-03f 7.061445e-05f -2.784501e-04f	5.237911e-03f -4.373261e-03f -5.314958e-03f	-3.194846e-03f 8.336578e-03f	-1.042711e-02f 1.853904e-03f
245	-3.622146e-03f 2.214230e-03f 3.810297e-03f	-2.685551e-03f 2.906515e-03f 1.052135e-02f	2.860146e-03f -1.637548e-03f	-1.810038e-02f -1.265239e-03f

246	1.655211e-03f 7.848798e-04f -3.474118e-03f	-5.312223e-03f -1.015646e-02f -8.148873e-03f	7.842831e-03f 4.334745e-03f	-1.617296e-02f -7.474289e-03f
247	-4.072711e-03f 1.258817e-03f -5.897508e-04f	-1.313786e-03f -4.525826e-03f -1.275762e-02f	7.231620e-03f -8.539935e-03f	-4.005538e-02f 2.265568e-02f
248	-1.684814e-03f 5.717116e-04f 1.618451e-02f	3.258137e-03f -8.364419e-03f 2.420768e-03f	1.656033e-03f -8.392453e-03f	-3.780806e-04f -1.722541e-03f
249	8.074787e-03f -2.796396e-02f -3.534284e-03f	1.700991e-06f -5.108903e-04f 1.250662e-03f	-4.033729e-03f -1.731766e-03f	2.833071e-03f -7.693001e-03f
250	-6.807478e-03f -1.547196e-03f -4.926458e-03f	3.440142e-03f 5.394689e-03f -8.328206e-03f	-2.280401e-03f -2.597917e-04f	3.833268e-03f 1.457782e-02f
251	2.294977e-03f -1.028991e-02f 1.063693e-02f	6.763667e-03f -3.523227e-04f -2.175203e-03f	-8.895122e-04f -3.199688e-03f	2.811571e-03f -4.481031e-03f
252	3.937041e-03f -3.396326e-03f -4.365937e-03f	-5.549578e-03f -2.495042e-03f -5.325665e-03f	-2.943946e-03f 2.024741e-02f	1.916947e-03f 1.332101e-04f
253	-2.502442e-03f -1.206020e-02f 1.123403e-02f	-5.870495e-03f -1.757752e-03f 3.502432e-03f	-1.369882e-02f -1.102435e-02f	-1.841288e-03f -1.421059e-03f
254	5.034698e-03f 3.478883e-04f 2.054781e-02f	2.726211e-03f 8.950127e-03f 4.282601e-03f	8.391868e-03f 1.946151e-03f	-2.610117e-04f -6.372063e-03f
255	1.515230e-03f 9.183522e-04f 4.782481e-03f	-6.864538e-04f -2.806403e-03f 2.085263e-02f	-1.757583e-03f 1.144045e-02f	-1.619812e-02f -2.567438e-03f

ISO/JTC 1/SC 29 **N2203CELP**

Date: 1998-05-13

ISO/IEC 14496-3 FCD

ISO/JTC 1/SC 29/WG11

Secretariat:

Information Technology - Coding of Audiovisual Objects

Part 3: Audio

Subpart 3: CELP

Document type: International standard
Document:sub-type if applicable
Document:stage (20) Préparation
Document:language E

DATEINAMEISOSTD Basic Version 1.8 1996-10-30

FCD 14496-3 Subpart 3

CELP Audio

NORMATIVE PART OF MPEG-4 CELP CODER

1. Glossary

For the purposes of this part of ISO/IEC 14496, the following additional definitions apply.

1. **adaptive codebook,** an approach to encode the periodicity of the signal. The entries of the codebook consists of overlapping segments of past excitations.
2. **bandwidth scalability,** the possibility to change the bandwidth of the signal during transmission.
3. **bitrate scalability,** the possibility to transmit a subset of the bitstream and still decode the bitstream with the same decoder.
4. **CELP,** Code Excited Linear Prediction
5. **complexity scalability,** the possibility to decode the bitstream with a lower complexity decoder.
6. **demultiplexing,** splitting one bitstream into several.
7. **excitation,** the excitation signal represents the input to the LPC module. The signal consists e.g. of contributions that cannot be covered by the LPC model.
8. **enhancement layer(s),** the part(s) of the bitstream that is possible to drop in a transmission and still decode the bitstream.
9. **fine rate control,** the possibility to change the bit rate by, under some circumstances, skipping transmission of the LPC indices.
10. **fixed codebook,** the fixed codebook contains excitation vectors for speech synthesis filters. The contents of the codebook are non-adaptive (i.e. fixed).
11. **index,** number indicating the quantized value(s).
12. **LAR,** Log Area Ratio.
13. **lossless coding,** coding of bits without loss of information. By means of lossless coding the required information is reduced.
14. **LPC,** Linear Predictive Coding.
15. **LSP,** Line Spectral Pairs.
16. **MPE,** Multi Pulse Excitation.
17. **multiplexing,** combining several bitstreams into one.
18. **postfilter,** this filter is applied to the output of the synthesis filter to enhance the perceptual quality of the reconstructed speech.
19. **RPE,** Regular Pulse Excitation.
20. **scalar quantizer,** quantizing one value to an index.
21. **unvoiced frame,** unvoiced speech looks like random noise with no periodicity.
22. **variable bit rate,** the number of bits corresponding to a coded frame varies over time.
23. **vector quantizer,** quantizing several values to one index.
24. **voiced frame,** a voiced speech segment is known by its relatively high energy content, but more importantly it contains periodicity which is called the pitch of the voiced speech.

2. Introduction of CELP Core

2.1 General Description of CELP Decoder

This section provides a very brief over view of the CELP decoder. A more extensive description can be found in [1]. A basic block diagram of a CELP decoder is given in Figure 1.

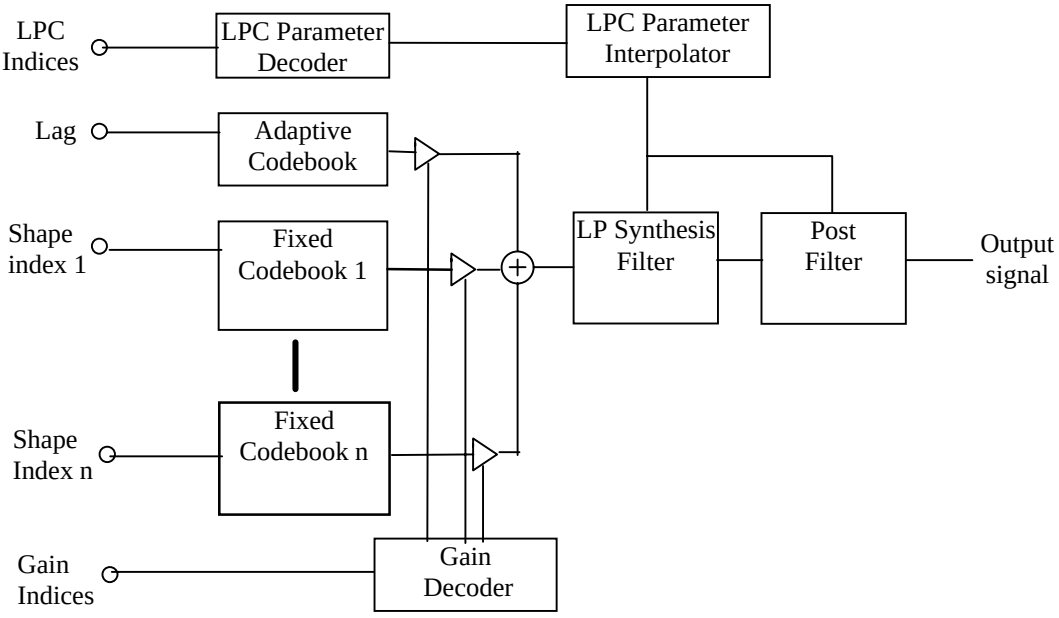


Figure 1. Block diagram for a CELP decoder

The CELP decoder primarily consists of an excitation source and a synthesis filter. Additionally CELP decoders often also include a postfilter. The excitation source has both periodic components, contributed by the adaptive codebook, and random components contributed by one or more fixed codebooks. At the decoder, the excitation signal is reconstructed using the codebook indices (pitch lag for the adaptive codebook and shape index for the fixed codebook) and gain indices (adaptive and fixed codebook gains). This excitation signal is then filtered by the linear predictive synthesis filter (LP synthesis filter). This filter is obtained by interpolating the LPC coefficients of successive analysis frames, where the LPC coefficients are reconstructed using the LPC indices. Finally, a postfilter is applied in order to enhance the speech sound quality.

2.2 Functionalities of MPEG-4 CELP

What makes MPEG-4 CELP different from the conventional CELP is the flexibility that it offers. Conventionally, CELP schemes offer only compression functionality at a single bit rate targeted at a particular application. Within MPEG-4, high quality compression is only one of the many features, thus making it possible for one basic coding scheme to address various applications. The possibility to generate arbitrary bit rates, bit-rate scalability in steps of 2 kbit/s (for signals sampled at 8 kHz) and decoder complexity scalability are some of the novel features that the MPEG-4 CELP coder provides. The basic idea behind the MPEG-4 CELP coder is to cater for multiple functionalities.

2.2.1 Functionalities of the MPEG-4 CELP coder

The MPEG-4 CELP coder supports the following functionalities:

- Multiple bit rates
- Bit-Rate Scalability
- Bandwidth Scalability

- Complexity Scalability

Multiple bit rates: Two sampling rates are supported namely 8 and 16 kHz. The associated bandwidths are 200 – 3400 Hz for the 8 kHz and 50 – 7000 Hz for the 16 kHz sampling rates. Both fixed and variable bit rates are supported. The possible bit rates fall into two categories depending on the type of applied LPC Quantizer. For a sampling rate of 8 kHz, the following fixed bit rates are supported:

Bit rates for Scalar Quantizer (bit/s)	Bit rates for Vector Quantizer (bit/s)
4325, 4725, 5125, 5534, 5834, 6134, 6650, 6950, 7250, 7550, 7850, 8050, 8250, 8650, 9250, 9650, 10050, 10450, 10850, 11250, 11450, 11650, 12900, 13300, 13700, 13900, 14100	3850, 4250, 4650, 4900, 5200, 5500, 5700, 6000, 6300, 6600, 6900, 7100, 7300, 7700, 8300, 8700, 9100, 9500, 9900, 10300, 10500, 10700, 11000, 11400, 11800, 12000, 12200

Table 1. Fixed bit rates for speech sampled at 8 kHz

For a sampling rate of 16 kHz the following fixed bit rates are supported:

Bit rates for Scalar Quantizer (bit/s)	Bit rates for Vector Quantizer (bit/s)
13667, 15867, 18200, 20133, 24000	13267, 15067, 17000, 19333, 23200

Table 2. Fixed bit rates for speech sampled at 16 kHz

When variable bit rate operation is required, it is possible to encode at an arbitrary bit rate within the range of bit rates tabulated in Table 1 and Table 2.

Bit-Rate Scalability: For the 8 kHz sampling rate, bit-rate scalability is provided by adding enhancement layers. An enhancement layer can be added with a step of 2000 bit/s. A maximum of three enhancement layers may be combined with a bit rate chosen from Table 1.

Bandwidth Scalability: Bandwidth scalability to cover both the sampling rates is realized by incorporating a bandwidth extension tool to the CELP coder. This is an additional tool, supported in the 8 kHz sampling rate mode only, which may optionally be added if scalability to 16 kHz sampling rate is required. The complete coder with bandwidth scalability consists of the CELP coder for 8 kHz sampling rate plus the bandwidth extension tool providing a single layer of scalability. It should be noted that the addition of this tool is not to be confused with the default 16 kHz sampling rate mode. With respect to the 8 kHz sampling rate mode, both configurations (8kHz sampling rate coder + bandwidth scalability and 16 kHz sampling rate coder) increase intelligibility and naturalness of the decoded speech by expanding the bandwidth to 7 kHz. If no bandwidth scalability is required, it is recommended to use the 16 kHz sampling rate coder.

Complexity Scalability: For 16 kHz sampling rate mode, it is possible to run the decoder at various complexity levels. Currently, 3 levels of complexity are defined. Two different methods of interpolation and the possibility of reading less LPC coefficients from the bit stream are described. These levels can be used to decode a bitstream at decreased complexity. The choice of these levels are discussed in detail in the decoding process (see section Lower Complexity Interpolation: Interpolation On LARs (Used in complexity levels C1 and C2)).

Complexity level	Description
C3	Improved LPC interpolation (LSP): Full decoding
C2	Simplified LPC interpolation (LAR)
C1	Reduced order LPC synthesis filter

Table 3. Possible Decoder complexity levels

These complexity levels allow different implementations and are not influenced in any way by the syntax. In the case of a software decoder, the complexity level can even be changed during run-time so as to cope with the

computational load on limited-capacity terminal or in a multi-tasking environment. Complexity level C2 is the mode with the best quality/complexity tradeoff.

2.2.2 Configuration of the MPEG-4 CELP coder

Apart from the bit-rate setting, the MPEG-4 CELP syntax offers the possibility of configuring the coder using 3 parameters namely

1. SampleRateMode: Configures the sampling rate for 8 or 16 kHz. The type of fixed codebooks used are Regular Pulse Excitation (RPE) for speech sampled at 16 kHz and Multipulse Excitation (MPE) for speech sampled at 8 kHz.
2. QuantizationMode: Configures the use of either a scalar quantizer (SQ) or a vector quantizer (VQ). Owing to the fact that different representation of the LPC can be transmitted, a tradeoff is often made between efficiency of transmission and the complexity of calculating the parameters that can be efficiently quantized to realize efficient transmission. Two such representations namely the Log Area Ratios (LARs) and the Line Spectral Pairs (LSPs) are incorporated in the MPEG-4 CELP coder. The scalar quantizer is based on uniform quantization of the LARs and the vector quantization is based on LSPs. Accordingly, there are two types of LPC decoders.
3. FineRateControl: Enables Fine step bit-rate control (permitting variable bit rate operation). This is achieved purely by controlling the transmission rate of the LPC parameters using a combinations of the two bit stream elements **interpolation_flag** and **LPC_present flag**. Using FineRateControl it is possible to generate any arbitrary bit rate between two anchor bit rates. These anchor bit rates are provided in Table 2 and Table 1.

The combination of these three parameters leads to eight possible modes illustrated in the figure below.

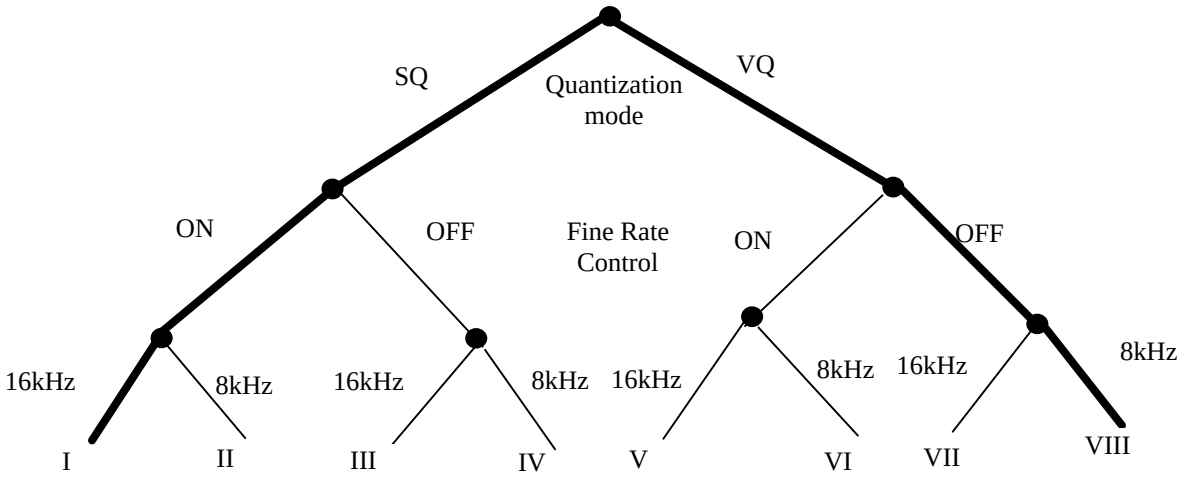


Figure 2. Possible modes in which the MPEG-4 coding scheme can operate

In order to clarify the various modes along with additional information about these modes, a short summary is provided below.

Mode I and Mode VIII are the default modes for 16 kHz and 8 kHz sampling rates respectively. These default modes are shown using bold lines in Figure 2.

Mode-I: This is the default mode for the MPEG-4 CELP coder, when a sampling rate of 16 kHz is chosen. In this mode, the coder uses a scalar quantizer and fine-rate control is turned on. In this mode, the coder is capable of generating arbitrary bit rate between 13.667-20.133 kbit/s and 21.8-24 kbit/s. The delay in this mode, is a function of the bit rate.

Bit rate (bit/s)	Delay (ms)
13667 – 15866	41.25
15867 – 18200	28.75
18201 – 20133	40.625
21800 – 24000	41.75

Table 4. Delays for the default 16 kHz mode

It should be pointed out that, for some specific bit rates, it is possible to achieve a much lower delay as explained in mode III (e.g. 18.2 kbit/s can be synthesized with a delay as low as 18.75 ms).

Mode VIII: This is the default mode for the MPEG-4 CELP coder, when a sampling rate of 8 kHz is chosen. In this mode, the coder uses a vector quantizer and fine-rate control is turned off. The bit rate and the corresponding delay figures are provided in Table 5.

Bit rate (bit/s)	Delay (ms)
3850, 4250, 4650	45
4900, 5200, 5500	35
5700, 6000, 6300, 6600, 6900, 7100, 7300, 7700, 8300, 8700, 9100, 9500, 9900, 10300, 10500, 10700	25
11000, 11400, 11800, 12000, 12200	15

Table 5. Delays for a sampling rate of 8 kHz, with fine-rate control disabled

Mode II: 8 kHz sampling rate with scalar quantizer and fine-rate is control turned on. In this mode it is possible to generate an arbitrary bit rate between 4250 and 12200 bit/s. This comes at an expense of higher delay (additional one frame delay is introduced w.r.t Mode-VIII).

Mode III: 16 kHz sampling rate with scalar quantizer and fine-rate control turned off. In this mode the bit rate is fixed, as opposed to mode I where fine-rate control is enabled. Now only a fixed number of bit rates are possible namely 13667, 15867, 18200, 20133, 21800 and 24000 bit/s. The advantage is that, for most of the bitrates supported in this mode, the coder has a lower delay. The delay figures are tabulated in Table 6.

Bit rate (bit/s)	Delay (ms)
13667	41.25
15867	26.25
18200	18.75
20133	25.625
21800	41.75
24000	26.75

Table 6. Delays for a sampling rate of 16 kHz, with fine-rate control disabled

For 15867, 18200, 20133, and 24000 bit/s, the LPC parameters are transmitted every frame. For the bit rates 13667 and 21800 bit/s, the LPC parameters are present every alternate frames. As a result of the latter, fixed bit rate is achieved over 2 frames (hence the delay in these modes is higher).

Mode IV: 8 kHz sampling rate with a scalar quantizer and fine-rate control is turned off. This mode is allows for a lower complexity than in mode VII because a scalar quantizer is used instead of a vector quantizer. As a result, the bit rate is higher. The bit rate mapping w.r.t modeVIII is tabulated in Table 7.

Bit rate (bit/s)	Delay (ms)
4325, 4725, 5125	45
5534, 5834, 6134	35
6650, 6950, 7250, 7550, 7850, 8050, 8250, 8650, 9250, 9650, 10050, 10450, 10850, 11250, 11450,	25

11650	
12900, 13300, 13700, 13900, 14100	15

Table 7. Delays for a sampling rate of 8 kHz with Rate Control turned off

Mode V: 16 kHz sampling rate with vector quantizer and fine-rate control is turned on. This is more complex compared to Mode-I. The advantage is that higher quality can be achieved for the same bit rate.

Bit rate (bit/s)	Delay (ms)
13267 – 15066	41.25
15067 – 17000	28.75
17533 – 19333	40.625
21400 – 23200	41.75

Table 8. Delays for a sampling rate of 16 kHz, with fine-rate control enabled

Mode VI: 8 kHz sampling rate with vector quantizer and fine-rate control enabled. In this mode it is possible to generate arbitrary bit rates between 3850 and 12200 bit/s. This comes at an expense of higher delay (additional one frame delay is introduced w.r.t Mode-VIII).

Mode VII: 16 kHz sampling rate with a vector quantizer and fine-rate control disabled. This is similar to mode mode V but only a fixed amount of bit rates are possible.

Bit rate (bit/s)	Delay (ms)
15067	26.25
17000	18.75
19333	25.625
23200	26.75

Table 9. Delays for a sampling rate of 16 kHz, with fine-rate control disabled

2.2.3 Additional functionality

Additionally, it is possible to

- Lower the bit rate without loss in quality (Lossless Coding) in those modes where a Scalar Quantizer is used (modes I and II). This reduces the bit rate on an average.
- Offer bandwidth scalability in Mode VIII. This makes it possible to increase intelligibility and naturalness of the decoded speech by expanding the bandwidth to 7kHz at the cost of higher complexity and increased delay.

Additional bit rates (bit/s)	Delay (ms)
+9200, +10400, +11600, +12400	50
+9467, +10667, +11867, +12667	40
+10000, +11200, +12400, +13200	30
+11600, +12800, +14000, +14800	20

2.3 Overview of MPEG-4 CELP configurations

The table below provides an overview of how to configure each mode, the required toolset and the supported functionalities.

Mode	I	II	III	IV	V	VI	VII	VIII
Sampling Rate(kHz)	16	8	16	8	16	8	16	8
Quantization	SQ	SQ	SQ	SQ	VQ	VQ	VQ	VQ
FineRateControl	ON	ON	OFF	OFF	ON	ON	OFF	OFF
Bit Rate Scalability		•		•		•		•
Complexity Scalability	•	•	•	•				
Lossless Coding	•	•						
Bandwidth Scalability								•
Variable Bit rate	•	•			•	•		
Excitation Type	RPE	MPE	RPE	MPE	RPE	MPE	RPE	MPE
Min Delay (ms)	28.75 41.75	25	18.75	15	28.75	25	18.75	15
Max Delay (ms)		85	41.75	45	41.75	85	26.75	45

Table 10. Summary of the functionalities in the MPEG-4 CELP modes

3. Bitstream Syntax

CelpSpecificConfig()

CELP Base Layer

The CELP core in the unscalable mode or as the base layer in the scalable mode requires the following CelpSpecificConfig():

```
class CelpSpecificConfig( uint(4) sampleRateIndex ){
        CelpHeader( sampleRateIndex );
}
```

CELP Enhancement Layer

The CELP core is used for both bitrate and bandwidth scalable modes. In the bitrate scalable mode, the enhancement layer requires no CelpSpecificConfig(). In the bandwidth scalable mode, the enhancement layer has the following CelpSpecificConfig():

```
class CelpSpecificConfig(){
        CelpBWSenhHeader();
}
```

Transmission of Celp bitstreams in AL-PDUs

Each scaleable layer of an MPEG-4 Celp audio bitstream is transmitted in Elementary Stream. In an **AL-PDU payload** the following dynamic data for CELP Audio has to be included:

CELP Base Layer

```
alPduPayload {
        CelpBaseFrame();
}
```

CELP Enhancement Layer

To parse and decode the CELP enhancement layer, information decoded from the CELP base layer is required. For the bitrate scalable mode, the following data for the CELP enhancement layer has to be included:

```
alPduPayload {
        CelpBRSenhFrame();
}
```

For the bandwidth scalable mode, the following data for the CELP enhancement layer has to be included:

```
alPduPayload {
        CelpBWSenhFrame();
}
```

3.1 Header Syntax

Syntax	No. of bits	Mnemonic
CelpHeader(sampleRateIndex)		
{		
QuantizationMode	**1**	**uimsbf**
FineRateControl	**1**	**uimsbf**
if ((QuantizationMode == ScalarQuantizer) &&		
(FineRateControl == ON)) {		
LosslessCodingMode	**1**	**uimsbf**
}		
if (sampleRateIndex==0x8) {		
SampleRateMode=8kHz		
WB_Configuration	**3**	**uimsbf**
Wideband_VQ	**1**	**uimsbf**
}		
if (sampleRateIndex==0xb) {		
SampleRateMode=16kHz		
NB_Configuration	**5**	**uimsbf**
NumEnhLayers	**2**	**uimsbf**
BandwidthScalabilityMode	**1**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
CelpBWSenhHeader()		
{		
BWS_configuration	**2**	**uimsbf**
}		

3.2 Frame Syntax

Syntax	No. of bits	Mnemonic
CelpBaseFrame()		
{		
Celp_LPC()		
if (SampleRateMode==8kHz) {		
NarrowBandframe()		
}		
if (SampleRateMode==16kHz) {		
WideBandframe()		
}		
}		

Syntax	No. of bits	Mnemonic
CelpBRSenhFrame()		
{		
for(subframe=0; subframe<nrof_subframes; subframe++){		
shape_enh_positions[subframe**][**enh_layer**]**	**4, 12**	**uimsbf**
shape_enh_signs[subframe**][**enh_layer**]**	**2, 4**	**uimsbf**
gain_enh_index[subframe**][**enh_layer**]**	**4**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
CelpBWSenhFrame()		
{		
BandScalable_LSP()		
for (subframe=0; subframe<nrof_subframe_bws; subframe++){		
shape_bws_delay[subframe**]**	**3**	**uimsbf**
shape_bws_positions[subframe**]**	**22, 26, 30, 32**	**uimsbf**
shape_bws_signs[subframe**]**	**6, 8, 10, 12**	**uimsbf**
gain_bws_index[subframe**]**	**11**	**uimsbf**
}		
}		

3.3 LPC Syntax

Syntax	No. of bits	Mnemonic
Celp_LPC()		
{		
if (QuantizationMode == ScalarQuantizer) {		
Interpolation_flag	**1**	**uimsbf**
LPC_Present	**1**	**uimsbf**
if (LPC_Present == YES) {		
if (FineRateControl == ON){		
if (LosslessCodingMode==ON) {		
LosslessCoded_LPC()		
} else {		
if (SampleRateMode==8kHz) {		
NarrowBandPacked_LPC()		
}		
if (SampleRateMode==16kHz) {		
WideBandPacked_LPC()		
}		
}		
} else {		
if (SampleRateMode==8kHz) {		
NarrowBandPacked_LPC()		
}		
if (SampleRateMode==16kHz) {		
WideBandPacked_LPC()		
}		
}		
}		
} else {		
if (FineRateControl == ON){		
Interpolation_flag	**1**	**uimsbf**
LPC_Present	**1**	**uimsbf**
if (LPC_Present == YES) {		
if (SampleRateMode==8kHz) {		
NarrowBand_LSP()		
}		
if (SampleRateMode==16kHz) {		
if (Wideband_VQ==Optimized_VQ) {		
Wideband_LSP()		
} else {		
Narrowband_LSP()		
BandScalable_LSP()		
}		
}		

```
        }
      } else {
        if (SampleRateMode==8kHz)  {
          NarrowBand_LSP()
        }
        if (SampleRateMode==16kHz)  {
          if (Wideband_VQ==Optimized_VQ) {
            WideBand_LSP()
          } else {
            Narrowband_LSP()
            BandScalable_LSP()
          }
        }
      }
    }
}
```

Syntax	No. of bits	Mnemonic
WideBandPacked_LPC()		
{		
LPC_indices[0]	**9**	**uimsbf**
LPC_indices[1]	**15**	**uimsbf**
LPC_indices[2]	**11**	**uimsbf**
LPC_indices[3]	**6**	**uimsbf**
LPC_indices[4]	**3**	**uimsbf**
LPC_indices[5]	**3**	**uimsbf**
LPC_indices[6]	**3**	**uimsbf**
LPC_indices[7]	**8**	**uimsbf**
LPC_indices[8]	**8**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
NarrowBand_LSP()		
{		
LPC_indices[0]	**4**	**uimsbf**
LPC_indices[1]	**4**	**uimsbf**
LPC_indices[2]	**7**	**uimsbf**
LPC_indices[3]	**6**	**uimsbf**
LPC_indices[4]	**1**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
NarrowBandPacked_LPC()		
{		
LPC_indices[0]	**12**	**uimsbf**
LPC_indices[1]	**12**	**uimsbf**
LPC_indices[2]	**12**	**uimsbf**
LPC_indices[3]	**3**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
BandScalable_LSP()		
{		
LPC_indices[5]	**4**	**uimsbf**
LPC_indices[6]	**7**	**uimsbf**
LPC_indices[7]	**4**	**uimsbf**
LPC_indices[8]	**6**	**uimsbf**
LPC_indices[9]	**7**	**uimsbf**
LPC_indices[10]	**4**	**uimsbf**
}		

Syntax	No. of bits	Mnemonic
LosslessCoded_LPC()		
{		
Statistics_Update	**1**	**uimsbf**
if (Statistics_Update)		
{		
for (lar_index=0; lar_index< LPC_order; lar_index++)		
{		
for (i=0; i<nrof_levels_lar[lar_index]; i++) {		
CodeLength[lar_index][i]	**4**	**uimsbf**
}		
}		
}		
for (lar_index=0; lar_index< LPC_order; lar_index++)		
{		
hufcod[htab[lar_index]][indices[lar_index]]	**1…16**	**uimsbf**
}		
}		

3.4 Excitation Syntax

Syntax	No. of bits	Mnemonic
WideBandframe()		
{		
for (subframe = 0; subframe < nrof_subframes; subframe++) {		
shape_delay[subframe**]**	**8**	**uimsbf**
shape_index[subframe**]**	**11,12**	**uimsbf**
gain_indices[0][subframe**]**	**6**	**uimsbf**
gain_indices[1][subframe**]**	**3,5**	**uimsbf**
}		
}		

Syntax	No. of bits	Mnemonic
NarrowBandframe()		
{		
signal_mode	**2**	**uimsbf**
rms_index	**6**	**uimsbf**
for(subframe=0; subframe<nrof_subframes; subframe++){		
shape_delay[subframe**]**	**8**	**uimsbf**
shape_positions[subframe**]**	**14 ... 32**	**uimsbf**
shape_signs[subframe**]**	**3 ... 12**	**uimsbf**
gain_index[subframe**]**	**6**	**uimsbf**
}		
}		

4. Semantics

This section describes the semantics of the syntactic elements. Bitstream elements are shown in **bold**-face and the help variables that appear in the syntax and are needed to extract the bitstream elements are shown in *italics*.

SampleRateMode a one bit identifier representing the sampling frequency. Two sampling frequencies are supported.

SampleRateMode	SampleRateID	Description
0	8kHz	8 kHz Sampling frequency
1	16kHz	16 kHz Sampling frequency

QuantizationMode a one bit identifier representing whether a scalar or a vector quantizer is used.

QuantizationMode	QuantizerID	Description
0	ScalarQuantizer	Scalar Quantizer is used
1	VectorQuantizer	Vector Quantizer is used

Wideband_VQ a one bit flag indicating whether scalable or optimized vector quantizer is used.

Wideband_VQ	VQ_ID	Description
0	Scalable_VQ	The scalable VQ is used
1	Optimized_VQ	(reserved)

While the optimized VQ is not supported in the current version, it is reserved for future use.

FineRateControl A one bit flag indicating whether fine rate control in very fine steps is enabled or disabled.

FineRateControl	RateControlID	Description
0	OFF	Fine-rate control is disabled
1	ON	Fine-rate control is enabled

LosslessCodingMode A one bit flag indicating whether the quantized LPC coefficients are lossless encoded or not.

LosslessCodingMode	LosslessID	Description
0	OFF	Lossless coding is not applied to LPC coefficients
1	ON	Lossless coding is applied to LPC coefficients

WB_Configuration This is a 3 bit identifier which configures the MPEG-4 CELP coder for a sampling frequency of 16 kHz. This parameter directly determines the anchor bit-rate allocation “WDB_RateX”. The anchor bit rates determine the coder configuration (Table 12) and the number of subframes in a CELP frame (Table 13).

WB_Configuration	Anchor bit rate
000	WDB_Rate1
001	WDB_Rate2
010	WDB_Rate3
011	WDB_Rate4
100	WDB_Rate5
101	WDB_Rate6
110	Reserved
111	Reserved

Table 11. Definition of Anchor bit rates for 16 kHz sampling rate

where WDB_RateX is dependent on the QuantizationMode as follows.

Rate Allocation	Scalar Quantizer	Vector Quantizer
WDB_Rate1	13667	13267
WDB_Rate2	15867	15067
WDB_Rate3	18200	17000
WDB_Rate4	20133	19333
WDB_Rate5	21800	21000
WDB_Rate6	24000	23200

Table 12. Rate allocation in the 16 kHz mode as a function of Quantization Mode

NB_Configuration This is a 5 bit field which configures the MPEG-4 CELP coder for a sampling frequency of 8 kHz. This parameter determines the variables nrof_subframes and nrof_subframes_bws, which are tabulated in Table 14 and Table 15. This parameter also specifies the number of bits for shape_positions[i], shape_signs[i], shape_enh_positions[i][j] and shape_enh_signs[i][j] tabulated in Table 16 and Table 17 respectively.

nrof_subframes is a help parameter, specifying the number of subframes in a CELP frame and is used to signal how many times the excitation parameters must be read. For the sampling rate of 16 kHz, this variable is dependent on the anchor bit rates (as defined in Table above) as follows:

Bit rate (kbit/s)	nrof_subframes
WDB_Rate1 <= Bit Rate <= WDB_Rate2	6
WDB_Rate2 < Bit Rate <= WDB_Rate3	4
WDB_Rate3 < Bit Rate <= WDB_Rate4	8
WDB_Rate5 <= Bit Rate <= WDB_Rate6	10

Table 13. Definition of nrof_subframes for 16 kHz

and for a sampling frequency of 8 kHz, it is derived from the NB_Configuration as follows:

NB_Configuration	nrof_subframes
0,1,2	4
3,4,5	3
6 ... 12	2
13 ... 21	4
22 ... 26	2
27 ... 31	reserved

Table 14. Definition of nrof_subframes for 8 kHz

NumEnhLayers This is a two bit field specifying the number of enhancement layers that are used.

NumEnhLayers	nrof_enh_layers
0	0
1	1
2	2
3	3

BandwidthScalabilityMode This is a one bit identifier which indicates whether bandwidth scalability is enabled. This mode is only valid when SampleRateMode = 8kHz and QuantizationMode = VectorQuantizer.

BandwidthScalabilityMode	ScalableID	Description
0	OFF	Bandwidth scalability is disabled
1	ON	Bandwidth scalability is enabled

BWS_Configuration This is a two bit field which configures the bandwidth extension tool. This identifier is only valid when BandwidthScalabilityMode = ON. This parameter specifies the number of bits for shape_bws_positions[i], shape_bws_signs[i].

nrof_subframes_bws This parameter, which is a help variable, represents the number of subframes in the bandwidth scalable layer is derived from the NB_Configuration as follows:

NB_Configuration	nrof_subframes_bws
0,1,2	8
3,4,5	6
6 ... 12	4
13 ... 21	4
22 ... 26	2
27 ... 31	reserved

Table 15 . Definition of nrof_subframes_bws

Interpolation_flag This is a one bit flag. When set, it indicates that the LPC parameters for the current frame must be derived using interpolation.

Interpolation_flag	InterpolationID	Description
0	OFF	LPC coefficients of the frame do not have to be interpolated
1	ON	LPC coefficients of the frame must be retrieved by interpolation

LPC_Present This bit indicates whether LPC parameters are attached to the current frame. These LPC parameters are either of the current frame or the next frame.

LPC_Present	LPCID	Description
0	NO	Frame does not carry LPC data
1	YES	Frame carries LPC data

Together the Interpolation_flag and the LPC_Present flag describe how the LPC parameters are to be derived.

Interpolation_flag	LPC_Present	Description
1	1	LPC Parameters of the current frame must be extracted using interpolation. The current frame carries LPC Parameters belonging to the next frame.
1	0	RESERVED
0	1	LPC Parameters of the current frame are present in the current frame.
0	0	LPC Parameters of the current frame were received in the previous frame.

LPC_order This parameter is a help variable which indicates the number of coefficients present in the bit stream. These coefficients will be referred to as LPC coefficients. *LPC_order* is not to be confused with

lpc_order, representing the number of coefficients used in the decoder for synthesis filtering and post filtering. The value of this field is dependent on the sampling frequency.

SampleRateMode	LPC_order	lpc_order
16kHz	20	14, 17 or 20
8kHz	10	10

Statistics_Update This one bit flag indicates the presence of Huffman tables to decode the lossless encoded LPC parameters. Statistics_Update is only defined if LosslessCodingMode = ON.

Statistics_Update	UpdateID	Description
0	NO	Huffman tables are not present
1	YES	Huffman tables are present

nrof_levels_lars This help variable indicates the number of bins that were used to collect the statistics of each LAR index.

lar_index	nrof_levels_lar
0	36
1	28
2	15
3	14
4	13
5	13
6	12
7	11
8	9
9	8

lar_index	nrof_levels_lar
10	8
11	7
12	7
13	8
14	7
15	6
16	6
17	7
18	6
19	6

CodeLength[i**][**j**]** This four bit field indicates the length of the code for LAR coefficient i and level number j.

hufcod[i**][**j**]** This 1…16 bits field represents the j^{th} entry in the i^{th} LAR table. When Statistics_Update = YES, these tables will be updated. The initial tables are provided in Annex B.

LPC_indices[] These are multi-bit fields representing LPC coefficients. When QuantizationMode = ScalarQuantizer, these indices contain information needed to extract the Log Area Ratios. When QuantizationMode = VectorQuantizer, these contain information needed to extract the LSP coefficients. The exact extraction procedure is described in the Decoding Process.

shape_delay[subframe**]** This is an 8 bit field representing the adaptive codebook lag. The decoding of this field depends on the sampling rate.

shape_index[subframe**]** This index contains information needed to extract the fixed codebook contribution of the regular pulse codebook. The number of bits consumed by this field depend on the bit rate (extracted from the configuration).

bit_rate	number of bits representing shape_index
if bit_rate>WDB_Rate3	12
else	11

gain_indices[0**][**subframe**]** These bit fields specify the adaptive codebook gain using 6 bits. It is read from the bitstream every subframe.

gain_indices[1**][**subframe**]** These bit fields specify the fixed codebook gain. It is read from the bitstream for every subframe. The number of bits read to represent this field depends on the subframe number. For the first subframe this is 5 bits and 3 bits for the remaining subframes.

gain_index[subframe**]** This 6 bit fields represents the gains for the adaptive codebook and the multi-pulse excitation in the CELP coder of 8 kHz.

gain_enh_index[subframe**]** This 4 bit fields represents the gain for the enhancement multi-pulse excitation in the CELP coder of 8 kHz.

gain_bws_index[subframe**]** This 11 bit fields represents the gains for the adaptive codebook and two multi-pulse excitation in the bandwidth extension tool.

signal_mode This 2 bits field represents the type of signal. This information is used for a sampling frequency of 8 kHz only. The gain codebooks are switched depending this information.

signal_mode	Description
0	Unvoiced
1,2,3	Voiced

rms_index This parameter indicates the rms level of the frame. This information is only utilized in the narrowband mode.

shape_positions[i], shape_signs[i] These bit-fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation. The length of the bit-field is dependent on NB_Configurations.

NB_Configuration	Shape_positions[i] (bits)	Shape_signs[i] (bits)
0	14	3
1	17	4
2	20	5
3	20	5
4	22	6
5	24	7
6	22	6
7	24	7
8	26	8
9	28	9
10	30	10
11	31	11
12	32	12
13	13	4
14	15	5
15	16	6
16	17	7
17	18	8
18	19	9
19	20	10
20	20	11
21	20	12
22	18	8
23	19	9
24	20	10
25	20	11
26	20	12
27 ... 31	reserved	

Table 16. Definitions of shape_positions[][] and shape_signs[][]

shape_enh_positions[i][j], shape_enh_signs[i][j] These bit-fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation in each enhancement layers. The length of the bit-field is dependent on NB_Configuration.

NB_Configuration	Shape_enh_positions[i][j] (bits)	Shape_enh_signs[i][j] (bits)
0 ... 12	12	4
13 ... 26	4	2
27 ... 31	reserved	

Table 17. Definition of shape_enh_positions[][] and shape_enh_signsp[][]

shape_bws_delay[i] This 3 bit field is utilized in decoding the adaptive codebook for the bandwidth extension tool. This value indicates the differential lag from the lag described in shape_delay.

shape_bws_positions[i], shape_bws_signs[i] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation in each enhancement layers. The length of the bit-field is dependent on BWS_Configuration. This is tabulated below.

BWS_Configuration	Shape_bws_positions[i] (bits)	Shape_bws_signs[i] (bits)
0	22	6
1	26	8
2	30	10
3	32	12

5. MPEG-4 CELP Decoder tools

This section provides a brief description of the functionality, parameter definition and the decoding processes of the tools supported by the MPEG-4 CELP core. The description of each tool comprises three parts and an optional part containing tables used by the tool:

1. Tool description: a short description of the functionality of the tool is given together with its interface.
2. Definitions: the input and output parameters as well as help elements of the tool are described here. Each element is either bold or italic. Bold names indicate that the element is read from the bitstream, italic names indicate auxiliary elements. If elements are already used by another tool, a reference to the previous definition is given.
3. Decoding process: The decoding process is explained here in detail with aid of mathematical equations and pseudo-C code.
4. Tables: this optional fourth part contains tables that are used by the tool.

5.1 General Introduction to the MPEG-4 CELP tool-set

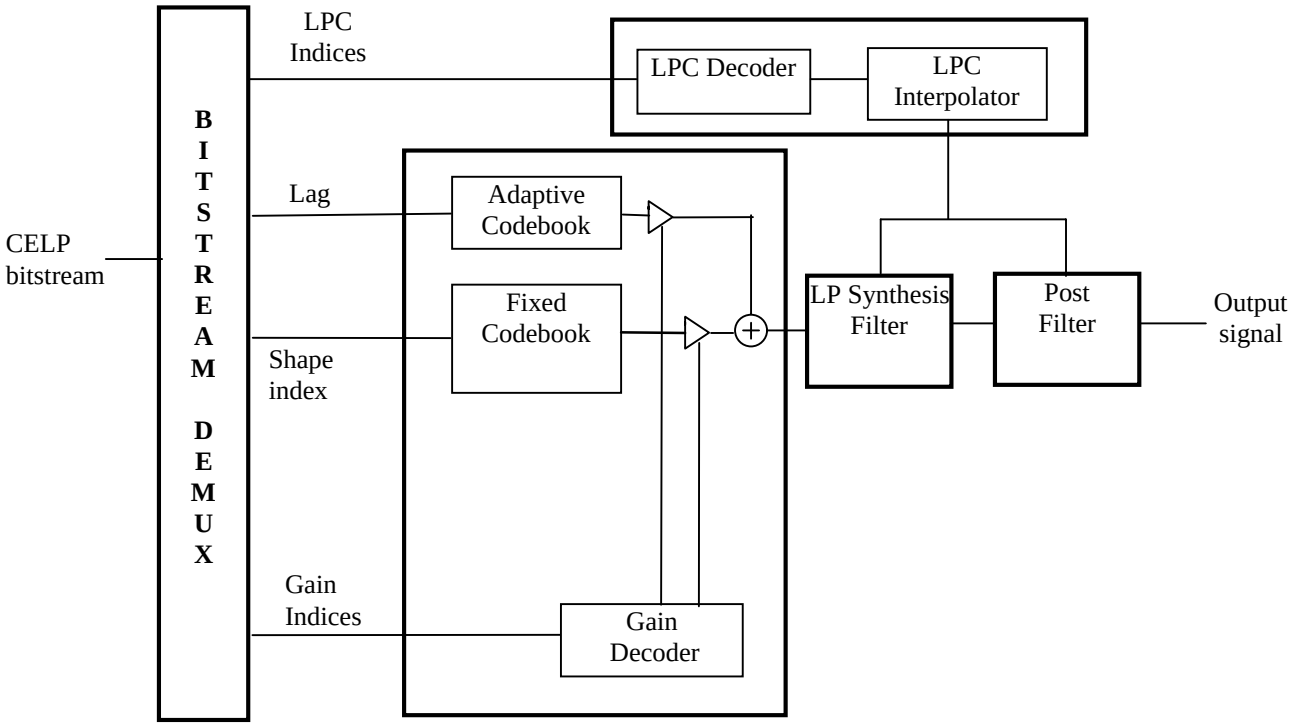


Figure 3. Block diagram illustrating the tools used in the MPEG-4 CELP coder

Figure 3 illustrates the MPEG-4 CELP decoder. It operates at a sampling rate of 16 or 8 kHz. One fixed and one adaptive codebook is used. One or more individual blocks have been grouped together (outlined in bold), forming the tools available for MPEG-4 CELP decoding. The following tools are supported:

- CELP bitstream demultiplexer
- CELP LPC decoder and interpolator
    - Scalar quantizer
    - Vector quantizer
    - Lossless decoder (optional)
    - Bandwidth scalable decoder (optional)
- CELP excitation generator
    - multi-pulse excitation
    - regular pulse excitation
    - bandwidth scalable multi-pulse excitation generation (optional)
- CELP LPC synthesis filter
- CELP post processor

The decoding is performed on a frame basis and each frame is divided into subframes. A frame in the bitstream is demultiplexed by the CELP bitstream demultiplexer module. The parameters that are extracted from the bitstream are header information, codes representing LPC coefficients of the frame and the excitation parameters for each subframe. These codes are decoded and interpolated for each subframe by the CELP LPC decoder and interpolator module. When the LosslessCodingMode = ON, the Huffman codes representing the LARs are routed through a lossless decoder prior to computing the LPC coefficients. For each subframe the excitation parameters are used to generate the excitation signal using the CELP excitation generator module. The CELP LPC synthesis filter module reconstructs the speech signal on a subframe basis from the interpolated LPC coefficients and the generated excitation signal. Enhancement of the synthesized signal is obtained by the CELP post processor module. When the CELP decoder is used as a “core coder” (see the T/F part), for large-step scalability, the postfilter is switched off.

5.2 Helping variables

Although each tool has a description of the variables it uses, provided in this section are the most commonly used variables shared by multiple tools.

frame_size: This field indicates the number of samples in a frame. The decoder outputs a frame with frame_size samples.

nrof_subframes: A frame is built up of a number of subframes. The number of subframes is specified in this field.

sbfrm_size: A subframe consists of a number of samples, which is indicated by this field. The number of samples in a frame must always be equal to the sum of the number of samples in the subframes. So, the following relation must always hold

*frame_size = nrof_subframes * sbfrm_size*

These three parameters depend on the sampling rate and the bit rate settings as tabulated in Table 18, for 16 kHz sampling rate and Table 19, for 8 kHz sampling rate.

Bit rate (kbit/s)	Frame_size (#samples)	nrof_subframes	sbfrm_size (#samples)
WDB_Rate1 <= Bit Rate <= WDB_Rate2	240	6	40
WDB_Rate2 < Bit Rate <= WDB_Rate3	160	4	40
WDB_Rate3 < Bit Rate <= WDB_Rate4	240	8	30
WDB_Rate5 <= Bit Rate <= WDB_Rate6	240	10	24

Table 18. CELP coder configuration for 16 kHz sampling rate, see Table 12 for the definition of the Anchor bit rates WDB_RateX

NB_Configuration	Frame_size (#samples)	nrof_subframes	sbfrm_size (#samples)
0,1,2	320	4	80
3,4,5	240	3	80
6 ... 12	160	2	80
13 ... 21	160	4	40
22 ... 26	80	2	40
27 ... 31	reserved		

Table 19. CELP coder configuration for 8 kHz sampling rate

lpc_order: This field indicates the number of coefficients used for Linear Prediction. By default, the value of this field is 20 for a sampling rate of 16 kHz and 10 for 8 kHz.

For a sampling rate of 16 kHz, the value of this field also depends on the decoder complexity level. For the lowest complexity level, i.e. level C1 (Table 3), reduced order filtering is enabled and the value of this field may be set to 14, 17 or 20. For all higher complexity levels the value of this field is fixed to 20.

For a sampling rate of 8 kHz, the value of this field is fixed to 10.

num_lpc_indices. This parameter specifies the number of indices containing LPC information, that must be read from the bitstream. This is not equal to the LPC-order. For a scalar quantizer, in the 16 kHz sampling rate mode, 20 LAR indices are packed in to 9 **LPC_indices**, thus *num_lpc_indices* equals 9, while for sampling rate of 8 kHz *num_lpc_indices* equals 4 (10 LAR indices packed in to 4 **LPC_indices**). For the Vector Quantizer the *num_lpc_indices* is 5 in the 8 kHz mode and an additional 6 for the band scalable layer.

For a sampling rate of 16 kHz, the value of this field also depends on the decoder complexity level. In the lowest complexity level (level C1), the value of this field is either 7, 8 or 9 depending on whether the order is 14, 17 or 20 respectively. If a higher complexity level is chosen, the value of this field is 9.

5.3 Bitstream elements for the MPEG-4 CELP tool-set

This section lists all the bitstream variables and in which tools they are used.

SamplingRate This field indicates the sampling frequency. For the MPEG-4 CELP coder, it can be either 8 kHz or 16 kHz.

QuantizationMode A one bit identifier representing whether a scalar or a vector quantizer is used.

Wideband_VQ A one bit flag indicating whether scalable or optimized vector quantizer is used. While the optimized VQ is not supported in the current version, it is reserved for future use.

FineRateControl a one bit flag indicating whether fine rate control is enabled or disabled.

LosslessCodingMode This field indicates whether the quantized LPC coefficients must be lossless decoded or not. This is only possible if QuantizationMode = ScalarQuantizer and FineRateControl = ON. In the lossless coding mode, the indices representing the Log Area Ratios have been further lossless encoded and put on the bitstream. Thus, either the LARs are packed before they are put on the bitstream or the LARs are lossless encoded before they are put on he bitstream.

LPC_indices These fields represent the LPC coefficients. When QuantizationMode = ScalarQuantizer, these indices contain information needed to extract the Log Area Ratios. When QuantizationMode = VectorQuantizer, these contain information needed to extract the LSP coefficients. The exact extraction procedure is described in the Decoding Process.

interpolation_flag: This is a one bit flag. When set, the flag indicates that the frame under consideration is an incomplete frame, i.e. the frame does not carry the LPC coefficients of the current speech frame, but only its excitation parameters (adaptive and fixed codebook parameters). The LPC coefficients for the speech frame under consideration should be obtained using interpolation of the LPC coefficients of the adjacent frames.

1 LPC coefficients of the frame must be retrieved by interpolation
0 LPC coefficients of the frame do not have to be interpolated.

For maintaining good subjective quality, there may never be more than one frame in succession without the LPC information, i.e. the interpolation_flag may not have the value 1 in two successive CelpFrames.

LPC_Present This field indicates the presence of LPC parameters in the speech frame under consideration. These LPC coefficients are either of the current speech frame or those of a subsequent frame. When used in combination with the interpolation_flag**,** the two parameters completely describe how the LPC coefficients of the current frame are derived. If the interpolation flag is set, the LPC coefficients of the current frame are calculated by using the LPC coefficients of the previous and next frame. Generally, this would mean that the decoding of the current frame must be delayed by one frame. To avoid this additional delay in the decoder, the LPC coefficients of the next frame are enclosed in the current frame. In this case, the LPC_Present flag is set. Since the LPC coefficients of the next frame are already present in the current frame, the next frame will not contain LPC information.

- If the interpolation_flag is “1” and the LPC_Present is “1”, then (a) the LPC parameters of the current frame are derived using the LPC parameters of the previous frame and that of the next frame and (b) the current frame (frame under consideration) carries LPC parameters of a subsequent frame, but not those of the frame under consideration. The LPC coefficients of the frame under consideration are obtained by interpolation of the previously received LPC parameters and the LPC parameters received in the frame under consideration.
- If the interpolation_flag is “0” and the LPC_Present is “0”, the LPC parameters to be used with the frame under consideration are those received in the previous frame.
- When interpolation_flag is “0” and the LPC_Present is “1”, then the current frame is a complete frame and the LPC parameters received in the current frame belongs to the current frame.

Such a construction is chosen so as to minimize the delay when the decoder begins reconstructing the frame, the LPC coefficients of which, are obtained using interpolation without having to wait for the next frame to arrive. Secondly, such a combination makes it possible to decode the bitstream from any point (random access). In the fixed bit rate configuration, these two flags exhibit a fixed pattern

01, 01, 01, 01, 01, 01, … The string 01 is repeated
11, 00, 11, 00, 11, 00, … The string 11, 00 is repeated (Fixed bit rate achieved over two frames)
For variable bit rate (when FineRateControl = ON), the string will, generally, not exhibit a fixed pattern.

NumEnhLayers The number of enhancement layers that are used. This parameter is only valid for SampleRateMode=8kHz.

BandwidthScalabilityMode This is a one bit identifier which indicates whether bandwidth scalability is enabled. This mode is only valid when SampleRateMode = 8kHz and QuantizationMode = VectorQuantizer.

BWS_Configuration This is a two bit field which configures the bandwidth extension tool. This identifier is only valid when BandwidthScalabilityMode = ON. This parameter specifies the number of bits for shape_bws_positions[i], shape_bws_signs[i].

Statistics_Update This one bit flag indicates the presence of Huffman tables to decode the lossless encoded LPC parameters. Statistics_Update is only defined if LosslessCodingMode = ON.

Statistics_Update	UpdateID	Description
0	NO	Huffman tables are not present
1	YES	Huffman tables are present

CodeLength[i][j] This four bit field indicates the length of the code for LAR coefficient i and level number j.

hufcod[i][j] This 1…16 bits field represents the j^{th} entry in the i^{th} LAR table. When Statistics_Update = YES, these tables will be updated. The initial tables are provided in Annex B.

shape_delay This is an 8 bit field representing the adaptive codebook lag. The decoding of this field depends on the sampling rate.

shape_index This index contains information needed to extract the fixed codebook contribution of the regular pulse codebook.

signal_mode This 2 bits field represents the type of signal. This information is used for a sampling frequency of 8 kHz only. The gain codebooks are switched depending on this information.

signal_mode	Description
0	Unvoiced
1,2,3	Voiced

rms_index This parameter indicates the rms level of the frame. This information is only utilized for 8 kHz sampling rate.

shape_positions[i], shape_signs[i] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation.

shape_enh_positions[i][j], shape_enh_signs[i][j] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation in each enhancement layer.

shape_bws_delay[i] This field is utilized in decoding the adaptive codebook for the bandwidth extension tool. This value indicates the differential lag from the lag described in shape_delay.

shape_bws_positions[i], shape_bws_signs[i] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation in each enhancement layers.

gain_indices[0] The adaptive codebook gain. It is read from the bitstream for every subframe.

gain_indices[1] These fields specify the fixed codebook gain. It is read from the bitstream for every subframe.

gain_index These fields represent the gains for the adaptive codebook and the multi-pulse excitation in the CELP coder of 8 kHz. It is read from the bit-stream for every sub-frame.

gain_enh_index These fields represent the gain for the enhancement multi-pulse excitation in the CELP coder of 8 kHz. It is read from the bit-stream for every sub-frame.

gain_bws_index These fields represents the gains for the adaptive codebook and two multi-pulse excitation in the bandwidth extension tool. It is read from the bit-stream for every sub-frame.

5.4 CELP bitstream demultiplexer

5.4.1 Tool description

The tool CELP bitstream demultiplexer demultiplexes a received frame from the bitstream.

5.4.2 Definitions

All the bitstream elements and the associated help variables have been defined in Section Semantics.

5.4.3 Decoding process

The decoding of the bitstream elements is accordance with the Syntax described in Section Bitstream Syntax.

5.5 CELP LPC decoder and interpolator

5.5.1 Tool description

The tool CELP LPC decoder and interpolator has two functions:

1. Retrieve LPC coefficients from the **LPC_indices**.
2. Interpolate the retrieved LPC coefficients for every subframe.

Depending on the quantization mode, the LPC_indices contain information needed to retrieve either the Log Area Ratios (Scalar Quantizer) or LSP Coefficients (Vector Quantizer). Thus, the retrieval process is divided into two parts namely LAR Decoding Process and LSP Decoding process. The outputs of this process are the filter coefficients, called LPC coefficients, or a-parameters that can be used in the direct form filter.

5.5.2 Definitions

Input

lpc_indices[]: The dimension of this array is num_lpc_indices and contains the packed lpc indices (see section 5.4.2).

interpolation_flag: This flag indicates whether the LPC parameters for the current frame must be derived using interpolation (see section Bitstream elements for the MPEG-4 CELP tool-set).

LPC_Present This flag indicates whether LPC parameters are attached to the current frame (see section Bitstream elements for the MPEG-4 CELP tool-set).

Output

int_Qlpc_coefficients[]: This array contains the LPC coefficients for each subframe. The LPC coefficients are quantized and interpolated as described in the decoding process. The LPC coefficients are stacked one after the other in blocks of lpc_order. Thus, the dimension of the array is lpc_order * nrof_subframes.

Configuration

lpc_order: This field indicates the order of LPC being used (see section Helping variables).

num_lpc_indices: This field contains the number of packed LPC codes (see section Helping variables). For the LAR decoding process, the variable num_lpc_indices is set to 9 for 16 kHz sampling rate and 4 for 8 kHz sampling rate. For the LSP decoding process, num_lpc_indices is set to 5 for 8kHz sampling rate and 11 if the bandwidth scalable layer is used (BandwidthScalabilityMode = ON).

nrof_subframes: This field contains the number of subframes (see section Helping variables).

nrof_subframes_bws: This parameter, which is a help variable, represents the number of subframes in the bandwidth scalable layer. Only needed if Bandwidth scalable part is used (BandwidthScalabilityMode = ON).

Help elements used in LAR decoding process are summarized below:

rfc_table[]	look-up-table for representation of lpc_indices
rfc_offset[]	table containing offset in rfc_table
Convert2lpc()	function for converting LSPs to LPCs
Convert2lsp()	function for converting LPCs to LSPs
prev_coeffs[]	array containing the LPC coefficients of the previous frame
cur_coeffs[]	array containing the LPC coefficients of the current frame
next_coeffs[]	array containing the LPC coefficients of the next frame

5.5.3 Decoding Process

5.5.3.1 LAR Decoding Process

The LAR decoding process is dependent on whether LosslesCodingMode is ON or OFF and the decoder complexity level. This is summarized in Figure 4. When LosslessCodingMode = ON, the Huffman codes are decoded by a lossless decoder, and when it is OFF the LPC_indices are unpacked. The output of either of these is indices[]. These are then further decoded using a look-up table. Finally, depending on the decoder complexity level, the interpolation is either performed on the LARs or on the LSPs.

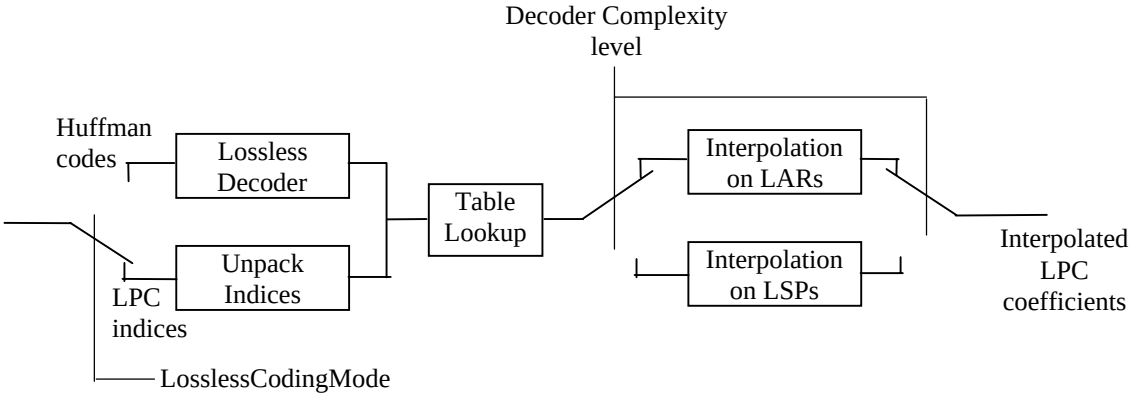


Figure 4. Options for retrieving LPC Coefficients in the Scalar QuantizationMode

In this section, the decoding process with “LosslessCodingMode = OFF” is described first as this is the default mode. It is noted there are possibilities to lower the decoding further and these are described in a separate section (Possibilities to obtain Lower Complexity).

5.5.3.1.1 Retrieving LPC Parameters

The LAR decoding process for retrieving interpolated LPC coefficients for each subframe is built up of a number of stages. Next, each of these stages will be discussed.

Step I: Determine order of LPC

The order of LPC is dependent on the sampling frequency, being 20 for the 16 kHz sampling rate and 10 for the 8kHz sampling rate. The number of coefficients extracted from the LPC_indices is equal to the lpc_order.

Step II: Indices unpacking

If the LPC indices are present (LPC_Present = YES), then the `indices[]` are computed in the following way. For 16 kHz sampling rate, the `indices[]` are derived as follows:

```
indices[0]   = ( lpc_indices[0] / 14);
indices[1]   = ( lpc_indices[1] / 1155);
indices[2]   = ((lpc_indices[1] / 77) MOD 15);
indices[3]   = ( lpc_indices[0] MOD 14);
indices[4]   = ( lpc_indices[2] / 156);
indices[5]   = ((lpc_indices[2] / 12) MOD 13);
indices[6]   = ( lpc_indices[2] MOD 12);
indices[7]   = ((lpc_indices[1] / 7) MOD 11);
indices[8]   = ( lpc_indices[3] / 7);
indices[9]   = ( lpc_indices[4]);
indices[10]  = ( lpc_indices[5]);
indices[11]  = ( lpc_indices[1] MOD 7);
indices[12]  = ( lpc_indices[3] MOD 7);
indices[13]  = ( lpc_indices[6]);
indices [14] = ( lpc_indices [7] / 36);
indices [15] = ((lpc_indices [7] / 6) MOD 6);
indices [16] = ( lpc_indices [7] MOD 6);
indices [17] = ( lpc_indices [8] / 36);
indices [18] = ((lpc_indices [8] / 6) MOD 6);
indices [19] = ( lpc_indices [8] MOD 6);
```

For 8 kHz sampling rate, the `indices[]` are derived from the LPC_indices as follows:

```
indices [0]  = ( lpc_indices [0] / 108);
indices [1]  = ( lpc_indices [1] / 143);
indices [2]  = ( lpc_indices [2] / 182);
indices [3]  = ((lpc_indices [2] / 13) mod 14);
indices [4]  = ((lpc_indices [1] / 11) mod 13);
indices [5]  = ( lpc_indices [2] mod 13);
indices [6]  = ((lpc_indices [0] /  9) mod 12);
indices [7]  = ( lpc_indices [1] mod 11);
indices [8]  = ( lpc_indices [0] mod 9);
indices [9]  = ( lpc_indices [3] );
```

Step III: Table Look Up

Using these indices (which actually represent uniformly quantized Log Area Ratios (LARs)) reflection coefficients can be directly obtained from a look-up table (i.e, the mapping of LARs to reflection coefficients has already been performed so that the index directly points to the corresponding reflection coefficient). The look-up procedure is defined as follows:

```
for (k=0; k < lpc_order; k++)
{
    rfc[k] = rfc_table[rfc_offset[k]+indices[k]];
}
```

Both `rfc_table[]` and `rfc_offset[]` are tabulated in Table 20 and Table 21 respectively.

If the LPC indices are not present in the current frame (LPC_Present = NO), the indices have been received and decoded in the previous frame. Thus, only interpolation needs to be performed to compute the direct form filter coefficients for the subframes of the current frame.

5.5.3.1.2 Interpolating the LPC Parameters

Once the indices are extracted from the packed LPC indices, they are used to calculate the LPC coefficients for (each subframe of) the current frame. Therefore, the unpacked LPC indices have to be transformed to a representation of LPC coefficients that is more suitable for interpolation. It is conventional to use the same representation of the LPC coefficients for interpolation as is used in the encoder. Thus, even though LARs may be used in the encoder for interpolation, the use of LSPs in the decoder enhances the quality of the reconstructed speech, especially when LPC coefficients of a frame are frequently retrieved via interpolation of LPC coefficients of adjacent frames. Depending on the specified complexity level, one of the following representations of the LPC coefficients can be used for performing interpolation:

1. Line Spectral Pairs (LSPs)
2. Log Area Ratios (LARs)

In the highest complexity level, i.e. level C3 (See Table 3), the interpolation is performed on the LSPs. In lower complexity levels, the decoding is performed on the Log Area Ratios. This helps to provide complexity scalability in the decoder.

5.5.3.1.2.1 Lower Complexity Interpolation: Interpolation On LARs (Used in complexity levels C1 and C2)

Step I: Conversion (reflection coeffs to LARs)

If the complexity level indicates that interpolation on LARs has to be performed, the reflection coefficients that are stored in array rfc are transformed to LARs. This transformed is done for k = 0, 1, …, lpc_order-1 in the following way:

$$cur\_coeffs[k] = \log\left(\frac{1+rfc[k]}{1-rfc[k]}\right)$$

Step II: Interpolation (on LARs)

If the interpolation flag is set, it is an indication that the coefficients of the current frame have to be interpolated using the coefficients of the adjacent frames. The LPC coefficients of the adjacent frames are not interpolated. As already described, if the interpolation flag is set, the LPC coefficients of the next frame are received in the current frame. This means that the coefficients stored in array cur_coeffs are the coefficients belonging to the next frame. The LPC coefficients of the next frame are stored in array next_coeffs in order to use them in the next frame. The coefficients belonging to the current frame are computed by linear interpolation.

```
for (k=0; k < lpc_order; k++)
{
    next_coeffs[k] = cur_coeffs [k];
    cur_coeffs[k] = (prev_coeffs[k] + next_coeffs [k]) / 2;
}
```

If the interpolation flag is not set and the packed LPC indices were present in the current frame, the computed coefficients belong to the current frame and no interpolation has to be performed.

If the interpolation flag is not set and the packed LPC indices were not present in the current frame, the LPC coefficients are already received in the previous frame and stored in the array next_coeffs. Therefore, the array next_coeffs has to be copied to the array cur_coeffs.

Using the coefficients of the current frame and the previous frame, the coefficients of each subframe are computed by linear interpolation and put block-wise in the array int_coeffs.

```
for (n = 0; n < nrof_subframes; n++)
{
  for (k =0; k < lpc_order; k++)
  {
     int_coeffs[n*lpc_order+k] = ((n_subframe-n-1) * prev_coeffs[k] +
                                  (1+n)*cur_coeffs[k])/nrof_subframes;
  }
}
```

Step III: Conversion to Direct-form filter coefficients (from LARs)

When the interpolation is to be performed using LARs, the conversion of LARs to LPC coefficients is performed in two steps. First the interpolated LARs are transformed in reflection coefficients:

```
for (n = 0; n < nrof_subframes; n++)
{
    for (k=0; k < lpc_order; k++) {
         rfc[n*lpc_order+k] = (1-e^int_coeffs[n*lpc_order+k]) / (1+e^int_coeffs[n*lpc_order+k]);
    }
}
```

The conversion from reflection coefficients to direct-form filter coefficients is described next. The direct form filter coefficients, more generally known as a-parameters (represented using alpha), are calculated using the reflection coefficients (for i = 1, 2, …, lpc_order):

$$\alpha_i^{(i)} = rfc[i-1]$$

$$\alpha_j^{(i)} = \alpha_j^{(i-1)} - rfc[i-1] \cdot \alpha_{i-j}^{(i-1)}, 1 \le j \le i-1$$

The a-parameters or LPC coefficients of the subframes are stacked one after the other in array int_Qlpc_coefficients.

5.5.3.1.2.2 Higher Quality Interpolation: Interpolation On LSPs (Used in complexity level C3)

Step I: Conversion (reflection coeffs to LSPs)

First the reflection coefficients are converted to a-parameters (as described above).These a-parameters are then converted to LSPs using the utility function *Convert2lsp()*. The conversion of a-parameters to LSPs is discussed in [2]. After the conversion, the array cur_coeffs contains the LSPs that are used for interpolation.

Step II: Interpolation (on LSPs)

Interpolation is performed exactly as discussed in Section Lower Complexity Interpolation: Interpolation On LARs (Used in complexity levels C1 and C2) above. The only difference is that the arrays used for interpolation contain LSPs instead of LARs

Step III: Conversion to Direct-form filter coefficients (from LSPs)

The interpolated coefficients are then converted back to a-parameters, called LPC coefficients, to be used in the synthesis filter.

In the highest complexity mode of the decoder, the LSP representation of the LPC coefficients are used for interpolation. Accordingly, the conversion of LSPs to LPC coefficients is done for each subframe using the auxiliary function *Convert2lpc()*:

```
for (n = 0; n < nrof_subframes; n++)
{
    Convert2lpc(lpc_order, int_coeffs + n*lpc_order, int_Qlpc_coefficients
                                                      + n*lpc_order);
}
```

This conversion is described in [2].

Storing the coefficients

After the calculation of the LPC coefficients, the current LPC coefficients have to be stored is memory, since they are used for interpolation.

```
for (k=0; k < lpc_order; k++) {
     prev_coeffs[k] = cur_coeffs[k];
}
```

5.5.3.1.3 Lossless Decoding of the LAR indices

5.5.3.1.3.1 Tool description

In addition to the LAR quantizer, this is a tool that one can use if further reduction in bit rate is required. This can only be used in combination with the Scalar Quantizer. Whether this tool has been used is indicated by setting the LosslessCodingMode = ON.

5.5.3.1.3.2 Definitions

Input

Statistics_Update This parameter indicates whether the Huffman tables, representing the lossless encoded LPC indices, have to be updated.

CodeLength[i][j] For each LAR i, the array contains the number of Huffman code-word lengths i. This array is used to generate the Huffman tables.

hufcod [i][j] This 1…16 bits field represents the jth entry in the ith LAR table. When Statistics_Update = YES, these tables will be updated. The initial tables are provided in Annex B.

htab[i] Huffman table entry [i]

Output

indices[] These are the Huffman decoded LAR indices.

configuration

nr_levels_las: This parameter indicates the number of bins that where used to collect the statistics of each LAR index.

The array *nr_levels_lar* is defined as follows:

Lar coefficient index	*nr_levels_lar*
0	36
1	28
2	15
3	14

Lar coefficient index	*nr_levels_lar*
10	8
11	7
12	7
13	8

4	13
5	13
6	12
7	11
8	9
9	8

14	7
15	6
16	6
17	7
18	6
19	6

Both the encoder and the decoder use the same procedure to build a Huffman table for each LAR. These tables are built using **CodeLength[][]** as follows:

Consider the a LAR table with Lar coefficient index number 11 which has 7 entries, each entry corresponding to a level. The information that is transmitted to the decoder are the lengths of the codes associated with the levels. The minimum codeword length is 1 (0000 means length 16). As an example consider the following table:

Entry for Lar coefficient 11 (7 levels)	**CodeLength[11][]**	Binary representation of **CodeLength[11][]**
0	4	0100
1	4	0100
2	3	0011
3	2	0010
4	3	0011
5	4	0100
6	4	0100

The decoder now calculates the number of codewords of each length:

Code length	Number of occurrences
2	1
3	2
4	4

The following codewords are subsequently formed, (for each length the first codeword is the first possible in the Huffman tree, starting at 0):

Code length	Huffman code word
2	00
3	010
3	011
4	1000
4	1001
4	1010
4	1011

Finally these codewords, in this order, are allocated to the entries, according to their length. As can been seen from the table, equal length codes are assigned in the order of the index.

Index	Length	Codeword, htab[11]
0	4	1000
1	4	1001
2	3	010
3	2	00
4	3	011
5	4	1010
6	4	1011

Using these Huffman tables, decoding is performed. The initial CodeLength[][] and the resulting Huffman tables are provided in Annex B. New tables are transmitted and generated only when the **StatisticsUpdate** flag is set. For the rest of the frames, the latest version of the Huffman tables is used to decode the incoming Huffman codes.

5.5.3.1.3.3 Decoding Process

The string **hufcod**[htab[i]][j] means that the LAR table 'i'. is used to decode the bitstream. The result is assigned to 'j', where 'j' are the decoded indices: indices[i]=j. In other words indices[] are obtained directly as a result of the Huffman decoding.

The rest of the steps (interpolation and conversion to the direct-form filter coefficients) are same as described in Section LAR Decoding Process.

When decoding starts at an arbitrary position in the bit stream and lossless_coding_mode is set to 1, it can happen that no statistics information is present at the decoder side. If Statistics_Update has not been called prior to or at the same time lossless_coding_mode is 1, default tables are generated according to the procedure above for decoding the LAR's. These tables are dependent on the number of possible levels for each LAR coefficient.

5.5.3.1.4 Possibilities to obtain Lower Complexity

It is possible to obtain very low complexity decoding using the CELP decoder. When the sampling rate is 16 kHz, the lpc_order is set to 20 and for 8 kHz sampling rate the lpc_order is set to 10. However, for 16 kHz sampling rate it is possible to lower the order of LPC and as a result the computational load due to LPC Synthesis filtering reduces significantly. The reduction in complexity is proportional to the reduction in lpc_order. The order can be reduced to either 17 or 14 (complexity reduction up to 30%). This reduction in order is only possible in the lowest complexity level, i.e. level C1 (Table 3). Should this be desired, then the last 3 or the last 6 indices can be ignored and the order of LPC must be reduced accordingly. The rest of the decoding process remains unchanged. For 8 kHz sampling rate the lpc_order remains fixed at 10.

As explained in the demultiplexing process, it is possible that the LPC_indices that have been received in the current frame actually represent the LPC_indices of the next frame, in which case the LPC coefficients of the current frame are extracted from the LPC_indices of the adjacent frames using interpolation. As explained earlier, due to fact that the LPC_indices of the frame to come can be received in the current frame, there is no additional delay in the decoder.

Quantization Tables for the Scalar Quantizer (Mode I, II, III and IV)

Index	rfc_table	Index	rfc_table
0	-0.9896	25	0.4621
1	-0.9866	26	0.5546
2	-0.9828	27	0.6351
3	-0.9780	28	0.7039
4	-0.9719	29	0.7616
5	-0.9640	30	0.8093
6	-0.9540	31	0.8483
7	-0.9414	32	0.8798
8	-0.9253	33	0.9051
9	-0.9051	34	0.9253
10	-0.8798	35	0.9414
11	-0.8483	36	0.9540
12	-0.8093	37	0.9640
13	-0.7616	38	0.9719
14	-0.7039	39	0.9780
15	-0.6351	40	0.9828
16	-0.5546	41	0.9866
17	-0.4621	42	0.9896
18	-0.3584	43	0.9919
19	-0.2449	44	0.9937

20	-0.1244	45	0.9951
21	0.0000	46	0.9961
22	0.1244	47	0.9970
23	0.2449	48	0.9977
24	0.3584		

Table 20. Representation table of the reflection coefficients

Index	rfc_offset	Index	rfc_offset
0	13	10	18
1	0	11	17
2	16	12	19
3	12	13	17
4	16	14	19
5	13	15	18
6	16	16	19
7	14	17	17
8	18	18	19
9	16	19	18

Table 21. Offsets in rfc_table

5.5.3.2 LSP Decoding Process

5.5.3.2.1 LSP Decoding

The following are help elements used in the LSP decoding process:

lsp_tbl[][][]	look-up tables for the first stage decoding process
d_tbl[][][]	look-up tables for the second stage decoding process of the VQ without interframe prediction
pd_tbl[][][]	look-up tables for the second stage decoding process of the VQ with interframe prediction.
sign	sign of code vector for the second stage decoding process
idx	unpacked index for the second stage decoding process
lsp_predict[]	the LSPs predicted from *lsp_previous[]* and *lsp_first[]*
lsp_previous[]	the LSPs decoded at the previous frame
lsp_current[]	the LSPs decoded at the current frame
lsp_first[]	the LSPs decoded at the first stage decoding process
lsp_subframe[][]	the LSPs interpolated at each subframe
ratio_predict	prediction ratio for predicting *lsp_predict[]*
ratio_sub	interpolation ratio for calculating *lsp_subframe[][]*
Convert2lpc()	function for converting LSPs to LPCs
dim[][]	dimensions for the split vector quantization

The LSP decoding process for retrieving interpolated LPC coefficients for each subframe is described below.

5.5.3.2.2 Converting indices to LSPs

The LSPs of the current frame (*lsp_current[]*), which are coded by split and two-stage vector quantization, are decoded with a two-stage decoding process. The dimension of each vector is described in Table 22 and Table 23. The **lpc_indices[0]**,**[1]** and **lpc_indices[2]**,**[3]** represent indices for the first and the second stage respectively.

Split Vector Index: i	Vector Dimension: dim[0][i]
0	5
1	5

Table 22. Dimension of the first stage LSP vector

Split Vector Index: i	Vector Dimension: dim[1][i]
0	5
1	5

Table 23. Dimension of the second stage LSP vector

In the first stage, the LSP vector of the first stage *lsp_first[]* is decoded simply by looking up the table *lsp_tbl[][][]* . (The table *lsp_tbl[][][]* is shown in Annex B)

```
for(i = 0; i < dim[0][0]; i++)
{
   lsp_first[i] = lsp_tbl[0][lpc_indices[0]][i];
}

for(i = 0; i < dim[0][1]; i++)
{
   lsp_first[dim[0][0]+i] = lsp_tbl[1][lpc_indices[1]][i];
}
```

In the second stage, there are two types of decoding processes, namely, decoding process of VQ without interframe prediction and VQ with interframe prediction. The **lpc_indices[4]** indicates which process should be selected.

LPC Index: lpc_indices[4]	Decoding process
0	VQ without interframe prediction
1	VQ with interframe prediction

Table 24. Decoding process for the second stage

5.5.3.2.3 Decoding process of VQ without interframe prediction

In order to obtain LSPs of the current frame *lsp_current[]* , the decoded vectors in the second stage are added to the decoded first stage LSP vector *lsp_first[]*. The MSB of the **lpc_indices[]** represents the sign of the decoded vector, and the remaining bits represent the index for the table *d_tbl[][][].* (The table *d_tbl[][][]* is shown in Annex B)

```
sign = lpc_indices[2]>>6;
idx = lpc_indices[2]&0x3f;
if(sign==0) {
   for(i=0;i<dim[1][0];i++){
      lsp_current[i] = lsp_first[i] + d_tbl[0][idx][i];
   }
}else {
    for(i=0;i<dim[1][0];i++) {
      lsp_current[i] = lsp_first[i] - d_tbl[0][idx][i];
   }
}
sign = lpc_indices[3]>>5;
idx = lpc_indices[3]&0x1f;
if(sign==0) {
   for(i=0;i<dim[1][1];i++){
      lsp_current[dim[1][0]+i] = lsp_first[dim[1][0]+i] + d_tbl[1][idx][i];
   }
}else {
    for(i=0;i<dim[1][1];i++) {
      lsp_current[dim[1][0]+i] = lsp_first[dim[1][0]+i] - d_tbl[1][idx][i];
   }
```

```
}
```

5.5.3.2.4 Decoding process of VQ with interframe prediction

In order to obtain LSPs of the current frame *lsp_current[]* , the decoded vectors of the second stage are added to the LSP vector *lsp_predict[]* which are predicted from the decoded LSPs of the previous frame *lsp_previous[]* and the decoded first stage LSP vector *lsp_first[]*. Same as the decoding process of VQ without interframe prediction, the MSB of the **lpc_indices[]** represents the sign of the decoded vector, and the remaining bits represent the index for the table *pd_tbl[][][]*. (The table *pd_tbl[][][]* is shown in Annex B)

```
for(i=0;i<lpc_order;i++){
   lsp_predict[i]=(1-ratio_predict)*lsp_first[i]
     +ratio_predict*lsp_previous[i]
}
```

where *ratio_predict* = 0.5

```
sign = lpc_indices[2]>>6;
idx = lpc_indices[2]&0x3f;
if(sign==0) {
   for(i=0;i<dim[1][0];i++){
     lsp_current[i] = lsp_predict[i] + pd_tbl[0][idx][i];
   }
}else {
    for(i=0;i<dim[1][0];i++) {
     lsp_current[i] = lsp_predict[i] - pd_tbl[0][idx][i];
   }
}
sign = lpc_indices[3]>>5;
idx = lpc_indices[3]&0x1f;
if(sign==0) {
   for(i=0;i<dim[1][1];i++){
     lsp_current[dim[1][0]+i]
         = lsp_predict[dim[1][0]+i] + pd_tbl[1][idx][i];
   }
}else {
    for(i=0;i<dim[1][1];i++) {
     lsp_current[dim[1][0]+i]
         = lsp_predict[dim[1][0]+i] - pd_tbl[1][idx][i];
   }
}
```

5.5.3.2.5 Stabilization of LSPs

The decoded LSPs *lsp_current[]* are stabilized in order to ensure stability of the LPC synthesis filter which is derived from the decoded LSPs. The decoded LSPs are arranged in ascending order, having a minimum distance between adjacent coefficients.

```
   for(i=0;i<lpc_order;i++) {
     if(lsp_current[i] < min_gap) lsp_current[i] = min_gap;
   }
   for(i=0;i<lpc_order-1;i++) {
     if(lsp_current[i+1]-lsp_current[i] < min_gap) {
        lsp_current[i+1] = lsp_current[i]+min_gap;
     }
   }
   for(i=0;i<lpc_order;i++) {
     if(lsp_current[i] > 1-min_gap) lsp_current[i] = 1-min_gap;
   }
   for(i=lpc_order-1;i>0;i--) {
     if(lsp_current[i]-lsp_current[i-1] < min_gap) {
```

```
        lsp_current[i-1] = lsp_current[i]-min_gap;
      }
   }
```

where min_gap = 6.0/256.0

5.5.3.2.6 Interpolation of LSPs

The decoded LSPs *lsp_current[]* are interpolated linearly at each subframe using the LSPs of the previous frame *lsp_previous[]*.

```
for(n=0;n<nrof_subframes;n++){
   ratio_sub=(n+1)/nrof_subframes
   for(i=0;i<lpc_order;i++){
      lsp_subframe[n][i]
        = ((1-ratio_sub)*lsp_previous[i]+ratio_sub*lsp_current[i]))
   }
}
```

5.5.3.2.7 LSP to LPC Conversion

The interpolated LSPs are converted to the LPCs using the auxiliary function *Convert2lpc()*.

```
for(n=0;n<nrof_subframes;n++){
   Convert2lpc(lpc_order, lsp_subframe[n],
        int_Qlpc_coeffiecients + n*lpc_order]);
}
```

5.5.3.2.8 Storing the coefficients

After the calculation of the LPC coefficients, the current LSPs have to be stored in memory, since they are used for interpolation at the next frame.

```
for (i=0;i<lpc_order;i++) {
     lsp_previous[i] = lsp_current[i];
}
```

It must be noted that the stored LSPs *lsp_previous[]* must be initialized as described below when whole the decoder is initialized.

```
for (i=0;i<lpc_order;i++) {
     lsp_previous[i] = (i+1) / (lpc_order+1);
}
```

5.5.3.3 Bandwidth Scalable LSP Decoding Process

5.5.3.3.1 Tool description

The LSP decoding tool in the bandwidth extension tool is utilized to add bandwidth scalability to the 8 kHz sampling rate mode. This scalability enhancement is allowed only for VIII. This tool is connected with the LSP decoding tool in the narrowband mode. The LPC coefficients for each subframes are reconstructed by retrieving **lpc_indices[]** and converting the decoded LSPs in the narrowband mode. The following help elements are used in the decoding procedure. The LSP tables for this tool are given in Annex B.

lsp_bws_tbl[][][] look-up tables for bandwidth scalable LSP decoding process

lsp_bws_buf[][] buffer containing LSP prediction residual
bws_ma_prdct[][] prediction coefficients for moving average prediction
bws_nw_prdct[] prediction coefficients for conversion for LSPs from narrowband to wideband
lsp_current[] the decoded LSPs in the narrowband CELP coder
lsp_bws_current[] the decoded LSPs at the current frame in the bandwidth extension tool
lsp_bws_previous[] the decoded LSPs at the previous frame in the bandwidth extension tool
lpc_order_bws(=20) the order of LPC in the bandwidth extension tool . This is twice the lpc_order in narrowband CELP.

5.5.3.3.2 Decoding process

The LPC coefficients at each subframe are reproduced using **lpc_indices[5], ... , lpc_indices[10]** and the current LSPs (*lsp_current[]*) for the narrowband CELP coder. The decoding procedure consists of three steps, decoding LSPs at last subframe, interpolation for the other subframes and conversion of LSPs to LPC coefficients.

The LSPs at the last subframe (*lsp_bws_current[]*) for the bandwidth extension tool are reconstructed by table look up and prediction as follows:

```
for ( i=0;i<5;i++) {
    lsp_bws_buf[0][i] =
lsp_bws_tbl[0][lpc_indices[5]][i]+lsp_bws_tbl[2][lpc_indices[7]][i]
}
for ( i=5;i<10;i++) {
    lsp_bws_buf[0][i] =
lsp_bws_tbl[0][lpc_indices[5]][i]+lsp_bws_tbl[3][lpc_indices[8]][i-5]
}
for ( i=10;i<15;i++) {
    lsp_bws_buf[0][i] = lsp_bws_tbl[1][lpc_indices[6]][i-
10]+lsp_bws_tbl[4][lpc_indices[9]][i-10]
}
for ( i=15;i<20;i++) {
    lsp_bws_buf[0][i] = lsp_bws_tbl[1][lpc_indices[6]][i-
10]+lsp_bws_tbl[5][lpc_indices[10]][i-15]
}
for ( n=0;n<=2;n++) {
    for (i=0;i<20;i++) {
       lsp_bws_current[i] += bws_ma_prdct[n][i]*lsp_bws_buf[n][i]
   }
}
for (i=0;i<10;i++) {
   lsp_bws_current[i] += bws_nw_prdct[i]*lsp_current[i]
}
```

where *lsp_bws_tbl[][][]* are LSP codebooks listed in Annex B. *bws_ma_prdct[][]* and *bws_nw_prdct[]* are prediction coefficients for moving average prediction and conversion for LSPs from narrowband to wideband, respectively. *lsp_bws_buf[][]* is a buffer containing LSP prediction residual at the current frame and previous two ones. This buffer is shifted for the next frame operation as follows:

```
for (n = 2; n > 0; n--)
{
  for (i = 0; i < 20; i++)
  {
     lsp_bws_buf[n][i] = lsp_bws_buf[n-1][i];
  }
}
```

The decoded LSPs are interpolated linearly at each subframe.

```
for(n = 0; n < nrof_subframes_bws; n++)
{
  ratio_sub = (n+1)/nrof_subframes_bws
  for(i = 0; i < 2*lpc_order; i++)
```

```
   {
      lsp_bws_subframe[n][i]=((1-ratio_sub)*lsp_bws_previous[i]
                              + ratio_sub*lsp_bws_current[i]))
   }
}

for(i=0;i<2*lpc_order;i++)
{
   lsp_bws_previous[i] = lsp_bws_subframe[nrof_subframes_bws-1][i]
}
```

The interpolated LSPs is converted to the LPC coefficients at each subframes.

```
for(n=0;n<nrof_subframes_bws;n++)
{
   Convert2lpc(lpc_order_bws,lsp_bws_subframe[n],&int_Qlpc_coeffiecients[n*l
pc_order_bws]);
}
```

5.6 CELP excitation generator

5.6.1 Tool description

The tool CELP excitation generator generates the excitation signal for one subframe from the received **gain_indices**, **shape_index** and **shape_delay** of that subframe. For 16 kHz sampling rate, regular pulse excitation (RPE) is used (Regular Pulse Excitation Process), whereas for 8 kHz sampling frequency, multi pulse excitation (MPE) generation (Multi-Pulse Excitation Process) is used.

5.6.2 Decoding Process

5.6.2.1 Regular Pulse Excitation Process

Figure 5 illustrates the excitation generation process used for 16 kHz sampling rate. The excitation signal is constructed from a periodic component (adaptive codebook contribution) and non periodic component (RPE contribution) scaled by their respective gains. Using the **shape_delay[***sub_frame***]** and the **gain_indices[0][***sub_frame***]**, the adaptive codebook contribution is computed. The RPE contribution is computed by using **shape_index[***sub_frame***]** and **gain_indices[1][***sub_frame***]**. For clarity, the indexing based on *sub_frame* is dropped. It is noted that the excitation generation process is repeated every subframe.

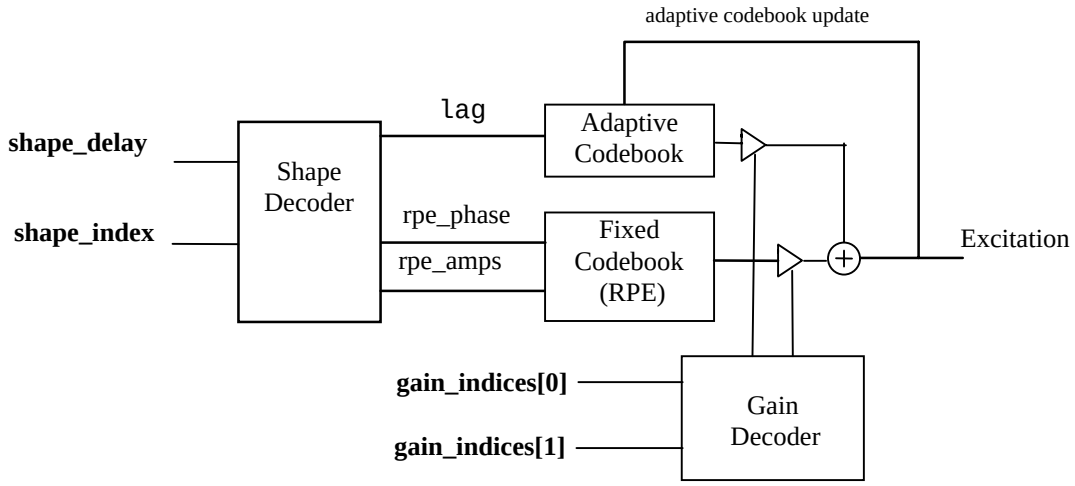


Figure 5. Excitation generation used for 16 kHz sampling rate

5.6.2.1.1 Definitions

Input

shape_delay[]: This array is of dimension *nrof_subframes* and contains the adaptive codebook lag (see section 5.4.2).

shape_index[]: This array is of dimension *nrof_subframes* and contains the RPE codebook index (see section 5.4.2).

gain_indices[0][]: This array is of dimension *nrof_subframes* and contains the adaptive codebook gain index (see section 5.4.2).

gain_indices[1][]: This array is of dimension *nrof_subframes* and contains the RPE codebook gain (see section 5.4.2).

int_Qlpc_coefficients[]: This is an array of dimension lpc_order and contain the quantized and interpolated LPC coefficients of one subframe. This is array is a sub-array of int_Qlpc_coefficients as specified in section 5.5.2. This field is used for 8 kHz sampling frequency only.

Output

excitation[]: This array is of dimension sbfrm_size and contains the excitation signal. This signal is reconstructed from shape and gain vectors using the adaptive and fixed codebooks.

lag: This field contains the decoded lag (pitch period) for adaptive codevector.

adaptive_gain: This field contains the decoded gain for adaptive codevector.

Configuration

lpc_order: This field indicates the order of LPC that is used (see section Helping variables). This field is used for 8 kHz sampling rate only.

sbfrm_size: This field indicates the number of samples in the subframe (see section Helping variables).

nrof_subframes: This field indicates the number of subframes (see section Helping variables).

The additional elements used in the RPE excitation mode are as follows:

tbl_cba_gain[]	look-up-table for adaptive codebook gain
tbl_cbf_gain[]	look-up-table for fixed codebook gain
tbl_cbf_gain_dif []	look-up-table for fixed codebook gain difference
cba[]	adaptive codebook
prev_Gf	fixed codebook gain of the previous subframe

5.6.2.1.2 Shape Decoder

This block describes the extraction of Adaptive codebook lag and the RPE parameters. The adaptive codebook lag is derived from the **shape_delay** in the following way:

```
lag = Lmax - Lmin - shape_delay
```

where Lmax and Lmin are the maximum and minimum lag, resp 295 and 40. The RPE parameters, namely the RPE phase (`rpe_phase`) and the RPE amplitudes (`rpe_amps`) are derived as follows:

```
    rpe_index = shape_index;
    rpe_phase = rpe_index MOD D
    rpe_index = rpe_index /D;
    for (n = Np - 1; n >= 0; n--)
    {
        rpe_amps[n]  = (rpe_index MOD 3) - 1;
        rpe_index   = rpe_index / 3;
    }
```

5.6.2.1.3 Gain Decoder

This block derives the scalar quantized adaptive and fixed codebook gains from the gain indices. The adaptive codebook gain Ga is determined by a table look-up on `cba_gain[](`Table 26):

```
if (gain_indices[0] > 31)
{
  Ga = -1 * cba_gain[((64 - gain_indices[0]) - 1)];
}
else
{
  Ga = cba_gain[gain_indices[0]];
}
```

The decoding of the fixed codebook gain depends on the subframe under consideration. The gain vector is used for retrieving the fixed codebook gain Gf . For the first subframe in a frame the gain is decoded with:

```
    Gf = cbf_gain[gain_indices[1]];
```

where cbf_gain[] is provided in Table 27. For all later subframes the gain is decoded using cbf_gain_dif[] provided in Table 28:

```
    Gf = cbf_gain_dif[gain_indices[1]] * prev_Gf;
```

where prev_Gf is the decoded gain of the previous subframes.

5.6.2.1.4 Adaptive Codebook Generation

First the generation of the excitation signal due to the adaptive codebook will be described. The excitation for the adaptive codebook, ya, is computed using the shape vector:

```
for (n = 0; n < sbfrm_size; n ++)
{
  ya[n] = cba[lag + n];
}
```

5.6.2.1.5 Fixed Codebook Generation

The excitation for the fixed codebook is computed in two steps. To generate the fixed codebook, we also need two extra parameters, namely D (the pulse spacing or the decimation factor) and Np the number of pulses. The value of these are dependent on the bit-rate and are tabulated in Table 25.

Bit rate (kbit/s)	D	Np
WDB_Rate1 <= Bit Rate <= WDB_Rate3	8	5
WDB_Rate3 < Bit Rate <= WDB_Rate4	5	6
WDB_Rate5 <= Bit Rate <= WDB_Rate6	4	6

Table 25. Allocation of pulse spacing and number of pulses in RPE

Once the phase and amplitudes are known, the amplitudes are placed on a regular grid. The amplitudes can have 3 different values: -1, 0, 1. The excitation for the fixed codebook, yf, is computed using the RPE-formula:

```
for (n = 0; n < sbfrm_size; n++)
{
    yf[n] = 0;
}
for (n = 0; n < Np; n++)
{
    yf[phase + D*n] = amp[n];
}
```

5.6.2.1.6 Excitation Generation

The excitation is the sum of adaptive and fixed excitation multiplied by their corresponding gains:

```
for (n = 0; n < sbfrm_size; n++)
{
    excitation[n] = Ga * ya[n] + Gf * yf[n];
}
```

Finally, the adaptive codebook has to be updated using the excitation. This is done by shifting the adaptive codebook by sbfrm_size entries and filling the empty entries with the calculated excitation.

```
for (n=sbfrm_size; n < Lmax; n++)
{
    cba[n-sbfrm_size] = cba[n];
}
for (n=0; n < sbfrm_size; n++)
{
    cba[Lmax-1-n] = excitation[sbfrm_size - 1 - n];
}
```

The adaptive codebook is initialized by filling it with zeros.

Index	cba_gain	Index	cba_gain
0	0.0945	16	0.9728
1	0.2121	17	1.0256
2	0.2936	18	1.0849
3	0.3597	19	1.1538
4	0.4183	20	1.2394
5	0.4705	21	1.3465
6	0.5186	22	1.4907
7	0.5652	23	1.7012
8	0.6110	24	2.0636
9	0.6559	25	2.8398
10	0.7000	26	5.4233
11	0.7445	27	10.847
12	0.7884	28	21.693
13	0.8329	29	43.386
14	0.8779	30	86.772
15	0.9242	31	173.54

Table 26. Representation table for Adaptive codebook gain

Index	Cbf_gain	Index	cbf_gain
0	2.0	16	76.4025
1	2.8267	17	88.6170

2	3.6320	18	102.7842
3	4.8605	19	117.6620
4	6.4876	20	135.2277
5	8.3800	21	154.7980
6	10.7813	22	178.1373
7	13.5654	23	204.7267
8	17.0676	24	239.0199
9	21.1388	25	280.5263
10	26.2852	26	334.0282
11	32.1299	27	408.3218
12	39.0193	28	505.7104
13	46.9503	29	665.3362
14	55.9060	30	1026.0
15	65.7851	31	reserved

Table 27. Representation table for Fixed codebook gain

Index	cbf_gain_dif
0	0.1000
1	0.3912
2	0.6614
3	0.8954
4	1.1651
5	1.5580
6	2.7000
7	6.5000

Table 28. Representation table for differential fixed codebook gain

5.6.2.2 Multi-Pulse Excitation Process

For 8 kHz sampling rate, the excitation signal is constructed from a periodic component (adaptive codebook vector) and non-periodic component (fixed codebook vector) scaled by their respective gains. Both the adaptive and the fixed codebook vectors are decoded from **shape_delay, shape_positions, shape_signs**. The gains are decoded from three types of indices, **signal_mode**, **rms_index** and **gain_index**.

5.6.2.2.1 Definitions

Input

shape_delay[]: This array is of dimension num_cbks and contains the adaptive codebook lag (see section 5.4.2).

shape_positions[]: This array is of dimension num_cbks and contains the pulse-positions (see section 5.4.2).
shape_signs[]: This array is of dimension num_cbks and contains the pulse-signs (see section 5.4.2).

gain_index[]: This array is of dimension num_cbks and contains the adaptive codebook gain index and the fixed codebook gain index (see section 5.4.2).

rms_index: This field defines the index for the signal power. This field is used for 8 kHz sampling rate only.

signal_mode: This field contains the voiced/unvoiced flag. This field is used for the 8 kHz sampling rate mode only.

int_Qlpc_coefficients[]: This is an array of dimension lpc_order and contain the quantized and interpolated LPC coefficients of one subframe. This is array is a sub-array of int_Qlpc_coefficients as specified in section 5.5.2. This field is used for 8 kHz sampling frequency only.

Output

excitation[]: This array is of dimension sbfrm_size and contains the excitation signal. This signal is reconstructed from shape and gain vectors using the adaptive and fixed codebooks.

acb_delay: This field contains the decoded lag for adaptive codevector. This field is used for 8 kHz sampling rate only.

adaptive_gain: This field contains the decoded gain for adaptive codevector. This field is used for 8 kHz sampling rate only.

configuration

lpc_order: This field indicates the order of LPC that is used (see section Helping variables). This field is used for 8 kHz sampling rate only.

sbfrm_size: This field indicates the number of samples in the subframe (see section Helping variables).

nrof_subframes: This field indicates the number of subframes (see section Helping variables).

The additional elements used in the MPE mode are as follows:

RMSTable[]	look-up-table for the signal power
AdaptGainCB[]	look-up-table of a gain for the periodic component
FixedGainCB[]	look-up-tables of a gain for the non-periodic component
AdaptExtCB[]	look-up-table of a excitation for the periodic component
FixedExtCB[][][]	look-up-tables of a excitation for the non-periodic component
ac_ex[]	decoded excitation signal as the periodic component
sc_ex[]	decoded excitation signal as the non-periodic component
qacg	decoded gain for the periodic component
qscg	decoded gain for the non-periodic component
rms[]	decoded root mean squares of the speech signal
ref[]	reflection coefficients converted from the **int_Qlpc_coefficients[]**
resid	prediction residual energy of a subframe
pow_ac	energy of *ac_ex[]*
pow_sc	energy of *sc_ex[]*
frac	fractional lag
sgn[]	signs of two codevectors for the non-periodic component
idx[]	indices for the FixedExtCB[][]

5.6.2.2.2 Decoding of Signal_mode

Signal_mode represents one of four modes for each frame. Modes 0 and 1 correspond to unvoiced and transition frames, respectively. Modes 2 and 3 correspond to voiced frames, and the latter indicates higher pitch periodicity than the former. This information is utilized in decoding the frame energy, the multi-pulse excitation and gains.

5.6.2.2.3 Decoding of Frame Energy

The root mean squared (rms) value at last subframe is reconstructed using **signal_mode** and **rms_index**. The rms value is decoded in the μ-law scale. μ-law parameters are dependent on **signal_mode**. The rms values of other subframes are obtained by linear-interpolating the decoded rms values at the last subframe of the current and the previous frames. The quantized rms values are used for gain normalization.

```
delt = 1.0 / 64;
aa = 1.0 / log10(1.0 + mu_law);
bb = rms_max / mu_law;
```

```
pwk = aa * log10(1.0 + pqxnorm / bb);
qwk = delt*(rms_index+1);
for (i = 0; i < n_subframes; i++ )
{
      nwk = (qwk-pwk)*(i+1)/n_subframes + pwk;
      qxnorm[i] = bb * (pow((double)10.0,(nwk/aa)) - 1.0);
}
pqxnorm = qxnorm [n_subframes-1];
```

Signal_mode	rms_max	mu_law
0	7932	1024
1, 2, 3	15864	512

5.6.2.2.4 **Decoding of the adaptive codebook vector**

The integer and the fractional parts of the pitch delay are obtained from **shape_delay**. The mapping table between **shape_delay** and the pitch delay is shown in Table 29. The adaptive codebook vector *acb[n]* is computed by interpolating the past excitation signal *pacb[n]* at the decoded integer delay *acb_delay* and fraction *acb_frac*. The interpolation is done using a FIR filter *int_fil[k], k=0,...,36* based on a Hamming-windowed sinc function.

```
for (n = 0; n < sbfrm_size; n++ )
{
  tt += acb_frac;
  kt = acb_delay + tt / 6;
  tt = tt % 6;
  for (i = 0; i < kt && n < sbfrm_size; i++, n++ )
  {
    for ( k = -6; k <= 6; k++ )
    {
      kk = (k+1) * 6 - tt;
      acb[n] += int_fil[abs(kk)] * pacb[150-(kt-i+k)];
    }
  }
}
```

where the filter coefficients are represented as follows:

```
int_fil[0] = 1.0;
for (k = 1; k <= 6 * 6; k++)
{
  int_fil[k] = sin(PAI * k / 6) / (PAI * k / 6) *
                (0.54 + 0.46 * cos(2 * PAI * k / (2 * 6 * 6)));
}
```

shape_delay	*acb_delay*	*acb_frac*
0 – 161	**shape_delay**/3+17	(2*__shape_delay__)%6
162 – 199	(**shape_delay**- 162)/2+71	(3*(**shape_delay**-162))%6
200 – 254	**shape_delay**-200+90	0
255	0	0

Table 29. The mapping table between the shape_delay and the pitch delay

5.6.2.2.5 **Decoding of the fixed codebook vector**

The fixed codebook vector contains several non-zero pulses and is represented by the pulse position and pulse amplitudes. The pulse positions *pul_pos[i]* are extracted from **shape_positions**. The pulse amplitudes *pul_amp[i]* are obtained from **shape_signs**.

```
for (i = num_pulse-1, k = 0; i >= 0; i-- )
```

```
{
  for ( j = 0; j < num_bit_pos[i]; j++ )
  {
    pos_idx[i] |= ((shape_positions>>k)&0x1)<<j;
    k++;
  }
  pul_amp[i] = 1.0;
  if (((shape_signs>>(num_pulse-1-i))&0x1) == 1 )
  {
    pul_amp[i] = -1.0;
  }
  pul_pos[i] = pos_tbl[i][pos_idx[i]];
}
```

NB_Configuration	num_pulse	NB_Configuration	num_pulse	NB_Configuration	num_pulse
0	3	10	10	20	11
1	4	11	11	21	12
2	5	12	12	22	8
3	5	13	4	23	9
4	6	14	5	24	10
5	7	15	6	25	11
6	6	16	7	26	12
7	7	17	8	27 ... 31	reserved
8	8	18	9		
9	9	19	10		

where *num_pulse* is the number of pulses and is set from **NB_Configuration**. *num_bit_pos[i]* is the number of bits for encoding the *i* th pulse position. *pos_tbl[i][j]* is the restriction table, which indicates the possible positions for each pulse. The table indicates the possible positions for each pulse. The table is set up in accordance with the combination of *subfrm_size*, *num_pulse* and *num_bit_pos[]* as follows:

```
step = subfrm_size / min_num_bit_pos;
for (i = 0; i < num_pulse; i++)
{
  m = 1 << (num_bit_pos[i]-min_num_bit_pos);
  for ( j = 0, k = 0; k < m; )
  {
    ch[j] = i;
    k++;
    j += (long)((float)step/m + 0.5);
    j = j % step;
  }
}

for (i = 0; i < num_pulse; i++ )
{
  for ( l=0,k = 0; k < step; k++ )
  {
    if (i == ch[k])
    {
      for ( j = 0; j < min_num_bit_pos; j++ )
      {
        pos[i][l++] = k + step * j;
      }
    }
  }
}
```

For example, the pulse position table used in 6 kbit/s narrowband coder is shown in Table 30.

Pulse number: i	Num_bit_pos[i]	Pulse positions: pos_tbl[i][j]
0	4	0,5,10,15,20,25,30,35, 40,45,50,55,60,65,70,75
1	4	1,6,11,16,21,26,31,36, 41,46,51,56,61,66,71,76
2	4	2,7,12,17,22,27,32,37 42,47,52,57,62,67,72,77
3	3	3,13,23,33,43,53,63,73
4	3	4,14,24,34,44,54,64,74
5	3	8,18,28,38,48,58,68,78
6	3	9,19,29,39,49,59,69,79

Table 30. The pulse position table for 6kbit/s coder

The fixed codebook vectors *fcb[n]* is computed from *pul_pos[i]* and *pul_amp[i]* as follows:

```
for (n = 0; n < sbfrm_size; n++)
{
  fcb[n] = 0.0;
}
for (i = 0; i < num_pulse; i++)
{
  fcb[pul_pos[i]] = pul_amp[i];
}
```

If the integer delay *acb_delay* is less than the subframe size *sbfrm_size*, the fixed codebook vector signal *fcb[n]* is modified by zero-state-combfiltering as follows:

```
for (n = 0; n < sbfrm_size; n++)
{
  if ( n - acb_delayt >= 0 )
  {
    ix = fcb[n - acb_delay];
  }
  else
  {
    ix = 0.0;
  }
  fcb[n] += cga[flag] * ix;
}
```

where *cga[4]={0.0,0.0,0.6,0.8}* are the gains of the comb-filter. *flag* is decoded from **signal_mode**.

5.6.2.2.6 Decoding of the adaptive and the fixed codebook gains

The **gain_index** is converted to the normalized gains *nga, ngf* for the adaptive and the fixed codebook vectors by looking up the gain table. The gain table is changed in accordance with **signal_mode** and *sbfrm_size*. The gain tables for this tool are given in Annex B. The adaptive codebook gain *ga* and the fixed codebook gain *gf* are calculated as follows:

```
ga = nga * sqrt( norm / acb_energy );
gf = ngf * sqrt( norm / fcb_energy );
```

where the *acb_energy* and *fcb_energy* are the energy for the adaptive codebook and fixed codebook vectors. The *norm* is the normalization factor.

```
norm = (qxnorm*subfrm_size)*(qxnorm*subfrm_size);
for(i = 0; i < lpc_order; i++)
{
  norm *= (1 - par[i] * par[i]);
}
```

where *par[]* are the reflection coefficients and calculated from the LP coefficients *int_Qlpc_coefficients[]*. The *qxnorm* is the quantized subframe energy, which is decoded from the **rms_indx** (see section 5.6.3.2.2).

5.6.2.2.7 **Excitation signal generation**

The excitation signal (*excitation[]*) is calculated by summing *acb[]* and *fcb[]* scaled by *ga and gf* respectively.

```
for(i=0;i<sbfrm_size;i++)
{
  excitation[i] = ga*acb[i] + gf*fcb[i]
}
```

5.6.2.2.8 **Pitch Pre Filtering**

The excitation signal *excitation[]* is modified to enhance pitch harmonics as follows:

```
gp = 0.4 * ga;
if (gp > 0.4)
{
  gp = 0.4;
}
for (n = 0; n < sbfrm_size; n++)
{
  for (k = -6; k <= 6; k++ )
  {
    kk = (k+1) * 6 - acb_frac;
    ix += int_fil[abs(kk)] * pacb[150-(acb_delay-i+k)];
  }
  excitation[n] += gp * ix;
}
```

5.6.2.2.9 Enhancement excitation generation tool

The bitrate scalable decoder is realized by adding the enhancement excitation decoding layers to the 8 kHz sampling rate mode. This scalability enhancement is allowed only for Mode VI and VIII. This tool is connected with the multi-pulse excitation generation tool in the 8 kHz sampling rate mode. The enhancement excitation signal is reconstructed by retrieving **shape_enh_positions, shape_enh_ signs, gain_enh_index** and utilizing the decoded multi-pulse excitation signal in the 8 kHz sampling rate mode.

5.6.2.2.9.1 Decoding of the enhancement fixed codebook vector

The enhancement fixed codebook vector also consists of several non-zero pulses. The pulse positions and amplitudes are extracted from **shape_enh_positions, shape_enh_signs** by the same decoding algorithm as the fixed codebook. The enhancement fixed codebook vectors enh_f*cb[n]* is computed from *pul_pos[i]* and *pul_amp[i]* as follows:

```
for (i = num_pulse_enh-1, k = 0; i >= 0; i-- )
{
  for ( j = 0; j < num_bit_pos[i]; j++ )
  {
    pos_idx[i] |= ((shape_enh_positions>>k)&0x1)<<j;
    k++;
  }
  pul_amp[i] = 1.0;
  if (((shape_enh_signs>>(num_pulse_enh-1-i))&0x1) == 1 )
  {
    pul_amp[i] = -1.0;
  }
  pul_pos[i] = pos_tbl[i][pos_idx[i]];
}

for (n = 0; n < sbfrm_size; n++)
{
  fcb[n] = 0.0;
}
for (i = 0; i < num_pulse_enh; i++)
{
  fcb[pul_pos[i]] = pul_amp[i];
}
```

sbfrm_size	*num_pulse_enh*
40 (5ms)	2
80 (10ms)	4

The pulse position table used in 2 kbit/s enhancement is adaptively controlled so that the enhancement pulses stand at different positions from those of the pulses selected in the core excitation. The pulse position table is temporarily generated by the same decoding algorithm as the fixed codebook. The temporary pulse position table is modified as follows,

```
for ( n = 0; n < num_enh; n++ )
{
  for ( i = num[n]-1, k = 0; i >= 0; i-- )
  {
    pul_loc = 0;
    for ( j = 0; j < bit_pos[i]; j++ )
    {
      pul_loc |= ((idx[n]>>k)&0x1)<<j;
      k++;
    }
    pul_loc = chn_pos[i*len+pul_loc];
    for ( l = 0; l < 10; l++ )
    {
      for ( m = 0; m < (1<<bit_pos_org[l]); m++ )
      {
        if ( pul_loc == chn_pos_org[l*len+m] )
        {
          chn_ctr[l]++;
          break;
        }
      }
    }
  }
```

```
  for ( i = 0; i < 10; i++ ) ctr_tmp[i] = chn_ctr[i];
  for ( i = 0; i < num[n+1]; i++ )
  {
    min_ctr = len;
    for ( j = 0; j < 10; j++ )
    {
      if ( ctr_tmp[j] < min_ctr )
      {
        min_ctr = ctr_tmp[j];
        min_chn = j;
      }
    }
    ctr_tmp[min_chn] = len;
    bit_pos[i] = bit_pos_org[min_chn];
    for ( j = 0; j < (1<<bit_pos_org[min_chn]); j++ )
    {
      chn_pos[i*len+j] = chn_pos_org[min_chn*len+j];
    }
  }
}

for ( i = 0; i < num[num_enh]; i++ )
{
  bit[i] = bit_pos[i];
  for ( j = 0; j < (1<<bit[i]); j++ )
  {
    pos[i*len+j] = chn_pos[i*len+j];
  }
}
```

The temporary pulse position tables used in the enhancement excitation tool is shown in Table 30. The table changes depending on the subframe length (5 or 10 ms).

Pulse number: i	bit_pos_org[i]	Pulse positions: chn_pos_org[i][j]
0	3	0,10,20,30,40,50,60,70
1	3	1,11,21,31,41,51,61,71
2	3	2,12,22,32,42,52,62,72
3	3	3,13,23,33,43,53,63,73
4	3	4,14,24,34,44,54,64,74
5	3	5,15,25,35,45,55,65,75
6	3	6,16,26,36,46,56,66,76
7	3	7,17,27,37,47,57,67,77
8	3	8,18,28,38,48,58,68,78
9	3	9,19,29,39,49,59,69,79

Table 31. The temporary pulse position table for 10 ms subframe

Pulse number: i	bit_pos_org[i]	Pulse positions: chn_pos_org[i][j]
0	2	0,10,20,30
1	2	1,11,21,31
2	2	2,12,22,32
3	2	3,13,23,33
4	2	4,14,24,34
5	2	5,15,25,35
6	2	6,16,26,36
7	2	7,17,27,37
8	2	8,18,28,38
9	2	9,19,29,39

Table 32. The temporary pulse position table for 5 ms subframe

5.6.2.2.9.2 Decoding of the enhancement fixed codebook gain

The **gain_enh_index** is converted to the normalized gains *nge* for the enhancement fixed codebook vectors by looking up the gain table. The gain table is changed in accordance with the *flag* and is given in Annex B. The enhancement fixed codebook gain *ge* are calculated as follows:

```
ge = nge * sqrt( norm / enh_fcb_energy );
```

where the *enh_fcb_energy* is the energy for the enhancement fixed codebook vectors. The *norm* is the normalization factor.

5.6.2.2.9.3 Enhanced excitation signal generation

The enhanced excitation signal (enh_*excitation[]*) is calculated by adding the enhancement codebook vector to excitation signals.

```
for(i = 0; i < sbfrm_size; i++)
{
   enh_excitation[i] = excitation[i] + ge*enh_fcb[i]
}
```

5.6.2.2.10 Excitation generation in the bandwidth extension tool

The bandwidth scalable decoder is realized by adding the bandwidth extension tool to the 8 kHz sampling rate mode. The bandwidth extension tool is based on CELP. Therefore, the tool consists of celp_bitstream_demux, bandwidth scalable LSP decoding tool (see section 5.5.3.3) and bandwidth scalable excitation generation tool. This scalability enhancement is allowed only for mode VIII.

In this section, the decoding process in bandwidth scalable excitation generation tool is described. This tool is connected with the multi-pulse excitation generation tool in the 8 kHz sampling rate mode. The scalable excitation signal is reconstructed by retrieving **shape_bws_positions, shape_bws_ signs, gain_bws_index** and utilizing the decoded outputs for 8 kHz sampling rate.

5.6.2.2.10.1 Decoding Process

For the 8 kHz sampling rate decoder with bandwidth scalability, the excitation signal at 16 kHz sampling is constructed from a periodic component (adaptive codebook vector) and two non-periodic components (fixed codebook vector 1 and 2) scaled by their respective gains. The adaptive and the two fixed codebook vectors are decoded using both **shape_delay** for the 8 kHz sampling rate CELP and **shape_bws_delay, shape_bws_positions,** The LSP decoding tool in the bandwidth extension tool is utilized to add bandwidth scalability to the 8 kHz sampling rate mode.

shape_bws_signs for the bandwidth extension tool. The gains are decoded from three types of indices, **signal_mode**, **rms_index** for the 8 kHz sampling rate CELP and **gain_bws_index** for the bandwidth extension tool.

5.6.2.2.10.1.1 Decoding of Signal_mode

Signal_mode in the 8 kHz sampling rate mode is also utilized in decoding the frame energy, the multi-pulse excitation and gains in this decoding process.

5.6.2.2.10.1.2 Decoding of Frame Energy

The decoding procedure is same as the 8 kHz sampling rate CELP coder.

5.6.2.2.10.1.3 Decoding of the adaptive codebook vector

The integer and the fractional parts of the pitch delay are obtained from **shape_delay** and **shape_bws_delay**. The acb_delay and acb_frac in 8 kHz sampling frequency are decoded using **shape_delay** by the same procedure as those in the 8 kHz sampling rate CELP core. The 8 kHz sampling rate parameters are converted to 16 kHz sampling rate parameters acb_delay_wb, acb_frac_wb in 16 kHz sampling frequency as follows:

```
op_delay_wb = 2 * acb_delay
if (acb_frac != 0)
{
  op_delay_wb ++
}

if (op_delay_ wb == 0)
{
  op_idx_wb = 778
}
else
{
   op_idx_wb = (op_delay_wb -32) * 3 + 2
}

st_idx_wb = op_idx_wb - 4
if (st_idx_wb < 0)
{
   st_idx_wb = 0
}
if ((st_idx_wb + 7) >= 778)
{
   st_idx_wb = 778 - 8
}

if (op_idx_wb == 778)
{
   acb_idx_wb = 778
}
else
{
   acb_idx_wb = st_idx_wb + shape_bws_delay
}
```

The mapping table between acb_idx_wb and the pitch delay parameters acb_delay_wb, acb_frac_wb is shown in Table 33. The adaptive codebook vector *acb[n]* is computed by interpolating the past excitation signal *pacb[n]* at the decoded integer delay acb_delay_wb and fraction acb_frac_wb. The interpolation is done using a FIR filter *int_fil[k], k=0,...,66* based on a Hamming windowed sinc function.

acb_idx_wb	*acb_delay_wb*	*acb_frac_wb*
0, 1	32	(2*(acb_idx_wb+1)%6
2 - 777	32+2*(acb_idx_wb-2)/6	(2*(acb_idx_wb-2))%6
778	0	0

Table 33. The mapping table between the shape_delay and the pitch delay

```
for (n = 0; n < sbfrm_size; n++ )
{
  tt += acb_frac;
  kt = acb_delay + tt / 6;
  tt = tt % 6;
  for ( i=0; i<kt && n < sbfrm_size; i++, n++ )
  {
    for ( k = -11; k <= 11; k++ )
    {
      kk = (k+1) * 6 - tt;
      acb[n] += int_fil[abs(kk)] * pacb[306-(kt-i+k)];
    }
  }
}
```

where the filter coefficients are represented as follows:

```
int_fil[0] = 1.0;
for (k = 1; k <= 11 * 6; k++)
{
  int_fil[k] = sin(PAI * k / 6) / (PAI * k / 6)
               * (0.54 + 0.46 * cos(2 * PAI * k / (2 * 11 * 6)));
}
```

5.6.2.2.10.1.4 Decoding of the fixed codebook vector 1

The fixed codebook vector 1 fcb1[n] is obtained by sampling-rate expansion of the fixed codebook vector nb_fcb[n] used in 8 kHz sampling rate as follows:

```
for (n = 0; n < sbfrm_size/2; n++)
{
  fcb1[2*n] = nb_fcb[n]
  fcb1[2*n+1] = 0
}
```

5.6.2.2.10.1.5 Decoding of the fixed codebook vector 2

The fixed codebook vector 2 contains several non-zero pulses and is represented by the pulse position and pulse amplitudes. The pulse positions *pul_pos[i]* are extracted from **shape_bws_positions**. The pulse amplitudes *pul_amp[i]* are obtained from **shape_bws_signs**.

```
for ( i = num_pulse_bws-1, k = 0; i >= 0; i-- )
{
  for ( j = 0; j < num_bit_pos[i]; j++ )
  {
    pos_idx[i] |= ((shape_bws_positions>>k)&0x1)<<j;
    k++;
  }
  pul_amp[i] = 1.0;
```

```
  if (((shape_bws_signs>>(num_pulse_bws-1-i))&0x1) == 1 )
  {
    pul_amp[i] = -1.0;
  }
  pul_pos[i] = pos_tbl[i][pos_idx[i]];
}
```

where *num_pulse_bws* is the number of pulses and is one of 6, 8, 10, 12. The choice is dependent on BWS_configuration. *num_bit_pos[i]* is the number of bits for encoding the *i* th pulse position. *pos_tbl[i][j]* is the restriction table, which indicates the possible positions for each pulse. The table indicates the possible positions for each pulse.

BWS_configuration	*num_pulse_bws*
0	6
1	8
2	10
3	12

The pulse position tables for each number of pulses are also set up in accordance with the combination of *subfrm_size*, *num_pulse* and *num_bit_pos[]* by the same procedure as the 8 kHz sampling rate core.

The fixed codebook vector *fcb2[n]* is computed from *pul_pos[i]* and *pul_amp[i]* as follows:

```
for (n = 0; n < sbfrm_size; n++)
{
  fcb2[n] = 0.0;
}
for (i = 0; i < num_pulse_bws; i++)
{
  fcb2[pul_pos[i]] = pul_amp[i];
}
```

If the integer delay **acb_delay_wb** is less than the subframe size **sbrm_size**, the fixed codebook vector signal *fcb2[n]* is modified by zero-state-comb filtering as follows:

```
for (n = 0; n < sbfrm_size; n++)
{
  if ( n - acb_delayt >= 0 )
  {
    ix = fcb2[n - acb_delay];
  }
  else
  {
    ix = 0.0;
  }
  fcb2[n] += cga[flag] * ix;
}
```

where *cga[4]={0.0,0.0,0.6,0.8}* are the gains of the comb-filter. *flag* is decoded from **signal_mode**.

5.6.2.2.10.1.6 Decoding of the adaptive and the fixed codebook gains

The **gain_bws_index** is converted to two index.

qga_idx = gain_bws_index >> 7;
qgc_idx = gain_bws_index - (gain_bws_index << 7);

The qga_idx is converted to the normalized gains *nga*, *ngf* for the adaptive and the fixed codebook vector 2 by looking up the gain table. The qgc_idx is converted to the normalized gain *ngn* for the fixed codebook vector 1 by looking up the gain table. The gain table is changed in accordance with the *flag* and is given in Annex B. The

adaptive codebook gain *ga* , the fixed codebook 1 gain gn and the fixed codebook 2 gain *gf* are calculated as follows:

```
ga = nga * sqrt( norm / acb_energy );
gn = ngn * sqrt( norm / fcb1_energy );
gf = ngf * sqrt( norm / fcb2_energy );
```

where the *acb_energy*, *fcb2_energy* and *fcb1_energy* are the energy for the adaptive codebook and the two fixed codebook vectors. The *norm* is the normalization factor.

```
norm = (qxnorm*subfrm_size)*(qxnorm*subfrm_size);
for(i = 0; i < lpc_order; i++)
{
  norm *= (1 - par[i] * par[i]);
}
```

where *par[]* are the reflection coefficients and calculated from the LP coefficients **int_Qlpc_coefficients[]**. The *qxnorm* is the quantized subframe energy which is decoded from the **rms_indx**.

5.6.2.2.10.1.7 Excitation signal generation

The excitation signal (*excitation[]*) is calculated by summing *acb[]* and *fcb[]* scaled by *ga and gf* respectively.

```
for(i=0;i<sbfrm_size;i++)
{
  excitation[i] = ga*acb[i] + gn*fcb1[i] + gf*fcb2[i]
}
```

5.6.2.2.10.1.8 Pitch Pre-Filtering

The excitation signal *excitation[]* is modified to enhance pitch harmonics as follows:

```
gp = 0.3 * ga;
if (gp > 0.3)
{
  gp = 0.3;
}
for (n = 0; n < sbfrm_size; n++ )
{
  for ( k = -11; k <= 11; k++ )
  {
    kk = (k+1) * 6 - acb_frac;
    ix += int_fil[abs(kk)] * pacb[306-(acb_delay-i+k)];
  }
  excitation[n] += gp * ix;
}
```

5.7 CELP LPC synthesis filter

5.7.1 Tool description

The tool CELP LPC synthesis filter constructs the synthesized signal from the LPC coefficients and the excitation signal for one subframe. If the lowest complexity level is used, reduced-order filtering is enabled.

5.7.2 Definitions

Input

excitation[]: This array contains the excitation signal for one subframe (see section 5.6.2.1.1).

int_Qlpc_coefficients[]: This array of dimension lpc_order contains the quantized and interpolated LPC coefficients (see section 5.5.2).

Output

synth_signal[]: The excitation signal, excitation[], is fed through the synthesis filter using the LPC coefficients of int_Qlpc_coefficients. The dimension of this array is lpc_order.

configuration

lpc_order: This field contains the order of LPC used (see section Helping variables).

sbfrm_size: This field contains the number of samples in the subframe (see section Helping variables).

5.7.3 Decoding Process

Using the interpolated LPC coefficients of one subframe, the excitation signal is fed through the following filter:

$$H_S(z) = \frac{1}{\hat{A}(z)} = \frac{1}{1 - \sum_{k=1}^{lpc\_order} \hat{a}_k \cdot z^{-k}}$$

where $\hat{A}(z)$ is the LPC inverse filter using the quantized LPC coefficients. The coefficient $\hat{a}_k$ is the k^{th} LPC coefficients (int_Qlpc_coefficients[k-1]). The output of the inverse filter is the reconstructed speech.

For 16 kHz sampling rate, by selecting the lowest complexity level (Table 3), the order of LPC can be lowered to 14 or 17 without significant degradation of the reconstructed speech signal. This leads to a significant reduction of complexity. For 8 kHz sampling rate, the LPC order is fixed to 10.

The following algorithm is an implementation of the above filter.

```
for(n = 0; n < sbfrm_size; n++)
{
    tmp = excitation[n];
    for(k = 0; k < lpc_order; k++)
    {
         tmp = tmp + Filter_states[k] * int_Qlpc_coefficients[k];
    }
    synth_signal[n] = tmp;
    for(k = lpc_order-1; k > 0; k--)
    {
         Filter_states[k] = Filter_states[k-1];
    }
    Filter_states[0] = synth_signal[n];
}
```

The array Filter_states is initially set to zero.

INFORMATIVE PART OF MPEG-4 CELP CODER

6. MPEG-4 CELP Decoder tools

6.1 CELP post-processor

6.1.1 Tool description

The tool CELP post processor enhances the reconstructed speech signal generated by the synthesis filter for one subframe. The postfiltering tools comprise a formant postfilter and a tilt compensation postfilter. REFFor 8 kHz sampling rate, also a pitch postfilter is incorporated.

For 16 kHz sampling rate, the LPC order is dependent on the complexity level (see Table 3). The postfilter can be used in combination with any of the 3 complexity levels. When using complexity level C1, reduced order LPC synthesis is applied in order to reduce the computational load. Since the postfilter constitutes to major part of the complexity in the CELP decoder, the postfilter will commonly be used in conjunction with complexity levels C2 and C3. However, should it be decided that the postfilter is used when applying C1, the postfilter must run at reduced LPC order.

For 8 kHz sampling rate the LPC order is fixed to 10.

6.1.2 Definitions

Input

synth_signal[]: This array contains the reconstructed speech signal. (see 5.7.2)

int_Qlpc_coefficients[]: This array contains the LPC coefficients for each subframe. (see 5.7.2)

acb_delay: This field indicates the pitch delay (see 5.6.2.2), which is used for the pitch postfilter. If the pitch postfilter is not needed, *acb_delay* must be set to the value less than 10.

adaptive_gain: This field indicates the gain coefficient for the periodic component of the excitation signal (see 5.6.2.2). This gain coefficient is used for the pitch postfilter. If the pitch postfilter is not needed, *adaptive_gain* must be set to the value less than 0.4.

Output

PP_synth_signal[]: This array contains the postfiltered (enhanced) speech signal. The dimension of this array is sbfrm_size.

Configuration

lpc_order: This field indicates the order of LPC that is used (Helping variables).

sbfrm_size: This field indicates the number of samples in a subframe. (Helping variables).

6.1.3 Decoding process

The decoding procedure is composed of postfiltering and adaptive gain control. The postfilter $H_{fpt}(z)$ is the cascade of three filters: a formant postfilter $H_f(z)$, a optional pitch postfilter $H_p(z)$, and a tilt compensation filter $H_t(z)$:

$$H_{fpt}(z) = H_f(z) \cdot H_p(z) \cdot H_t(z)$$

The formant postfilter is given by

$$H_f(z) = \frac{\hat{A}(z/\gamma_n)}{\hat{A}(z/\gamma_d)} = \frac{1 - \sum_{i=1}^{lpc\_order} \gamma_n^i \hat{a}_i z^{-i}}{1 - \sum_{i=1}^{lpc\_order} \gamma_d^i \hat{a}' z^{-i}},$$

where $\hat{A}(z)$ is the LPC inverse filter and the factors γ_n and γ_d control the degree of formant postfiltering. γ_n and γ_d are set to 0.65 and 0.75, respectively.

The pitch postfilter is given by

$$H_p(z) = \frac{1}{1+\gamma_p g_a}(1 + \gamma_p g_a z^{-acb\_delay}),$$

$$g_a = \frac{\sum_{n=0}^{sbfrm\_size-1} s_r(n) s_r(n - acb\_delay)}{\sum_{n=0}^{sbfrm\_size-1} s_r^2(n - acb\_delay)},$$

where $s_r(n)$ is the residual signal produced by filtering the input signal through the numerator part of the formant postfilter. The gain coefficient g_a is bounded by 1. The factor γ_p controls the degree of pitch postfiltering and has the value of 0.5. The pitch postfilter is applied only if the gain is more than 0.4 and the pitch delay is more than 10. In 16 kHz sampling rate mode, the pitch postfilter is not applied.

The filter $H_t(z)$ compensates the high-frequency tilt and is given by

$$H_t(z) = 1 - \gamma_t z^{-1},$$

where γ_t is a tilt factor of 0.3.

The synthesized signal (*synth_signal[]*) is filtered through the numerator part of the formant postfilter $\hat{A}(z/\gamma_n)$ to produce the residual signal. In 8 kHz sampling rate mode, the residual signal is then filtered through the pitch postfilter $H_p(z)$, whereas in 16 kHz sampling rate mode no pitch postfilter is used. The pitch postfiltered residual is fed through the denominator part of the formant postfilter $\hat{A}(z/\gamma_d)$. The output signal of the formant and pitch postfilters is passed through the tilt compensation filter $H_t(z)$ to generate the postfiltered synthesized signal which is not yet gain compensated.

Adaptive gain control compensates gain differences between the synthesized signal $s(n)$ and the postfiltered signal $s_p(n)$. The gain control factor G for the current subframe is computed by

$$G = \sqrt{\frac{\sum_{n=0}^{sbfrm\_size-1} s^2(n)}{\sum_{n=0}^{sbfrm\_size-1} s_p^2(n)}}.$$

For 8 kHz sampling rate, the gain-scaled postfiltered signal $s_p^{'}(n)$ is given by

$$s'(n) = g(n)s_p(n), \quad 0 \le n \le sbfrm\_size - 1,$$

where $g(n)$ is updated on a sample-by-sample basis and given by

$$g(n) = 0.95g(n-1) + 0.05G, \quad 0 \le n \le sbfrm\_size - 1.$$

The initial value of $g(-1) = 0.0$ is used. Then for each new subframe, $g(-1)$ is set equal to $g(N-1)$ of the previous subframe.

For 16 kHz sampling rate, the postfiltered signal is multiplied by G_p, which is the gain calculated in the

previous subframe. Here, the gain is updated on a subframe basis:

$$G_p = 0.9375 \cdot G_p + 0.0625 \cdot G$$

Initially $G_p = 0$.

The adaptive gain control is applied to the postfiltered signal to match the energy of the synthesized signal. *PP_synth_signal[]* is the resulting signal.

7. MPEG-4 CELP Encoder tools

This section provides a brief description of the functionality, parameter definition and the encoding processes of the tools supported by the MPEG-4 CELP core. The description of each tool comprises up to four parts: tool description, definitions, encoding process and tables.

7.1 General Introduction to the MPEG-4 CELP tool-set

The following encoder tools are supported:

- CELP preprocessing
- CELP LPC analysis
- CELP LPC quantizer and interpolator
  - Vector Quantizer
  - Scalar Quantizer
  - Lossless encoder
  - Bandwidth Scalable encoder
- CELP LPC analysis filter
- CELP weighting module
- CELP excitation analysis
  - regular pulse excitation
  - multi pulse excitation
- CELP bitstream multiplexer

The encoding is performed on frame basis and each frame is divided in subframes. The CELP excitation analysis tool is used every subframe, while the other tools are used every frame.

7.2 Helping variables

For each encoder tool a description of the variables it uses is given. In this section the most commonly used variables are provided, which are shared by multiple tools.

frame_size: This field indicates the number of samples in a frame. The decoder outputs a frame with frame_size samples.

nrof_subframes: A frame is built up of a number of subframes. The number of subframes is specified in this field.

sbfrm_size: A subframe consists of a number of samples, which is indicated by this field. The number of samples in a frame must always be equal to the sum of the number of samples in the subframes. So, the following relation must always hold

*frame_size = nrof_subframes * sbfrm_size*

These three parameters depend on the sampling rate and the bit rate settings as tabulated in Table 18, for 16 kHz sampling rate and Table 19, for 8 kHz sampling rate.

lpc_order: This field indicates the number of coefficients used for Linear Prediction. By default, the value of this field is 20 for a sampling rate of 16 kHz and 10 for 8 kHz.

num_lpc_indices. This parameter specifies the number of indices containing LPC information, that must be written to the bitstream. This is not equal to the LPC-order. For a scalar quantizer, in the 16 kHz sampling rate mode, 20 LAR indices are packed in to 9 **LPC_indices**, thus *num_lpc_indices* equals 9, while for sampling rate of 8 kHz *num_lpc_indices* equals 4 (10 LAR indices packed in to 4 **LPC_indices**). For the Vector Quantizer the *num_lpc_indices* is 5 in the 8 kHz mode and an additional 6 for the band scalable layer.

n_lpc_analysis: This field indicates how often LPC analysis is performed per frame. It is possible to perform several LPC analyses per frame with a varying window size and offset. For 16 kHz sampling rate, the value of this field is 1, indicating that LPC analysis is only performed once. For 8 kHz sampling rate, the value of this field is determined by the ratio *sbfrm_size*/80.

window_offsets[]: This array contains the offsets of the LPC analysis windows and its dimension is n_lpc_analysis.

window_sizes[]: This array contains the window sizes for the LPC analysis. Since the LPC analysis is performed n_lpc_analysis times, the dimension of this array is n_lpc_analysis.

WB_Configuration	Window sizes[] (samples)	Window offsets[] (samples)
2	320	160
other	400	280

Table 34. Window size and offset for 16 kHz sampling rate

NB_Configuration	Window sizes[] (samples)	Window offsets[] (samples)
0,1,2	160	0, 80, 160, 240
3,4,5	160	0, 80, 160
6 ... 12	160	0, 80
13 ... 21	160	0, 80
22 ... 26	160	0

Table 35. Window size and offset for 8kHz sampling rate

windows[]: This array contains the window for each analysis, so the length of this array is the sum of *window_sizes* times *n_lpc_analysis*. For 16 kHz sampling rate a squared Hamming window is used:

```
for(x=0; x < window_sizes[i];x++)
{
  window[i][x] = (0.54 - 0.46 * cos(2 * pi * (x/window_sizes[i])));
  window[i][x] = window[i][x] * window[i][x];
}
```

For 8 kHz sampling rate a conventional Hamming window is used.

gamma_be[]: This array is of size lpc_order and contains the weights for applying bandwidth expansion on the

LPC coefficients.

```
gamma_be[0]  = GAMMA;
for(x=1; x < lpc_order; x++)
{
      gamma_be[x] = GAMMA * gamma_be[x-1];
}
```

The value of GAMMA is 0.9883 for 16 kHz sampling rate and 0.9902 for 8 kHz sampling rate.

n_lag_candidates: This fields contains the number of pitch candidates. This information is used only for 16 kHz sampling rate, the value of this field is 15.

max_pitch_frequency: This field contains the maximum frequency of the pitch (lag). For 16 kHz sampling rate, this field has the value 0.025, since the minimum lag is 40. For 8kHz sampling rate, this field has the value 0.05882, since the minimum lag is 17.

min_pitch_frequency: This field contains the minimum frequency of the pitch (lag). The value of this field for 16 kHz sampling rate is 3.3898E-3, since the maximum lag is 295. For 8kHz sampling rate, this field has the value 6.944E-3, since the maximum lag is 144.

7.3 Bitstream elements for the MPEG-4 CELP tool-set

This section lists all the bitstream variables and in which tools they are used.

SamplingRate This field indicates the sampling frequency. For the MPEG-4 CELP coder, it can be either 8 kHz or 16 kHz.

QuantizationMode A one bit identifier representing whether a scalar or a vector quantizer is used.

Wideband_VQ A one bit flag indicating whether scalable or optimized vector quantizer is used. While the optimized VQ is not supported in the current version, it is reserved for future use.

FineRateControl A one bit flag indicating whether fine rate control is enabled or disabled.

LosslessCodingMode This field indicates whether the quantized LPC coefficients must be lossless decoded or not. This is only possible if QuantizationMode = ScalarQuantizer and FineRateControl = ON. In the lossless coding mode, the indices representing the Log Area Ratios have been further encoded losslessly and put on the bitstream. Thus, either the LARs are packed before they are put on the bitstream or the LARs are lossless encoded before they are put on he bitstream.

LPC_indices These fields represent the LPC coefficients. When QuantizationMode = ScalarQuantizer, these indices contain information needed to extract the Log Area Ratios. When QuantizationMode = VectorQuantizer, these contain information needed to extract the LSP coefficients. The exact extraction procedure is described in the Decoding Process.

interpolation_flag: This is a one bit flag. When set, the flag indicates that the frame under consideration is an incomplete frame, i.e. the frame does not carry the LPC coefficients of the current speech frame, but only its excitation parameters (adaptive and fixed codebook parameters). The LPC coefficients for the speech frame under consideration should be obtained using interpolation of the LPC coefficients of the adjacent frames.

1	LPC coefficients of the frame must be retrieved by interpolation
0	LPC coefficients of the frame do not have to be interpolated.

For maintaining good subjective quality, there may never be more than one frame in succession without the LPC information, i.e. the interpolation_flag may not have the value 1 in two successive CelpFrames.

LPC_Present indicates the presence of LPC parameters in the speech frame under consideration. These LPC coefficients are either of the current speech frame or those of a subsequent frame. When used in combination with the interpolation_flag**,** the two parameters completely describe how the LPC coefficients of the current frame are derived. If the interpolation flag is set, the LPC coefficients of the current frame are calculated by using the LPC coefficients of the previous and next frame. Generally, this would mean that the decoding of the current frame must be delayed by one frame. To avoid this additional delay in the decoder, the LPC coefficients of the next frame are enclosed in the current frame. In this case, the LPC_Present flag is set. Since the LPC coefficients of the next frame are already present in the current frame, the next frame will not contain LPC information.

- If the interpolation_flag is "1" and the LPC_Present is "1", then (a) the LPC parameters of the current frame are derived using the LPC parameters of the previous frame and that of the next frame and (b) the current frame (frame under consideration) carries LPC parameters of a subsequent frame, but not those of the frame under consideration. The LPC coefficients of the frame under consideration are obtained by interpolation of the previously received LPC parameters and the LPC parameters received in the frame under consideration.
- If the interpolation_flag is "0" and the LPC_Present is "0", the LPC parameters to be used with the frame under consideration are those received in the previous frame.
- When interpolation_flag is "0" and the LPC_Present is "1", then the current frame is a complete frame and the LPC parameters received in the current frame belongs to the current frame.

Such a construction is chosen so as to minimize the delay when the decoder begins reconstructing the frame, the LPC coefficients of which, are obtained using interpolation without having to wait for the next frame to arrive. Secondly, such a combination makes it possible to decode the bitstream from any point. In the fixed bit rate configuration, these two flags exhibit a fixed pattern

01, 01, 01, 01, 01, 01, … in which the string 01 is repeated or
11, 00, 11, 00, 11, 00, … in which the string 11, 00 is repeated (Fixed bit rate achieved over two frames).

For variable bit rate (when FineRateControl = ON), the string will, generally, not exhibit a fixed pattern.

NumEnhLayers The number of enhancement layers that are used. This parameter is only valid for 8 kHz sampling frequency.

BandwidthScalabilityMode This is a one bit identifier which indicates whether bandwidth scalability is enabled. This mode is only valid when SampleRateMode = 8kHz and QuantizationMode = VectorQuantizer.

BWS_Configuration This is a two bit field which configures the bandwidth extension tool. This identifier is only valid when BandwidthScalabilityMode = ON. This parameter specifies the number of bits for shape_bws_positions[i], shape_bws_signs[i].

Statistics_Update This one bit flag indicates the presence of Huffman tables to decode the lossless encoded LPC parameters. Statistics_Update is only defined if LosslessCodingMode = ON.

Statistics_Update	UpdateID	Description
0	NO	Huffman tables are not present
1	YES	Huffman tables are present

CodeLength[i][j] This four bit field indicates the length of the code for LAR coefficient i and level number j.

hufcod[i][j] This 1…16 bits field represents the j^{th} entry in the i^{th} LAR table. When Statistics_Update = YES, these tables will be updated. The initial tables are provided in Annex B.

shape_delay This is an 8 bit field representing the adaptive codebook lag. The decoding of this field depends on the sampling rate.

shape_index This index contains information needed to extract the fixed codebook contribution of the

regular pulse codebook.

signal_mode This 2 bits field represents the type of signal. This information is used for a sampling frequency of 8 kHz only. The gain codebooks are switched depending on this information.

signal_mode	Description
0	Unvoiced
1,2,3	Voiced

rms_index This parameter indicates the rms level of the frame. This information is only utilized for 8 kHz sampling rate.

shape_positions[i], shape_signs[i] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation.

shape_enh_positions[i][j], shape_enh_signs[i][j] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation in each enhancement layer.

shape_bws_delay[i] This field is utilized in decoding the adaptive codebook for the bandwidth extension tool. This value indicates the differential lag from the lag described in shape_delay.

shape_bws_positions[i], shape_bws_signs[i] These fields represent the pulse-positions and the pulse-signs for the multi-pulse excitation in each enhancement layers.

gain_indices[0] The adaptive codebook gain. It is read from the bitstream for every subframe.

gain_indices[1] These fields specify the fixed codebook gain. It is read from the bitstream for every subframe.

gain_index These fields represent the gains for the adaptive codebook and the multi-pulse excitation in the CELP coder of 8 kHz. It is read from the bit-stream for every sub-frame.

gain_enh_index These fields represent the gain for the enhancement multi-pulse excitation in the CELP coder of 8 kHz. It is read from the bit-stream for every sub-frame.

gain_bws_index These fields represents the gains for the adaptive codebook and two multi-pulse excitation in the bandwidth extension tool. It is read from the bit-stream for every sub-frame.

7.4 CELP Preprocessing

7.4.1 Tool description

The CELP preprocessing tool produces a DC free speech signal.

7.4.2 Definitions

Input

s[]: This is an array of dimension frame_size and contains input speech samples.

Output

pp_s []: This is an array of length frame_size and contains the DC free speech samples.

Input/Output

prev_x, *prev_y*: memory of the preprocessing filter

Configuration

*frame_size***:** This field indicates the number of samples in the input signal (see section Helping variables).

7.4.3 Encoding Process

This block removes the DC element from the input signal, s[n]. It is first order recursive filter of the form:

$$H_{pre}(z) = \frac{1 - z^{-1}}{1 - 0.99 \cdot z^{-1}}$$

An implementation of this filter is given below:

```
for (n = 0; n < frame_size; n++)
{
    pp_s[n] = s[n] – prev_x + 0.99 * prev_y;
    prev_x = s[n];
    prev_y = pp_s[n];
}
```

The input/output filter states *prev_x* and *prev_y* are initialized to zero.

7.5 CELP LPC Analysis

7.5.1 Tool description

The CELP LPC Analysis tool estimates the short-term spectrum. The LPC analysis is performed on the preprocessed speech signal, pp_s[]. The order of linear prediction is specified by the parameter lpc_order. A window, with size specified in window_size[], is used to weight the preprocessed speech. The parameter, window_offset[], is provided to specify the offset for each window.

7.5.2 Definitions

Input

PP_InputSignal[]: This array contains the preprocessed speech signal. This dimension is frame_size. (see section 7.4).

Output

lpc_coefficients[] : This array contains the computed LPC coefficients and has size lpc_order (see section Helping variables).

first_order_lpc_par : This output field contains the LPC coefficient for 1st-order fit. This parameter is used for the preselection in the adaptive codebook search.

Configuration

frame_size: This field denotes the number of samples in frame (see section Helping variables).

window_offset[]: This array contains the offset of the window (see section Helping variables).

window_size[]: This array contains the LPC analysis-window size (see section Helping variables).

windows[]: This array contains the windows used to weight the speech signal (see section (see section Helping

variables).

gamma_be[]: This array contains the gamma coefficients that are used for bandwidth expansion of the LPC coefficients (see section Helping variables)..

lpc_order: This field indicates the order of LPC (see section Helping variables)..

n_lpc_analysis: This field denotes the number of LPC analysis (see section Helping variables).

7.5.3 Encoding Process

The Linear Prediction analysis is performed n_lpc_analysis times, each with a different window size and offset as specified in the array's window_size and window_offset. Each time the input signal PP_InputSignal is weighted, sw[n].

By means of the weighted signal the autocorrelation coefficients are derived using:

$$acf[k] = \sum_{n=0}^{frame\_size-k-1} sw[n] \cdot sw[n+k], 0 \le k \le lpc\_order$$

There are lpc_order+1 autocorrelation coefficients.

The LPC coefficients are computed using the conventional Levinson-Durbin recursion. The first LPC coefficient is assigned to first_order_lpc_par. Bandwidth expansion is applied on the LPC coefficients using array gamma_be. For each LPC analysis, the calculated lpc_coefficients are stacked, resulting in n_lpc_analysis * lpc_order coefficients.

7.6 CELP LPC quantizer and interpolator

7.6.1 Tool description

The LPC coefficients are quantized by using either Vector Quantization or Scalar Quantization. Optional tools that are available are: lossless coding, fine rate control and bandwidth extension.

7.6.2 Definitions

Input

lpc_coefficients[]: This is an array of dimension lpc_order and contains the current unquantized LPC coefficients.

Output

int_Qlpc_coefficients[]: This is an array of length nrof_subframes * lpc_order and contains the interpolated and quantized LPC coefficients for each subframe. The LPC coefficients for each subframe are stacked one after the other, resulting in a nrof_subframes * lpc_order array.

lpc_indices[]: This is an array of dimension num_lpc_indices and contains the packed lpc indices that are written to the bitstream.

interpolation_flag: This field indicates if interpolation on the LPC coefficients between frames is done.

1	Interpolation between frames is performed
0	Interpolation between frames is not performed

If the interpolation flag is set, the current LPC coefficients are computed from the previous and next LPC coefficients.

signal_mode: This field indicates the voiced/unvoiced flag.

LPC_Present: This flag indicates whether the current frame is complete or incomplete.

Configuration

lpc_order: This field contains the order of LPC (see section Helping variables).

num_lpc_indices: This fields indicates the number of packed LPC codes (see section Helping variables).

n_lpc_analysis: This field contains the number of LPC per (see section Helping variables).

nrof_subframes: This field contains the number of subframes (see section Helping variables).

7.6.3 Encoding process

The array lpc_coefficients contains the LPC coefficients of the next frame, which is the frame following the frame that is currently encoded. Therefore, a memory is needed which stores the previous LPC coefficients and the current LPC coefficients. In the sequel previous refers to the last frame being encoded, current refers to the frame that is currently being encoded and next refers to the frame that is encoded next.

7.6.3.1 LAR Quantization Process

First, the LPC coefficients are transformed to reflection coefficients using the following recurrent relation:

```
Initially: a(j)(lpc_order-1) = a[j-1] for 1 ≤ j ≤ lpc_order
Then for i going down from lpc_order to 1.
rfc[i-1] = a(i)(i)
a(j)(i-1) = (a(j)(i)+a(i)(i)*a(i-j)(i))/(1-rfc[i-1]²) for 1 ≤ j ≤ i-1
```

The computed reflection coefficients are quantized using a look-up table which is given in Table 36. Besides the quantized reflection coefficients, the indices and the Log Area Ratios (LARs) are computed.

```
for(n = 0; n < lpc_order; n++)  {
    int lindex = rfc_offset[n];
    int uindex = lindex + rfc_level[n] - 1;
    int index  = lindex;

    while( (rfc[k] > rfc_table_quant[index]) && (index < uindex) )
    {
       index++;
    }
    indices[n]  = (index - lindex);
    next_qrfc[n] = rfc_table[index];
    next_lar[n] = log((1-next_qrfc[n])/(1+next_qrfc[n]));
}
```

The table rfc_offset contains the offsets in the rfc quantization table for each reflection coefficient and is given in Table 21. The table rfc_level contains the number of quantization levels in the rfc quantization table (see Table 37). The table rfc_table_quant is the rfc quantization table and rfc_table is the reflection coefficients representation level table, which is also given in Table 20.

The decision whether or not the LARs of the speech frame under analysis are transmitted to the decoder depends on the amount of change *d*, between the spectrum described by the LARs of the current frame and the spectrum described by the LARs obtained by interpolating the LARs of the adjacent frames. If *d* is greater than a particular threshold then the coefficients are transmitted to the decoder. This threshold is further made dependent on the desired bit-rate setting as follows: The threshold is raised if the actual bit rate is higher than the desired bit-rate setting and is lowered otherwise.

The interpolated LARs, denoted by i_lar[], are obtained from the LARs of the previous and next frame as follows:

```
for(k = 0; k < lpc_order; k++) {
    i_lar[k] = (prev_lar[k] + next_lar[k])/2;
}
```

The amount of change *d*, between the current LARs current_lar[] and the interpolated LARs i_lar[], can be computed in the spectral domain or simply by computing the L1-norm. The distance *d* is then compared to the threshold which is dependent on the desired bit-rate setting and the actual bit rate as explained earlier. If the LARs of the speech frame under analysis are not transmitted, then the interpolation flag is set.

If the interpolation flag is set, the LPC indices of the next frame are written to the bitstream and the LPC_Present flag is set to 1. If no interpolation is performed and the indices of the current frame are not yet written to the bitstream, the current indices are written and the LPC_Present flag is set to 1. If no LPC indices are present, LPC_Present is set to 0. The indices are packed in the following way for lpc_order equal to 20:

```
lpc_indices[0] =((indices[0] * 14  ) + indices[3]);
lpc_indices[1] =((indices[1] * 1155) +(indices[2] * 77)
                + (indices[7] * 7) + indices[11]);
lpc_indices[2] =((indices[4] * 156 ) +(indices[5] * 12) + indices[6]);
lpc_indices[3] =((indices[8] * 7     ) + indices[12]);
lpc_indices[4] =indices[9];
lpc_indices[5] =indices[10];
lpc_indices[6] =indices[13];
lpc_indices[7] =((indices[14] * 36 ) +(indices[15] * 6) + indices[16]);
lpc_indices[8] =((indices[17] * 36 ) +(indices[18] * 6) + indices[19]);
```

For lpc_order is 10, the indices are packed as follows:

```
lpc_indices[0] =((indices[0] * 108 )+(indices[6] * 9) + indices[8]);
lpc_indices[1] =((indices[1] * 143 )+(indices[4] * 11) + indices[7]);
lpc_indices[2] =((indices[2] * 182 )+(indices[3] * 13) + indices[5]);
lpc_indices[3] =  indices[9];
```

For each subframe, the LARs are interpolated and the corresponding LPC coefficients are calculated:

```
for(n = 0; n < nrof_subframes; n++)
{
    for(k = 0; k < lpc_order; k++)
    {
        int_lar[k] = ((nrof_subframes - n - 1) * prev_lar[k]
                      + (1 + n) * current_lar[k])/nrof_subframes;
    }
    Lar2Rfc(lpc_order, int_lar, reflection_coeffs);
    Rfc2Apar(lpc_order,reflection_coeffs,
                   &int_Qlpc_coefficients[n*lpc_order]);
}
```

The transformation of LARs to reflection coefficients and reflection coefficients to LPC coefficients is given in the description of the decoder.

Finally, the memory containing the previous and current LPC coefficients is updated.

7.6.3.2 Lossless encoding of the LAR indices

7.6.3.2.1 Tool description

Using the lossless coding scheme, it is possible to lower the bit rate of the speech coder by up to 1.3 kbit/s on an average. The advantage is that of achieving a lower bit rate without affecting the quality.

The Lossless coding scheme is applied only in case of a scalar quantizer, when the order of LPC is either 10 or 20. The principle of lossless coding is that a shorter code is allocated to the more probable values. Then it is possible to reduce the average code length. Such an entropy coding scheme is discussed at length at various places in the literature, for instance in [3,4].

7.6.3.2.2 Definitions

Input

indices[] These are the Huffman decoded LAR indices.

Output

Statistics_Update This parameter indicates whether the Huffman tables, representing the lossless encoded LPC indices, have to be updated.

CodeLength[i][j] For each LAR i, the array contains the number of Huffman code-word lengths i. This array is used to generate the Huffman tables.

huf_cod [i][j]

htab[i] These are the Huffman codes representing LAR 'i'.

Configuration

nrof_levels_lars: This parameter indicates the number of bins that where used to collect the statistics of each LAR index.

The array *nr_levels_lar* is defined as follows:

Lar coefficient index	*nr_levels_lar*
0	36
1	28
2	15
3	14
4	13
5	13
6	12
7	11
8	9
9	8

Lar coefficient index	*nr_levels_lar*
10	8
11	7
12	7
13	8
14	7
15	6
16	6
17	7
18	6
19	6

7.6.3.2.3 Encoding Process

The distributions of the quantized LARs change depending on the segments under analysis and vary from speaker to speaker. Accordingly, the distributions and the allocation of codewords must be updated to harness the potential of the proposed lossless coding scheme. An update of statistics, once every 1000 frames is sufficient. Since the statistics must be transmitted to the decoder, this does count as an overhead.

The encoder generates the information necessary for the decoder to form these codewords, which means it must adaptively generate the entropy code for each LPC parameter. The complexity is limited, because of the relatively small number of bins in which the statistics are analyzed (at most 36 bins and much less on average and 227 in total). This too has to be done only once per 1000 frames. The encoder updates the LPC parameter statistics for each frame, which requires a single increment operation per LPC parameter.

The encoder signals the decoder about the statistics update by setting the **Statistics_Update** flag to 1. The Huffman codeword lengths are limited to 16, so 4 bits are needed to specify the length (1..16), since there are a total of 227 entries for all the LARs, the number of bits used for transmitting all the codeword lengths equals 227*4 = 908 bit. The generation of the Huffman table is discussed in the decoder. Also the Huffman encoding of the LPC codes is discussed there.

Quantization Tables for the scalar quantizer (Mode I, II, III and IV)

Index	rfc_table_quant	Index	rfc_table_quant
0	-0.9882	25	0.5098
1	-0.9848	26	0.5964
2	-0.9806	27	0.6710
3	-0.9751	28	0.7341
4	-0.9682	29	0.7866
5	-0.9593	30	0.8298
6	-0.9481	31	0.8649
7	-0.9338	32	0.8932
8	-0.9158	33	0.9158
9	-0.8932	34	0.9338
10	-0.8649	35	0.9481
11	-0.8298	36	0.9593
12	-0.7866	37	0.9682
13	-0.7341	38	0.9751
14	-0.6710	39	0.9806
15	-0.5964	40	0.9848
16	-0.5098	41	0.9882
17	-0.4116	42	0.9908
18	-0.3027	43	0.9928
19	-0.1853	44	0.9944
20	-0.0624	45	0.9956
21	0.0624	46	0.9966
22	0.1853	47	0.9973
23	0.3027	48	0.9999
24	0.4116	49	------

Table 36. Quantization table for reflections coefficients

Index	rfc_level	Index	rfc_level
0	36	10	8
1	28	11	7
2	15	12	7
3	14	13	8
4	13	14	7
5	13	15	6
6	12	16	6
7	11	17	7
8	9	18	6
9	8	19	6

Table 37. Number of levels used for quantization of reflection coefficients

7.6.3.3 LSP Quantization

The LPCs are converted into LSPs and quantized. As described in decoding section, there are two methods for quantizing the LSPs; a two-stage VQ without interframe prediction, and a combination of the VQ and the inter-

frame predictive VQ. At the encoding process, both methods are tried to quantize the LSPs and determined which method should be applied by comparing the quantization error. The quantization error is calculated as a weighted Euclidean distance. The weighting coefficients (*w[]*) are,

$$w[\mathrm{i}] = \begin{cases} \dfrac{1}{lsp[0]} + \dfrac{1}{lsp[1] - lsp[0]} & (i = 0) \\ \dfrac{1}{lsp[i] - lsp[i-1]} + \dfrac{1}{lsp[i+1] - lsp[i]} & (0 < i < Np - 1), \\ \dfrac{1}{lsp[Np-1] - lsp[Np-2]} + \dfrac{1}{1.0 - lsp[Np-1]} & (i = Np - 1) \end{cases}$$

where *Np* is the LP analysis order *lpc_order* and *lsp[]* is the LSPs converted from the LPCs.

The first stage quantizer is the same for each quantization method. The LSPs are quantized by using a two-split vector quantizer and corresponding indices are stored in lpc_indices[0] and lpc_indices[1]. In order to carry out delayed decision, two indices, namely the best and the second best are stored as candidates for the second stage. The quantization error in the first stage *err1*[] is given by:

$$err1[n] = \sum_{i=0}^{\dim-1} \left\{ \left(lsp[sp+i] - lsp\_tbl[n][m][i] \right)^2 \cdot w[sp+i] \right\} \quad n = 0,1$$

where *n* is the split vector number, *m* is the index of the candidate split vector , sp is the starting LSP order of the n-th split vector and dim is the dimension of the n-th split vector. (*lsp_tbl[][][]* is shown in Annex B)

Split vector number: n	Starting LSP order: sp	Dimension of the vector: dim
0	0	5
1	5	5

Table 38. Starting order and dimension of the first stage LSP vector

In the second stage, above-mentioned two quantization methods, which are also two-split vector quantizer, are applied respectively. Total quantization errors in the second stage are calculated for all combinations of the first stage candidates and the second stage candidates and the one which has the minimum error is selected. As a result, indices of the first stage are determined and corresponding indices and signs for the second stage are stored in lpc_indices[2] and lpc_indices[3]. The flag which indicates the selected quantization method is also stored in lpc_indices[4]. The quantization error in the second stage *err2_total* is given by:

VQ without interframe prediction:

$$err2\_total = err2[0] + err2[1]$$
$$err2[n] = \sum_{i=0}^{\dim-1} \left\{ \left(lsp\_res[sp+i] - sign[n] \cdot d\_tbl[n][m][i] \right)^2 \cdot w[sp+i] \right\} \qquad n = 0,1$$
$$lsp\_res[sp+i] = lsp[sp+i] - lsp\_first[sp+i]$$

where *lsp_first*[] is the quantized LSP vector of the first stage, *n* is the split vector number, *m* is the index of the candidate split vector , sp is the starting LSP order of the n-th split vector and dim is the dimension of the n-th split vector. (*d_tbl[][][]* is shown in Annex B)

VQ with interframe prediction:

$$err2\_total = err2[0] + err2[1]$$
$$err2[n] = \sum_{i=0}^{\dim-1} \left\{ \left(lsp\_pres[sp+i] - sign[n] \cdot pd\_tbl[n][m][i] \right)^2 \cdot w[sp+i] \right\} \qquad n = 0,1$$

$$lsp\_pres[sp+i] = lsp[sp+i] - \{(1-ratio\_predict)\cdot lsp\_first[sp+i] + ratio\_predict \cdot lsp\_previous[sp+i]\}$$

where *lsp_first*[] is the quantized LSP vector of the first stage, *n* is the split vector number, *m* is the index of the candidate split vector , sp is the starting LSP order of the n-th split vector, dim is the dimension of the n-th split vector and ratio_predict=0.5. (*pd_tbl[][][]* is shown in Annex B)

Split vector number: n	Starting LSP order: sp	Dimension of the vector: dim
0	0	5
1	5	5

Table 39 Starting order and dimension of the second stage LSP vector

The quantized LSPs *lsp_current[]* are stabilized in order to ensure stability of the LPC synthesis filter which is derived from the quantized LSPs. The quantized LSPs are arranged in ascending order, having a minimum distance between adjacent coefficients.

```
for (i = 0; i < lpc_order; i++)
{
   if (lsp_current[i] < min_gap)
   {
      lsp_current[i] = min_gap;
   }
}

for (i = 0; i < lpc_order - 1; i++)
{
   if (lsp_current[i+1] - lsp_current[i] < min_gap)
   {
      lsp_current[i+1] = lsp_current[i]+min_gap;
   }
}

for(i = 0; i < lpc_order; i++)
{
   if (lsp_current[i] > 1-min_gap)
   {
      lsp_current[i] = 1-min_gap;
   }
}

for (i = lpc_order - 1; i > 0; i--)
{
   if (lsp_current[i]-lsp_current[i-1] < min_gap)
   {
      lsp_current[i-1] = lsp_current[i]-min_gap;
   }
}
```

where min_gap = 6.0/256.0

After the quantization process, the quantized LSPs are interpolated linearly at each subframe.

```
for (n = 0; n < nrof_subframes; n++)
{
   ratio_sub=(n+1)/nrof_subframes
   for(i = 0; i < lpc_order; i++)
   {
      lsp_subframe[n][i]=((1-ratio_sub)*lsp_previous[i] + ratio_sub*lsp_current[i]))
   }
}
```

The interpolated LSPs are converted to the LPCs using the auxiliary function *Convert2lpc()*.

```
for(n = 0; n < nrof_subframes; n++)
{
   Convert2lpc(lpc_order, lsp_subframe[n], int_Qlpc_coeffiecients + n*lpc_order]);
}
```

After the calculation of the LPC coefficients, the current LSPs have to be stored in memory, since they are used for interpolation at the next frame.

```
for (i = 0; i < lpc_order; i++)
{
     lsp_previous[i] = lsp_current[i];
}
```

It must be noted that the stored LSPs *lsp_previous[]* must be initialized as described below when whole the encoder is initialized.

```
for (i = 0; i < lpc_order; i++)
{
     lsp_previous[i] = (i+1) / (lpc_order+1);
}
```

7.6.3.4 LSP Quantization for the bandwidth extension tool

The input 16 kHz sampling rate LSPs are vector quantized with intraframe and interframe prediction approaches. The intraframe prediction module produces an estimated LSPs by converting the quantized LSPs obtained in the 8 kHz sampling rate CELP coder. Furthermore, an interframe moving average predictive VQ is also employed for more accurate prediction.

7.7 CELP LPC Analysis filter

7.7.1 Tool Description

The CELP LPC Analysis filter tool feeds the input signals through a filter with LPC coefficients and returns the residue signal.

7.7.2 Definitions

Input

PP_InputSignal[]: This array has dimension sbfrm_size and contains the input signal.

int_Qlpc_coefficients[]: This array has dimension lpc_order and contains the LPC coefficients (see section 7.6.2).

Output

lpc_residual[]: This array has dimension sbfrm_size and contains the LPC filtered residual signal.

Configuration

lpc_order: This field indicates the order of LPC (see section Helping variables).

sbfrm_size: This field indicates the number of samples in subframe (see section Helping variables).

7.7.3 Encoding process

The input signal is filtered using the filter coefficients.

```
for (k = 0; k < sbfrm_size; k++)
{
   tmp = PP_InputSignal[k];
   for (j = 0; j < lpc_order; j++)
   {
      tmp =  tmp - int_Qlpc_coefficients[j] * Filter_States[j];
   }
   lpc_residual[k] = tmp;
   update_Filter_States;
}
```

The initial filter states are set to zero.

7.8 CELP Weighting module

7.8.1 Tool Description

The CELP weighting module computes the weights to be applied on the LPC coefficients.

7.8.2 Definitions

Input

lpc_coefficients[]: This is an array of dimension lpc_order, containing the LPC Coefficients.

gamma_num: This field contains the weighting factor of the numerator.

gamma_den: This field contains the weighting factor of denominator.

Output

Wnum_coeff[]: This is an array of dimension Wnum_order, containing the weighted numerator coefficients.

Wden_coeff[]: This array has dimension Wden_order and contains the weighted denominator coefficients.

Configuration

lpc_order: This field contains the order of the LPC (see section Helping variables).

7.8.3 Encoding process

The weighted coefficients of the numerator are computed by:

```
for (k = 0; k < Wnum_order; k++)
{
   Wnum_coeff[k] = lpc_coefficients[k] * gamma_num^(k+1)
}
```

The weigthed coefficients of the denominator are computed by:

```
for (k=0; k < Wden_order; k++)
{
  Wden_coeff[k] = lpc_coefficients[k] * gamma_den^(k+1)
}
```

7.9 CELP Excitation analysis

7.9.1 Tool description

The tool CELP excitation analysis computes the shape and gain vectors as well as the decoded synthesized speech signal. For excitation analysis modules Regular Pulse Excitation (RPE) and Multi Pulse Excitation (MPE) are defined.

For Regular Pulse Excitation, the output of the excitation analysis are**shape_delay[]**, **shape_index[]** and **gain indices[]**. The shape and gain indices are generated every subframe. The vector **shape_delay[]** contains the adaptive codebook lag for each subframe, while the vector **shape_index[]** contains the fixed codebook index. The vector **gain_indices[0][]** contains the adaptive codebook gain for every subframe. The fixed codebook gain for every subframe is stored in vector **gain_indices[1][]**.

For Multi Pulse Excitation, the output of the excitation analysis are **shape_delay[]**, **shape_positions[]**, **shape_signs[]** and **gain_index[]**.The shape and gain indices are generated every subframe. The vector **shape_delay[]** contains the adaptive codebook lag.for each subframe, while the vectors **shape_positions[]** and **shape_signs[]** contain the pulse positions and signs respectively. The adaptive codebook gain and multi-pulse gain for each subframe are vector quantized and stored in the vector **gain_index[]**.

7.9.2 Definitions

Input

PP_inputSignal[]: This array contains the preprocessed input signal and is of dimension sfrm_size.

lpc_residual[]: This contains the lpc residual signal (see section Helping variables).

int_Qlpc_coefficients[]: This array contains the interpolated and quantized LPC coefficients (see section 7.6.2).

Wnum_coeff[]: This array contains the weighting filter coefficients of the numerator (see section Helping variables).

Wden_coeff[]: This array contains the weighting filter coefficients of the denominator (see section Helping variables).

first_order_lpc_par: This field indicates the first LPC coefficient (see section 7.5).

lag_candidates[]: This array contains the lag candidates.

signal_mode: This field contains the voiced/unvoiced flag .

rms_index: This field defines the index for the signal power.

Output

shape_delay[]: This array is of dimension num_cbks. It contains the codebook lag for the adaptive and fixed codebooks.

shape_index[]: This array is of dimension num_cbks. It contains the shape indices for the adaptive and fixed codebooks.
gain_indices[][]: This array is of dimension 2 * nrof_subframes. It contains the gain indices for the adaptive and fixed codebooks.

shape_positions[]: This array is of dimension num_cbks. It contains the pulse positions for 8 kHz sampling rate.

shape_signs[]: This array is of dimension num_cbks. It contains the pulse signs for 8 kHz sampling rate.

gain_index[]: This array is of dimension num_cbks. It contains the vector quantized gains for the adaptive codebook and the multi-pulse for 8 kHz sampling rate.

decoded_excitation[]: This array is of length sbfrm_size and contains the synthesized speech signal.

Configuration

n_lag_candidates: This field indicates the number of lag candidates (see section Helping variables).

frame_size: This field indicates the number of samples in one frame (see section Helping variables).

sbfrm_size: This field indicates the number of samples in one (see section Helping variables).

nrof_subframes: This field indicates the number of subframes in one frame (see section Helping variables).

lpc_order: This field indicates the order of LPC (see section Helping variables).

7.9.3 Encoding process

7.9.3.1 Regular Pulse Excitation

All blocks are performed on a once per subframe basis. For each subframe the following steps are performed:
- perceptual weighting
- adaptive codebook search preselection
- adaptive codebook search
- fixed codebook search preselection
- fixed codebook search
- simulation of decoder

Next, each step will be described in detail. For convenience, *aq*[] is used instead of *int_Qlpc_coefficients*[], so the *aq*[] contains the quantized LPC coefficients for the subframe under consideration.

Perceptual weighting is performed by filtering the input signal *PP_InputSignal[]* by the following filter:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 - \sum_{k=0}^{lpc\_order-1} aq[k] \cdot z^{-k-1}}{1 - \sum_{k=0}^{lpc\_order-1} aq[k] \cdot \gamma^{k+1} \cdot z^{-k-1}}$$

with $\gamma = 0.8$. The resulting signal is called *ws[n]*.
The zero-input response *z[n]*, is determined by computing the response of $S(z)$ to an zero-valued input signal.

$$S(z) = \frac{1}{A(z/\gamma)} = \frac{1}{1 - \sum_{k=0}^{lpc\_order-1} aq[k] \cdot \gamma^{k+1} \cdot z^{-k-1}}$$

where $\gamma = 0.8$.

The weighted target signal *t[n]* is obtained by subtracting *z[n]* from *ws*[*n*]*:*

```
for  (n=0;n< Nm;n++) {
    t[n] = ws[n] - z[n];
}
```

The impulse response *h[n]* is calculated as follows:

```
for (k=0;k<lpc_order;k++) {
    tmp_states[k] = 0;
}
tmp = 1.0;
for  (n=0; n < sbfrm_size; n++) {
    g = 0.8;
    for  (k=0; k<lpc_order;k++) {
        tmp = tmp + aq[k][s] * g * tmp_states[k];
        g = 0.8 * g;
    }
    h[n] = tmp;
    for  (k=lpc_order-1;k>0;k--) {
        tmp_states[k] = tmp_states[k-1];
    }
    tmp_states[0] = tmp;
}
```

After the above mentioned computations, preselection on the adaptive codebook is performed. The adaptive codebook contains 256 codebook sequences of which 5 are preselected. Preselection is achieved by evaluating the term (for 0 <= l < Lm and 0 <= i < nrof_subframes):

$$rap[i \cdot Lm + l] = \frac{\left(\sum_{n=0}^{sbfrm\_size-1} ca[L\min + i \cdot sbfrm\_size + l \cdot 3 - n - 1] \cdot ta[n] \right)^2}{Ea[i \cdot Lm + l]}$$

where:

$$Lm = 1 + \left\lfloor \frac{sbfrm\_size - 1}{3} \right\rfloor$$

and *ca*[Lmin + *i sbfrm_size* + 3 *l* - *n*-1] and *ta*[*n*] representing the codebook sequence at delay (Lmin+*i sbfrm_size*+3 *l*) and the 'backward-filtered' target signal *t*[*n*], respectively.
Backward filtering comprises the time-reversal *of t[n],* filtering by *S(z)* and time-reversal again. *Lmin* is the minimum lag in samples. The value of *Lmin* is 40.

Ea[i Lm+l] is the energy of that codebook sequence filtered by a reduced complexity synthesis filter $S_p(z)$ which is a first-order estimation of the LPC synthesis filter *S* (*z*):

$$S_p(z) = \frac{1}{A_p(z/\gamma)} = \frac{1}{1 - a \cdot \gamma \cdot z^{-1}}$$

The first LPC coefficient of a frame is taken for *a*. Depending on the number of subframe, which is under consideration, the first LPC coefficient of the current or the previous is taken using the following:

```
if (subframe_number < nr_subframes/2)
    a = prev_a;
else
    a = cur_a;
```

After evaluation of *rap[]* the 5 maximum sequences are selected together with their 2 neighbor sequences resulting in 15 candidates on which full search is applied. The indices of the preselected sequences and their neighbors are stored in *ia*[*r*] (0 <= *r* < 15).

The adaptive codebook search minimizes the mean-squared weighted error between the original and reconstructed speech. This is achieved by searching for the index *r* maximizing the term

$$ra[r] = \frac{\left(\sum_{n=0}^{sbfrm\_size-1} t[n] \cdot y[r][n]\right)^2}{\sum_{n=0}^{sbfrm\_size-1} y^2[r][n]}$$

The signal *t*[*n*] represents the weighted target signal and *y*[*r*][*n*] is the convolution of the sequence *ca*[*ia*[*r*]-*n*-1] with the impulse response *h*[*n*]. The index (*ia*[*r*]-*Lmin*) of the maximum is referred as **shape_delay[**subframe**]**.

After determining the index the gain factor is computed according to

$$g = \frac{\sum_{n=0}^{sbfrm\_size-1} t[n] \cdot y'[n]}{\sum_{n=0}^{sbfrm\_size-1} y'^2[n]}$$

with *y'*[*n*] equal to the convolution of *ca*[*I*+Lmin-*n*-1] with *h*[*n*]. The gain factor is quantized by a non-uniform quantizer:

```
for (j = 0; abs(g) > cba_gain_quant[j] && j < 31; j ++);

if (g < 0)
{
   Ga =  -cba_gain[j];
   gain_indices[0] = (((-j - 1)) & 63);
}
else
{
    Ga = cba_gain[j];
    Gain_indices[0][subframe] = j;
}
```

The quantized gain factor is referred as *Ga*. With **shape_delay[]** and *Ga* the contribution *p*[*n*] of the adaptive codebook is computed according to

$$p[n] = Ga \cdot y'[n]$$

The contribution *p*[*n*] of the adaptive codebook is subtracted from the weigthed target signal *t*[*n*] to obtain the residual signal *e*[*n*]:

$$e[n] = t[n] - p[n], 0 \le n < sbfrm\_size$$

The residual signal *e*[*n*] is 'backward-filtered' to obtain *tf*[*n*].

The fixed codebook search also consists of a preselection and a selection phase. An RPE-codebook is used for the fixed codebook excitation. Each codebook vector has sbfrm_size pulses of which Np pulses may have an amplitude of +1, 0 or -1. These Np pulses are positioned on a regular grid defined by the phase *p* and the pulse spacing D such that the grid positions are at *p* + D*l* where *l* is between 0 and Np. The leaving (sbfrm_size-Np) pulses are zero. D and Np are dependent on the bit-rate setting as is given in Table 25.

To reduce complexity a local RPE-codebook is generated for each subframe containing a subset of 16 entries. All vectors of this local RPE-codebook have the same phase P which computed as follows:

```
max = 0;
for (p = 0; p < D; p++)
{
    tmp_max = 0;
    for(l=0; l < Np-1; l++)
    {
        tmp_max += abs(tf[p+D * l]);
```

```
        }
        if (tmp_max > max)
        {
            max = tmp_max;
            P = p;
        }
    }
```

Having determined the phase P it is required that the amplitude of pulse l, 0 <= l < Np, is either zero or equal to the sign of the corresponding sample of tf[n]. The sign is stored in amp[l] as follows:

```
    for (l=0; l < Np; l++)
        amp[l] = sign(tf[P + D * l]);
```

where sign is +1 for values greater than 0 and –1 otherwise.

For Np-4 pulses the possibility of an zero-amplitude is excluded a-priori at those positions where tf[n] is maximal. Therefore, array pos[] is used, which has the following semantic:

- pos[l]=0 indicates that zero-amplitude possibility is included for pulse l.
- pos[l]=1 indicates that zero-amplitude possibility is excluded.

The following algorithm is used to determine the array pos[]:

1. initialize pos[] with zeros,
2. Determine n such that abs(tf[P + D * n]) >= abs(tf[P + D * i]) for all i not equal to n
3. pos[n] = 1
4. tf[P + D * n] = 0
5. Proceed with step 2 until Np-4 positions are fixed

Based on P, amp[] and pos the local RPE-codebook cf[k][n] is generated, using the following algorithm:

```
for(k=0;k < 16; k++)
    for(n=0;n<sbfrm_size};n++)
         cf[k][n] = 0;
for(l=0;l<Np;l++)
    cf[0][P+D*l]=amp[l];
m=1;
for(l=0;l<Np;l++)
{
  if(pos[l]==0)
  {
      c=m;
      for(q=0;q<c;q++)
      {
          for(n=0;n < Np;n++)
               cf[m][P+D*n] = cf[q][P+D*n];
          cf[m++][P+D*l] = 0;
      }
  }
}
```

Based on the local RPE-codebook, 5 vectors out of 16 are preselected for the closed-loop search. Preselection is achieved by evaluating the term for 0 <= k < 16:

$$rfp[k] = \frac{\left(\sum_{n=0}^{sbfrm\_size-1} cf[k][n] \cdot tf[n]\right)^2}{Ef[l]}$$

with *cf*[*k*][*n*] representing a vector of the local RPE-codebook. *Ef*[*k*] is the energy of that codebook vector filtered by the reduced complexity synthesis filter $S_p(z)$. The indices of the preselected vectors are stored in *if*[*r*].

Using these preselected indices, a closed-loop fixed codebook search is performed. By the closed-loop fixed codebook search it is searched for the index *r* maximizing the term

$$rf[r] = 2 \cdot Gf \cdot \sum_{n=0}^{sbfrm\_size-1} e[n] \cdot y[r][n] - Gf^2 \cdot \sum_{n=0}^{sbfrm\_size-1} y^2[r][n]$$

The signal $e[n]$ represents the residual signal of the adaptive codebook search and $y[r][n]$ is the convolution of $cf[if[r]][n]$ with the impulse response $h[n]$. Gf is the quantized gain factor g which is determined according to

$$g = \frac{\sum_{n=0}^{sbfrm\_size-1} e[n] \cdot y[r][n]}{\sum_{n=0}^{sbfrm\_size-1} y^2[r][n]}$$

The quantization process of the fixed codebook gain is given below, where Gp is the previous quantized fixed codebook gain.

```
if (first subframe)
{
    for (m = 0; g > cbf_gain_quant[m] && m < 30; m ++);
    Gf = cbf_gain[m];
}
else
{
    g = g / Gp;
    for (m = 0; g > cbf_gain_quant_dif[m]  && m < 7; m ++);
    g = Gp * cbf_gain_diff[m];
}
gain_indices[0] = m;
```

The representation tables cbf_gain and cbf_gain_dif are given in Table 27 and Table 28, whereas the quantization tables cbf_gain_quant and cbf_gain_quant_dif are given below.

Finally, the decoder is simulated by performing the following steps:
- the excitation of the fixed codebook is computed
- the excitation of the fixed and adaptive codebook are summed
- the memory of the adaptive codebook is updated
- the memory of the filters is updated

A detailed description of these steps is given in the decoder.

Tables used for gain quantization

Index	Cba_gain_quant	Index	cba_gain_quant
0	0.1622	16	0.9989
1	0.2542	17	1.0539
2	0.3285	18	1.1183
3	0.3900	19	1.1933
4	0.4457	20	1.2877
5	0.4952	21	1.4136
6	0.5425	22	1.5842
7	0.5887	23	1.8559
8	0.6341	24	2.3603
9	0.6783	25	3.8348
10	0.7227	26	7.6697
11	0.7664	27	15.339
12	0.8104	28	30.679
13	0.8556	29	61.357
14	0.9005	30	122.71
15	0.9487	31	245.43

Table 40. Quantization table for Adaptive codebook gain

Index	Cbf_gain_quant	Index	cbf_gain_quant

0	2.4726	16	82.3374
1	3.1895	17	95.3755
2	4.2182	18	109.8997
3	5.6228	19	126.3037
4	7.3781	20	144.3995
5	9.5300	21	165.5142
6	12.1013	22	190.9742
7	15.2262	23	220.6299
8	19.0319	24	258.2699
9	23.6342	25	305.5086
10	29.1562	26	368.5894
11	35.3606	27	453.5156
12	42.8301	28	573.6164
13	51.1987	29	801.6422
14	60.6440	30	9999.9
15	70.9884	31	------

Table 41. Quantization table for fixed codebook gain

Index	cbf_gain_quant_dif	Index	cbf_gain_quant_dif
0	0.2500	4	1.3356
1	0.5378	5	1.8869
2	0.7795	6	4.2000
3	1.0230	7	9999.9

Table 42. Quantization table for differential fixed codebook gain

7.9.3.2 Multi-Pulse Excitation

For the 8 kHz sampling rate coder, the excitation signal is extracted and encoded on an analysis-by-synthesis basis with three steps, the adaptive codebook search for a periodic component, the fixed codebook search for a non-periodic component and the quantization of the gains for each component. The target signal is obtained by subtracting the zero-input response of the synthesis and perceptual weighting filters from the weighted input speech signal.

Frame Energy Quantization

The root mean squared (rms) value of subframe input samples is calculated at the last subframe. The rms value is scalar-quantized in the μ-law scale. The rms values of other subframes are obtained by linear-interpolating the quantized rms values at the last subframe of the current and the previous frames. The quantized rms values are used for gain normalization.

Open-loop Pitch Estimation and Mode Decision

The open-loop pitch estimation and mode decision procedures are jointly performed every analysis interval of 10 ms using the perceptually weighted signal. The pitch estimate is selected to minimize the squared error between the current and previous blocks of the weighted input samples. A pitch prediction gain is calculated for each analysis interval. Every frame is classified into one of the four modes based on the average pitch prediction gain. Modes 0 and 1 correspond to unvoiced and transition frames, respectively. Modes 2 and 3 correspond to voiced frames, and the latter has higher periodicity than the former. If the subframe length is 5 ms, the same estimated pitch is used in two subframes for a closed-loop pitch search.

Encoding of Excitation Signal

The excitation signal is represented by a linear combination of the adaptive codevector and the fixed codevector scaled by their respective gains. Each component of the excitation signal is successively chosen by an analysis-by-synthesis search procedure so that the perceptually weighted error between the input signal and the reconstructed signal is minimized. The adaptive codevector parameters are a closed-loop delay and a gain. The

closed-loop delay is selected in the range around the estimated open-loop delay. The adaptive codevector is generated from a block of the past excitation signal samples with the selected closed-loop delay. The fixed codevector contains several non-zero pulses. The pulse positions are restricted in an algebraic structure. The restriction table of the pulse positions are set up from the parameters. In order to improve the performance, after determining plural sets of pulse position candidates, a combination search between the pulse position candidates and the pulse amplitude index is carried out. The gains for both the adaptive and the multi-pulse codevectors are normalized and vector-quantized. The normalization factor is calculated from the quantized LP coefficients, the quantized subframe rms and an excitation signal rms. The gain codebook is changed in accordance with the mode.

Adaptive Codebook Search

For each subframe the optimal delay is determined through closed-loop analysis so that mean-squared error between the target signal $x(n)$ and the weighted synthesized signal $y_t(n)$ of the adaptive codevector is minimized.

$$E_t = \sum_{n=0}^{N-1} (x(n) - g_t y_t(n))^2, \quad t_{op} - 8 < t < t_{op} + 8$$

where t_{op} is the open-loop delay determined in the open-loop pitch analysis. g_t is the optimal gain as follows:

$$g_t = \frac{\sum_{n=0}^{N-1} x(n) y_t(n)}{\sum_{n=0}^{N-1} y_t^2(n)}$$

The optimal pitch delay is encoded with 8 bits based on the relationship between the shape_indices[0] and the delay (see Decoding process).

Fixed Codebook Search

The fixed codebook vector is represented by the pulse position and pulse amplitudes. The pulse positions and amplitudes are searched to minimize the mean-squared error between the target signal and the weighted synthesized signal.

$$E_k = \sum_{n=0}^{N-1} (x(n) - g_t y_t(n) - g_k z_k(n))^2$$

where g_t is the optimal gain as follows:

$$g_k = \frac{\sum_{n=0}^{N-1} (x(n) - g_t y_t(n)) z_k(n)}{\sum_{n=0}^{N-1} z_k^2(n)}$$

where k indicates the possible combination of the shape_indices[1] and the shape_indices[2] (see Decoding process). The fixed codebook vector is constructed from the pulse positions and the pulse amplitudes (see Decoding process). The weighted signal $z_k(n)$ is computed by filtering the fixed codebook vector through the LP synthesis filter $1/A(z)$ and the perceptual weighting filter $W(z)$.

Gain Quantization

Gains for the adaptive codevector and the fixed codevector are normalized by the prediction residual energy *rs* and scalar-quantized. The residual energy *rs* is calculated based on frame energy and reflection coefficients. The frame energy is calculated every subframe as a root mean square (RMS) value and quantized in the μ-law domain. The reflection coefficients *k*(*i*) are converted from the interpolated LPCs *int_Qlpc_coefficients*[]. Consequently, the prediction residual energy (*rs*) of a subframe is

$$rs = sbfrm\_len \cdot q\_amp^2 \cdot \prod_{i=1}^{N_p} \left(1 - k^2(i)\right),$$

where *Np* is the LP analysis order, *q_amp* is the quantized RMS amplitude and *sbfrm_len* is the subframe length.

The gain quantization is carried out by finding the pair of gain indices which minimize the error between the target and the weighted synthesized signal. The error err_g due to gain quantization is given by:

$$err_g = \sum_{i=0}^{sbfrm\_len-1} \left(\sqrt{\frac{rs}{pow_{AC}}} \cdot AdaptGainCB[m] \cdot ac\_syn(i) + \sqrt{\frac{rs}{pow_{SC}}} \cdot FixedGainCB[n] \cdot sc\_syn(i) - t\arg(i) \right)^2$$

$$pow\_ac = \sum_{i=0}^{sbfrm\_len-1} ac\_ex^2(i)$$

$$pow\_sc = \sum_{i=0}^{sbfrm\_len-1} sc\_ex^2(i)$$

where *m* and *n* are indices, *ac_ex*(*i*) is the selected adaptive codevector, *ac_syn*(*i*) is the signal which is synthesized from *ac_ex*(*i*) and perceptually weighted, *sc_ex*(*i*) is the selected stochastic codevector and *sc_syn*(*i*) is the signal which is synthesized from *sc_ex*(*i*)and perceptually weighted.

The index pair of *n* and *m* which minimize the error err_g is selected and is stored in *gain_indices*[]. Then the quantized adaptive gain *qacg*, the quantized stochastic *qscg* and the excitation signal *decoded_excitation*(*i*) are calculated as follows.

$$decoded\_excitation(i) = qacg \cdot ac\_ex(i) + qscg \cdot sc\_ex(i) \qquad (0 \le i \le sbfrm\_len - 1)$$

$$qacg = \sqrt{\frac{rs}{pow\_ac}} \cdot AdaptGainCB(m)$$

$$qscg = \sqrt{\frac{rs}{pow\_sc}} \cdot FixedGainCB(n)$$

7.9.3.3 Enhancement Multi-Pulse Excitation

The fixed codebook vector is represented by the pulse position and pulse amplitudes. The pulse positions and amplitudes are searched to minimize the mean-squared error between the target signal and the weighted synthesized signal.

$$E_k = \sum_{n=0}^{N-1} (x(n) - g_t y_t(n) - g_k z_k(n))^2$$

where g_t is the optimal gain as follows:

$$g_k = \frac{\sum_{n=0}^{N-1} (x(n) - g_t y_t(n)) z_k(n)}{\sum_{n=0}^{N-1} z_k^2(n)}$$

where k indicates the possible combination of the shape_enh_positions and the shape_enh_signs (see Decoding process). The fixed codebook vector is constructed from the pulse positions and the pulse amplitudes (see

Decoding process). The weighted signal $z_k(n)$ is computed by filtering the fixed codebook vector through the LP synthesis filter $1/A(z)$ and the perceptual weighting filter $W(z)$.

7.9.3.4 Multi-Pulse Excitation for the bandwidth extension tool

For the bandwidth extension tool, the excitation signal is extracted and encoded on an analysis-by-synthesis basis with three steps, the adaptive codebook search for a periodic component, the fixed codebook search for a non-periodic component and the quantization of the gains for each component. The target signal is obtained by subtracting the zero-input response of the synthesis and perceptual weighting filters from the weighted input speech signal.

Frame Energy Quantization

The same root mean squared (rms) value as the 8 kHz sampling rate CELP is used.

Open-loop Pitch Estimation and Mode Decision

The open-loop pitch estimation is done by converting the 8 kHz sampling rate pitch delay (see Decoding process). The same mode as the 8 kHz sampling rate CELP is used.

Encoding of Excitation Signal

The excitation signal is represented by a linear combination of the adaptive codevector and the two fixed codevectors scaled by their respective gains. The pitch delay is decided in the range around the open-loop estimated pitch delay. One of the two fixed codevectors is obtained by sampling-rate expansion of the fixed codevector used in the 8 kHz sampling rate coder. The another fixed codevector is determined by an analysis-by-synthesis search procedure.

Adaptive Codebook Search

For each subframe the optimal delay is determined through closed-loop analysis so that mean-squared error between the target signal $x(n)$ and the weighted synthesized signal $y_t(n)$ of the adaptive codevector is minimized.

$$E_t = \sum_{n=0}^{N-1} (x(n) - g_t y_t(n))^2, \quad t_{op} - 8 < t < t_{op} + 8$$

where t_{op} is the open-loop delay determined in the open-loop pitch analysis. g_t is the optimal gain as follows:

$$g_t = \frac{\sum_{n=0}^{N-1} x(n) y_t(n)}{\sum_{n=0}^{N-1} y_t^2(n)}$$

The difference between the optimal pitch delay and the open-loop pitch delay is encoded with 3 bits based on the relationship between the shape_bws_delay and the differential delay (see Decoding process).

Fixed Codebook 1 Search

The fixed codevector 1 is obtained by sampling-rate expansion of the fixed codevector used in the 8 kHz sampling rate coder (see Decoding Process).

Fixed Codebook 2 Search

The fixed codebook vector is represented by the pulse position and pulse amplitudes. The pulse positions and amplitudes are searched to minimize the mean-squared error between the target signal and the weighted synthesized signal.

$$E_k = \sum_{n=0}^{N-1} (x(n) - g_t y_t(n) - g_k z_k(n))^2$$

where g_t is the optimal gain as follows:

$$g_k = \frac{\sum_{n=0}^{N-1} (x(n) - g_t y_t(n)) z_k(n)}{\sum_{n=0}^{N-1} z_k^2(n)}$$

where k indicates the possible combination of the shape_bws_positions and the shape_bws_signs (see Decoding process). The fixed codebook vector is constructed from the pulse positions and the pulse amplitudes (see Decoding process). The weighted signal $z_k(n)$ is computed by filtering the fixed codebook vector through the LP synthesis filter $1/A(z)$ and the perceptual weighting filter $W(z)$.

Gain Quantization

Gains for the adaptive codevector and the two fixed codevector are normalized by the prediction residual energy *rs* and quantized. The residual energy *rs* is calculated based on frame energy and reflection coefficients. The frame energy is calculated every subframe as a root mean square (RMS) value and quantized in the µ-law domain. The reflection coefficients *k*(*i*) are converted from the interpolated LPCs *int_Qlpc_coefficients*[]. Consequently, the prediction residual energy (*rs*) of a subframe is

$$rs = sbfrm\_len \cdot q\_amp^2 \cdot \prod_{i=1}^{N_p} \left(1 - k^2(i)\right),$$

where *Np* is the LP analysis order, *q_amp* is the quantized RMS amplitude and *sbfrm_len* is the subframe length.

The gains for the adaptive codevector and the fixed codevector 2 are vector quantized. The gain for the fixed codevector 1 is scalar quantized. These quantization operations are achieved by minimizing the perceptually weighted distortion (closed-loop).

7.10 CELP Bitstream multiplexer

7.10.1 Tool description

The tool CELP bitstream multiplexer multiplexes a frame into the bitstream.

7.10.2 Definitions

All the bitstream elements and the helping variables have been defined in Section Helping variables and Section Bitstream elements for the MPEG-4 CELP tool-set.

7.10.3 Encoding Process

The parameters are encoded into a bitstream according to the syntax described in Section Bitstream Syntax.

8. Annex A Interface for CELP Modules

8.1 Bitstream multiplexer

```
typedef struct {
       unsigned char *p_bitstream_buffer_start;
       int buffer_length;
       int start_offset;
       int valid_bits;
} BITSTREAM;
```

8.2 Encoder Module Prototypes

```
void celp_initialisation_encoder
(
BITSTREAM  * const p_bitstream,      /* In/Out: Bitstream                          */
long     bit_rate,                   /* In: bit rate                               */
long     sampling_frequency,         /* In: sampling frequency                     */
long     SampleRateMode,             /* In: SampleRate Mode                        */
long     QuantizationMode,           /* In: Type of Quantization                   */
long     FineRateControl,            /* In: Fine Rate Control switch               */
long     LosslessCodingMode,         /* In: Lossless Coding Mode                   */
long     *WB_Configuration,          /* In: Wideband configuration                 */
long     Wideband_VQ,                /* Out: Wideband VQ mode                      */
long     *NB_Configuration,          /* Out: Narrowband configuration              */
long     NumEnhLayers,               /* In: Number of Enhancement Layers           */
long     BandwidthScalabilityMode,   /* In: bandwidth switch                       */
long     *BWS_Configuration,         /* Out: BWS_configuration                     */
long     BWS_nb_bitrate,             /* In: narrowband bitrate for BWS             */
long     *frame_size,                /* Out: frame size                            */
long     *n_subframes,               /* Out: number of  subframes                  */
long     *sbfrm_size,                /* Out: subframe size                         */
long     *lpc_order,                 /* Out: LP analysis order                     */
long     *num_lpc_indices,           /* Out: number of LPC indices                 */
long     *num_shape_cbks,            /* Out: number of Shape Codebooks             */
long     *num_gain_cbks,             /* Out: number of Gain Codebooks              */
long     *n_lpc_analysis,            /* Out: number of LP analysis per frame       */
long     **window_offsets,           /* Out: window offset for each LP ana         */
long     **window_sizes,             /* Out: window size for each LP ana           */
long     *n_lag_candidates,          /* Out: number of pitch candidates            */
float    *min_pitch_frequency  ,     /* Out: minimum pitch frequency               */
float    *max_pitch_frequency,       /* Out: maximum pitch frequency               */
long     **org_frame_bit_allocation, /* Out: bit num. for each index               */
void     **InstanceContext           /* Out: handle to initialised instance context */
);


void celp_coder
(
float     **InputSignal,             /* In: Multichannel Speech                    */
BITSTREAM * p_bitstream,             /* Out: Bitstream                             */
long        sampling_frequency,      /* In:  Sampling Frequency                    */
long        bit_rate,                /* In:  Bit rate                              */
long        SampleRateMode,
long        QuantizationMode,        /* In: Type of Quantization                   */
long        FineRateControl,         /* In: Fine Rate Control switch               */
long        LosslessCodingMode,      /* In: Lossless Coding Mode                   */
long        WB_Configuration,        /* In: Wideband configuration                 */
long        Wideband_VQ,             /* In: Wideband VQ mode                       */
long        NB_Configuration,        /* In: Narrowband configuration               */
long        NumEnhLayers,            /* In: Number of Enhancement Layers           */
long        BandwidthScalabilityMode,/* In: bandwidth switch                       */
long        BWS_Configuration,       /* In: BWS_configuration                      */
long        PreProcessingSW,         /* In: PreProcessingSW                        */
long        frame_size,              /* In:  Frame size                            */
long        n_subframes,             /* In:  Number of subframes                   */
long        sbfrm_size,              /* In:  Subframe size                         */
long        lpc_order,               /* In:  Order of LPC                          */
long        num_lpc_indices,         /* In:  Number of LPC indices                 */
long        num_shape_cbks,          /* In:  Number of Shape Codebooks             */
long        num_gain_cbks,           /* In:  Number of Gain Codebooks              */
long        n_lpc_analysis,          /* In:  Number of LPCs per frame              */
```

```
long        window_offsets[],        /* In:  Offset for LPC-frame v.window         */
long        window_sizes[],          /* In:  LPC Analysis Window size              */
long        max_n_lag_candidates,    /* In:  Maximum search candidates             */
float       min_pitch_frequency,     /* IN:  Min Pitch Frequency                   */
float       max_pitch_frequency,     /* IN:  Max Pitch Frequency                   */
long        org_frame_bit_allocation[], /* In: Frame BIt alolocation               */
void        *InstanceContext         /* In/Out: instance context                   */
);


void celp_close_encoder
(
    long SampleRateMode,
    long BandwidthScalabilityMode,
    long sbfrm_size,              /* In: subframe size                 */
    long frame_bit_allocation[],  /* In: bit num. for each index       */
    long window_offsets[],        /* In: window offset for each LP ana  */
    long window_sizes[],          /* In: window size for each LP ana    */
    long n_lpc_analysis,          /* In: number of LP analysis/frame    */
    void **InstanceContext        /* In/Out: handle to instance context */
);


void                            /* Return Value: Void                  */
celp_preprocessing
(
float InputSignal[],            /* In:  Input Signal                   */
float PP_InputSignal[],         /* Out: Preprocessed Input Signal      */
float *prev_x,                  /* In/Out: Previous x                  */
float *prev_y,                  /* In/Out: Previous y                  */
long  frame_size,               /* In:  Number of samples in frame     */
long  sampling_frequency        /* In:  Sampling Frequency             */
);


void celp_lpc_analysis
(
float PP_InputSignal[],         /* In:  Input Signal                    */
float lpc_coefficients[],       /* Out: LPC Coefficients[0..lpc_order-1] */
float *first_order_lpc_par,     /* Out: a_parameter for 1st-order fit    */
long  frame_size,               /* In:  Number of samples in frame       */
long  window_offsets[],         /* In:  offset for window w.r.t curr. fr */
long  window_sizes[],           /* In:  LPC Analysis-Window Size         */
float *windows[],               /* In:  Array of LPC Analysis windows    */
float gamma_be[],               /* In:  Bandwidth expansion coefficients */
long  lpc_order,                /* In:  Order of LPC                     */
long  n_lpc_analysis            /* In:  Number of LP analysis/frame      */
);


void SQ_celp_lpc_quantizer
(
float lpc_coefficients[],      /* In:  Current unquantised a-pars            */
float int_Qlpc_coefficients[], /* Out: Qaunt/interpolated a-pars             */
long  lpc_indices[],           /* Out: Codes thar are transmitted            */
long  lpc_order,               /* In:  Order of LPC                          */
long  num_lpc_indices,         /* In:  Number of packes LPC codes            */
long  n_lpc_analysis,          /* In:  Number of LPC/frame                   */
long  n_subframes,             /* In:  Number of subframes                   */
long  *interpolation_flag,     /* Out: Interpolation Flag                    */
long  *send_lpc_flag,          /* Out: Send LPC flag                         */
PHI_PRIV_TYPE *PHI_Priv        /* In/Out: PHI private data (instance context) */
);


void VQ_celp_lpc_quantizer
(
float lpc_coefficients[],      /* In:  Current unquantised a-pars           */
float lpc_coefficients_8[],    /* In:  Current unquantised a-pars(8 kHz)    */
float int_Qlpc_coefficients[], /* Out: Qaunt/interpolated a-pars            */
long  lpc_indices[],           /* Out: Codes thar are transmitted           */
long  lpc_order,               /* In:  Order of LPC                         */
long  num_lpc_indices,         /* In:  Number of packes LPC codes           */
long  n_lpc_analysis,          /* In:  Number of LPC/frame                  */
long  n_subframes,             /* In:  Number of subframes                  */
long  *interpolation_flag,     /* Out: Interpolation Flag                   */
long  *send_lpc_flag,          /* Out: Send LPC flag                        */
long  Wideband_VQ,
PHI_PRIV_TYPE *PHI_Priv       /* In/Out: PHI private data (instance context) */
);
```

```
void celp_lpc_analysis_filter
(
float PP_InputSignal[],         /* In:  Input Signal [0..sbfrm_size-1]         */
float lpc_residual[],           /* Out: LPC residual [0..sbfrm_size-1]         */
float int_Qlpc_coefficients[],  /* In:  LPC Coefficients[0..lpc_order-1]       */
long  lpc_order,                /* In:  Order of LPC                           */
long  sbfrm_size,               /* In:  Number of samples in subframe          */
PHI_PRIV_TYPE *PHI_Priv         /* In/Out: PHI private data (instance context) */
);


void celp_weighting_module
(
float lpc_coefficients[],       /* In:  LPC Coefficients[0..lpc_order-1] */
long  lpc_order,                /* In:  Order of LPC                     */
float Wnum_coeff[],             /* Out: num. coeffs[0..lpc_order-1]      */
float Wden_coeff[],             /* Out: den. coeffs[0..lpc_order-1]      */
float gamma_num,                /* In:  Weighting factor: numerator      */
float gamma_den                 /* In:  Weighting factor: denominator    */
);


void celp_excitation_analysis
(
float PP_InputSignal[],           /* Preprocessed Input signal           */
float lpc_residual[],             /* Inverse Filtered Signal             */
float int_Qlpc_coefficients[],    /* Interpolated LPC Coeffs             */
long  lpc_order,                  /* Order of LPC                        */
float Wnum_coeff[],               /* Weighting Filter: Numerator         */
float Wden_coeff[],               /* Weighting Filter: Denominator       */
float first_order_lpc_par,        /* apar corresponding to 1st-order fit */
long  lag_candidates[],           /* Array of Lag candidates             */
long  n_lag_candidates,           /* Number of lag candidates            */
long  frame_size,                 /* Number of samples in the frame      */
long  sbfrm_size,                 /* Number of samples in the subframe   */
long  n_subframes,                /* Number of subframes                 */
long  signal_mode,                /* Configuration Input                 */
long  frame_bit_allocation[],     /* Configuration Input                 */
long  shape_indices[],            /* Adaptive and Fixed codebook lags    */
long  gain_indices[],             /* Adaptive and Fixed codebook gains   */
long  num_shape_cbks,             /* Number of shape codebooks           */
long  num_gain_cbks,              /* Number of gain codebooks            */
long  *rms_index,                 /* RMS Value ????                      */
float decoded_excitation[]        /* Synthesised Signal                  */
);


void nb_abs_classifier
(
    float PP_InputSignal[],                     /* in: preprocessed input signal */
    long org_frame_bit_allocation[],            /* in: bit number for each index */
    long *signal_mode,                          /* out: signal mode */
    long frame_bit_allocation[],                /* out: bit number for each index */
    long frame_size,                            /* in: frame size */
    long sampling_frequency,                    /* in: sampling frequency */
    long prev_adaptive_pitch_lag,               /* in: pitch lag of previous frame */
        long num_indices
);


void nb_abs_lpc_quantizer
(
    float lpc_coefficients[],           /* in: LPC */
    float int_Qlpc_coefficients[],      /* out: quantized & interpolated LPC */
    long lpc_indices[],                 /* out: LPC code indices */
    long lpc_order,                     /* in: order of LPC */
    long num_lpc_indices,               /* in: number of LPC indices */
    long n_lpc_analysis,                /* in: number of LP analysis per frame */
    long n_subframes,                   /* in: number of subframes */
    long *interpolation_flag,           /* out: interpolation flag */
    long signal_mode,                   /* inp: signal mode */
    long frame_bit_allocation[],        /* in: bit number for each index */
    long sampling_frequency,            /* in: sampling frequency */
    float *prev_Qlsp_coefficients
);


void bws_lpc_quantizer
(
        float   lpc_coefficients_16[],
        float   int_Qlpc_coefficients_16[],
        long    lpc_indices_16[],
```

```
        long    lpc_order_8,
        long    lpc_order_16,
        long    num_lpc_indices_16,
        long    n_lpc_analysis_16,
        long    n_subframes_16,
        float   buf_Qlsp_coefficients_16[],
        float   prev_Qlsp_coefficients_16[],
        long    frame_bit_allocation[]
);


void nb_abs_excitation_analysis
(
        float PP_InputSignal[],         /* in: preprocessed input signal */
        float lpc_residual[],           /* in: LP residual signal */
        float int_Qlpc_coefficients[], /* in: interpolated LPC */
        long lpc_order,                 /* in: order of LPC */
        float Wnum_coeff[],             /* in: weighting coeff.(numerator) */
        float Wden_coeff[],             /* in: weighting coeff.(denominator) */
        float first_order_lpc_par,      /* in: first order LPC */
        long lag_candidates[],          /* in: lag candidates */
        long n_lag_candidates,          /* in: number of lag candididates */
        long frame_size,                /* in: frame size */
        long sbfrm_size,                /* in: subframe size */
        long n_subframes,               /* in: number of subframes */
        long *signal_mode,              /* out: signal mode */
        long frame_bit_allocation[],    /* in: bit number for each index */
        long shape_indices[],           /* out: shape code indices */
        long gain_indices[],            /* out: gain code indices */
        long num_shape_cbks,            /* in: number of shape codebooks */
        long num_gain_cbks,             /* in: number of gain codebooks */
        long *rms_index,                /* out: RMS code index */
        float decoded_excitation[],     /* out: decoded excitation */
        long num_enhstages,
        float bws_mp_exc[]
);


void bws_excitation_analysis
(
        float PP_InputSignal[],         /* in: preprocessed input signal */
        float int_Qlpc_coefficients[], /* in: interpolated LPC */
        long lpc_order,                 /* in: order of LPC */
        float Wnum_coeff[],             /* in: weighting coeff.(numerator) */
        float Wden_coeff[],             /* in: weighting coeff.(denominator) */
        long frame_size,                /* in: frame size */
        long sbfrm_size,                /* in: subframe size */
        long n_subframes,               /* in: number of subframes */
        long signal_mode,               /* in: signal mode */
        long frame_bit_allocation[],    /* in: bit number for each index */
        long shape_indices[],           /* out: shape code indices */
        long gain_indices[],            /* out: gain code indices */
        long num_shape_cbks,            /* in: number of shape codebooks */
        long num_gain_cbks,             /* in: number of gain codebooks */
        float decoded_excitation[],     /* out: decoded excitation */
        float bws_mp_exc[],
        long  *acb_index_8,
        long  rms_index                 /* in: RMS code index */
);
```

8.3 Decoder Module Protypes

```
void celp_initialisation_decoder
(
BITSTREAM * p_bitstream,                /* In: Bitstream                        */
long     bit_rate,                      /* in: bit rate                         */
long     complexity_level,              /* In: complexity level decoder         */
long     reduced_order,                 /* In: reduced order decoder            */
long     DecEnhStage,
long     DecBwsMode,
long     PostFilterSW,
long     *frame_size,                   /* Out: frame size                      */
long     *n_subframes,                  /* Out: number of  subframes            */
long     *sbfrm_size,                   /* Out: subframe size                   */
long     *lpc_order,                    /* Out: LP analysis order               */
long     *num_lpc_indices,              /* Out: number of LPC indices           */
long     *num_shape_cbks,               /* Out: number of Shape Codeb.          */
long     *num_gain_cbks,                /* Out: number of Gain Codeb.           */
long     **org_frame_bit_allocation,    /* Out: bit num. for each index         */
```

```
long     * SampleRateMode,               /* Out: SampleRate Mode                        */
long     * QuantizationMode,             /* Out: Type of Quantization                   */
long     * FineRateControl,              /* Out: Fine Rate Control switch               */
long     * LosslessCodingMode,           /* Out: Lossless Coding Mode                   */
long     * WB_Configuration,             /* Out: Wideband configuration                 */
long     * Wideband_VQ,                  /* Out: Wideband VQ mode                       */
long     * NB_Configuration,             /* Out: Narrowband configuration               */
long     * NumEnhLayers,                 /* Out: Number of Enhancement Layers           */
long     * BandwidthScalabilityMode,     /* Out: bandwidth switch                       */
long     * BWS_configuration,            /* Out: BWS_configuration                      */
void     **InstanceContext               /* Out: handle to initialised instance context */
);


void celp_decoder
(
BITSTREAM * p_bitstream,               /* In: Bitstream                    */
float     **OutputSignal,              /* Out: Multichannel Output         */
long        SampleRateMode,            /* In: SampleRate Mode              */
long        QuantizationMode,          /* In: Type of Quantization         */
long        FineRateControl,           /* In: Fine Rate Control switch     */
long        LosslessCodingMode,        /* In: Lossless Coding Mode         */
long        WB_Configuration,          /* In: Wideband configuration       */
long        Wideband_VQ,               /* In: Wideband VQ mode             */
long        NB_Configuration,          /* In: Narrowband configuration     */
long        NumEnhLayers,              /* In: Number of Enhancement Layers */
long        BandwidthScalabilityMode,  /* In: bandwidth switch             */
long        BWS_configuration,         /* In: BWS_configuration            */
long        frame_size,                /* In:  Frame size                  */
long        n_subframes,               /* In:  Number of subframes         */
long        sbfrm_size,                /* In:  Subframe size               */
long        lpc_order,                 /* In:  Order of LPC                */
long        num_lpc_indices,           /* In:  Number of LPC indices       */
long        num_shape_cbks,            /* In:  Number of Shape Codebooks   */
long        num_gain_cbks,             /* In:  Number of Gain Codebooks    */
long        *org_frame_bit_allocation, /* In: bit num. for each index      */
void        *InstanceContext           /* In: pointer to instance context  */
);


void celp_close_decoder
(
   long SampleRateMode,
   long BandwidthScalabilityMode,
   long frame_bit_allocation[],         /* In: bit num. for each index        */
   void **InstanceContext               /* In/Out: handle to instance context */
);


void SQ_celp_lpc_decode
(
long  lpc_indices[],           /* In: Received Packed LPC Codes               */
float int_Qlpc_coefficients[], /* Out: Qaunt/interpolated a-pars              */
long  lpc_order,               /* In:  Order of LPC                           */
long  num_lpc_indices,         /* In:  Number of packes LPC codes             */
long  n_subframes,             /* In:  Number of subframes                    */
long  interpolation_flag,      /* In:  Was interpolation done?                */
PHI_PRIV_TYPE *PHI_Priv        /* In/Out: PHI private data (instance context) */
);


void VQ_celp_lpc_decode
(
long  lpc_indices[],           /* In: Received Packed LPC Codes               */
float int_Qlpc_coefficients[], /* Out: Qaunt/interpolated a-pars              */
long  lpc_order,               /* In:  Order of LPC                           */
long  num_lpc_indices,         /* In:  Number of packes LPC codes             */
long  n_subframes,             /* In:  Number of subframes                    */
long  interpolation_flag,      /* In:  Was interpolation done?                */
long  Wideband_VQ,             /* In:  Wideband VQ switch                     */
PHI_PRIV_TYPE *PHI_Priv        /* In/Out: PHI private data (instance context) */
);


void celp_excitation_generation
(
long  shape_indices[],          /* Lag indices for Adaptive & Fixed cbks    */
long  gain_indices[],           /* Gains for Adaptive & Fixed cbks          */
long  num_shape_cbks,           /* Number of shape codebooks                */
long  num_gain_cbks,            /* Number of gain codebooks                 */
long  rms_index,                /* NOT USED HERE: RMS value subframe ??     */
float int_Qlpc_coefficients[],  /* Interpolated LPC coeffs of subframe      */
```

```
long  lpc_order,                /* Order of LPC                              */
long  sbfrm_size,               /* In: Number of samples in the subframe     */
long  n_subframes,              /* In: Number of subframes                   */
long  signal_mode,              /* In: Configuration Input                   */
long  frame_bit_allocation[],   /* In: Configuration Input                   */
float excitation[],             /* Out: Excitation Signal                    */
long  *acb_delay,
float *adaptive_gain,
PHI_PRIV_TYPE * PHI_Priv        /* In/Out: private data (instance context)  */
);


void celp_lpc_synthesis_filter
(
float excitation[],             /* In:  Input Signal [0..sbfrm_size-1]          */
float synth_signal[],           /* Out: LPC residual [0..sbfrm_size-1]          */
float int_Qlpc_coefficients[],  /* In:  LPC Coefficients[0..lpc_order-1]        */
long  lpc_order,                /* In:  Order of LPC                            */
long  sbfrm_size,               /* In:  Number of samples in subframe           */
PHI_PRIV_TYPE *PHI_Priv         /* In/Out: PHI private data (instance context)  */
);


void celp_postprocessing
(
const float synth_signal[],          /* In:  Input sig [0..sbfrm_size -1]       */
      float PP_synth_signal[],       /* Out: Postprocessed ooutput              */
const float int_Qlpc_coefficients[], /* In:  Decoded LPC coefficients           */
const long  lpc_order,               /* In:  Order of LPC                       */
const long  sbfrm_size,              /* In:  #Samps to be processed             */
const long  acb_delay,               /* In:  Pitch-like information             */
const float adaptive_gain,
PHI_PRIV_TYPE *PHI_Priv              /* In/Out: private data (instance context) */
);


void nb_abs_lpc_decode
(
    long lpc_indices[],                  /* in: LPC code indices */
    float int_Qlpc_coefficients[],       /* out: quantized & interpolated LPC */
    long lpc_order,                      /* in: order of LPC */
    long num_lpc_indices,                /* in: number of LPC indices */
    long n_subframes,                    /* in: number of subframes */
    long interpolation_flag,             /* in: interpolation flag */
    long signal_mode,                    /* in: signal mode */
    long org_frame_bit_allocation[],     /* in: bit number for each index */
    float *prev_Qlsp_coefficients
);


void bws_lpc_decoder
(
        long    lpc_indices_16[],
        float   int_Qlpc_coefficients_16[],
        long    lpc_order_8,
        long    lpc_order_16,
        long    n_subframes_16,
        long    num_lpc_indices_16,
        float   buf_Qlsp_coefficients_16[],
        float   prev_Qlsp_coefficients_16[],
        long    frame_bit_allocation[]
);


void nb_abs_excitation_generation
(
        long shape_indices[],            /* in: shape code indices */
        long gain_indices[],             /* in: gain code indices */
        long num_shape_cbks,             /* in: number of shape codebooks */
    long num_gain_cbks,                  /* in: number of gain codebooks */
        long rms_index,                  /* in: RMS code index */
        float int_Qlpc_coefficients[],   /* in: interpolated LPC */
        long lpc_order,                  /* in: order of LPC */
        long sbfrm_size,                 /* in: subframe size */
        long n_subframes,                /* in: number of subframes */
        long signal_mode,                /* in: signal mode */
        long org_frame_bit_allocation[], /* in: bit number for each index */
        float excitation[],              /* out: decoded excitation */
        float bws_mp_exc[],              /* out: decoded excitation */
        long *acb_delay,                 /* out: adaptive code delay */
        float *adaptive_gain,            /* out: adaptive code gain */
    long dec_enhstages,
```

```
    long postfilter
);


void bws_excitation_generation
(
        long shape_indices[],              /* in: shape code indices */
        long gain_indices[],               /* in: gain code indices */
        long num_shape_cbks,               /* in: number of shape codebooks */
        long num_gain_cbks,                /* in: number of gain codebooks */
        long rms_index,                    /* in: RMS code index */
        float int_Qlpc_coefficients[],     /* in: interpolated LPC */
        long lpc_order,                    /* in: order of LPC */
        long sbfrm_size,                   /* in: subframe size */
        long n_subframes,                  /* in: number of subframes */
        long signal_mode,                  /* in: signal mode */
        long org_frame_bit_allocation[],   /* in: bit number for each index */
        float excitation[],                /* out: decoded excitation */
        float bws_mp_exc[],                /* in: decoded mp excitation */
        long acb_indx_8[],                 /* in: acb_delay */
        long *acb_delay,                   /* out: adaptive code delay */
        float *adaptive_gain,              /* out: adaptive code gain */
        long postfilter
);


void nb_abs_postprocessing
(
        float synth_signal[],              /* input */
        float PP_synth_signal[],           /* output */
        float int_Qlpc_coefficients[],     /* input */
        long lpc_order,                    /* configuration input */
        long sbfrm_sizes,                  /* configuration input */
        long acb_delay,                    /* input */
        float adaptive_gain,               /* input */
        long dec_bwsmode
);
```

9. ANNEX B Tables

9.1 BWS Tool Downsampling filter Table coefficients

LPF Coefficients for BWS Tool

0	0.487498	25	0.006314	50	0.003645	75	0.001263
1	0.318007	26	0.009227	51	0.001586	76	-0.000190
2	0.012479	27	-0.005043	52	-0.003242	77	-0.001148
3	-0.105197	28	-0.008776	53	-0.001706	78	0.000159
4	-0.012414	29	0.003938	54	0.002858	79	0.005156
5	0.062159	30	0.008312	55	0.001785	80	-0.000092
6	0.012306	31	-0.002977	56	-0.002497		
7	-0.043381	32	-0.007836	57	-0.001830		
8	-0.012155	33	0.002141	58	0.002158		
9	0.032701	34	0.007355	59	0.001844		
10	0.011965	35	-0.001414	60	-0.001842		
11	-0.025712	36	-0.006869	61	-0.001831		
12	-0.011735	37	0.000785	62	0.001551		
13	0.020722	38	0.006383	63	0.001794		
14	0.011468	39	-0.000241	64	-0.001284		
15	-0.016942	40	-0.005902	65	-0.001737		
16	-0.011167	41	-0.000222	66	0.001042		
17	0.013954	42	0.005428	67	0.001663		
18	0.010830	43	0.000616	68	-0.000825		
19	-0.011527	44	-0.004961	69	-0.001576		
20	-0.010467	45	-0.000942	70	0.000632		
21	0.009494	46	0.004508	71	0.001478		
22	1.01E-02	47	0.001210	72	-0.000462		
23	-7.78E-03	48	-0.004068	73	-0.001373		
24	-9.66E-03	49	-0.001423	74	0.000315		

9.2 Initial tables for lossless coding

In this annex, the initial Huffman tables **hufcod[***LAR_coeff***][***level_nr***]** are defined. The index *LAR_coeff* selects the table corresponding to the required LAR coefficient. The index *level_nr* selects the level nummer. In the distribution of the codeword lengths for each LAR coefficient are given.

LAR coefficients: 0 #levels: 36			
0 : 1110010	9 : 1111010	18 : 10000	27 : 0101
1 : 1110011	10 : 1111011	19 : 10001	28 : 0110
2 : 1110100	11 : 1111100	20 : 10010	29 : 0111
3 : 1110101	12 : 1111101	21 : 10011	30 : 10110
4 : 1110110	13 : 1111110	22 : 10100	31 : 10111
5 : 1110111	14 : 1111111	23 : 10101	32 : 11000
6 : 1111000	15 : 110100	24 : 0010	33 : 11001
7 : 1111001	16 : 110101	25 : 0011	34 : 110111
8 : 000	17 : 110110	26 : 0100	35 : 111000

LAR coefficients: 1 #levels: 28		
0 : 11111100	9 : 11000	18 : 1001
1 : 11111101	10 : 11001	19 : 11010

2 : 1111010	11 : 0010	20 : 11011
3 : 1111011	12 : 0011	21 : 000
4 : 111010	13 : 0100	22 : 1010
5 : 111011	14 : 0101	23 : 11100
6 : 111100	15 : 0110	24 : 1111100
7 : 10110	16 : 0111	25 : 1111101
8 : 10111	17 : 1000	26 : 11111110
		27 : 11111111

LAR coefficients: 2 #levels: 15	
0 : 1111110	8 : 011
1 : 11100	9 : 100
2 : 11101	10 : 1100
3 : 1010	11 : 1101
4 : 1011	12 : 11110
5 : 000	13 : 111110
6 : 001	14 : 1111111
7 : 010	

LAR coefficients: 3 #levels: 14	
0 : 1111110	7 : 011
1 : 1111111	8 : 100
2 : 11100	9 : 00
3 : 11101	10 : 1100
4 : 1010	11 : 1101
5 : 1011	12 : 11110
6 : 010	13 : 111110

LAR coefficients: 4 #levels: 13	
0 : 11111110	7 : 101
1 : 111110	8 : 1101
2 : 11100	9 : 11101
3 : 1100	10 : 11110
4 : 100	11 : 1111110
5 : 00	12 : 11111111
6 : 01	

LAR coefficients: 5 #levels: 13	
0 : 111111110	7 : 00
1 : 111111111	8 : 01
2 : 1111110	9 : 101
3 : 111110	10 : 1110
4 : 1100	11 : 11110
5 : 1101	12 : 11111110
6 : 100	

LAR coefficients: 6 #levels: 12	
0 : 1111110	6 : 01
1 : 11100	7 : 101
2 : 11101	8 : 1101
3 : 1100	9 : 11110
4 : 100	10 : 111110
5 : 00	11 : 1111111

LAR coefficients: 7 #levels: 11	
0 : 11111110	6 : 00

1 : 1111110	7 : 01
2 : 11110	8 : 110
3 : 1110	9 : 111110
4 : 100	10 : 11111111
5 : 101	

LAR coefficients: 8 #levels: 9
0 : 1111110
1 : 111110
2 : 1110
3 : 00
4 : 01
5 : 10
6 : 110
7 : 11110
8 : 1111111

LAR coefficients: 9 #levels: 8
0 : 111110
1 : 11110
2 : 110
3 : 00
4 : 01
5 : 10
6 : 1110
7 : 111111

LAR coefficients: 10 #levels: 8
0 : 1111110
1 : 111110
2 : 1110
3 : 0
4 : 10
5 : 110
6 : 11110
7 : 1111111

LAR coefficients: 11 #levels: 7
0 : 11110
1 : 1110
2 : 00
3 : 01
4 : 10
5 : 110
6 : 11111

LAR coefficients: 12 #levels: 7
0 : 111110
1 : 1110
2 : 0
3 : 10
4 : 110
5 : 11110
6 : 111111

LAR coefficients: 13 #levels: 8
0 : 1111110
1 : 11110
2 : 110
3 : 10
4 : 0
5 : 1110
6 : 111110
7 : 1111111

LAR coefficients: 14 #levels: 7
0 : 111110
1 : 1110
2 : 0
3 : 10
4 : 110
5 : 11110
6 : 111111

LAR coefficients: 15 #levels: 6
0 : 11110
1 : 110
2 : 10
3 : 0
4 : 1110
5 : 11111

LAR coefficients: 16 #levels: 6
0 : 11110
1 : 1110
2 : 0
3 : 10
4 : 110
5 : 11111

LAR coefficients: 17 #levels: 7
0 : 111110
1 : 11110
2 : 110
3 : 10
4 : 0
5 : 1110
6 : 111111

LAR coefficients: 18 #levels: 6
0 : 11110
1 : 110
2 : 0
3 : 10
4 : 1110
5 : 11111

LAR coefficients: 19 #levels: 6
0 : 11110

1 : 110
2 : 10
3 : 0
4 : 1110
5 : 11111

		#code words of length								
Lar	**#levels**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
0	36	0	0	14	5	10	6	1	0	0
1	28	0	4	4	3	7	9	1	0	0
2	15	0	0	2	1	3	4	5	0	0
3	14	0	0	2	1	3	4	3	1	0
4	13	0	2	1	1	3	2	2	2	0
5	13	2	1	1	1	1	3	2	2	0
6	12	0	0	2	1	3	2	2	2	0
7	11	0	2	1	1	1	1	3	2	0
8	9	0	0	2	1	1	1	1	3	0
9	8	0	0	0	2	1	1	1	3	0
10	8	0	0	2	1	1	1	1	1	1
11	7	0	0	0	0	2	1	1	3	0
12	7	0	0	0	2	1	1	1	1	1
13	8	0	0	2	1	1	1	1	1	1
14	7	0	0	0	2	1	1	1	1	1
15	6	0	0	0	0	2	1	1	1	1
16	6	0	0	0	0	2	1	1	1	1
17	7	0	0	0	2	1	1	1	1	1
18	6	0	0	0	0	2	1	1	1	1
19	6	0	0	0	0	2	1	1	1	1

Table 43. Initial code-lengths for generating the Huffman tables

9.3 LSP VQ Tables and Gain VQ Tables for 8 kHz sampling rate

Table: lspnw_1a[160]

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	-0.005002	-0.009553	-0.008342	-0.000317	-0.006073	-0.000545	-0.006360	0.004155
1	-0.007131	-0.019459	-0.007332	0.008566	-0.005555	0.005556	-0.010595	0.017529
2	-0.011556	-0.022138	-0.017086	0.014243	-0.009087	0.007195	-0.015248	0.027887
3	-0.034704	-0.031549	-0.029616	0.015577	-0.033254	0.005670	-0.028523	0.047464
4	-0.035823	-0.027451	-0.028753	0.013382	-0.027487	0.009952	-0.031861	0.050407
5	-0.050852	-0.018956	-0.122493	0.021996	-0.045370	0.002120	-0.049798	0.044292
6	-0.083804	0.002329	-0.032297	0.059861	-0.060636	0.013608	-0.052506	0.112393
7	-0.089622	0.005887	-0.059733	0.044310	-0.060793	0.002854	-0.043725	0.070975
8	-0.105240	0.028916	-0.078945	0.098373	-0.168483	0.008050	-0.037686	0.147884
9	-0.222625	0.012649	-0.096674	0.091453	-0.122121	0.076833	-0.020089	0.195782

	Index 8	Index 9	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	-0.004386	-0.000773	-0.002279	0.005200	-0.003397	0.002866	-0.000428	0.003532
1	-0.006992	0.006831	-0.004728	0.027845	-0.005879	0.014422	0.004857	0.022112
2	-0.013908	0.012656	-0.005525	0.047208	-0.008510	0.030372	0.003344	0.049537
3	-0.031122	0.014254	-0.013397	0.047701	-0.025400	0.026121	0.000145	0.061526
4	-0.031898	0.019049	-0.012423	0.051548	-0.027141	0.057387	0.004969	0.079777
5	-0.063496	0.023710	-0.020448	0.073417	-0.033780	0.043612	-0.002067	0.101181
6	-0.063261	0.024145	-0.019350	0.078598	-0.048384	0.048367	0.001000	0.142465
7	-0.149957	0.044813	-0.032569	0.140266	-0.060880	0.081080	0.000204	0.196492
8	-0.061107	0.023280	-0.042119	0.103923	-0.068691	0.032492	-0.008524	0.189858
9	-0.109932	0.009549	-0.070736	0.104443	-0.125967	0.095869	-0.021612	0.214706

static float nec_lspnw_1b[1280] = {

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	1.331424	1.426288	1.410787	1.520185	1.229077	1.396079	1.272960	1.650291
1	1.436251	1.537470	1.540626	1.721434	1.314904	1.500190	1.409893	1.762308
2	1.544155	1.673326	1.641212	1.856254	1.476630	1.758007	1.682794	1.890709
3	1.655895	1.818836	1.780241	1.985196	1.846687	1.891747	1.902321	2.010978
4	1.814177	2.031235	1.907111	2.170217	2.084517	2.015053	2.085217	2.136753
5	2.029328	2.223329	1.996817	2.323602	2.262692	2.228863	2.258935	2.224452
6	2.277232	2.405871	2.132169	2.448956	2.456607	2.454860	2.438040	2.361544
7	2.556229	2.522768	2.433232	2.592527	2.583520	2.625707	2.607625	2.473177
8	2.731484	2.629238	2.694906	2.757081	2.728730	2.752453	2.762280	2.648949
9	2.878972	2.749689	2.848046	2.905166	2.881977	2.913326	2.934570	2.767421

	Index 8	Index 9	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	1.378606	1.486205	1.506362	1.591833	1.290855	1.559354	1.389494	1.735760
1	1.491385	1.619724	1.620753	1.734533	1.378508	1.639085	1.509490	1.841807
2	1.626360	1.773898	1.718508	1.882093	1.433515	1.724581	1.649682	1.946420
3	1.876187	1.908814	1.828538	2.043731	1.583587	1.831671	1.798110	2.043760
4	2.067285	2.065499	1.937165	2.181013	1.944300	2.028106	1.996394	2.169381
5	2.175182	2.202107	2.049910	2.322880	2.180935	2.259484	2.149770	2.297613
6	2.321251	2.351223	2.343805	2.490124	2.381061	2.445954	2.254223	2.461981
7	2.548024	2.469327	2.605530	2.606179	2.552161	2.613314	2.391366	2.575161
8	2.746486	2.620170	2.762257	2.732304	2.719845	2.763839	2.640599	2.732298
9	2.893359	2.727764	2.912454	2.960595	2.902193	2.929481	2.831955	2.816937

	Index 16	Index 17	Index 18	Index 19	Index 20	Index 21	Index 22	Index 23
0	1.423344	1.476708	1.398696	1.551870	1.325005	1.427358	1.368592	1.628044
1	1.531424	1.624369	1.563865	1.790956	1.465195	1.668191	1.571382	1.741669
2	1.649844	1.800694	1.786272	1.905167	1.625798	1.781458	1.722109	1.890020
3	1.772980	1.972114	1.902890	2.039643	1.740576	1.885730	1.856748	2.052896
4	1.871129	2.161017	2.008751	2.193366	1.990747	2.079617	2.096992	2.240948
5	2.026479	2.287414	2.117884	2.381575	2.242444	2.248660	2.254360	2.355382
6	2.404526	2.429964	2.297783	2.502944	2.426318	2.438320	2.409942	2.459827
7	2.614092	2.533778	2.564324	2.646649	2.563133	2.621334	2.573425	2.550745
8	2.746449	2.657156	2.735492	2.780472	2.726024	2.758821	2.759133	2.678059
9	2.853537	2.762098	2.879911	2.939791	2.919034	2.928730	2.935493	2.773918

	Index 24	Index 25	Index 26	Index 27	Index 28	Index 29	Index 30	Index 31
0	1.381184	1.543290	1.590224	1.589042	1.449497	1.493503	1.437024	1.730529
1	1.544035	1.719491	1.674979	1.735868	1.540957	1.664104	1.652807	1.859522
2	1.687346	1.885426	1.780515	1.936238	1.663040	1.861118	1.885226	2.027293
3	1.838559	2.010899	1.947840	2.086147	1.981734	1.985555	2.019837	2.171734
4	2.076765	2.157243	2.166451	2.201984	2.182068	2.164928	2.150656	2.310551
5	2.266094	2.282095	2.300291	2.337221	2.301394	2.326989	2.240898	2.398206
6	2.573117	2.424096	2.425953	2.465062	2.422727	2.494353	2.383557	2.502954
7	2.676755	2.535175	2.532341	2.625900	2.527043	2.668233	2.492463	2.596253
8	2.771793	2.661813	2.675858	2.791797	2.661123	2.805842	2.648291	2.736403
9	2.845253	2.760715	2.781377	2.939306	2.821368	2.989680	2.861605	2.821327

	Index 32	Index 33	Index 34	Index 35	Index 36	Index 37	Index 38	Index 39
0	1.410490	1.496976	1.497604	1.548648	1.344081	1.498648	1.318837	1.675205
1	1.521932	1.620490	1.618705	1.716612	1.431969	1.615746	1.434727	1.778317
2	1.626530	1.734975	1.738916	1.901249	1.534868	1.732015	1.654805	1.880723
3	1.731639	1.875118	1.850099	2.098327	1.861894	1.907857	1.989009	2.005055
4	1.968962	2.019500	1.954454	2.259243	2.144282	2.132930	2.212404	2.158784
5	2.139679	2.159175	2.080074	2.392471	2.266628	2.301959	2.346713	2.298751
6	2.303630	2.395912	2.251609	2.514835	2.393296	2.478583	2.492167	2.465666
7	2.580907	2.530850	2.391418	2.632819	2.507033	2.652907	2.597559	2.564595
8	2.771240	2.686557	2.580009	2.750241	2.647338	2.789627	2.705434	2.690866
9	2.927644	2.823091	2.746789	2.908792	2.826234	2.972766	2.843161	2.778220

	Index 40	Index 41	Index 42	Index 43	Index 44	Index 45	Index 46	Index 47
0	1.499128	1.590174	1.580804	1.612437	1.379575	1.537946	1.352403	1.782826
1	1.619695	1.701042	1.706036	1.738151	1.467237	1.637079	1.493156	1.896399
2	1.743336	1.830435	1.818285	1.897713	1.565528	1.792667	1.795925	2.018924
3	1.923814	1.945976	1.930488	2.052482	1.717484	1.990007	1.955209	2.116451
4	2.078359	2.073133	2.033188	2.224500	2.180405	2.147552	2.117610	2.224694
5	2.175626	2.214197	2.128792	2.360012	2.370417	2.319058	2.222394	2.314370
6	2.333552	2.408617	2.289296	2.529775	2.468388	2.491308	2.363866	2.440855
7	2.592680	2.529177	2.522630	2.666938	2.578678	2.664483	2.487015	2.549471
8	2.771941	2.669160	2.733438	2.785499	2.690225	2.800071	2.619473	2.703904
9	2.930360	2.758893	2.862229	2.873858	2.801219	2.983526	2.791746	2.793187

	Index 48	Index 49	Index 50	Index 51	Index 52	Index 53	Index 54	Index 55
0	1.462356	1.542312	1.572729	1.612606	1.408316	1.419642	1.416001	1.713642
1	1.582093	1.661184	1.684571	1.748995	1.504221	1.676002	1.579243	1.829809
2	1.698693	1.768338	1.814717	1.879250	1.614691	1.828124	1.798086	1.972095
3	1.839592	1.988617	2.037981	2.087803	1.851406	2.001830	1.964124	2.101910
4	2.115125	2.182534	2.170610	2.247112	2.090841	2.175164	2.142002	2.249927

5	2.258347	2.367546	2.277703	2.374741	2.306850	2.311871	2.310059	2.374617
6	2.367754	2.499397	2.380880	2.513249	2.458403	2.486061	2.480664	2.505365
7	2.503302	2.613083	2.514330	2.647308	2.608851	2.641232	2.657703	2.588874
8	2.712896	2.749786	2.752390	2.790800	2.760267	2.781462	2.803466	2.697565
9	2.891681	2.918386	2.914643	2.966627	2.938607	2.950011	2.983992	2.779292

	Index 56	Index 57	Index 58	Index 59	Index 60	Index 61	Index 62	Index 63
0	1.510889	1.610844	1.613676	1.659698	1.441788	1.583592	1.462422	1.777288
1	1.645913	1.730171	1.744235	1.783857	1.578195	1.703838	1.680556	1.896192
2	1.749107	1.846842	1.827780	1.907966	1.775065	1.846961	1.861877	2.057441
3	1.869062	1.963170	1.914867	2.047190	1.959585	2.042743	2.070133	2.193904
4	2.118135	2.147824	2.053806	2.204611	2.193354	2.215421	2.195235	2.350973
5	2.338606	2.300856	2.274566	2.336131	2.414785	2.343812	2.310577	2.453632
6	2.509223	2.445590	2.448042	2.484290	2.562323	2.454145	2.448198	2.584777
7	2.609154	2.567426	2.607927	2.642418	2.649945	2.655015	2.582654	2.667735
8	2.713873	2.735504	2.768948	2.790190	2.752342	2.775734	2.748520	2.775744
9	2.809696	2.825475	2.938251	2.965532	2.875831	2.935680	2.933576	2.841777

	Index 64	Index 65	Index 66	Index 67	Index 68	Index 69	Index 70	Index 71
0	1.357870	1.448394	1.469562	1.533085	1.331409	1.484015	1.375478	1.666693
1	1.454452	1.579097	1.599319	1.722489	1.429910	1.607750	1.482713	1.809009
2	1.587661	1.769320	1.705353	1.910469	1.544795	1.812705	1.715440	1.963716
3	1.811261	1.933494	1.844161	2.042359	1.837732	1.934935	1.951019	2.074371
4	1.952621	2.085066	2.003099	2.183699	2.098223	2.040168	2.133706	2.187038
5	2.094329	2.226128	2.115109	2.315348	2.305199	2.270905	2.325950	2.273309
6	2.417777	2.423564	2.225772	2.514058	2.495020	2.448421	2.481276	2.393211
7	2.643182	2.564016	2.495688	2.647446	2.643354	2.607317	2.634671	2.505141
8	2.770287	2.706376	2.734623	2.789689	2.785187	2.765013	2.786650	2.696989
9	2.895622	2.807619	2.895800	2.922713	2.955662	2.937933	2.955688	2.806309

	Index 72	Index 73	Index 74	Index 75	Index 76	Index 77	Index 78	Index 79
0	1.450942	1.547545	1.525910	1.612792	1.399087	1.571707	1.443595	1.751300
1	1.549794	1.677685	1.658521	1.731382	1.486572	1.666066	1.622055	1.871815
2	1.660181	1.845462	1.782624	1.893552	1.581164	1.768857	1.770147	1.987493
3	1.901186	1.979906	1.899951	2.006327	1.694667	1.916230	1.936981	2.112618
4	2.043817	2.107840	1.980523	2.185159	1.917489	2.102799	2.102713	2.253821
5	2.168245	2.195570	2.116425	2.383359	2.278531	2.232566	2.203149	2.381506
6	2.410725	2.319509	2.412164	2.513238	2.501283	2.469060	2.281116	2.540686
7	2.623253	2.427789	2.645062	2.644377	2.609928	2.668239	2.399354	2.637668
8	2.778756	2.584597	2.815682	2.796690	2.711699	2.835617	2.703522	2.777820
9	2.936136	2.702883	2.953277	2.953608	2.824880	2.962790	2.875816	2.851258

	Index 80	Index 81	Index 82	Index 83	Index 84	Index 85	Index 86	Index 87
0	1.479072	1.552840	1.415581	1.580226	1.452271	1.514225	1.308815	1.613022
1	1.606373	1.648865	1.686014	1.778737	1.570116	1.686122	1.562955	1.770691
2	1.726469	1.856849	1.843482	1.947824	1.712788	1.821262	1.797576	1.954154
3	1.870282	1.967908	1.966079	2.073966	1.820911	1.930206	1.961939	2.081348
4	1.983189	2.123160	2.112598	2.224343	1.958627	2.099009	2.133811	2.226677
5	2.158482	2.241247	2.217483	2.361876	2.226187	2.349248	2.302436	2.352857
6	2.524453	2.430397	2.356514	2.508843	2.396475	2.485478	2.489334	2.496105
7	2.655629	2.541952	2.578144	2.658322	2.562912	2.641020	2.607956	2.607514
8	2.750747	2.751108	2.767331	2.835263	2.749189	2.774582	2.755323	2.737062
9	2.840104	2.854101	2.932109	2.980831	2.935121	2.932689	2.932100	2.831562

	Index 88	Index 89	Index 90	Index 91	Index 92	Index 93	Index 94	Index 95

0	1.465693	1.602409	1.564084	1.601933	1.476698	1.542015	1.443977	1.728725
1	1.607781	1.714294	1.708095	1.805765	1.597372	1.696784	1.640726	1.870761
2	1.754246	1.908305	1.843694	1.941308	1.752628	1.858988	1.874481	2.008670
3	1.988356	2.043956	1.966835	2.079811	2.027571	2.019352	1.994279	2.155644
4	2.140122	2.162268	2.079503	2.216134	2.239051	2.180537	2.135813	2.304878
5	2.286623	2.297284	2.204771	2.342375	2.368424	2.340133	2.289495	2.448690
6	2.566005	2.452130	2.507776	2.530645	2.489155	2.498200	2.502387	2.601198
7	2.693141	2.585247	2.650741	2.659362	2.581492	2.681442	2.614072	2.736549
8	2.787288	2.754661	2.764870	2.778235	2.698312	2.805106	2.726760	2.881752
9	2.868753	2.837101	2.849416	2.933954	2.812171	2.994272	2.819519	2.987287

	Index 96	Index 97	Index 98	Index 99	Index 100	Index 101	Index 102	Index 103
0	1.470293	1.530221	1.557400	1.579188	1.396142	1.490351	1.379860	1.676203
1	1.568200	1.662214	1.664573	1.674500	1.496402	1.589207	1.530401	1.799764
2	1.660499	1.788131	1.783159	1.941925	1.638070	1.831394	1.829957	1.922379
3	1.763906	1.901292	1.928409	2.059247	1.976890	2.012264	2.093293	2.071527
4	2.029094	2.062712	2.074149	2.230596	2.265651	2.176899	2.233050	2.211847
5	2.249241	2.269420	2.224693	2.368802	2.390107	2.341002	2.336246	2.366992
6	2.457410	2.431008	2.354593	2.506311	2.533197	2.482582	2.472848	2.492715
7	2.638685	2.559786	2.488120	2.767454	2.664599	2.616642	2.587815	2.614276
8	2.766328	2.725177	2.682380	2.835411	2.855458	2.766931	2.716529	2.771517
9	2.920428	2.812824	2.907153	2.933950	2.978893	2.938847	2.875861	2.873055

	Index 104	Index 105	Index 106	Index 107	Index 108	Index 109	Index 110	Index 111
0	1.514509	1.653452	1.565927	1.606138	1.456346	1.599915	1.424321	1.856753
1	1.626651	1.779739	1.703956	1.765367	1.555355	1.708263	1.538017	1.969487
2	1.735324	1.894463	1.831551	1.903776	1.660214	1.802933	1.798138	2.083688
3	1.974561	2.008818	1.971881	2.053337	1.841526	1.998269	1.996810	2.193000
4	2.124776	2.109630	2.087746	2.231615	2.183763	2.151301	2.135622	2.313043
5	2.257236	2.217339	2.221607	2.412457	2.363897	2.317358	2.248109	2.395057
6	2.436796	2.378645	2.316242	2.559087	2.500644	2.493763	2.403062	2.506191
7	2.582521	2.570007	2.589733	2.687011	2.625911	2.633504	2.561491	2.598891
8	2.757079	2.760655	2.875205	2.818243	2.731175	2.839197	2.739856	2.740717
9	2.911357	2.911625	2.991431	2.969807	2.837698	2.964154	2.916943	2.811717

	Index 112	Index 113	Index 114	Index 115	Index 116	Index 117	Index 118	Index 119
0	1.468259	1.556878	1.548931	1.610412	1.458115	1.518578	1.434862	1.649545
1	1.588480	1.688728	1.687537	1.745651	1.555070	1.681780	1.645192	1.789912
2	1.708203	1.857780	1.871868	1.966507	1.700525	1.861053	1.792316	1.948307
3	1.845139	2.015306	2.032162	2.102745	1.916643	2.044849	1.924006	2.116386
4	2.009516	2.224287	2.149721	2.241953	2.121542	2.221373	2.169238	2.304571
5	2.245882	2.374192	2.254041	2.390627	2.332628	2.397103	2.329177	2.451353
6	2.515994	2.525152	2.441248	2.519775	2.477738	2.529803	2.442642	2.574146
7	2.743675	2.618201	2.632918	2.642267	2.625546	2.644255	2.549993	2.658465
8	2.903559	2.737973	2.775075	2.776215	2.783859	2.790194	2.708124	2.776133
9	2.999232	2.824428	2.949080	2.951524	2.946509	2.955395	2.915711	2.857489

	Index 120	Index 121	Index 122	Index 123	Index 124	Index 125	Index 126	Index 127
0	1.548951	1.656279	1.618675	1.666186	1.508472	1.603063	1.520520	1.855436
1	1.677756	1.762518	1.784262	1.802942	1.650143	1.733728	1.676574	1.991133
2	1.792912	1.858796	1.882353	1.945360	1.820188	1.888038	1.944173	2.128338
3	1.909813	1.991387	1.982847	2.093433	2.038167	2.055178	2.089325	2.257289
4	2.129025	2.172344	2.132060	2.239970	2.254559	2.197612	2.212930	2.393685
5	2.332338	2.363621	2.319606	2.377509	2.435484	2.346968	2.327814	2.504846
6	2.573645	2.521647	2.504214	2.522455	2.610264	2.516396	2.456738	2.647598

7	2.713824	2.638838	2.692991	2.675565	2.724514	2.690151	2.556790	2.750498
8	2.854495	2.760451	2.867752	2.807986	2.861255	2.814657	2.684592	2.862716
9	2.959323	2.874022	2.993799	2.957510	2.973614	2.988422	2.810189	2.956086

Table : static float nec_lspnw_2a[80] = {

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	-0.002659	-0.010449	0.002314	-0.003515	0.003596	-0.004081	0.004905	0.007516
1	-0.000603	-0.021996	-0.001918	-0.003924	0.006542	-0.016139	-0.000042	0.018020
2	-0.011054	-0.019496	-0.001182	0.008184	-0.005843	0.007465	-0.014782	0.001024
3	-0.053560	-0.002745	-0.007593	-0.011969	-0.010872	0.019308	0.012902	-0.010344
4	0.004356	0.014130	-0.046695	-0.003942	0.027108	-0.012140	-0.000364	-0.009212

	Index 8	Index 9	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	0.001389	-0.005125	0.003220	-0.002157	-0.000530	-0.002607	0.011443	0.006477
1	0.014058	-0.001804	0.010860	0.008105	0.001568	-0.006179	0.014062	0.056035
2	0.023611	0.007820	0.015689	0.052758	0.008793	0.010796	0.010518	0.016776
3	-0.008804	0.015497	0.018755	0.021683	0.017361	0.064091	0.021662	0.015425
4	0.014951	0.019553	-0.018032	0.012280	0.059845	0.008465	0.018930	0.004049

Table : static float nec_lspnw_2b[320] = {

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	-0.033504	-0.007403	-0.033931	-0.048630	-0.016720	0.001696	-0.029686	-0.016742
1	-0.002646	-0.023979	-0.048531	-0.018207	-0.010328	-0.090213	-0.013321	0.016787
2	-0.053048	-0.034425	0.012792	-0.047866	-0.041435	0.003807	-0.005240	-0.031405
3	0.016497	-0.014611	-0.031201	-0.086772	-0.042881	0.003068	0.002178	-0.000467
4	-0.003916	-0.009096	0.011326	-0.057778	0.034768	-0.019568	-0.020594	-0.024901

	Index 8	Index 9	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	-0.025611	0.060360	0.013839	-0.074847	0.003595	0.032265	-0.038956	-0.028414
1	-0.003107	0.034749	-0.048798	0.010160	-0.002898	-0.013563	-0.044606	0.025880
2	-0.010901	-0.022983	-0.011019	-0.009814	-0.007945	-0.019767	0.029020	0.009441
3	-0.024675	-0.033606	-0.036745	-0.003666	-0.080325	-0.035975	0.020109	-0.012231
4	0.007489	-0.029362	0.014007	-0.001483	0.002400	-0.021239	-0.028805	-0.051449

	Index 16	Index 17	Index 18	Index 19	Index 20	Index 21	Index 22	Index 23
0	-0.021993	0.015655	-0.003012	0.002615	0.022593	0.008436	-0.024701	0.009707
1	-0.038979	-0.036214	-0.024732	0.016688	-0.013756	-0.004025	0.014572	0.012289
2	-0.015435	-0.057630	0.008808	-0.015698	-0.037205	-0.034196	-0.010509	-0.056176
3	0.017340	0.015956	-0.026680	-0.018963	-0.010251	-0.000604	0.036374	-0.027742
4	0.028707	-0.005347	-0.030294	0.026031	0.037070	-0.041277	-0.047000	-0.005175

	Index 24	Index 25	Index 26	Index 27	Index 28	Index 29	Index 30	Index 31
0	0.011753	0.016939	0.005086	-0.028685	-0.004661	0.084359	0.000773	-0.018826
1	0.016117	0.014957	-0.023311	0.019490	0.012096	-0.018593	-0.014519	0.050456
2	-0.058710	-0.012343	-0.002230	0.013530	0.008622	-0.029916	0.054612	-0.028526
3	0.028448	-0.022324	0.001763	0.000306	-0.000464	0.008432	0.001075	-0.025524
4	-0.013763	-0.089358	0.000610	0.029726	-0.007118	-0.005161	-0.030187	0.001372

	Index 32	Index 33	Index 34	Index 35	Index 36	Index 37	Index 38	Index 39
0	-0.009278	0.022733	-0.012229	-0.003422	-0.014080	0.029532	-0.000854	0.015146
1	-0.026647	0.005058	-0.018926	0.007330	0.018228	-0.032799	-0.007660	0.040396
2	-0.024853	-0.013884	0.043690	-0.012857	-0.032706	0.001133	0.006553	-0.014145

3	0.027889	-0.003565	0.001612	-0.032371	0.017288	0.002770	0.007295	0.006965
4	-0.030383	-0.009165	0.020894	-0.026426	0.048844	-0.042303	-0.050094	-0.042397

	Index 40	Index 41	Index 42	Index 43	Index 44	Index 45	Index 46	Index 47
0	-0.003673	0.008936	0.020774	-0.018891	-0.001351	0.036739	0.011049	-0.028170
1	-0.000651	0.025942	-0.052477	0.039817	-0.010604	0.001184	-0.033597	0.005865
2	-0.030292	0.015073	0.017407	-0.011646	0.011639	0.008212	0.019126	0.035907
3	0.012630	-0.047396	0.008994	0.029151	-0.025346	-0.022319	0.042268	-0.032893
4	0.012834	0.005823	0.045344	0.004883	0.040601	0.028809	-0.009920	-0.011273

	Index 48	Index 49	Index 50	Index 51	Index 52	Index 53	Index 54	Index 55
0	-0.002293	0.031076	0.022627	-0.007396	0.007433	0.026795	0.009006	0.026997
1	-0.007822	-0.010314	-0.023412	0.035659	0.002438	-0.004084	0.022039	0.034531
2	-0.035517	-0.021780	0.036940	0.006090	0.004846	-0.018305	0.028555	-0.024268
3	0.071746	0.033927	-0.033199	-0.034083	0.016636	0.038740	0.028163	0.008807
4	0.003507	0.018280	0.001096	0.083643	0.038744	-0.050626	-0.040156	0.019066

	Index 56	Index 57	Index 58	Index 59	Index 60	Index 61	Index 62	Index 63
0	0.012542	0.020332	0.024141	-0.017309	0.014640	0.049478	-0.024212	-0.013070
1	0.015442	0.014202	-0.005213	-0.002848	0.032915	0.016163	0.014736	0.062326
2	-0.004862	0.022628	0.024098	0.001721	0.024167	0.012076	0.043806	0.029802
3	0.033568	-0.023441	0.012921	0.035671	0.000250	0.013223	0.029711	-0.005611
4	-0.006423	-0.039718	0.004252	0.010347	0.015459	-0.022791	-0.005715	-0.013107

Table : static float nec_lspnw_2b[320] = {

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	-0.033504	-0.007403	-0.033931	-0.048630	-0.016720	0.001696	-0.029686	-0.016742
1	-0.002646	-0.023979	-0.048531	-0.018207	-0.010328	-0.090213	-0.013321	0.016787
2	-0.053048	-0.034425	0.012792	-0.047866	-0.041435	0.003807	-0.005240	-0.031405
3	0.016497	-0.014611	-0.031201	-0.086772	-0.042881	0.003068	0.002178	-0.000467
4	-0.003916	-0.009096	0.011326	-0.057778	0.034768	-0.019568	-0.020594	-0.024901

	Index 8	Index 9	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	-0.025611	0.060360	0.013839	-0.074847	0.003595	0.032265	-0.038956	-0.028414
1	-0.003107	0.034749	-0.048798	0.010160	-0.002898	-0.013563	-0.044606	0.025880
2	-0.010901	-0.022983	-0.011019	-0.009814	-0.007945	-0.019767	0.029020	0.009441
3	-0.024675	-0.033606	-0.036745	-0.003666	-0.080325	-0.035975	0.020109	-0.012231
4	0.007489	-0.029362	0.014007	-0.001483	0.002400	-0.021239	-0.028805	-0.051449

	Index 16	Index 17	Index 18	Index 19	Index 20	Index 21	Index 22	Index 23
0	-0.021993	0.015655	-0.003012	0.002615	0.022593	0.008436	-0.024701	0.009707
1	-0.038979	-0.036214	-0.024732	0.016688	-0.013756	-0.004025	0.014572	0.012289
2	-0.015435	-0.057630	0.008808	-0.015698	-0.037205	-0.034196	-0.010509	-0.056176
3	0.017340	0.015956	-0.026680	-0.018963	-0.010251	-0.000604	0.036374	-0.027742
4	0.028707	-0.005347	-0.030294	0.026031	0.037070	-0.041277	-0.047000	-0.005175

	Index 24	Index 25	Index 26	Index 27	Index 28	Index 29	Index 30	Index 31
0	0.011753	0.016939	0.005086	-0.028685	-0.004661	0.084359	0.000773	-0.018826
1	0.016117	0.014957	-0.023311	0.019490	0.012096	-0.018593	-0.014519	0.050456
2	-0.058710	-0.012343	-0.002230	0.013530	0.008622	-0.029916	0.054612	-0.028526
3	0.028448	-0.022324	0.001763	0.000306	-0.000464	0.008432	0.001075	-0.025524
4	-0.013763	-0.089358	0.000610	0.029726	-0.007118	-0.005161	-0.030187	0.001372

	Index 32	Index 33	Index 34	Index 35	Index 36	Index 37	Index 38	Index 39
0	-0.009278	0.022733	-0.012229	-0.003422	-0.014080	0.029532	-0.000854	0.015146
1	-0.026647	0.005058	-0.018926	0.007330	0.018228	-0.032799	-0.007660	0.040396
2	-0.024853	-0.013884	0.043690	-0.012857	-0.032706	0.001133	0.006553	-0.014145
3	0.027889	-0.003565	0.001612	-0.032371	0.017288	0.002770	0.007295	0.006965
4	-0.030383	-0.009165	0.020894	-0.026426	0.048844	-0.042303	-0.050094	-0.042397

	Index 40	Index 41	Index 42	Index 43	Index 44	Index 45	Index 46	Index 47
0	-0.003673	0.008936	0.020774	-0.018891	-0.001351	0.036739	0.011049	-0.028170
1	-0.000651	0.025942	-0.052477	0.039817	-0.010604	0.001184	-0.033597	0.005865
2	-0.030292	0.015073	0.017407	-0.011646	0.011639	0.008212	0.019126	0.035907
3	0.012630	-0.047396	0.008994	0.029151	-0.025346	-0.022319	0.042268	-0.032893
4	0.012834	0.005823	0.045344	0.004883	0.040601	0.028809	-0.009920	-0.011273

	Index 48	Index 49	Index 50	Index 51	Index 52	Index 53	Index 54	Index 55
0	-0.002293	0.031076	0.022627	-0.007396	0.007433	0.026795	0.009006	0.026997
1	-0.007822	-0.010314	-0.023412	0.035659	0.002438	-0.004084	0.022039	0.034531
2	-0.035517	-0.021780	0.036940	0.006090	0.004846	-0.018305	0.028555	-0.024268
3	0.071746	0.033927	-0.033199	-0.034083	0.016636	0.038740	0.028163	0.008807
4	0.003507	0.018280	0.001096	0.083643	0.038744	-0.050626	-0.040156	0.019066

	Index 56	Index 57	Index 58	Index 59	Index 60	Index 61	Index 62	Index 63
0	0.012542	0.020332	0.024141	-0.017309	0.014640	0.049478	-0.024212	-0.013070
1	0.015442	0.014202	-0.005213	-0.002848	0.032915	0.016163	0.014736	0.062326
2	-0.004862	0.022628	0.024098	0.001721	0.024167	0.012076	0.043806	0.029802
3	0.033568	-0.023441	0.012921	0.035671	0.000250	0.013223	0.029711	-0.005611
4	-0.006423	-0.039718	0.004252	0.010347	0.015459	-0.022791	-0.005715	-0.013107

Table : static float nec_lspnw_2c[640] = {

	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	0.011870	-0.084478	-0.090545	-0.027227	-0.012635	-0.050640	-0.073422	-0.118368
1	-0.045158	0.038443	0.042818	-0.038228	-0.037158	-0.003237	0.040869	-0.029286
2	-0.006750	0.023507	-0.054640	0.011451	-0.094501	-0.049441	0.003971	0.021383
3	0.003793	-0.052411	-0.030262	-0.075596	-0.005673	-0.002791	-0.000404	-0.016118
4	-0.075747	-0.089093	0.022036	-0.013216	0.042470	-0.013734	-0.028855	-0.033977

	Index 8	Index 9	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	-0.049780	0.088395	-0.048218	-0.049330	0.025234	0.025423	-0.018884	0.003204
1	-0.018581	0.055945	0.028428	-0.013308	-0.014236	0.017180	0.022588	0.001463
2	-0.005643	0.006581	-0.003724	0.049949	-0.044489	0.000005	-0.053875	-0.047535
3	-0.023432	-0.060148	0.002915	0.017828	-0.064469	-0.034891	0.020416	0.075186
4	0.000605	-0.067554	0.028462	-0.074385	-0.075233	-0.051266	-0.056506	-0.013193

	Index 16	Index 17	Index 18	Index 19	Index 20	Index 21	Index 22	Index 23
0	0.003965	-0.042267	-0.016687	0.043395	0.015845	0.032195	-0.030399	-0.050890
1	-0.020884	0.059944	0.044400	-0.033013	-0.022674	-0.046485	-0.100824	-0.088461
2	-0.047478	-0.010397	-0.004051	-0.019436	-0.077688	-0.008477	0.055386	-0.074178
3	0.009514	-0.066512	0.027841	-0.041132	-0.126253	0.017961	0.031291	0.076881
4	0.005020	0.020043	-0.003030	-0.014608	0.003297	-0.015391	-0.030860	0.016841

	Index 24	Index 25	Index 26	Index 27	Index 28	Index 29	Index 30	Index 31
0	-0.028912	-0.045577	-0.015171	0.007189	0.048829	0.023935	-0.141105	0.023561
1	-0.032620	0.017991	0.002419	-0.057116	0.022110	0.042378	-0.082339	-0.002687
2	-0.021298	0.041317	0.010508	0.021506	-0.009494	0.013252	0.049861	-0.009131

3	0.030114	-0.030709	-0.043224	-0.026913	0.014691	0.001865	0.053075	0.034834
4	0.017201	0.010512	0.038336	-0.011581	-0.087140	-0.034038	0.032271	0.008829

	Index 32	Index 33	Index 34	Index 35	Index 36	Index 37	Index 38	Index 39
0	0.011755	-0.006623	-0.037728	0.016742	0.007567	0.032813	-0.089770	-0.074888
1	-0.001495	0.073523	0.051119	0.038422	-0.075853	0.006237	0.057939	0.030640
2	0.021022	0.045919	-0.003310	0.026747	-0.028315	-0.064453	0.086775	-0.025924
3	0.019802	-0.015711	0.021762	-0.024090	0.002567	-0.030942	0.022197	0.057815
4	-0.040151	-0.091066	0.085987	0.005193	0.030378	0.009247	-0.036164	0.001718

	Index 40	Index 41	Index 42	Index 43	Index 44	Index 45	Index 46	Index 47
0	0.001544	0.037447	-0.032776	-0.020675	0.022614	0.056444	-0.024414	0.017662
1	-0.028851	-0.005377	-0.015445	-0.039685	0.015372	0.047451	0.028896	-0.028869
2	-0.006938	0.063091	0.023970	0.023701	-0.009605	0.006230	0.047491	-0.073181
3	-0.010687	0.014799	0.019284	0.015825	-0.069686	0.026023	0.016802	0.081643
4	0.023365	-0.080369	0.044521	-0.025348	0.004729	0.035685	-0.009770	0.099413

	Index 48	Index 49	Index 50	Index 51	Index 52	Index 53	Index 54	Index 55
0	0.072437	0.027548	-0.073064	0.064668	0.093864	0.045206	-0.005907	-0.028508
1	-0.009548	0.060579	0.035048	-0.010172	0.037220	0.034689	-0.005116	-0.025188
2	-0.003885	0.025630	0.062712	-0.020959	-0.010340	-0.021885	0.076381	0.007063
3	0.064659	-0.053754	0.041363	0.001278	-0.047751	0.044412	-0.012561	0.104515
4	-0.001369	0.053941	0.040210	0.011509	0.052526	-0.027539	0.038346	0.052778

	Index 56	Index 57	Index 58	Index 59	Index 60	Index 61	Index 62	Index 63
0	0.012988	0.003796	0.020406	0.058180	0.084508	0.014194	-0.021204	0.047869
1	0.024059	-0.013417	0.000864	0.007892	0.069956	0.069453	-0.085190	0.012925
2	0.016658	0.094399	0.032198	0.048974	0.037961	0.087888	0.114632	0.004127
3	0.015414	0.038408	-0.009083	0.009048	-0.001418	0.051556	0.036650	0.083438
4	0.010999	-0.024523	0.090119	0.017611	-0.017637	0.023422	0.061965	0.068765

	Index 64	Index 65	Index 66	Index 67	Index 68	Index 69	Index 70	Index 71
0	-0.045683	-0.053697	-0.089128	0.018641	-0.088598	-0.010915	-0.149642	-0.065828
1	-0.085526	0.003416	-0.021981	-0.029538	-0.069223	-0.046379	0.064762	-0.049199
2	0.004994	-0.024643	-0.027172	0.030777	-0.071688	-0.082928	0.023752	-0.014639
3	-0.028651	-0.064376	0.019063	-0.047538	-0.068150	-0.032108	-0.010462	0.035663
4	-0.079437	-0.048991	0.039277	-0.077285	0.029343	-0.058130	0.011363	-0.031626

	Index 72	Index 73	Index 74	Index 75	Index 76	Index 77	Index 78	Index 79
0	-0.024572	0.031667	-0.081171	-0.040429	-0.089363	-0.001758	-0.036358	-0.009949
1	-0.058097	0.019597	-0.001682	-0.037388	-0.106207	-0.014692	0.003547	-0.051867
2	-0.025928	-0.008663	0.014374	0.073001	-0.153685	-0.020951	-0.002647	-0.058178
3	-0.012447	-0.082281	-0.046199	-0.025578	-0.341236	-0.024431	-0.004736	0.045797
4	-0.020079	-0.147118	0.068732	-0.034993	-0.404666	-0.028690	-0.065684	-0.041762

	Index 80	Index 81	Index 82	Index 83	Index 84	Index 85	Index 86	Index 87
0	0.048061	-0.014770	-0.013904	-0.005521	-0.002981	-0.016368	-0.044808	-0.003414
1	-0.032214	0.019997	0.038906	-0.035658	0.008720	-0.010678	-0.067530	-0.062736
2	-0.085330	-0.048611	-0.050829	-0.036347	-0.039159	-0.020032	0.029397	0.019607
3	0.021355	-0.049833	0.012936	-0.056277	-0.088974	0.026722	-0.018525	0.049841
4	-0.014394	-0.008381	0.017634	0.017340	0.078583	-0.026386	0.022813	0.012583

	Index 88	Index 89	Index 90	Index 91	Index 92	Index 93	Index 94	Index 95
0	-0.060832	-0.021939	0.017812	0.042723	0.033987	-0.008642	-0.055081	0.005635
1	-0.009822	0.029892	-0.049536	-0.045546	-0.009719	0.055524	-0.040826	0.003009

2	0.025647	-0.004176	0.028763	0.055762	-0.042488	-0.023918	0.078332	-0.015117
3	0.018639	-0.025693	-0.057661	0.002956	0.009944	-0.017982	0.069417	0.025148
4	-0.001871	-0.011161	0.050998	-0.023001	-0.054890	-0.051506	-0.002261	0.050819

	Index 96	Index 97	Index 98	Index 99	Index 100	Index 101	Index 102	Index 103
0	-0.018009	-0.010572	0.041393	0.042101	0.034991	0.016236	-0.043497	-0.004799
1	-0.001528	0.066789	0.030085	-0.002100	-0.038535	0.020810	0.100788	0.052824
2	0.027427	0.029733	-0.032087	0.040609	0.021964	-0.030887	0.034797	0.002091
3	-0.025118	-0.053742	-0.009509	-0.051521	0.024158	0.003850	0.000486	0.068683
4	-0.031731	-0.034100	0.099634	0.000819	0.041285	-0.011981	0.008527	0.028597

	Index 104	Index 105	Index 106	Index 107	Index 108	Index 109	Index 110	Index 111
0	-0.030748	0.075701	-0.024186	0.019243	0.071144	0.010556	-0.027835	-0.037765
1	-0.004750	-0.006322	-0.049789	-0.036259	0.029401	0.032673	0.027789	0.008902
2	-0.039180	0.012100	-0.008788	0.013833	-0.049523	-0.021658	0.008738	-0.056499
3	-0.023226	0.002331	-0.000177	0.062120	-0.097383	-0.022488	0.050872	0.055520
4	0.046120	-0.040373	0.085640	-0.045884	-0.010375	0.034628	-0.047625	0.051036

	Index 112	Index 113	Index 114	Index 115	Index 116	Index 117	Index 118	Index 119
0	0.023602	0.015006	-0.007378	0.029433	0.043790	-0.019119	0.002928	0.009152
1	-0.043797	0.078158	0.044261	-0.002158	-0.026609	-0.002242	-0.017290	-0.000219
2	-0.044263	-0.013826	0.032727	0.006971	-0.045521	0.012388	0.037577	0.046678
3	0.078458	-0.006582	0.001940	-0.008343	-0.028692	0.058869	0.003128	0.045762
4	0.027959	0.012308	0.047146	-0.009422	0.061917	0.002993	0.008711	0.032853

	Index 120	Index 121	Index 122	Index 123	Index 124	Index 125	Index 126	Index 127
0	-0.017636	0.015214	0.035100	0.036558	0.058665	0.008057	0.030153	0.055335
1	0.000330	0.024382	0.002435	0.015462	0.033029	0.073266	-0.034968	0.016659
2	-0.001222	0.066477	0.006218	0.034399	-0.021149	0.044741	0.067764	-0.056888
3	0.001939	-0.009713	-0.018622	0.051726	-0.022892	0.036171	0.095289	0.037401
4	0.002198	-0.032520	0.040115	-0.016121	-0.012223	-0.038869	0.012272	0.034105

Table : lspnw_2d

D	Index 0	Index 1	Index 2	Index 3	Index 4	Index 5	Index 6	Index 7
0	-0.028197	0.000930	0.022233	-0.046727	-0.060254	0.029011	-0.015856	-0.035875
1	-0.079027	-0.027795	-0.031420	-0.026040	-0.008205	0.018150	0.045136	0.041996
2	-0.025293	-0.058723	0.029915	-0.002975	-0.002686	-0.015994	-0.018243	0.035203
3	0.013172	-0.047380	0.012786	-0.035450	0.012922	-0.050681	-0.020294	-0.010491
4	-0.005029	0.007307	-0.029342	-0.044432	0.021629	-0.052981	0.029154	-0.042390

	Index 8	Index 8	Index 10	Index 11	Index 12	Index 13	Index 14	Index 15
0	0.011235	0.066596	0.000667	0.003106	-0.015685	0.051470	0.029040	-0.021045
1	-0.028613	0.002672	0.005602	-0.011314	-0.040158	0.057261	0.021209	0.037991
2	-0.048071	-0.025582	-0.023930	0.022814	0.038862	0.030756	0.013971	0.057650
3	0.037469	-0.011182	0.021427	-0.031734	0.042564	-0.004045	0.041401	0.037516
4	0.038835	0.015289	-0.018836	0.024236	0.042878	-0.016226	0.038627	0.005631

Table : lspnw_p

	Index 0	Index 1	Index 2	Index 3	Index 12
0	0.985931	0.183860	0.116774	0.501652	0.000000
1	0.886075	0.175261	0.110151	0.499353	0.000000
2	0.987123	0.133840	0.077885	0.496165	0.000000
3	0.901710	0.157511	0.080712	0.507355	0.000000

4	0.840150	0.204697	0.124973	0.503295	0.000000
5	0.861737	0.255349	0.151524	0.513664	0.000000
6	0.833506	0.261359	0.146579	0.509292	0.000000
7	0.752088	0.254515	0.149699	0.516079	0.000000
8	0.826712	0.306935	0.178027	0.513368	0.000000
9	0.710357	0.289605	0.173726	0.526145	0.000000
10	0.624831	0.252788	0.122511		
11	0.651755	0.238015	0.110066		
12	0.675479	0.220147	0.104483		
13	0.682763	0.212398	0.104716		
14	0.662548	0.219429	0.118611		
15	0.638192	0.227064	0.135110		
16	0.632092	0.227060	0.141408		
17	0.634684	0.220997	0.145237		
18	0.640460	0.209553	0.150183		
19	0.702685	0.187497	0.110224		

Table : Gain VQ Tables for Bandwidth Extension Tool (Mode 0)

Number	Codebook1	Codebook2	Number	Codebook1	Codebook2
0	0.00348	0.00963	64	0.01081	0.02806
1	0.01892	1.28207	65	0.15108	1.31328
2	0.01692	0.67434	66	0.08509	0.77328
3	0.52628	1.66628	67	0.60279	1.89198
4	0.00634	0.44261	68	0.07878	0.48393
5	0.97237	1.03064	69	1.09696	1.25316
6	0.05331	1.03717	70	0.03159	1.11130
7	0.18090	4.19072	71	0.60998	4.73539
8	0.03263	0.23893	72	0.08326	0.27217
9	0.21012	1.47372	73	0.40772	1.50714
10	0.41566	0.41362	74	0.38486	0.51600
11	0.14668	2.79586	75	0.13927	3.31814
12	0.24806	0.31121	76	0.22815	0.39926
13	0.84419	1.76801	77	0.87652	1.98615
14	0.47141	0.96679	78	0.57811	0.97422
15	0.43523	6.96842	79	1.81157	7.79287
16	0.04055	0.07034	80	0.08305	0.10351
17	0.04088	1.98784	81	0.20598	1.97492
18	0.29112	0.70800	82	0.36796	0.76119
19	0.58774	2.56796	83	0.86515	2.61224
20	0.13673	0.45739	84	0.20151	0.45840
21	0.81152	1.51485	85	0.97158	1.57979
22	0.27737	0.97519	86	0.29631	1.06547
23	0.17988	5.39050	87	0.79250	5.68511
24	0.04334	0.33329	88	0.08011	0.39620
25	0.45337	1.26292	89	0.51164	1.38624
26	0.47011	0.72108	90	0.54674	0.81433
27	1.12315	3.38061	91	1.59159	3.34907
28	0.24598	0.50289	92	0.30245	0.52992
29	1.54475	1.96165	93	1.73535	2.37277
30	0.71273	0.92717	94	0.73462	1.08809
31	0.56207	13.59443	95	0.42414	20.02453
32	0.03974	0.03516	96	0.04851	0.04056
33	0.07668	1.52720	97	0.04666	1.62375
34	0.15600	0.64413	98	0.20772	0.75467

35	0.45982	2.08513	99	0.71346	2.20442
36	0.00312	0.55498	100	0.10121	0.57292
37	1.28027	1.48861	101	1.64524	1.31897
38	0.10107	0.92712	102	0.20486	0.91115
39	1.15050	4.27138	103	1.32909	4.87617
40	0.13796	0.25774	104	0.19353	0.21307
41	0.30279	1.63643	105	0.36514	1.82242
42	0.47996	0.52927	106	0.60045	0.52629
43	0.57178	3.55842	107	1.00497	3.87176
44	0.30693	0.38036	108	0.31557	0.45106
45	0.95927	2.32213	109	1.25555	2.46424
46	0.40143	1.10484	110	0.51226	1.14405
47	0.23791	8.80790	111	2.02153	10.46220
48	0.02979	0.15416	112	0.09660	0.19113
49	0.17877	2.29212	113	0.44385	2.36221
50	0.29791	0.83936	114	0.39153	0.89015
51	0.62585	3.01050	115	1.03300	2.93484
52	0.17197	0.53527	116	0.23185	0.60602
53	1.14551	1.76206	117	1.21210	2.03693
54	0.20748	1.15091	118	0.32310	1.29150
55	1.77299	5.46216	119	1.57083	6.34064
56	0.11584	0.34377	120	0.15697	0.39800
57	0.63617	1.46344	121	0.69285	1.64603
58	0.65451	0.68404	122	0.86496	0.78387
59	1.76288	3.82096	123	1.81776	4.41429
60	0.31188	0.61037	124	0.40233	0.62406
61	1.41710	2.83647	125	1.88117	2.96659
62	0.66244	1.24713	126	0.85757	1.29501
63	0.00000	31.04177	127	0.17897	0.32810

Table : Gain VQ Tables for Bandwidth Extension Tool (Mode 1)

Number	Codebook1	Codebook2	Number	Codebook1	Codebook2
0	0.00879	0.01590	64	0.03544	0.04537
1	0.18644	0.41815	65	0.20879	0.45665
2	0.00805	0.45867	66	0.00253	0.52176
3	0.45084	0.43158	67	0.49515	0.49559
4	0.07985	0.39725	68	0.09708	0.46106
5	0.27304	0.53523	69	0.27679	0.56171
6	0.13457	0.55812	70	0.14383	0.58844
7	0.02613	0.86296	71	0.11044	0.94678
8	0.05259	0.28307	72	0.04481	0.35051
9	0.31837	0.44487	73	0.34423	0.48382
10	0.03794	0.55921	74	0.04054	0.60462
11	0.58581	0.43668	75	0.62620	0.52489
12	0.18986	0.48958	76	0.20913	0.50630
13	0.38912	0.63529	77	0.42206	0.63948
14	0.27645	0.58807	78	0.27915	0.62399
15	0.47030	1.45389	79	0.75311	1.54461
16	0.03516	0.13749	80	0.04202	0.21035
17	0.28138	0.48279	81	0.29311	0.51628
18	0.05470	0.47877	82	0.06817	0.51674
19	0.46257	0.58429	83	0.45450	0.61974
20	0.10817	0.49808	84	0.13305	0.51783
21	0.33611	0.55667	85	0.36677	0.55677

22	0.16668	0.62121	86	0.18733	0.65445
23	0.25307	0.88138	87	0.33784	0.97926
24	0.21978	0.26709	88	0.30074	0.22891
25	0.36850	0.50966	89	0.39714	0.52555
26	0.04053	0.67271	90	0.06824	0.74860
27	0.83442	0.56952	91	0.79539	0.75453
28	0.21315	0.53995	92	0.24124	0.55241
29	0.47626	0.66787	93	0.49483	0.73048
30	0.23242	0.73464	94	0.28952	0.75658
31	0.23580	2.30352	95	0.65153	2.87362
32	0.02292	0.08155	96	0.09718	0.09480
33	0.25113	0.41744	97	0.25648	0.46169
34	0.05528	0.58327	98	0.03961	0.63562
35	0.47609	0.54652	99	0.52553	0.55283
36	0.13037	0.43694	100	0.15927	0.47211
37	0.30516	0.56006	101	0.31893	0.59277
38	0.17935	0.56749	102	0.20233	0.59769
39	0.07065	1.19272	103	0.02526	1.54242
40	0.11574	0.30828	104	0.15555	0.35111
41	0.37379	0.45366	105	0.41835	0.48969
42	0.09361	0.60633	106	0.12387	0.63525
43	0.57690	0.58563	107	0.64384	0.67067
44	0.23487	0.49091	108	0.25268	0.51463
45	0.36420	0.66915	109	0.40827	0.71198
46	0.31446	0.63006	110	0.30773	0.67427
47	1.11157	1.36159	111	1.42616	1.86969
48	0.09240	0.17461	112	0.14633	0.21525
49	0.31029	0.49629	113	0.32649	0.52652
50	0.08730	0.53364	114	0.09936	0.55965
51	0.50415	0.60744	115	0.54968	0.65739
52	0.16195	0.52052	116	0.18389	0.53660
53	0.35211	0.60578	117	0.39099	0.59205
54	0.22528	0.63027	118	0.25189	0.67311
55	0.26797	1.14028	119	0.49731	1.14498
56	0.27488	0.35475	120	0.41860	0.33307
57	0.40305	0.55737	121	0.43028	0.56357
58	0.13299	0.69358	122	0.16504	0.78502
59	0.93303	1.03281	123	1.31055	0.82177
60	0.22983	0.57952	124	0.24769	0.60588
61	0.58645	0.78959	125	0.63314	0.96394
62	0.36413	0.75337	126	0.42109	0.85819
63	0.70529	4.60989	127	1.04826	10.61058

Table : Gain VQ Tables for Bandwidth Extension Tool (Mode 2)

Number	Codebook1	Codebook2	Number	Codebook1	Codebook2
0	0.00440	0.00408	64	0.00870	0.01579
1	0.46055	0.16051	65	0.50324	0.19423
2	0.27837	0.14331	66	0.29761	0.19509
3	0.02249	0.77025	67	0.04488	0.83809
4	0.02835	0.41864	68	0.02893	0.50145
5	0.73180	0.16042	69	0.78153	0.20207
6	0.51048	0.25609	70	0.53809	0.29170
7	0.25451	2.58452	71	0.48568	2.27219
8	0.06145	0.20921	72	0.05411	0.26465

9	0.60138	0.16309	73	0.62590	0.21466
10	0.35133	0.15344	74	0.39919	0.19011
11	0.12266	1.18911	75	0.25823	1.31526
12	0.31788	0.42037	76	0.34494	0.50483
13	0.84016	0.18801	77	0.90081	0.21860
14	0.57859	0.52727	78	0.62724	0.65350
15	0.04531	6.43672	79	1.37165	7.35902
16	0.05220	0.05585	80	0.02168	0.10615
17	0.58039	0.24874	81	0.58298	0.31431
18	0.29159	0.26657	82	0.29681	0.33418
19	0.06020	0.93373	83	0.01396	1.10543
20	0.20478	0.38328	84	0.25784	0.41430
21	0.78815	0.37291	85	0.85792	0.39609
22	0.60283	0.39581	86	0.66213	0.41587
23	0.60927	3.32770	87	0.85575	3.84592
24	0.20653	0.15754	88	0.25233	0.21060
25	0.67093	0.25017	89	0.72673	0.23449
26	0.40105	0.26245	90	0.42524	0.29830
27	0.55065	1.26649	91	0.62474	1.47852
28	0.40665	0.43623	92	0.46457	0.43074
29	0.97989	0.24016	93	1.08995	0.28779
30	0.74190	0.84797	94	0.95905	0.87020
31	0.86906	15.74998	95	0.45891	14.20312
32	0.01856	0.03115	96	0.04198	0.02709
33	0.54763	0.15685	97	0.55049	0.21499
34	0.33028	0.22737	98	0.36185	0.24699
35	0.07270	0.83463	99	0.16360	0.79672
36	0.11992	0.27425	100	0.11972	0.36194
37	0.76278	0.28371	101	0.81615	0.28555
38	0.54424	0.36840	102	0.52756	0.45555
39	1.08649	2.59193	103	1.56142	2.96032
40	0.15390	0.20841	104	0.17679	0.29173
41	0.65623	0.15631	105	0.68469	0.19110
42	0.44745	0.22472	106	0.47223	0.27434
43	0.03457	1.41777	107	0.19230	1.64408
44	0.27178	0.60113	108	0.36154	0.69657
45	0.88305	0.30156	109	0.97783	0.37918
46	0.77810	0.67220	110	0.54511	0.91355
47	2.60590	8.76116	111	1.08845	11.55844
48	0.07762	0.11173	112	0.12457	0.15063
49	0.61893	0.27861	113	0.65149	0.32626
50	0.35332	0.31656	114	0.39238	0.36393
51	0.29700	0.95423	115	0.30860	1.05041
52	0.10317	0.55695	116	0.18688	0.50805
53	0.89636	0.51887	117	1.03021	0.56291
54	0.69532	0.52186	118	0.78020	0.48666
55	0.24591	4.47499	119	1.29356	4.67622
56	0.20907	0.23379	120	0.23965	0.29379
57	0.70494	0.31198	121	0.72510	0.38918
58	0.45047	0.34556	122	0.49676	0.35281
59	1.13126	1.54884	123	0.82064	1.84339
60	0.42744	0.54153	124	0.48575	0.61715
61	1.16905	0.43169	125	1.42070	0.59306
62	0.84229	1.07881	126	1.17047	1.22804
63	2.78187	21.53200	127	0.00000	25.90823

Table : Gain VQ Tables for Bandwidth Extension Tool (Mode 3)

Number	Codebook1	Codebook2	Number	Codebook1	Codebook2
0	0.00274	0.00491	64	0.03857	0.03627
1	0.59029	0.11475	65	0.62482	0.11660
2	0.39098	0.10462	66	0.44421	0.07682
3	0.53919	0.25213	67	0.57210	0.23031
4	0.30774	0.18876	68	0.32721	0.23590
5	0.77014	0.16179	69	0.80881	0.13401
6	0.42642	0.13112	70	0.46706	0.13243
7	1.24530	0.13587	71	1.36384	0.04562
8	0.19256	0.12523	72	0.21688	0.20345
9	0.66442	0.11215	73	0.68318	0.14333
10	0.41080	0.28024	74	0.43790	0.32980
11	0.69438	0.21688	75	0.71993	0.24903
12	0.41622	0.53186	76	0.49298	0.62366
13	0.95899	0.10935	77	0.99830	0.15468
14	0.50288	0.10156	78	0.50746	0.14457
15	1.52073	0.08616	79	1.54728	0.09743
16	0.07212	0.30465	80	0.13647	0.49364
17	0.66231	0.18153	81	0.65490	0.22232
18	0.36163	0.19890	82	0.36778	0.25863
19	0.59761	0.34060	83	0.64642	0.32436
20	0.28071	0.44999	84	0.26240	0.69740
21	0.80227	0.19733	85	0.84338	0.22065
22	0.49583	0.20570	86	0.52529	0.21037
23	1.46261	0.05700	87	1.48840	0.05334
24	0.28992	0.08709	88	0.31750	0.11980
25	0.71343	0.11899	89	0.74760	0.13061
26	0.48797	0.33286	90	0.52486	0.30721
27	0.69686	0.30555	91	0.70724	0.37346
28	0.69133	0.71794	92	0.87925	0.74453
29	0.89016	0.38042	93	0.94584	0.33286
30	0.58475	0.16284	94	0.59430	0.19750
31	1.06856	0.22696	95	1.04861	0.34999
32	0.01601	0.13149	96	0.14220	0.18608
33	0.61316	0.16014	97	0.64541	0.14961
34	0.38322	0.15264	98	0.41131	0.17485
35	0.56615	0.28839	99	0.59986	0.27126
36	0.29901	0.30633	100	0.34954	0.33191
37	0.85547	0.15763	101	0.91057	0.18045
38	0.44612	0.17754	102	0.47798	0.17772
39	1.39026	0.09287	103	1.35369	0.17811
40	0.24908	0.12776	104	0.27780	0.16272
41	0.69140	0.17775	105	0.71995	0.17108
42	0.46443	0.25921	106	0.50061	0.26270
43	0.74778	0.28722	107	0.77470	0.34700
44	0.51990	1.04773	108	0.51029	2.74230
45	0.89699	0.25704	109	0.97340	0.24447
46	0.54858	0.11358	110	0.56116	0.14775
47	1.62475	0.13134	111	1.61234	0.20116
48	0.20739	0.31126	112	0.26271	0.24242
49	0.63456	0.26597	113	0.67270	0.26292
50	0.40248	0.21943	114	0.43973	0.22385

51	0.63681	0.41445	115	0.61784	0.54886
52	0.38277	0.41387	116	0.47323	0.41233
53	0.80382	0.26853	117	0.84161	0.31590
54	0.53240	0.16511	118	0.55665	0.19409
55	1.45397	0.09240	119	1.47392	0.10570
56	0.35232	0.09446	120	0.34618	0.15168
57	0.73984	0.19668	121	0.76520	0.22652
58	0.54128	0.36367	122	0.55567	0.44847
59	0.73071	0.47715	123	0.80975	0.42009
60	1.09633	0.89699	124	1.57856	1.34128
61	0.88011	0.50114	125	1.01092	0.54900
62	0.61197	0.22580	126	0.63141	0.19242
63	1.17984	0.35386	127	1.36317	0.44651

Table : Gain VQ Tables for Narrowband CELP coder (Subframe 10ms)

	Mode 0		Mode 1		Mode 2		Mode 3	
	Codebook0	Codebook1	Codebook0	Codebook1	Codebook0	Codebook1	Codebook0	Codebook1
0	0.005881	0.017177	0.022466	0.057102	0.079688	0.178891	0.184477	0.388664
1	0.051549	0.533273	0.357407	0.317504	0.456277	0.251318	0.584891	0.151331
2	0.053167	0.377189	0.060183	0.583301	0.352755	0.266917	0.478874	0.158363
3	0.275015	0.579468	0.397295	0.658454	0.740383	0.226809	0.723288	0.158275
4	0.043298	0.286508	0.058996	0.447717	0.360442	0.733670	0.090312	0.654322
5	0.043115	0.668181	0.541573	0.378444	0.592588	0.237913	0.737609	0.360216
6	0.236856	0.382637	0.078269	0.946046	0.532155	0.372427	0.551917	0.257699
7	0.078060	0.977704	0.639081	1.050880	0.903070	0.593177	0.881007	0.183082
8	0.043699	0.202235	0.068693	0.312209	0.067115	0.589188	0.376106	0.185163
9	0.187090	0.577519	0.415612	0.499703	0.604716	0.394715	0.705759	0.262625
10	0.183433	0.510241	0.256033	0.602592	0.209677	0.532716	0.620392	0.265869
11	0.458465	0.602462	0.531796	0.590934	0.975410	0.319835	0.769850	0.225823
12	0.141081	0.261530	0.228447	0.436173	0.165129	1.091088	0.462320	0.359931
13	0.189618	0.787439	0.713488	0.392423	0.730762	0.355920	0.909416	0.319394
14	0.357891	0.545607	0.170705	1.368295	0.567574	0.538101	0.596586	0.517510
15	0.358153	1.164616	1.194512	0.598419	1.434373	0.534854	1.054718	0.396448
16	0.034013	0.136923	0.047136	0.208997	0.117661	0.420041	0.370494	0.330826
17	0.122752	0.558669	0.455115	0.350034	0.581124	0.324221	0.661432	0.249395
18	0.111368	0.441815	0.056437	0.783321	0.477827	0.341790	0.589130	0.218269
19	0.339222	0.680765	0.487412	0.785388	0.851843	0.254370	0.791364	0.168017
20	0.119123	0.370595	0.170347	0.576834	0.580118	0.720696	0.300030	0.735996
21	0.036277	0.751632	0.614069	0.435337	0.663547	0.360372	0.790239	0.385964
22	0.331181	0.398336	0.295751	0.947681	0.488682	0.514020	0.554979	0.362241
23	0.192724	1.392354	0.882293	0.943388	0.847544	0.889443	0.957998	0.251681
24	0.128256	0.180015	0.223156	0.241829	0.064698	0.802579	0.487065	0.257020
25	0.180871	0.679978	0.523343	0.495731	0.684646	0.427136	0.750226	0.283853
26	0.272783	0.469740	0.265084	0.760263	0.355902	0.477879	0.629019	0.349052
27	0.633530	0.885928	0.738745	0.612964	1.042971	0.496651	0.839384	0.297477
28	0.219986	0.300950	0.336470	0.516798	0.517920	1.159787	0.462789	0.571118
29	0.318610	0.815375	0.886048	0.544325	0.826682	0.480912	0.912442	0.499930
30	0.447855	0.488218	0.470872	1.342377	0.718544	0.521057	0.707700	0.569683
31	0.576533	1.591783	1.240887	1.475744	1.380054	0.887236	1.411868	0.540164
32	0.030060	0.076283	0.063302	0.126188	0.226243	0.250470	0.283021	0.225221
33	0.046363	0.603817	0.363640	0.424047	0.528699	0.272362	0.659175	0.156611
34	0.046661	0.459195	0.068109	0.677246	0.402806	0.361285	0.535469	0.194294
35	0.287376	0.633129	0.486490	0.670924	0.776928	0.292101	0.730526	0.214962

36	0.100046	0.311195	0.144573	0.478384	0.442816	0.884532	0.226341	0.553060
37	0.106794	0.703929	0.633596	0.323280	0.662072	0.241227	0.734780	0.442619
38	0.217095	0.450487	0.140802	1.136513	0.543633	0.445335	0.588320	0.304898
39	0.123103	1.144733	0.873660	1.239329	0.997853	0.746097	0.877594	0.253533
40	0.087054	0.230896	0.159270	0.344097	0.131961	0.670111	0.430418	0.238479
41	0.221976	0.623994	0.445974	0.569840	0.628221	0.474113	0.711265	0.312573
42	0.230407	0.530564	0.305109	0.664261	0.279325	0.620678	0.663645	0.306593
43	0.540835	0.710978	0.615998	0.667044	1.145286	0.378113	0.799178	0.265587
44	0.168827	0.327264	0.271504	0.496650	0.231646	1.565139	0.516158	0.443815
45	0.248039	0.942748	0.867251	0.375487	0.804897	0.366971	0.945782	0.390643
46	0.371793	0.609076	0.207035	1.771307	0.642578	0.595366	0.659776	0.445829
47	0.598548	1.183792	1.593611	0.834527	1.715553	0.821322	1.205263	0.390728
48	0.080738	0.133669	0.130995	0.210709	0.284827	0.400523	0.376182	0.477732
49	0.133360	0.622769	0.469689	0.439131	0.635841	0.305813	0.686147	0.209723
50	0.140873	0.481242	0.166854	0.821475	0.456001	0.434753	0.632825	0.204985
51	0.420663	0.729597	0.596600	0.844791	0.874539	0.351406	0.827042	0.218840
52	0.172146	0.402853	0.175561	0.687668	0.677540	0.836865	0.448088	1.228660
53	0.076444	0.831108	0.711473	0.492610	0.706113	0.297791	0.825886	0.464916
54	0.343169	0.476061	0.448186	1.001364	0.488053	0.631429	0.595110	0.411168
55	0.181029	1.767716	1.199950	0.947754	1.090179	1.042897	1.053918	0.262708
56	0.203595	0.220365	0.266470	0.349868	0.210173	0.799233	0.522556	0.311966
57	0.262243	0.711410	0.610995	0.534468	0.753482	0.432903	0.789749	0.324193
58	0.293729	0.529141	0.371977	0.773541	0.395957	0.572426	0.683984	0.372609
59	0.943925	0.989379	0.754844	0.774747	1.199178	0.625543	0.856786	0.372851
60	0.302678	0.314689	0.356060	0.581748	0.812357	1.452663	0.588012	0.729024
61	0.436071	0.894314	0.982964	0.737669	0.924617	0.445253	1.039155	0.596346
62	0.611943	0.564882	0.792314	1.742021	0.770208	0.646973	0.844092	0.661950
63	1.170664	1.592143	1.696629	1.473382	1.573745	1.435942	1.390967	0.961560

Table : Gain VQ Tables for Narrowband CELP coder (Subframe 5ms)

	Mode 0		Mode 1		Mode 2		Mode 3	
	Codebook0	Codebook1	Codebook0	Codebook1	Codebook0	Codebook1	Codebook0	Codebook1
0	0.023886	0.065233	0.058640	0.144226	0.076819	0.184885	0.101809	0.163384
1	0.343509	0.489696	0.493511	0.620373	0.453452	0.228699	0.550692	0.119836
2	0.050595	0.492885	0.070558	0.598393	0.081093	0.632270	0.453740	0.142782
3	0.252670	0.732489	0.754514	0.269927	0.759191	0.171138	0.845792	0.143328
4	0.167035	0.465527	0.286783	0.249874	0.363425	0.183289	0.177225	0.318552
5	0.424463	0.615371	0.461872	0.273592	0.651659	0.189766	0.780940	0.136941
6	0.036913	0.721584	0.104837	0.958828	0.509826	0.545691	0.631832	0.292522
7	0.100144	1.445904	1.166639	0.366979	1.083018	0.277114	1.101504	0.204877
8	0.055866	0.362400	0.064481	0.420561	0.079809	0.398934	0.364607	0.147839
9	0.416963	0.712033	0.697082	0.648715	0.669876	0.281341	0.644164	0.122182
10	0.167091	0.606836	0.239347	0.578534	0.184504	1.080969	0.486874	0.393499
11	0.320156	0.995736	0.874790	0.454536	0.907812	0.217090	0.997673	0.169863
12	0.271640	0.302131	0.364516	0.587694	0.435631	0.338604	0.381963	0.644228
13	0.564469	0.713876	0.612629	0.429143	0.739913	0.390336	0.829978	0.233927
14	0.067606	0.885286	0.151674	1.498575	0.809212	0.486254	0.736480	0.290792
15	0.603998	1.093232	1.189677	0.789588	1.583176	0.564530	1.408914	0.309216
16	0.075480	0.256758	0.176964	0.212471	0.185598	0.153912	0.284690	0.157009
17	0.330207	0.596671	0.480750	0.803195	0.547458	0.200284	0.625060	0.207430
18	0.035404	0.619133	0.062634	0.775478	0.318381	0.818103	0.569957	0.212746
19	0.262439	0.825867	0.872641	0.253655	0.784817	0.244072	0.911861	0.153845
20	0.253000	0.471132	0.421979	0.418566	0.275956	0.316559	0.151986	0.606666

21	0.598848	0.597644	0.627190	0.264855	0.733715	0.300764	0.780614	0.212109
22	0.166293	0.720206	0.479662	1.016095	0.666698	0.756802	0.706205	0.199530
23	0.454068	1.411700	1.457550	0.428631	1.429907	0.365405	1.112699	0.355910
24	0.155152	0.328571	0.266513	0.407024	0.247027	0.645501	0.445673	0.281520
25	0.407925	0.798706	0.741644	1.036583	0.671251	0.455548	0.715471	0.129179
26	0.246561	0.601887	0.340549	0.777898	0.354257	1.558401	0.677850	0.379857
27	0.588556	0.904207	0.939849	0.636648	0.981044	0.340027	0.961614	0.292888
28	0.442543	0.420935	0.514832	0.455326	0.587567	0.348706	0.257724	1.280378
29	0.798787	0.767658	0.744996	0.434796	0.895923	0.349690	0.890921	0.264217
30	0.096611	1.064134	0.692875	1.443769	1.049886	0.601191	0.836453	0.396452
31	1.090193	1.320268	1.338360	1.127649	1.450660	0.949545	1.715301	0.349473
32	0.066269	0.158911	0.077379	0.276397	0.168840	0.277921	0.205490	0.144833
33	0.394843	0.543492	0.580333	0.671559	0.520639	0.303758	0.599046	0.153063
34	0.109207	0.551374	0.161379	0.678677	0.103967	0.801479	0.507320	0.181974
35	0.329629	0.752112	0.812736	0.345575	0.835125	0.192488	0.871109	0.201055
36	0.202987	0.535600	0.354685	0.330746	0.363198	0.280606	0.332981	0.376291
37	0.490814	0.647821	0.544259	0.352822	0.715432	0.225675	0.811338	0.180666
38	0.086594	0.779762	0.164592	1.164506	0.611071	0.574171	0.688953	0.284568
39	0.253871	1.754026	1.201532	0.566025	1.221386	0.325865	1.213684	0.271546
40	0.116008	0.415719	0.167912	0.485567	0.211284	0.468670	0.407798	0.214566
41	0.498020	0.774423	0.803750	0.762354	0.662844	0.360491	0.672176	0.167160
42	0.211682	0.664877	0.283091	0.679399	0.523940	1.098405	0.589870	0.432571
43	0.422948	1.101662	1.012645	0.489898	0.979987	0.234068	1.027083	0.255295
44	0.317134	0.401770	0.411112	0.682111	0.488715	0.424424	0.588279	0.804716
45	0.653219	0.773930	0.668442	0.520432	0.824365	0.392846	0.836023	0.298723
46	0.193193	0.923841	0.327951	1.766489	0.865355	0.642479	0.763474	0.353141
47	0.767912	1.277727	1.482601	0.792347	1.821306	0.529374	1.547436	0.536622
48	0.164619	0.223656	0.179983	0.343240	0.271282	0.201610	0.323479	0.239968
49	0.364039	0.662780	0.628169	0.839091	0.605607	0.262410	0.668879	0.228978
50	0.109902	0.661709	0.204912	0.813587	0.476532	0.717327	0.572632	0.302960
51	0.344330	0.866581	0.987783	0.333568	0.859493	0.274557	0.937517	0.218426
52	0.283408	0.542076	0.447681	0.526077	0.353851	0.421466	0.158247	0.857496
53	0.729263	0.528586	0.671709	0.353881	0.799137	0.317922	0.785773	0.271715
54	0.180596	0.798368	0.438575	1.265720	0.878357	0.972175	0.737145	0.234488
55	0.705184	1.707457	1.809571	0.689988	1.306232	0.548379	1.280966	0.489299
56	0.220877	0.392125	0.325823	0.491425	0.379379	0.569835	0.516294	0.262576
57	0.464595	0.913925	0.976659	1.219266	0.727304	0.549881	0.745287	0.173961
58	0.290736	0.665763	0.320952	0.933252	0.980668	1.543974	0.730342	0.501705
59	0.773946	0.977486	0.995429	0.884491	1.093275	0.422927	1.007476	0.395910
60	0.498277	0.528792	0.568232	0.537728	0.586390	0.440824	0.566853	1.531965
61	1.142292	0.903324	0.788392	0.549286	0.936493	0.467591	0.905749	0.344255
62	0.225944	1.177042	1.069528	1.706326	1.182395	0.801622	0.913155	0.593349
63	1.342490	1.689171	1.684790	1.482933	1.809565	1.133312	1.848412	0.638932

Table : Enhanced Gain Tables for Bandwidth Extension Tool (nec_egc)

	Mode 0	Mode 1	Mode 2	Mode 3
0	0.028072	0.203987	0.294636	0.552033
1	0.308173	0.197199	0.136863	0.106123
2	0.203134	0.090433	0.026536	0.053874
3	0.533764	0.327097	0.244737	0.253918
4	0.131561	0.008212	0.072444	0.218614
5	0.379095	0.251019	0.184093	0.167010
6	0.256279	0.149637	0.088474	0.041480
7	0.920816	0.524871	0.351534	0.469090

8	0.073191	0.061327	0.151955	0.350775
9	0.339503	0.222804	0.159734	0.135252
10	0.230383	0.122977	0.060760	0.000006
11	0.678233	0.388039	0.286317	0.329052
12	0.172881	0.051769	0.014683	0.127361
13	0.441157	0.282553	0.212241	0.204537
14	0.281695	0.173226	0.113130	0.074764
15	1.419244	0.916329	0.486387	0.696846

Table : Gain Tables for Excitation Codebook 1 of Bandwidth Extension Tool

	Mode 0	Mode 1	Mode 2	Mode 3
0	-2.41466	-0.08157	-0.27330	-1.50892
1	-0.90706	0.10468	-0.01335	-0.57114
2	-0.28410	0.24487	0.13601	-0.24324
3	0.04386	0.35212	0.27768	-0.06368
4	0.27431	0.43982	0.39665	0.05930
5	0.41922	0.51652	0.50259	0.17276
6	0.58054	0.58663	0.60302	0.28251
7	0.73560	0.65527	0.71173	0.40053
8	0.85908	0.73262	0.83204	0.53217
9	1.02850	0.81775	0.95545	0.68400
10	1.24813	0.91555	1.10872	0.82044
11	1.50969	1.04372	1.29315	0.98938
12	1.98968	1.22804	1.57483	1.22263
13	2.73335	1.58464	2.15740	1.63360
14	4.10306	2.28667	3.62861	2.54699
15	6.41216	6.73568	7.69762	5.41726

LSP table for the first stage (lsp_tbl[0][Index][])

Index					
0	0.07598	0.11583	0.17414	0.24063	0.29242
1	0.07173	0.12996	0.23673	0.33478	0.41112
2	0.04741	0.07080	0.13131	0.26953	0.40107
3	0.07831	0.15027	0.27770	0.37885	0.47900
4	0.05969	0.10877	0.19471	0.27769	0.36779
5	0.06486	0.09199	0.14439	0.29585	0.47720
6	0.05179	0.08288	0.16003	0.31263	0.39873
7	0.13309	0.20671	0.31115	0.38814	0.47545
8	0.05826	0.09023	0.15236	0.23677	0.35072
9	0.08321	0.13578	0.19905	0.28777	0.43009
10	0.06081	0.09133	0.13986	0.22547	0.42498
11	0.11176	0.17961	0.26738	0.34168	0.42726
12	0.09952	0.15653	0.22035	0.28657	0.34705
13	0.06992	0.09932	0.19952	0.37430	0.47837
14	0.02762	0.06221	0.20961	0.31313	0.41778
15	0.11720	0.20569	0.35140	0.44773	0.53333

LSP table for the first stage (lsp_tbl[1][Index][])

Index					
0	0.40731	0.58450	0.67794	0.79244	0.88222
1	0.51037	0.60160	0.69040	0.77783	0.85504
2	0.52064	0.59144	0.67402	0.75091	0.82005
3	0.57769	0.63867	0.70291	0.78317	0.83912
4	0.40140	0.52678	0.67764	0.77287	0.86054

5	0.51066	0.60945	0.70428	0.80726	0.88893
6	0.47101	0.55620	0.67867	0.77972	0.85860
7	0.54955	0.65929	0.73672	0.83096	0.87698
8	0.43273	0.59773	0.67489	0.74992	0.85408
9	0.51326	0.58573	0.65387	0.79552	0.87019
10	0.47985	0.56486	0.64538	0.73176	0.83672
11	0.52866	0.63432	0.70865	0.79899	0.85865
12	0.42655	0.52939	0.62236	0.74035	0.86132
13	0.47342	0.59800	0.68964	0.79949	0.88504
14	0.46200	0.55920	0.62868	0.79236	0.86986
15	0.60633	0.69637	0.75950	0.83586	0.87617

LSP table for the second stage of VQ without interframe prediction (d_tbl[0][Index][])

Index					
0	0.00477	-0.00616	0.00197	0.01659	0.00208
1	-0.01293	-0.00214	0.01018	0.03619	-0.00278
2	-0.00056	0.00503	0.04248	-0.00143	0.00298
3	0.00171	0.00945	0.01692	0.00673	0.00622
4	0.01186	0.01969	0.01880	-0.03597	0.01433
5	0.01126	0.01143	0.01759	0.03706	0.00045
6	0.00187	-0.00441	0.00028	-0.00029	0.02483
7	0.00727	0.01295	-0.00524	-0.00646	0.00963
8	0.01037	0.01090	-0.00421	0.03606	-0.02177
9	0.00709	0.01496	0.01230	0.00093	-0.01304
10	-0.01167	0.00567	0.03880	-0.01805	0.02714
11	0.02643	0.02713	-0.00252	-0.02043	-0.00449
12	0.00329	0.00665	-0.01693	0.01194	0.02110
13	0.02027	0.01446	0.02525	-0.00146	0.02122
14	-0.00091	0.00377	-0.01380	0.03608	0.00612
15	0.01515	0.03233	-0.00662	0.00744	-0.01400
16	0.01986	0.02892	-0.00033	0.00828	0.01991
17	-0.00646	-0.01405	0.01302	0.01717	0.01242
18	-0.02659	0.01111	0.02986	0.01570	0.01040
19	0.02568	0.00669	0.01904	-0.01169	-0.00715
20	-0.00779	0.02303	0.00127	-0.00662	0.03122
21	0.00477	0.00398	0.01337	0.02330	0.02342
22	0.00385	-0.01277	0.02118	0.00532	0.03282
23	0.02054	0.02103	-0.03378	0.00398	0.02801
24	0.02587	0.01514	-0.00222	0.03504	0.01814
25	0.00037	-0.00145	0.02360	0.01322	-0.01316
26	-0.01770	-0.01549	0.03910	0.00855	0.01901
27	0.00502	0.01816	0.03006	-0.01874	-0.00469
28	0.00131	0.00021	-0.01987	-0.00342	0.04966
29	0.03437	0.03039	0.01658	0.01857	-0.00438
30	-0.00342	-0.01514	-0.00513	0.01969	0.03160
31	0.02194	0.03330	0.01638	-0.01008	-0.02494
32	0.00298	0.01203	-0.00231	0.01390	-0.00387
33	0.00009	0.00016	0.02148	0.04612	-0.02311
34	-0.01150	-0.00366	0.05505	0.02114	-0.01785
35	0.00155	-0.00085	0.01697	-0.01007	0.01142
36	0.02049	0.02801	-0.00973	-0.02903	0.02599
37	0.01247	0.01536	0.03795	0.02223	0.00571
38	-0.01870	0.00430	0.00033	0.02004	0.02852
39	0.03556	0.00600	-0.01345	-0.00659	0.01318
40	0.02849	0.01914	-0.02411	0.02441	-0.00875
41	-0.00481	0.03266	0.02336	0.01152	-0.01895
42	-0.01815	-0.00331	0.01707	-0.00265	0.04277
43	0.00567	0.02878	0.00803	-0.01316	0.00104
44	0.00038	-0.00372	-0.02421	0.03706	0.03302
45	0.00440	0.01773	0.02820	0.01056	0.03648
46	-0.00140	-0.02063	-0.00159	0.04211	0.01029
47	0.03197	0.01451	-0.00250	0.00774	-0.02553
48	-0.00053	0.02982	0.00480	0.02684	0.01311
49	-0.00106	-0.01731	0.03039	0.03241	0.00170
50	-0.01794	0.03688	0.02812	-0.00218	0.01145
51	0.03027	-0.00779	0.01530	0.01762	-0.00190
52	0.00772	0.00414	0.00661	-0.02226	0.03793
53	0.00111	0.00042	0.01775	0.04644	0.02931
54	0.03141	-0.02314	-0.00690	0.01526	0.03342
55	0.03822	0.04648	-0.03499	-0.00967	0.00834
56	0.02343	-0.00441	-0.02337	0.02691	0.00577
57	0.01448	0.01050	0.02109	0.01853	-0.03287
58	-0.03326	-0.04106	0.05219	0.05212	0.03232
59	0.00970	0.01258	0.03967	-0.00355	-0.02565
60	0.01406	0.01372	0.00025	0.01253	0.04618
61	0.02889	0.04655	0.02928	0.00053	0.01530

62	-0.02341	-0.02510	0.00861	0.02897	0.04689
63	0.05407	0.06949	0.00980	-0.01152	-0.02534

LSP table for the second stage of VQ without interframe prediction (d_tbl[1][Index][])

Index					
0	0.00502	0.03415	-0.01517	-0.00311	0.01502
1	0.00080	0.00211	0.00117	0.00169	0.01083
2	0.01722	-0.01130	0.02393	0.01865	-0.01395
3	0.00412	-0.01634	0.00979	0.00169	0.02914
4	0.00767	0.02240	0.01015	-0.02235	0.00866
5	-0.00571	-0.01079	0.01526	0.02208	0.00762
6	0.01234	0.01213	-0.02434	0.00534	0.03182
7	-0.01354	0.01370	0.01556	0.00522	0.00253
8	-0.00451	0.02393	-0.00778	0.02266	-0.00030
9	-0.01308	0.01932	0.00246	-0.00712	0.02661
10	0.00515	0.01141	0.00863	0.00885	-0.01013
11	0.02019	-0.01093	0.00976	-0.00417	0.00945
12	0.02196	0.01562	-0.01708	0.01054	-0.00026
13	0.02591	0.01228	0.01471	0.03125	0.00681
14	0.01186	-0.00301	-0.00467	0.02626	-0.00433
15	0.01558	0.00808	0.03026	-0.01245	-0.00815
16	0.00724	0.03725	0.00627	-0.00209	-0.01251
17	0.01575	0.01078	0.01084	0.00366	0.01616
18	-0.00899	0.00499	0.03976	0.01701	-0.01461
19	-0.00579	-0.00318	0.02820	-0.00390	0.01446
20	0.03297	0.01205	-0.00924	-0.01740	0.01306
21	-0.02123	0.00581	0.01722	0.03848	-0.00085
22	0.01939	-0.01373	-0.01226	0.01493	0.02571
23	-0.03024	0.03724	0.01106	0.01023	0.00486
24	-0.00152	0.00589	-0.01737	0.04133	0.01634
25	-0.01738	0.00330	-0.00181	0.01982	0.02813
26	0.02341	0.00998	0.00255	0.00061	-0.01222
27	0.03923	-0.01013	0.01415	-0.00111	-0.00992
28	0.03998	-0.00871	-0.01859	0.01885	0.00401
29	0.00280	0.02945	0.02469	0.02073	0.01573
30	0.01612	-0.02934	0.00591	0.03507	0.00767
31	-0.01157	0.03409	0.04039	-0.01216	-0.00938

LSP table for the second stage of VQ with interframe prediction (pd_tbl[0][Index][]) =

Index					
0	0.00378	0.00210	0.01699	0.01285	-0.01660
1	0.00625	-0.00140	-0.02068	0.01521	0.02242
2	-0.00351	0.00287	0.00793	-0.00088	0.01681
3	-0.01230	0.00035	0.01622	0.01399	-0.00042
4	0.01180	0.01097	0.00364	0.00115	-0.01225
5	0.00089	-0.00939	0.00564	0.01846	-0.00012
6	0.00158	0.01305	0.00537	-0.00470	-0.00004
7	-0.00179	-0.01380	0.03205	0.01869	-0.00920
8	0.00265	0.00414	0.00864	0.01827	0.01109
9	0.01597	0.01515	-0.00596	-0.00611	-0.00017
10	0.00422	0.01094	-0.00470	-0.00651	0.01406
11	0.00249	-0.00080	0.00531	0.00313	0.00329
12	0.01100	0.01477	0.01383	0.00063	0.01176
13	-0.00424	-0.00028	-0.00535	0.01472	0.01213
14	0.00203	0.00478	0.01811	0.00132	-0.00491
15	-0.00563	0.00317	0.02432	-0.00438	0.00782
16	0.00269	0.00270	0.01390	0.03095	-0.02626
17	0.01015	-0.00048	-0.00674	0.00453	0.01215
18	-0.00443	-0.00458	0.02077	-0.00664	0.03037
19	0.02237	-0.00325	0.00015	0.01386	-0.00823
20	0.01046	0.01315	-0.00641	0.02367	-0.01448
21	-0.00513	-0.01318	-0.00490	0.02996	0.01842
22	0.00412	0.01065	0.00819	-0.02121	0.01865
23	-0.01489	-0.01333	0.04306	0.00597	0.01460
24	0.01468	0.01826	-0.00153	0.01723	0.00973
25	0.01408	0.02363	-0.02616	-0.00164	0.01047
26	-0.00118	-0.00286	-0.00263	-0.00701	0.03270
27	-0.01014	0.02133	-0.00252	0.01830	0.00411
28	0.02764	0.02552	0.00584	-0.00575	-0.02327
29	0.00586	0.00707	0.00693	0.01346	0.02904
30	0.00598	0.01556	0.02446	-0.00860	-0.01610
31	-0.01268	0.02646	0.00876	-0.00666	0.02082
32	0.00207	0.00794	0.00548	0.01306	-0.00552
33	0.00926	0.00146	-0.01891	0.03100	0.00718
34	0.00820	-0.00331	0.01136	-0.01065	0.01185
35	-0.00347	-0.01191	0.01657	0.00766	0.00990
36	0.00598	0.02145	0.00863	0.01094	-0.02167
37	0.00012	-0.00227	-0.00330	0.03092	-0.00516
38	0.00797	0.01197	0.01439	-0.01691	-0.00137
39	-0.00429	0.00931	0.04599	0.01189	-0.01405
40	0.00491	0.00352	0.02612	0.01710	0.01288
41	0.00683	0.03362	0.00002	-0.00560	-0.00263
42	0.00016	0.01025	-0.01434	0.00227	0.02664
43	0.00475	0.00825	-0.00966	0.00976	0.00094
44	0.02057	0.01880	0.01794	0.01391	-0.00163
45	-0.00086	-0.01035	0.00124	0.00766	0.02070
46	0.01852	-0.00267	0.02349	-0.00230	-0.00755
47	-0.00082	0.00985	0.03462	-0.01962	0.01051
48	0.00464	0.00622	0.01705	0.03154	-0.00446
49	0.03220	-0.00048	-0.01119	-0.00247	0.01826
50	-0.01661	-0.00526	0.01099	0.01173	0.02890
51	0.02070	-0.02200	0.00538	0.01390	0.01246
52	0.02447	0.01801	-0.01629	0.01158	-0.00990
53	-0.00755	-0.01452	0.01421	0.03541	0.00675
54	0.01918	0.02689	-0.00027	-0.02498	0.00668
55	-0.01769	-0.01716	0.05356	0.03959	0.02985
56	0.00929	0.00755	0.00612	0.03549	0.01795
57	0.02207	0.03489	-0.00887	0.00315	0.02895
58	0.00986	0.01098	-0.00561	-0.02000	0.03869
59	-0.01941	0.02375	0.02624	0.01548	0.00747
60	0.04063	0.04685	0.01714	0.00585	0.00081
61	-0.00144	-0.00515	-0.00743	0.01905	0.04445

62	-0.00968	0.04194	0.02330	0.00072	-0.01073
63	0.00943	0.02251	0.02587	0.00424	0.02988

LSP tabel for the second stage of VQ with interframe prediction (pd_tbl[1][Index][])

Index					
0	0.01800	-0.01362	0.01190	0.01769	-0.00530
1	0.01009	0.00396	0.00188	0.00308	-0.00505
2	-0.00335	0.00720	-0.01063	0.02097	0.00663
3	0.00509	0.01687	-0.01489	0.00124	0.01227
4	-0.00062	0.00151	0.01270	0.01213	-0.00643
5	-0.00135	0.01027	0.02565	-0.01057	0.00008
6	0.01561	0.01088	0.00632	-0.01711	0.00584
7	-0.00449	0.01785	0.00392	0.00530	-0.00357
8	0.01767	0.01506	0.01274	0.00136	-0.01476
9	0.00072	-0.00410	0.00663	-0.00468	0.02318
10	0.01582	-0.00335	-0.01080	0.01835	0.00015
11	-0.00778	0.02214	0.00448	-0.00927	0.01282
12	-0.01971	0.01722	0.00423	0.02336	0.00548
13	-0.00042	0.03487	0.02230	0.01536	0.00712
14	0.01646	0.00119	-0.00456	-0.00262	0.01050
15	0.01798	0.01689	0.00087	0.02354	0.00990
16	-0.00096	-0.00651	0.00857	0.03401	0.00004
17	0.00084	0.00178	0.00349	0.00140	0.00761
18	-0.00728	-0.00924	0.00455	0.01976	0.01794
19	-0.00789	0.01430	-0.00833	0.00882	0.02691
20	-0.00033	-0.01269	0.02588	0.00764	0.00440
21	0.01747	-0.00284	0.02385	-0.00730	-0.00602
22	0.01151	0.03042	-0.00120	-0.00982	-0.00075
23	0.01018	0.01812	-0.00693	0.01223	-0.00942
24	0.03501	-0.00183	0.00155	0.00068	-0.00532
25	0.01410	-0.01283	0.00828	-0.00023	0.00928
26	0.01538	-0.01414	-0.01287	0.01720	0.01999
27	-0.01613	0.00632	0.01511	0.00396	0.01159
28	-0.01125	0.00807	0.03190	0.01675	-0.00949
29	-0.00813	0.03342	0.02101	-0.00681	-0.01129
30	0.03025	0.01036	-0.01858	-0.00336	0.01336
31	0.01336	0.00910	0.01561	0.00797	0.01261

10. ANNEX C References

[1] B. Atal, J. Remde, “A new Model of LPC Excitation for Producing Natural Sounding Speech at low bit rates”, Proc. ICASSP, pp 614-617, 1982

[2] Peter Kabal, Ravi Prakash Ramachandran, “The Computation of Line Spectral Frequencies Using Chebyshev Polynimials”, IEEE Trans. On Acoutics, Speech and Signal Processing, vol. ASSP-34, no. 6, pp. 1419-1426, December 1986.

[3] J. D. Markel and A. H. Gray, Jr. *Linear Prediction of Speech.* Berlin Heidelberg New York: Springer Verlag, 1976.

[4] R. Veldhuis and M. Breeuwer, *An Introduction to Source Coding*, Prentice Hall, 1993.

11. ANNEX D CELP System Layer definition

In the CELP syntax, the header information is called once at the beginning. After that, subsequent frames are processed. Below, a pseudo code example of how this can be implemented into a system layer is provided.

```
CelpHeader()
while(1) {
  CelpFrame()
}
```

12. Annex E Patent Holders information

(informative)

Patent Holders

List of patent holders

The user’s attention is called to the possibility that - for some of the processes specified in this part of ISO/IEC 14496 - conformance with this part of ISO/IEC 14496 may require use of an invention covered by patent rights. By publication of this part of ISO/IEC 14496, no position is taken with respect to the validity of this claim or of any patent rights in connection therewith. However, each company listed in this annex has undertaken to file with the Information Technology Task Force (ITTF) a statement of willingness to grant a license under such rights that they hold on reasonable and non-discriminatory terms and conditions to applicants desiring to obtain such a license.
Information regarding such patents can be obtained from the following organizations.
The table summarizes the formal patent statements received and indicates the parts of the standard to which the statement applies. Three “N”s in the row corresponding to a company mean that the statement from the company did not mention whether part 1, part 2 or part 3 is the object of the statement. The list includes all organizations that have submitted informal statements. However, if no "X" is present, no formal statement has yet been received from that organization.

Company	ISO/IEC 14496 -2	ISO/IEC 14496 -3	ISO/IEC 14496 -1	ISO/IEC 14496 -7

Subpart 4 of FCD 14496-3 is split into several files:
r

w2203tfs	Syntax, semantics and decoder description
w2203tft	This file. T/F tool descriptions and normative Annex
w2203tfa	Informative annex (Transport streams, Encoder tools)
w2203tvq	Twin-VQ vector quantizer tables

3 T/F-TOOL DESCRIPTIONS 2

3 T/F-Tool Descriptions

3.1 Quantization

(identical to ISO/IEC 13818-7)

3.1.1 Tool description

For quantization of the spectral coefficients in the encoder a non uniform quantizer is used. Therefore the decoder must perform the inverse non uniform quantization after the Huffman decoding of the scalefactors (see clause 9 and 11) and spectral data (see clause 9).

3.1.2 Definitions

Help elements:

x_quant[g][win][sfb][bin]	quantized spectral coefficient for group *g*, window *win*, scalefactor band *sfb*, coefficient *bin*.
x_invquant[g][win][sfb][bin]	spectral coefficient for group *g*, window *win*, scalefactor band *sfb*, coefficient *bin* after inverse quantization.

3.1.3 Decoding process

The inverse quantization is described by the following formula:

$$x\_invquant = Sign(x\_quant) \cdot |x\_quant|^{\frac{4}{3}} \quad \forall \quad k$$

The maximum allowed absolute amplitude for *x_quant* is 8191. The inverse quantization is applied as follows:

```
for( g=0; g<num_window_groups; g++ ) {
   for( sfb=0; sfb < max_sfb; sfb++ ) {
      width = (swb_offset [g][sfb+1] - swb_offset [g][sfb] );
      for( win = 0; win < window_group_len[g]; win++ ) {;
         for( bin=0; bin<width; bin++ ) {
            x_invquant[g][win][sfb][bin] = sign(x_quant[g][win][sfb][bin]) *
                                 abs(x_quant[g][win][sfb][bin]) ^(4/3);
         }
      }
   }
}
```

3.2 Scalefactors

(identical to ISO/IEC 13818-7)

3.2.1 Tool description

The basic method to adjust the quantization noise in the frequency domain is the noise shaping using scalefactors. For this purpose the spectrum is divided in several groups of spectral coefficients called scalefactor bands which share one scalefactor (see clause 8.3.4). A scalefactor represents a gain value which is used to change the amplitude of all spectral coefficients in that scalefactor band. This mechanism is used to change the allocation of the quantization noise in the spectral domain generated by the non uniform quantizer.
For window_sequences which contain SHORT_WINDOWs grouping can be applied, i.e. a specified number of consecutive SHORT_WINDOWs may have only one set of scalefactors. Each scalefactor is then applied to a group of scalefactor bands corresponding in frequency (see clause 8.3.4).
In this tool the scalefactors are applied to the inverse quantized coefficients to reconstruct the spectral values.

3.2.2 Definitions

Bit stream elements:

global_gain	An 8-bit unsigned integer value representing the value of the first scalefactor. It is also the start value for the following differential coded scalefactors (see Table 6.12)
scale_factor_data()	Part of bit stream which contains the differential coded scalefactors (see Table 6.14)
hcod_sf[]	Huffman codeword from the Huffman code table used for coding of scalefactors, see Table 6.14 and clause 9.2

Help elements:

dpcm_sf[g][sfb]	Differential coded scalefactor of group g, scalefactor band sfb.
x_rescal[]	rescaled spectral coefficients

sf[g][sfb]	Array for scalefactors of each group
get_scale_factor_gain()	Function that returns the gain value corresponding to a scalefactor

3.2.3 Decoding process

3.2.3.1 Scalefactor bands

Scalefactors are used to shape the quantization noise in the spectral domain. For this purpose, the spectrum is divided into several scalefactor bands (see section 8.3.4). Each scalefactor band has a scalefactor, which represents a certain gain value which has to be applied to all spectral coefficients in this scalefactor band. In case of EIGHT_SHORT_SEQUENCE a scalefactor band may contain multiple scalefactor window bands of consecutive SHORT_WINDOWs (see clause 8.3.4 and 8.3.5).

3.2.3.2 Decoding of scalefactors

For all scalefactors the difference to the preceeding value is coded using the Huffman code book given in table A.1. See clause 9 for a detailed description of the Huffman decoding process. The start value is given explicitly as a 8 bit PCM in the bitstream element **global_gain**. A scalefactor is not transmitted for scalefactor bands which are coded with the Huffman codebook ZERO_HCB. If the Huffman codebook for a scalefactor band is coded with INTENSITY_HCB or INTENSITY_HCB2, the scalefactor is used for intensity stereo (see clause 9 and 12.2). In that case a normal scalefactor does not exist (but is initialized to zero to have an valid in the array).

The following pseudo code describes how to decode the scalefactors *sf[g][sfb]*:

```
last_sf = global_gain;
for( g=0; g < num_window_groups; g++ ) {
   for( sfb=0; sfb<max_sfb; sfb++ ) {
      if( sfb_cb[g][sfb] != ZERO_HCB && sfb_cb[g][sfb] != INTENSITY_HCB
          && sfb_cb[g][sfb] != INTENSITY_HCB2 ) {
         dpcm_sf = decode_huffman() - index_offset; /* see clause 4 */
         sf[g][sfb] = dpcm_sf + last_sf;
         last_sf = sf[g][sfb];
      }
      else {
         sf[g][sfb] = 0;
      }
   }
}
```

3.2.3.3 Applying scalefactors

The spectral coefficients of all scalefactor bands which correspond to a scalefactor have to be rescaled according to their scalefactor. In case of a window sequence that contains groups of short windows all coefficients in grouped scalefactor window bands have to be scaled using the same scalefactor.
In case of window_sequences with only one window, the scalefactor bands and their corresponding coefficients are in spectral ascending order. In case of EIGHT_SHORT_SEQUENCE and grouping the spectral coefficients of grouped short windows are interleaved by scalefactor window bands. See clause 8.3.5 for more detailed information.
The rescaling operation is done according to the following pseudo code:

```
for( g=0; g<num_window_groups; g++ ) {
   for( sfb=0; sfb < max_sfb; sfb++ ) {
      width = (swb_offset [sfb+1] - swb_offset [sfb] );
      for( win = 0; win < window_group_len[g]; win++ ) {;
         gain = get_scale_factor_gain( sf[g][sfb] );
         for( k=0; k<width; k++ ) {
            x_rescal[g][window][sfb][k] =
               x_invquant[g][window][sfb][k] * gain;
         }
      }
   }
}
```

The function *get_scale_factor_gain(sf[g][sfb])* returns the gain factor that corresponds to a scalefactor. The return value follows the equation:

$$gain = 2^{0.25 \cdot (sf[g][sfb] - SF\_OFFSET)}$$

The constant SF_OFFSET must be set to 100.

The following pseudo code describes this operation:

```
get_scale_factor_gain( sf[g][sfb] )  {
   SF_OFFSET = 100;
   gain = 2^(0.25 * ( sf[g][sfb] - SF_OFFSET));
   return( gain );
}
```

3.3 Noiseless Coding

(similar to ISO/IEC 13818-7)

3.3.1 Tool description

Noiseless coding is used to further reduce the redundancy of the scalefactors and the quantized spectrum of each audio channel.

The global_gain is coded as an 8 bit unsigned integer. The first scalefactor associated with the quantized spectrum is differentially coded relative to the global_gain value and then Huffman coded using the scalefactor codebook. The remaining scalefactors are differentially coded relative to the previous scalefactor and then Huffman coded using the scalefactor codebook.

Noiseless coding of the quantized spectrum relies on two divisions of the spectral coefficients. The first is a division into scalefactor bands that contain a multiple of 4 quantized spectral coefficients. See clause 8.3.4 and 8.3.5.

The second division, which is dependent on the quantized spectral data, is a division by scalefactor bands to form sections. The significance of a section is that the quantized spectrum within the section is represented using a single Huffman codebook chosen from a set of 11 possible codebooks. The length of a section and its associated Huffman codebook must be transmitted as side information in addition to the section's Huffman coded spectrum. Note that the length of a section is given in scalefactor bands rather than scalefactor window bands (see clause 8.3.4). In order to maximize the match of the statistics of the quantized spectrum to that of the Huffman codebooks the number of sections is permitted to be as large as the number of scalefactor bands. The maximum size of a section is max_sfb scalefactor bands.

As indicated in Table 00.22, spectrum Huffman codebooks can represent signed or unsigned n-tuples of coefficients. For unsigned codebooks, sign bits for every non-zero coefficient in the n-tuple immediately follow the associated codeword.

The noiseless coding has two ways to represent large quantized spectra. One way is to send the escape flag from the escape (ESC) Huffman codebook, which signals that the bits immediately following that codeword plus optional sign bits are an escape sequence that encodes values larger than those represented by the ESC Huffman codebook. A second way is the pulse escape method, in which relatively large-amplitude coefficients can be replaced by coefficients with smaller amplitudes in order to enable the use of Huffman code tables with higher coding efficiency. This replacement is corrected by sending the position of the spectral coefficient and the differences in amplitude as side information. The frequency information is represented by the combination of the scalefactor band number to indicate a base frequency and an offset into that scalefactor band.

3.3.2 Definitions

sect_cb[g][i] spectrum Huffman codebook used for section i in group g (see 6.3, Table 6.13).

sect_len_incr used to compute the length of a section, measures number of scalefactor bands from start of section. The length of **sect_len_incr** is 3 bits if window_sequence is EIGHT_SHORT_SEQUENCE and 5 bits otherwise (see 6.3, Table 6.13).

global_gain global gain of the quantized spectrum, sent as unsigned integer value (see 6.3, Table 6.12).

hcod_sf[] Huffman codeword from the Huffman code table used for coding of scalefactors (see 6.3, Table 6.14).

hcod[sect_cb[g][i]][w][x][y][z] Huffman codeword from codebook **sect_cb[g][i]** that encodes the next 4-tuple (w, x, y, z) of spectral coefficients, where w, x, y, z are quantized spectral coefficients. Within an n-tuple, w, x, y, z are ordered as described in 8.3.5. so that x_quant[group][win][sfb][bin] = w, x_quant[group][win][sfb][bin+1] = x,

	x_quant[group][win][sfb][bin+2] = y and x_quant[group][win][sfb][bin+3] = z. N-tuples progress from low to high frequency within the current section (see 6.3, Table 6.16).
hcod[sect_cb[g][i]][y][z]	Huffman codeword from codebook **sect_cb[g][i]** that encodes the next 2-tuple (y, z) of spectral coefficients, where y, z are quantized spectral coefficients. Within an n-tuple, y, z are ordered as described in 8.3.5 so that x_quant[group][win][sfb][bin] = y and x_quant[group][win][sfb][bin+1] = z. N-tuples progress from low to high frequency within the current section (see 6.3, Table 6.16).
quad_sign_bits	sign bits for non-zero coefficients in the spectral 4-tuple. A '1' indicates a negative coefficient, a '0' a positive one. Bits associated with lower frequency coefficients are sent first (see 6.3, Table 6.16).
pair_sign_bits	sign bits for non-zero coefficients in the spectral 2-tuple. A '1' indicates a negative coefficient, a '0' a positive one. Bits associated with lower frequency coefficients are sent first (see 6.3, Table 6.16).
hcod_esc_y	escape sequence for quantized spectral coefficient y of 2-tuple (y,z) associated with the preceeding Huffman codeword (see 6.3, Table 6.16).
hcod_esc_z	escape sequence for quantized spectral coefficient z of 2-tuple (y,z) associated with the preceeding Huffman codeword (see 6.3, Table 6.16).
pulse_data_present	1 bit indicating whether the pulse escape is used (1) or not (0) (see 6.3, Table 6.17). Note that pulse_data_present must be 0 for an EIGHT_SHORT_SEQUENCE.
number_pulse	2 bits indicating how many pulse escapes are used. The number of pulse escapes is from 1 to 4 (see 6.3, Table 6.17).
pulse_start_sfb	6 bits indicating the index of the lowest scalefactor band where the pulse escape is achieved (see 6.3, Table 6.17).
pulse_offset[i]	5 bits indicating the offset (see 6.3, Table 6.17).
pulse_amp[i]	4 bits indicating the unsigned magnitude of the pulse (see 6.3, Table 6.17).
sect_start[g][i]	offset to first scalefactor band in section i of group g (see 6.3, Table 6.13).
sect_end[g][i]	offset to one higher than last scalefactor band in section i of group g (see 6.3, Table 6.13).
num_sec[g]	number of sections in group g (see 6.3, Table 6.13).
escape_flag	the value of 16 in the ESC Huffman codebook
escape_prefix	the bit sequence of N 1's
escape_separator	one 0 bit
escape_word	an N+4 bit unsigned integer word, msb first
escape_sequence	the sequence of *escape_prefix*, *escape_separator* and *escape_word*
escape_code	2^(N+4) + escape_word
x_quant[g][win][sfb][bin]	Huffman decoded value for group *g*, window *win*, scalefactor band *sfb*, coefficient *bin*
spec[w][k]	de-interleaved spectrum. *w* ranges from 0 to *num_windows*-1 and *k* ranges from 0 to *swb_offset[num_swb]*-1.

The noiseless coding tool requires these constants (see clause 6.3, spectral_data()).

ZERO_HCB	0
FIRST_PAIR_HCB	5
ESC_HCB	11
QUAD_LEN	4
PAIR_LEN	2
NOISE_HCB	13
INTENSITY_HCB2	14
INTENSITY_HCB	15
ESC_FLAG	16

3.3.3 Decoding Process

Four-tuples or 2-tuples of quantized spectral coefficients are Huffman coded and transmitted starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient. For the case of multiple windows per block (EIGHT_SHORT_SEQUENCE), the grouped and interleaved set of spectral coefficients is treated as a single set of coefficients that progress from low to high. The set of coefficients may need to be de-interleaved after they are decoded (see clause 8.3.5). Coefficients are stored in the array x_quant[g][win][sfb][bin], and the order of transmission of the Huffman codewords is such that when they are decoded in the order received and stored in the array, *bin* is the most rapidly incrementing index and *g* is the most slowly incrementing index. Within a codeword, for those associated with spectral four-tuples, the order of decoding is w, x, y, z; for codewords associated with spectral two-tuples, the order of decoding is y, z. The set of coefficients is divided into sections and the sectioning information is transmitted starting from the lowest frequency section and progressing to the highest frequency section. The spectral information for sections that are coded with the "zero" codebook is not sent as this spectral information is zero. Similarly, spectral information for sections coded with the "intensity" codebooks is not sent. The spectral information for all scalefactor bands at and above **max_sfb**, for which there is no section data, is zero.

There is a single differential scalefactor codebook which represents a range of values as shown in Table 00.11. The differential scalefactor codebook is shown in Table A.1. There are eleven Huffman codebooks for the spectral data, as shown in Table 00.22. The codebooks are shown in Tables A.2 through A.12. There are four other "codebooks" above and beyond the actual Huffman codebooks, specifically the "zero" codebook, indicating that neither scalefactors nor quantized data will be transmitted, and the "intensity" codebooks indicating that this individual channel is part of a channel pair, and that the data that would normally be scalefactors is instead steering data for intensity stereo. Similarly, the "noise substitution" codebook indicates that the spectral coefficients are derived from random numbers rather than quantized spectral values, and that the data that would normally be scalefactors is instead noise energy data. In these cases, no quantized spectral data are transmitted. Codebook index 12 is reserved.

The spectrum Huffman codebooks encode 2- or 4-tuples of signed or unsigned quantized spectral coefficients, as shown in Table 00.22. This table also indicates the largest absolute value (LAV) able to be encoded by each codebook and defines a boolean helper variable array, unsigned_cb[], that is 1 if the codebook is unsigned and 0 if signed.

The result of Huffman decoding each differential scalefactor codeword is the codeword index, listed in the first column of Table A.1. This is translated to the desired differential scalefactor by adding index_offset to the index. Index_offset has a value of –60, as shown in Table 9.1. Likewise, the result of Huffman decoding each spectrum n-tuple is the codeword index, listed in the first column of Tables A.2 through A.12. This index is translated to the n-tuple spectral values as specified in the following pseudo C-code:

`unsigned` = Boolean value unsigned_cb[i], listed in second column of Table 9.2.
`dim` = Dimension of codebook, listed in the third column of Table 9.2.
`lav` = LAV, listed in the fourth column of Table 9.2.
`idx` = codeword index

```
if (unsigned) {
   mod = lav + 1;
   off = 0;
}
else {
   mod = 2*lav + 1;
   off = lav;
}

if (dim == 4) {
   w = INT(idx/(mod*mod*mod)) - off;
   idx -= (w+off)*(mod*mod*mod)
   x = INT(idx/(mod*mod)) - off;
   idx -= (x+off)*(mod*mod)
   y = INT(idx/mod) - off;
   idx -= (y+off)*mod
   z = idx - off;
}
else {
```

```
   y = INT(idx/mod) - off;
   idx -= (y+off)*mod
   z = idx - off;
}
```

If the Huffman codebook represents signed values, the decoding of the quantized spectral n-tuple is complete after Huffman decoding and translation of codeword index to quantized spectral coefficients. If the codebook represents unsigned values then the sign bits associated with non-zero coefficients immediately follow the Huffman codeword, with a '1' indicating a negative coefficient and a '0' indicating a positive one. For example, if a Huffman codeword from codebook 7

hcod[7][y][z]

has been parsed, then immediately following this in the bitstream is

pair_sign_bits

which is a variable length field of 0 to 2 bits. It can be parsed directly from the bitstream as

```
   if (y != 0)
      if (one_sign_bit == 1)
         y = -y
   if (z != 0)
      if (one_sign_bit == 1)
         z = -z
```

where one_sign_bit is the next bit in the bitstream and **pair_sign_bits** is the concatenation of the one_sign_bit fields.

The ESC codebook is a special case. It represents values from 0 to 16 inclusive, but values from 0 to 15 encode actual data values, and the value16 is an *escape_flag* that signals the presence of **hcod_esc_y** or **hcod_esc_z**, either of which will be denoted as an *escape_sequence*. This *escape_sequence* permits quantized spectral elements of LAV>15 to be encoded. It consists of an *escape_prefix* of N 1's, followed by an *escape_separator* of one zero, followed by an *escape_word* of N+4 bits representing an unsigned integer value. The *escape_sequence* has a decoded value of 2^(N+4)+*escape_word*. The desired quantized spectral coefficient is then the sign indicated by the pair_sign_bits applied to the value of the *escape_sequence*. In other words, an *escape_sequence* of 00000 would decode as 16, an *escape_sequence* of 01111 as 31, an *escape_sequence* of 1000000 as 32, one of 1011111 as 63, and so on. Note that restrictions in clause 10.3 dictate that the length of the escape_sequence is always less than 24 bits. For escape Huffman codewords the ordering of bitstream elements is Huffman codeword followed by 0 to 2 sign bits followed by 0 to 2 escape sequences.

When **pulse_data_present** is 1 (the pulse escape is used), one or several quantized coefficients have been replaced by coefficients with smaller amplitudes in the encoder. The number of coefficients replaced is indicated by **number_pulse**. In reconstructing the quantized spectral coefficients *x_quant* this replacement is compensated by adding **pulse_amp** to or subtracting **pulse_amp** from the previously decoded coefficients whose frequency indices are indicated by **pulse_start_sfb** and **pulse_offset**. Note that the pulse escape method is illegal for a block whose **window_sequence** is EIGHT_SHORT_SEQUENCE. The decoding process is specified in the following pseudo-C code:

```
if (pulse_data_present) {
   g = 0;
   win = 0;
   k = swb_offset[pulse_start_sfb];
   for (j = 0; j<number_pulse+1; j++) {
      k += pulse_offset[j];

      /* translate_pulse_parameters(); */
      for( sfb = pulse_start_sfb; sfb<num_swb;sfb++) {
         if( k < swb_offset[sfb+1]) {
            bin = k - swb_offset[sfb] ;
            break;
         }
      }

      /* restore coefficients */
      if (x_quant[g][win][sfb][bin] > 0 )
         x_quant[g][win][sfb][bin] += pulse_amp[j];
      else
         x_quant[g][win][sfb][bin] -= pulse_amp[j];
   }
}
```

Several decoder tools (TNS, filterbank) access the spectral coefficients in a non-interleaved fashion, i.e. all spectral coefficients are ordered according to window number and frequency within a window. This is indicated by using the notation spec[w][k] rather than x_quant[g][w][sfb][bin].

The following pseudo C-code indicates the correspondence between the four-dimensional, or interleaved, structure of array x_quant[ ][ ][ ][ ] and the two-dimensional, or de-interleaved, structure of array spec[ ][ ]. In the latter array the first index increments over the individual windows in the window sequence, and the second index increments over the spectral coefficients that correspond to each window, where the coefficients progress linearly from low to high frequency.

```
quant_to_spec() {
   k=0;
   for( g=0; g<num_window_groups; g++ ) {
      j=0;
      for( sfb=0; sfb < num_swb; sfb ++ ) {
         width = swb_offset[sfb+1] - swb_offset[sfb];
         for( win=0; win<window_group_length[g]; win++ ) {
            for( bin=0; bin<width; bin++ ) {
               spec[win+k][bin+j] = x_quant[g][win][sfb][bin] ;
            }
         }
         j+=width;
      }
      k+=window_group_length[g];
   }
}
```

3.3.4 Tables

Table 0.1 – Scalefactor Huffman codebook parameters

Codebook Number	Dimension of Codebook	index_offset	Range of values	Codebook listed in Table
0	1	-60	-60 to +60	A.1

Table 0.2 – Spectrum Huffman codebooks parameters

Codebook Number, i	unsigned_cb[i]	Dimension of Codebook	LAV for codebook	Codebook listed in Table
0	-	-	0	-
1	0	4	1	A.2
2	0	4	1	A.3
3	1	4	2	A.4
4	1	4	2	A.5
5	0	2	4	A.6
6	0	2	4	A.7
7	1	2	7	A.8
8	1	2	7	A.9
9	1	2	12	A.10
10	1	2	12	A.11
11	1	2	(16) ESC	A.12
12	-	-	(reserved)	-
13	-	-	percept. noise subst.	-
14	-	-	intensity out-of-phase	-
15	-	-	intensity in-phase	-

3.4 Interleaved Vector Quantization

3.4.1 Tool description

This process generates flattened MDCT spectrum using vector quantization. This quantization tool provides high coding gain, even at lower bitrates. Bitstream for this quantizer has a simple fixed-length structure, thus it is robust against transmission channel errors.
The decoding process consists of vector quantization part and reconstruction part. In the vector quantization part, subvectors are specified by codevector index. Then, subvectors are interleaved and combined into one output vector (see fig.3.4.1).

3.4.2 Definitions

Inputs:

f b_shift[][]	Syntax element indicating base frequency of active frequency band of the adaptive bandwidth control.
index0[]:	bitstream element indicating the codevector number of codebook 0
index1[]:	bitstream element indicating the codevector number of codebook 1
window_sequence:	bitstream element indicating window sequence type
side_info_bits	number of bits for side information
bitrate:	system parameter indicating bitrate
used_bits:	number of bits used by variable bit-rate tool, such as long term prediction tool
lyr:	indicates enhancement layer number. Number 0 is assigned for the base layer.

Outputs:

x_flat[]:	reconstructed coefficients

Parameters:

FRAME_SIZE	frame length
MAXBIT	maximum bits for shape codebook index representation
N_CH	number of channels
N_DIV	number of subvectors
N_SF	number of subframes in a frame
sp_cv0[][]	shape codebook of conjugate channel 0
sp_cv1[][]	shape codebook of conjugate channel 1
SP_CB_SIZE	shape codebook size
shape_index0	points the selected codevector of shape codebook 0
shape_index1	points the selected codevector of shape codebook 1
pol0	negates the selected codevector of shape codebook 0
pol1	negates the selected codevector of shape codebook 1

3.4.3 Parameter settings

The assignment of the shape codebook vectors, sp_cv0[][] and sp_cv1[][] is dependent on the window block types as listed in Tables from C.1 to C.30.

Parameters are set initially as listed below:

```
MAXBIT_SHAPE = 6
MAXBIT = MAXBIT_SHAPE + 1
SP_CB_SIZE=(1<<MAXBIT_SHAPE)
FRAME_SIZE = N_FR_L * N_CH
N_SF=  N_FR_L / N_FR
```

3.4.4 Decoding process

3.4.4.1 Initializations

Number of available bits, bits_available_vq is calculated as follows:

```
bits_available_vq =
   (int)(FRAME_SIZE*bitrate/sampling_frequency) - side_info_bits - used_bits;
```

N_DIV represents the number of subvectors. The length of each subvector is calculated by

```
   N_DIV = ((int)((bits_available_vq + MAXBIT*2-1)/(MAXBIT*2)))

   for(idiv=0; idiv<ntt_N_DIV; idiv++){
      length[idiv] = (FRAME_SIZE + N_DIV - 1 - idiv) / N_DIV
   }
```

If codevector length, length[], exceeds the number of codevector elements which are described in tables from C.1 to C.55, undefined elements of sp_cv0[] and sp_cv1[] (i.e. elements beyond defined area) are set to zero.

3.4.4.2 Index unpacking

The quantization index consists of the polarity and shape code information. So in the first stage of the inverse quantization, input indices are unpacked, and polarities and shapes are extracted.

The extracting of polarities is described as follows:

```
for (idiv=0; idiv<N_DIV; idiv++){
   pol0[idiv] = 2 * (index0 [idiv] / SP_CB_SIZE) - 1
   pol1[idiv] = 2 * (index1 [idiv] / SP_CB_SIZE) - 1
}
```

where

pol0[]: polarity of conjugate channel 0

pol1[]: polarity of conjugate channel 1

The shape code extraction is described as follows:

```
for (idiv=0; idiv<N_DIV; idiv++){
   index_shape0[idiv] = index0 [idiv] % SP_CB_SIZE
   index_shape1[idiv] = index1 [idiv] % SP_CB_SIZE
}
```

3.4.4.3 Reconstruction

Output coefficients are reconstructed as follows:

```
for (idiv=0; idiv<N_DIV; idiv++){
   for (icv=0; icv<length[idiv]; icv++){
      if ((icv<length[0]-1) &&
          ((N_DIV%(N_SF*N_CH)==0 && (N_SF*N_CH)>1) || ((N_SF*N_CH)&0x1)==0)))
         itmp = ((idiv+icv)%N_DIV)+icv*N_DIV;
      else
         itmp = idiv + icv * N_DIV;
      ismp = itmp / (N_SF*N_CH) + ((itmp % (N_SF*N_CH)) * (FRAME_SIZE /
(N_SF*N_CH)));
      x_flat_tmp[ismp] =
         (pol0[idiv]*sp_cv0[index_shape0[idiv]][icv]
          + pol1[idiv]*sp_cv1[index_shape1[idiv]][icv]) / 2;
   }
```

```
}
```

where
icv: indicates sample number in the shape code vector
idiv: indicates interleaved-division subvector
ismp: indicates sample number in the subframe
itmp: an integer

3.4.4.4 Adaptive active band selection

This procedure, which is activated in scaleabl configuration modes, limits the active band (see Fig. 3.4.2). For the compression modes, this procedure is not activated and the output x_flat[] is simply copied from x_flat_tmp[].

If lyr=0 or lyr=1, the active band is fixed as listed below:

lyr	ac_btm	ac_top
0 (base)	0.0	1/3
1	0.0	2/3

where ac_btm and ac_top is the bottom and top frequency of the active band, respectively. Values are ranged from 0 to 1 (i.e. 1.0 means the highest frequency).

If lyr >1, active band is selected according to the syntax element fb_shift as follows:

fb_shift	ac_btm	ac_top
0	0.0	2/3
1	1/12	3/4
2	1/6	5/6
3	1/3	1.0

The lower and upper boundaries in MDCT domain are calculated as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   LOWER_BOUNDARY[lyr][i_ch] = ac_btm[lyr][i_ch] * N_FR;
   UPPER_BOUNDARY[lyr][i_ch] = ac_top[lyr][i_ch] * N_FR;
}
```

Then, the output x_flat[] is copied from x_flat_tmp[] as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   for (isf=0; isf<N_SF; isf++){
      for (ismp=0; ismp<LOWER_BOUNDARY[lyr][i_ch]; ismp++){
         x_flat[ismp+(isf+i_ch*N_SF)*N_FR] = 0.;
      }
      for (ismp=LOWER_BOUNDARY[lyr][i_ch]; ismp<UPPER_BOUNDARY[lyr][i_ch]; ismp++){
         ismp2 = ismp - LOWER_BOUNDARY[lyr][i_ch];
         x_flat[ismp+(isf+i_ch*N_SF)*N_FR] = x_flat_tmp[ismp2+(isf+i_ch*N_SF)*N_FR];
      }
      for (ismp=UPPER_BOUNDARY[lyr][i_ch]; ismp<N_FR; ismp++){
         x_flat[ismp+(isf+i_ch*N_SF)*N_FR] = 0.;
      }
   }
}
```

3.4.5 Diagrams

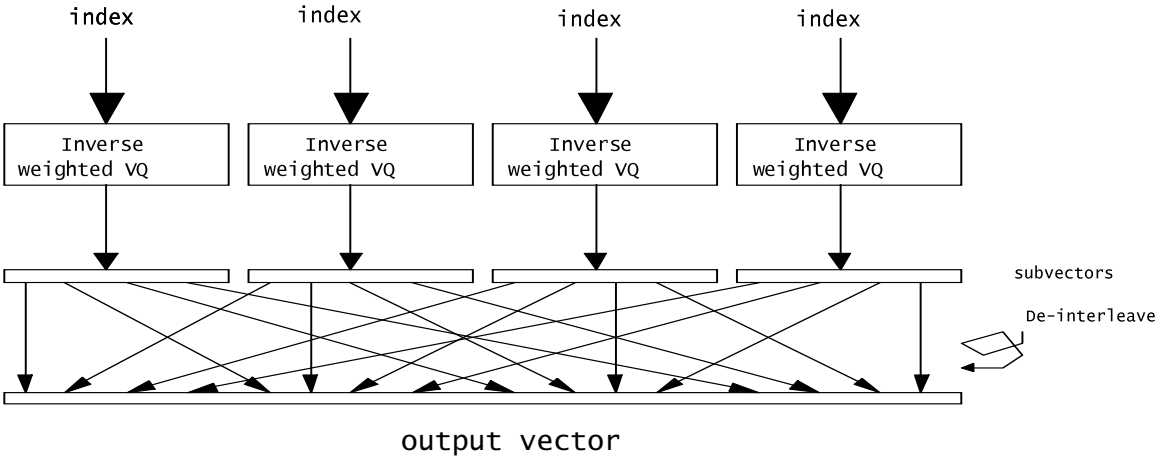


figure 3.4.1 : Decoding process of interleaved voector quantization tool.

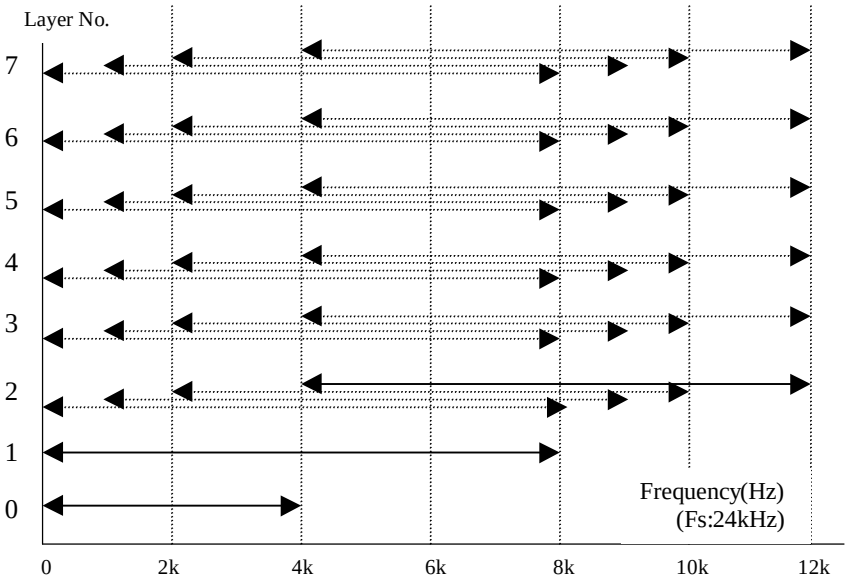


Fig. 3.4.2 Adaptive active band selection.(SAMPLING_FREQUENCY=24kHz)

3.5 Prediction

(non low complexity part identical to ISO/IEC 13818-7)

3.5.1 Tool description

Prediction is used for an improved redundancy reduction and is especially effective in case of more or less stationary parts of a signal which belong to the most demanding parts in terms of required bitrate. Prediction can be applied to every channel using an intra channel (or mono) predictor which exploits the auto-correlation between the spectral components of consecutive frames. Because a window_sequence of type EIGHT_SHORT_SEQUENCE indicates signal changes, i.e. non-stationary signal characteristics, prediction is only used if window_sequence is of type ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE or LONG_STOP_SEQUENCE. The use of the prediction tool is profile dependent. See clause 7 for detailed information.

For each channel prediction is applied to the spectral components resulting from the spectral decomposition of the filterbank. For each spectral component up to limit specified by PRED_SFB_MAX, there is one corresponding predictor resulting in a bank of predictors, where each predictor exploits the auto-correlation between the spectral component values of consecutive frames.

The overall coding structure using a filterbank with high spectral resolution implies the use of backward adaptive predictors to achieve high coding efficiency. In this case, the predictor coefficients are calculated from preceding quantized spectral components in the encoder as well as in the decoder and no additional side information is needed for the transmission of predictor coefficients - as would be required for forward adaptive predictors. A second order backward-adaptive lattice structure predictor is used for each spectral component, so that each predictor is working on the spectral component values of the two preceding frames. The predictor parameters are adapted to the current signal statistics on a frame by frame base, using an LMS based adaptation algorithm. If prediction is activated, the quantizer is fed with a prediction error instead of the original spectral component, resulting in a coding gain.

In order to keep storage requirements to a minimum, predictor state variables are quantized prior to storage.

3.5.2 Definitions

predictor_data_present 1 bit indicating whether prediction is used in current frame (1) or not (0) (always present for ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE and LONG_STOP_SEQUENCE, see 6.3, Table 6.11).

predictor_reset 1 bit indicating whether predictor reset is applied in current frame (1) or not (0) (only present if **predictor_data_present** flag is set, see 6.3, Table 6.11).

predictor_reset_group_number 5 bit number specifying the reset group to be reset in current frame if predictor reset is enabled (only present if **predictor_reset** flag is set, see 6.3, Table 6.11).

prediction_used 1 bit for each scalefactor band (sfb) where prediction can be used indicating whether prediction is switched on (1) / off (0) in that sfb. If **max_sfb** is less than PRED_SFB_MAX then for i greater than or equal to max_sfb, prediction_used[i] is not transmitted and therfore is set to off (0) (only present if **predictor_data_present** flag is set, see 6.3, Table 6.11).

The following table specifies the upper limit of scalefactor bands up to which prediction can be used:

Sampling Frequency (Hz)	Pred_SFB_MAX	Number of Predictors	Maximum Frequency using Prediction (Hz)
96000	33	512	24000.00
88200	33	512	22050.00
64000	38	664	20750.00
48000	40	672	15750.00
44100	40	672	14470.31
32000	40	672	10500.00
24000	41	652	7640.63
22050	41	652	7019.82
16000	37	664	5187.50
12000	37	664	3890.63
11025	37	664	3574.51
8000	34	664	2593.75

This means that at 48 kHz sampling rate prediction can be used in scalefactor bands 0 through 39. According to table 8.5 these 40 scalefactor bands include the MDCT lines 0 through 671, hence resulting in max. 672 predictors.

3.5.3 Decoding process

For each spectral component up to the limit specified by PRED_SFB_MAX of each channel there is one predictor. Prediction is controlled on a single_channel_element or channel_pair_element basis by the transmitted side information in a two step approach, first for the whole frame at all and then conditionally for each scalefactor band individually, see clause 0. The predictor coefficients for each predictor are calculated from preceding reconstructed values of the corresponding spectral component. The details of the required predictor

processing are described in clause 0. At the start of the decoding process, all predictors are initialized. The initialization and a predictor reset mechanism are described in clause 0.

3.5.3.1 Predictor side information

The following description is valid for either one single_channel_element or one channel_pair_element and has to be applied to each such element. For each frame the predictor side information has to be extracted from the bitstream to control the further predictor processing in the decoder. In case of a single_channel_element the control information is valid for the predictor bank of the channel associated with that element. In case of a channel_pair_element there are the following two possibilities: If **common_window** = 1 then there is only one set of the control information which is valid for the two predictor banks of the two channels associated with that element. If **common_window** = 0 then there are two sets of control information, one for each of the two predictor banks of the two channels associated with that element.

If window_sequence is of type ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE or LONG_STOP_SEQUENCE, the **predictor_data_present** bit is read. If this bit is not set (0) then prediction is switched off at all for the current frame and there is no further predictor side information present. In this case the **prediction_used** bit for each scalefactor band stored in the decoder has to be set to zero. If the **predictor_data_present** bit is set (1) then prediction is used for the current frame and the **predictor_reset** bit is read which determines whether predictor reset is applied in the current frame (1) or not (0). If **predictor_reset** is set then the next 5 bits are read giving a number specifying the group of predictors to be reset in the current frame, see also clause 0 for the details. If the **predictor_reset** is not set then there is no 5 bit number in the bitstream. Next, the **prediction_used** bits are read from the bitstream, which control the use of prediction in each scalefactor band individually, i.e. if the bit is set for a particular scalefactor band, then prediction is enabled for all spectral components of this scalefactor band and the quantized prediction error of each spectral component is transmitted instead of the quantized value of the spectral component. Otherwise, prediction is disabled for this scalefactor band and the quantized values of the spectral components are transmitted.

3.5.3.2 AAC predictor processing

3.5.3.2.1 General

The following description is valid for one single predictor and has to be applied to each predictor. A second order backward adaptive lattice structure predictor is used. Figure 13.1 shows the corresponding predictor flow graph on the decoder side. In principle, an estimate $x_{est}(n)$ of the current value of the spectral component $x(n)$ is calculated from preceding reconstructed values $x_{rec}(n-1)$ and $x_{rec}(n-2)$, stored in the register elements of the predictor structure, using the predictor coefficients $k_1(n)$ and $k_2(n)$. This estimate is then added to the quantized prediction error $e_q(n)$ reconstructed from the transmitted data resulting in the reconstructed value $x_{rec}(n)$ of the current spectral component $x(n)$. Figure 13.2 shows the block diagram of this reconstruction process for one single predictor.

Due to the realization in a lattice structure, the predictor consists of two so-called basic elements which are cascaded. In each element, the part $x_{est,m}(n)$, $m=1, 2$ of the estimate is calculated according to

$$x_{est,m}(n) = b \cdot k_m(n) \cdot a \cdot r_{q,m-1}(n-1) ,$$

where

$$r_{q,m}(n) = r_{q,m-1}(n-1) - b \cdot k_m(n) \cdot e_{q,m-1}(n)$$

and $$e_{q,m}(n) = e_{q,m-1}(n) - x_{est,m}(n) .$$

Hence, the overall estimate results to: $$x_{est}(n) = x_{est,1}(n) + x_{est,2}(n)$$

The constants

$$a \text{ and } b , \qquad 0 < a, b \le 1$$

are attenuation factors which are included in each signal path contributing to the recursivity of the structure for the purpose of stabilization. By this means, possible oscillations due to transmission errors or drift between predictor coefficients on the encoder and decoder side due to numerical inaccuracy can be faded out or even prevented.

In the case of stationary signals and with $a = b = 1$, the predictor coefficient of element m is calculated by

$$k_m = \frac{E\left[e_{q,m-1}(n) \cdot r_{q,m-1}(n-1)\right]}{\frac{1}{2} \cdot \left(E\left[e^2_{q,m-1}(n)\right] + E\left[r^2_{q,m-1}(n-1)\right]\right)}, \qquad m = 1, 2 \quad \text{and} \quad e_{q,0}(n) = r_{q,0}(n) = x_{rec}(n)$$

In order to adapt the coefficients to the current signal properties, the expected values in the above equation are substituted by time average estimates measured over a limited past signal period. A compromise has to be chosen between a good convergence against the optimum predictor setting for signal periods with quasi stationary characteristic and the ability of fast adaptation in case of signal transitions. In this context algorithms with iterative improvement of the estimates, i.e. from sample to sample, are of special interest. Here, a "least mean square" (LMS) approach is used and the predictor coefficients are calculated as follows

$$k_m(n+1) = \frac{COR_m(n)}{VAR_m(n)}$$

with

$$COR_m(n) = \alpha \cdot COR_m(n-1) + r_{q,m-1}(n-1) \cdot e_{q,m-1}(n)$$

$$VAR_m(n) = \alpha \cdot VAR_m(n-1) + 0.5 \cdot \left(r^2_{q,m-1}(n-1) + e^2_{q,m-1}(n)\right)$$

where α is an adaptation time constant which determines the influence of the current sample on the estimate of the expected values. The value of α is chosen to

$$\alpha = 0.90625 .$$

The optimum values of the attenuation factors *a* and *b* have to be determined as a compromise between high prediction gain and small fade out time. The chosen values are

$$a = b = 0.953125 .$$

Independent of whether prediction is disabled - either at all or only for a particular scalefactor band - or not, all the predictors are run all the time in order to always adapt the coefficients to the current signal statistics.

If window_sequence is of type ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE and LONG_STOP_SEQUENCE only the calculation of the reconstructed value of the quantized spectral components differs depending on the value of the **prediction_used** bit:

- If the bit is set (1), then the quantized prediction error reconstructed from the transmitted data is added to the estimate $x_{est}(n)$ calculated by the predictor resulting in the reconstructed value of the quantized spectral component, i.e. $x_{rec}(n) = x_{est}(n) + e_q(n)$
- If the bit is not set (0), then the quantized value of the spectral component is reconstructed directly from the transmitted data.

In case of short blocks, i.e. window_sequence is of type EIGHT_SHORT_SEQUENCE, prediction is always disabled and a reset is carried out for all predictors in all scalefactor bands, which is equivalent to a reinitialization, see clause 0.

For a single_channel_element, the predictor processing for one frame is done according to the following pseudo code:

(It is assumed that the reconstructed value y_rec(c) - which is either the reconstructed quantized prediction error or the reconstructed quantized spectral coefficient - is available from previous processing.)

```
if (ONLY_LONG_SEQUENCE || LONG_START_SEQUENCE || LONG_STOP_SEQUENCE) {
   for ( sfb=0; sfb<PRED_SFB_MAX; sfb++) {
      fc = swb_offset_long_window[fs_index][sfb];
      lc = swb_offset_long_window[fs_index][sfb+1];
      for (c= fc; c<lc; c++) {
```

```
        x_est[c] = predict();
        if (predictor_data_present && prediction_used[sfb] )
           x_rec[c] = x_est[c] + y_rec[c];
        else
           x_rec[c] = y_rec[c];
     }
   }
}
else {
   reset_all_predictors();
}
```

In case of channel_pair_elements with **common_window** = 1, the only difference is that the computation of x_est and x_rec in the inner for loop is done for both channels associated with the channel_pair_element. In case of channel_pair_elements with **common_window** = 0, each channel has prediction applied using that channel's prediction side information.

3.5.3.2.2 Quantization in Predictor Calculations

For a given predictor six state variables need to be saved: r_0, r_1, COR_1, COR_2, VAR_1 and VAR_2. These variables will be saved as truncated IEEE floating-point numbers (i.e. the 16 msb of a float storage word).

The predicted value x_{est} will be rounded to a 16-bit floating point representation (i.e. round to a 7-bit mantissa) prior to being used in any calculation. The exact rounding algorithm to be used is shown in pseudo-C function flt_round_inf(). Note that for complexity considerations, *round to nearest, infinity* is used instead of *round to nearest, even.*

The expressions (b / VAR_1) and (b / VAR_2) will be rounded to a 16-bit floating point representation (i.e. round to a 7-bit mantissa), which permits the ratio to be computed via a pair of small look-up tables. C-code for generating such tables is shown in pseudo-C function make_inv_tables().

All intermediate results in every floating point computation in the prediction algorithm will be represented in single precision floating point using rounding described below.

The IEEE Floating Point computational unit used in executing all arithmetic in the prediction tool will enable the following options:

- Round-to-Nearest, Even - Round to nearest representable value; round to the value with the least significant bit equal to zero (even) when the two nearest representable values are equally near.
- Overflow exception - Values whose magnitude is greater than the largest representable value will be set to the representation for infinty.
- Underflow exception - Gradual underflow (de-normalized numbers) will be supported; values whose magnitude is less than the smallest representable value will be set to zero.

3.5.3.2.3 Fast Algorithm for Rounding

```
/* this does not conform to IEEE conventions of round to
 * nearest, even, but it is fast
 */
static void
flt_round_inf(float *pf)
{
    int flg;
    ulong tmp, tmp1;
    float *pt = (float *)&tmp;
    *pt = *pf;                          /* write float to memory */
    tmp1 = tmp;                         /* save in tmp1 */
    flg = tmp & (ulong)0x00008000;      /* rounding position */
    tmp &= (ulong)0xffff0000;    /* truncated float */
    *pf = *pt;
    /* round 1/2 lsb toward infinity */
    if (flg) {
      tmp = tmp1 & (ulong)0xff810000;   /* 1.0 * 2^e + 1 lsb */
      *pf += *pt;                       /* add 1.0 * 2^e+ 1 lsb */
      tmp &= (ulong)0xff800000;         /* 1.0 * 2^e */
      *pf -= *pt;                       /* subtract 1.0 * 2^e */
   }
}
```

3.5.3.2.4 Generating Rounded b / Var

```
static float mnt_table[128];
static float exp_table[256];

/* function flt_round_even() only works for arguments in the range
 *           1.0 < *pf  < 2.0 - 2^-24
 */
static void
flt_round_even(float *pf)
{
   float f1, f2;

   f1 = 1.0;
   f2 = f1 + (*pf / (1<<15));
   f2 = f2 - f1;
   f2 = f2 * (1<<15);
   *pf = f2;
}

static void
make_inv_tables(void)
{
   int i;
   ulong tmp1, tmp;
   float *pf = (float *)&tmp;
   float ftmp;

   *pf = 1.0;
   tmp1 = tmp;                  /* float 1.0 */
   /* mantissa table */
   for (i=0; i<128; i++) {
      tmp = tmp1 + (i<<16);     /* float 1.m, 7 msb only */
      ftmp = b / *pf;           /* predictor constant b as in 8.3.2 */
      flt_round_even(&ftmp);    /* round to 16 bits */
      mnt_table[i] = ftmp;
   }

   /* exponent table */
   for (i=0; i<256; i++) {
      tmp = tmp1 + i<<23;       /* float 1.0 * 2^exp */
      ftmp = 1.0 / *pf;
      exp_table[i] = ftmp;
   }
}
```

3.5.3.3 Low complexity predictor processing

This Low Complexity (LC) mode of backward adaptive prediction delivers the same performance as the main mode (AAC predictor) described above, but with almost half complexity. The bitstream syntax of this prediction mode is exactly the same as in the AAC mode. The adaptation function is however different so that the LC mode is not functionally conformant to the AAC predictor. When compliance with MPEG-2 AAC is not necessary, this mode can be used to achieve lower complexity of decoding.

As the only difference between these two predictors is in the adaptive coefficient update part, this part is presented in this section. Other parts required by the LC prediction can be found in prediction section.

As the adaptive lattice predictor, the LC prediction also uses the second order predictor.

$$x_{est}(n) = a_1 x_{rec}(n-1) + a_2 x_{rec}(n-2)$$

The predictor coefficients are calculated according to the reconstructed spectral components. In order to reduce the complexity, we update (or estimate) the predictor every four samples. The covariance estimates of the reconstructed signal are computed by

$$r_{0,0} = 2\sum_{i=1}^{L-2} x_{rec}{}^2(n-i)\,,\ r_{1,1} = \sum_{i=1}^{L-2}(x_{rec}{}^2(n-i-1) + x_{rec}{}^2(n-i+1))\,,$$

$$r_{0,1} = r_{1,0} = r_1 = \sum_{i=1}^{L-2}(x_{rec}(n-i)x_{rec}(n-i-1) + x_{rec}(n-i+1)x_{rec}(n-i))\,,$$

$$r_2 = 2\sum_{i=1}^{L-2} x_{rec}(n-i-1)x_{rec}(n-i+1)$$

For data length five, assume that we have the following data available

$$x_{rec}(n-4), x_{re}(n-3), x_{rec}(n-2), x_{rec}(n-1), x_{rec}(n)\,.$$

Then an obvious efficient algorithm is

$$r_{0,0} = 2*(x_{rec}{}^2(n-1) + x_{rec}{}^2(n-2) + x_{rec}{}^2(n-3))\,,$$

$$r_{1,1} = 2*x_{rec}{}^2(n-2) + x_{rec}{}^2(n) + x_{rec}{}^2(n-1) + x_{rec}{}^2(n-3) + x_{rec}{}^2(n-4)\,,$$

$$r_{0,1} = r_{1,0} = r_1 = x_{rec}(n-4)x_{rec}(n-3) + 2*(x_{rec}(n-3) + x_{rec}(n-1))*x_{rec}(n-2) + x_{rec}(n-1)x_{rec}(n)$$

$$r_2 = ((x_{rec}(n-4) + x_{rec}(n))*x_{rec}(n-2) + x_{rec}(n-1)x_{rec}(n-3))*2\,.$$

With these covariances, the LP coefficients can be calculated by the following equation:

$$a_1 = \frac{(r_{1,1} - r_2)r_{0,1}}{r_{0,0}r_{1,1} - r_{0,1}^2},$$

$$a_2 = \frac{r_{0,0}r_2 - r_{0,1}r_1}{r_{0,0}r_{1,1} - r_{0,1}^2}.$$

If window_sequence is of type ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE and LONG_STOP_SEQUENCE only the calculation of the reconstructed value of the quantized spectral components differs depending on the value of the **prediction_used** bit:

- If the bit is set (1), then the quantized prediction error reconstructed from the transmitted data is added to the estimate $x_{est}(n)$ calculated by the predictor resulting in the reconstructed value of the quantized spectral component, i.e. $x_{rec}(n) = x_{est}(n) + e_q(n)$
- If the bit is not set (0), then the quantized value of the spectral component is reconstructed directly from the transmitted data.

In case of short blocks, i.e. window_sequence is of type EIGHT_SHORT_SEQUENCE, prediction is always disabled and a reset is carried out for all predictors in all scalefactor bands, which is equivalent to a reinitialization, see clause 0.

For a single_channel_element, the predictor processing for one frame is done according to the following pseudo code:

(It is assumed that the reconstructed value y_rec(c) - which is either the reconstructed quantized prediction error or the reconstructed quantized spectral coefficient - is available from previous processing.)

```
if (ONLY_LONG_SEQUENCE || LONG_START_SEQUENCE || LONG_STOP_SEQUENCE) {
   for ( sfb=0; sfb<PRED_SFB_MAX; sfb++) {
      fc = swb_offset_long_window[fs_index][sfb];
      lc = swb_offset_long_window[fs_index][sfb+1];
      for (c= fc; c<lc; c++) {
         x_est[c] = predict();
         if (predictor_data_present && prediction_used[sfb] )
            x_rec[c] = x_est[c] + y_rec[c];
         else
            x_rec[c] = y_rec[c];
      }
   }
}
else {
   reset_all_predictors();
}
```

In case of channel_pair_elements with **common_window** = 1, the only difference is that the computation of x_est and x_rec in the inner for loop is done for both channels associated with the channel_pair_element. In case of channel_pair_elements with **common_window** = 0, each channel has prediction applied using that channel's prediction side information.

Quantization in Predictor Calculations

For a given predictor seven state variables need to be saved: $x_{rec}(n-4), x_{rec}(n-3), x_{rec}(n-2), x_{rec}(n-1), x_{rec}(n), a_1, a_2$. Because we update the predictor every four samples, not all variables are neccessary to be saved. Actually, for reconstructed spectral components, we only need to save ten variables every time. Including eight prediction coefficients, a total of eighteen variables needs to be saved. The ten reconstructed spectral compoenents will be saved as truncated IEEE floating-point numbers (i.e. the 16 msb of a float storage word) and other eight prediction coefficients will be uniformly quantized into eight bits.

The dequantized error spectral components are rounded toa 16-bit foating point representation(i.e. round to a 7-bit mantissa). The exact rounding algorithm to be used is shown in pseudo-C function flt_round_inf(). Note that for complexity considerations, *round to nearest, infinity* is used instead of *round to nearest, even* The recontructed spectral components (the predicted value x_{est} plus the dequantized spectral compoenent. error) will be truncated to a 16-bit floating point representation prior to being used in any calculation.

3.5.3.4 Predictor reset

Initialization of a predictor means that the predictor's state variables are set as follows: $r_0 = r_1 = 0$, $COR_1 = COR_2 = 0$, $VAR_1 = VAR_2 = 1$. When the decoding process is started, all predictors are initialized.

A cyclic reset mechanism is applied by the encoder and signaled to the decoder, in which all predictors are initialized again in a certain time interval in an interleaved way. On one hand this increases predictor stability by re-synchronizing the predictors of the encoder and the decoder and on the other hand it allows defined entry points in the bitstream.

The *whole set of predictors is subdivided into 30 so-called reset groups according to the following table:*

Reset group number	*Predictors of reset group*
1	*P0, P30, P60, P90,...*
2	*P1, P31, P61, P91,...*
3	*P2, P32, P62, P92,...*
...	
30	*P29, P59, P89, P119,...*

where P_i is the predictor which corresponds to the spectral coefficient indexed by i.

Whether or not a reset has to be applied in the current frame is determined by the **predictor_reset** bit. If this bit is set then the number of the predictor reset group to be reset in the current frame is specified in **predictor_reset_group_number**. All predictors belonging to that reset group are then initialized as described above. This initialization has to be done after the normal predictor processing for the current frame has been carried out. Note that **predictor_reset_group_number** cannot have the value 0 or 31.

A typical reset cycle starts with reset group number 1 and the reset group number is then incremented by 1 until it reaches 30, and then it starts with 1 again. Nevertheless, it may happen, e.g. due to switching between programs (bitstreams) or cutting and pasting, that there will be a discontinuity in the reset group numbering. If this is the case, these are the following three possibilities for decoder operation:

- Ignore the discontinuity and carry on the normal processing. This may result in a short audible distortion due to a mismatch (drift) between the predictors in the encoder and decoder. After one complete reset cycle

(reset group n, n+1, ..., 30, 1, 2, ..., n-1) the predictors are re-synchronized again. Furthermore, a possible distortion is faded out because of the attenuation factors a and b.

- Detect the discontinuity, carry on the normal processing but mute the output until one complete reset cycle is performed and the predictors are re-synchronized again.
- Reset all predictors.

An encoder is required to signal the reset of a group at least once every 8 frames. Groups do not have to be reset in ascending order, but every group must be reset within the maximum reset interval of 8 x 30 = 240 frames. The bitstream syntax permits the encoder to signal the reset of a group at every frame, resulting in a minimum reset interval of 1 x 30 = 30 frames.

In case of a single_channel_element or a channel_pair_element with **common_window** = 0, the reset has to be applied to the predictor bank(s) of the channel(s) associated with that element. In case of a channel_pair_element with **common_window** = 1, the reset has to be applied to the two predictor banks of the two channels associated with that element.

In the case of a short block (i.e. window_sequence of type EIGHT_SHORT_SEQUENCE) all predictors in all scalefactor bands must be reset.

3.5.4 Diagrams

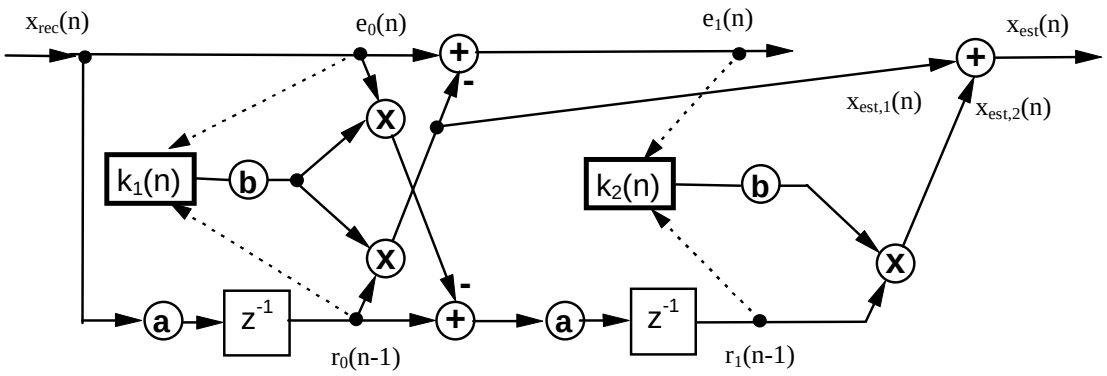


Figure 0.1 – Flow graph of AAC intra channel predictor for one spectral component in the decoder. The dotted lines indicate the signal flow for the adaptation of the predictor coefficients.

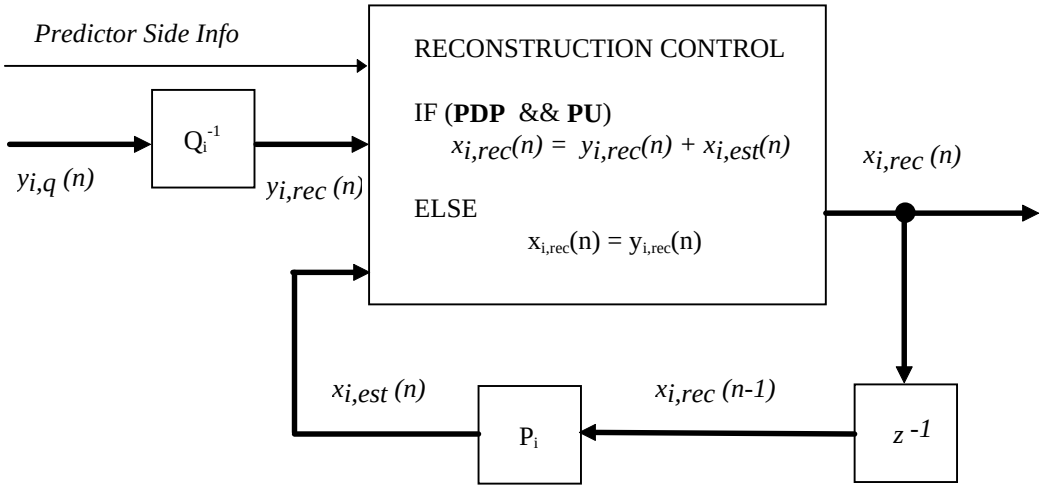


Figure 13.2 - Block diagram of decoder prediction unit for one single spectral component with predictor P_i and inverse quantizer Q_i^{-1}. The following abbreviations for the predictor side information:
PDP - predictor_data_present, PU - prediction_used.

3.6 Long Term Prediction

3.6.1 Tool description

Long term prediction (LTP) is an efficient tool for reducing the redundancy of signal between successive coding blocks. This tool is especially effective for the parts of a signal which have clear pitch property. The implementation complexity of LTP is significantly lower than the complexity of Backward Adaptive Prediction. Because the Long Term Predictor is a forward adaptive predictor (prediction coefficients are sent as side information), it is inherently less sensitive to round-off numerical errors in the decoder or bit errors in the transmitted spectral coefficients.
With MPEG-2 AAC based coding LTP can be used for any window type. With Interleaved Vector Quantisation (Twin-VQ) LTP can be only used for the long window type.

3.6.2 Definitions

ltp_data_present 1 bit indicating whether prediction is used in current frame (1) or not (0) (always present)

ltp_lag 11 bit number specifying the optimal delay from 0 to 2047

ltp_coef 3 bit index indicating the LTP coefficient in the table below. For all short windows in the current frame, the same coefficient is always used.

value of **ltp_coef**	value of LTP coefficient
000	0.570829
001	0.696616
010	0.813004
011	0.911304
100	0.984900
101	1.067894
110	1.194601
111	1.369533

ltp_short_used 1 bit indicating whether LTP is used for each short window (1) or not (0)

ltp_short_lag_present 1 bit indicating whether **ltp_short_lag** is actually transmitted (1), or omitted (0) from the bit-stream, which means that the value of **ltp_short_lag** is 0

ltp_short_lag 4 bit number specifying the relative delay for each short window to **ltp_lag** from -8 to 7

ltp_long_used 1 bit for each scalefactor band (sfb) where LTP can be used indicating whether LTP is switched on (1) or off (0) in that sfb.

3.6.3 Decoding process

The decoding process for LTP is carried out on each window of the current frame by applying 1-tap IIR filtering in the time domain to predict samples in the current frame by (quantised) samples in the previous frames. The processis controlled by the transmitted side information in a two step approach. The first control step defines whether LTP is used at all for the current frame. In the case of long window, the second control step defines, on which scalefactor bands LTP is used. In case of short windows the second control step defines which of the short windows in the coding block LTP is applied to. At the start of the decoding process, the reconstructed time samples are initialized by zeros.

For each frame, the LTP side information is extracted from the bitstream to control the further predictor processing at the decoder. In case of a single_channel_element the control information is valid for the channel with that element. In case of a channel_pair_element there are two sets of control data.

First, the **ltp_data_present** bit is read. If this bit is not set (0) then LTP is switched off for current frame and there is no further predictor side information present. In this case the **ltp_long_used** flag for each scalefactor band stored in the decoder has to be set to zero. If the **ltp_data_present** bit is set (1) then LTP is used for the current frame and the LTP parameters are read. The decoding process is different for long and short windows.

For long window, the LTP parameters are used to calculate the predicted time signals using the following formula:

$$x\_est(i) = \text{ltp\_coef} * x\_rec(-N-1-\text{ltp\_lag}+i),$$
$$i = 0,\ldots,N$$

where $x\_est(i)$ are the predicted samples
$x\_rec(i)$ are reconstructed time domain samples
N is the length of transform window

Using the MDCT for long window, the predicted spectral components are obtained for current frame from the predicted time domain signal. Next, the **ltp_long_used** bits are read from the bitstream, which control the use of prediction in each scalefactor band individually, i.e. if the bit is set for a particular scalefactor band, all the predicted spectral components of this scalefactor band are used. Otherwise the predicted spectral components are set to zeros. That is, if the **ltp_long_used** bit is set, then the quantized prediction error reconstructed from the transmitted data is added to the predicted spectral component. If the bit is not set (0), then the quantized value of spectral component is reconstructed directly from the transmitted data.

For each short window, the bit **ltp_short_used** is read from the bitstream. If the **ltp_short_used** is not set, the quantized value of spectral component is reconstructed directly from the transmitted data and the time domain signal can be reconstructed for this particular subframe. If the **ltp_short_used** is set, the **ltp_short_lag**_present is read. If **ltp_short_lag_present** is set then **ltp_short_lag** is read. If **ltp_short_lag_present** is not set, the value of **ltp_short_lag** is set to 0. The value of **ltp_short_lag** is combined with **ltp_lag** and **ltp_coef** to calculate the predicted time domain signal for this particular subframe. Using the MDCT for short window, the predicted spectral components are calculated and the spectral components in the first eight scalefactor bands are added to the quantized prediction error reconstructed from the transmitted data.

The signal reconstruction part of decoding process for one channel can be described as following pseudo code. Here *x_est* is the predicted time domain signal, *X_est* is the corresponding frequency domain vector, *Y_rec* is the vector of decoded spectral coefficients and *X_rec* is the vector of reconstructed spectral coefficients.

```
if (ONLY_LONG_SEQUENCE || LONG_START_SEQUENCE || LONG_STOP_SEQUENCE) {
   x_est = predict();
   X_est = MDCT(x_est)
   for (sfb=0; sfb<NUMBER_SCALEFACTOR_BAND; sfb++) {
        if (ltp_data_present && ltp_long_used[sfb] )
        X_rec = X_est + Y_rec;
        else
           X_rec = Y_rec;
     }
}
else {
     for (w=0; w<num_windows; w++) {
        if(ltp_data_present && ltp_short_used[w] {
           x_est = predict();
           X_est = MDCT(x_est)
        for ( sfb=0; sfb<8; sfb++)
           X_rec = X_est + Y_rec;
        }
        else
             X_rec = Y_rec
      }
}
```

3.7 Joint Coding

3.7.1 M/S Stereo

(similar to ISO/IEC 13818-7)

3.7.1.1 Tool description

The M/S joint channel coding operates on channel pairs. Channels are most often paired such that they have symmetric presentation relative to the listener, such as left/right or left surround/right surround. The first channel in the pair is denoted “left” and the second “right.” On a per-spectral-coefficient basis, the vector formed by the left and right channel signals is reconstructed or de-matrixed by either the identity matrix

$$\begin{bmatrix} l \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} l \\ r \end{bmatrix}$$

or the inverse M/S matrix

$$\begin{bmatrix} l \\ r \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} m \\ s \end{bmatrix}$$

The decision on which matrix to use is done on a scalefactor band by scalefactor band basis as indicated by the ms_used flags. M/S joint channel coding can only be used if common_window is ‘1’ (see clause 8.3.1).

3.7.1.2 Definitions

ms_mask_present this two bit field indicates that the MS mask is
00 All zeros
01 A mask of max_sfb bands of ms_used follows this field
10 All ones
11 Reserved
(see 6.3, Table 6.10)

ms_used[g][sfb] one-bit flag per scalefactor band indicating that M/S coding is being used in windowgroup g and scalefactor band sfb (see 6.3, Table 6.10).

l_spec[] Array containing the left channel spectrum of the respective channel pair.
r_spec[] Array containing the right channel spectrum of the respective channel pair.
is_intensity(g,sfb) function returning the intensity status, defined in 12.2.3
is_noise(g,sfb) function returning the noise substitution status, defined in 2.12.1

3.7.1.3 Decoding Process

Reconstruct the spectral coefficients of the first (“left”) and second (“right”) channel as specified by the **mask_present** and the **ms_used[][]** flags as follows:

```
if (mask_present >= 1) {
   for (g=0; g<num_window_groups; g++) {
      for (b=0; b<window_group_length[g]; b++) {
         for(sfb=0; sfb<max_sfb; sfb++) {
            if ((ms_used[g][sfb] || mask_present == 2) &&
                  !is_intensity(g,sfb) && !is_noise(g,sfb)) {
               for (i=0; i< swb_offset[sfb+1]-swb_offset[sfb]; i++) {
                  tmp = l_spec[g][b][sfb][i] -
                      r_spec[g][b][sfb][i];
                  l_spec[g][b][sfb][i] =  l_spec[g][b][sfb][i] +
                                          r_spec[g][b][sfb][i];
                  r_spec[g][b][sfb][i] = tmp;
               }
            }
```

```
        }
      }
   }
}
```

Please note that ms_used[][] is also used in the context of intensity stereo coding and perceptual noise substitution. If intensity stereo coding or noise substitution is on for a particular scalefactor band, no M/S stereo decoding is carried out.

3.7.1.4 Diagrams

3.7.1.5 Tables

3.7.2 Intensity Stereo

(identical to ISO/IEC 13818-7)

3.7.2.1 Tool description

This tool is used to implement joint intensity stereo coding between both channels of a channel pair. Thus, both channel outputs are derived from a single set of spectral coefficients after the inverse quantization process. This is done selectively on a scalefactor band basis when intensity stereo is flagged as active.

3.7.2.2 Definitions

hcod_sf[]	Huffman codeword from the Huffman code table used for coding of scalefactors (see clause 9.2)
dpcm_is_position[][]	Differentially encoded intensity stereo position
is_position[group][sfb]	Intensity stereo position for each group and scalefactor band
l_spec[]	Array containing the left channel spectrum of the respective channel pair
r_spec[]	Array containing the right channel spectrum of the respective channel pair

3.7.2.3 Decoding Process

The use of intensity stereo coding is signaled by the use of the pseudo codebooks INTENSITY_HCB and INTENSITY_HCB2 (15 and 14) in the right channel (use of these codebooks in a left channel of a channel pair element is illegal). INTENSITY_HCB and INTENSITY_HCB2 signal in-phase and out-of-phase intensity stereo coding, respectively.
In addition, the phase relationship of the intensity stereo coding can be reversed by means of the ms_used field: Because M/S stereo coding and intensity stereo coding are mutually exclusive for a particular scalefactor band and group, the primary phase relationship indicated by the Huffman code tables is changed from in-phase to out-of-phase or vice versa if the corresponding ms_used bit is set for the respective band.
The directional information for the intensity stereo decoding is represented by an "intensity stereo position" value indicating the relation between left and right channel scaling. If intensity stereo coding is active for a particular group and scalefactor band, an intensity stereo position value is transmitted instead of the scalefactor of the right channel.
Intensity positions are coded just like scalefactors, i.e. by Huffman coding of differential values with two differences:

- there is no first value that is sent as PCM. Instead, the differential decoding is started assuming the last intensity stereo position value to be zero.
- Differential decoding is done separately between scalefactors and intensity stereo positions. In other words, the scalefactor decoder ignores interposed intensity stereo position values and vice versa (see clause 11.3.2)

The same codebook is used for coding intensity stereo positions as for scalefactors.

Two pseudo functions are defined for use in intensity stereo decoding:

```
function is_intensity(group,sfb) {
   +1   for window groups / scalefactor bands with right channel
          codebook sfb_cb[group][sfb] == INTENSITY_HCB
   -1   for window groups / scalefactor bands with right channel
          codebook sfb_cb[group][sfb] == INTENSITY_HCB2
```

```
   0    otherwise
   }

function invert_intensity(group,sfb)   {
   1-2*ms_used[group][sfb]   if (ms_mask_present == 1)
   +1                        otherwise
   }
```

The intensity stereo decoding for one channel pair is defined by the following pseudo code:

```
p = 0;
for (g=0; g<num_window_groups; g++)  {

   /* Decode intensity positions for this group */
   for (sfb=0; sfb<max_sfb; sfb++)
      if (is_intensity(g,sfb))
         is_position[g][sfb] = p += dpcm_is_position[g][sfb];

   /* Do intensity stereo decoding */
   for (b=0; b<window_group_length[g]; b++)  {
      for (sfb=0; sfb<max_sfb; sfb++)  {
         if (is_intensity(g,sfb))  {

            scale = is_intensity(g,sfb) * invert_intensity(g,sfb) *
                      0.5^(0.25*is_position[g][sfb]);
            /* Scale from left to right channel,
               do not touch left channel */
            for (i=0; i<swb_offset[sfb+1]-swb_offset[sfb]; i++)
               r_spec[g][b][sfb][i] = scale * l_spec[g][b][sfb][i];

         }

      }
   }

}
```

3.7.2.4 Integration with Intra Channel Prediction Tool

For scalefactor bands coded in intensity stereo the corresponding predictors in the right channel are switched to "off" thus effectively overriding the status specified by the prediction_used mask. The update of these predictors is done by feeding the intensity stereo decoded spectral values of the right channel as the "last quantized value" $x_{rec(n-1)}$. These values result from the scaling process from left to right channel as described in the pseudo code. The function of Long Term Prediction does not depend on Intensity Stereo.

3.7.3 Coupling Channel

(identical to ISO/IEC 13818-7)

3.7.3.1 Tool description

Coupling channel elements provide two functionalities: First, coupling channels may be used to implement generalized intensity stereo coding where channel spectra can be shared across channel boundaries. Second, coupling channels may be used to dynamically perform a downmix of one sound object into the stereo image. Note that this tool includes certain profile dependent parameters (see clause 7.1).

3.7.3.2 Definitions

ind_sw_cce_flag	one bit indicating whether the coupled target syntax element is an independently switched (1) or a dependently switched (0) CCE (see 6.3, Table 6.18).
num_coupled_channels	number of coupled target channels (see 6.3, Table 6.18)
cc_target_is_cpe	one bit indicating if the coupled target syntax element is a CPE (1) or a SCE (0) (see 6.3, Table 6.18).
cc_target_tag_select	four bit field specifying the element_instance_tag of the coupled target syntax element (see 6.3, Table 6.18).
cc_l	one bit indicating that a list of gain_element values is applied to the left channel of a channel pair (see 6.3, Table 6.18).

cc_r one bit indicating that a list of gain_element values is applied to the right channel of a channel pair (see 6.3, Table 6.18).

cc_domain one bit indicating whether the coupling is performed before (0) or after (1) the TNS decoding of the coupled target channels (see 6.3, Table 6.18)

gain_element_sign one bit indicating if the transmitted gain_element values contain information about in-phase / out-of-phase coupling (1) or not (0) (see 6.3, Table 6.18)

gain_element_scale determines the amplitude resolution *cc_scale* of the scaling operation according to Table 00.11 (see 6.3, Table 6.18)

common_gain_element_present[c] one bit indicating whether Huffman coded common_gain_element values are transmitted (1) or whether Huffman coded differential gain_elements are sent (0) (see 6.3, Table 6.18)

dpcm_gain_element[][] Differentially encoded gain element

gain_element[group][sfb] Gain element for each group and scalefactor band

common_gain_element[] Gain element that is used for all window groups and scalefactor bands of one coupling target channel

spectrum_m(idx, domain) Pointer to the spectral data associated with the single_channel_element with index idx. Depending on the value of "domain", the spectral coefficients before (0) or after (1) TNS decoding are pointed to.

spectrum_l(idx, domain) Pointer to the spectral data associated with the left channel of the channel_pair_element with index idx. Depending on the value of "domain", the spectral coefficients before (0) or after (1) TNS decoding are pointed to.

spectrum_r(idx, domain) Pointer to the spectral data associated with the right channel of the channel_pair_element with index idx. Depending on the value of "domain", the spectral coefficients before (0) or after (1) TNS decoding are pointed to.

3.7.3.3 Decoding Process

The coupling channel is based on an embedded single_channel_element which is combined with some dedicated fields to accomodate its special purpose.

The coupled target syntax elements (SCEs or CPEs) are addressed using two syntax elements. First, the cc_target_is_cpe field selects whether a SCE or CPE is addressed. Second, a cc_target_tag_select filed selects the instance_tag of the SCE/CPE.

The scaling operation involved in channel coupling is defined by gain_element values which describe the applicable gain factor and sign. In accordance with the coding procedures for scalefactors and intensity stereo positions, gain_element values are differentially encoded using the Huffman table for scalefactors. Similarly, the decoded gain factors for coupling relate to window groups of spectral coefficients.

Independently switched CCEs vs. dependently switched CCEs

There are two kinds of CCEs. They are “independently switched” and “dependently switched” CCEs. An independently switched CCE is a CCE in which the window state (i.e. window_sequence and window_shape) of the CCE does not have to match that of any of the SCE or CPE channels that the CCE is coupled onto (target channels). This has several important implications:

- First, it is required that an independently switched CCE must only use the common_gain element, not a list of gain_elements.
- Second, the independently switched CCE must be decoded all the way to the time domain (i.e. including the synthesis filterbank) before it is scaled and added onto the various SCE and CPE channels that it is coupled to in the case that window state does not match.

A dependently switched CCE, on the other hand, must have a window state that matches all of the target SCE and CPE channels that it is coupled onto as determined by the list of cc_l and cc_r elements. In this case, the CCE only needs to be decoded as far as the frequency domain and then scaled as directed by the gain list before it is added to the target SCE or CPE channels.

The following pseudo code in function decode_ coupling_channel() defines the decoding operation for a dependently switched coupling channel element. First the spectral coefficients of the embedded single_channel_element are decoded into an internal buffer. Since the gain elements for the first coupled target (list_index == 0) are not transmitted, all gain_element values associated with this target are assumed to be 0, i.e.

the coupling channel is added to the coupled target channel in its natural scaling. Otherwise the spectral coefficients are scaled and added to the coefficients of the coupled target channels using the appropriate list of gain_element values.

An independently switched CCE is decoded like a dependently switched CCE having only common_gain_element's. However, the resulting scaled spectrum is transformed back into its time representation and then coupled in the time domain.

Please note that the gain_element lists may be shared between the left and the right channel of a target channel pair element. This is signalled by both cc_l and cc_r being zero as indicated in the table below:

cc_l,	cc_r	shared gain list present	left gain list present	right gain list present
0,	0	yes	no	no
0,	1	no	no	yes
1,	0	no	yes	no
1,	1	no	yes	yes

```
decode_coupling_channel()
{
   - decode spectral coefficients of embedded single_channel_element
     into buffer "cc_spectrum[]".

   /* Couple spectral coefficients onto target channels */
   list_index = 0;
   for (c=0; c<num_coupled_elements+1; c++)  {
      if (!cc_target_is_cpe[c])  {
         couple_channel( cc_spectrum,
                         spectrum_m( cc_target_tag_select[c], cc_domain ),
                         list_index++ );
      }
      if (cc_target_is_cpe[c])  {
         if (!cc_l[c]  &&  !cc_r[c])  {
            couple_channel( cc_spectrum,
                            spectrum_l( cc_target_tag_select[c], cc_domain ),
                            list_index );
            couple_channel( cc_spectrum,
                            spectrum_r( cc_target_tag_select[c], cc_domain ),
                            list_index++ );
         }
         if (cc_l[c])  {
            couple_channel( cc_spectrum,
                            spectrum_l( cc_target_tag_select[c], cc_domain ),
                            list_index++ ) );
         }
         if (cc_r[c])  {
            couple_channel( cc_spectrum,
                            spectrum_r( cc_target_tag_select[c], cc_domain ),
                            list_index++ ) );
         }
      }
   }
}


couple_channel( source_spectrum[], dest_spectrum[], gain_list_index )
{
   idx = gain_list_index;
   a = 0;
   cc_scale = cc_scale_table[gain_element_scale];
   for (g=0; g<num_window_groups; g++)  {

      /* Decode coupling gain elements for this group */
      if (common_gain_element_present[idx])  {

         for (sfb=0; sfb<max_sfb; sfb++)  {
            cc_sign[idx][g][sfb] = 1;
            gain_element[idx][g][sfb] = common_gain_element[idx];
```

```
        }

     }  else  {

        for (sfb=0; sfb<max_sfb; sfb++)  {
           if ( sfb_cb[g][sfb] == ZERO_HCB )
              continue;

           if (gain_element_sign)  {
              cc_sign[idx][g][sfb] =
                 1 - 2*(dpcm_gain_element[idx][g][sfb] & 0x1);
              gain_element[idx][g][sfb] =
                 a += (dpcm_gain_element[idx][g][sfb] >> 1);
           }
           else  {
              cc_sign[idx][g][sfb] = 1;
              gain_element[idx][g][sfb] =
                 a += dpcm_gain_element[idx][g][sfb];
           }
        }

     }


     /* Do coupling onto target channels */
     for (b=0; b<window_group_length[b]; b++)  {
        for (sfb=0; sfb<max_sfb; sfb++)  {

           if ( sfb_cb[g][sfb] != ZERO_HCB )  {
              cc_gain[idx][g][sfb] =
                 cc_sign[idx][g][sfb] * cc_scale^gain_element[idx][g][sfb];

              for (i=0; i<swb_offset[sfb+1]-swb_offset[sfb]; i++)
                 dest_spectrum[g][b][sfb][i] +=
                    cc_gain[idx][g][sfb] * source_spectrum[g][b][sfb][i];

           }

        }
     }

  }
}
```

Note: The array sfb_cb represents the codebook data respect to the CCE's embedded single_channel_element (not the coupled target channel).

3.7.3.4 Tables

Table 0.1 – Scaling resolution for channel coupling (cc_scale_table)

Value of "gain_element_scale"	Amplitude Resolution "cc_scale"	Stepsize [dB]
0	2^(1/8)	0.75
1	2^(1/4)	1.50
2	2^(1/2)	3.00
3	2^1	6.00

3.8 Temporal Noise Shaping (TNS)

(similar to ISO/IEC 13818-7)

3.8.1 Tool description

Temporal Noise Shaping is used to control the temporal shape of the quantization noise within each window of the transform. This is done by applying a filtering process to parts of the spectral data of each channel.
Note that this tool includes certain profile dependent parameters (see clause 2.1).

3.8.2 Definitions

n_filt[w]	number of noise shaping filters used for window w (see 1.3, Table 6.15)
coef_res[w]	token indicating the resolution of the transmitted filter coefficients for window w, switching between a resolution of 3 bits (0) and 4 bits (1) (see 1.3, Table 6.15)
length[w][filt]	length of the region to which one filter is applied in window w (in units of scalefactor bands) (see 1.3, Table 6.15)
order[w][filt]	order of one noise shaping filter applied to window w (see 1.3, Table 6.15).
direction[w][filt]	1 bit indicating whether the filter is applied in upward (0) or downward (1) direction (see 1.3, Table 6.15)
coef_compress[w][filt]	1 bit indicating whether the most significant bit of the coefficients of the noise shaping filter filt in window w are omitted from transmission (1) or not (0) (see 1.3, Table 6.15)
coef[w][filt][i]	coefficients of one noise shaping filter applied to window w (see 1.3, Table 6.15)
spec[w][k]	Array containing the spectrum for the window w of the channel being processed

Note: Depending on the window_sequence the size of the following bitstream fields is switched for each transform window according to its window size:

Name	Window with 128 spectral lines	Other window size
'n_filt'	1	2
'length'	4	6
'order'	3	5

3.8.3 Decoding Process

The decoding process for Temporal Noise Shaping is carried out separately on each window of the current frame by applying all-pole filtering to selected regions of the spectral coefficients (see function tns_decode_frame). The number of noise shaping filters applied to each window is specified by "n_filt". The target range of spectral coefficients is defined in units of scalefactor bands counting down "length" bands from the top band (or the bottom of the previous noise shaping band).
First the transmitted filter coefficients have to be decoded, i.e. conversion to signed numbers, inverse quantization, conversion to LPC coefficients as described in function tns_decode_coef().
Then the all-pole filters are applied to the target frequency regions of the channel's spectral coefficients (see function tns_ar_filter()). The token "direction" is used to determine the direction the filter is slid across the coefficients (0=upward, 1=downward).
The constant TNS_MAX_BANDS defines the maximum number of scalefactor bands to which Temporal Noise Shaping is applied. The maximum possible filter order is defined by the constant TNS_MAX_ORDER. Both constants are profile dependent parameters.

The decoding process for one channel can be described as follows pseudo code:

```
/* TNS decoding for one channel and frame */
tns_decode_frame()
{
   for (w=0; w<num_windows; w++) {

      bottom = num_swb;
      for (f=0; f<n_filt[w]; f++)  {

         top = bottom;
         bottom = max( top - length[w][f], 0 );
         tns_order = min( order[w][f], TNS_MAX_ORDER );
         if (!tns_order)  continue;

         tns_decode_coef( tns_order, coef_res[w]+3, coef_compress[w][f],
                          coef[w][f], lpc[] );

         start = swb_offset[min(bottom,TNS_MAX_BANDS,max_sfb)];
         end =   swb_offset[min(top,TNS_MAX_BANDS,max_sfb)];
         if ((size = end - start) <= 0)  continue;

         if (direction[w][f])  {
            inc = -1;   start = end - 1;
         }  else  {
```

```
            inc =  1;
         }

         tns_ar_filter( &spec[w][start], size, inc, lpc[], tns_order );

      }

   }
}


/* Decoder transmitted coefficients for one TNS filter */
tns_decode_coef( order, coef_res_bits, coef_compress, coef[], a[] )
{

   /* Some internal tables */
   sgn_mask[]  = {  0x2,  0x4,  0x8 };
   neg_mask[]  = { ~0x3, ~0x7, ~0xf };

                                          /* size used for transmission */
   coef_res2 = coef_res_bits - coef_compress;
   s_mask = sgn_mask[ coef_res2 - 2 ];     /* mask for sign bit */
   n_mask = neg_mask[ coef_res2 - 2 ];     /* mask for padding neg. values */

   /* Conversion to signed integer */
   for (i=0; i<order; i++)
      tmp[i] = (coef[i] & s_mask) ? (coef[i] | n_mask) : coef[i];

   /* Inverse quantization */
   iqfac   = ((1 << (coef_res_bits-1)) - 0.5) / (π/2.0);
   iqfac_m = ((1 << (coef_res_bits-1)) + 0.5) / (π/2.0);
   for (i=0; i<order; i++)  {
      tmp2[i] = sin( tmp[i] / ((tmp[i] >= 0) ? iqfac : iqfac_m) );
   }

   /* Conversion to LPC coefficients */
   a[0] = 1;
   for (m=1; m<=order; m++)  {
      a[m] = tmp2[m-1];
      for (i=1; i<m; i++)  {
         b[i] = a[i] + tmp2[m-1] * a[m-i];
      }
      for (i=0; i<m; i++)  {
         a[i] = b[i];
      }
   }

}

tns_ar_filter( spectrum[], size, inc, lpc[], order )
{
   - Simple all-pole filter of order "order" defined by
     y(n) =  x(n) - lpc[1]*y(n-1) - ... - lpc[order]*y(n-order)

   - The state variables of the filter are initialized to zero every time

   - The output data is written over the input data ("in-place operation")

   - An input vector of "size" samples is processed and the index increment
     to the next data sample is given by "inc"
}
```

3.8.4 TNS in the scalable coder

Decoding of a scaleable core based AAC bitstream with one ore more Mono AAC Layer and one or more Stereo AAC Layer. Basically it is possible that either the core or the Mono AAC Layer(s) or the AAC stereo Layer(s) are omitted .

Case 1 Mono core + Mono AAC

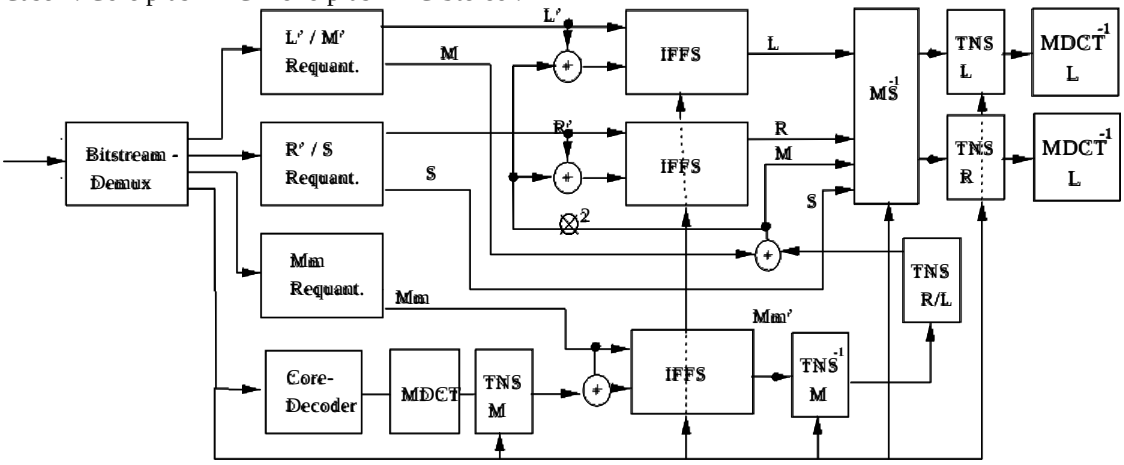
If TNS is used in a scalable coder with a core coder, the TNS encoder filters have to be applied to the output of the MDCT which is employed to generate the spectrum of the core coder. These encoder filters use the LPC coefficients already decoded for the corresponding TNS decoder filters. The filters are slid across the specified target frequency range exactly the way described for the decoder filter. The difference between decoder and encoder filtering is that each all-pole (auto-regressive) decoder filter used for TNS decoding is replaced by its inverse (all-zero, moving average) filter.

The filter equation is :

```
y[n] = x[n] + lpc[1] * x[n-1] + ... + lpc[order] * x[n-order]
```

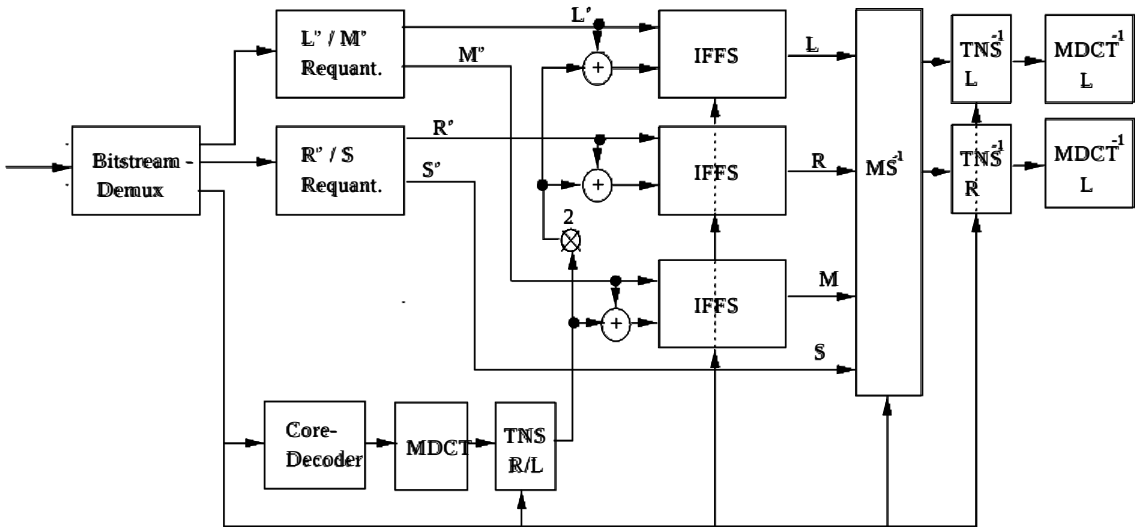
The number of filters, filtering direction etc. is controlled exactly like in the decoding process.

Case 2: Core plus AAC mono plus AAC Stereo :



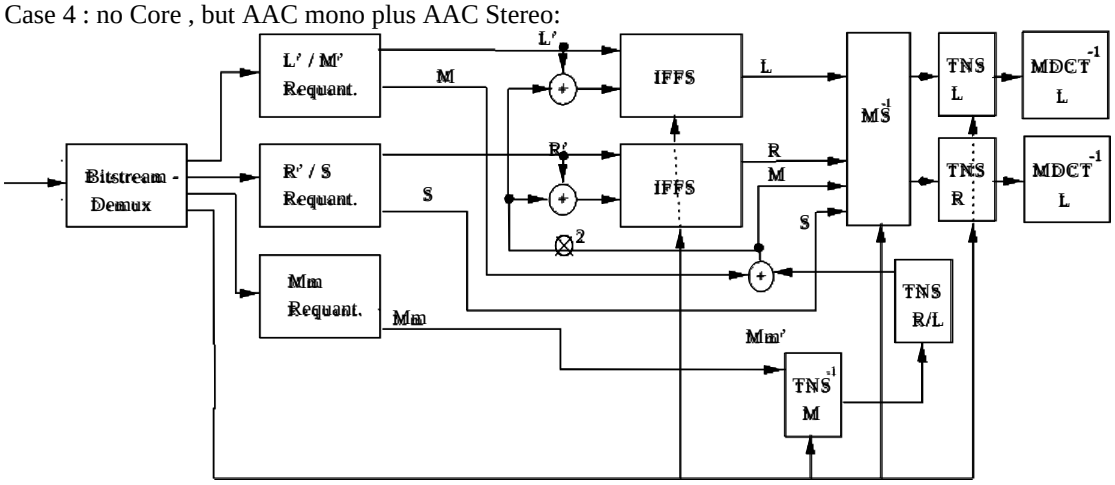
Here there are 3 sets of TNS coefficients transmitted : one set in the first AAC mono bitstream , and two (one for each channel) in the first AAC stereo Layer. The TNS filter coefficients of the mono layer are applied to the output of the MDCT for the Core, this spectrum is added to the requantized spectrum of the last Mono depending of the **diff_control** flags, then the inverse TNS filter is applied with the coefficients of the first Mono AAC Layer. After that a forward TNS filter is calculated with the TNS coefficients of ether the left or the right channel of the first stereo AAC layer depending on the bit **tns_channel_mono_layer.** The output is then added/omitted to either the requantized spectrum of the left or right channel of the last stereo AAC Layer depending on the **ms_used** bits and depending on the **diff_control_lr** bits. Then the invers MS matrix is calculated depending on the ms_usecd bits. The output is two reconstructed MDCT spectras, one for the left and one the right channel. Both are then feed into there inverse TNS filter one for each channel , the TNS coefficients are transmitted in the first AAC stereo layer.

Case 3 : Core plus AAC Stereo:



Here there are 2 sets of TNS coefficients transmitted : one for each channel in the first AAC stereo Layer. After decoding and upsampling of the Core layer a MDCT is calculated. Then a forward TNS filter is calculated with the TNS coefficients of ether the left or the right channel of the first stereo AAC layer depending on the bit **tns_channel_mono_layer .** After that a the output is then added/omitted to the requantized spectrum of either the left or right channel of the last stereo AAC Layer depending on the **ms_used** bits and depending on the **diff_control_lr** . Then the invers MS matrix is calculated depending on the ms_usecd bits. The output is two reconstructed MDCT spectras, one for the left and one the right channel. Both are then feed into there inverse TNS filter one for each channel , the TNS coefficients are transmitted in the first AAC stereo layer.

Case 4 : no Core , but AAC mono plus AAC Stereo:



Here there are 3 sets of TNS coefficients transmitted : one set in the first AAC mono bitstream , and two (one for each channel) in the first AAC stereo Layer. The inverse TNS filter is applied to the output spectrum of the last mono layer with the coefficients of the first Mono AAC Layer. After that a forward TNS filter is calculated with the TNS coefficients of ether the left or the right channel of the first stereo AAC layer depending on the bit **tns_channel_mono_layer .** The output is then added/omitted to either the requantized spectrum of the left or right channel of the last stereo AAC Layer depending on the **ms_used** bits and depending on the **diff_control_lr**. Then the invers MS matrix is calculated depending on the **ms_used** bits. The output is two reconstructed MDCT spectras, one for the left and one the right channel. Both are then feed into there inverse TNS filter one for each channel , the TNS coefficients are transmitted in the first AAC stereo layer.

3.9 Spectrum Normalization

3.9.1 Tool Description

In the TwinVQ decoder, spectral de-normalization is used in combination with inverse vector quantization of the MDCT coefficients, whose reproduction has globally flat shape. Using this tool, the spectral envelope is regenerated by decoding gain, Bark-scale envelope and an envelope specified with LPC parameters. Bark-scale envelope is reconstructed using a vector quantization decoder. LPC coefficients are quantized in LSP domain by means of 2-stage split vector quantization with moving average interframe prediction. Decoded LSP coefficients are directly used for generating an amplitude spectrum (square root of the power spectral envelope).

In a long MDCT block size mode, periodic peak components are optionally added to the flattened MDCT coefficients for low rate coder.

3.9.2 Definitions

α:	MA prediction coefficients used for LSP quantization
alfq[][] :	Predictive coefficients for Bark-envelope
AMP_MAX:	maximum value of mu-law quantizer for global gain
AMP_NM:	normalization factor for global gain
BAND_UPPER:	implicit bandwidth
BASF_STEP:	step size of base frequency quantizer for periodic peak components coding
bfreq[]:	base frequency of periodic peak components
blim_h[]:	bandwidth control factor (higher part)
blim_l[]:	bandwidth control factor (lower part)
BLIM_STEP_H:	number of steps of bandwidth control quantization (higher part)
BLIM_STEP_L:	number of steps of bandwidth control quantization (lower part)
CUT_M_H:	minimum bandwidth ratio (higher part)
CUT_M_L:	maximum bandwidth ratio (lower part)
cv_env[][]:	code vectors of envelope codebook
env[][][]:	Bark-scale envelope projected onto Bark-scale frequency axis
f b_shift[][]	Syntax element indicating the base frequency of active frequency band of the adaptive bandwidth control.
FW_ALF_STEP:	MA prediction coefficient for quantization of Bark-scale envelope
FW_CB_LEN:	length of code vector of Bark-scale envelop codebook
FW_N_DIV:	number of interleave division of Bark-scale envelope vector quantization
gain_p[][]:	gain factors of the periodic peak components
gain[][]:	gain factors of MDCT coefficients
global_gain[]:	global gain of MDCT coefficients normalized by AMP_NM
index_blim_h[]:	syntax element indicating higher part bandwidth control
index_blim_l[]:	syntax element indicating lower part of bandwidth control
index_env[][][]:	syntax elements indicating Bark-scale envelope elements
index_fw_alf[]:	syntax element indicating the MA prediction switch of Bark-scale envelope quantization
index_gain[][]:	syntax elements indicating global gain of MDCT coefficients
index_gain_sb[][][]:	syntax elements indicating subblock gain of MDCT coefficients
index_lsp0[][]:	syntax elements indicating MA prediction coefficients used for LSP quantization
index_lsp1[]:	syntax element indicating the first-stage LSP quantization
index_lsp2[][]:	syntax element indicating the second-stage LSP quantization
index_pgain[]:	syntax elements indicating gain of periodic peak components
index_pit[][]:	syntax elements indicating base frequency of periodic peak components
index_shape0_p[]:	syntax element indicating peak elements quantization index for shape vector of conjugate channel 0
index_shape1_p[]:	periodic peak elements quantization index for shape vector of conjugate channel 1
isp[]:	split point table for second-stage LSP quantization
lengthp[]:	lengths of code vectors for periodic peak components quantization
lnenv[][]	Bark-scale envelope projected onto linear-scale frequency axis
LOWER_BOUNDARY[][]:	lower boundary of active frequency band used in scaleable layers
lpenv[][]:	LPC spectral envelope

lsp[][]: LPC coefficients which range is set from zero to π.
lyr: indicates enhancement layer number. Number 0 is assigned for the base layer.
LSP_SPLIT: number of splits of 2nd-stage vector quantization for LSP coding
MU: mu factor for mu-law quantization for gain
N_CRB: number of subbands for Bark-scale envelope coding
N_DIV_P: number of interleave division for periodic peak components coding
N_FR: number of samples in a subframe
N_FR_P: number of elements of periodic peak components
N_SF: number of subframe in a frame. The value is set in section 3.4.3.
p_cv_env[][]: Bark-scale envelope vector reconstructed in the previous frame
pit[]: periodic peak components
pit_seq[][][]: periodic peak components projected into linear scale
PIT_CB_SIZE: size of codebook for periodic peak components quantization
PGAIN_MAX: maximum value of mu-law quantizer for gain of periodic peak components
PGAIN_MU: mu factor for mu-law quantization for periodic peak components
PGAIN_STEP: step size of mu-law quantizer for gain of periodic peak components
pol0_p: polarity of conjugate channel 0 for periodic peak components quantization
pol1_p polarity of conjugate channel 1 for periodic peak components quantization
sp_cv0[]: reconstructed shape of conjugate channel 0 for periodic peak components quantization
sp_cv1[]: reconstructed shape of conjugate channel 1 for periodic peak components quantization
STEP: step size of mu-law quantizer for global gain
SUB_AMP_MAX: maximum amplitude of mu-law quantizer for subframe gain ratio
SUB_AMP_NM: normalization factor for subframe gain ratio
SUB_STEP: step size of mu-law quantizer for subframe gain
subg_ratio[]: subframe gain ratio
UPPER_BOUNDARY[][]:
upper boundary of active frequency band used in scaleable layers
v[]: LSP coefficients
*v*1[]: reconstructed vector from 1st-stage VQ for LSP coding
*v*2[]: reconstructed vector from 2nd-stage VQ for LSP coding

x_flat[]: normalized MDCT coefficients (input)
spec[][][]: de-normalized MDCT coefficients (output)

3.9.3 Decoding process

The decoding process consists of five parts: gain decoding, Bark-scale envelope decoding, periodic peak components decoding, LPC spectrum decoding, and inverse normalization (see Fig. 3.9.1).

3.9.3.1 Initializations

Befor starting any process, prediction memories p_cv_env[][] and $\alpha_i^{(j)}$ are cleared.

3.9.3.2 Gain decoding

In the first step of gain decoding, the global gain is decoded using μ-law inverse quantizer described as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   g_temp = index_gain * STEP + STEP / 2
   global_gain =
      (AMP_MAX * (exp10(g_temp * log10(1.+MU) / AMP_MAX) -1) / MU) / AMP_NM;
}
```

Next, subband gain ratios are decoded using mu-law inverse quantizer described as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   if (N_SF > 1){
      for(isf=0; isf<N_SF; isf++){
         g_temp = index_gain_sb[i_ch][isf+1] * SUB_STEP +SUB_STEP/2.;
         subg_ratio[isf] =
           (SUB_AMP_MAX*(exp10(g_temp*log10(1.+MU)/SUB_AMP_MAX)-1)/MU)
            / SUB_AMP_NM;
      }
   }
   else{
      subg_ratio[i_ch][0] = 1
   }
}
```

Finally, gain factors are reconstructed as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   for(isf=0; isf<N_SF; isf++){
      gain[i_ch][isf] = global_gain[i_ch] * subg_ratio[i_ch][isf] / SUB_AMP_NM;
   }
}
```

3.9.3.3 Decoding of periodic peak components

Periodic peak components are optionally added to the input coefficients. The periodic peak components are coded using vector quantization. This process is active when the parameter ppc_present is set to TRUE. Otherwise, all the elements of output array, pit_seq[] are set to zero and the process is skipped.

This process works when the block length type is LONG and the configuration mode is 16_16 of 08_06.

3.9.3.3.1 Decoding of polarity

```
for (idiv=0; idiv<N_DIV_P; idiv++){
   pol0[idiv] = 2*(index_shape0_p[idiv] / PIT_CB_SIZE) - 1
   pol1[idiv] = 2*(index_shape1_p[idiv] / PIT_CB_SIZE) - 1
}
```

3.9.3.3.2 Decoding of shape code

```
for (idiv=0; idiv<N_DIV_P; idiv++){
   index0[idiv] = index_shape0_p[idiv] % PIT_CB_SIZE
   index1[idiv] = index_shape1_p[idiv] % PIT_CB_SIZE
}
```

3.9.3.3.3 Decoding gain of periodic peak components

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   temp = index_pgain[i_ch] * PGAIN_STEP + PGAIN_STEP / 2
   gain_p[i_ch] =
      (PGAIN_MAX*(exp10(temp*log10(1.+PGAIN_MU)/PGAIN_MAX)-1)/PGAIN_MU);
}
```

3.9.3.3.4 Reconstruction of periodic peak components

There are two steps of procedures. First the lengths of code vectors for periodic peak components, lengthp[], are calculated. Then, the periodic peak components pit[] are calculated.

```
for (idiv=0; idiv<N_DIV_P; idiv++){
   lengthp[idiv] = (N_FR_P*N_CH+N_DIV_P-1-idiv) / N_DIV_P;
}

for (idiv=0; idiv<N_DIV_P; idiv++){
   for (icv=0; icv<lengthp[idiv]; icv++){
      ismp = idiv + icv * N_DIV_P
      pit[ismp] = gain_p * (pol0[idiv]*pit_cv0[index0[idiv][icv]] \\
                    + pol1[idiv]*pit_cv1[index1[idiv]][icv]) / 2
   }
}
```

3.9.3.3.5 Projecting periodic peak components into linear scale

First, parameters are calculated as following:

```
fcmin = log2((N_FR/SAMPF)*0.2);
fcmax = log2((N_FR/SAMPF)*2.4);
```

If bitrate mode is 16 kbit/s, then
`bandwidth =2.`
If bitrate mode is 6 kbit/s (8kHz sampling), then
`bandwidth = 1.5`

where fcmin is the minimum frequency to be quantization expressed in log scale, fcmax is the maximum.

Next, base frequency of the periodic peak components is decoded according to the following procedure:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   dtmp = (double)index_pit[i_ch]/ (double) BASF_STEP;
   dtmp = dtmp * (fcmax-fcmin) + fcmin;
   bfreq[i_ch] = (double)pow2(dtmp);
}
```

Before projecting the perodic peak components into linear scale, all the elements of target array pit_seq[][] is set to zero:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   for (ismp=0; ismp<N_FR; ismp++){
      pit_seq[i_ch][ismp] = 0.;;
   }
}
```

Then, reconstructed pereodic peak components are projected to linear-scale as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   npcount = (int)( N_FR_P*bandwidth/(N_FR/bfreq[i_ch]));
   iscount=0;
   for (jj=0; jj<npcount/2; jj++){
          pit_seq[i_ch][jj] = pit[jj+i_ch*N_FR_P];
          iscount ++;
   }
   for (ii=0; ii<(N_FR_P )&& (iscount<N_FR_P); ii++){
      i_smp = (int)(bfreq[i_ch]*(ii+1)+0.5);
      for (jj=-npcount/2; jj<(npcount-1)/2+1; jj++){
         pit_seq[i_ch][i_smp+jj] = pit[iscount+i_ch*N_FR_P];
         iscount ++;
         if(iscount >= N_FR_P) break;
      }
   }
}
```

In case of skipping the periodic peak components decoding process, all the elements of pitch component array pit_seq[][] are set to zero.

3.9.3.4 Decoding of Bark-scale envelope

The Bark-scale envelope is decoded in each subframe. There are two procedure stages: inverse quantizing of the envelope vectors env[][] and projecting the bark-scale envelopes env[][] onto the linear scale envelopes lnenv[][].

3.9.3.4.1 Inverse quantization of envelope vector

The inverse quantization part is illustrated in Fig. 3.9.2.

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   for (isf=0; isf<N_SF; isf++){
      alfq[i_ch][isf] = index_fw_alf[i_ch][isf] * FW_ALF_STEP
      for (ifdiv=0; ifdiv<FW_N_DIV; ifdiv++){
         for (icv=0; icv<FW_CB_LEN; icv++){
            ienv = FW_N_DIV * icv + ifdiv
            dtmp = cv_env[index_env[ich][isf][ifdiv]][icv]
            env[i_ch][isf][ienv] = dtmp + alfq[i_ch][isf] * p_cv_env[i_ch][icv] + 1
            p_cv_env[i_ch][icv] = dtmp
         }
      }
   }
}
```

The cv_env[][] is the Bark-scale envelope codebook.

3.9.3.4.2 Projecting the Bark-scale envelope onto a linear scale

The envelopes env[][][] are expressed using the Bark scale on the frequency axis. The de-normalization procedure requires a linear-scale envelopes.

Befor the projecting process, the boundary table of the bark-scale subband, crb_tbl[], is determined. If the scaleable layer number lyr is equal or less than 1, values of the Bark-scale subband table is assigned dependent on the window type and configuration mode as listed in the tables from 3.9.6 to 3.9.23. If lyr>1, values of the Bark-scale subband table are stored in the temporal memory, crb_tbl_tmp.

After the crb_tbl[] is determined, the projecting process is done as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   if (lyr>1){
      for (ienv=0; ienv<N_CRB; ienv++){
         crb_tbl[ienv] =
            crb_tbl_tmp[ienv]+LOWER_BOUNDARY[lyr][i_ch] - LOWER_BOUNDARY[2][i_ch];
         }
      }
   }
   for (isf=0; isf<N_SF; isf++){
      ismp=0
      for (ienv=0; ienv<N_CRB; ienv++){
         while (ismp<crb_tbl[ienv]){
            lnenv[i_ch][isf][ismp] = env[i_ch][isf][ienv]
            ismp++
         }
      }
   }
}
```

Values of the UPPER_BOUNDARY[][] and LOWER_BOUNDARY[][] are defined in section 3.4.4.4.

3.9.3.5 Decoding of LPC spectrum

The LPC spectrum is represented by LSP coefficients. In the decoding process, LSP coefficients are reconstructed first; then they are transformed into the LPC spectrum, which represents the square root of the power spectrum.

3.9.3.5.1 LSP coefficients decoding using the MA prediction

MA prediction coefficients are determined by referring to the coefficient table $\alpha\, t_i^{(j)}$. The rule is:

$$\alpha_i^{(j)}[i\_ch] = \alpha\, t_i^{(j)}[i\_ch](index\_lsp0[i\_ch]) \qquad \text{for } i = 1 \text{ to } N\_PR,\ j = 1 \text{ to } MA\_NP,\ i\_ch = 0 \text{ to } N\_CH - 1,$$

where i is LPC order and j is MA prediction order. The coefficient table $\alpha\, t_i^{(j)}$ is chosen according to the configuration mode among codebooks listed in tables C.48, C.51,
C.54, C.57, and C.60.

3.9.3.5.2 First-stage inverse quantization of LSP decoding

$$v1_i[i\_ch] = lspcode1_i(index\_lsp1[i\_ch]) \qquad \text{for } i = 1 \text{ to } N\_PR,\ i\_ch = 0 \text{ to } N\_CH - 1,$$

where lspcode1 is the first-stage LSP codebook chosen according to the configuration mode among codebooks listed in tables C.46, C.49, C.52 C.55, and C.58.

3.9.3.5.3 Second-stage inverse quantization of LSP decoding

$$v2_i[i\_ch] = lspcode2_i(index\_lsp2[i\_ch][k])$$
$$\text{for } k = 0 \text{ to } LSP\_SPLIT - 1,\ \ i = isp(k) + 1 \text{ to } isp(k+1) - 1,\ \ i\_ch = 0 \text{ to } N\_CH - 1,$$

where lspcode2 is the second-stage LSP codebook chosen according to the configuration mode among codebooks listed in tables C.47, C.50, C.53, C.56, and C.59. Values of isp(k) are assigned dependent on bitrate modes as listed in the tables from 3.9.24 to 3.9.26.

3.9.3.5.4 Reconstruction of LSP coefficients

LSP coefficients lsp[][] are calculated as follows:

$$\vec{v}[i\_ch] = \vec{v1}[i\_ch] + \vec{v2}[i\_ch], \qquad \text{for } i\_ch = 0 \text{ to } N\_CH - 1$$

$$\alpha_i^{(0)}[i\_ch] = 1 - \sum_{j=1}^{MA\_NP} \alpha_i^{(j)}[i\_ch] \qquad \text{for } i = 1 \text{ to } N\_PR,\ i\_ch = 0 \text{ to } N\_CH - 1$$

$$lsp[i\_ch][i] = \sum_{j=0}^{MA\_NP} \alpha_i^{(j)}[i\_ch] \cdot v_i^{(-j)}[i\_ch] \qquad \text{for } i = 1 \text{ to } N\_PR,\ i\_ch = 0 \text{ to } N\_CH - 1$$

$$\vec{v}^{(j-1)}[i\_ch] = \vec{v}^{(j)}[i\_ch] \qquad \text{for } j = -MA\_NP - 1 \text{ to } 0,\ i\_ch = 0 \text{ to } N\_CH - 1$$

3.9.3.5.5 Transforming LSP parameters into LPC spectrum

The LPC spectrum corresponding to ii-th MDCT coefficient, lpenv[][] is defind as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   for (ii=1; ii<=N_FR-1; ii++){
        for (i=2, P[i_ch]=1.0; i<=N_PR; i+=2)
        P[i_ch] *= (cos(PI*ii/N_FR)-cos(lsp[i]))^2;
      for (i=1, Q[i_ch]=1.0; i<=N_PR; i+=2)
        Q[i_ch] *= (cos(PI*ii/N_FR)-cos(lsp[i]))^2;
      lpenv[ii] = 1/((1-cos(PI*ii/N_FR))*P[i_ch] + (1+cos(PI*ii/N_FR))*Q[i_ch]);
   }
}
```

If this tool is used as an element of scalable coder, LPC spectrum is squeezed into the active frequency band:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   nfr_lu = UPPER_BOUNDARY[lyr][i_ch] - LOWER_BOUNDARY[lyr][i_ch];
   for (ismp=0; ismp<LOWER_BOUNDARY[lyr][i_ch]; ismp++){
      lpenv_tmp[i_ch][ismp] = 0;
   }
   for(ismp=0; ismp<nfr_lu; ismp++){
      lpenv_tmp[i_ch][ismp+LOWER_BOUNDARY[lyr][i_ch]] =
         lpenv[i_ch][(int)(ismp*N_FR*/(ac_top - ac_btm))];
   }
   for (ismp=UPPER_BOUNDARY; ismp<N_FR; ismp++){
      lpenv_tmp[i_ch][ismp] = 0;
   }
   for(ismp=0; ismp<N_FR; ismp++){
      lpenv[i_ch][ismp] = lpenv_tmp[i_ch][ismp];
   }
}
```

The values of UPPER_BOUNDARY, LOWER_BOUNDARY, ac_top and ac_btm are defined in section 3.4.4.4.

3.9.3.6 Inverse normalization

Input coefficients x_flat[] are applied to inverse normalization according to the following procedure, and output coefficients spec[][][] are created.

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   for (isf=0; isf<N_SF; isf++){
      for (ismp=0; ismp<N_FR; ismp++){
         spec[isf][ismp] =
          (x_flat[ismp+(isf+i_ch*N_SF)*N_FR]*lpenv[i_ch][ismp]*lnenv[i_ch][isf][ismp]
           + pit_seq[i_ch][ismp]) * gain[i_ch][isf];
      }
   }
}
```

3.9.3.7 Bandwidth control

This functionaliry is valid only in the compression configuration modes.

After the inverse normalization, upper and lower bands of output coefficients spec[][][] are set to zero.

In the bandwidth decoding modules, higher signal bandwidth ratio *blim_h*[] is decoded as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   blim_h[i_ch] =
      (1. - (1.-CUT_M_H) * (double)index_blim_h[i_ch]/(double)BLIM_STEP_H))
      * BAND_UPPER;
}
```

The blim_l[] is decoded as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   if (index_blim_l[i_ch] == 1) blim_l[i_ch] = CUT_M_L;
   else                         blim_l = 0;
}
```

In the bandwidth limitation module, higher and lower parts of MDCT coefficients are set to zero as follows:

```
for (i_ch=0; i_ch<N_CH; i_ch++){
   NbaseH = blim_h[i_ch] * N_FR
   NbaseL = blim_l[i_ch] * N_FR
   for (isf=0; isf<N_SF; isf++){
      for (ismp=NbaseH; ismp<N_FR; ismp++){
         spec[i_ch][isf][ismp] = 0
      }
      for (ismp=0; ismp<NbaseL; ismp++){
         spec[i_ch][isf][ismp] = 0
      }
   }
}
```

3.9.4 Diagrams

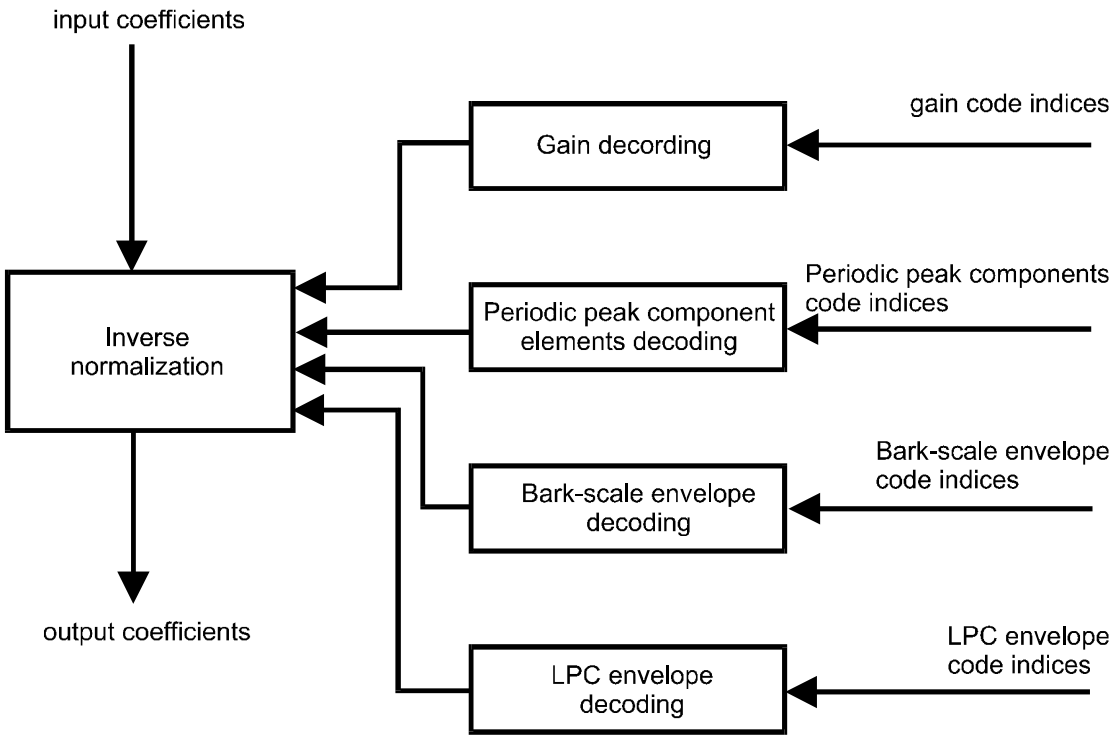


Figure 3.9. 1 — Decoding process of the TwinVQ

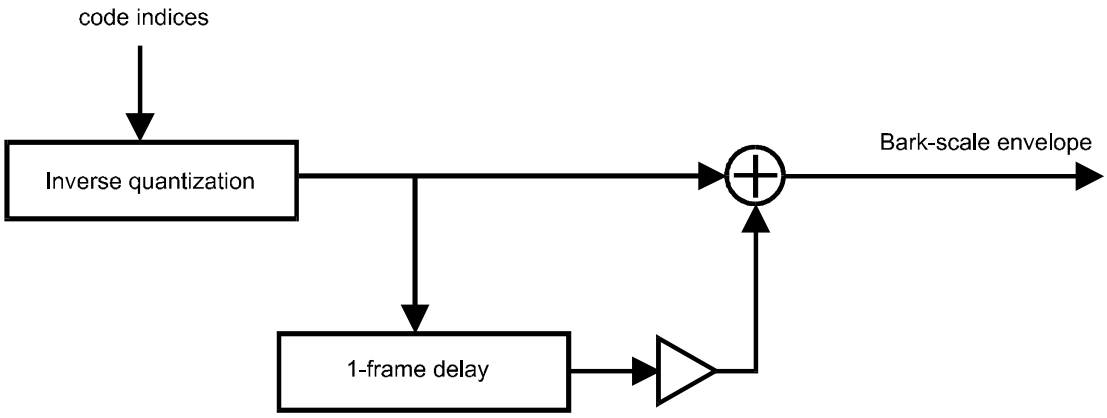


Figure 3.9. 2 — Reconstruction process of the Bark-scale envelope

3.9.5 Tables

Parameters used in the inverse spectrum normalization process are set as follows:

Table 3.9.1: Parameter table for the 24_06/24_06_960 configuration mode(Long-Short)

Parameter	Value	Meaning
Common:		
AMP_MAX	16000	Maximum amplitude in the mu-law coding of the frame gain
AMP_NM	1024	Normalized amplitude of flattened coefficients
BAND_UPPER	0.33333	Implicit bandwidth
BLIM_BITS_H	0	Bits for higher bandwidth control
BLIM_BITS_L	0	Bits for lower bandwidth control
BLIM_STEP_H	0	Number of steps of band limitation quantization
CUT_M_H	-	Minimum badwidth ratio (higher part)
CUT_M_L	-	Maximum bandwidth ratio (lower part)
FW_ALF_STEP	0.5	MA prediction coefficient for the envelope coding
GAIN_BIT	9	Number of bits for the frame gain coding
LSP_BIT0	1	Number of bits for prediction switch (LSP coding)
LSP_BIT1	6	Number of bits at the first stage (LSP coding)
LSP_BIT2	4	Number of bits per split VQ at the second stage (LSP coding)
LSP_SPLIT	3	Number of split at the second stage (LSP coding)
MA_NP	1	MA prediction order (LSP coding)
MU	100	Parameter mu in the mu-law coding of the frame power
N_PR	20	LPC order
NUM_STEP	512	Number of gain-quantization steps
STEP	AMP_MAX / (NUM_STEP-1) = 31.31	Step width of gain-quantization
SUB_AMP_MAX	4700	Maximum of subframe-gain to frame-gain ratio
SUB_AMP_NM	1024	Normalized subframe amplitude of flattened coefficients
SUB_GAIN_BIT	4	Number of bits for the subframe gain coding
SUB_NUM_STEP	16	Number of sub-gain-quantization steps
SUB_STEP	SUB_AMP_MAX/ (SUB_NUM_STEP-1)= 313.3	Step width of sub-gain-quantization
Long block:		
FW_CB_LEN	9	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	7	Number of division in the envelope coding
N_CRB	64	Numbef of bark-scale subbands

N_FR	1024/960	MDCT block size
PIT_N_BIT	0	Available bits for periodic peak components quantization
Short block:		
FW_CB_LEN	-	Envelope code vector length
FW_N_BIT	0	Envelope code bits
FW_N_DIV	-	Number of division in the envelope coding
N_CRB	-	Numbef of bark-scale subbands
N_FR	128/120	MDCT block size

Table 3.9.2: Parameter table for the SCL_1/SCL_1_960 configuration mode

Parameter	Value	Meaning
Common:		
AMP_MAX	8000	Maximum amplitude in the mu-law coding of the frame gain
AMP_NM	1024	Normalized amplitude of flattened coefficients
FW_ALF_STEP	0.5	MA prediction coefficient for the envelope coding
GAIN_BIT	8	Number of bits for the frame gain coding
LSP_BIT0	1	Number of bits for prediction switch (LSP coding)
LSP_BIT1	6	Number of bits at the first stage (LSP coding)
LSP_BIT2	4	Number of bits per split VQ at the second stage (LSP coding)
LSP_SPLIT	3	Number of split at the second stage (LSP coding)
MA_NP	1	MA prediction order (LSP coding)
MU	100	Parameter mu in the mu-law coding of the frame power
N_PR	20	LPC order
NUM_STEP	256	Number of gain-quantization steps
STEP	AMP_MAX / (NUM_STEP-1) = 31.37	Step width of gain-quantization
SUB_AMP_MAX	6000	Maximum of subframe-gain to frame-gain ratio
SUB_AMP_NM	1024	Normalized subframe amplitude of flattened coefficients
SUB_GAIN_BIT	4	Number of bits for the subframe gain coding
SUB_NUM_STEP	16	Number of sub-gain-quantization steps
SUB_STEP	SUB_AMP_MAX/ (SUB_NUM_STEP-1)= 400.0	Step width of sub-gain-quantization
Long block:		
FW_CB_LEN	8	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	8	Number of division in the envelope coding
N_CRB	64	Numbef of bark-scale subbands
N_FR	1024/960	MDCT block size
Short block:		
FW_CB_LEN	-	Envelope code vector length
FW_N_BIT	0	Envelope code bits
FW_N_DIV	-	Number of division in the envelope coding
N_CRB	-	Numbef of bark-scale subbands
N_FR	128/120	MDCT block size

Table 3.9.3: Parameter table for the SCL_2/SCL_2_960 configuration mode

Parameter	Value	Meaning
Common:		
AMP_MAX	8000	Maximum amplitude in the mu-law coding of the frame gain
AMP_NM	1024	Normalized amplitude of flattened coefficients
FW_ALF_STEP	0.5	MA prediction coefficient for the envelope coding
GAIN_BIT	7	Number of bits for the frame gain coding
LSP_BIT0	1	Number of bits for prediction switch (LSP coding)
LSP_BIT1	6	Number of bits at the first stage (LSP coding)
LSP_BIT2	4	Number of bits per split VQ at the second stage (LSP coding)

LSP_SPLIT	3	Number of split at the second stage (LSP coding)
MA_NP	1	MA prediction order (LSP coding)
MU	100	Parameter mu in the mu-law coding of the frame power
N_PR	20	LPC order
NUM_STEP	128	Number of gain-quantization steps
STEP	AMP_MAX / (NUM_STEP-1) = 62.99	Step width of gain-quantization
SUB_AMP_MAX	6000	Maximum of subframe-gain to frame-gain ratio
SUB_AMP_NM	1024	Normalized subframe amplitude of flattened coefficients
SUB_GAIN_BIT	4	Number of bits for the subframe gain coding
SUB_NUM_STEP	16	Number of sub-gain-quantization steps
SUB_STEP	SUB_AMP_MAX/ (SUB_NUM_STEP-1)= 400.0	Step width of sub-gain-quantization
Long block:		
FW_CB_LEN	8	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	8	Number of division in the envelope coding
N_CRB	64	Numbef of bark-scale subbands
N_FR	1024/960	MDCT block size
Short block:		
FW_CB_LEN	-	Envelope code vector length
FW_N_BIT	0	Envelope code bits
FW_N_DIV	0	Number of division in the envelope coding
N_CRB	0	Numbef of bark-scale subbands
N_FR	128/120	MDCT block size

Table 3.9.4: Parameter table for the 16_16 configuration mode

Parameter	Value	Meaning
Common:		
AMP_MAX	10000	Maximum amplitude in the mu-law coding of the frame gain
AMP_NM	1024	Normalized amplitude of flattened coefficients
BAND_UPPER	1.0	Implicit bandwidth
BLIM_BITS_H	2	Bits for higher bandwidth control
BLIM_BITS_L	1	Bits for lower bandwidth control
BLIM_STEP_H	3	Number of steps of band limitation quantization
CUT_M_H	0.7	Minimum bandwidth ratio (higher part)
CUT_M_L	0.005	Maximum bandwidth ratio (lower part)
FW_ALF_STEP	0.5	MA prediction coefficient for the envelope coding
GAIN_BIT	8	Number of bits for the frame gain coding
LSP_BIT0	1	Number of bits for prediction switch (LSP coding)
LSP_BIT1	6	Number of bits at the first stage (LSP coding)
LSP_BIT2	4	Number of bits per split VQ at the second stage (LSP coding)
LSP_SPLIT	3	Number of split at the second stage (LSP coding)
MA_NP	1	MA prediction order (LSP coding)
MU	100	Parameter mu in the mu-law coding of the frame power
N_PR	16	LPC order
NUM_STEP	256	Number of gain-quantization steps
SAMPF	16000.0	Sampling frequency of input audio signal
STEP	AMP_MAX / (NUM_STEP-1) = 39.1	Step width of gain-quantization
SUB_AMP_MAX	4500	Maximum of subframe-gain to frame-gain ratio
SUB_AMP_NM	1024	Normalized subframe amplitude of flattened coefficients
SUB_GAIN_BIT	5	Number of bits for the subframe gain coding
SUB_NUM_STEP	32	Number of sub-gain-quantization steps
SUB_STEP	SUB_AMP_MAX/ (SUB_NUM_STEP-1)= 142.8	Step width of sub-gain-quantization
Long block:		

FW_CB_LEN	7	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	3	Number of division in the envelope coding
N_CRB	21	Numbef of bark-scale subbands
N_FR	512	MDCT block size
PIT_N_BIT	28	Available bits for periodic peak components quantization
N_DIV_P	2	Number of division in the periodic peak components quantization
PIT_CB_SIZE	64	Size of codebook for periodic peak components quantization
PGAIN_MAX	20000.	maximum value of mu-law quantizer for gain of periodic peak components
PGAIN_MU	200	mu factor for mu-law quantization for periodic peak components
PGAIN_BITS	7	coding bits for gain of periodic peak components
PGAIN_STEP	PGAIN_MAX/ ((1<<PGAIN_BITS)-1)	step size of mu-law quantizer for gain of periodic peak components
BASF_BITS	9	bits for base frequncy coding in PPC coding
N_FR_P	30	number of elements of periodic peak components
Medium block:		
FW_CB_LEN	10	Envelope code vector length
FW_N_BIT	5	Envelope code bits
FW_N_DIV	2	Number of division in the envelope coding
N_CRB	20	Numbef of bark-scale subbands
N_FR	256	MDCT block size
Short block:		
FW_CB_LEN	10	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	1	Number of division in the envelope coding
N_CRB	10	Numbef of bark-scale subbands
N_FR	64	MDCT block size

Table 3.9.5: Parameter table for the 08_06 configuration mode

Parameter	Value	Meaning
Common:		
AMP_MAX	10000	Maximum amplitude in the mu-law coding of the frame gain
AMP_NM	1024	Normalized amplitude of flattened coefficients
BAND_UPPER	1.0	Implicit bandwidth
BLIM_BITS_H	0	Bits for higher bandwidth control
BLIM_BITS_L	0	Bits for lower bandwidth control
BLIM_STEP_H	0	Number of steps of band limitation quantization
CUT_M_H	-	Minimum badwidth ratio (higher part)
CUT_M_L	-	Maximum bandwidth ratio (lower part)
FW_ALF_STEP	0.5	MA prediction coefficient for the envelope coding
GAIN_BIT	8	Number of bits for the frame gain coding
LSP_BIT0	1	Number of bits for prediction switch (LSP coding)
LSP_BIT1	5	Number of bits at the first stage (LSP coding)
LSP_BIT2	3	Number of bits per split VQ at the second stage (LSP coding)
LSP_SPLIT	3	Number of split at the second stage (LSP coding)
MA_NP	1	MA prediction order (LSP coding)
MU	100	Parameter mu in the mu-law coding of the frame power
N_PR	12	LPC order
NUM_STEP	256	Number of gain-quantization steps
STEP	AMP_MAX / (NUM_STEP-1) = 39.1	Step width of gain-quantization
SUB_AMP_MAX	4700	Maximum of subframe-gain to frame-gain ratio
SUB_AMP_NM	1024	Normalized subframe amplitude of flattened coefficients
SUB_GAIN_BIT	4	Number of bits for the subframe gain coding
SUB_NUM_STEP	16	Number of sub-gain-quantization steps
SUB_STEP	SUB_AMP_MAX/	Step width of sub-gain-quantization

	(SUB_NUM_STEP-1)= 313.3	
Long block:		
FW_CB_LEN	10	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	3	Number of division in the envelope coding
N_CRB	30	Numbef of bark-scale subbands
N_FR	512	MDCT block size
PIT_N_BIT	28	Available bits for periodic peak components quantization
N_DIV_P	2	Number of division in the periodic peak components quantization
PIT_CB_SIZE	64	Size of codebook for periodic peak components quantization
PGAIN_MAX	20000.	maximum value of mu-law quantizer for gain of periodic peak components
PGAIN_MU	200	mu factor for mu-law quantization for periodic peak components
PGAIN_BITS	6	coding bits for gain of periodic peak components
PGAIN_STEP	PGAIN_MAX/ ((1<<PGAIN_BITS)-1	step size of mu-law quantizer for gain of periodic peak components
BASF_BITS	8	bits for base frequncy coding in PPC coding
N_FR_P	20	number of elements of periodic peak components
Medium block:		
FW_CB_LEN	10	Envelope code vector length
FW_N_BIT	6	Envelope code bits
FW_N_DIV	20	Number of division in the envelope coding
N_CRB	16	Numbef of bark-scale subbands
N_FR	256	MDCT block size
Short block:		
FW_CB_LEN	10	Envelope code vector length
FW_N_BIT	3	Envelope code bits
FW_N_DIV	1	Number of division in the envelope coding
N_CRB	12	Numbef of bark-scale subbands
N_FR	64	MDCT block size

Table 3.9.6: Bark-scale subband table for 24_06 configuration mode (LONG:1024)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	3	11	83
1	8	12	99
2	13	13	119
3	18	14	145
4	24	15	180
5	30	16	228
6	36	17	298
7	43	18	406
8	51	19	596
9	60	20	1024
10	71		

Table 3.9.7: Bark-scale subband table for 24_06_960 configuration mode (LONG:960)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	3	11	83
1	8	12	99
2	13	13	119
3	18	14	145
4	24	15	180
5	30	16	228
6	36	17	298
7	43	18	406
8	51	19	596

9	60	20	960
10	71		

Table 3.9.8: Bark-scale subband table for 24_06 configuration mode (MIDIUM:256)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	2	8	21
1	3	9	26
2	5	10	33
3	7	11	43
4	9	12	58
5	11	13	83
6	13	14	131
7	17	15	256

Table 3.9.9: Bark-scale subband table for 24_06 configuration mode (MIDIUM:240)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	2	8	21
1	3	9	26
2	5	10	33
3	7	11	43
4	9	12	58
5	11	13	83
6	13	14	131
7	17	15	240

Table 3.9.10: Bark-scale subband table for 24_06 configuration mode (SHORT:128)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	2	6	14
1	4	7	16
2	6	8	22
3	8	9	36
4	10	10	60
5	12	11	128

Table 3.9.11: Bark-scale subband table for 24_06_960 configuration mode (SHORT:120)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	2	6	14
1	4	7	16
2	6	8	22
3	8	9	36
4	10	10	60
5	12	11	120

Table 3.9.12: Bark-scale subband table for 24_06 configuration mode (SHORT:64)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	1	6	7
1	2	7	8
2	3	8	11
3	4	9	18
4	5	10	30
5	6	11	64

Table 3.9.13: Bark-scale subband table for 24_06_960 configuration mode (SHORT:60)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	1	6	7
1	2	7	8
2	3	8	11
3	4	9	18
4	5	10	30
5	6	11	60

Table 3.9.14: Bark-scale subband table for SCL_1 configuration mode (LONG:1024)

isf	crb_tbl[isf]	isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	3	22	70	44	201
1	5	23	73	45	212
2	8	24	77	46	223
3	11	25	81	47	235
4	14	26	85	48	248
5	16	27	90	49	262
6	19	28	94	50	278
7	22	29	99	51	294
8	25	30	104	52	312
9	28	31	108	53	332
10	31	32	114	54	353
11	34	33	119	55	376
12	37	34	125	56	402
13	40	35	131	57	430
14	43	36	137	58	462
15	46	37	143	59	496
16	49	38	150	60	535
17	52	39	158	61	578
18	56	40	165	62	627
19	59	41	174	63	683
20	62	42	182		
21	66	43	191		

Table 3.9.15: Bark-scale subband table for SCL_1_960 configuration mode (LONG:960)

isf	crb_tbl[isf]	isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	3	22	66	44	189
1	5	23	72	45	199
2	8	24	74	46	209
3	10	25	77	47	221
4	13	26	80	48	233
5	15	27	85	49	246
6	18	28	88	50	261
7	21	29	93	51	276
8	23	30	98	52	293
9	26	31	101	53	312
10	29	32	107	54	331
11	32	33	112	55	353

12	35	34	117	56	377
13	38	35	123	57	404
14	40	36	129	58	434
15	43	37	134	59	465
16	46	38	142	60	502
17	49	39	148	61	542
18	53	40	155	62	589
19	56	41	164	63	640
20	59	42	171		
21	62	43	179		

Table 3.9.16: Bark-scale subband table for SCL_2_960 configuration mode (LONG:960)

isf	crb_tbl[isf]	isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	324	22	444	44	643
1	328	23	451	45	650
2	333	24	458	46	668
3	337	25	466	47	682
4	341	26	473	48	696
5	347	27	481	49	710
6	352	28	488	50	723
7	357	29	496	51	738
8	361	30	503	52	753
9	367	31	512	53	769
10	372	32	521	54	786
11	377	33	531	55	802
12	383	34	540	56	818
13	388	35	549	57	845
14	394	36	558	58	855
15	399	37	568	59	874
16	406	38	578	60	895
17	412	39	590	61	915
18	418	40	600	62	936
19	423	41	610	63	960
20	431	42	622		
21	437	43	632		

Table 3.9.17: Bark-scale subband table for SCL_2 configuration mode (LONG:1024)

isf	crb_tbl[isf]	isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	346	22	474	44	686
1	350	23	481	45	698
2	355	24	489	46	713
3	360	25	497	47	727
4	364	26	504	48	742
5	370	27	513	49	757
6	375	28	521	50	771
7	381	29	529	51	787
8	385	30	537	52	803
9	391	31	546	53	820
10	397	32	556	54	838

11	402	33	566	55	855
12	408	34	576	56	872
13	414	35	586	57	901
14	420	36	596	58	911
15	426	37	606	59	932
16	433	38	617	60	955
17	439	39	629	61	976
18	446	40	640	62	998
19	453	41	651	63	1024
20	460	42	663		
21	466	43	674		

Table 3.9.18: Bark-scale subband table for 16_16 configuration modes (LONG)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	6	11	96
1	12	12	110
2	18	13	127
3	25	14	147
4	31	15	171
5	38	16	201
6	46	17	239
7	54	18	288
8	63	19	360
9	73	20	512
10	83		

Table 3.9.19: Bark-scale subband table for 16_16 configuration mode (MEDIUM)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	6	10	83
1	12	11	96
2	18	12	110
3	25	13	125
4	31	14	143
5	38	15	163
6	46	16	183
7	54	17	204
8	63	18	230
9	73	19	256

Table 3.9.20: Bark-scale subband table for 16_16 configuration mode (SHORT)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	2	5	12
1	3	6	19
2	5	7	29
3	7	8	45
4	9	9	64

Table 3.9.21: Bark-scale subband table for 08_06 configuration mode (LONG)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	7	15	142
1	15	16	154

2	22	17	168
3	30	18	183
4	38	19	199
5	46	20	217
6	54	21	236
7	62	22	257
8	70	23	381
9	79	24	308
10	88	25	338
11	98	26	373
12	108	27	413
13	119	28	459
14	130	29	512

Table 3.9.22: Bark-scale subband table for 08_06 configuration mode (MEDIUM)

isf	crb_tbl[isf]	isf	crb_tbl[isf]
0	6	10	75
1	11	11	84
2	17	12	95
3	23	13	108
4	29	14	123
5	35	15	141
6	42	16	161
7	49	17	186
8	57	18	217
9	65	19	256

Table 3.9.23: Bark-scale subband table for 08_06 configuration mode (SHORT)

isf	crb_tbl[isf]
0	3
1	6
2	9
3	12
4	16
5	21
6	27
7	35
8	47
9	64

Table 3.9.24: Values of isp[] for 24_06, SCL_1, SCL_1_960, SCL_2 and SCL_2_960 configuration modes

split_num	isp[split_num]
0	0
1	5
2	14
3	20

Table 3.9.25: Values of isp[] for 16_16 configuration mode

split_num	isp[split_num]
0	0
1	5
2	10
3	16

Table 3.9.26: Values of isp[] for 08_06 configuration mode

split_num	isp[split_num]
0	0
1	3
2	8
3	12

3.11 Filterbank and Block Switching

3.11.1 Tool Description

The time/frequency representation of the signal is mapped onto the time domain by feeding it into the filterbank module. This module consists of an inverse modified discrete cosine transform (IMDCT), and a window and an overlap-add function. In order to adapt the time/frequency resolution of the filterbank to the characteristics of the input signal, a block switching tool is also adopted. N represents the window length, where N is a function of the **window_sequence**, see 3.3.3. For each channel, the N/2 time-frequency values $X_{i,k}$ are transformed into the N time domain values $x_{i,n}$ via the IMDCT. After applying the window function, for each channel, the first half of the $z_{i,n}$ sequence is added to the second half of the previous block windowed sequence $z_{(i-1),n}$ to reconstruct the output samples for each channel $out_{i,n}$.

3.11.2 Definitions

The syntax elements for the filterbank are specified in the raw data stream for the **single_channel_element** (see 1.3, Table 6.9)**, channel_pair_element** (see 1.3, Table 6.10)**,** and the **coupling_channel** (see 1.3, Table 6.18). They consist of the control information **window_sequence** and **window_shape**.

window_sequence 2 bit indicating which window sequence (i.e. block size) is used (see 1.3, Table 6.11).

window_shape 1 bit indicating which window function is selected (see 1.3, Table 6.11).

Table 3.3 shows the four **window_sequences** (ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE, EIGHT_SHORT_SEQUENCE, LONG_STOP_SEQUENCE).

3.11.3 Decoding Process

3.11.3.1 IMDCT

The analytical expression of the IMDCT is:

$$x_{i,n} = \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} spec[i][k]\cos\left(\frac{2\pi}{N}\left(n+n_0\right)\left(k+\frac{1}{2}\right)\right) \text{ for } \quad 0 \le n < N$$

where:

n = sample index

i = window index

k = spectral coefficient index

N = window length based on the window_ sequence value

n_0 = (N / 2 + 1) / 2

The synthesis window length N for the inverse transform is a function of the syntax element **window_sequence** and the algorithmic context. It is defined as follows:

AAC-derived coder

$$N = \begin{cases} 2048, & \text{if ONLY\_LONG\_SEQUENCE (0x0)} \\ 2048, & \text{if LONG\_START\_SEQUENCE (0x1)} \\ 256, & \text{if EIGHT\_SHORT\_SEQUENCE (0x2), (8 times)} \\ 2048, & \text{if LONG\_STOP\_SEQUENCE (0x3)} \end{cases}$$

AAC-derived scalable coder with core coder frame sizes of multiples of 10ms

$$N = \begin{cases} 1920, & \text{if ONLY\_LONG\_SEQUENCE (0x0)} \\ 1920, & \text{if LONG\_START\_SEQUENCE (0x1)} \\ 240, & \text{if EIGHT\_SHORT\_SEQUENCE (0x2), (8 times)} \\ 1920, & \text{if LONG\_STOP\_SEQUENCE (0x3)} \end{cases}$$

The meaningful block transitions are as follows:

from ONLY_LONG_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from LONG_START_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE
from LONG_STOP_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from EIGHT_SHORT_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE

In addition to the meaningful block transitions the following transitions are possible:

from ONLY_LONG_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE
from LONG_START_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from LONG_STOP_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE
from EIGHT_SHORT_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE

This will still result in a reasonably smooth transition from one block to the next.

3.11.3.2 Windowing and block switching

Depending on the **window_sequence** and **window_shape** element different transform windows are used. A combination of the window halves described as follows offers all possible window_sequences.

For **window_shape** == 1, the window coefficients are given by the Kaiser - Bessel derived (KBD) window as follows:

$$W_{KBD\_LEFT,N}(n) = \sqrt{\frac{\sum_{p=0}^{n}\left[W'(n,\alpha)\right]}{\sum_{p=0}^{N/2}\left[W'(p,\alpha)\right]}} \quad \text{for} \quad 0 \le n < \frac{N}{2}$$

$$W_{KBD\_RIGHT,N}(n) = \sqrt{\frac{\sum_{p=0}^{N-n}[W'(n,\alpha)]}{\sum_{p=0}^{N/2}[W'(p,\alpha)]}} \qquad \text{for} \quad \frac{N}{2} \le n < N$$

where:

W', Kaiser - Bessel kernel window function, see also [5], is defined as follows:

$$W'(n,\alpha) = \frac{I_0\left[\pi\alpha\sqrt{1.0-\left(\frac{n-N/4}{N/4}\right)^2}\right]}{I_0[\pi\alpha]} \qquad \text{for } 0 \le n \le \frac{N}{2}$$

$$I_0[x] = \sum_{k=0}^{\infty}\left[\frac{\left(\frac{x}{2}\right)^k}{k!}\right]^2$$

α = kernel window alpha factor, $\alpha = \begin{cases} 4 \text{ for N = 2048 (1920)} \\ 6 \text{ for N = 256 (240)} \end{cases}$

Otherwise, for **window_shape** == 0, a sine window is employed as follows:

$$W_{SIN\_LEFT,N}(n) = \sin\left(\frac{\pi}{N}(n+\frac{1}{2})\right) \qquad \text{for} \quad 0 \le n < \frac{N}{2}$$

$$W_{SIN\_RIGHT,N}(n) = \sin\left(\frac{\pi}{N}(n+\frac{1}{2})\right) \qquad \text{for} \quad \frac{N}{2} \le n < N$$

The window length N can be 2048 (1920) or 256 (240) for the KBD and the sine window. How to obtain the possible window sequences is explained in the parts a)-d) of this clause. All four window_sequences described below have a total length of 2048 samples.

For all kinds of window_sequences the window_shape of the left half of the first transform window is determined by the window shape of the previous block. The following formula expresses this fact:

$$W_{LEFT,N}(n) = \begin{cases} W_{KBD\_LEFT,N}(n), \text{ if } \quad window\_shape\_previous\_block == 1 \\ W_{SIN\_LEFT,N}(n), \text{ if } \quad window\_shape\_previous\_block == 0 \end{cases}$$

where:

window_shape_previous_block: **window_shape** of the previous block (i-1).

For the first block of the bitstream to be decoded the **window_shape** of the left and right half of the window are identical.

a) ONLY_LONG_SEQUENCE:

The **window_sequence** == ONLY_LONG_SEQUENCE is equal to one LONG_WINDOW (see Table 3.3) with a total window length N_l of 2048 (1920).

For **window_shape**==1 the window for ONLY_LONG_SEQUENCE is given as follows:

$$w(n) = \begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0 \le n < N\_l/2 \\ W_{KBD\_RIGHT,N\_l}(n), & \text{for } N\_l/2 \le n < N\_l \end{cases}$$

If **window_shape**==0 the window for ONLY_LONG_SEQUENCE can be described as follows:

$$w(n) = \begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0 \le n < N\_l/2 \\ W_{SIN\_RIGHT,N\_l}(n), & \text{for } N\_l/2 \le n < N\_l \end{cases}$$

After windowing, the time domain values ($z_{i,n}$) can be expressed as:

$$z_{i,n} = w(n) \cdot x_{i,n};$$

b) LONG_START_SEQUENCE:

The LONG_START_SEQUENCE is needed to obtain a correct overlap and add for a block transition from a ONLY_LONG_SEQUENCE to a EIGHT_SHORT_SEQUENCE.

Window length N_l and N_s is set to 2048 (1920) and 256 (240) respectively.

If **window_shape**==1 the window for LONG_START_SEQUENCE is given as follows:

$$w(n) = \begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0 \le n < N\_l/2 \\ 1.0, & \text{for } N\_l/2 \le n < \frac{3N\_l - N\_s}{4} \\ W_{KBD\_RIGHT,N\_s}(n + \frac{N\_s}{2} - \frac{3N\_l - N\_s}{4}), & \text{for } \frac{3N\_l - N\_s}{4} \le n < \frac{3N\_l + N\_s}{4} \\ 0.0, & \text{for } \frac{3N\_l + N\_s}{4} \le n < N\_l \end{cases}$$

If **window_shape**==0 the window for LONG_START_SEQUENCE looks like:

$$w(n) = \begin{cases} W_{LEFT,N\_l}(n), & \text{for } 0 \le n < N\_l/2 \\ 1.0, & \text{for } N\_l/2 \le n < \frac{3N\_l - N\_s}{4} \\ W_{SIN\_RIGHT,N\_s}(n + \frac{N\_s}{2} - \frac{3N\_l - N\_s}{4}), & \text{for } \frac{3N\_l - N\_s}{4} \le n < \frac{3N\_l + N\_s}{4} \\ 0.0, & \text{for } \frac{3N\_l + N\_s}{4} \le n < N\_l \end{cases}$$

The windowed time-domain values can be calculated with the formula explained in a).

c) EIGHT_SHORT

The **window_sequence** == EIGHT_SHORT comprises eight overlapped and added SHORT_WINDOWs (see Table 3.3) with a length N_s of 256 (240) each. The total length of the window_sequence together with leading and following zeros is 2048 (1920). Each of the eight short blocks are windowed separately first. The short block number is indexed with the variable j = 0,…, M-1 (M=N_l/N_s).
The **window_shape** of the previous block influences the first of the eight short blocks (W_0(n)) only.
If **window_shape**==1 the window functions can be given as follows:

$$W_0(n) = \begin{cases} W_{LEFT,N\_s}(n), & \text{for } 0 \le \text{n} < N\_s/2 \\ W_{KBD\_RIGHT,N\_s}(n), & \text{for } \text{N\_s}/2 \le \text{n} < N\_s \end{cases}$$

$$W_{1-(M-1)}(n) = \begin{cases} W_{LEFT,N\_s}(n), & \text{for } 0 \le \text{n} < N\_s/2 \\ W_{KBD\_RIGHT,N\_s}(n), & \text{for } \text{N\_s}/2 \le \text{n} < N\_s \end{cases}$$

Otherwise, if **window_shape**==0, the window functions can be described as:

$$W_0(n) = \begin{cases} W_{LEFT,N\_s}(n), & \text{for } 0 \le \text{n} < N\_s/2 \\ W_{SIN\_RIGHT,N\_s}(n), & \text{for } \text{N\_s}/2 \le \text{n} < N\_s \end{cases}$$

$$W_{1-(M-1)}(n) = \begin{cases} W_{SIN\_LEFT,N\_s}(n), & \text{for } 0 \le \text{n} < N\_s/2 \\ W_{SIN\_RIGHT,N\_s}(n), & \text{for } \text{N\_s}/2 \le \text{n} < N\_s \end{cases}$$

The overlap and add between the EIGHT_SHORT **window_sequence** resulting in the windowed time domain values $z_{i,n}$ is described as follows:

$$z_{i,n} = \begin{cases} 0, & \text{for } 0 \le n < \frac{N\_l - N\_s}{4} \\ x_{0,n-\frac{N\_l-N\_s}{4}} \cdot W_0(n - \frac{N\_l-N\_s}{4}), & \text{for } \frac{N\_l-N\_s}{4} \le n < \frac{N\_l+N\_s}{4} \\ x_{j-1,n-\frac{N\_l+(2j-3)\cdot N\_s}{4}} \cdot W_{j-1}(n - \frac{N\_l+(2j-3)N\_s}{4}) + x_{j,n-\frac{N\_l+(2j-1)N\_s}{4}} \cdot W_j(n - \frac{N\_l+(2j-1)N\_s}{4}), & \\ & \text{for } 1 \le j < M,\ \frac{N\_l+(2j-1)N\_s}{4} \le n < \frac{N\_l+(2j+1)N\_s}{4} \\ x_{M-1,n-\frac{N\_l+(2M-3)N\_s}{4}} \cdot W_{M-1}(n - \frac{N\_l+(2M-3)N\_s}{4}), & \\ & \text{for } \frac{N\_l+(2M-1)N\_s}{4} \le n < \frac{N\_l+(2M+1)N\_s}{4} \\ 0, & \text{for } \frac{N\_l+(2M+1)N\_s}{4} \le n < N\_l \end{cases}$$

d) LONG_STOP_SEQUENCE

This window_sequence is needed to switch from a EIGHT_SHORT_SEQUENCE back to a ONLY_LONG_SEQUENCE.

If **window_shape**==1 the window for LONG_STOP_SEQUENCE is given as follows:

$$W(n) = \begin{cases} 0.0, & \text{for } 0 \le \text{n} < \frac{N\_l-N\_s}{4} \\ W_{LEFT,N\_s}(n - \frac{N\_l-N\_s}{4}), & \text{for } \frac{N\_l-N\_s}{4} \le \text{n} < \frac{N\_l+N\_s}{4} \\ 1.0, & \text{for } \frac{N\_l+N\_s}{4} \le \text{n} < N\_l/2 \\ W_{KBD\_RIGHT,N\_l}(n), & \text{for } N\_l/2 \le \text{n} < N\_l \end{cases}$$

If **window_shape**==0 the window for LONG_START_SEQUENCE is determined by:

$$w(n) = \begin{cases} 0.0, & \text{for } 0 \le n < \frac{N\_l - N\_s}{4} \\ W_{LEFT,N\_s}(n - \frac{N\_l - N\_s}{4}), & \text{for } \frac{N\_l - N\_s}{4} \le n < \frac{N\_l + N\_s}{4} \\ 1.0, & \text{for } \frac{N\_l + N\_s}{4} \le n < N\_l / 2 \\ W_{SIN\_RIGHT,N\_l}(n), & \text{for } N\_l / 2 \le n < N\_l \end{cases}$$

The windowed time domain values can be calculated with the formula explained in a).

3.11.3.3 Overlapping and adding with previous window sequence

Besides the overlap and add within the EIGHT_SHORT **window_sequence** the first (left) half of every **window_sequence** is overlapped and added with the second (right) half of the previous **window_sequence** resulting in the final time domain values $out_{i,n}$. The mathematic expression for this operation can be described as follows. It is valid for all four possible window_sequences.

$$out_{i,n} = z_{i,n} + z_{i-1,n+\frac{N}{2}}; \qquad \text{for } 0 \le n < \frac{N}{2}, \quad N = 2048\ (1920)$$

3.11.4 Three block type mode

Three block type mode is an extention of the filterbank and block switching tool. This mode is enabled when interleaved vector quantization and spectrum and normalization tools are used. In this extention, block types with medium length are used additionally.

3.11.4.1 Syntax elements

The syntax elements for extended filterbank are specified in the raw data stream element_stream_vq(). Differences from non-extended mode are:

- In this mode the syntax element **window_shape** is not transmitted but set to zero.
- The syntax element **window_sequence** is indicated by 4 bits.

3.11.4.2 Block sizes

Window length N of non-extended filterbank are set to 1024 for long blocks, and 128 for short blocks. However, for extended mode, N is set to 2*N_FR for each frame length (see tables from ? to ?). Window length for long, medium, and short block is represented as N_L, N_M, N_S respectively.

For extended mode, three block lengths, N_L, N_M, and N_S is necessary for filterbank settings. N_L, N_M, N_S is set to 2*N_FR for long, medium, short frame length respectively (see tables from ? to ?).

3.11.4.3 IMDCT

In extended mode, medium-length blocks are used. So nine window sequences are possible, and the synthesis window length N for the IMDCT as a function of **window_sequence** is defined as follows:

$$N = \begin{cases} N_L, & \text{if ONLY\_LONG\_SEQUENCE (0x0)} \\ N_L, & \text{if LONG\_SHORT\_SEQUENCE (0x1)} \\ N_S, & \text{if ONLY\_SHORT\_SEQUENCE (0x2),} \quad (N_L / N_S \text{ times}) \\ N_L, & \text{if SHORT\_LONG\_SEQUENCE (0x3)} \\ N_M, & \text{if SHORT\_MEDIUM\_SEQUENCE (0x4),} \quad (N_L / N_M \text{ times}) \\ N_L, & \text{if MEDIUM\_LONG\_SEQUENCE (0x5)} \\ N_L, & \text{if LONG\_MEDIUM\_SEQUENCE (0x6)} \\ N_M, & \text{if MEDIUM\_SHORT\_SEQUENCE (0x7),} \quad (N_L / N_M \text{ times}) \\ N_M, & \text{if ONLY\_MEDIUM\_SEQUENCE (0x8),} \quad (N_L / N_M \text{ times}) \end{cases}$$

3.11.4.4 Window sequences

The meaningful block length transitions out of all possible transitions are marked as ‘o’ in following diagram.

window sequence from	OL	LS	OS	SL	SM	ML	LM	MS	OM
ONLY_LONG_SEQUENCE	o	o					o		
LONG_SHORT_SEQUENCE			o	o	o				
ONLY_SHORT_SEQUENCE			o	o	o				
SHORT_LONG_SEQUENCE	o	o					o		
SHORT_MEDIUM_SEQUENCE						o		o	o
MEDIUM_LONG_SEQUENCE	o	o					o		
LONG_MEDIUM_SEQUENCE						o		o	o
MEDIUM_SHORT_SEQUENCE			o	o	o				
ONLY_MEDIUM_SEQUENCE						o		o	o

3.11.4.5 Windowing and block switching

a) If window_sequence == ONLY_LONG_SEQUENCE, windowing and block switching is the same as non-extended mode, total length N is set to N_L.

b) If window_sequence == LONG_SHORT_SEQUENCE, windowing and block switching is the same as LONG_START_SEQUENCE of non-extended mode.

c) If window_sequence == ONLY_SHORT_SEQUENCE, windowing and block switching is the same as EIGHT_SHORT_SEQUENCE of non-extended mode. Window length N_l is set to N_L, N_s is set to N_S.

d) If window_sequence == SHORT_LONG_SEQUENCE, windowing and block switching is the same as LONG_STOP_SEQUENCE, window length N_l is set to N_L, N_s is set to N_S.

e) If window_sequence == SHORT_MEDIUM_SEQUENCE, windowing and block switching is the same as ONLY_SHORT_SEQUENCE, window length N_l is set to N_L, N_s is set to N_M, except for the first window function W_0 is given from equation used in SHORT_LONG_WINDOW, N_l=N_M, N_s=N_S.

f) If window_sequence == MEDIUM_LONG_SEQUENCE, windowing and block switching is the same as SHORT_LONG_SEQUENCE, window length N_l is set to N_L, N_s is set to N_M.

g) If window_sequence == LONG_MEDIUM_SEQUENCE, windowing and block switching is the same as LONG_SHORT_SEQUENCE, window length N_l is set to N_L, N_s is set to N_M.

h) If window_sequence == MEDIUM_SHORT_SEQUENCE, windowing and block switching is the same as ONLY_SHORT_SEQUENCE, N_l = N_L, N_s = N_M, except for the last window function W_{M-1} ($M=N_L/N_M$) is given from equation used in LONG_SHORT_WINDOW, N_l = N_M, N_s=N_S.

I) If window_sequence == ONLY_MEDIUM_SEQUENCE, windowing and block switching is the same as ONLY_SHORT_SEQUENCE, window length N_l is set to NL, N_s is set to NM.

1.4.1 Overlapping and adding with previous window sequence

For this mode, window length for overlapping and adding is set to N_L, instead of 2048.

3.12 Gain Control

3.12.1 Tool description

The gain control tool is made up of several gain compensators and overlap/add processing stages, and an IPQF (Inverse Polyphase Quadrature Filter) stage. This tool receives non-overlapped signal sequences provided by the IMDCT stages, window_sequence and gain_control_data, and then reproduces the output PCM data. The block diagram for the gain control tool is shown in Figure 00.11.

Due to the characteristics of the PQF filterbank, the order of the MDCT coefficients in each even PQF band must be reversed. This is done by reversing the spectral order of the MDCT coefficients, i.e. exchanging the higher frequency MDCT coefficients with the lower frequency MDCT coefficients.

If the gain control tool is used, the configuration of the filter bank tool is changed as follows. In the case of an EIGHT_SHORT_SEQUENCE window_sequence, the number of coefficients for the IMDCT is 32 instead of 128 and eight IMDCTs are carried out. In the case of other window_sequence values, the number of coefficients for the IMDCT is 256 instead of 1024 and one IMDCT is performed. In all cases, the filter bank tool outputs a total of 2048 non-overlapped values per frame. These values are supplied to the gain control tool as $U_{W,B}(j)$ defined in 11.3.3.

The IPQF combines four uniform frequency bands and produces a decoded time domain output signal. The aliasing components introduced by the PQF in the encoder are cancelled by the IPQF.

The gain values for each band can be controlled independently except for the lowest frequency band. The step size of gain control is 2 ^ n where n is an integer.

The gain control tool outputs a time signal sequence which is $AS(n)$ defined in 11.3.4.

3.12.2 Definitions

gain control data	side information indicating the gain values and the positions used for the gain change.
IPQF band	each split band of IPQF.
adjust_num	3-bit field indicating the number of gain changes for each IPQF band. The maximum number of gain changes is seven (see 1.3, Table 6.23).
max_band	2-bit field indicating the number of IPQF bands which contain spectral data counting from the lowest IPQF band to higher IPQF bands. The number of IPQF bands which contain spectral data is **max_band** + 1 (see 1.3, Table 6.23).
alevcode	4-bit field indicating the gain value for one gain change (see 1.3, Table 6.23).
aloccode	2-, 4-, or 5-bit field indicating the position for one gain change. The length of this data varies depending on the window sequence (see 1.3, Table 6.23).

3.12.3 Decoding process

The following four processes are required for decoding.

(1) Gain control data decoding

(2) Gain control function setting
(3) Gain control windowing and overlapping
(4) Synthesis filter

3.12.3.1 Gain control data decoding

Gain control data are reconstructed as follows.

(1)

$$NAD_{W,B} = \textbf{adjust\_num}[B][W]$$

(2)

$$ALOC_{W,B}(m) = AdjLoc(\textbf{aloccode}[B][W][m-1]), \quad 1 \le m \le NAD_{W,B}$$

$$ALEV_{W,B}(m) = 2^{AdjLev(\textbf{alevcode}[B][W][m-1])}, \quad 1 \le m \le NAD_{W,B}$$

(3)

$$ALOC_{W,B}(0) = 0$$

$$ALEV_{W,B}(0) = \begin{cases} 1, & if\ NAD_{W,B} == 0 \\ ALEV_{W,B}(1), & otherwise \end{cases}$$

(4)

$$ALOC_{W,B}(NAD_{W,B}+1) = \begin{cases} 256, & W == 0 \quad if\ \text{ONLY\_LONG\_SEQUENCE} \\ \left.\begin{matrix} 112, & W == 0 \\ 32, & W == 1 \end{matrix}\right\} & if\ \text{LONG\_START\_SEQUENCE} \\ 32, & 0 \le W \le 7 \quad if\ \text{EIGHT\_SHORT\_SEQUENCE} \\ \left.\begin{matrix} 112, & W == 0 \\ 256, & W == 1 \end{matrix}\right\} & if\ \text{LONG\_STOP\_SEQUENCE} \end{cases}$$

$$ALEV_{W,B}(NAD_{W,B}+1) = 1$$

where

$NAD_{W,B}$:	Gain Control Information Number, an integer
$ALOC_{W,B}(m)$:	Gain Control Location, an integer
$ALEV_{W,B}(m)$:	Gain Control Level, an integer-valued real number
B:	Band ID, an integer from 1 to 3
W:	Window ID, an integer from 0 to 7
m:	an integer

aloccode[*B*][*W*][*m*] must be set so that $\{ALOC_{W,B}(m)\}$ satisfies the following conditions.

$$ALOC_{W,B}(m_1) < ALOC_{W,B}(m_2), \quad 1 \le m_1 < m_2 \le NAD_{W,B}+1$$

In cases of LONG_START_SEQUENCE and LONG_STOP_SEQUENCE, the values 14 and 15 of **aloccode**[*B*][*0*][*m*] are invalid. *AdjLoc*() is defined in Table 00.11. *AdjLev*() is defined in Table 00.22.

3.12.3.2 Gain control function setting

The Gain control function is obtained as follows.

(1)

$$M_{W,B,j} = Max\{m: ALOC_{W,B}(m) \le j\},$$

$$\begin{cases} 0 \le j \le 255, & W == 0 \quad if\ \text{ONLY\_LONG\_SEQUENCE} \\ \left.\begin{matrix} 0 \le j \le 111, & W == 0 \\ 0 \le j \le 31, & W == 1 \end{matrix}\right\} & if\ \text{LONG\_START\_SEQUENCE} \\ 0 \le j \le 31, & 0 \le W \le 7 \quad if\ \text{EIGHT\_SHORT\_SEQUENCE} \end{cases}$$

$$\left.\begin{array}{l} 0 \le j \le 111, \quad W == 0 \\ 0 \le j \le 255, \quad W == 1 \end{array}\right\} \quad if \text{ LONG\_ STOP\_ SEQUENCE}$$

(2)

$$FMD_{W,B}(j) = \begin{cases} Inter\begin{pmatrix} ALEV_{W,B}(M_{W,B,j}), \\ ALEV_{W,B}(M_{W,B,j}+1), \\ j - ALOC_{W,B}(M_{W,B,j}) \end{pmatrix}, \\ \qquad\qquad if\ ALOC_{W,B}(M_{W,B,j}) \le j \le ALOC_{W,B}(M_{W,B,j}) + 7 \\ ALEV_{W,B}(M_{W,B,j}+1), \quad otherwise \end{cases}$$

(3)

if ONLY_LONG_SEQUENCE

$$GMF_{0,B}(j) = \begin{cases} ALEV_{0,B}(0) \times PFMD_B(j), & 0 \le j \le 255 \\ FMD_{0,B}(j-256), & 256 \le j \le 511 \end{cases}$$

$$PFMD_B(j) = FMD_{0,B}(j), \quad 0 \le j \le 255$$

if LONG_START_SEQUENCE

$$GMF_{0,B}(j) = \begin{cases} ALEV_{0,B}(0) \times ALEV_{1,B}(0) \times PFMD_B(j), & 0 \le j \le 255 \\ ALEV_{1,B}(0) \times FMD_{0,B}(j-256), & 256 \le j \le 367 \\ FMD_{1,B}(j-368), & 368 \le j \le 399 \\ 1, & 400 \le j \le 511 \end{cases}$$

$$PFMD_B(j) = FMD_{1,B}(j), \quad 0 \le j \le 31$$

if EIGHT_SHORT_SEQUENCE

$$GMF_{W,B}(j) = \begin{cases} ALEV_{W,B}(0) \times PFMD_B(j), & W == 0, \quad 0 \le j \le 31 \\ ALEV_{W,B}(0) \times FMD_{W-1,B}(j), & 1 \le W \le 7, \quad 0 \le j \le 31 \\ FMD_{W,B}(j-32), & 0 \le W \le 7, \quad 32 \le j \le 63 \end{cases}$$

$$PFMD_B(j) = FMD_{7,B}(j), \quad 0 \le j \le 31$$

if LONG_STOP_SEQUENCE

$$GMF_{0,B}(j) = \begin{cases} 1, & 0 \le j \le 111 \\ ALEV_{0,B}(0) \times ALEV_{1,B}(0) \times PFMD_B(j-112), & 112 \le j \le 143 \\ ALEV_{1,B}(0) \times FMD_{0,B}(j-144), & 144 \le j \le 255 \\ FMD_{1,B}(j-256), & 256 \le j \le 511 \end{cases}$$

$$PFMD_B(j) = FMD_{1,B}(j), \quad 0 \le j \le 255$$

(4)

$$AD_{W,B}(j) = \frac{1}{GMF_{W,B}(j)},$$

$0 \le j \le 511, \quad W == 0 \quad if$ ONLY_ LONG_ SEQUENCE
$0 \le j \le 511, \quad W == 0 \quad if$ LONG_ START_ SEQUENCE
$0 \le j \le 63, \quad 0 \le W \le 7 \quad if$ EIGHT_SHORT_SEQUENCE
$0 \le j \le 511, \quad W == 0 \quad if$ LONG_ STOP_ SEQUENCE

where

$FMD_{W,B}(j)$: Fragment Modification Function, a real number

$PFMD_B(j)$: Fragment Modification Function of previous frame, a real number

$GMF_{W,B}(j)$: Gain Modification Function, a real number

$AD_{W,B}(j)$: Gain Control Function, a real number

$ALOC_{W,B}(m)$: Gain Control Location defined in 0, an integer

$ALEV_{W,B}(m)$: Gain Control Level defined in 0, an integer-valued real number

B: Band ID, an integer from 1 to 3

W: Window ID, an integer from 0 to 7

$M_{W,B,j}$: an integer

m: an integer

and

$$Inter(a,b,j) = 2^{\frac{(8-j)\log_2(a)+j\log_2(b)}{8}}$$

Note that the initial value of $PFMD_B(j)$ must be set 1.0.

3.12.3.3 Gain control windowing and overlapping

Band Sample Data are obtained through the processes (1) to (2) shown below.

(1) Gain Control Windowing

if B=0

$$T_{W,B}(j) = U_{W,B}(j),$$

$0 \le j \le 511, \quad W == 0 \quad if$ ONLY_LONG_SEQUENCE

$0 \le j \le 511, \quad W == 0 \quad if$ LONG_START_SEQUENCE

$0 \le j \le 63, \quad 0 \le W \le 7 \quad if$ EIGHT_SHORT_SEQUENCE

$0 \le j \le 511, \quad W == 0 \quad if$ LONG_STOP_SEQUENCE

else

$$T_{W,B}(j) = AD_{W,B}(j) \times U_{W,B}(j),$$

$0 \le j \le 511, \quad W == 0 \quad if$ ONLY_LONG_SEQUENCE

$0 \le j \le 511, \quad W == 0 \quad if$ LONG_START_SEQUENCE

$0 \le j \le 63, \quad 0 \le W \le 7 \quad if$ EIGHT_SHORT_SEQUENCE

$0 \le j \le 511, \quad W == 0 \quad if$ LONG_STOP_SEQUENCE

(2) Overlapping

if ONLY_LONG_SEQUENCE

$$V_B(j) = PT_B(j) + T_{0,B}(j), \quad 0 \le j \le 255$$

$$PT_B(j) = T_{0,B}(j+256), \quad 0 \le j \le 255$$

if LONG_START_SEQUENCE

$$V_B(j) = PT_B(j) + T_{0,B}(j), \quad 0 \le j \le 255$$

$$V_B(j+256) = T_{0,B}(j+256), \quad 0 \le j \le 111$$

$$PT_B(j) = T_{0,B}(j+368), \quad 0 \le j \le 31$$

if EIGHT_SHORT_SEQUENCE

$$V_B(j) = PT_B(j) + T_{W,B}(j), \quad W == 0, \quad 0 \le j \le 31$$

$$V_B(32W+j) = T_{W-1,B}(j+32) + T_{W,B}(j), \quad 1 \le W \le 7, \quad 0 \le j \le 31$$

$$PT_B(j) = T_{w,B}(j+32), \quad W == 7, \quad 0 \le j \le 31$$

if LONG_STOP_SEQUENCE

$$V_B(j) = PT_B(j) + T_{0,B}(j+112), \quad 0 \le j \le 31$$
$$V_B(j+32) = T_{0,B}(j+144), \quad 0 \le j \le 111$$
$$PT_B(j) = T_{0,B}(j+256), \quad 0 \le j \le 255$$

where

$U_{W,B}(j)$: Band Spectrum Data, a real number

$T_{W,B}(j)$: Gain Controlled Block Sample Data, a real number

$PT_B(j)$: Gain Controlled Block Sample Data of previous frame, a real number

$V_B(j)$: Band Sample Data, a real number

$AD_{W,B}(j)$: Gain Control Function defined in 0, a real number

B: Band ID, an integer from 0 to 3
W: Window ID, an integer from 0 to 7
j: an integer

Note that the initial value of $PT_B(j)$ must be set 0.0.

3.12.3.4 Synthesis filter

Audio Sample Data are obtained from the following equations.

(1)

$$\tilde{V}_B(j) = \begin{cases} V_B(k), & if\ j == 4k, \\ 0, & else \end{cases} \quad 0 \le B \le 3$$

(2)

$$Q_B(j) = Q(j) \times \cos\left(\frac{(2B+1)(2j-3)\pi}{16}\right), \quad 0 \le j \le 95, \quad 0 \le B \le 3$$

(3)

$$AS(n) = \sum_{B=0}^{3} \sum_{j=0}^{95} Q_B(j) \times \tilde{V}_B(n-j)$$

where

$AS(n)$: Audio Sample Data

$V_B(n)$: Band Sample Data defined in 0, a real number

$\tilde{V}_B(j)$: Interpolated Band Sample Data, a real number

$Q_B(j)$: Synthesis Filter Coefficients, a real number

$Q(j)$: Prototype Coefficients given below, a real number

B: Band ID, an integer from 0 to 3
W: Window ID, an integer from 0 to 7
n: an integer
j: an integer
k: an integer

The values of *Q*(0) to *Q*(47) are shown in Table 00.33. The values of *Q*(48) to *Q*(95) are obtained from the following equation.

$$Q(j) = Q(95-j), \quad 48 \le j \le 95$$

3.12.4 Diagrams

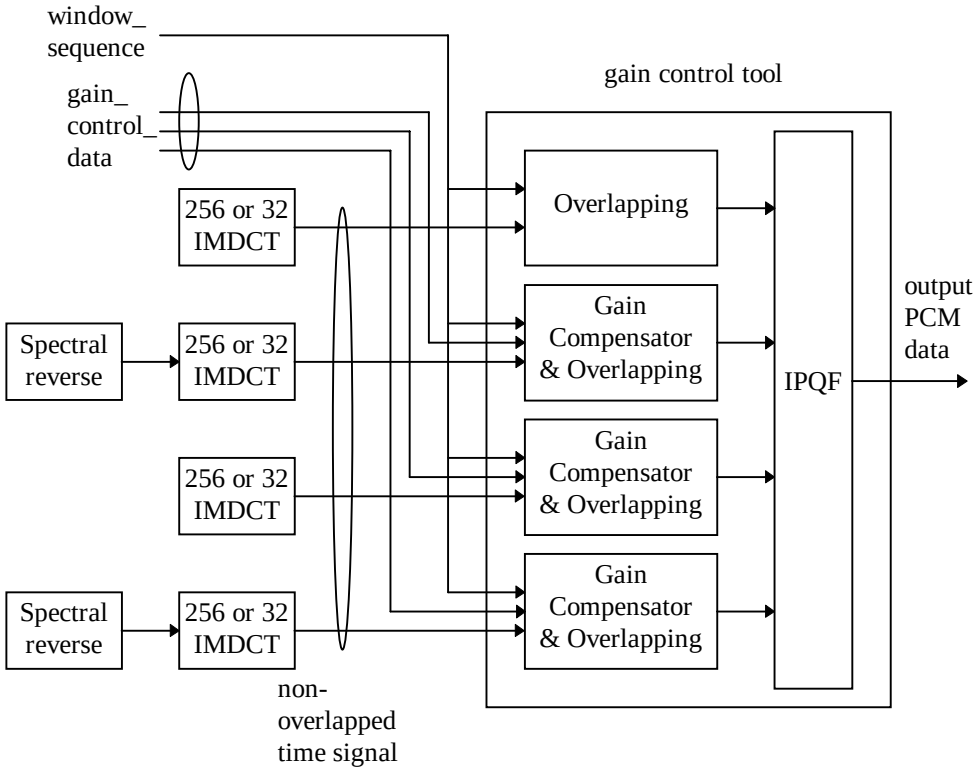


Figure 0.1 – Block diagram of gain control tool

3.12.5 Tables

Table 0.1 – *AdjLoc()*

AC	***AdjLoc(AC)***	***AC***	***AdjLoc(AC)***
0	0	16	128
1	8	17	136
2	16	18	144
3	24	19	152
4	32	20	160
5	40	21	168
6	48	22	176
7	56	23	184
8	64	24	192
9	72	25	200
10	80	26	208
11	88	27	216
12	96	28	224
13	104	29	232
14	112	30	240
15	120	31	248

Table 0.2 – *AdjLev()*

AV	***AdjLev(AV)***
0	-4
1	-3
2	-2
3	-1
4	0

5	1
6	2
7	3
8	4
9	5
10	6
11	7
12	8
13	9
14	10
15	11

Table 0.3 – *Q()*

j	*Q(j)*	*j*	*Q(j)*
0	9.7655291007575512E-05	24	-2.2656858741499447E-02
1	1.3809589379038567E-04	25	-6.8031113858963354E-03
2	9.8400749256623534E-05	26	1.5085400948280744E-02
3	-8.6671544782335723E-05	27	3.9750993388272739E-02
4	-4.6217998911921346E-04	28	6.2445363629436743E-02
5	-1.0211814095158174E-03	29	7.7622327748721326E-02
6	-1.6772149340010668E-03	30	7.9968338496132926E-02
7	-2.2533338951411081E-03	31	6.5615493068475583E-02
8	-2.4987888343213967E-03	32	3.3313658300882690E-02
9	-2.1390815966761882E-03	33	-1.4691563058190206E-02
10	-9.5595397454597772E-04	34	-7.2307890475334147E-02
11	1.1172111530118943E-03	35	-1.2993222541703875E-01
12	3.9091309127348584E-03	36	-1.7551641029040532E-01
13	6.9635703420118673E-03	37	-1.9626543957670528E-01
14	9.5595442159478339E-03	38	-1.8073330670215029E-01
15	1.0815766540021360E-02	39	-1.2097653136035738E-01
16	9.8770514991715300E-03	40	-1.4377370758549035E-02
17	6.1562567291327357E-03	41	1.3522730742860303E-01
18	-4.1793946063629710E-04	42	3.1737852699301633E-01
19	-9.2128743097707640E-03	43	5.1590021798482233E-01
20	-1.8830775873369020E-02	44	7.1080020379761377E-01
21	-2.7226498457701823E-02	45	8.8090632488444798E-01
22	-3.2022840857588906E-02	46	1.0068321641150089E+00
23	-3.0996332527754609E-02	47	1.0737914947736096E+00

3.13 Noiseless Coding for the Small Step Scalability : BSAC

BSAC stands for bit sliced arithmetic coding and is the name of a noiseless coding kernl that provides a fine granule scalability in the MPEG-4 audio T/F coder. The BSAC noiseless coding module is an alternative to the AAC coding module, with all other modules of the AAC-based coder remaining unchanged. The BSAC noiseless coding is used to make the bitstream scalable and further reduce the redundancy of the scalefactors and the quantized spectrum. The BSAC noiseless decoding process is split into 4 clauses. Clause 3.13.1 to 3.13.4 describe the detailed decoding process of the spectral data, the stereo or pns related data, the scalefactors and the arithmetic model index.

3.13.1 Arithmetic decoding of Bit-Sliced Spectral Data

3.13.1.1 Tool description

BSAC uses the bit-slicing scheme of the quantized spectral samples in order to provide the small step scalability. And it encode the bit-sliced data using arithmetic coding scheme in order to reduce the average bits transmitted while suffering no loss of fidelity.

In BSAC scalable coding scheme, a quantized sequence is divided into coding bands, as shown in clause 2.3.11.4. And, a quantized sequence is mapped into a bit-sliced sequence within a coding band. Four-dimensional vectors are formed from the bit-sliced sequence of the quantized spectrum and each 4-dimensional vector is divided into two subvectors. The noiseless coding of the subvectors relies on the arithmetic model index of the coding band, the significance, the dimension of the bit-sliced vector and the previous state.

The significance of the bit-sliced data is the bit-position of the vector to be coded.
The previous states are updated with coding the vectors from MSB to LSB. They are initialized to 0. And they are set to 1 when bit-value is non-zero.

The arithmetic model index for encoding the bit-sliced data within each coding band is transmitted starting from the lowest frequency coding band and progressing to the highest frequency coding band. For the detailed description of the arithmetic model index, see clause 3.13.4. Table 3.13.4 lists 32 arithmetic models which are used for encoding/decoding the bit-sliced data. The BSAC arithmetic model consists of several sub-models. Sub-models are classified and chosen according to the significance, the previous state and the dimension of the vector to be decoded as shown Table B.18 to Table B.48. Two subvectors are arithmetic encoded using the sub-model chosen from a set of several possible sub-models of BSAC arithmetic model.

3.13.1.2 Definitions

Bit stream elements:

acod_vec0[ArModel[i]][snf][subvector0] Arithmetic codeword from arithmetic coding with the model **BSAC_arith_model[ArModel[i]][snf][dim0]** that encodes the next subvector0, where subvector0 is composed of bit-values whose previous state is 0 among 4-tuple (w,x,y,z) which is formed from the bit-sliced sequence of the quantized spectral coefficients. Thus, w = x_quant[sfb][bin][Abit[i]] & (1<<(snf-1)), x = x_quant[sfb][bin+1][Abit[i]] & (1<<(snf-1)), y = x_quant[sfb][bin+2][Abit[i]] & (1<<(snf-1)) and z = x_quant[sfb][bin+3][Abit[i]] & (1<<(snf-1)) , where snf is the significance of the bit. N-tuples progress from low to high frequency within the current coding band and from MSB to LSB.

acod_vec1[ArModel[i]][snf][subvector1] Arithmetic codeword from arithmetic coding with the model **BSAC_arith_model[ArModel[i]][snf][dim1]** that encodes the next subvector1, where subvector1 is composed of bit-values whose previous state is 1 among 4-tuple (w,x,y,z) which is formed from the bit-sliced sequence of the quantized spectral coefficients. Thus, w = x_quant[sfb][bin][Abit[i]] & (1<<(snf-1)), x = x_quant[sfb][bin+1][Abit[i]] & (1<<(snf-1)), y = x_quant[sfb][bin+2][Abit[i]] & (1<<(snf-1)) and z = x_quant[sfb][bin+3][Abit[i]] & (1<<(snf-1)) , where snf is the significance of the bit. N-tuples progress from low to high frequency within the current coding band and from MSB to LSB.

acod_sign Arithmetic codeword from arithmetic coding sign_bit with the **sign_arith_model** given in Table B.15. sign_bit indicates sign bit for non-zero coefficient. A ‘1’ indicates a negative coefficient, a ‘0’ a positive one. When the bit value of the quantized signal is assigned 1 for the first time, sign bit is arithmetic coded and sent.

Help elements:

cur_snf [i]	current significance of the *i*-th vector. cur_snf[] is initialized to Abit[cband]. See clause 2.3.11.5.
maxsnf	maximum of current significance of the vectors to be decoded. See clause 2.3.11.5.
snf	significance index
last_index	maximum of layer_index values of each channel. See clause 2.3.11.5.

layer_index[ch][layer]	array containing the index of the highest spectral coefficient of band-limit band in each layer for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows
layer_index_offset_long[layer]	table containing the index of the highest spectral coefficient of band-limit band in each layer for long windows. See Table 2.16
layer_index_offset_short[layer]	table containing the index of the highest spectral coefficient of band-limit band in each layer for short windows. See Table 2.17
dim0	dimension of subvector 0
dim1	dimension of subvector 1
sample[ch][i]	quantized spectral coefficients reconstructed from the decoded bit-sliced data of spectral line i in channel ch. See clause 2.3.11.5
prestate[ch][i]	previous state that indicates whether the previously decoded value of frequency line i is 0 or not in channel ch. See clause 3.13.1.3
total_estimated_bits	Total bits estimated to be used for decoding the bitstream. See clause 3.13.1.3
available_bits[layer]	array containing the available bits to be used in the scalability layer. See clause 2.3.11.5.
sign_bit	sign bit for non-zero coefficient. A ‘1’ indicates a negative coefficient, a ‘0’ a positive one. When the bit value of the quantized signal is assigned 1 for the first time, sign bit is arithmetic coded and sent.
layer	scalability layer index
snf	significance of vector to be decoded.
ch	channel index
nch	the number of channel

3.13.1.3 Decoding Process

Decoding of bit-sliced data

In BSAC encoder, a quantized sample is bit-sliced. In order to further reduce the redundancy of bit-sliced data, the vector is formed which consists of successive non-overlapping 4-tuples of the MSB data starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient..

The vectors are divided into two subvectors depending upon the previous states. One- to four-dimensional subvector of bit-sliced sequence are arithmetic coded and transmitted. For the case of multiple windows per block, the concatenated and possibly grouped and interleaved set of spectral coefficients is treated as a single set of coefficients that progress from low to high. This set of spectral coefficients may need to be de-interleaved after they are decoded. The spectral information for all scalefactor bands equal to or greater than **max_sfb** is set to zero.

After all MSB data are encoded from the lowest frequency line to the highest, the same encoding process is repeated until LSB data is encoded.

Four-dimensional vector is the basic unit in encoding/decoding the bit-sliced data. The 4-dimensional vector is made up of two sub-vectors upon the previous states. *prestate[]* indicates the state of the sample whose bit-sliced data has been decoded previously. Before the decoding of the bit-sliced data is started, all previous states are set to 0. The previous states are updated with coding the bit-sliced data of the sample from MSB to LSB. And they are set to 1 when bit-value is non-zero. The dimension of two subvectors depend on the previous state as follows :

```
offset = the frequency line offset of the vector
dim0 = 0 /* the dimension of the subvector 0 */
dim1 = 0 /* the dimension of the subvector 1 */
for (k=0; k<4; k++) {
    if (prestate[offset+k])
       dim1++
    else
       dim0++
}
```

After the dimension of the subvector is determined, the first subvector of the bit-sliced data is decoded with the arithmetic model which depends on the arithmetic model index, the dimension and the previous state value. And, the second vector is decoded.

Detailed arithmetic decoding procedure will be described in this clause. But, the model used for arithmetic decoding should be defined in order to decode the subvector. Arithmetic model of the bit-sliced data relies on the

arithmetic model index of the coding band, the dimension and the significance of the sub-vector and the previous states, as listed in Table B.18 to Table B.48.
The dimension of the subvector and the previous state of the decoded sample can calculated in the previous procedure.
There are 32 arithmetic models which can be used for encoding/decoding the bit-sliced data. In order to transmit the arithmetic model information used in encoding process, the index of arithmetic model is coded and included in the syntax of bsac_side_info(). After the model index is decoded, the decoding of the bit-sliced data shall be started.
The current significance of the sub-vector represent the bit-position of the vector to be decoded. Table 3.13.4 shows the allocated bit of the decoded sample according to the decoded model index. Current significance, *cur_snf[]* of all vector within the coding band are initialized to the allocated bit. For the detailed initialization process, see clause 2.3.11.5.

The sign bits associated with non-zero coefficients follow the arithmetic codeword when the bit-value of the quantized spectral coefficient is 1 for the first time, with a _1_ indicating a negative coefficient and a _0_ indicating a positive one. For example, if an arithmetic codeword has been decoded and the decoded bit-value of the quantized spectral coefficient is 1 for the first time, then immediately following this in the bitstream is

acod_sign

sign_bit of a sample can be arithmetic decoded from the bitstream using sign_arithmetic model given in Table B.15.

Two decoded subvectors of the bit-sliced data need to be de-interleaved and reconstructed to the sample. For the detailed reconstruction of the blt-sliced data, see **Reconstruction of the decoded sample from bit-sliced data** part in clause 2.3.11.2

Arithmetic decoding procedure
The pseudo code fragment shown below describes

- how to decode the arithmetic codeword
- the summary of the arithmetic decoding procedure
- how to use the arithemtic model in arithmetic decoding procedure
- how to estimate the bits necessary for decoding the arithmetic codeword.

```
while (1) {
       if (high < Half) {
          /* nothing */
       }
       else if (low >= Half ) {
          low -= Half;
          high -= Half;
          value -= Half;
       }
       else if (low >= First_qtr &&
          high <Third_qtr) {
          low -= First_qtr;
          high -= First_qtr;
          value -= First_qtr;
       }
       else
          break;

       low = 2*low;         /* Scale up code range. */
       high = 2*high+1;
       value = 2*value;

       value += next_bit(); /* Move in next input bit. */
    }

    range = (long)(high-low) + 1;
    cum = (((long)(value-low)+1)*16384-1)/range; /* Find cum freq */

    for (symbol=0; armodel[symbol]>cum; symbol++);

    /* Narrow the code region to that allotted to this symbol. */
    if (symbol>0) {
       high  = low + (range * armodel[symbol-1])/16384 - 1;
```

```
    }
    low  = low + (range * armodel[symbol])/16384;
```

Symbol is the value decoded from arithmetic codeword. next_bit() is the function that extract 1 bit value from the bit-stream. Half, First_qtr and Third_qtr are defined as 8192, 4096 and 12288 respectively. The frequency range for the *i*th symbol with the exception of the 0^{th} symbol is from armodel[i-1] to armodel[i]. In case of the 0^{th} symbol, the frequency range is from 16384 to armodel[0]. As *i* decreases, armodel[i] increases.

The estimated bits are accumulated in order to calculate the total bits as follows.

```
    if (symbol>0)
       prob = (double)(armodel[symbol-1]-armodel[symbol])/16384;
    else
    prob = (double)(16384-armodel[symbol])/16384;

    estimated_bits += (double)((int)(-log(prob)/log((double)2.)*4096))/4096.;
```

3.13.2 Arithmetic Decoding of stereo_info, ms_used or noise_flag

3.13.2.1 Tool description

The BSAC scalable coding scheme includes the noiseless coding which is different from MPEG-4 AAC coding and further reduce the redundancy of the stereo-related data.
Decoding of the stereo-related data and Perceptual Noise Substitution(pns) data is depended on pns_data_present and ms_mask_present which indicates the stereo mask. Since the decoded data is the same value with MPEG-4 AAC, the MPEG-4 AAC stereo-related and pns processing follows the decoding of the stereo-related data and pns data.

3.13.2.2 Definition

Bit stream elements:

acode_ms_used[g][sfb] arithmetic codeword from the arithmetic coding of ms_used which is one-bit flag per scalefactor band indicating that M/S coding is being used in window group g and scalefactor band sfb, as follows :
0 Independent
1 ms_used

acode_stereo_info[g][sfb] arithmetic codeword from the arithmetic coding of stereo_info which is two-bit flag per scalefactor band indicating that M/S coding or Intensity coding is being used in window group g and scalefactor band sfb, as follows :
00 Independent
01 ms_used
10 Intensity_in_phase
11 Intensity_out_of_phase or noise_flag_is_used
Note : If ms_mask_present is 3, noise_flag_l and noise_flag_r are 0 value, then stereo_info is interpreted as out-of-phase intensity stereo regardless the value of pns_data_present.

acode_noise_flag[g][sfb] arithmetic codeword from the arithmetic coding of noise_flag which is 1-bit flag per scalefactor band indicating whether the perceptual noise substitution is used(1) or not(0) in window group g and scalefactor band sfb.

acode_noise_flag_l[g][sfb] arithmetic codeword from the arithmetic coding of noise_flag_l which is 1-bit flag per scalefactor band indicating whether the perceptual noise substitution is used(1) or not(0) in the left channel, window group g and scalefactor band sfb .

acode_noise_flag_r[g][sfb] arithmetic codeword from the arithmetic coding of noise_flag which is 1-bit flag per scalefactor band indicating whether the perceptual noise substitution is used(1) or not(0) in the right channel, window group g and scalefactor band sfb.

acode_noise_mode[g][sfb] arithmetic codeword from the arithmetic coding of noise_mode which is two-bit flag per scalefactor band indicating that which noise substitution is being used in window group g and scalefactor band sfb, as follows :
00 Noise Subst L+R (independent)
01 Noise Subst L+R (correlated)
10 Noise Subst L+R (correlated, out-of-phase)

11 reserved

Help elements:

ch	channel index
g	group index
sfb	scalefacotor band index within group
layer_sfb[layer]	array containing the index of the lowest scalefactor band to be added newly in the scalability layer for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows. See clause 2.3.11.5
layer_sfb_offset_long[layer]	table containing the index of the lowest scalefactor band to be added newly in the scalability layer for long windows. See Table 2.18.
layer_sfb_offset_short[layer]	table containing the index of the lowest scalefactor band to be added newly in the scalability layer for short windows. See Table 2.19.
num_window_groups	number of groups of windows which share one set of scalefactors. See clause 2.3.11.4.
layer	scalability layer index
nch	the number of channel
ms_mask_present	this two bit field indeicates that the stereo mask is 00 Independent 01 1 bit mask of max_sfb bands of ms_used is located in the layer side information part. 10 All ms_used are ones 11 2 bit mask of max_sfb bands of stereo_info is located in the layer side information part.

3.13.2.3 Decoding Process

Decoding process of stereo_info, noise_flag or ms_used is depended on pns_data_present, number of channel, ms_mask_present. pns_data_present flag is conveyed as a element in syntax of bsac_channel_stream(). pns_data_present indicates whether pns tool is used or not at each frame. ms_mask_present indeicates the stereo mask as follows :

00 Independent
01 1 bit mask of max_sfb bands of ms_used is located in the layer side information part.
10 All ms_used are ones
11 2 bit mask of max_sfb bands of stereo_info is located in the layer side information part.

Decoding process is classified as follows :

- 1 channel, no pns data
  If the number of channel is 1 and pns data is not present, there is no bit-stream elements related to stereo or pns.
- 1 channel, pns data
  If the number of channel is 1 and pns data is present, noise flag of the scalefactor bands between **pns_start_sfb** to **max_sfb** is arithmetic decoded using model shown in Table B.16. Perceptual noise substitution is done according to the decoded noise flag.
- 2 channel, ms_mask_present=0 (Independent), No pns data
  If ms_mask_present is 0 and pns data is not present, arithmetic decoding of stereo_info or ms_used is not needed.
- 2 channel, ms_mask_present=0 (Independent), pns data
  If ms_mask_present is 0 and pns data is present, noise flag for pns is arithmetic decoded using model shown in Table B.16. Perceptual noise substitution of independent mode is done according to the decoded noise flag.
- 2 channel, ms_mask_present=2 (all ms_used), pns data or no pns data
  All ms_used values are ones in this case. So, M/S stereo processing of AAC is done at all scalefactor band. And naturally there can be no pns processing regardless of pns_data_present flag.
- 2 channel, ms_mask_present=1 (optional ms_used), pns data or no pns data
  1 bit mask of max_sfb bands of ms_used is conveyed in this case. So, ms_used is arithmetic decoded using the ms_used model given in Table B.13. M/S stereo processing of AAC is done or not according to the decoded ms_used. And there is no pns processing regardless of pns_data_present flag

- 2 channel, ms_mask_present=3 (optional ms_used/intensity/pns), no pns data

  At first, stereo_info is arithmetic decoded using the stereo_info model given in Table B.14.
  stereo_info is is two-bit flag per scalefactor band indicating that M/S coding or Intensity coding is being used in window group g and scalefactor band sfb as follows :

  00 Independent
  01 ms_used
  10 Intensity_in_phase
  11 Intensity_out_of_phase

  If stereo_info is not 0, M/S stereo or intensity stereo of AAC is done with these decoded data. Since pns data is not present, we don_t have to process pns.
- 2 channel, ms_mask_present=3 (optional ms_used/intensity/pns), pns data

  stereo_info is arithmetic decoded using the stereo_info model given in Table B.14.
  If stereo_info is 1 or 2, M/S stereo or intensity stereo processing of AAC is done with these decoded data and there is no pns processing.
  If stereo_info is 3 and scalefactor band is larger than or equal to pns_start_sfb, noise flag for pns is arithmetic decoded using model given in Table B.16. And then if the both noise flags of two channel are 1, noise substitution mode is arithmetic decoded using model given in Table B.17. The perceptual noise is substituted or out_of_phase intensity stereo processing is done according to the substitution mode. Otherwise, the perceptual noise is substituted only if noise flag is 1.
  If stereo_info is 3 and scalefactor band is smaller than pns_start_sfb, out_of_phase intensity stereo processing is done.

3.13.3 Arithmetic Decoding of scalefactors

3.13.3.1 Tool description

The BSAC scalable coding scheme includes the noiseless coding which is different from AAC and further reduce the redundancy of the scalefactors.

The noiseless coding has two ways to represent the scalefactors. One way is to use coding scheme similar to AAC. The max_scalefactor is coded as an 8 bit unsigned integer. The first scalefactor associated with the quantized spectrum is differentially coded relative to the max_scalefactor value and the arithmetic coded using the differential scalefactor arithmetic model. The remaining scalefactors are differentially coded relative to the previous scalefactor and then Arithmetic coded using the differential scalefactor model.
Another way is the BSAC scalefactor coding method. The max_scalefactor is coded as an 8 bit unsigned integer. The scalefactors are differentially coded relative to the offset value, max_scalefactor, and then arithmetic coded using the scalefactor arithmetic model.

3.13.3.2 Definitions

Bit stream element:

acode_scf[ch][g][sfb] Arithmetic codeword from the coding of the differential scalefactors.
acode_scf_index[ch][g][sfb] Arithmetic codeword from the coding of the index which is converted from the differential scalefactor.
acode_esc_scf_index[ch][g][sfb] Arithmetic codeword from the coding of the escape code for the index which is converted from the differential scalefactor.
acode_dpcm_noise_energy[ch][g][sfb] Arithmetic codeword from the coding of the differential noise energy for PNS.
acode_dpcm_noise_energy_index[ch][g][sfb] Arithmetic codeword from the coding of the index which is converted from the the differential noise energy.
acode_esc_dpcm_noise_energy_index[ch][g][sfb] Arithmetic codeword from the coding of the escape code for the index which is converted from the differential noise energy.
acode_is_position[ch][g][sfb] Arithmetic codeword from the coding of the differential intensiy stereo position.
acode_is_position_index[ch][g][sfb] Arithmetic codeword from the coding of the index which is converted from the differential intensiy stereo position.
acode_esc_is_position_index[ch][g][sfb] Arithmetic codeword from the coding of the escape code for the index which is converted from the differential intensity stereo position.

acode_pcm_noise_energy[ch][g][sfb] Arithmetic codeword from the coding of the noise energy for the first PNS scalefactor band. Arithmetic model for coding pcm_noise_energy is given as follows :

```
arithmetic_model[0] = 16384 - 32;
for (index=1; index<512;  index++)
        arithmetic_model[index] = arithmetic_model[index-1] - 32;
```

Help elements:

ch	channel index
g	group index
sfb	scalefacotor band index within group
layer_sfb[layer]	array containing the index of the lowest scalefactor band to be added newly in the scalability layer for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows. See clause 2.3.11.5
layer_sfb_offset_long[layer]	table containing the index of the lowest scalefactor band to be added newly in the scalability layer for long windows. See Table 2.18
layer_sfb_offset_short[layer]	table containing the index of the lowest scalefactor band to be added newly in the scalability layer for short windows. See Table 2.19.
num_window_groups	number of groups of windows which share one set of scalefactors. See clause 2.3.11.4
layer	scalability layer index
nch	the number of channel
scf_coding[ch]	indicates the decoding method of the scalefactors.

The noiseless coding of the scalefactor requires two constants, ESC_SCF_INDEX and ESC_INDEX whose values are defined as 54. (See bsac_side_info())

3.13.3.3 Decoding Process

The spectral coefficients are divided into scalefactor bands that contain a multiple of 4 quantized spectral coefficients. Each scalefactor band has a scalefactor. The noiseless coding has two ways to represent the scalefactors. **scf_coding[ch]** indicates which method the scalefactor is coded with.

One way is to use coding scheme similar to AAC. For all scalefactors the difference to the preceding value is mapped into new index using Table B.1. If the newly mapped index is smaller than 54, it is arithmetic-coded using the arithmetic model given in Table B.3. Otherwise, the escape value 54 is arithmetic coded using the scalefactor arithmetic model given in Table B.3 and the difference to escape value 54 is arithmetic coded using the arithmetic model given in Table B.4. The initial preceding value is given explicitly as a 8 bit PCM in the bitstream element **max_scalefactor**.
The max_scalefactor is coded as an 8 bit unsigned integer. The first scalefactor associated with the quantized spectrum is differentially coded relative to the max_scalefactor value and arithmetic coded using the differential scalefactor arithmetic model. The remaining scalefactors are differentially coded relative to the previous scalefactor and then arithmetic coded using the differential scalefactor model, as shown in Table 3.13.2.

A second way is BSAC scalefactor coding method. For all scalefactors the difference to the offset value is arithmetic-coded using the arithmetic model, as shown in Table 3.13.3. The arithmetic model used for coding differential scalefactors is given as a 2-bit unsigned integer in the bitstream element, **scalefactor_model**. The offset value is given explicitly as a 8 bit PCM in the bitstream element **max_scalefactor**.

The following pseudo code describes how to decode the scalefactors *sf[ch][g][sfb]* in base layer and each enhancement layer:

```
for (ch=0; ch<nch; ch++) {
  if (scf_coding[ch]==1) {
     for (g=0; g<num_window_group; g++) {
        for( sfb=layer_sfb[layer]; sfb<layer_sfb[layer+1]; sfb++ ) {
```

```
            diff_scf = arithmetic_decoding();
            sf[ch][g][sfb] = max_scalefactor – diff_scf;
         }
      }
   }
   else {
      for (g=0; g<num_window_group; g++) {
         for( sfb=layer_sfb[layer]; sfb<layer_sfb[layer+1]; sfb++ ) {
         diff_scf_index = arithmetic_decoding();
            if (diff_scf_index==ESC_SCF_INDEX) {
               esc_scf_index = arithmetic_decoding();
               diff_scf_index += esc_scf_index;
            }
            if (sfb==0)
                scf_index = max_scalefactor – diff_scf_index;
            else
                scf_index = sf[ch][g][sfb-1] – diff_scf_index;
            sf[ch][g][sfb] = index2sf[scf_index];
         }
      }
   }
 }
```

where, layer_sfb[layer] is the start scalefactor band and layer_sfb[layer+1] is the end scalefactor band for decoding the scalefactors in each layer.

3.13.4 Arithmetic Decoding of arithmetic model index

3.13.4.1 Tool description

In BSAC scalable coding scheme, the spectral coefficients are divided into coding bands which contain 32 quantized spectral coefficients for the noiseless coding. Coding bands are the basic units used for the noiseless coding. The set of bit-sliced sequence is divided into coding bands. The arithmetic model index for encoding the bit-sliced data within each coding band is transmitted starting from the lowest frequency coding band and progressing to the highest frequency coding band. For all arithmetic model indexes the difference to the offset value is arithmetic-coded using the arithmetic model **ArModel_model,** as shown in Table 3.13.3.

3.13.4.2 Definition

Bit stream element:

acode_ArModel[ch][cband] Arithmetic codeword from the arithmetic coding of arithmeticmodel index for coding-band cband.

Help elements:

g	group index
sfb	scalefacotor band index within group
layer_sfb[layer]	array containing the index of the lowest scalefactor band to be added newly in the scalability layer for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows
num_window_group	number of groups of windows which share one set of scalefactors
swb_offset[sfb]	array containing the index of the lowest spectral coefficient of scalefactor band sfb for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows
cband	coding band index within group
index2cb(ch, i)	function returning coding band which the index of the spectral coefficient i is mapped into by the mapping table, index2cband[][].
layer	scalability layer index
ch	channel index
nch	the number of channel

3.13.4.3 Decoding Process

For all arithmetic model indexes the difference to the offset value is arithmetic-coded using the arithmetic model **ArModel_model,** as shown in Table 3.13.3. The arithmetic model used for coding differential arithmetic model index is given as a 2-bit unsigned integer in the bitstream element, **ArModel_model**. The offset value is given explicitly as a 5 bit PCM in the bitstream element **min_ArModel**.

The following pseudo code describes how to decode the arithmetic model index *ArModel[cband]* in base layer and each enhancement layer:

```
for (ch=0; ch<nch; ch++)
  for( sfb=layer_sfb[layer]; sfb<layer_sfb[layer+1]; sfb++ )
     for (g=0; g<num_window_groups; g++) {
        band = (sfb * num_window_groups) + g
        for (i=0; swb_offset[band]; i<swb_offset[band+1]; i+=4) {
           cband = index2cb(ch, i);
           if (!decode_cband[ch][cband]) {
              ArModel[ch][cband] = min_ArModel + arithmetic_decoding();
              decode_cband[ch][cband] = 1;
           }
        }
     }
```

where, layer_sfb[layer] is the start scalefactor band and layer_sfb[layer+1] is the end scalefactor band for decoding the arithmetic model index in each layer, and decode_cband[ch][cband] is flag indicating whether the arithmetic model has been decoded (1) or not (0).

3.13.5 Tables

Table 3.13.1 Arithmetic Model 0 of mapped Scalefactor

Model Number	Dimension of Codebook	Range of values	Model listed in Table
0	1	0 to 54	B.3
1	1	0 to 66	B.4

Table 3.13.2 Arithmetic Model 1 of Differential Scalefactor

Model Number	Largest Differential Scalefactor	Model listed in Table
0	7	B.5
1	15	B.6
2	31	B.7
3	63	B.8

Table 3.13.3 Arithmetic Model of Differential ArModel

Model Number	Largest Differential ArModel	Model listed in Table
0	3	B.9
1	7	B.10
2	15	B.11
3	31	B.12

Table 3.1.34 BSAC Arithmetic Model Parameters

Arithmetic Model index	allocated bit / sample within coding band	Model listed in Table	Arithmetic model index	allocated bit / sample within coding band	Model listed in Table

0	0	B.18	16	8	B.33
1	-	not used	17	8	B.34
2	1	B.19	18	9	B.35
3	1	B.20	19	9	B.36
4	2	B.21	20	10	B.37
5	2	B.22	21	10	B.38
6	3	B.23	22	11	B.39
7	3	B.24	23	11	B.40
8	4	B.25	24	12	B.41
9	4	B.26	25	12	B.42
10	5	B.27	26	13	B.43
11	5	B.28	27	13	B.44
12	6	B.29	28	14	B.45
13	6	B.30	29	14	B.46
14	7	B.31	30	15	B.47
15	7	B.32	31	15	B.48

3.14 Perceptual Noise Substitution (PNS)

3.14.1 Tool description

This tool is used to implement perceptual noise substitution coding within an ICS. Thus, certain sets of spectral coefficients are derived from random vectors rather than from Huffman coded symbols and an inverse quantization process. This is done selectively on a scalefactor band and group basis when perceptual noise substitution is flagged as active.

3.14.2 Definitions

hcod_sf[] Huffman codeword from the Huffman code table used for coding of scalefactors (see ISO 13818-7 clause 4.2)

dpcm_noise_nrg[][] Differentially encoded noise energy
noise_nrg[group][sfb] Noise energy for each group and scalefactor band
spec[] Array containing the channel spectrum of the respective channel

3.14.3 Decoding Process

The use of the percpetual noise substitution tool is signaled by the use of the pseudo codebook NOISE_HCB (13).

Furthermore, if the same scalefactor band and group is coded by perceptual noise substitution in both channels of a channel pair, the correlation of the noise signal can be controlled by means of the ms_used field: While the default noise generation process works independently for each channel (separate generation of random vectors), the same random vector is used for both channels if ms_used[] is set for a particular scalefactor band and group. In this case, no M/S stereo coding is carried out (because M/S stereo coding and noise substitution coding are mutually exclusive).

The energy information for percpetual noise substitution decoding is represented by a "noise energy" value indicating the overall power of the substituted spectral coefficients in steps of 1.5 dB. If noise substitution coding is active for a particular group and scalefactor band, a noise energy value is transmitted instead of the scalefactor of the respective channel.

Noise energies are coded just like scalefactors, i.e. by Huffman coding of differential values:

- the start value for the DPCM decoding is given by global_gain.
- Differential decoding is done separately between scalefactors, intensity stereo positions and noise energies. In other words, the noise energy decoder ignores interposed scalefactors and intensity stereo position values and vice versa (see ISO 13818-7 clause 6.3.2)

The same codebook is used for coding of noise energies as for scalefactors.

One pseudo function is defined for use in perceptual noise substitution decoding:

```
function is_noise(group,sfb) {
   1    for window groups / scalefactor bands with
          codebook sfb_cb[group][sfb] == NOISE_HCB
   0    otherwise
   }
```

The noise substitution decoding process for one channel is defined by the following pseudo code:

```
nrg = global_gain - NOISE_OFFSET - 256;
for (g=0; g<num_window_groups; g++)  {

   /* Decode noise energies for this group */
   for (sfb=0; sfb<max_sfb; sfb++)
      if (is_noise(g,sfb))
         noise_nrg[g][sfb] = nrg += dpcm_noise_nrg[g][sfb];

   /* Do perceptual noise substitution decoding */
   for (b=0; b<window_group_length[g]; b++)  {
      for (sfb=0; sfb<max_sfb; sfb++)  {
         if (is_noise(g,sfb))  {

            offs = swb_offset[sfb];
            size = swb_offset[sfb+1] - offs;

            /* Generate random vector */
            gen_rand_vector( &spec[g][b][sfb][0], size );
            scale = 1/(size * sqrt(MEAN_NRG));
            scale *= 2.0^(0.25*noise_nrg [g][sfb]);
            /* Scale random vector to desired target energy */
            for (i=0; i<len; i++)
               spec[g][b][sfb][i] *= scale;

         }

      }
   }

}
```

The constant NOISE_OFFSET is used to adapt the range of average noise energy values to the usual range of scalefactors and has a value of 90.

The function gen_rand_vector(addr, size) generates a vector of length <size> with signed random values of average energy MEAN_NRG per random value. A suitable random number generator can be realized using one multiplication/accumulation per random value.

3.14.4 Diagrams

3.14.5 Tables

3.14.6 Integration with Intra Channel Prediction Tools

For scalefactor bands coded using PNS the corresponding predictors are switched to “off“, thus effectively overriding the status specified by the prediction_used mask. In addition, for scalefactor bands coded by perceptual noise substitution the predictors belonging to the corresponding spectral coefficients are reset (see ISO 13818-7 clause 8.3.3). The update of these predictors is done by feeding a value of zero as the "last quantized value" $x_{rec(n-1)}$.

In Long Term Prediction, the scalefactor bands coded using PNS are not predicted.

3.14.7 Integration with other AAC Tools

The following interactions between the perceptual noise substitution tool and other AAC tools take place:

Definition of a new pseudo Huffman codebook number NOISE_HCB = 13

- During Huffman decoding of the quantized spectral coefficients, the Huffman codebook table NOISE_HCB is treated exactly like the zero codebook ZERO_HCB, i.e. no Huffman codewords are read for the corresponding scalefactor band and group.
- If the same scalefactor band and group is coded by perceptual noise substitution in both channels of a channel pair, no M/S stereo decoding is carried out for this scalefactor band and group .
- The pseudo noise components generated by the perceptual noise substitution tool are injected into the output spectrum prior to the temporal noise shaping (TNS) processing step.

3.14.8 Integration into a Scalable AAC-based Coder

The following rules apply for usage of the perceptual noise substitution tool in a scalable AAC-based coder:

If a particular scalefactor band and group is coded by perceptual noise substitution, its contribution to the spectral components of the reconstructed output signal for the update of the intra channel predictor is omitted.

- If a particular scalefactor band and group is coded by perceptual noise substitution, its contribution to the spectral components of the output signal is omitted if spectral coefficients are transmitted for this scalefactor band and group in any of the higher (enhancement) layers by means of a non-zero codebook number (i.e. a Huffman codebook != ZERO_HCB).
- If a particular scalefactor band and group is coded by perceptual noise substitution in both channels of a channel pair, the higher (enhancement) layers may still use the M/S stereo flag ms_used[][] to signal the use of M/S stereo decoding.

3.15 Frequency Selective Switch Module

3.15.1 Definitions

dc_group	Four consecutive scalefactor bands if the window type is not SHORT_WINDOW. One band of diff_short_lines, if the window type is SHORT_WINDOW.
no_of_dc_groups	If the window type is not SHORT_WINDOW, the number of groups depending on the sampling frequency is given in table 2 below. If the window type is SHORT_WINDOW, no_of_dc_groups is ‘1’.
diff_short_lines	Only used, if the window type is SHORT_WINDOW. The number of spectral lines in the single dc_group per window, depending on the sampling rate, is given in table 2 below.
diff_control[w][dc_group]	for each window and dc_group the switch control information for one dc_group. If the window type is not SHORT_WINDOW **diff_control[w][dc_group]** is huffman encoded using table 1 in the bitstream.
diff_control_sfb[w][sfb]	Only applies, if the window type is not SHORT_WINDOW. The decoded **diff_control[w][dc_group]** values; 1 bit or each scale factor band.

The Inverse Frequency Selective Switching Unit (IFFS) connects one of two input signals to the output, depending on diff_control[w][dc_group].

In the bitstream diff_control[w][dc_group] is huffman encoded using the following table:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Code	0	20	21	22	23	24	25	8	9	26	27	28	29	30	31	1
Length	2	5	5	5	5	5	5	4	4	5	5	5	5	5	5	2

Table 1: diff_control_hc_tab

Sampling Rate	96	88.2	64	48	44.1	32	24	22.05	16	12	11.025	8
Core Sampling Rate	8	7.35	8	8	7.35	8	8	7.35	8	12	11.025	8
no_of_dc_groups	4	4	5	5	5	6	8	8	8	10	10	9
diff_short_lines	8	8	13	18	18	26	36	36	54	104	104	104

Table 2: number of dc_groups for each sampling rate

3.15.2 Decoding Process

Decoding if the window type is not SHORT_WINDOW:

After huffman decoding diff_control[w][dc_group] from the bitstream, the array diff_control_sfb[w][sfb] is generated according to:

```
if ( ! SHORT_WINDOW ) {
   dc_group = 0;
   while( dc_group < no_of_ dc_groups ) {
       for( i=0; i<4; i++ ) {
           diff_control_sfb[0][dc_group*4+i] = diff_control[0][dc_group]  & 0x8;
           diff_control[0][dc_group] <<= 1;
       }
   }
}
```

For all scale factor bands which did not get a value assigned in diff_control_sfb[w][sfb] in the above procedure, diff_control_sfb[w][sfb] is set to ‘1’;

Finally the switching for all scale factor bands is done according to:

```
if ( diff_control_sfb[w][sfb] == 0 ) {
   spec_out[w][sfb] = spec_requantized[w][sfb] + spec_core[w][sfb];
} else {
   spec_out[w][sfb] = spec_requantized[w][sfb];
}
```

Decoding if the window type is SHORT_WINDOW:

If the window type is SHORT_WINDOW, there is only one band of diff_short_lines per window, where the diff control mechanism is applied.

For spectral lines #0 to #diff_short_lines:

```
if ( diff_control_sfb[w][0] == 0 ) {
   spec_out[w] = spec_requantized[w] - spec_core[w];
} else {
   spec_out[w] = spec_requantized[w];
}
```

For the remaining lines the output of the switch is identical to the input:

```
   spec_out[w] = spec_requantized[w];
```

3.16 Upsampling Filter Tool

3.16.1 Tool Description

The Upsampling filter tool is used to adapt the sampling rate of the core coder to the sampling rate of the time/frequency coder. The upsampling filter is implemented based on the MDCT filterbank of the AAC-derived encoder, which is used without any changes. This filterbank is very similar to the IMDCT filterbank already used in the decoder.

The filterbank takes a block of time samples of the core coder output and inserts an appropriate number of zeroes between these samples to generate a signal at the desired higher sampling rate. The upsampled samples are then modulated by an appropriate window function, and the MDCT is performed. Each block of input samples is overlapped by 50% with the immediately preceding block and the following block. The transform input block length N is set to either 2048 (1920) or 256 (240) samples. The filterbank is switched synchronously to the IMDCT using both window_sequence and window_shape.

The output of the filterbank is connected to the FSS module, which only uses the output values in the FSS-bands. Since the upper FSS band doesn't exceed half of the lower sampling rate, there are no aliasing effects.

3.16.2 Definitions

The syntax elements for the filterbank are identical to those used for the IMDCT filterbank. They consist of the control information **window_sequence** and **window_shape**.

window_sequence	2 bit indicating which window sequence (i.e. block size) is used (see 1.3, Table 6.11).
window_shape	1 bit indicating which window function is selected (see 1.3, Table 6.11).
up-sampling-factor	ratio of T/F coder sampling rate and core coder sampling rate.
$x_{\text{in-mdct-core}}$	temporal data field, which is used to hold the up-sampled input to the MDCT filterbank
$x_{\text{out-core}}[i]$	Output samples of the core decoder

3.16.3 Decoding Process

The analysis window length N for the transform is a function of the syntax element **window_sequence** and the algorthmic context. It is derived in an identical way to the procedure described for the Filterbank and Blockswitching tool.

3.16.3.1 Upsampling by insertion of zeroes

The input to the filterbank is generated by :

$x_{\text{in-mdct-core}}[k] = 0$ for k = [0 : N/2-1]

$x_{\text{in-mdct-core}}[\text{up-sampling-factor}*i] = x_{\text{out-core}}[i]$ for i = [0 : N/2/up-sampling-factor-1]

3.16.3.2 Windowing and block switching

The adaptation of the time-frequency resolution of the filterbank is done by shifting between transforms whose input lengths are either 2048 (1920) or 256 (240) samples, synchronously to the decoder IMDCT filterbank. The selection between the 2048/256 or the 1920/240 pairs is done depending on the frame length of the core coder. The 1920/240 pair is used for all core coders having a frame length of a multiple of 10 ms.

The windowed time domain values can be calculated in using exactly the same windows w(n) as defined for the IMDCT filterbank.

The windowed coefficients are calculated by

$$z_{i,n} = w(n) \cdot x'_{\text{in\_mdct\_core}}(n);$$

Finally the windowed coefficients are transformed with the MDCT as defined in the following paragraph.

3.16.3.3 MDCT

The spectral coefficient, $X_{i,k}$, are defined as follows:

$$X_{i,k} = 2 \cdot \sum_{n=0}^{N-1} z_{i,n} \cos\left(\frac{2\pi}{N}\left(n + n_0\right)\left(k + \frac{1}{2}\right)\right) \text{ for } 0 \le k < N/2 .$$

where:

z_{in} = windowed input sequence

n = sample index

k = spectral coefficient index

i = block index

N = window length of the one transform window based on the window_ sequence value

n_0 = (N / 2 + 1) / 2

Only the output values from 0 to N/ 2/ up-sampling-factor-1can be used without aliasing distortions. This is ensured by the subsequently following FSS module.

4

Annex A

Table A.1 – Scalefactor Huffman Codebook

index	length	codeword (hexadecimal)	index	length	codeword (hexadecimal)
0	18	3ffe8	61	4	a
1	18	3ffe6	62	4	c
2	18	3ffe7	63	5	1b
3	18	3ffe5	64	6	39
4	19	7fff5	65	6	3b
5	19	7fff1	66	7	78
6	19	7ffed	67	7	7a
7	19	7fff6	68	8	f7
8	19	7ffee	69	8	f9
9	19	7ffef	70	9	1f6
10	19	7fff0	71	9	1f9
11	19	7fffc	72	10	3f4
12	19	7fffd	73	10	3f6
13	19	7ffff	74	10	3f8
14	19	7fffe	75	11	7f5
15	19	7fff7	76	11	7f4
16	19	7fff8	77	11	7f6
17	19	7fffb	78	11	7f7
18	19	7fff9	79	12	ff5
19	18	3ffe4	80	12	ff8
20	19	7fffa	81	13	1ff4
21	18	3ffe3	82	13	1ff6
22	17	1ffef	83	13	1ff8
23	17	1fff0	84	14	3ff8
24	16	fff5	85	14	3ff4
25	17	1ffee	86	16	fff0
26	16	fff2	87	15	7ff4
27	16	fff3	88	16	fff6
28	16	fff4	89	15	7ff5
29	16	fff1	90	18	3ffe2
30	15	7ff6	91	19	7ffd9
31	15	7ff7	92	19	7ffda
32	14	3ff9	93	19	7ffdb
33	14	3ff5	94	19	7ffdc
34	14	3ff7	95	19	7ffdd
35	14	3ff3	96	19	7ffde
36	14	3ff6	97	19	7ffd8
37	14	3ff2	98	19	7ffd2
38	13	1ff7	99	19	7ffd3
39	13	1ff5	100	19	7ffd4
40	12	ff9	101	19	7ffd5
41	12	ff7	102	19	7ffd6
42	12	ff6	103	19	7fff2
43	11	7f9	104	19	7ffdf
44	12	ff4	105	19	7ffe7
45	11	7f8	106	19	7ffe8
46	10	3f9	107	19	7ffe9
47	10	3f7	108	19	7ffea
48	10	3f5	109	19	7ffeb
49	9	1f8	110	19	7ffe6
50	9	1f7	111	19	7ffe0

51	8	fa	112	19	7ffe1
52	8	f8	113	19	7ffe2
53	8	f6	114	19	7ffe3
54	7	79	115	19	7ffe4
55	6	3a	116	19	7ffe5
56	6	38	117	19	7ffd7
57	5	1a	118	19	7ffec
58	4	b	119	19	7fff4
59	3	4	120	19	7fff3
60	1	0			

Table A.2 – Spectrum Huffman Codebook 1

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword** (hexadecimal)
0	11	7f8	41	5	14
1	9	1f1	42	7	65
2	11	7fd	43	5	16
3	10	3f5	44	7	6d
4	7	68	45	9	1e9
5	10	3f0	46	7	63
6	11	7f7	47	9	1e4
7	9	1ec	48	7	6b
8	11	7f5	49	5	13
9	10	3f1	50	7	71
10	7	72	51	9	1e3
11	10	3f4	52	7	70
12	7	74	53	9	1f3
13	5	11	54	11	7fe
14	7	76	55	9	1e7
15	9	1eb	56	11	7f3
16	7	6c	57	9	1ef
17	10	3f6	58	7	60
18	11	7fc	59	9	1ee
19	9	1e1	60	11	7f0
20	11	7f1	61	9	1e2
21	9	1f0	62	11	7fa
22	7	61	63	10	3f3
23	9	1f6	64	7	6a
24	11	7f2	65	9	1e8
25	9	1ea	66	7	75
26	11	7fb	67	5	10
27	9	1f2	68	7	73
28	7	69	69	9	1f4
29	9	1ed	70	7	6e
30	7	77	71	10	3f7
31	5	17	72	11	7f6
32	7	6f	73	9	1e0
33	9	1e6	74	11	7f9
34	7	64	75	10	3f2
35	9	1e5	76	7	66
36	7	67	77	9	1f5
37	5	15	78	11	7ff
38	7	62	79	9	1f7
39	5	12	80	11	7f4
40	1	0			

Table A.3 – Spectrum Huffman Codebook 2

index	length	codeword (hexadecimal)	index	length	codeword (hexadecimal)
0	9	1f3	41	5	7
1	7	6f	42	6	1d
2	9	1fd	43	5	b
3	8	eb	44	6	30
4	6	23	45	8	ef
5	8	ea	46	6	1c
6	9	1f7	47	7	64
7	8	e8	48	6	1e
8	9	1fa	49	5	c
9	8	f2	50	6	29
10	6	2d	51	8	f3
11	7	70	52	6	2f
12	6	20	53	8	f0
13	5	6	54	9	1fc
14	6	2b	55	7	71
15	7	6e	56	9	1f2
16	6	28	57	8	f4
17	8	e9	58	6	21
18	9	1f9	59	8	e6
19	7	66	60	8	f7
20	8	f8	61	7	68
21	8	e7	62	9	1f8
22	6	1b	63	8	ee
23	8	f1	64	6	22
24	9	1f4	65	7	65
25	7	6b	66	6	31
26	9	1f5	67	4	2
27	8	ec	68	6	26
28	6	2a	69	8	ed
29	7	6c	70	6	25
30	6	2c	71	7	6a
31	5	a	72	9	1fb
32	6	27	73	7	72
33	7	67	74	9	1fe
34	6	1a	75	7	69
35	8	f5	76	6	2e
36	6	24	77	8	f6
37	5	8	78	9	1ff
38	6	1f	79	7	6d
39	5	9	80	9	1f6
40	3	0			

Table A.4 – Spectrum Huffman Codebook 3

index	length	codeword (hexadecimal)	index	length	codeword (hexadecimal)
0	1	0	41	10	3ef
1	4	9	42	9	1f3
2	8	ef	43	9	1f4
3	4	b	44	11	7f6
4	5	19	45	9	1e8
5	8	f0	46	10	3ea
6	9	1eb	47	13	1ffc

7	9	1e6	48	8	f2
8	10	3f2	49	9	1f1
9	4	a	50	12	ffb
10	6	35	51	10	3f5
11	9	1ef	52	11	7f3
12	6	34	53	12	ffc
13	6	37	54	8	ee
14	9	1e9	55	10	3f7
15	9	1ed	56	15	7ffe
16	9	1e7	57	9	1f0
17	10	3f3	58	11	7f5
18	9	1ee	59	15	7ffd
19	10	3ed	60	13	1ffb
20	13	1ffa	61	14	3ffa
21	9	1ec	62	16	ffff
22	9	1f2	63	8	f1
23	11	7f9	64	10	3f0
24	11	7f8	65	14	3ffc
25	10	3f8	66	9	1ea
26	12	ff8	67	10	3ee
27	4	8	68	14	3ffb
28	6	38	69	12	ff6
29	10	3f6	70	12	ffa
30	6	36	71	15	7ffc
31	7	75	72	11	7f2
32	10	3f1	73	12	ff5
33	10	3eb	74	16	fffe
34	10	3ec	75	10	3f4
35	12	ff4	76	11	7f7
36	5	18	77	15	7ffb
37	7	76	78	12	ff7
38	11	7f4	79	12	ff9
39	6	39	80	15	7ffa
40	7	74			

Table A.5 – Spectrum Huffman Codebook 4

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword (hexadecimal)**
0	4	7	41	7	6b
1	5	16	42	8	e3
2	8	f6	43	7	69
3	5	18	44	9	1f3
4	4	8	45	8	eb
5	8	ef	46	8	e6
6	9	1ef	47	10	3f6
7	8	f3	48	7	6e
8	11	7f8	49	7	6a
9	5	19	50	9	1f4
10	5	17	51	10	3ec
11	8	ed	52	9	1f0
12	5	15	53	10	3f9
13	4	1	54	8	f5
14	8	e2	55	8	ec
15	8	f0	56	11	7fb
16	7	70	57	8	ea
17	10	3f0	58	7	6f

18	9	1ee	59	10	3f7
19	8	f1	60	11	7f9
20	11	7fa	61	10	3f3
21	8	ee	62	12	fff
22	8	e4	63	8	e9
23	10	3f2	64	7	6d
24	11	7f6	65	10	3f8
25	10	3ef	66	7	6c
26	11	7fd	67	7	68
27	4	5	68	9	1f5
28	5	14	69	10	3ee
29	8	f2	70	9	1f2
30	4	9	71	11	7f4
31	4	4	72	11	7f7
32	8	e5	73	10	3f1
33	8	f4	74	12	ffe
34	8	e8	75	10	3ed
35	10	3f4	76	9	1f1
36	4	6	77	11	7f5
37	4	2	78	11	7fe
38	8	e7	79	10	3f5
39	4	3	80	11	7fc
40	4	0			

Table A.6 – Spectrum Huffman Codebook 5

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword** (hexadecimal)
0	13	1fff	41	4	a
1	12	ff7	42	7	71
2	11	7f4	43	8	f3
3	11	7e8	44	11	7e9
4	10	3f1	45	11	7ef
5	11	7ee	46	9	1ee
6	11	7f9	47	8	ef
7	12	ff8	48	5	18
8	13	1ffd	49	4	9
9	12	ffd	50	5	1b
10	11	7f1	51	8	eb
11	10	3e8	52	9	1e9
12	9	1e8	53	11	7ec
13	8	f0	54	11	7f6
14	9	1ec	55	10	3eb
15	10	3ee	56	9	1f3
16	11	7f2	57	8	ed
17	12	ffa	58	7	72
18	12	ff4	59	8	e9
19	10	3ef	60	9	1f1
20	9	1f2	61	10	3ed
21	8	e8	62	11	7f7
22	7	70	63	12	ff6
23	8	ec	64	11	7f0
24	9	1f0	65	10	3e9
25	10	3ea	66	9	1ed
26	11	7f3	67	8	f1
27	11	7eb	68	9	1ea
28	9	1eb	69	10	3ec

29	8	ea	70	11	7f8
30	5	1a	71	12	ff9
31	4	8	72	13	1ffc
32	5	19	73	12	ffc
33	8	ee	74	12	ff5
34	9	1ef	75	11	7ea
35	11	7ed	76	10	3f3
36	10	3f0	77	10	3f2
37	8	f2	78	11	7f5
38	7	73	79	12	ffb
39	4	b	80	13	1ffe
40	1	0			

Table A.7 – Spectrum Huffman Codebook 6

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword** (hexadecimal)
0	11	7fe	41	4	3
1	10	3fd	42	6	2f
2	9	1f1	43	7	73
3	9	1eb	44	9	1fa
4	9	1f4	45	9	1e7
5	9	1ea	46	7	6e
6	9	1f0	47	6	2b
7	10	3fc	48	4	7
8	11	7fd	49	4	1
9	10	3f6	50	4	5
10	9	1e5	51	6	2c
11	8	ea	52	7	6d
12	7	6c	53	9	1ec
13	7	71	54	9	1f9
14	7	68	55	8	ee
15	8	f0	56	6	30
16	9	1e6	57	6	24
17	10	3f7	58	6	2a
18	9	1f3	59	6	25
19	8	ef	60	6	33
20	6	32	61	8	ec
21	6	27	62	9	1f2
22	6	28	63	10	3f8
23	6	26	64	9	1e4
24	6	31	65	8	ed
25	8	eb	66	7	6a
26	9	1f7	67	7	70
27	9	1e8	68	7	69
28	7	6f	69	7	74
29	6	2e	70	8	f1
30	4	8	71	10	3fa
31	4	4	72	11	7ff
32	4	6	73	10	3f9
33	6	29	74	9	1f6
34	7	6b	75	9	1ed
35	9	1ee	76	9	1f8
36	9	1ef	77	9	1e9
37	7	72	78	9	1f5
38	6	2d	79	10	3fb
39	4	2	80	11	7fc

40	4	0			

Table A.8 – Spectrum Huffman Codebook 7

index	length	codeword (hexadecimal)	index	length	codeword (hexadecimal)
0	1	0	32	8	f3
1	3	5	33	8	ed
2	6	37	34	9	1e8
3	7	74	35	9	1ef
4	8	f2	36	10	3ef
5	9	1eb	37	10	3f1
6	10	3ed	38	10	3f9
7	11	7f7	39	11	7fb
8	3	4	40	9	1ed
9	4	c	41	8	ef
10	6	35	42	9	1ea
11	7	71	43	9	1f2
12	8	ec	44	10	3f3
13	8	ee	45	10	3f8
14	9	1ee	46	11	7f9
15	9	1f5	47	11	7fc
16	6	36	48	10	3ee
17	6	34	49	9	1ec
18	7	72	50	9	1f4
19	8	ea	51	10	3f4
20	8	f1	52	10	3f7
21	9	1e9	53	11	7f8
22	9	1f3	54	12	ffd
23	10	3f5	55	12	ffe
24	7	73	56	11	7f6
25	7	70	57	10	3f0
26	8	eb	58	10	3f2
27	8	f0	59	10	3f6
28	9	1f1	60	11	7fa
29	9	1f0	61	11	7fd
30	10	3ec	62	12	ffc
31	10	3fa	63	12	fff

Table A.9 – Spectrum Huffman Codebook 8

index	length	codeword (hexadecimal)	index	length	codeword (hexadecimal)
0	5	e	32	7	71
1	4	5	33	6	2b
2	5	10	34	6	2d
3	6	30	35	6	31
4	7	6f	36	7	6d
5	8	f1	37	7	70
6	9	1fa	38	8	f2
7	10	3fe	39	9	1f9
8	4	3	40	8	ef
9	3	0	41	7	68
10	4	4	42	6	33
11	5	12	43	7	6b
12	6	2c	44	7	6e
13	7	6a	45	8	ee

14	7	75	46	8	f9
15	8	f8	47	10	3fc
16	5	f	48	9	1f8
17	4	2	49	7	74
18	4	6	50	7	73
19	5	14	51	8	ed
20	6	2e	52	8	f0
21	7	69	53	8	f6
22	7	72	54	9	1f6
23	8	f5	55	9	1fd
24	6	2f	56	10	3fd
25	5	11	57	8	f3
26	5	13	58	8	f4
27	6	2a	59	8	f7
28	6	32	60	9	1f7
29	7	6c	61	9	1fb
30	8	ec	62	9	1fc
31	8	fa	63	10	3ff

Table A.10 – Spectrum Huffman Codebook 9

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword** (hexadecimal)
0	1	0	85	12	fda
1	3	5	86	12	fe3
2	6	37	87	12	fe9
3	8	e7	88	13	1fe6
4	9	1de	89	13	1ff3
5	10	3ce	90	13	1ff7
6	10	3d9	91	11	7d3
7	11	7c8	92	10	3d8
8	11	7cd	93	10	3e1
9	12	fc8	94	11	7d4
10	12	fdd	95	11	7d9
11	13	1fe4	96	12	fd3
12	13	1fec	97	12	fde
13	3	4	98	13	1fdd
14	4	c	99	13	1fd9
15	6	35	100	13	1fe2
16	7	72	101	13	1fea
17	8	ea	102	13	1ff1
18	8	ed	103	13	1ff6
19	9	1e2	104	11	7d2
20	10	3d1	105	10	3d4
21	10	3d3	106	10	3da
22	10	3e0	107	11	7c7
23	11	7d8	108	11	7d7
24	12	fcf	109	11	7e2
25	12	fd5	110	12	fce
26	6	36	111	12	fdb
27	6	34	112	13	1fd8
28	7	71	113	13	1fee
29	8	e8	114	14	3ff0
30	8	ec	115	13	1ff4
31	9	1e1	116	14	3ff2
32	10	3cf	117	11	7e1
33	10	3dd	118	10	3df

34	10	3db	119	11	7c9
35	11	7d0	120	11	7d6
36	12	fc7	121	12	fca
37	12	fd4	122	12	fd0
38	12	fe4	123	12	fe5
39	8	e6	124	12	fe6
40	7	70	125	13	1feb
41	8	e9	126	13	1fef
42	9	1dd	127	14	3ff3
43	9	1e3	128	14	3ff4
44	10	3d2	129	14	3ff5
45	10	3dc	130	12	fe0
46	11	7cc	131	11	7ce
47	11	7ca	132	11	7d5
48	11	7de	133	12	fc6
49	12	fd8	134	12	fd1
50	12	fea	135	12	fe1
51	13	1fdb	136	13	1fe0
52	9	1df	137	13	1fe8
53	8	eb	138	13	1ff0
54	9	1dc	139	14	3ff1
55	9	1e6	140	14	3ff8
56	10	3d5	141	14	3ff6
57	10	3de	142	15	7ffc
58	11	7cb	143	12	fe8
59	11	7dd	144	11	7df
60	11	7dc	145	12	fc9
61	12	fcd	146	12	fd7
62	12	fe2	147	12	fdc
63	12	fe7	148	13	1fdc
64	13	1fe1	149	13	1fdf
65	10	3d0	150	13	1fed
66	9	1e0	151	13	1ff5
67	9	1e4	152	14	3ff9
68	10	3d6	153	14	3ffb
69	11	7c5	154	15	7ffd
70	11	7d1	155	15	7ffe
71	11	7db	156	13	1fe7
72	12	fd2	157	12	fcc
73	11	7e0	158	12	fd6
74	12	fd9	159	12	fdf
75	12	feb	160	13	1fde
76	13	1fe3	161	13	1fda
77	13	1fe9	162	13	1fe5
78	11	7c4	163	13	1ff2
79	9	1e5	164	14	3ffa
80	10	3d7	165	14	3ff7
81	11	7c6	166	14	3ffc
82	11	7cf	167	14	3ffd
83	11	7da	168	15	7fff
84	12	fcb			

Table A.11 – Spectrum Huffman Codebook 10

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword** (hexadecimal)
0	6	22	85	9	1c7

1	5	8	86	9	1ca
2	6	1d	87	9	1e0
3	6	26	88	10	3db
4	7	5f	89	10	3e8
5	8	d3	90	11	7ec
6	9	1cf	91	9	1e3
7	10	3d0	92	8	d2
8	10	3d7	93	8	cb
9	10	3ed	94	8	d0
10	11	7f0	95	8	d7
11	11	7f6	96	8	db
12	12	ffd	97	9	1c6
13	5	7	98	9	1d5
14	4	0	99	9	1d8
15	4	1	100	10	3ca
16	5	9	101	10	3da
17	6	20	102	11	7ea
18	7	54	103	11	7f1
19	7	60	104	9	1e1
20	8	d5	105	8	d4
21	8	dc	106	8	cf
22	9	1d4	107	8	d6
23	10	3cd	108	8	de
24	10	3de	109	8	e1
25	11	7e7	110	9	1d0
26	6	1c	111	9	1d6
27	4	2	112	10	3d1
28	5	6	113	10	3d5
29	5	c	114	10	3f2
30	6	1e	115	11	7ee
31	6	28	116	11	7fb
32	7	5b	117	10	3e9
33	8	cd	118	9	1cd
34	8	d9	119	9	1c8
35	9	1ce	120	9	1cb
36	9	1dc	121	9	1d1
37	10	3d9	122	9	1d7
38	10	3f1	123	9	1df
39	6	25	124	10	3cf
40	5	b	125	10	3e0
41	5	a	126	10	3ef
42	5	d	127	11	7e6
43	6	24	128	11	7f8
44	7	57	129	12	ffa
45	7	61	130	10	3eb
46	8	cc	131	9	1dd
47	8	dd	132	9	1d3
48	9	1cc	133	9	1d9
49	9	1de	134	9	1db
50	10	3d3	135	10	3d2
51	10	3e7	136	10	3cc
52	7	5d	137	10	3dc
53	6	21	138	10	3ea
54	6	1f	139	11	7ed
55	6	23	140	11	7f3
56	6	27	141	11	7f9
57	7	59	142	12	ff9

58	7	64	143	11	7f2
59	8	d8	144	10	3ce
60	8	df	145	9	1e4
61	9	1d2	146	10	3cb
62	9	1e2	147	10	3d8
63	10	3dd	148	10	3d6
64	10	3ee	149	10	3e2
65	8	d1	150	10	3e5
66	7	55	151	11	7e8
67	6	29	152	11	7f4
68	7	56	153	11	7f5
69	7	58	154	11	7f7
70	7	62	155	12	ffb
71	8	ce	156	11	7fa
72	8	e0	157	10	3ec
73	8	e2	158	10	3df
74	9	1da	159	10	3e1
75	10	3d4	160	10	3e4
76	10	3e3	161	10	3e6
77	11	7eb	162	10	3f0
78	9	1c9	163	11	7e9
79	7	5e	164	11	7ef
80	7	5a	165	12	ff8
81	7	5c	166	12	ffe
82	7	63	167	12	ffc
83	8	ca	168	12	fff
84	8	da			

Table A.12 – Spectrum Huffman Codebook 11

index	**length**	**codeword** (hexadecimal)	**index**	**length**	**codeword** (hexadecimal)
0	4	0	145	10	38d
1	5	6	146	10	398
2	6	19	147	10	3b7
3	7	3d	148	10	3d3
4	8	9c	149	10	3d1
5	8	c6	150	10	3db
6	9	1a7	151	11	7dd
7	10	390	152	8	b4
8	10	3c2	153	10	3de
9	10	3df	154	9	1a9
10	11	7e6	155	9	19b
11	11	7f3	156	9	19c
12	12	ffb	157	9	1a1
13	11	7ec	158	9	1aa
14	12	ffa	159	9	1ad
15	12	ffe	160	9	1b3
16	10	38e	161	10	38b
17	5	5	162	10	3b2
18	4	1	163	10	3b8
19	5	8	164	10	3ce
20	6	14	165	10	3e1
21	7	37	166	10	3e0
22	7	42	167	11	7d2
23	8	92	168	11	7e5
24	8	af	169	8	b7

25	9	191	170	11	7e3
26	9	1a5	171	9	1bb
27	9	1b5	172	9	1a8
28	10	39e	173	9	1a6
29	10	3c0	174	9	1b0
30	10	3a2	175	9	1b2
31	10	3cd	176	9	1b7
32	11	7d6	177	10	39b
33	8	ae	178	10	39a
34	6	17	179	10	3ba
35	5	7	180	10	3b5
36	5	9	181	10	3d6
37	6	18	182	11	7d7
38	7	39	183	10	3e4
39	7	40	184	11	7d8
40	8	8e	185	11	7ea
41	8	a3	186	8	ba
42	8	b8	187	11	7e8
43	9	199	188	10	3a0
44	9	1ac	189	9	1bd
45	9	1c1	190	9	1b4
46	10	3b1	191	10	38a
47	10	396	192	9	1c4
48	10	3be	193	10	392
49	10	3ca	194	10	3aa
50	8	9d	195	10	3b0
51	7	3c	196	10	3bc
52	6	15	197	10	3d7
53	6	16	198	11	7d4
54	6	1a	199	11	7dc
55	7	3b	200	11	7db
56	7	44	201	11	7d5
57	8	91	202	11	7f0
58	8	a5	203	8	c1
59	8	be	204	11	7fb
60	9	196	205	10	3c8
61	9	1ae	206	10	3a3
62	9	1b9	207	10	395
63	10	3a1	208	10	39d
64	10	391	209	10	3ac
65	10	3a5	210	10	3ae
66	10	3d5	211	10	3c5
67	8	94	212	10	3d8
68	8	9a	213	10	3e2
69	7	36	214	10	3e6
70	7	38	215	11	7e4
71	7	3a	216	11	7e7
72	7	41	217	11	7e0
73	8	8c	218	11	7e9
74	8	9b	219	11	7f7
75	8	b0	220	9	190
76	8	c3	221	11	7f2
77	9	19e	222	10	393
78	9	1ab	223	9	1be
79	9	1bc	224	9	1c0
80	10	39f	225	10	394
81	10	38f	226	10	397

82	10	3a9	227	10	3ad
83	10	3cf	228	10	3c3
84	8	93	229	10	3c1
85	8	bf	230	10	3d2
86	7	3e	231	11	7da
87	7	3f	232	11	7d9
88	7	43	233	11	7df
89	7	45	234	11	7eb
90	8	9e	235	11	7f4
91	8	a7	236	11	7fa
92	8	b9	237	9	195
93	9	194	238	11	7f8
94	9	1a2	239	10	3bd
95	9	1ba	240	10	39c
96	9	1c3	241	10	3ab
97	10	3a6	242	10	3a8
98	10	3a7	243	10	3b3
99	10	3bb	244	10	3b9
100	10	3d4	245	10	3d0
101	8	9f	246	10	3e3
102	9	1a0	247	10	3e5
103	8	8f	248	11	7e2
104	8	8d	249	11	7de
105	8	90	250	11	7ed
106	8	98	251	11	7f1
107	8	a6	252	11	7f9
108	8	b6	253	11	7fc
109	8	c4	254	9	193
110	9	19f	255	12	ffd
111	9	1af	256	10	3dc
112	9	1bf	257	10	3b6
113	10	399	258	10	3c7
114	10	3bf	259	10	3cc
115	10	3b4	260	10	3cb
116	10	3c9	261	10	3d9
117	10	3e7	262	10	3da
118	8	a8	263	11	7d3
119	9	1b6	264	11	7e1
120	8	ab	265	11	7ee
121	8	a4	266	11	7ef
122	8	aa	267	11	7f5
123	8	b2	268	11	7f6
124	8	c2	269	12	ffc
125	8	c5	270	12	fff
126	9	198	271	9	19d
127	9	1a4	272	9	1c2
128	9	1b8	273	8	b5
129	10	38c	274	8	a1
130	10	3a4	275	8	96
131	10	3c4	276	8	97
132	10	3c6	277	8	95
133	10	3dd	278	8	99
134	10	3e8	279	8	a0
135	8	ad	280	8	a2
136	10	3af	281	8	ac
137	9	192	282	8	a9
138	8	bd	283	8	b1

139	8	bc	284	8	b3
140	9	18e	285	8	bb
141	9	197	286	8	c0
142	9	19a	287	9	18f
143	9	1a3	288	5	4
144	9	1b1			

Table A.13 – *Kaiser-Bessel window for SSR profile EIGHT_SHORT_SEQUENCE*

i	***w(i)***	***i***	***w(i)***
0	0.0000875914060105	16	0.7446454751465113
1	0.0009321760265333	17	0.8121892962974020
2	0.0032114611466596	18	0.8683559394406505
3	0.0081009893216786	19	0.9125649996381605
4	0.0171240286619181	20	0.9453396205809574
5	0.0320720743527833	21	0.9680864942677585
6	0.0548307856028528	22	0.9827581789763112
7	0.0871361822564870	23	0.9914756203467121
8	0.1302923415174603	24	0.9961964092194694
9	0.1848955425508276	25	0.9984956609571091
10	0.2506163195331889	26	0.9994855586984285
11	0.3260874142923209	27	0.9998533730714648
12	0.4089316830907141	28	0.9999671864476404
13	0.4959414909423747	29	0.9999948432453556
14	0.5833939894958904	30	0.9999995655238333
15	0.6674601983218376	31	0.9999999961638728

Table A.14 – *Kaiser-Bessel window for SSR profile for other window sequences.*

i	***w(i)***	***i***	***w(i)***
0	0.0005851230124487	128	0.7110428359000029
1	0.0009642149851497	129	0.7188474364707993
2	0.0013558207534965	130	0.7265597347077880
3	0.0017771849644394	131	0.7341770687621900
4	0.0022352533849672	132	0.7416968783634273
5	0.0027342299070304	133	0.7491167073477523
6	0.0032773001022195	134	0.7564342060337386
7	0.0038671998069216	135	0.7636471334404891
8	0.0045064443384152	136	0.7707533593446514
9	0.0051974336885144	137	0.7777508661725849
10	0.0059425050016407	138	0.7846377507242818
11	0.0067439602523141	139	0.7914122257259034
12	0.0076040812644888	140	0.7980726212080798
13	0.0085251378135895	141	0.8046173857073919
14	0.0095093917383048	142	0.8110450872887550
15	0.0105590986429280	143	0.8173544143867162
16	0.0116765080854300	144	0.8235441764639875
17	0.0128638627792770	145	0.8296133044858474
18	0.0141233971318631	146	0.8355608512093652
19	0.0154573353235409	147	0.8413859912867303
20	0.0168678890600951	148	0.8470880211822968
21	0.0183572550877256	149	0.8526663589032990
22	0.0199276125319803	150	0.8581205435445334
23	0.0215811201042484	151	0.8634502346476508
24	0.0233199132076965	152	0.8686552113760616
25	0.0251461009666641	153	0.8737353715068081
26	0.0270617631981826	154	0.8786907302411250
27	0.0290689473405856	155	0.8835214188357692

28	0.0311696653515848	156	0.8882276830575707
29	0.0333658905863535	157	0.8928098814640207
30	0.0356595546648444	158	0.8972684835130879
31	0.0380525443366107	159	0.9016040675058185
32	0.0405466983507029	160	0.9058173183656508
33	0.0431438043376910	161	0.9099090252587376
34	0.0458455957104702	162	0.9138800790599416
35	0.0486537485902075	163	0.9177314696695282
36	0.0515698787635492	164	0.9214642831859411
37	0.0545955386770205	165	0.9250796989403991
38	0.0577322144743916	166	0.9285789863994010
39	0.0609813230826460	167	0.9319635019415643
40	0.0643442093520723	168	0.9352346855155568
41	0.0678221432558827	169	0.9383940571861993
42	0.0714163171546603	170	0.9414432135761304
43	0.0751278431308314	171	0.9443838242107182
44	0.0789577503982528	172	0.9472176277741918
45	0.0829069827918993	173	0.9499464282852282
46	0.0869763963425241	174	0.9525720912004834
47	0.0911667569410503	175	0.9550965394547873
48	0.0954787380973307	176	0.9575217494469370
49	0.0999129187977865	177	0.9598497469802043
50	0.1044697814663005	178	0.9620826031668507
51	0.1091497100326053	179	0.9642224303060783
52	0.1139529881122542	180	0.9662713777449607
53	0.1188797973021148	181	0.9682316277319895
54	0.1239302155951605	182	0.9701053912729269
55	0.1291042159181728	183	0.9718949039986892
56	0.1344016647957880	184	0.9736024220549734
57	0.1398223211441467	185	0.9752302180233160
58	0.1453658351972151	186	0.9767805768831932
59	0.1510317475686540	187	0.9782557920246753
60	0.1568194884519144	188	0.9796581613210076
61	0.1627283769610327	189	0.9809899832703159
62	0.1687576206143887	190	0.9822535532154261
63	0.1749063149634756	191	0.9834511596505429
64	0.1811734433685097	192	0.9845850806232530
65	0.1875578769224857	193	0.9856575802399989
66	0.1940583745250518	194	0.9866709052828243
67	0.2006735831073503	195	0.9876272819448033
68	0.2074020380087318	196	0.9885289126911557
69	0.2142421635060113	197	0.9893779732525968
70	0.2211922734956977	198	0.9901766097569984
71	0.2282505723293797	199	0.9909269360049311
72	0.2354151558022098	200	0.9916310308941294
73	0.2426840122941792	201	0.9922909359973702
74	0.2500550240636293	202	0.9929086532976777
75	0.2575259686921987	203	0.9934861430841844
76	0.2650945206801527	204	0.9940253220113651
77	0.2727582531907993	205	0.9945280613237534
78	0.2805146399424422	206	0.9949961852476154
79	0.2883610572460804	207	0.9954314695504363
80	0.2962947861868143	208	0.9958356402684387
81	0.3043130149466800	209	0.9962103726017252
82	0.3124128412663888	210	0.9965572899760172
83	0.3205912750432127	211	0.9968779632693499
84	0.3288452410620226	212	0.9971739102014799
85	0.3371715818562547	213	0.9974465948831872
86	0.3455670606953511	214	0.9976974275220812

87	0.3540283646950029	215	0.9979277642809907
88	0.3625521080463003	216	0.9981389072844972
89	0.3711348353596863	217	0.9983321047686901
90	0.3797730251194006	218	0.9985085513687731
91	0.3884630932439016	219	0.9986693885387259
92	0.3972013967475546	220	0.9988157050968516
93	0.4059842374986933	221	0.9989485378906924
94	0.4148078660689724	222	0.9990688725744943
95	0.4236684856687616	223	0.9991776444921379
96	0.4325622561631607	224	0.9992757396582338
97	0.4414852981630577	225	0.9993639958299003
98	0.4504336971855032	226	0.9994432036616085
99	0.4594035078775303	227	0.9995141079353859
100	0.4683907582974173	228	0.9995774088586188
101	0.4773914542472655	229	0.9996337634216871
102	0.4864015836506502	230	0.9996837868076957
103	0.4954171209689973	231	0.9997280538466377
104	0.5044340316502417	232	0.9997671005064359
105	0.5134482766032377	233	0.9998014254134544
106	0.5224558166913167	234	0.9998314913952471
107	0.5314526172383208	235	0.9998577270385304
108	0.5404346525403849	236	0.9998805282555989
109	0.5493979103766972	237	0.9999002598526793
110	0.5583383965124314	238	0.9999172570940037
111	0.5672521391870222	239	0.9999318272557038
112	0.5761351935809411	240	0.9999442511639580
113	0.5849836462541291	241	0.9999547847121726
114	0.5937936195492526	242	0.9999636603523446
115	0.6025612759529649	243	0.9999710885561258
116	0.6112828224083939	244	0.9999772592414866
117	0.6199545145721097	245	0.9999823431612708
118	0.6285726610088878	246	0.9999864932503106
119	0.6371336273176413	247	0.9999898459281599
120	0.6456338401819751	248	0.9999925223548691
121	0.6540697913388968	249	0.9999946296375997
122	0.6624380414593221	250	0.9999962619864214
123	0.6707352239341151	251	0.9999975018180320
124	0.6789580485595255	252	0.9999984208055542
125	0.6871033051160131	253	0.9999990808746198
126	0.6951678668345944	254	0.9999995351446231
127	0.7031486937449871	255	0.9999998288155155

5 Annex B

Table B.1 transition table 0 (differential scalefactor to index)

DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX
0	68	16	87	32	46	48	25	64	9	80	40	96	96	112	112
1	69	17	88	33	47	49	19	65	10	81	43	97	97	113	113
2	70	18	89	34	48	50	20	66	12	82	44	98	98	114	114
3	71	19	72	35	49	51	14	67	13	83	45	99	99	115	115
4	75	20	90	36	50	52	15	68	17	84	52	100	100	116	116
5	76	21	73	37	51	53	16	69	18	85	53	101	101	117	117
6	77	22	65	38	41	54	11	70	21	86	63	102	102	118	118
7	78	23	66	39	42	55	7	71	22	87	56	103	103	119	119
8	79	24	58	40	35	56	8	72	26	88	64	104	104	120	120

9	80	25	67	41	36	57	5	73	27	89	57	105	105	121	121
10	81	26	59	42	37	58	2	74	28	90	74	106	106	122	122
11	82	27	60	43	29	59	1	75	31	91	91	107	107	123	123
12	83	28	61	44	38	60	0	76	32	92	92	108	108	124	124
13	84	29	62	45	30	61	3	77	33	93	93	109	109	125	125
14	85	30	54	46	23	62	4	78	34	94	94	110	110	126	126
15	86	31	55	47	24	63	6	79	39	95	95	111	111	127	127

Table B.2 transition table 1 (index to differential scalefactor)

INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF	INDEX	DIFF
0	60	16	53	32	76	48	34	64	88	80	9	96	96	112	112
1	59	17	68	33	77	49	35	65	22	81	10	97	97	113	113
2	58	18	69	34	78	50	36	66	23	82	11	98	98	114	114
3	61	19	49	35	40	51	37	67	25	83	12	99	99	115	115
4	62	20	50	36	41	52	84	68	0	84	13	100	100	116	116
5	57	21	70	37	42	53	85	69	1	85	14	101	101	117	117
6	63	22	71	38	44	54	30	70	2	86	15	102	102	118	118
7	55	23	46	39	79	55	31	71	3	87	16	103	103	119	119
8	56	24	47	40	80	56	87	72	19	88	17	104	104	120	120
9	64	25	48	41	38	57	89	73	21	89	18	105	105	121	121
10	65	26	72	42	39	58	24	74	90	90	20	106	106	122	122
11	54	27	73	43	81	59	26	75	4	91	91	107	107	123	123
12	66	28	74	44	82	60	27	76	5	92	92	108	108	124	124
13	67	29	43	45	83	61	28	77	6	93	93	109	109	125	125
14	51	30	45	46	32	62	29	78	7	94	94	110	110	126	126
15	52	31	75	47	33	63	86	79	8	95	95	111	111	127	127

Table B.3 differential scalefactor arithmetic model 0

size	cumulative frequencies
55	8192, 6144, 5120, 4096, 3072, 2560, 2048, 1792, 1536, 1280, 1024, 896, 768, 640, 576, 512, 448, 384, 320, 288, 256, 224, 192, 176, 160, 144, 128, 112, 96, 88, 80, 72, 64, 56, 48, 44, 40, 36, 32, 28, 24, 22, 20, 18, 16, 14, 13, 12, 11, 10, 9, 8, 7, 6, 0,

Table B.4 differential scalefactor arithmetic model 1

size	cumulative frequencies
67	15018, 13653, 12288, 10922, 10240, 9557, 8874, 8192, 7509, 6826, 6144, 5802, 5461, 5120, 4949, 4778, 4608, 4437, 4266, 4096, 3925, 3840, 3754, 3669, 3584, 3498, 3413, 3328, 3242, 3157, 3072, 2986, 2901, 2816, 2730, 2645, 2560, 2474, 2389, 2304, 2218, 2133, 2048, 1962, 1877, 1792, 1706, 1621, 1536, 1450, 1365, 1280, 1194, 1109, 1024, 938, 853, 768, 682, 597, 512, 426, 341, 256, 170, 85, 0,

Table B.5 differential scalefactor arithmetic model 2

size	cumulative frequencies
8	1342, 790, 510, 344, 214, 127, 57, 0,

Table B.6 differential scalefactor arithmetic model 3

size	cumulative frequencies
16	2441, 2094, 1798, 1563, 1347, 1154, 956, 818, 634, 464, 342, 241, 157, 97, 55, 0,

Table B.7 differential scalefactor arithmetic model 4

size	cumulative frequencies
32	3963, 3525, 3188, 2949, 2705, 2502, 2286, 2085, 1868, 1668, 1515, 1354, 1207, 1055, 930, 821, 651, 510, 373, 269, 192, 134, 90, 58, 37, 29, 24, 15, 10, 8, 5, 0,

Table B.8 differential scalefactor arithmetic model 5

size	cumulative frequencies
64	13587, 13282, 12961, 12656, 12165, 11721, 11250, 10582, 10042, 9587, 8742, 8010, 7256, 6619, 6042, 5480, 4898, 4331, 3817, 3374, 3058, 2759, 2545, 2363, 2192, 1989, 1812, 1582, 1390, 1165, 1037, 935, 668, 518, 438, 358, 245, 197, 181, 149, 144, 128, 122, 117, 112, 106, 101, 85, 80, 74, 69, 64, 58, 53, 48, 42, 37, 32, 26, 21, 16, 10, 5, 0,

Table B.9 differential ArModel arithmetic model 0

size	cumulative frequencies
4	9868, 3351, 1676, 0,

Table B.10 differential ArModel arithmetic model 1

size	cumulative frequencies
8	12492, 8600, 5941, 3282, 2155, 1028, 514, 0,

Table B.11 differential ArModel arithmetic model 2

size	cumulative frequencies
16	14316, 12248, 9882, 7516, 6399, 5282, 4183, 3083, 2247, 1411, 860, 309, 185, 61, 31, 0,

Table B.12 differential ArModel arithmetic model 3

size	cumulative frequencies
40	12170, 7956, 6429, 4901, 4094, 3287, 2982, 2677, 2454, 2230, 2062, 1894, 1621, 1348, 1199, 1050, 854, 658, 468, 278, 169, 59, 38, 18, 17, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0,

Table B.13 MS_used model

size	cumulative frequencies
2	8192, 0

Table B.14 stereo_info model

size	cumulative frequencies
4	13926, 4096, 1638, 0

Table B.15 sign arithmetic model

size	cumulative frequencies
2	8192, 0

Table B.16 noise_flag arithmetic model

size	cumulative frequencies
2	8192, 0

Table B.17 noise_mode arithmetic model

size	cumulative frequencies
4	12288, 8192,4096, 0

Table B.18 BSAC arithmetic model 0

Allocated bit = 0

BSAC arithmetic model 1

not used

Table B.19 BSAC arithmetic model 2

Allocated bit = 1

snf	pre_state	dimension	cumulative frequencies
1	0	4	14858, 13706, 12545, 11546, 10434, 9479, 8475, 7619, 6457, 5456, 4497, 3601, 2600, 1720, 862, 0,

Table B.20 BSAC arithmetic model 3

Allocated bit = 1

snf	pre_state	dimension	cumulative frequencies
1	0	4	5476, 4279, 3542, 3269, 2545, 2435, 2199, 2111, 850, 739, 592, 550, 165, 132, 21, 0,

Table B.21 BSAC arithmetic model 4

Allocated bit = 2

snf	pre_state	dimension	cumulative frequencies
2	0	4	4292, 3445, 2583, 2473, 1569, 1479, 1371, 1332, 450, 347, 248, 219, 81, 50, 15, 0,
1	0	4	15290, 14389, 13434, 12485, 11559, 10627, 9683, 8626, 7691, 6727, 5767, 4655, 3646, 2533, 1415, 0,
		3	15139, 13484, 11909, 9716, 8068, 5919, 3590, 0,
		2	14008, 10384, 6834, 0,
		1	11228, 0
	1	4	10355, 9160, 7553, 7004, 5671, 4902, 4133, 3433, 1908, 1661, 1345, 1222, 796, 714, 233, 0,
		3	8328, 6615, 4466, 3586, 1759, 1062, 321, 0,
		2	4631, 2696, 793, 0,
		1	968, 0,

Table B.22 BSAC arithmetic model 5

Allocated bit = 2

snf	pre_state	dimension	cumulative frequencies
2	0	4	3119, 2396, 1878, 1619, 1076, 1051, 870, 826, 233, 231, 198, 197, 27, 26, 1, 0,
1	0	4	3691, 2897, 2406, 2142, 1752, 1668, 1497, 1404, 502, 453, 389, 368, 131, 102, 18, 0,
		3	11106, 8393, 6517, 4967, 2739, 2200, 608, 0,
		2	10771, 6410, 2619, 0,
		1	6112, 0
	1	4	11484, 10106, 7809, 7043, 5053, 3521, 2756, 2603, 2296, 2143, 1990, 1531, 765, 459, 153, 0,
		3	10628, 8930, 6618, 4585, 2858, 2129, 796, 0,
		2	7596, 4499, 1512, 0,
		1	4155, 0,

Table B.23 BSAC arithmetic model 6

Allocated bit = 3

snf	pre_state	dimension	cumulative cumulative frequencies
3	0	4	2845, 2371, 1684, 1524, 918, 882, 760, 729, 200, 198, 180, 178, 27, 25, 1, 0,
2	0	4	1621, 1183, 933, 775, 645, 628, 516, 484, 210, 207, 188, 186, 39, 35, 1, 0,
		3	8800, 6734, 4886, 3603, 1326, 1204, 104, 0,
		2	8869, 5163, 1078, 0,
		1	3575, 0,
	1	4	12603, 12130, 10082, 9767, 8979, 8034, 7404, 6144, 4253, 3780, 3150, 2363, 1575, 945, 630, 0,
		3	10410, 8922, 5694, 4270, 2656, 1601, 533, 0,
		2	8459, 5107, 1670, 0
		1	4003, 0,
1	0	4	5185, 4084, 3423, 3010, 2406, 2289, 2169, 2107, 650, 539, 445, 419, 97, 61, 15, 0,
		3	13514, 11030, 8596, 6466, 4345, 3250, 1294, 0,
		2	13231, 8754, 4635, 0,
		1	9876, 0,
	1	4	14091, 12522, 11247, 10299, 8928, 7954, 6696, 6024, 4766, 4033, 3119, 2508, 1594, 1008, 353, 0,
		3	12596, 10427, 7608, 6003, 3782, 2580, 928, 0,
		2	10008, 6213, 2350, 0,
		1	5614, 0,

Table B.24 BSAC arithmetic model 7

Allocated bit = 3

snf	pre_state	dimension	cumulative frequencies
3	0	4	3833, 3187, 2542, 2390, 1676, 1605, 1385, 1337, 468, 434, 377, 349, 117, 93, 30, 0,
2	0	4	6621, 5620, 4784, 4334, 3563, 3307, 2923, 2682, 1700, 1458, 1213, 1040, 608, 431, 191, 0,
		3	11369, 9466, 7519, 6138, 3544, 2441, 1136, 0,
		2	11083, 7446, 3439, 0,
		1	8823, 0,
	1	4	12027, 11572, 9947, 9687, 9232, 8126, 7216, 6176, 4161, 3705, 3055, 2210, 1235, 780, 455, 0,
		3	9566, 7943, 4894, 3847, 2263, 1596, 562, 0,
		2	7212, 4217, 1240, 0,

		1	3296, 0,
1	0	4	14363, 13143, 12054, 11153, 10220, 9388, 8609, 7680, 6344, 5408, 4578, 3623, 2762, 1932, 1099, 0,
		3	14785, 13256, 11596, 9277, 7581, 5695, 3348, 0,
		2	14050, 10293, 6547, 0,
		1	10948, 0,
	1	4	13856, 12350, 11151, 10158, 8816, 7913, 6899, 6214, 4836, 4062, 3119, 2505, 1624, 1020, 378, 0,
		3	12083, 9880, 7293, 5875, 3501, 2372, 828, 0,
		2	8773, 5285, 1799, 0,
		1	4452, 0,

Table B.25 BSAC arithmetic model 8

Allocated bit = 4

snf	pre_state	dimension	cumulative frequencies
4	0	4	2770, 2075, 1635, 1511, 1059, 1055, 928, 923, 204, 202, 190, 188, 9, 8, 1, 0,
3	0	4	1810, 1254, 1151, 1020, 788, 785, 767, 758, 139, 138, 133, 132, 14, 13, 1, 0,
		3	7113, 4895, 3698, 3193, 1096, 967, 97, 0,
		2	6858, 4547, 631, 0,
		1	4028, 0,
	1	4	13263, 10922, 10142, 9752, 8582, 7801, 5851, 5071, 3510, 3120, 2730, 2340, 1560, 780, 390, 0,
		3	12675, 11275, 7946, 6356, 4086, 2875, 1097, 0,
		2	9473, 5781, 1840, 0,
		1	3597, 0,
2	0	4	2600, 1762, 1459, 1292, 989, 983, 921, 916, 238, 233, 205, 202, 32, 30, 3, 0,
		3	10797, 8840, 6149, 5050, 2371, 1697, 483, 0,
		2	10571, 6942, 2445, 0,
		1	7864, 0,
	1	4	14866, 12983, 11297, 10398, 9386, 8683, 7559, 6969, 5451, 4721, 3484, 3007, 1882, 1208, 590, 0,
		3	12611, 10374, 8025, 6167, 4012, 2608, 967, 0,
		2	10043, 6306, 2373, 0,
		1	5766, 0,
1	0	4	6155, 5057, 4328, 3845, 3164, 2977, 2728, 2590, 1341, 1095, 885, 764, 303, 188, 74, 0,
		3	12802, 10407, 8142, 6263, 3928, 3013, 1225, 0,
		2	13131, 9420, 4928, 0,
		1	10395, 0,
	1	4	14536, 13348, 11819, 11016, 9340, 8399, 7135, 6521, 5114, 4559, 3521, 2968, 1768, 1177, 433, 0,
		3	12735, 10606, 7861, 6011, 3896, 2637, 917, 0,
		2	9831, 5972, 2251, 0,
		1	4944, 0,

Table B.26 BSAC arithmetic model 9

Allocated bit = 4

snf	pre_state	dimension	cumulative frequencies
4	0	4	3383, 2550, 1967, 1794, 1301, 1249, 1156, 1118, 340, 298, 247, 213, 81, 54, 15, 0,
3	0	4	7348, 6275, 5299, 4935, 3771, 3605, 2962, 2814, 1295, 1143, 980, 860, 310, 230, 75, 0,
		3	9531, 7809, 5972, 4892, 2774, 1782, 823, 0,

		2	11455, 7068, 3383, 0,
		1	9437, 0,
	1	4	12503, 9701, 8838, 8407, 6898, 6036, 4527, 3664, 2802, 2586, 2371, 2155, 1293, 431, 215, 0,
		3	11268, 9422, 6508, 5277, 3076, 2460, 1457, 0,
		2	7631, 4565, 1506, 0,
		1	2639, 0,
2	0	4	11210, 9646, 8429, 7389, 6252, 5746, 5140, 4692, 3350, 2880, 2416, 2014, 1240, 851, 404, 0,
		3	12143, 10250, 7784, 6445, 3954, 2528, 1228, 0,
		2	10891, 7210, 3874, 0,
		1	9537, 0,
	1	4	14988, 13408, 11860, 10854, 9631, 8992, 7834, 7196, 5616, 4793, 3571, 2975, 1926, 1212, 627, 0,
		3	12485, 10041, 7461, 5732, 3669, 2361, 940, 0,
		2	9342, 5547, 1963, 0,
		1	5140, 0,
1	0	4	14152, 13258, 12486, 11635, 11040, 10290, 9740, 8573, 7546, 6643, 5903, 4928, 4005, 2972, 1751, 0,
		3	14895, 13534, 12007, 9787, 8063, 5761, 3570, 0,
		2	14088, 10108, 6749, 0,
		1	11041, 0,
	1	4	14817, 13545, 12244, 11281, 10012, 8952, 7959, 7136, 5791, 4920, 3997, 3126, 2105, 1282, 623, 0,
		3	12873, 10678, 8257, 6573, 4186, 2775, 1053, 0,
		2	9969, 6059, 2363, 0,
		1	5694, 0,

Table B.27 BSAC arithmetic model 10

Allocated bit (Abit) = 5

snf	pre_state	dimension	cumulative frequencies
Abit	0	4	2335, 1613, 1371, 1277, 901, 892, 841, 833, 141, 140, 130, 129, 24, 23, 1, 0,
Abit-1	0	4	1746, 1251, 1038, 998, 615, 611, 583, 582, 106, 104, 101, 99, 3, 2, 1, 0,
		3	7110, 5230, 4228, 3552, 686, 622, 46, 0,
		2	6101, 2575, 265, 0,
		1	1489, 0,
	1	4	13010, 12047, 11565, 11083, 9637, 8673, 6264, 5782, 4336, 3855, 3373, 2891, 2409, 1927, 963, 0,
		3	10838, 10132, 8318, 7158, 5595, 3428, 2318, 0,
		2	8209, 5197, 1287, 0,
		1	4954, 0,
Abit-2	0	4	2137, 1660, 1471, 1312, 1007, 1000, 957, 951, 303, 278, 249, 247, 48, 47, 1, 0,
		3	9327, 7413, 5073, 4391, 2037, 1695, 205, 0,
		2	8658, 5404, 1628, 0,
		1	5660, 0,
	1	4	13360, 12288, 10727, 9752, 8484, 7899, 7119, 6631, 5363, 3900, 3023, 2535, 1852, 1267, 585, 0,
		3	13742, 11685, 8977, 7230, 5015, 3426, 1132, 0,
		2	10402, 6691, 2828, 0,
		1	5298, 0,
Abit-3	0	4	4124, 3181, 2702, 2519, 1959, 1922, 1733, 1712, 524, 475, 425, 407, 78, 52, 15, 0,
		3	10829, 8581, 6285, 4865, 2539, 1920, 594, 0,

		2	11074, 7282, 3092, 0,
		1	8045, 0,
	1	4	14541, 13343, 11637, 10862, 9328, 8783, 7213, 6517, 5485, 5033, 4115, 3506, 2143, 1555, 509, 0,
		3	13010, 11143, 8682, 7202, 4537, 3297, 1221, 0,
		2	9941, 5861, 2191, 0,
		1	5340, 0,
other snf	0	4	9845, 8235, 7126, 6401, 5551, 5131, 4664, 4320, 2908, 2399, 1879, 1506, 935, 603, 277, 0,
		3	13070, 11424, 9094, 7203, 4771, 3479, 1486, 0,
		2	13169, 9298, 5406, 0,
		1	10371, 0,
	1	4	14766, 13685, 12358, 11442, 10035, 9078, 7967, 7048, 5824, 5006, 4058, 3400, 2350, 1612, 659, 0,
		3	13391, 11189, 8904, 7172, 4966, 3183, 1383, 0,
		2	10280, 6372, 2633, 0,
		1	5419, 0,

Table B.28 BSAC arithmetic model 11

Allocated bit (Abit) = 5

snf	pre_state	dimension	cumulative frequencies
Abit	0	4	2872, 2294, 1740, 1593, 1241, 1155, 1035, 960, 339, 300, 261, 247, 105, 72, 34, 0,
Abit-1	0	4	3854, 3090, 2469, 2276, 1801, 1685, 1568, 1505, 627, 539, 445, 400, 193, 141, 51, 0,
		3	10654, 8555, 6875, 4976, 3286, 2229, 826, 0,
		2	10569, 6180, 2695, 0,
		1	6971, 0,
	1	4	11419, 11170, 10922, 10426, 7943, 6950, 3723, 3475, 1737, 1489, 1241, 992, 744, 496, 248, 0,
		3	11013, 9245, 6730, 4962, 3263, 1699, 883, 0,
		2	6969, 4370, 1366, 0,
		1	3166, 0,
Abit-2	0	4	9505, 8070, 6943, 6474, 5305, 5009, 4290, 4029, 2323, 1911, 1591, 1363, 653, 443, 217, 0,
		3	11639, 9520, 7523, 6260, 4012, 2653, 1021, 0,
		2	12453, 8284, 4722, 0,
		1	9182, 0,
	1	4	13472, 12295, 10499, 9167, 7990, 7464, 6565, 6008, 4614, 3747, 2818, 2477, 1641, 1084, 557, 0,
		3	13099, 10826, 8476, 6915, 4488, 2966, 1223, 0,
		2	9212, 5772, 2053, 0,
		1	4244, 0,
Abit-3	0	4	14182, 12785, 11663, 10680, 9601, 8758, 8135, 7353, 6014, 5227, 4433, 3727, 2703, 1818, 866, 0,
		3	13654, 11814, 9714, 7856, 5717, 3916, 2112, 0,
		2	12497, 8501, 4969, 0,
		1	10296, 0,
	1	4	15068, 13770, 12294, 11213, 10230, 9266, 8439, 7438, 6295, 5368, 4361, 3620, 2594, 1797, 895, 0,
		3	13120, 10879, 8445, 6665, 4356, 2794, 1047, 0,
		2	9311, 5578, 1793, 0,
		1	4695, 0
other snf	0	4	15173, 14794, 14359, 13659, 13224, 12600, 11994, 11067, 10197, 9573, 9081, 7624, 6697, 4691, 3216, 0,
		3	15328, 13985, 12748, 10084, 8587, 6459, 4111, 0,

		2	14661, 11179, 7924, 0,
		1	11399, 0,
	1	4	14873, 13768, 12458, 11491, 10229, 9164, 7999, 7186, 5992, 5012, 4119, 3369, 2228, 1427, 684, 0,
		3	13063, 10913, 8477, 6752, 4529, 3047, 1241, 0,
		2	10101, 6369, 2615, 0,
		1	5359, 0,

Table B.29 BSAC arithmetic model 12
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 6

Table B.30 BSAC arithmetic model 13
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 6

Table B.31 BSAC arithmetic model 14
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 7

Table B.32 BSAC arithmetic model 15
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 7

Table B.33 BSAC arithmetic model 16
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 8

Table B.34 BSAC arithmetic model 17
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 8

Table B.35 BSAC arithmetic model 18
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 9

Table B.36 BSAC arithmetic model 19
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 9

Table B.37 BSAC arithmetic model 20
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 10

Table B.38 BSAC arithmetic model 21
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 10

Table B.39 BSAC arithmetic model 22
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 11

Table B.40 BSAC arithmetic model 23
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 11

Table B.41 BSAC arithmetic model 24
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 12

Table B.42 BSAC arithmetic model 25
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 12

Table B.43 BSAC arithmetic model 26
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 13

Table B.44 BSAC arithmetic model 27
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 13

Table B.45 BSAC arithmetic model 28
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 14

Table B.46 BSAC arithmetic model 29
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 14

Table B.47 BSAC arithmetic model 30
same as BSAC arithmetic model 10, but Allocated bit (Abit) = 15

Table B.48 BSAC arithmetic model 31
same as BSAC arithmetic model 11, but Allocated bit (Abit) = 15

6 Annex C

Contents are available in w1903tvq.doc.

Annex C Codebook tables for TwinVQ

Table C. 1 Shape codebook 0 for MODE_VQ == 24_06 / 24_06_960, BLOCK_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1449.5	-1739.1	152.0	-156.9	-163.5	575.2	249.8	-10426.9	-103.8	-2236.0	-263.2	183.1
	12	-21.9	-51.7	495.6	-18.1	142.7	-359.5	-8.7	-52.9	3279.8	362.0	260.4	-3.2
	24	127.0	-40.9	509.9									
1	0	-454.0	-525.3	-4204.6	52.3	-202.7	-92.7	-189.6	136.2	30.4	-106.0	36.0	-24.0
	12	-112.2	-55.4	-73.3	15.2	-92.9	156.9	-122.3	248.9	-147.9	297.4	21.0	-338.9
	24	-20.3	912.7	1628.1									
2	0	-167.1	-226.2	-60.8	-34.0	-24.2	-89.3	97.5	51.8	115.4	485.8	323.4	-196.7
	12	239.0	25.4	435.5	4.2	11659.6	77.7	173.5	101.8	-148.0	-66.7	-198.2	-299.9
	24	46.8	80.1	294.3									
3	0	-293.2	-202.3	-87.5	-67.2	77.1	199.0	-104.6	-400.0	43.2	370.5	58.3	-6.8
	12	13209.6	-9.3	131.5	0.7	347.4	-48.1	-276.8	384.0	332.5	318.3	329.5	-153.8
	24	296.6	-101.4	-315.9									
4	0	410.8	254.0	-180.3	70.0	44.6	-58.6	-39.3	17.9	-23.5	-85.6	172.1	435.4
	12	-11.2	77.9	-186.1	-6136.4	-49.1	1576.5	42.7	-534.4	113.8	-35.3	-125.6	-5.4
	24	-69.0	-82.7	8.8									
5	0	-201.4	62.6	-71.4	-59.6	40.5	26.6	-319.6	-7165.8	260.5	197.5	-33.7	-527.5
	12	-9.7	-177.8	-7800.2	118.4	3.9	-337.8	-204.6	-498.9	-566.7	-648.3	122.0	208.5
	24	458.2	50.0	-664.0									
6	0	4108.6	102.9	51.4	-288.7	69.1	247.0	-104.4	8262.1	17.8	158.9	43.0	347.4
	12	-99.2	91.9	-108.9	59.4	100.0	37.0	97.8	74.5	766.2	98.2	-51.9	-321.1
	24	-144.4	-270.6	1062.4									
7	0	36.5	-3633.9	227.4	114.9	9.2	-31.3	-189.8	303.5	159.5	206.1	-147.7	159.8
	12	1.9	217.7	-6623.8	-40.9	219.8	-369.4	-281.8	-230.4	-591.7	-327.2	-33.1	238.4
	24	-156.9	431.5	-714.1									
8	0	1261.2	-402.9	4009.4	-215.6	-260.9	91.1	7172.0	132.9	31.9	-20.3	420.1	276.5
	12	-15.4	129.3	-99.9	-882.0	-74.8	579.8	-157.4	-72.4	-75.3	-23.2	-122.7	-956.5
	24	268.0	-161.2	495.5									
9	0	-559.4	-309.7	225.2	344.3	-845.5	-20.1	-295.0	-150.0	-79.0	-33.6	-82.3	-52.5
	12	-5865.7	-84.3	-114.3	-40.8	133.9	-588.7	-411.7	386.6	138.7	-208.8	606.0	53.1
	24	531.4	1281.7	-623.6									
10	0	312.7	-345.2	259.9	-108.7	-2.1	-117.0	105.4	38.3	-94.6	-45.8	-43.1	-3.9
	12	-47.3	4577.1	-149.6	62.8	-18.9	13.5	64.0	318.1	250.9	193.3	79.1	-135.1
	24	23.4	-33.2	-1421.1									
11	0	4092.5	-24.9	214.0	-1.1	-118.5	-5470.7	219.1	-81.5	282.3	-398.2	-104.1	-42.1
	12	55.4	464.8	-23.1	98.6	-88.2	39.7	-187.5	-344.0	-307.7	-323.8	163.7	145.7
	24	-53.1	-254.9	69.6									
12	0	1065.2	316.5	3003.8	791.7	50.8	99.8	271.3	-110.7	71.8	-7347.0	-104.5	-145.2
	12	-116.0	711.5	532.5	-217.1	93.1	54.6	-308.0	40.7	451.6	-198.8	55.6	-252.3
	24	15.8	676.8	-254.8									
13	0	-3004.9	-320.1	-80.1	70.0	92.5	-53.4	-122.2	34.0	-94.8	108.4	165.7	75.4
	12	-14.6	116.1	4.8	-179.0	56.9	-233.5	52.6	-1720.1	9.1	-109.5	0.5	31.8
	24	109.9	6.1	-538.5									
14	0	88.2	-3490.4	-129.0	171.2	-73.3	37.6	43.8	-87.1	205.7	-88.6	180.1	-43.3
	12	-35.5	-69.7	266.2	2.6	-310.2	-152.4	223.3	-8.5	156.5	271.7	-123.3	-464.9
	24	160.7	-8326.6	-1128.7									
15	0	342.8	81.8	172.6	-21.6	-91.1	-19.1	21.0	-413.9	-173.5	280.7	-66.7	11914.7
	12	-127.7	24.2	357.9	63.6	59.3	287.8	7.3	447.9	486.6	-43.8	33.0	60.8
	24	-123.9	60.3	1050.9									
16	0	50.6	-33.5	111.2	-6.4	-66.0	117.5	78.8	-157.7	16.9	-61.5	2.7	7.1
	12	-19.8	96.5	286.6	-15.2	-85.1	140.2	67.3	91.7	-92.1	13208.0	48.2	119.1
	24	54.6	201.2	285.7									
17	0	-267.9	-314.3	-274.8	-172.5	78.1	15.1	-61.9	-207.7	-84.2	-4808.8	5.2	-112.0
	12	-37.6	-42.7	-38.6	196.0	-90.7	67.1	190.0	-128.3	-84.4	-23.3	31.2	147.5
	24	-144.8	63.8	-473.0									
18	0	-65.0	40.3	-121.4	-41.1	-199.1	-16.8	-16.7	-91.3	-26.5	111.1	-47.2	154.5
	12	-186.2	-124.2	154.5	-90.3	-667.8	109.6	-11.5	396.9	63.9	63.3	-49.2	64.9
	24	-16384.0	-324.2	-119.8									
19	0	-442.2	-4840.0	-193.9	-5539.7	-91.0	-24.0	155.8	-168.4	231.0	-57.8	89.8	-70.8
	12	-55.2	-43.6	147.9	31.1	-244.3	207.2	411.9	-176.5	-182.2	46.4	-378.7	-184.0
	24	26.0	-31.0	359.2									
20	0	-426.9	-8856.8	-7.2	63.9	-31.9	-242.2	273.9	39.9	331.4	25.6	-154.7	-76.8
	12	37.5	-264.0	28.4	-279.1	2.6	-260.4	87.0	-52.6	-687.5	124.4	-98.4	-256.2
	24	34.0	205.8	-1060.7									
21	0	190.3	105.5	372.9	185.0	3.0	157.7	7.1	12.2	-97.8	-155.6	-46.7	79.7
	12	-9.3	15673.5	119.6	-117.4	-120.9	134.3	34.1	-52.9	-78.3	-1.7	194.2	-49.9
	24	-70.4	113.8	4.7									
22	0	-85.2	752.1	201.3	174.0	44.8	111.3	-251.2	-27.5	71.7	-72.2	354.5	-2.7
	12	13.0	7121.1	-14.6	9.1	7.8	104.7	-12.2	-48.3	-363.0	-57.7	27.1	42.0
	24	128.0	-64.7	16384.0									
23	0	-283.8	-5043.9	-11.4	-255.3	-7.2	-237.2	17.1	-148.5	-6158.4	-256.6	-19.6	-220.8
	12	-29.0	43.5	14.6	-31.7	-55.8	-50.1	-116.7	-74.4	736.0	-54.8	-79.3	70.7
	24	-52.0	111.5	-155.0									
24	0	32.9	-33.5	-233.0	-120.0	-57.4	19.7	-70.0	-58.9	-55.8	-15.0	-120.2	-20.4
	12	-122.6	-99.5	-40.6	-635.1	-238.0	14784.1	-159.2	286.0	29.3	76.9	61.1	-290.3
	24	222.2	67.9	-53.4									
25	0	5314.2	-5465.1	-40.8	-135.6	-23.6	-202.9	-73.4	-64.6	220.7	178.4	38.6	-29.6
	12	-67.8	-192.0	-74.1	2.8	-161.7	-45.0	43.5	251.7	-381.3	-51.5	22.5	141.6
	24	-123.3	80.8	-703.5									
26	0	-535.1	19.6	-17.8	509.8	-75.9	-10.6	-100.9	70.7	-140.9	-29.6	-13791.0	-2.8
	12	43.6	92.0	131.4	120.4	78.9	286.3	258.1	-63.2	61.6	41.5	81.3	-358.9
	24	52.2	300.4	-417.3									
27	0	-14.7	-4450.9	-344.6	167.7	-5842.0	-145.5	-142.7	-125.7	150.9	195.4	-107.5	-66.4
	12	-155.7	-21.8	273.6	-243.8	-61.3	26.5	6.5	66.2	-109.6	44.0	-89.8	97.9
	24	-40.1	101.1	1828.7									
28	0	95.1	-4495.7	167.5	1.7	5672.8	-74.7	202.6	85.9	59.2	80.9	186.3	233.2
	12	31.9	-98.4	148.9	144.5	97.1	-114.9	-189.0	-82.4	-58.7	-144.7	-27.3	-69.6
	24	45.6	-38.3	1144.4									
29	0	-152.0	-703.5	-227.0	69.1	66.4	-9.5	-114.3	-286.9	-62.1	-11.4	105.1	269.5
	12	-20.0	-81.1	9094.4	-108.2	173.8	-69.1	-26.4	-310.6	88.8	800.6	128.3	148.2
	24	54.5	93.9	-126.2									

30	0	-301.0	9.3	-537.8	-7.5	-514.1	423.3	81.9	317.8	29.5	-14174.8	4.2	-124.2
	12	91.0	292.3	-187.8	93.2	43.1	201.9	-86.4	332.4	-298.3	168.2	373.5	213.5
	24	67.8	273.7	307.1									
31	0	-223.0	1933.8	-184.0	-1521.2	39.9	-36.2	44.5	-35.8	25.5	-68.9	534.3	-67.3
	12	-48.5	99.1	-58.9	32.2	-26.1	32.4	175.2	150.3	109.4	238.5	-179.0	-214.0
	24	-64.2	-524.1	95.6									
32	0	90.6	707.4	63.6	182.9	-349.8	-4971.2	-319.4	-7579.1	517.2	758.0	48.2	-233.5
	12	-1.8	155.4	660.3	-118.7	260.4	23.8	39.7	286.4	8248.9	-54.7	39.9	-204.2
	24	105.4	-212.0	-244.6									
33	0	-30.5	-143.7	-12.9	-7033.1	-49.7	-325.1	18.6	7604.3	584.8	-252.0	64.4	-355.5
	12	24.2	191.6	269.9	-207.0	-78.4	39.3	-78.2	-664.4	-601.5	-38.4	22.2	86.7
	24	33.1	233.8	93.2									
34	0	-347.3	151.5	27.0	-4343.2	-124.9	14.3	137.7	-15.3	0.6	-9.7	-6448.0	59.9
	12	-107.4	-37.4	141.0	-17.6	125.0	237.3	184.3	-89.4	103.0	41.9	-207.2	-572.8
	24	146.0	-4.5	-862.3									
35	0	1006.8	-66.0	-93.4	66.5	42.5	21.4	-120.1	35.6	65.8	94.7	167.8	49.7
	12	-97.8	-93.6	380.1	-44.0	95.6	103.3	30.9	-78.8	113.3	-39.5	99.5	138.0
	24	-25.0	8774.7	-9271.0									
36	0	393.0	248.7	-61.1	-6645.5	107.7	283.5	164.0	-8995.6	1.3	194.9	2.4	121.8
	12	17.1	-219.8	168.9	-336.9	8.1	-17.9	-34.5	156.0	325.7	-259.9	-82.6	10.5
	24	1.7	-20.7	364.3									
37	0	-305.3	-256.5	-144.5	-2794.9	111.7	140.6	-4.1	-2.2	65.5	32.1	-312.5	-5.1
	12	-93.7	-72.6	-4.5	-8.3	-58.1	85.3	-257.9	33.7	138.2	75.2	-64.5	215.5
	24	8.5	1294.5	16384.0									
38	0	-295.9	569.0	126.9	-5946.4	-515.9	-6882.0	53.2	-351.3	-60.4	44.5	378.2	-283.4
	12	146.3	-326.5	-146.1	412.9	183.7	-87.5	51.8	299.2	175.7	-107.2	-60.6	-105.5
	24	289.0	38.5	1420.9									
39	0	433.7	508.2	-47.3	-55.2	53.0	490.6	-77.3	-13409.5	41.3	541.0	85.4	59.2
	12	-93.5	-50.3	349.3	45.4	-9.0	20.6	45.6	-305.3	-1118.3	116.5	49.6	37.9
	24	227.1	251.2	26.2									
40	0	481.3	-23.9	292.5	-10115.7	-88.1	-66.8	249.5	46.7	2.4	-219.1	37.4	135.9
	12	141.7	91.8	67.2	-77.3	25.8	109.8	-23.0	170.2	212.9	151.4	121.4	-89.9
	24	-162.6	153.4	-445.8									
41	0	672.7	5660.0	-344.2	-6422.3	-272.4	317.9	233.1	-28.1	-35.0	-20.5	114.5	-153.9
	12	29.7	-91.8	-289.8	92.0	-220.8	-30.8	-131.9	33.6	47.8	215.6	-215.8	-34.8
	24	-224.3	363.1	1412.5									
42	0	92.7	-390.0	111.0	90.9	1379.7	112.8	-68.8	-66.3	22.4	280.6	-302.7	-32.9
	12	198.0	-42.6	-76.8	110.1	-289.2	9.9	-7591.6	-161.4	11.2	-38.5	166.8	94.7
	24	-47.8	256.0	324.6									
43	0	-63.0	-194.5	-284.3	163.1	198.9	-3078.8	-39.1	-38.5	-5.4	-95.3	-91.8	-163.3
	12	-109.1	68.6	160.3	-30.9	-26.6	-23.2	-108.8	198.0	-350.8	22.9	-69.7	387.3
	24	21.9	58.9	10559.7									
44	0	174.2	195.7	-131.6	-177.3	68.4	65.9	-184.3	-288.9	-65.9	-52.6	-9.5	98.1
	12	16.7	119.6	-170.8	-6766.1	64.4	-208.1	20.6	11118.4	81.5	-51.7	25.9	153.7
	24	43.2	-58.8	98.9									
45	0	114.5	104.1	-378.9	5824.2	389.9	-6776.3	-12.5	-462.4	9.8	371.7	-147.6	-131.1
	12	8.9	-436.2	-23.3	-325.5	24.9	12.7	283.6	54.9	-1282.5	50.4	61.8	51.9
	24	-343.9	-379.4	566.6									
46	0	5116.5	85.5	109.9	407.7	5722.2	167.9	-46.5	-87.2	-16.8	375.1	-255.7	264.4
	12	-105.5	111.5	162.2	-281.4	50.7	-55.1	-195.4	-383.9	89.2	-173.8	86.4	56.9
	24	-95.7	-213.0	450.6									
47	0	-5409.0	-368.1	5792.1	-504.5	311.1	26.6	-135.1	-57.7	59.7	-216.9	226.9	-67.9
	12	-95.9	337.9	-159.9	-437.5	95.6	98.8	-277.6	188.7	140.3	-107.9	142.6	-61.5
	24	-250.8	163.2	279.7									
48	0	-402.4	-28.6	-969.0	28.7	-5.3	-137.9	4886.3	-206.4	-5.8	5.0	61.0	57.5
	12	48.3	-178.3	-94.7	-211.8	-81.8	-235.7	-105.0	311.8	-55.1	-125.4	58.4	26.6
	24	209.3	-553.1	-694.6									
49	0	178.4	177.2	5344.1	-131.3	-276.8	-7701.3	-68.4	-159.8	230.7	62.4	154.4	-117.7
	12	325.8	266.0	482.0	249.7	-158.5	-84.9	-215.9	134.3	102.2	163.1	-149.6	174.0
	24	134.4	36.8	1037.0									
50	0	-104.2	-535.7	77.7	-189.1	-213.7	216.3	-12768.7	58.6	68.9	17.4	321.8	-37.0
	12	82.3	-382.5	153.0	76.3	-50.1	305.9	61.9	-566.0	57.2	-37.8	-122.3	51.3
	24	-130.4	-258.5	139.6									
51	0	-4284.0	-171.2	-73.5	-8.3	56.0	-6612.0	-470.8	59.3	223.7	551.1	-225.6	-107.3
	12	150.6	-530.7	59.8	107.1	-52.2	-195.7	-141.9	-63.8	95.1	321.6	113.5	127.7
	24	-191.5	218.8	449.7									
52	0	-829.6	35.7	-174.5	18.4	-181.5	-69.6	145.8	57.9	-52.3	-132.1	309.2	-4.6
	12	-156.6	59.0	-55.3	66.3	6.3	-181.2	24.6	75.2	24.9	14.3	19.5	234.5
	24	332.1	16384.0	1256.6									
53	0	-11.2	606.0	-466.2	-86.7	-91.9	-11728.5	14.7	74.3	-38.6	-180.6	-140.3	222.0
	12	-393.3	-908.3	-48.6	-79.9	55.5	-26.4	-154.7	78.4	255.5	-170.1	206.3	64.1
	24	64.0	-324.0	827.0									
54	0	-528.9	29.9	223.8	-460.9	10693.7	89.7	-41.5	-98.5	185.8	193.6	-75.0	-111.5
	12	70.2	62.7	81.6	-204.4	182.9	-151.0	151.0	-632.9	165.7	-176.2	-47.5	45.6
	24	98.5	-99.4	719.2									
55	0	-9174.0	-394.2	-562.3	230.5	503.5	117.6	271.6	75.3	238.9	308.4	-125.2	95.0
	12	84.9	-130.3	-7.3	124.5	-135.0	-70.4	-40.8	-182.7	-204.3	99.4	109.7	166.9
	24	115.8	-81.5	704.7									
56	0	262.6	26.2	33.9	63.6	160.9	52.7	-53.0	-4413.8	84.2	218.9	-79.2	112.8
	12	-134.1	123.4	37.5	-310.5	-121.2	-105.4	112.9	-282.9	478.0	174.7	-16.7	13.4
	24	-148.3	260.8	183.3									
57	0	5.6	18.5	-4411.0	120.0	-10.1	101.2	-179.2	137.3	182.1	12.8	-68.1	-73.0
	12	180.7	7909.7	60.3	80.4	53.2	-116.2	-253.2	122.6	200.3	-276.4	110.8	280.4
	24	55.4	-206.8	611.0									
58	0	380.7	-501.4	-47.2	-51.3	-197.4	-35.6	-286.2	-122.0	-49.0	-13.3	-18.8	26.2
	12	-122.8	-112.5	5.7	16251.2	50.4	223.8	-142.8	4219.4	-31.9	-45.0	90.7	260.7
	24	-197.2	76.4	473.8									
59	0	152.8	-268.2	-787.1	-147.1	66.5	8.0	-196.6	-52.6	-3.5	26.9	250.4	-128.8
	12	-17.5	18.4	32.1	-279.7	-18.8	6.8	-277.0	68.9	39.2	-172.9	41.3	16384.0
	24	-54.8	-129.9	-199.8									
60	0	151.0	131.1	541.2	3.2	51.3	39.4	0.7	-96.0	-9.0	133.1	891.4	29.6
	12	129.5	-17.6	100.3	-80.2	-21.2	112.1	71.4	-318.1	57.4	-6.3	-84.9	7479.8
	24	27.2	-74.7	613.3									
61	0	-447.6	-133.0	-8797.0	-167.3	260.9	-83.5	56.9	-18.7	-17.1	-95.6	-67.1	-58.9
	12	10.0	160.4	143.9	-503.5	25.0	81.6	-141.9	151.4	74.4	-220.5	-215.1	329.2
	24	-192.6	104.9	-801.9									
62	0	-32.3	693.8	-14.4	136.9	96.9	-603.7	71.8	155.2	-9845.8	188.7	-59.0	366.7
	12	152.3	0.4	-89.3	-322.4	-71.5	317.7	-100.6	28.7	193.0	-24.3	307.4	1.4
	24	-76.2	398.0	143.7									
63	0	451.1	52.9	-1652.4	-18.0	190.9	-111.8	69.2	-83.2	3.4	-12.3	-93.5	-134.2

	12	72.4	-132.1	-52.8	51.9	-24.1	123.7	-403.3	-243.9	579.4	-30.5	138.3	758.7
	24	-26.5	-4740.6	-7423.3									

Table C. 2Shape codebook 1 for MODE_VQ == 24_06 / 24_06_960, BLOCK_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	244.8	279.5	-401.0	318.3	-150.9	-3442.1	251.6	73.8	101.6	-42.8	322.5	-55.2
	12	86.3	-193.6	-35.3	25.4	91.7	25.9	89.1	-77.1	7.2	-70.1	161.1	-275.7
	24	116.1	355.9	445.3									
1	0	3490.9	11.2	-486.1	36.5	-115.5	233.5	-9743.0	-135.8	304.1	-174.3	-122.9	154.5
	12	-404.2	417.0	58.9	-72.9	-296.4	187.0	-517.1	101.5	174.2	-135.1	-90.2	-80.8
	24	-12.8	523.5	-146.1									
2	0	77.0	-208.2	-94.7	-104.4	4.8	-59.9	191.8	110.5	69.0	-127.0	17.2	-46.5
	12	-44.8	-150.7	-3495.0	-64.0	-164.3	165.8	-80.8	73.2	3119.5	-67.8	-87.7	229.8
	24	-29.6	645.1	220.3									
3	0	-1865.5	505.4	106.2	-147.1	-162.4	104.3	-54.8	72.9	59.8	204.6	41.2	68.1
	12	-78.4	9970.1	-76.4	-20.9	72.8	-87.1	-24.1	54.4	-49.5	164.0	31.1	267.6
	24	68.2	47.7	969.5									
4	0	838.5	816.5	71.4	-62.4	-634.7	-206.6	-109.2	-54.2	-7.2	-8629.1	179.6	506.5
	12	12.2	-932.6	-162.7	-214.7	-274.2	127.3	-30.4	-805.7	318.0	-178.1	67.1	123.0
	24	-25.0	-32.1	1009.1									
5	0	62.3	-4394.2	323.6	-249.4	38.0	185.5	9.2	29.2	-198.4	-10.8	-135.2	134.8
	12	186.2	8693.9	70.8	233.0	-217.1	129.2	41.2	-135.2	409.3	-43.9	-276.2	-5.8
	24	91.7	353.5	749.9									
6	0	-32.9	-61.4	-130.6	88.2	-79.4	-112.1	-125.4	-496.2	27.7	165.2	-36.3	29.1
	12	-108.4	184.6	83.9	-32.4	-78.2	-61.4	4.8	78.5	-12839.0	35.6	-121.2	40.2
	24	-72.6	96.6	50.2									
7	0	-87.9	-267.0	52.9	-2745.9	-323.7	-121.0	75.7	-26.6	79.8	12.7	193.9	83.4
	12	-103.0	52.6	162.9	16.6	-182.0	28.3	-174.7	60.9	-265.9	57.1	-139.1	-82.3
	24	-14.7	89.3	-10482.4									
8	0	-526.2	-127.1	-93.6	-35.2	114.1	27.9	87.4	6.1	78.2	-85.9	-7085.2	102.1
	12	84.3	42.8	47.2	-96.4	168.6	572.4	-189.7	-179.1	64.9	-84.3	33.6	1498.0
	24	-94.9	-185.0	204.6									
9	0	-1592.1	-451.6	-46.5	1196.9	-3.2	139.5	174.9	32.5	208.7	-75.0	11790.1	57.6
	12	256.8	66.7	61.3	262.5	565.0	1376.2	195.1	-359.7	323.6	-41.2	-197.5	707.4
	24	-310.3	53.7	-942.4									
10	0	-49.2	3156.0	-0.2	45.3	-26.8	24.8	-54.6	-111.0	-0.6	55.4	-96.7	99.2
	12	-117.6	-19.7	-9.3	6.3	-2.2	-15.3	-72.0	42.2	-27.7	49.4	77.9	-218.2
	24	-43.3	-108.9	-3262.4									
11	0	72.1	-5078.9	-4782.6	-54.3	69.2	-100.3	-120.2	-22.5	198.0	464.3	-111.2	32.1
	12	-279.1	-236.9	177.2	133.3	193.1	221.2	-352.8	-496.8	54.2	-228.5	-154.6	-49.3
	24	101.3	713.6	360.5									
12	0	355.2	410.8	200.4	-90.4	110.6	-5223.4	176.9	-73.8	-21.5	-16.5	106.9	-8.8
	12	-147.7	7595.6	65.9	-40.1	-90.5	95.3	425.1	-137.6	78.5	-138.3	-67.8	-285.4
	24	-58.4	362.1	-6817.3									
13	0	-311.2	-88.4	54.0	239.3	-13.6	-64.8	-46.3	-369.7	-77.0	49.6	-86.6	7076.6
	12	-70.6	-391.0	-16.5	7255.3	87.2	-574.4	-52.1	1064.7	152.5	197.9	-41.9	-72.6
	24	89.9	-103.1	430.1									
14	0	765.9	858.6	256.3	-26.8	-822.7	137.6	201.9	137.8	111.1	10583.7	-45.3	61.3
	12	161.8	-284.0	-190.2	19.3	104.3	53.0	187.0	-957.1	205.6	315.3	462.5	-86.3
	24	-49.2	424.3	1146.5									
15	0	249.0	-2963.2	-129.8	232.5	-428.5	-9038.3	-392.0	412.2	183.2	-455.4	-39.3	315.5
	12	293.7	-96.0	-25.0	-60.1	-144.3	43.8	136.1	214.5	-587.1	359.6	-14.3	-142.2
	24	-0.4	-30.5	-80.1									
16	0	-291.2	-160.9	-61.8	59.6	1.0	70.6	132.1	95.4	-21.6	-81.7	-177.7	-75.4
	12	-43.8	-30.1	-8.4	647.1	29.8	37.1	42.8	11526.9	-111.4	156.8	43.1	-46.6
	24	-4.0	153.5	-67.2									
17	0	239.4	312.0	106.7	107.9	2621.9	-52.4	121.7	-27.4	126.6	72.5	92.6	127.1
	12	-5.3	72.8	257.8	0.1	166.2	-155.9	-15.8	-23.5	288.4	1.7	-109.9	-211.1
	24	106.7	-289.8	1498.9									
18	0	498.0	222.8	-110.5	-390.6	-5536.2	-40.3	424.6	-98.3	269.8	68.6	175.5	-6.0
	12	-21.4	128.7	322.0	-347.8	-151.8	-157.7	-611.2	-312.4	634.2	-94.0	88.0	175.0
	24	-1.5	-852.7	2641.4									
19	0	419.3	112.6	243.2	-99.3	407.0	116.4	-195.3	-59.4	149.3	-35.5	-207.7	1.4
	12	166.1	-45.3	221.2	161.2	-254.0	-80.2	11945.5	-5.3	-127.8	-20.4	-109.0	44.0
	24	-44.1	-59.1	125.7									
20	0	56.3	4759.0	244.6	-179.4	-10.3	-90.7	-49.8	-17.1	-429.4	-82.8	-76.9	171.4
	12	44.0	208.6	-4.1	-282.0	-432.9	-7275.9	-116.0	-628.3	-342.8	-389.9	-501.0	-52.0
	24	-130.0	-112.2	801.3									
21	0	64.9	-944.0	326.1	11.5	-65.5	16.7	-95.7	67.1	166.7	79.3	46.7	-39.7
	12	101.2	64.0	-57.8	-713.3	-105.6	-7860.0	49.6	1060.3	212.2	149.9	70.7	192.6
	24	299.8	87.5	-1137.5									
22	0	-222.9	175.5	-87.8	-169.6	-18.4	-12.6	2.3	-35.6	-142.4	-563.5	-24.0	110.4
	12	49.2	89.9	-192.2	-202.9	7167.3	65.9	-197.5	-69.7	-49.7	311.9	432.4	3.5
	24	259.4	-187.5	-106.6									
23	0	-197.9	-5502.8	4759.7	-98.8	4.3	102.4	-47.8	104.9	-2.5	339.9	265.7	-51.7
	12	-6.7	-208.2	-37.4	-162.1	100.2	-225.1	-151.3	446.9	127.7	4.5	102.8	-192.4
	24	-75.6	217.6	185.8									
24	0	63.6	-97.3	-110.2	-144.2	-98.1	238.4	-33.5	-94.5	55.3	-283.3	-1.4	-5900.5
	12	-147.6	-94.5	301.9	-60.2	67.5	-248.1	-156.9	-553.1	-275.4	-120.1	89.6	-236.4
	24	108.6	-183.9	588.7									
25	0	38.2	122.2	1491.5	301.5	-322.2	-14.2	346.3	434.4	289.1	-77.7	46.3	257.6
	12	57.8	132.7	18.9	233.5	-220.3	7379.9	104.6	1181.4	429.0	-157.9	228.5	462.7
	24	447.6	279.3	-1711.3									
26	0	-263.7	172.6	-99.3	-5459.4	4988.4	-9.4	364.1	-429.7	163.7	-486.0	127.7	-162.7
	12	-316.7	-309.0	-28.6	-19.6	-348.8	495.1	213.0	410.6	-382.5	42.1	312.8	-59.5
	24	-56.8	13.8	-1065.4									
27	0	-8.1	831.4	9381.7	299.2	537.7	107.6	225.3	-116.5	-109.2	107.5	-133.1	-59.2
	12	400.3	259.4	482.3	-621.4	89.3	-3.2	221.2	441.5	-171.3	-396.6	166.0	446.4
	24	-100.0	191.6	1420.4									
28	0	1118.0	8065.8	167.2	164.3	-105.5	277.6	-20.4	-252.3	-114.7	439.3	81.4	-5.6
	12	126.9	171.2	-235.9	169.2	83.5	334.4	-219.7	466.3	-1041.1	229.4	109.0	116.9
	24	255.9	-243.0	450.8									
29	0	827.0	409.9	-30.8	75.1	134.9	150.2	-82.7	-168.7	-13639.9	-130.8	-177.6	-166.3
	12	152.6	68.6	-143.6	63.3	2.5	-96.4	122.5	-596.2	-67.7	2.8	38.1	29.1

	24	-100.2	-214.0	-1544.8									
30	0	-1259.0	-284.3	18.9	-124.0	75.4	169.1	97.4	-67.3	277.9	182.4	62.3	9144.1
	12	98.9	124.1	231.9	-1149.6	31.6	-56.2	-41.5	-1361.7	-1536.7	-133.2	152.6	53.1
	24	-68.5	280.8	-562.8									
31	0	121.9	199.2	-60.4	59.5	-232.2	148.0	-94.8	-170.1	-4871.9	194.0	-103.3	192.4
	12	-129.6	3.3	60.1	-131.4	-33.1	-22.5	125.7	-19.2	323.6	23.6	89.0	-19.7
	24	78.9	83.9	1243.5									
32	0	1223.8	-174.4	-440.1	942.1	9209.6	-46.1	-123.2	190.6	-56.8	314.0	0.7	-144.3
	12	-262.0	-94.3	-190.2	83.4	-244.7	-131.2	8.7	955.3	598.2	1.5	203.6	163.2
	24	-49.8	342.1	780.9									
33	0	192.1	-59.2	4030.9	206.5	46.0	7781.5	-236.5	-199.6	617.1	165.9	97.6	-194.2
	12	-36.5	-460.9	63.6	73.1	11.1	-279.1	56.6	-8.3	-279.7	155.1	-218.8	234.3
	24	127.8	-113.7	541.5									
34	0	133.9	-1050.4	251.2	171.8	-22.3	-78.7	-38.9	31.3	26.0	186.4	-334.8	-67.9
	12	220.0	-28.8	224.0	-23.8	16.6	309.2	-293.3	-90.2	-109.5	103.7	-61.5	-49.8
	24	-302.7	16384.0	-2033.4									
35	0	163.3	-30.8	-479.7	-122.1	-342.2	102.5	-60.1	93.2	-57.3	142.7	-34.8	-49.7
	12	-230.8	76.0	111.4	-340.5	-1777.2	79.6	182.5	457.9	-269.3	458.7	-201.8	286.2
	24	8766.8	-106.7	1398.0									
36	0	240.1	18.4	67.4	-5453.9	-216.0	-81.9	261.2	14.7	-27.3	81.3	8724.1	53.8
	12	96.8	-157.1	134.0	-125.9	16.4	202.3	177.9	55.2	6.8	-99.2	-113.9	-221.8
	24	68.3	88.7	-607.3									
37	0	-152.7	-214.3	3098.8	284.4	-178.7	-57.0	-42.3	-72.5	61.3	4.0	88.6	48.0
	12	-66.7	30.5	13.7	4.7	-150.3	-150.4	-122.5	-230.5	-512.9	-89.7	67.1	-446.5
	24	-7.1	-19.9	2664.9									
38	0	-117.3	-44.2	190.0	642.4	-167.2	-84.4	65.7	-32.7	18.1	60.7	-60.3	6.1
	12	46.0	657.3	2.8	-160.2	99.1	110.9	68.3	165.0	87.0	-98.0	-10537.8	-7.6
	24	132.0	180.4	-50.8									
39	0	-62.4	72.9	427.5	406.6	-278.5	214.9	-71.0	31.2	225.4	-292.4	-324.5	-180.5
	12	7393.7	-92.1	-250.8	-153.4	-467.4	405.2	-497.5	455.4	-107.4	-545.4	396.4	-215.4
	24	-21.9	761.9	-359.1									
40	0	-204.4	-281.8	-38.0	-86.5	-116.6	40.1	-25.4	141.8	87.0	641.9	143.8	-169.9
	12	-8.8	140.5	476.4	-140.8	-154.1	-152.8	149.6	-45.7	-186.2	295.5	-13.7	-125.9
	24	-11496.7	467.5	1212.6									
41	0	-900.7	36.6	5155.5	-96.7	-40.1	-56.0	-7719.4	-111.9	113.2	-81.6	175.2	-295.6
	12	-166.9	-8.1	0.9	415.7	-101.7	72.7	26.8	445.1	150.1	226.5	-16.4	-192.5
	24	-407.8	-327.4	-106.1									
42	0	1693.2	-99.4	-69.9	-26.2	-191.7	120.8	27.6	-4.5	-7.7	-44.1	-44.0	183.7
	12	-21.8	126.9	90.3	-70.0	94.7	-79.0	104.1	-139.0	31.0	4.5	72.9	-324.2
	24	-135.6	-808.6	16384.0									
43	0	-256.9	-446.5	-478.2	-0.2	-157.6	66.0	36.6	17.6	-90.8	95.1	-45.3	-245.4
	12	56.6	-27.7	124.9	-12511.2	-109.2	355.6	-235.4	726.3	202.0	1.0	51.9	-338.8
	24	153.1	62.4	681.1									
44	0	545.4	424.7	117.5	-477.4	67.2	309.7	-90.9	-42.5	59.1	-173.3	-286.2	89.4
	12	51.2	7016.6	26.2	-43.1	22.2	-157.9	-406.1	-73.2	-480.8	-77.4	-16.0	114.2
	24	25.0	75.9	-16384.0									
45	0	-327.9	-120.6	20.6	-54.4	29.2	-226.0	66.5	-430.2	204.2	-182.2	-7.6	-14550.9
	12	-95.7	-275.5	1052.1	641.5	-88.2	315.4	-233.9	-468.9	-475.7	-59.0	180.2	188.6
	24	144.3	230.2	1420.6									
46	0	-4574.7	89.5	-27.1	-226.7	-252.3	-41.0	123.3	-140.2	-74.1	19.2	16.7	44.7
	12	-35.9	-224.0	41.3	6.1	-79.7	159.4	-6.0	340.2	-78.1	-277.8	-47.3	-17.8
	24	-2.0	-47.6	335.0									
47	0	-3388.1	405.6	79.2	4543.3	-255.2	262.4	61.6	105.4	97.6	-401.7	-311.7	-11.3
	12	-27.3	127.7	229.9	-283.0	-262.6	-10.7	33.5	5.2	105.2	-54.8	453.7	-92.7
	24	-107.7	480.7	-349.6									
48	0	-5229.3	226.9	-220.5	18.7	6407.0	187.4	-80.5	88.4	-136.7	-118.3	756.7	-55.5
	12	-42.0	2.9	279.8	-226.9	-22.2	-49.7	85.2	-569.5	-113.8	123.1	-19.8	44.6
	24	10.5	-504.8	-290.5									
49	0	-533.2	120.6	149.2	40.6	-29.3	59.3	-7.5	7150.3	-16.2	-112.8	-69.3	204.6
	12	-18.0	-128.5	-142.1	-64.1	-240.4	-152.1	318.4	-67.4	1746.3	16.8	34.9	-39.2
	24	1.8	257.8	-113.6									
50	0	9.0	-5060.6	-225.4	-210.1	-412.6	5020.2	211.3	-33.6	293.9	-95.2	-316.7	213.4
	12	388.0	-283.0	25.4	-332.3	-132.9	-204.8	96.3	-86.5	-327.6	267.6	116.0	1.9
	24	-171.3	257.4	-254.5									
51	0	-99.3	-48.2	-68.5	37.3	-158.9	-16.2	20.2	352.1	-41.3	-88.5	20.5	102.8
	12	-31.5	51.6	-14049.7	-64.3	-24.3	82.3	60.7	-157.9	-12.8	1564.4	279.5	-68.9
	24	123.0	48.7	50.9									
52	0	-407.9	-464.4	35.3	43.8	-270.4	82.4	84.4	-82.8	53.0	49.4	246.5	-17.0
	12	-245.6	-98.0	-70.1	101.1	279.5	-128.4	-64.5	-20.8	136.7	-307.0	155.4	348.1
	24	60.9	-12720.5	-2379.6									
53	0	304.4	791.7	-1227.3	520.7	-115.8	4853.8	-243.2	-85.7	267.8	-527.1	221.9	-85.1
	12	7.3	280.4	-37.4	65.4	-69.5	101.0	-700.0	-110.8	-108.4	-255.3	-57.4	1.2
	24	-44.4	351.4	5531.7									
54	0	601.6	288.0	-78.3	126.0	-61.1	-96.5	261.4	-90.2	8799.5	-42.5	-87.0	-88.3
	12	169.9	-0.2	-75.9	-709.5	-60.8	188.3	31.0	-430.9	607.3	-64.1	144.3	18.0
	24	51.0	243.3	1503.6									
55	0	686.2	-63.8	-288.2	-133.4	-77.7	335.3	8771.1	88.9	-108.0	-93.0	45.0	-5.0
	12	-247.9	-123.5	114.8	759.9	84.5	366.1	-160.4	-1082.9	168.5	136.5	-97.6	73.2
	24	-350.0	-32.4	-354.1									
56	0	68.4	53.7	-299.3	8521.4	-46.4	-274.6	-338.7	-68.4	4.2	-202.4	-274.6	106.8
	12	-157.3	-75.3	281.3	-39.4	-166.2	203.4	-25.4	-81.0	375.6	-70.4	16.1	-124.6
	24	-162.4	-335.8	-299.5									
57	0	-699.2	825.8	-6696.1	323.9	596.6	76.0	164.8	-213.4	-255.4	-214.4	415.2	-22.5
	12	78.2	-270.0	-2.5	-57.7	-81.8	165.3	-85.9	82.3	-108.1	113.0	-173.7	-208.0
	24	21.3	-96.2	3208.5									
58	0	-464.0	190.1	-535.9	-59.3	-62.3	-74.5	29.9	-115.0	140.3	-97.1	200.1	-736.6
	12	12.2	9.3	200.3	9792.4	-5.3	1014.5	48.1	-522.6	47.3	85.8	23.7	-47.3
	24	81.6	-115.8	585.5									
59	0	-40.5	142.5	-883.7	-5713.0	-967.1	-63.5	-3228.4	404.2	-166.3	13.6	-46.8	39.8
	12	42.5	-271.7	-126.4	232.3	-55.6	-60.7	37.8	-76.9	-161.0	-5.1	192.7	-58.4
	24	-324.7	-48.9	306.0									
60	0	4202.3	-98.0	-228.0	4866.5	-189.1	35.3	213.0	51.0	136.9	135.1	-160.0	-131.5
	12	20.9	-302.1	-109.5	-476.1	-76.3	176.8	-528.5	-879.1	620.8	-124.0	268.8	0.7
	24	36.3	304.2	-336.9									
61	0	170.9	-47.3	452.0	-18.9	86.9	-6.8	106.3	85.7	0.7	22.7	-398.2	18.3
	12	-124.7	-30.4	-181.6	48.4	440.7	112.6	-64.7	291.5	-57.4	61.4	3.1	11820.7
	24	21.2	-159.9	-426.8									
62	0	8348.6	712.1	-289.9	-167.4	108.6	32.1	294.9	-124.5	47.9	99.1	-47.3	11.3
	12	8.3	-173.3	79.0	-209.9	-251.8	17.1	12.3	-775.3	-161.3	-258.0	-90.7	457.3
	24	81.1	346.5	850.4									

63	0	5.1	-89.0	-316.2	300.9	9.8	152.3	-4259.8	-139.1	138.4	-150.9	-174.8	-123.6
	12	-12.8	-496.4	-130.0	245.9	123.3	5.8	55.3	-122.4	117.6	214.8	-63.8	-526.9
	24	102.4	-3857.9	-1225.4									

Table C. 3 Shape codebook 0 for MODE_VQ == 24_06 / 24_06_960, BLOCK_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	651.4	1015.7	-605.5	1101.9	-832.6	10033.8	350.6	222.0	-136.4	-555.2	585.6	-2.5
	12	-163.4	218.1	46.4	-446.2	468.4	-646.5	-728.4	-554.1	386.2	-513.2	-4.0	470.8
	24	3.7	-2.0	1.3									
1	0	-3020.3	4818.8	6247.1	5230.1	746.3	-704.3	556.2	711.9	1590.9	347.9	-152.8	404.4
	12	726.1	-57.7	13.3	185.4	169.8	722.6	-36.1	179.2	202.8	-289.4	481.1	205.5
	24	-31.8	-7.8	-9.0									
2	0	-343.3	-509.5	-737.5	7296.6	-3957.4	1106.8	-975.1	-20.9	1195.0	791.2	-382.9	86.6
	12	820.4	-123.7	-83.2	38.2	-398.4	-93.5	403.0	336.9	-20.9	-44.8	192.1	36.4
	24	-0.1	2.6	-4.4									
3	0	303.5	3451.6	-1959.0	1702.9	-1544.3	1769.6	1930.8	6192.2	451.7	280.6	-1961.9	-982.2
	12	-368.2	-397.6	-214.7	1570.5	436.5	402.2	-58.4	-426.4	-658.4	87.4	-368.4	113.7
	24	1.9	0.9	-1.5									
4	0	1394.1	-2802.7	886.8	1295.4	-1690.2	430.5	-6678.1	581.5	-1462.2	2439.7	332.6	288.5
	12	249.7	-905.0	-0.7	443.5	7.2	-308.2	793.2	-368.7	-1633.0	-4252.4	-370.3	-9.4
	24	3.2	0.3	1.1									
5	0	-1548.2	3070.3	-1972.7	-25.2	762.1	3148.3	-3680.5	-241.8	3816.3	1046.1	-840.5	180.5
	12	-175.2	-2798.2	-306.8	729.7	2281.5	493.5	-1053.4	-259.2	-1763.4	-1047.3	112.8	845.5
	24	-0.4	-0.0	-1.1									
6	0	426.8	252.0	-9.4	276.4	-11429.9	262.2	-275.6	-822.8	2.9	395.4	1273.0	358.1
	12	-547.8	315.4	-4.2	-438.5	-233.8	-422.4	-229.4	-388.1	-406.1	-438.1	322.5	-233.9
	24	-14.1	1.7	-1.5									
7	0	1904.5	3314.9	-5206.1	-3323.9	4609.8	2188.7	-1364.5	267.0	346.4	-604.0	96.2	-198.6
	12	-582.9	-12.0	417.5	-36.4	-269.1	58.4	248.2	-29.8	407.9	139.1	-385.2	320.7
	24	11.4	-7.2	8.9									
8	0	852.5	60.8	-167.5	-138.0	1030.8	20.0	-750.6	964.6	147.1	-1422.3	643.5	939.8
	12	-930.2	362.2	8741.1	-281.1	-1235.6	-788.6	79.6	-976.7	-56.4	-396.9	-199.4	-132.4
	24	-4.4	0.8	-4.2									
9	0	3157.6	-261.6	-1534.4	2597.2	7531.6	252.6	-1386.3	-2387.4	-426.6	-99.4	166.2	-535.7
	12	212.2	-609.6	-69.9	482.2	-420.2	328.0	-78.6	-122.7	-36.9	536.5	-139.3	-205.8
	24	3.0	3.6	-1.9									
10	0	258.7	-1438.7	-256.9	612.5	547.0	-388.2	515.8	341.2	820.5	183.9	1277.2	1533.0
	12	-1891.0	-254.9	100.9	-1545.8	7404.9	263.9	-33.1	76.0	-219.6	-952.5	-235.6	-1019.5
	24	-0.5	-1.1	1.1									
11	0	9640.9	4572.5	-5729.1	5191.8	-1490.1	1398.9	-607.7	113.5	-340.3	193.5	-32.1	184.4
	12	230.2	-165.4	217.2	-53.3	56.2	-190.3	274.1	-163.3	46.1	-41.1	10.2	-129.4
	24	-3.5	-7.9	-9.3									
12	0	691.2	-448.7	110.3	-19.5	-737.2	-99.1	-788.6	175.0	-437.9	-87.0	1838.8	9401.7
	12	-1523.1	-1163.1	184.2	-834.4	-605.1	-763.6	1258.6	-148.6	-1129.5	246.4	-225.8	151.9
	24	-3.7	-1.0	2.6									
13	0	-4148.6	993.1	-3108.5	2078.0	3796.7	-988.6	-2332.2	679.9	393.0	732.3	-126.9	-310.4
	12	-419.6	-101.9	401.9	-59.2	147.2	-696.6	-180.4	300.1	-475.5	242.2	130.5	-306.2
	24	-4.0	-0.1	0.4									
14	0	-172.2	65.5	18.2	-1575.5	-922.8	-736.1	-471.7	-498.7	-315.4	-82.9	-229.7	-204.4
	12	756.1	726.8	606.3	418.1	250.1	-0.9	-8051.0	-142.9	731.8	299.1	-35.7	159.0
	24	-1.9	-0.3	0.2									
15	0	3660.0	326.1	1532.2	684.8	683.5	-1272.6	-2601.8	-2525.3	325.7	-423.0	-463.2	-765.4
	12	-3791.1	-645.6	602.8	-448.0	281.8	2117.9	-748.1	-1435.1	3082.8	854.5	272.2	2855.7
	24	1.1	0.2	-0.5									
16	0	730.0	-203.8	-100.7	323.2	-165.4	806.5	645.0	-626.2	-1510.4	-1920.7	9022.0	-798.2
	12	784.5	35.0	410.8	-124.7	514.2	137.7	-1009.0	-97.9	-1210.5	-246.1	81.8	864.5
	24	-0.6	0.6	-0.3									
17	0	6405.6	1832.1	-2248.3	-911.8	168.7	903.9	-66.5	205.8	-170.2	-89.0	226.9	-50.2
	12	-6.0	-83.9	204.7	111.2	-91.2	-74.1	259.7	39.5	-47.2	47.3	-92.9	25.3
	24	11.9	-0.4	0.8									
18	0	-140.2	-1477.1	-308.2	890.2	-373.2	1097.6	-1874.2	-967.2	2483.2	-4073.1	-4079.6	-1632.8
	12	2366.1	-1547.1	1393.9	-133.2	-143.4	723.1	-915.7	-938.0	-2318.3	384.2	417.8	3155.5
	24	-1.9	2.1	0.8									
19	0	2071.8	-284.7	-2099.3	3975.8	139.0	33.1	-458.5	282.2	-411.1	-16.1	-160.6	391.6
	12	260.9	-129.7	-89.6	-57.5	100.3	51.9	45.7	-218.8	-51.7	93.5	112.9	-132.6
	24	0.5	-0.1	0.2									
20	0	504.7	-920.3	613.2	-464.1	-712.6	415.3	202.9	321.9	-949.4	-1301.5	505.7	-386.0
	12	-218.2	-308.4	81.7	-9125.3	-291.2	-430.1	-1185.9	-360.7	484.0	-171.4	-9.8	-66.2
	24	1.8	0.5	-0.2									
21	0	1218.4	-439.0	-831.7	-9659.7	176.9	2485.5	838.2	667.6	1748.1	13.9	391.9	-13.6
	12	-519.5	-376.5	-208.4	622.9	-191.9	-163.9	663.9	56.3	218.8	-121.9	-18.8	-124.6
	24	3.6	1.5	6.8									
22	0	-5200.4	42.4	-132.7	71.9	1909.9	7778.3	-600.8	-910.4	-357.4	-668.8	542.2	-1235.3
	12	-759.6	255.6	661.1	-217.2	154.0	-479.0	128.9	419.1	-87.7	209.9	475.7	10.4
	24	-2.3	-0.7	0.8									
23	0	3844.1	-2617.0	-2711.2	153.1	2228.7	347.8	306.6	93.9	19.6	364.8	63.2	-158.6
	12	337.0	165.4	-72.5	13.2	162.8	-362.3	43.3	-121.0	-178.2	174.2	188.3	50.5
	24	0.0	-0.5	-0.2									
24	0	1094.8	816.4	-1016.5	-1641.4	-346.8	391.6	-2750.7	-168.7	-1762.3	2143.9	661.7	-3749.9
	12	-534.3	-892.7	-1024.5	-1465.5	-365.0	752.9	795.6	-393.9	-889.8	358.4	428.9	-6666.8
	24	1.8	-1.5	-0.8									
25	0	6947.5	1655.3	3098.6	-4502.1	13.9	437.7	80.4	-520.5	-181.4	119.3	18.7	162.7
	12	178.8	-19.4	81.1	-83.3	-48.3	-151.6	172.5	206.2	60.6	-300.9	-38.2	-36.6
	24	9.6	-0.6	2.9									
26	0	830.7	-1599.9	679.9	11.3	-962.8	3498.2	-4593.9	-3007.1	-394.3	-850.9	-209.2	83.3
	12	-206.9	-119.0	104.6	741.7	-319.2	-101.2	96.4	-6107.0	-1201.3	476.8	-140.6	-86.9
	24	0.9	0.1	-2.5									
27	0	-1112.2	-4476.6	-4730.2	4114.1	3349.3	1058.1	33.6	-730.7	766.8	932.9	46.0	-50.2
	12	-466.0	24.5	-314.1	48.8	376.0	-187.4	89.1	13.6	127.0	-76.6	210.7	-57.2
	24	1.2	6.3	1.4									
28	0	-583.4	-428.8	136.3	255.1	-754.1	58.6	223.5	-310.7	510.0	87.7	-37.5	-47.7
	12	-1585.7	-9243.6	-905.8	599.1	-1276.2	-28.2	206.5	-759.5	397.6	171.4	163.8	26.1
	24	-1.4	5.6	4.5									
29	0	856.0	-5488.9	159.2	2036.4	1223.3	51.8	178.4	-360.9	-221.4	60.4	134.7	-72.1
	12	62.2	-71.9	-109.9	-364.1	31.0	42.1	-137.9	-61.1	-55.9	22.5	-81.9	-215.9

	24	2.1	-1.2	-0.4									
30	0	-177.1	-55.1	132.9	337.1	151.1	-178.9	17.4	896.7	-465.7	426.6	526.1	1384.4
	12	8898.0	121.2	812.1	-281.4	-1248.0	581.3	260.1	333.2	315.2	-605.5	-766.0	798.5
	24	-1.1	-0.4	0.2									
31	0	1158.6	-103.5	2723.0	-64.2	-115.7	1262.0	8304.4	-1755.2	1756.7	1018.1	1091.5	-389.3
	12	453.6	121.4	-694.3	650.7	78.9	-882.5	-138.7	1314.2	-698.3	429.9	65.3	385.5
	24	-1.2	-0.3	-0.1									
32	0	-772.8	-233.2	2323.9	-273.8	-1644.3	-3626.3	1317.3	1352.3	-4311.1	858.1	-894.5	-1193.5
	12	1505.5	-1890.7	-606.5	1985.4	988.9	1758.8	1300.5	-1763.3	-665.8	-298.3	-48.7	-99.0
	24	0.7	2.1	2.8									
33	0	436.0	-7841.0	1194.7	-2534.4	-2460.5	189.0	14.9	-571.3	265.1	197.5	58.9	267.7
	12	108.0	263.9	-223.1	-66.0	-27.8	-205.2	116.5	139.6	50.4	106.3	-109.9	165.6
	24	-1.2	0.6	-0.8									
34	0	-1772.8	-1517.2	2189.4	-951.1	-246.5	-819.2	1479.8	787.8	4159.7	6159.7	266.5	-687.1
	12	-570.2	578.8	114.3	1594.6	-463.1	-702.9	-1047.0	-379.8	-715.2	542.8	-838.6	811.2
	24	-1.9	0.2	-1.0									
35	0	6359.6	-1700.9	3805.2	2945.1	-8.5	-306.2	161.2	457.7	166.4	221.2	4.1	2.3
	12	-14.3	94.7	86.1	169.3	93.9	26.0	59.0	-66.5	-172.4	-40.4	38.1	185.4
	24	6.4	-0.2	1.5									
36	0	163.3	3655.4	-83.2	2989.4	-2699.3	-4750.4	1533.1	-2323.7	498.0	106.6	692.8	-1101.9
	12	1116.1	-105.4	162.6	-39.3	172.7	69.7	70.9	-186.9	470.8	608.3	296.0	767.5
	24	-2.8	0.5	-1.2									
37	0	-124.2	-6635.8	7060.3	3969.7	1082.0	-7.1	-1768.6	-678.0	396.3	-854.9	572.9	85.4
	12	-148.5	127.9	-16.4	-327.4	-232.9	-125.2	-522.6	-322.1	-133.9	-319.6	-15.4	340.4
	24	11.5	-6.5	-8.0									
38	0	377.2	-397.5	4596.4	-346.4	-2362.5	186.1	-540.8	513.7	117.8	-407.8	-428.8	-152.1
	12	-215.4	-92.3	103.1	133.8	171.8	108.5	-93.7	69.6	259.0	-391.6	150.2	4.2
	24	0.9	0.2	-1.6									
39	0	5790.7	10973.6	1479.3	939.6	480.7	521.3	316.4	276.2	113.8	-157.4	257.2	330.1
	12	36.8	-48.3	92.1	-239.3	-119.0	76.7	-104.4	6.6	-88.3	-93.9	-7.9	4.8
	24	8.4	4.7	-4.5									
40	0	-112.2	30.7	-50.0	130.7	-795.3	-698.6	-757.0	-819.8	240.4	-171.0	-828.2	-542.3
	12	-173.5	388.4	347.3	134.0	961.5	-8967.4	-148.9	-613.1	-317.3	115.9	-198.2	228.8
	24	1.5	-0.1	-0.1									
41	0	10547.8	1791.2	297.5	-414.6	1883.6	433.3	113.9	-224.8	-306.3	116.2	265.9	-94.3
	12	85.7	-15.8	19.7	39.8	-76.9	-146.5	158.8	73.7	-150.3	-44.9	-65.0	-140.9
	24	29.8	0.6	1.2									
42	0	-16.1	-471.4	2505.9	-77.5	-1178.3	-2300.1	-3192.2	2106.2	2578.3	-282.2	778.9	3882.3
	12	1842.4	307.1	-2829.7	-837.3	405.8	1139.4	-1384.5	-117.3	-1394.4	1144.0	1417.1	-371.9
	24	2.3	0.2	-0.2									
43	0	10956.1	2477.3	5547.2	187.9	-1860.8	761.0	452.7	291.1	-250.8	-47.2	293.6	97.5
	12	-115.5	-50.2	77.1	74.8	-39.1	-210.9	28.5	2.4	6.4	-25.9	-82.2	-3.3
	24	72.5	5.3	29.2									
44	0	4237.3	3167.4	1137.9	4756.6	-1233.7	363.9	-374.6	378.3	82.4	-84.8	-252.8	79.8
	12	3.3	17.5	2.2	48.2	-47.4	-261.7	-55.9	-7.5	-341.7	63.9	-126.1	-156.9
	24	1.3	-0.8	-0.7									
45	0	9089.6	-2510.9	-3832.2	-1127.5	581.6	478.3	59.9	-727.4	-352.1	-137.9	-0.9	294.7
	12	146.3	14.3	335.7	-141.3	-92.8	-160.8	249.1	-36.7	3.5	107.8	-60.6	-70.9
	24	5.1	-0.0	7.7									
46	0	4218.5	-4437.1	8757.5	-5176.8	139.0	-6.3	215.8	-544.1	282.0	-289.0	-221.5	-3.4
	12	29.6	212.9	211.9	79.0	-90.3	46.0	4.2	-2.2	-231.5	32.5	84.4	-25.4
	24	58.4	4.5	-5.0									
47	0	7021.8	-11490.1	-2107.6	373.8	87.9	442.6	-188.0	-29.3	-314.0	97.3	-54.0	-46.2
	12	-73.7	-24.4	-40.8	-8.3	-5.5	-85.1	125.3	-48.4	-57.0	-57.8	41.6	65.7
	24	4.5	-15.8	0.6									
48	0	-2068.8	-544.0	2676.3	1573.6	2443.7	-239.7	-3766.6	-876.2	-1222.2	-406.4	425.0	-1180.5
	12	-399.5	-675.2	-1453.8	398.0	254.2	-150.6	-228.7	1068.2	-563.0	1306.6	-60.5	-4.1
	24	0.7	0.5	-0.1									
49	0	10420.0	-3446.1	1498.7	2987.4	-126.9	190.8	-59.3	-175.2	113.1	186.5	113.1	322.8
	12	34.1	-49.7	116.0	91.3	-41.5	-0.8	120.5	-71.4	-113.4	30.3	-32.3	-60.1
	24	7.2	-1.5	6.3									
50	0	-1932.2	-1213.4	10124.7	-1589.4	1560.3	49.1	442.0	49.1	-359.6	324.6	-222.1	-304.5
	12	235.5	114.6	-399.3	145.6	-261.9	-22.5	-43.3	297.5	-30.8	104.0	183.1	37.7
	24	2.5	-4.8	1.7									
51	0	431.3	2370.9	1311.9	40.7	737.0	1141.5	564.8	-2504.0	862.7	-117.3	-36.4	869.4
	12	-260.7	-755.0	-625.0	-165.4	-533.5	-850.2	21.0	63.7	-25.5	1068.3	323.9	-7942.6
	24	1.4	1.3	0.4									
52	0	721.7	1559.0	2613.2	3041.3	4741.4	-3270.7	1402.4	-512.8	-212.5	-354.6	-3.9	272.8
	12	327.2	211.5	67.5	-304.1	-69.5	-403.6	336.1	-278.8	238.9	-299.0	-229.6	-484.5
	24	-2.0	0.2	-1.4									
53	0	336.3	-956.9	953.0	-151.8	1664.6	1311.3	-1508.9	-1890.9	-7877.4	-553.1	-1722.3	361.3
	12	-795.8	596.6	-205.5	595.0	-290.4	-1008.8	904.6	-346.5	159.6	294.7	295.7	2293.5
	24	-2.8	-1.5	-0.7									
54	0	-1956.6	1177.1	-107.0	1288.3	-1072.1	-321.7	1771.3	-1945.4	-8191.5	496.2	359.4	1052.4
	12	635.0	-1012.9	7.8	-10.1	21.0	739.1	-920.4	641.0	127.6	-745.4	4468.1	-64.2
	24	-4.3	9.3	5.9									
55	0	-3306.6	960.5	732.5	3870.5	-1849.3	2193.2	4702.9	-1230.3	-230.2	-41.2	977.3	474.6
	12	-84.8	923.5	151.3	411.9	452.4	439.4	-676.8	-898.0	-305.7	335.7	-217.3	105.9
	24	-0.2	0.2	-0.7									
56	0	273.7	-1928.1	1076.8	380.8	3147.3	3889.5	1242.7	-1847.8	-87.3	3737.5	35.6	131.3
	12	-539.0	1547.0	1412.2	146.4	-498.2	1138.4	128.2	648.5	-3214.2	1318.8	382.3	-503.3
	24	-2.3	-1.5	-0.4									
57	0	343.9	1900.7	2838.5	757.5	3228.2	760.8	418.0	3658.5	904.0	-3377.5	1114.5	1021.7
	12	143.4	1491.6	-774.7	-1746.0	-886.0	-1402.1	-20.2	298.6	-1414.6	-701.2	-359.7	115.4
	24	1.1	2.2	-2.4									
58	0	883.5	-201.1	125.4	71.5	246.8	-484.2	-1246.4	-439.5	603.2	10448.2	-599.5	1311.6
	12	-1185.2	-288.4	78.9	1067.5	939.2	1052.9	432.7	1826.9	96.0	82.7	484.6	321.2
	24	0.1	-2.2	-1.3									
59	0	457.3	3356.7	2542.8	-2242.7	2245.3	2945.5	-2988.8	-1127.1	-686.9	49.0	-242.3	66.1
	12	515.0	-282.2	-22.4	-716.0	136.6	480.3	-537.4	-121.5	168.4	-431.5	-38.6	156.7
	24	1.4	-0.3	-0.7									
60	0	-509.0	1380.8	1144.2	7417.6	3293.3	3548.7	319.0	-2620.1	869.1	-426.2	-2643.6	1.6
	12	-309.5	-123.5	-708.6	-66.5	-484.1	81.2	34.1	-72.9	409.4	109.4	-591.3	-809.1
	24	-7.6	1.0	-2.2									
61	0	3218.7	3460.0	-230.1	492.1	2987.0	-775.0	-128.1	2984.3	-518.4	2177.5	-3979.5	1131.2
	12	1128.7	-133.1	-183.8	95.7	607.3	1023.4	-335.2	-1579.7	-169.6	653.1	669.5	318.5
	24	6.6	0.2	0.5									
62	0	-1238.8	608.8	531.6	418.5	-172.3	966.4	-337.0	-10202.3	228.3	534.6	22.7	326.4
	12	-559.5	804.6	-145.5	271.2	-106.1	-44.1	-48.0	3.6	-293.3	-589.3	587.7	123.9
	24	-4.1	-1.9	-1.7									

63	0	-1724.1	426.1	3547.9	297.9	-1893.1	7023.9	-1487.4	915.7	3267.8	30.2	170.9	-1521.7
	12	1367.7	-862.3	-232.0	-863.4	380.5	1829.4	655.9	923.9	-823.3	849.2	204.9	-65.7
	24	0.0	-4.4	-7.2									

Table C. 4 Shape codebook 1 for MODE_VQ == 24_06 / 24_06_960, BLOCK_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-13024.2	-9148.6	458.9	-93.1	1412.3	-111.0	-257.6	180.5	-163.1	-104.9	48.9	20.8
	12	-25.6	106.2	56.9	-187.3	-94.6	0.2	35.3	-52.5	-181.4	-11.5	-10.5	-30.4
	24	21.9	-3.2	3.9									
1	0	2269.8	753.6	8586.1	2480.2	-1771.9	1238.6	-1567.8	523.9	-69.8	-132.8	-255.4	271.6
	12	-282.6	-171.0	-0.3	270.7	464.3	-183.6	204.6	75.4	42.0	-554.6	172.0	59.0
	24	-5.7	6.5	-0.7									
2	0	91.0	-1071.2	-763.7	1101.1	-1641.4	-5468.2	-2437.0	-2543.6	-553.6	-742.4	1098.8	616.5
	12	-513.0	-427.6	900.4	-564.3	68.6	-1045.4	1415.8	856.6	-888.7	275.0	-1936.7	245.0
	24	5.9	-1.0	2.6									
3	0	738.7	992.6	-1495.7	-1389.9	-251.6	2595.4	-1723.1	-757.2	-664.1	2277.9	860.4	1994.2
	12	1854.7	1065.6	269.4	101.4	484.4	-3918.0	642.8	1424.7	2241.5	-268.7	1131.5	894.8
	24	-0.5	0.8	-0.3									
4	0	-187.7	891.9	-1004.0	2421.1	1891.9	773.9	2104.2	2748.5	15.9	-1478.1	3407.3	-772.6
	12	-734.7	-2696.4	-935.4	1867.9	1434.0	675.0	-124.6	579.1	146.5	-948.6	-1546.3	1799.9
	24	-1.0	0.4	-1.8									
5	0	1752.0	2951.9	261.7	3274.2	-858.1	568.7	7081.1	1116.5	-1993.9	711.0	-752.1	-1336.3
	12	-833.4	-590.5	569.9	-531.0	137.3	131.6	-1380.7	-858.8	-499.4	-580.5	-327.1	-178.7
	24	1.9	-0.8	-1.9									
6	0	-1845.1	3310.4	243.9	7228.2	865.1	-957.2	1087.5	688.4	561.5	-107.4	-544.7	78.4
	12	-259.0	184.3	-159.0	14.4	-109.6	-126.8	588.0	-156.7	27.0	-323.2	-244.1	-12.3
	24	-3.2	-1.2	2.5									
7	0	-1127.5	-1288.3	-1230.6	1141.5	4184.0	3589.8	6866.3	36.1	-1672.3	1338.1	-269.1	-101.3
	12	865.9	253.3	-333.2	244.8	-116.0	629.8	163.8	-32.4	131.9	-8.9	-185.2	-538.1
	24	-0.2	-1.8	1.3									
8	0	-1518.2	1323.2	-390.9	-88.6	318.1	1691.1	-1089.4	8046.3	232.8	0.7	603.2	-339.8
	12	-404.8	707.2	458.0	353.2	63.3	-558.9	985.9	-882.1	-264.1	488.1	-664.7	777.2
	24	-7.1	0.2	-1.5									
9	0	-1306.6	1058.8	164.7	1294.1	-9297.7	-1359.0	-603.0	386.9	197.3	683.6	-915.8	-791.6
	12	372.6	-638.5	-386.8	202.2	-959.1	32.1	191.6	-757.1	195.2	248.0	-321.2	-88.9
	24	12.5	-0.1	2.7									
10	0	-937.3	703.3	117.6	-872.0	900.9	-42.9	-997.5	635.7	-976.6	-205.1	1099.5	1968.7
	12	818.6	1697.1	-7577.5	-929.0	-251.9	-87.6	-919.9	-1970.0	30.1	1.9	-350.9	42.1
	24	-1.4	0.3	-2.7									
11	0	-615.4	-5108.1	-18.2	1830.2	-5119.2	2347.8	2228.8	1422.1	-865.5	-322.6	-1345.7	408.7
	12	-493.3	-578.9	-187.6	-98.5	-273.5	180.7	-170.7	-506.1	477.1	-290.7	163.0	-149.2
	24	2.3	5.1	1.8									
12	0	-98.8	-339.3	-1271.2	505.4	-321.4	-1010.5	471.5	2721.2	1115.1	-4.9	-652.7	-278.1
	12	116.5	-264.5	89.6	264.6	145.1	484.7	298.8	-365.2	-248.4	-770.6	7965.6	411.6
	24	7.3	-1.5	-0.8									
13	0	-795.1	-268.4	1782.5	582.0	424.4	-586.1	812.0	94.1	3313.5	-7285.8	13.0	1465.1
	12	-166.3	-521.0	-458.4	647.5	406.6	527.4	-326.3	-72.5	-84.4	-267.5	-253.0	119.7
	24	-0.9	2.2	-1.9									
14	0	-481.2	-928.0	-253.4	160.8	-996.0	-370.7	1070.1	302.4	1245.7	-119.2	-1557.0	-819.7
	12	-1855.2	7081.3	-1509.5	1925.0	418.6	1267.5	-34.9	286.7	443.5	-1027.3	269.6	-975.2
	24	-1.6	-1.9	0.8									
15	0	-319.3	-3748.1	-9076.4	365.3	-1974.9	1141.7	-305.0	615.8	-95.3	-154.2	90.7	-159.2
	12	319.0	544.2	-490.4	-200.6	608.9	300.2	-196.9	109.3	-40.6	-50.8	-153.4	81.1
	24	-11.8	-2.4	-12.3									
16	0	459.2	10.3	39.3	-455.6	-1769.2	-1615.6	-228.1	229.8	1979.2	-513.4	1173.8	-6793.2
	12	-666.4	-1642.6	986.7	140.6	-1095.8	-1068.2	-1200.4	-267.6	-1468.0	1583.1	-211.3	-86.9
	24	-0.5	-0.5	-1.0									
17	0	-5837.9	-3979.7	-3438.8	-2194.5	486.0	-34.6	117.5	-466.3	-28.0	-78.6	241.2	94.4
	12	324.1	24.2	212.7	-9.2	-26.1	-47.2	200.5	-1.6	-1.6	-2.5	-60.8	-12.7
	24	6.6	-0.1	4.2									
18	0	-291.3	-845.7	-2366.6	-21.0	-582.6	-398.2	1150.6	-3522.7	105.9	-144.4	-2682.6	4637.0
	12	-371.2	-860.6	-1721.0	-1300.9	-1159.5	-741.2	-51.9	-1007.6	376.6	-94.3	141.9	298.1
	24	2.6	0.7	1.4									
19	0	932.0	-9744.7	-2666.6	841.7	938.7	-1337.8	-280.3	517.7	-356.5	-243.9	164.8	328.0
	12	306.1	-315.5	-186.4	426.3	-302.1	147.0	-266.3	192.1	-78.3	332.2	73.3	-18.4
	24	-1.8	4.0	-2.5									
20	0	151.7	412.3	728.6	74.8	402.3	-1140.2	-449.2	-187.0	1044.5	2758.5	7945.8	741.2
	12	-544.6	-1186.8	631.0	359.9	-19.9	831.7	-70.2	-566.2	-4.2	1081.1	74.0	-80.2
	24	-2.4	0.4	0.6									
21	0	-3700.7	-15.1	-2607.7	3053.7	-65.6	-356.4	-50.1	-67.6	-254.7	123.5	162.2	1.6
	12	317.1	-47.8	71.3	-120.3	138.1	-48.9	112.1	-153.3	-256.5	129.9	17.1	-21.0
	24	3.5	-0.4	0.4									
22	0	-3384.1	1201.7	-503.8	1028.4	308.0	-616.4	-397.3	-627.0	165.2	4332.7	122.7	-1697.2
	12	-3025.1	873.0	-1412.1	-4611.0	105.5	-558.2	-932.1	-442.7	-71.0	-345.7	173.9	1314.5
	24	0.2	0.7	1.5									
23	0	-5555.3	-1171.5	3478.5	-525.9	-479.5	-464.4	348.1	259.0	-209.8	23.6	253.9	141.3
	12	267.8	-27.5	251.6	85.1	24.1	-50.4	276.1	76.0	-161.2	-156.1	-149.0	226.7
	24	6.4	0.3	1.1									
24	0	-2839.7	1052.2	-286.4	-1250.9	-1447.0	1204.1	1155.8	648.3	1173.5	-195.7	-3706.9	1255.7
	12	-806.5	824.2	3890.2	-140.4	1667.6	2085.0	-952.9	-636.5	77.0	-749.8	-148.3	-2033.9
	24	-2.0	1.9	-0.9									
25	0	-10798.0	1122.5	322.4	-764.9	768.3	63.9	212.6	152.9	-247.3	129.3	152.1	152.6
	12	19.1	-39.8	30.6	106.9	-33.4	-226.3	238.7	14.9	-88.3	-93.3	12.7	-92.4
	24	35.0	1.3	2.1									
26	0	-3342.9	-3056.6	864.0	-737.2	-900.5	4630.2	937.6	-1412.7	-1371.3	843.5	666.1	668.5
	12	953.0	-172.0	-434.8	-28.4	39.3	268.9	-33.7	-739.9	-437.2	104.1	-173.6	-626.2
	24	1.1	0.1	1.0									
27	0	-2531.1	1134.6	1230.6	-1426.2	366.7	-663.3	62.0	-5957.6	1061.8	-2921.3	-841.4	-603.1
	12	744.4	533.9	274.7	666.7	196.1	20.1	40.7	-538.2	356.9	2147.3	-147.7	-49.2
	24	0.0	-0.7	-1.2									
28	0	-696.4	-120.3	928.4	-399.7	-608.6	2135.6	-373.1	101.8	-1469.7	-4017.3	-1230.1	-1499.2
	12	-389.3	-1708.8	-835.2	-456.1	-131.4	-78.1	1675.9	6026.2	494.5	43.4	328.0	-1321.8
	24	1.6	3.8	-1.4									
29	0	-3199.4	6821.5	531.6	-2692.1	788.4	412.6	544.7	-177.5	-587.2	-95.2	38.1	-118.3

	12	-170.2	-120.3	-124.5	-31.1	173.3	-290.5	-74.7	26.5	-239.5	37.7	164.5	35.0
	24	6.4	-1.4	0.5									
30	0	-2620.2	1219.5	-3828.9	-4960.9	-2653.5	-582.3	-662.7	-534.8	-660.8	-308.1	259.6	25.2
	12	-128.8	-215.3	-168.9	101.6	-71.9	234.1	240.0	143.9	-906.2	510.1	316.4	520.4
	24	2.8	-1.2	-2.2									
31	0	-250.0	-801.2	467.3	-1838.1	1608.5	-535.9	2131.5	-2038.7	-2282.9	2088.5	-5305.7	-663.8
	12	-836.5	-1868.3	-242.7	-115.0	-475.9	17.9	-138.4	-379.2	-4047.8	-61.8	67.6	1470.5
	24	1.7	1.1	0.7									
32	0	-54.9	1712.6	2006.5	-9946.6	-249.7	-177.7	-1130.3	-832.8	-203.3	-450.4	-1166.8	336.7
	12	360.2	93.3	188.5	-119.8	-486.8	-168.7	28.8	43.1	-527.5	315.0	-53.3	237.2
	24	-18.0	0.2	-6.3									
33	0	418.9	8969.3	-1595.1	-1391.1	-2825.4	-916.0	-198.5	-391.2	-229.0	72.8	74.5	153.8
	12	227.7	368.2	64.2	-33.3	-211.4	-407.3	-333.2	-73.9	0.3	3.7	-228.9	47.9
	24	-2.6	-0.8	0.5									
34	0	1115.6	1555.0	-467.4	1432.1	118.9	1415.5	-9945.0	-785.9	362.7	-246.6	16.6	1041.4
	12	-43.7	-780.7	-18.9	-240.7	20.5	77.3	-301.3	502.8	554.9	-292.6	-43.2	430.0
	24	-2.8	-2.9	0.9									
35	0	3144.9	-2108.2	-4858.8	6331.6	-112.2	-290.1	185.9	725.4	462.0	110.1	-186.1	316.4
	12	-115.1	130.4	231.3	-78.0	316.2	85.3	-133.8	-271.7	32.2	-119.4	251.3	-139.6
	24	11.8	-1.9	-2.1									
36	0	735.1	-562.3	-222.6	-5445.9	1083.2	-624.0	1956.6	-561.9	1030.0	155.8	626.4	582.4
	12	317.3	-200.1	46.3	234.4	-451.6	144.7	196.9	165.9	-10.7	-1486.9	126.6	-34.4
	24	-0.3	-0.1	-0.6									
37	0	-9399.7	9324.1	-522.8	1008.9	-460.7	-135.9	-106.3	449.3	-104.7	84.4	-42.5	-217.8
	12	107.6	-123.8	63.7	-31.3	51.0	-205.2	77.5	-50.9	11.8	5.7	37.5	16.5
	24	28.8	-8.8	17.9									
38	0	-1719.0	-323.3	-2445.3	-840.3	2423.8	-7.9	-4994.5	2505.2	1494.3	-2477.2	1051.2	-882.6
	12	-1114.5	71.0	362.8	-272.6	203.1	1129.3	31.6	567.7	396.0	-334.5	102.8	-79.5
	24	1.4	1.0	-0.5									
39	0	914.9	-4801.2	-3017.9	-1608.9	597.6	3229.3	396.1	212.5	445.6	183.7	496.2	-492.6
	12	94.5	234.3	31.6	358.6	-425.4	674.5	-407.5	305.4	134.6	245.2	3.9	133.5
	24	-1.9	-0.6	-0.4									
40	0	2175.6	3385.9	-2005.2	302.5	-763.4	287.6	-630.6	-6444.6	3497.9	294.4	900.4	341.7
	12	-1122.5	-50.7	345.9	510.5	184.8	758.2	-105.4	-929.6	-418.2	-739.4	417.6	286.5
	24	0.7	-0.8	0.3									
41	0	4883.4	-3839.9	2371.4	324.5	1010.1	-225.1	-14.2	-735.7	163.3	-96.3	50.3	-23.3
	12	-24.4	59.7	2.7	-73.4	53.4	40.1	-157.8	76.9	54.7	-208.9	122.5	0.5
	24	-3.9	-0.3	-1.2									
42	0	-715.3	1732.8	-80.0	292.5	251.7	171.1	682.0	-139.7	-68.3	-441.8	-729.9	10132.9
	12	1051.8	-1991.8	-588.0	3504.8	-798.5	1182.8	-342.7	583.9	-561.5	1050.0	-146.1	542.7
	24	5.6	1.3	-7.3									
43	0	11837.8	-6904.7	1221.5	1462.2	548.5	489.1	-113.3	-169.5	365.1	81.8	-108.2	133.1
	12	-60.6	-5.8	-324.6	35.2	140.1	206.8	-139.5	-67.6	125.5	-12.2	120.9	127.0
	24	-42.4	12.0	-10.3									
44	0	4681.2	-1325.9	-807.5	-917.9	-2879.7	-312.9	-913.7	455.8	224.8	-143.6	-166.3	504.5
	12	-32.2	-132.7	122.2	-50.8	211.9	188.3	-90.1	-74.3	104.7	315.1	-124.7	-195.0
	24	-4.5	0.1	0.6									
45	0	6388.6	9743.9	-3173.6	1442.3	-1635.9	555.6	136.8	-206.8	238.1	22.4	-70.0	21.1
	12	-82.8	-101.8	-62.1	-139.0	53.3	-21.0	65.3	-42.8	93.2	-15.4	-47.6	40.2
	24	-21.2	-2.9	-17.3									
46	0	-2666.5	2939.9	1711.1	-316.2	-216.2	-143.4	-4322.9	1133.2	-2660.4	3804.0	79.1	1362.2
	12	909.3	-131.5	1094.6	-58.7	-421.4	386.1	-957.8	941.8	615.6	605.1	-436.3	806.3
	24	1.8	1.3	2.5									
47	0	9178.4	3480.6	-1041.2	-1594.7	-123.5	78.5	239.9	446.7	12.6	-317.8	32.4	-213.6
	12	46.7	13.7	-289.8	-150.2	99.3	-106.4	-192.0	153.6	159.9	-86.2	-50.0	92.6
	24	-8.8	1.6	-8.0									
48	0	-3619.3	-1105.6	1202.9	-1888.0	8345.1	-746.8	722.9	-48.5	633.4	203.4	1214.1	-306.3
	12	119.0	-138.3	80.3	473.1	624.3	-241.6	-359.2	-85.9	-239.8	195.0	193.1	-307.7
	24	-3.5	-1.6	-3.4									
49	0	13254.9	757.7	1757.6	3139.0	152.3	234.7	461.7	279.2	264.3	-183.8	21.9	91.9
	12	-26.3	-79.4	-254.3	8.6	96.7	0.2	-186.2	-45.1	103.4	38.7	94.2	-38.8
	24	-19.1	0.5	-2.3									
50	0	243.0	834.0	-414.4	401.6	-583.3	589.9	-524.9	-3.1	359.9	-536.8	1045.9	-3248.7
	12	7823.6	573.9	-682.0	-1753.5	723.6	-956.7	-403.1	-71.9	-931.7	13.9	143.0	-1293.2
	24	1.8	0.4	0.8									
51	0	1658.1	-460.9	359.3	-10.8	587.6	10342.0	584.1	-319.5	425.9	264.1	-144.2	-639.8
	12	520.0	-104.1	-374.1	72.0	245.9	-175.2	-17.0	-6.9	-390.2	808.8	29.6	-91.7
	24	0.4	0.4	-0.2									
52	0	109.4	838.7	-1446.4	422.4	-2051.9	49.2	-308.8	2128.2	-8206.5	201.4	335.6	-477.9
	12	-309.8	555.8	674.6	183.0	374.4	237.6	334.9	488.2	-50.1	995.3	47.0	-461.0
	24	-0.5	-2.4	-2.4									
53	0	-722.9	-346.7	6574.1	-1315.0	1982.9	-1053.5	495.2	1368.7	-209.7	-548.2	-402.1	270.8
	12	-152.2	446.5	81.1	-42.9	773.7	-419.7	184.9	-851.4	-98.9	-234.7	-36.6	396.2
	24	0.2	-0.9	2.7									
54	0	521.9	-1966.2	-860.9	-37.0	1235.0	-2445.3	833.0	550.9	1065.6	560.8	-3095.6	-641.6
	12	4032.5	-624.1	1085.1	-62.0	2778.9	-1200.6	492.1	1163.8	1472.5	365.7	-923.8	-1027.7
	24	0.5	-0.3	-0.3									
55	0	-98.5	-420.8	807.2	970.1	537.1	244.6	1251.6	295.9	643.7	-344.2	443.6	-1027.6
	12	-992.3	869.0	1193.6	-1149.4	-7703.3	-401.5	-308.7	794.7	603.8	117.1	-278.5	-862.7
	24	-1.3	0.1	-1.9									
56	0	1108.7	907.9	-4437.6	804.6	2724.3	-606.6	530.4	-292.7	-229.5	187.2	292.4	164.7
	12	-302.0	58.7	17.0	94.6	55.3	317.5	-48.9	-303.4	-200.2	100.6	-60.6	-17.0
	24	-2.1	-1.0	0.8									
57	0	-681.4	-6611.7	1239.2	-4991.4	420.3	-541.4	-566.9	754.7	-304.5	-822.0	265.7	470.6
	12	-300.2	-223.5	36.4	2.7	69.8	185.3	-0.8	-85.6	-40.9	420.0	94.1	308.9
	24	-0.4	0.3	0.7									
58	0	-165.4	-1468.3	996.3	2414.5	265.2	-1226.7	-1273.3	-1102.3	-2531.3	-3128.8	-257.1	160.7
	12	-2421.2	-34.2	-454.2	4769.4	271.2	-245.7	-2672.8	417.0	1079.6	-1620.2	-667.5	-2528.9
	24	1.0	0.5	0.3									
59	0	1051.1	2821.5	716.8	961.8	-329.7	-134.9	-55.1	-63.6	-328.5	-217.7	-325.3	-138.4
	12	-204.1	28.2	13.1	64.9	-164.1	-19.1	108.1	-71.8	-307.6	8.0	-92.8	72.7
	24	-0.2	-0.4	-1.0									
60	0	-2911.4	-4475.8	1890.2	3875.2	1419.2	-2636.7	133.4	-828.4	22.3	439.1	66.4	-320.6
	12	724.1	281.4	481.7	165.6	26.7	3.1	543.2	96.2	160.5	28.4	-11.0	-386.4
	24	3.5	-1.9	-2.4									
61	0	2558.6	3443.3	4947.8	-2838.1	473.7	157.6	1432.9	542.1	169.3	553.9	-39.9	-372.9
	12	-482.9	57.1	15.2	47.9	53.3	-249.0	-247.4	-10.9	55.2	-156.5	-28.7	315.5
	24	2.3	-1.0	1.8									
62	0	73.7	-2878.2	3755.2	2430.4	568.6	1410.3	-1521.2	1600.4	27.9	780.3	238.5	209.1
	12	-1011.7	-135.6	39.0	-228.4	-58.2	-210.8	-342.1	-357.5	113.6	315.8	-142.9	321.7

	24	0.9	-0.6	0.1									
63	0	3245.7	-1017.5	575.3	3536.6	-553.2	538.1	-1680.3	-2065.2	2717.5	4196.4	-335.1	-797.6
	12	510.5	529.3	-935.3	-490.7	513.2	523.0	-987.9	227.2	-459.4	84.6	-218.4	901.9
	24	-0.6	0.5	2.7									

Table C. 5 Shape codebook 0 for MODE_VQ == 24_06, BLOCK_TYPE == MEDIUM

Table C. 6 Shape codebook 1 for MODE_VQ == 24_06, BLOCK_TYPE == MEDIUM

Table C. 7 Shape codebook 0 for MODE_VQ == 16_16, BLEN_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	216	-10253.7	12.5	257.5	300.9	-54.1	-171.9	14.1	27.1	-14.2	6.6	-13.9
	12	-66.7	98.5	225.1	-193.4	-85	-670.2						
1	0	2548.6	-91.3	1198.7	-292.5	-322.9	-96.1	-41.2	-59.1	-84	12.9	-53	-316.5
	12	-26.6	308.1	172.6	509.1	-70.3	-651.1						
2	0	211.7	83.8	-11.9	44.2	-48.5	-68.8	-5558.1	-72.8	-18.6	-11.9	91.4	-363.4
	12	-402.1	79.3	147.5	-67.2	28	1231						
3	0	446.7	19.5	-2808.4	13.7	46.8	53.3	-10.4	22.8	108.2	21.5	-28.1	198.3
	12	93.9	195.9	-92.1	-14.9	-46.7	194.9						
4	0	376.7	6963.7	-26.8	30.8	46.6	52.7	38.4	28.2	-76.9	-15.9	-24	-207.3
	12	9.2	-338.9	12.2	54.4	-15.6	-196.1						
5	0	99.6	-187.5	229.3	-27.6	4.7	-42.1	1897.7	103.5	-350.7	77.9	72.4	259.2
	12	-636.9	76.7	-577.3	-2290.7	-40.2	740.7						
6	0	-27.4	-38	28.5	-68.5	162.5	-39.9	25.8	13.2	115.2	5	16.9	66.5
	12	-131.4	9099.9	47.1	66.7	-50	-222.8						
7	0	34.5	-30.4	19.2	119.1	-24.6	43.2	24.4	-21.2	38.9	-18.9	-34.4	-17.5
	12	8744.3	-61.2	34.1	200.9	108	-195.2						
8	0	58.8	25.9	6.9	46.9	122.9	-52	121.4	-3419.8	155	138.4	-90.1	87
	12	-159	20.8	84.1	309.9	-59.1	102.6						
9	0	159.1	409.6	-863.3	2457.4	-897.2	337	-42.1	25.7	565.3	-13.6	-17.9	-61.7
	12	172.4	68.1	81	264	-286.3	162.9						
10	0	-141.6	1117.2	-328.5	-1412.2	-7.6	36	1110.3	40.6	-93.3	-37.3	68.1	203.3
	12	-101	-40.1	-413.1	3308.7	-57.9	-421.8						
11	0	-119.5	-95.3	-70.6	-9516.4	-17.6	-82.1	-28	40.8	8.9	9.5	-17.4	-128.6
	12	-116.2	67.8	160.6	95.4	64.5	-89.8						
12	0	-298.9	692.9	-23.4	141.8	-34.8	-1.8	39	-20.1	-52.6	35.9	-727.4	125.4
	12	-66.8	65.4	-27	191.8	128.5	324.1						
13	0	556.5	-99.4	-35.8	-32.1	14.1	50	9906.7	-6.7	17.9	-7.6	4.8	-100.6
	12	31.3	-64.5	-26.2	21	-96.9	382.8						
14	0	-1113.1	228.6	-1997	47	-32.9	-70.2	39.9	49.1	-7.1	-139.2	-26.5	-1969.7
	12	-338.6	-179.4	-210.4	789.1	-261.5	84.6						
15	0	96.8	293.3	1752.2	-2475.4	-2346.9	23.8	228.7	249	1092.4	-31.6	-97.6	-299.9
	12	317.4	-174.1	-283.2	-81.7	-1.1	348.9						
16	0	-59.7	-678.9	-76.1	3647.7	-48.3	122.3	279.8	145.3	43.2	-199.8	-105.3	236.4
	12	-30.5	3.6	94.2	297.5	254.4	-214.9						
17	0	799.7	-47	44.2	-5198.3	0.6	41.6	272.8	96.1	22.9	92.5	-85.7	-41.2
	12	169.1	92.9	50.8	396.2	115.2	-255.1						
18	0	2201.1	-3419.8	-122.7	97.4	149	54.8	451.2	103.3	173.2	28.7	-288.7	-894.6
	12	731.5	473.5	1197.8	-221.3	147.5	-888.3						
19	0	4885.4	1293.3	-251.9	42.2	-190.6	156.2	-82.1	-67.6	-124.8	89.8	22.5	14.8
	12	-36.8	324.6	11.3	-294.5	-106.9	-861.7						
20	0	114.1	-83.4	-188.2	39	-29	12.4	84.3	-6938.1	100.1	-39.7	-24.9	118.6
	12	-135.2	290.1	-49.4	77.2	57	62.6						
21	0	301.3	50.5	1982.5	52.5	-48.2	3044.3	311.2	-236.4	-1166.1	-123.1	27.9	-329.4
	12	-549.9	-386.1	-174.4	29.8	175.6	93.2						
22	0	-416.5	82.2	5894.4	15.2	-82	45.6	-9	-11.1	366.6	45	159.8	129.4
	12	54.9	-51.7	0.5	-205.5	-59.3	118.2						
23	0	-56.4	40.4	69.2	18.2	-2.8	-20.3	-35.2	-38.4	38.6	-4.7	-38.5	12
	12	52.9	50.4	-8493.7	-301.1	-11.2	113.2						
24	0	-42.6	-11.2	1022.6	131.4	1803.7	-23.1	-120.3	-573.7	87.2	-96.5	-82.4	160.3
	12	92.7	123.6	135.6	-224.5	-6.6	87.8						
25	0	994.6	-254.4	-90.9	-188	5.6	35.5	72.3	-1529.1	-68.9	-9	-66.7	-418.8
	12	-18.3	128.3	226.2	-57.8	-64.9	-15.9						
26	0	-123.7	-1230.2	-93.7	-53.2	78.7	3102.8	140.3	178.1	243.1	52.2	-48.1	18.5
	12	241.9	24.7	-141.7	-479.2	-83.3	-296.9						
27	0	242.2	1432.6	3984.8	401.6	42.6	-161.6	-213.5	-376.6	-929.2	-7	39.3	-10.8
	12	94.7	-284.4	-35.5	230	-311.3	191.7						
28	0	101.1	505.1	-32.5	-9.4	12.1	-116.8	12.4	18.4	19.2	17.5	-6970.2	-22.3
	12	77.2	-40.5	70.2	-147.6	22.7	19.8						
29	0	-9.4	-222.5	58	-38.4	-13.5	40.8	-135.3	97.2	3.8	3614.2	-50.4	188.5
	12	76.8	22.6	-437.6	-45.3	205.7	201.8						
30	0	13	92.6	2321.8	44.1	741.9	288.2	181.2	2661.7	107.9	-81.9	-156.7	129.3
	12	-80	1747.5	-175.4	-55	185.3	319.2						
31	0	41.9	244.1	-1176.7	-22.9	1.8	23.3	-58.3	-26.6	-58.5	5.7	7.9	11.8
	12	-63.3	-57.4	-201	93.4	5320.6	-165						
32	0	23.2	120.5	-2694.5	-3010.1	-220.7	14.1	-123.5	214.8	-434.9	-384.6	-115.9	193.3
	12	384.5	-143.3	575.9	147.3	206.9	311.1						
33	0	311.3	3020.6	2055	-59.5	-154.8	-115.6	145	-77.4	124.8	-42.4	50.7	-263.4
	12	-212.4	225.1	379.8	-91.7	530.4	253.8						
34	0	14.5	-79.5	-70.8	100.3	-19.5	-69.6	-67.3	49.2	-17.6	10472.6	-61.3	44.8
	12	-4	-39	-5.9	98.2	98.3	-189.3						
35	0	94.4	-70.5	10566.8	93.5	-91.1	-63.1	30.9	-25.2	-35.5	44.7	-64.5	-19.8
	12	-94.2	-44.2	19.1	2.1	-85.9	-55.3						
36	0	83.5	-6.3	35.7	57.3	4855.3	6.3	71.9	-20.9	295.5	24.2	35.3	80.9
	12	-98.8	-372.7	77.3	823.2	123.9	87.7						
37	0	4092.9	-50.8	2702.3	328.8	234.2	-114.6	215.7	102.2	364.5	-50.9	-120.1	-43.3
	12	477.9	121.2	156.4	-49.1	131.8	465.4						
38	0	-3535.8	-28.6	147	112.2	-22.8	-139.2	-123.6	62.1	118	-97.7	67.2	-111

	12	59.3	50.7	42.8	-12.2	-112.6	30						
39	0	-437.2	-60.5	-52.7	-53.2	-0.4	-3.4	42.7	3.5	-20	-63.6	-48.6	-5033
	12	59.1	40.2	95.3	-175.1	123.8	-150.2						
40	0	-462.6	527.5	-132.2	-162.1	2464.7	-18	-75.4	92.5	-1.5	247.1	33	-184
	12	54.6	48	24.4	270	-202.7	91.1						
41	0	-2219.2	-43.3	613.9	557.1	-813	19.8	436.5	71.9	-245.2	63.9	-18.7	-2.5
	12	-121	74.6	40.3	-140.1	-226.8	516						
42	0	873.6	-97	118.1	-116.2	-64.3	-2644.1	-1.3	68.5	148.7	55.6	-64.5	5.2
	12	130.7	-19.5	201.7	-100.7	226.7	-121.8						
43	0	-208.2	1098.3	-4341.9	-24.3	-22.4	-125.7	-21.5	29.4	2.6	56.9	14.7	-123.1
	12	-8.9	-272.2	58	-258.9	163.8	39.3						
44	0	-69.4	95.8	96.9	-972.2	-46.8	46.1	-36.5	-24.9	-57.9	-52.5	47.8	-24.1
	12	-858.9	-1404.4	1960.9	455.9	164.4	322						
45	0	-329.2	-311.4	138.7	-133.3	-0.5	-39.6	-3035.6	78.7	-53.8	-3.2	34.8	406.6
	12	301.2	-276.2	464.6	-723.4	-14.4	-871.1						
46	0	35.9	-102.1	-59.5	-10.8	33.6	82.2	-45.4	33.7	11288.3	-4.8	-9	-54.3
	12	-18.8	141.6	61.5	124	131.5	15.8						
47	0	-1144.2	-1701.2	1544.7	33.9	55.9	-17.7	-60.5	-0.8	85.5	65.5	46	35
	12	86.3	-6.4	167.8	-29.8	136	224.8						
48	0	276.8	107.9	-43.9	-86.4	9.4	5262.4	-64.3	63.5	49.5	124.8	12.2	50.8
	12	163	-32.9	-26.2	104.1	73.1	234.2						
49	0	602.7	-83	-640.8	20.5	92.7	139.3	-17	-23.5	3420.7	28.7	-75.1	-161.2
	12	531.7	121.5	370.6	-7.1	155.3	230.2						
50	0	-104.3	1276.8	114.5	-57.6	-31.7	44.7	32.8	69.5	2511.8	-25.5	98	-46.6
	12	-2025.3	85	-27	101.7	-26.6	-199.8						
51	0	18.3	2154.4	130.6	3657.6	-162.5	-285.2	155.5	62.1	72.6	-129.9	-5	-741.7
	12	873.3	-1034	-666.2	-931.2	244.3	98.7						
52	0	1774	107.9	-9.7	2169.4	138.3	-66.6	226.3	-50.3	80.1	-36.1	29.5	-262.5
	12	380.3	732.3	695.1	-389.1	503	114.4						
53	0	-358.9	1901.5	101.1	46.6	-44.7	19.2	-214.3	-56.7	-2.9	-3	28.9	-6.6
	12	195.8	-217.7	-26.2	-247.3	143.5	-26.2						
54	0	-104.2	4620.4	-23.2	106.8	-272.4	96.5	-45.3	-69.6	-23.1	92.8	-51	-20
	12	110.5	68	-116	-233.5	-267.2	-75.7						
55	0	1554.8	31	52.7	-85	-115.4	40.7	-2.5	218.4	-85	-13.2	-47.6	-23.6
	12	110.3	6	50.8	-68.4	-45.2	52.8						
56	0	211.7	-103	-145.9	-1786.1	-56.9	-18.9	-6.4	33	180	5.7	0.8	5.2
	12	91.9	262.9	131.5	-178.6	375	15.5						
57	0	6224.4	-539.4	58.5	-170.7	11.2	-10	36.1	126.2	-31.7	12.9	20.1	59.7
	12	-16.4	21.4	-1.3	290.2	190.3	-19.4						
58	0	-12352.6	-28.3	-23.7	37.8	111.8	86.5	-74.8	20.9	27.1	1.6	-4.4	-17.2
	12	77.4	-95.8	-83.2	-178.6	-28	-234.8						
59	0	19.1	80.5	378.6	-63.1	-244.4	26.4	-47.8	78.3	6435	15.9	-110.2	29.3
	12	-135.6	76.5	-204.5	221.6	-5.8	-48.3						
60	0	47.5	3001.3	-224	-155.5	286.7	214.5	298.5	55.4	-233.9	-3.7	39	-29.6
	12	652.6	56.8	516.5	-136.1	386.7	-711.5						
61	0	320	125.2	37.7	78.9	1175.8	3.9	44	36.6	-19.1	41.7	46.8	-393.9
	12	-164.2	-87.3	206	-834.3	2428.9	106						
62	0	158.1	-21.2	96.2	-5.5	18.5	1433.9	31.9	35.7	64.1	-11.6	16.6	-32.1
	12	75.9	135.5	-135.7	660.9	1394.9	-411.7						
63	0	-72.1	36.8	199.3	2.5	-143.4	15	-36.5	2.1	7.6	4.5	-2.1	-1.1
	12	2.5	9.5	-37.6	-40.2	19.8	-5265.2						

Table C. 8 Shape codebook 1 for MODE_VQ == 16_16, BLEN_TYPE == LONG

VN	FN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-162	116.6	42	-40.8	1.2	9446.7	-27	-15.7	19.1	-16.2	-178.7	44
	12	-140.9	5	-61.5	-240.1	3.3	-40.1						
1	0	-466.3	-1096.6	-25.7	-122.2	22.7	-6704	169.2	86.8	-12.1	-44.7	-102.3	270.4
	12	-211.2	33.8	-9.1	213.8	348.5	298.9						
2	0	-2638.7	-1235.1	-372.4	-103.8	-298.9	-30.2	120.6	-73.5	-9.9	100.9	-65.2	-120.3
	12	61.5	251.5	393.4	-127	-240.9	-848.2						
3	0	679.3	-102.5	99.6	2.1	-22.5	-51.2	57.2	2367.9	-59.1	213.1	114	-242.4
	12	187.1	82.4	-287	278.7	-308.7	147.3						
4	0	-3027.6	89.6	2213.5	-251.6	37.7	6.3	-22.7	5.2	-127.4	-96.1	-38.9	76.4
	12	14.6	99.2	-100.1	547.4	533.8	476.7						
5	0	-35.2	-37.7	-43.6	2.6	10517.9	11.3	0.1	49.1	-104.9	-14.5	6.6	-6.3
	12	370.6	169.9	34.1	45.9	-118.6	-64.4						
6	0	165.6	-575.5	-1785.5	-62.2	-18.8	8.4	88.2	55.4	-49.9	3	-9.8	30.7
	12	19.1	11.2	-107.9	13.5	43.7	-36.6						
7	0	-113.6	-1027	-65.4	31.4	-87.6	-1799.8	147.7	-69.4	151	-33.4	-17.1	60.1
	12	-149.1	-212.4	179	342.5	2405.3	-171.6						
8	0	-480.1	151.6	75.1	-59.7	75	-0.7	1875.1	76.9	127.6	-64.4	0.4	-21.1
	12	-0.4	-149.3	-149.9	137.6	-13.7	253.8						
9	0	1789.9	-88.1	28.3	-23.4	-63.3	2381.5	-51.7	39.3	11.3	78.2	75.3	-213.4
	12	56.3	-92.7	-122.9	-25.3	-38.9	67.2						
10	0	-8097.8	3	-73	-60.3	-78.1	85.5	120	15.6	-30.1	-8.3	-20.4	16.2
	12	56.1	32.2	59.9	-6.8	53.1	199.9						
11	0	93.8	-49.2	-153.7	60.6	-12.3	-119	4397.9	-3.6	-15.1	110.7	-50.1	-529.7
	12	-456.1	-14.5	122.7	-68.2	233.2	233.6						
12	0	1126.9	-41.6	-191.7	91.4	2281.2	42.3	-4.4	70.8	35	-7.6	10.3	-21.9
	12	131.3	48.9	-55.8	656.5	-258.7	-45.9						
13	0	-22.8	-823.1	-69.6	59.5	-7081.1	57.4	45.9	22.6	31.1	-11.1	-57.9	192.1
	12	371.1	390.8	-167.7	436.2	-104	155.2						
14	0	-110.8	-35.3	-140.2	220.9	-3413.3	-56.8	44.4	-37.5	215.9	67.2	63.5	-63.3
	12	532	-408.8	227.2	598.9	-125.4	296.4						
15	0	369.1	94.4	1990.9	-10.4	5069.6	11.7	-301.1	111.5	-502.4	-115.2	-94.5	-457.7
	12	-89.2	-88.4	-198.4	152.9	250.4	-123.5						
16	0	214.3	-4446.5	2490.2	-6.1	59.3	-98.2	-185.8	266.2	151.7	20.3	-877.8	-205.9
	12	-117.2	-478.7	-502.2	-346.1	-488.3	-387.4						
17	0	166.8	1382.1	-114.7	37.6	96.6	10.4	-3063.4	-41.2	-29	100.9	-59.1	230.3
	12	-750.3	439.5	-961.5	314.2	160.8	-26.1						
18	0	-204	-82.4	318.8	62	-3.2	-34.5	-19.8	-9755.1	1	-12	4.9	5
	12	141.3	-194.5	-12.5	-69.6	165.8	-403.3						
19	0	2278.9	3722.6	15.3	-380.5	-59.4	228.4	-490.2	-287.4	57.7	-11.8	-71.1	-146.4

	12	-204.5	70.6	48.5	96.1	-291.2	-224.7						
20	0	49.6	27.4	-4	-2686.6	-106.9	48.5	46.3	-149.9	-138.1	-13.6	15.4	-27.7
	12	161.9	19.9	255.7	-98.1	-89	-16384						
21	0	-239.5	56.4	-1025	57.7	-73.1	26.8	-76.5	-78.3	-106.4	-19.5	19.1	19.1
	12	10.3	17.5	-82.8	-31.9	-788.1	147.7						
22	0	-1710.6	-9.7	158.9	-637.1	48.9	195.2	60	170.3	694.9	53.9	-34.8	70.9
	12	677.2	108	-56.2	-263.3	576.6	157.3						
23	0	-297.8	3043.7	-28.9	84.5	78.3	-52.2	85.3	33.4	104	112.9	89.8	255.9
	12	-134.7	11.1	-132	326.5	-469.5	37.3						
24	0	345.2	52.4	80.1	214	-24.3	-37.9	30.6	-20.3	2534.6	8.4	52.7	78.9
	12	104.7	-1116.9	-314.7	401	-66.9	355.7						
25	0	252	87.6	70.3	97.2	-140.8	96.1	0.5	20.6	-4748.7	51	29.3	-69.2
	12	128	-257.1	-101	-79.7	-204.1	7						
26	0	43.2	317.6	133	3.8	58.3	33.9	11.1	41.2	-18.6	15.7	10760.2	15.2
	12	40.9	17.9	281.6	-324.3	299.5	-12.1						
27	0	-59.1	5.2	-1106.8	-2009.8	2200.6	636.4	-44	-177	1640.7	-7.7	-116.7	-6.5
	12	-198	15.1	-505.4	-1211.7	-345.7	282.6						
28	0	993.1	375.1	391.4	122.6	-165.7	-67.7	-35.9	-93.7	-41.2	37.2	-21.3	-94.8
	12	62	-71.7	37	-144.3	2.8	-243.8						
29	0	-4412.4	29.5	-81	-6.6	-47.2	82	7.6	44.9	-136.4	19.3	16.1	2.4
	12	93.8	-22.5	-170.5	-32.9	11.1	-93						
30	0	-95.7	159	-2868.3	-3.2	1565.5	-120.4	717.1	101.2	-69.6	-137.2	69.6	-29.5
	12	-68.9	555.6	155.9	-222.8	-228.9	27.9						
31	0	-402.3	-131.4	37.9	-4693.7	-60.9	-115	-87.2	121.9	17.4	-77.7	211.6	-190.4
	12	61.2	42.7	-100.6	149.8	-175.7	-142.6						
32	0	-50.4	-124.7	-4023.6	472.4	-176.3	7.9	-40.7	-64.4	26.1	61.7	-31.9	135.5
	12	40.2	-221.9	-9.5	23	438.6	-137.3						
33	0	-354.7	54.6	2542.5	-14.5	89.7	-2970.1	644.9	-62.4	773.6	19.3	78.2	-443.2
	12	34.6	-181.1	478.6	-438.9	45.2	-173.7						
34	0	-692	-2255.2	129.4	66.3	40.5	225.9	61.1	26	42.5	58.1	-37.1	-50.7
	12	-317.9	618.4	-119.9	65	59.2	76.3						
35	0	-1367.1	6077.1	147.6	85.9	98.9	-102.4	228.4	-25.9	26.3	11.3	-28.9	53
	12	259.5	48.6	751	-187.3	50.7	-178.3						
36	0	-1186	944.4	-363.1	-3250.1	-177.2	409.8	882.3	-45.3	261	-29.9	-280.5	356.5
	12	-589.1	622.3	689.8	-1077.6	254.7	307.1						
37	0	2027.9	-1932.5	-155.1	9.9	-171.9	-221.4	87.7	-138.6	36	-143.8	-0.2	617.4
	12	-11.5	-1069.6	-148.5	-151.3	12	153.9						
38	0	-108.2	-4535.5	-272.8	195	-534	-15.2	63.7	-246.6	-9.9	15.3	20.4	-11.6
	12	13.8	-94.7	-39.4	344	-24.9	-17.6						
39	0	908.5	-1364.5	18.6	-1728.8	16.7	-258.2	8.4	471.7	-551.3	105.8	-229.8	-151.9
	12	-1835.5	-950	262.4	-1181.2	-370.5	-41						
40	0	-168.9	208.2	-25.1	-560	-88.5	-162.7	-119.9	33.8	7.9	43.4	133.9	-206.7
	12	4363.8	-99.8	42	-300.7	-87	-203.5						
41	0	138.5	110.2	115.8	42.7	-1323.2	-6.9	135.1	-23	-32.9	-34	-26.9	-346.8
	12	-154.3	3866.9	247.8	46.6	350.8	413.5						
42	0	286.8	127.6	-147.9	309.9	374	-2382.7	45.7	-4.6	-211.7	5.5	-79.5	-213.9
	12	-313.9	661.5	-569.5	199.2	-366.6	281.6						
43	0	116.7	-88.8	239.8	99.2	-3.1	-65.8	-10.6	45	-72.2	-5981.3	-92.1	22.3
	12	100.3	199.3	-39.4	281.5	-45.2	-45.8						
44	0	-159.5	-22.5	2.2	-33.2	75.3	13.3	94.6	-87.8	-7.5	-30.7	37.8	10448.6
	12	-51.8	14.2	247.7	-35	70.7	-51.7						
45	0	-590	34.2	38.6	1539.8	81.7	-90.6	-9.9	140.2	152.6	-38.4	2.5	189
	12	58.5	139.6	2978.1	-282	-59.5	-523.8						
46	0	-3.6	-303.2	38.9	-290.5	8.3	-3974.8	-244.8	-1.3	-153	93.2	103.2	52
	12	128.3	-241.5	-529.8	-16.5	-91.1	100.9						
47	0	410.6	675.9	-63.8	55.1	-12.1	-5987.4	-153.2	21.9	237.8	154.1	30.2	61.9
	12	164.5	-25.9	108.8	-152.5	56.8	323.7						
48	0	26.3	-490.5	895.5	-8.8	55.5	322	205.7	-2850.1	-270.3	39.6	112	-214
	12	212.5	212.3	-195.2	-172.5	-920	1167.5						
49	0	1032	-149.1	-69.7	-79	-9.1	73.6	-7326.3	57.5	44.1	-21	-38.9	-3.4
	12	188.4	-87.6	-30.3	-55.1	-94.2	-278.5						
50	0	-2628.6	-1.8	164.7	1.5	3693.6	-52.8	190	30.2	-7.9	73.7	-140.4	-28.1
	12	629.8	343.2	-137.7	-1011.7	262.5	92.3						
51	0	-1436.9	-529.7	-218.1	256.6	69.8	133.3	-3347	69	12.4	228.5	-284.8	-924.8
	12	108.5	-189.8	92.2	237	22.8	3.3						
52	0	-311.5	-98.7	-871.5	134.7	5380.1	-30.7	110.3	-27.1	-41	47.9	-58	92.7
	12	116.4	177.2	-91.5	-3.3	-164.4	-170.4						
53	0	100	293.6	-50	-4.8	-82	-12.4	-116.6	-4744.6	2.7	25.9	-142.8	-183.7
	12	238.8	148.1	60	-252.8	47.5	-183.2						
54	0	101.1	1885.9	-122.8	-198.3	-2978	-5.4	65	131.1	-78.7	19.3	3.4	99.8
	12	-38.9	263.8	-334.4	-142	-46.7	-570						
55	0	-5.4	-100.5	-84.3	6919.4	-50.2	5	72.9	-0.7	79.4	42.1	-14.1	-228.6
	12	-146.3	1.7	-69.2	185.5	-22.5	-176.2						
56	0	229.6	11302.6	-12.1	287.9	280.7	-41.9	-157.5	23.3	34.9	49.2	104.2	154.5
	12	140.5	-100.6	-149.4	-161.9	42.1	0						
57	0	37.8	-1457.8	-110.3	63.2	-53.8	73.7	-337.1	-45.9	30.3	-7.6	17.7	-8.9
	12	-25.5	4.3	-219.1	547.1	-68.6	-95.5						
58	0	7.9	-217	25.8	-39.8	-21.7	93.3	25.3	-6.8	66	-10.2	-3130.6	-54.4
	12	88.8	8.6	21.2	-213.3	238.1	66.7						
59	0	188.9	-213.5	124.5	31.7	-35.2	-35.2	-29.8	-5.6	-14.4	-12.6	30.7	-3.8
	12	32.5	59.3	87.3	9468.9	-15.9	-5207.4						
60	0	161.8	41.2	6398.1	8.8	57.4	56.1	81.1	17.8	-57	-67.1	-6.2	2
	12	12.5	46.1	101.2	301.1	450.6	-54.8						
61	0	9.1	3.1	-39.6	-364.5	10.8	222	-322	99.1	182.7	-2452.9	52.1	129.5
	12	-215.9	-270.3	-1367.4	-18	317.2	337.8						
62	0	263.4	564.2	-228	-36	-12.7	37.8	55.5	-37.8	22.3	42.2	24.4	-35.8
	12	-1081.1	92.8	-95.8	213.8	369.3	409.4						
63	0	-56.5	7.4	54.2	-30.9	247.6	-14.1	-23.9	14.4	-39.1	16.2	1	5.5
	12	-14.8	53.5	-8.7	3.1	-83.9	7513.3						

Table C. 9 Shape codebook 0 for MODE_VQ == 16_16, BLEN_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	507.2	-1513.4	-4295.7	-576.3	-294.5	-285.3	18.5	510.5	-335.9	328.6	-182.5	1860.4

	12	54.9	1620.4	-466.8	-675	114.9	-102.7						
1	0	-509.4	-2626.3	396.9	3283.1	-398.6	218.4	-780.8	-31.5	94.9	-76.1	1472.3	-394.8
	12	437.7	877.5	-327.1	330	-130.3	-388.8						
2	0	355.6	415.5	904.1	-1615.5	269.1	398.5	413.7	-305.1	-3895.8	-152.1	473.7	38.2
	12	-492.6	197.7	396.4	109.4	-205.2	-515.8						
3	0	488.1	-113.7	829.5	-800.8	-354.5	-567.5	5808.8	103.1	440.8	-544.1	-708.6	379.5
	12	487.5	-424.3	-298.1	-381.8	-428	397.5						
4	0	1402.3	-403.4	1026.6	-441.4	426.1	-1105.3	-223.4	295.5	700.2	-1094.2	241.7	-988.5
	12	335.9	174.1	704.7	-466.9	-362.7	-164.4						
5	0	-1860.6	-308.8	1607.8	-179.7	76.1	1112.1	-525.2	-80.5	154.8	-3528.7	-266.2	-523
	12	242.4	421.4	208.5	870.2	598.3	117.8						
6	0	-925.8	896.8	-569.4	-430.8	353.1	-26.4	1109.2	-239.6	-70.9	1045.9	754.4	-1397.3
	12	165.7	-582.6	708.5	-1548.7	2583.3	-152.9						
7	0	245.7	242.5	-463.6	-67.3	-134.2	2054	-284.4	2805	-210	-5687.9	-728.6	496.6
	12	-731.9	575.6	79.9	274.9	650.3	207						
8	0	-21.2	908.4	-566.8	1927.8	766.1	-666.3	-724.1	-147.6	-549.8	-533.4	336.5	383.9
	12	279.1	-330.3	32	246.3	-446.8	1739.3						
9	0	435	105.7	1621.5	709.9	-229.6	586.3	-456.7	-516.8	1030.9	316.4	-3537.8	70.5
	12	-292.3	173.5	351.1	-557.1	-385.2	1463.4						
10	0	-3071.6	-825.1	-99.1	1039.9	1094.2	132	639.2	-485	92.9	1128.5	1113	-839.6
	12	-599	-795.9	446.6	-621.6	-70.2	221						
11	0	403.7	2295.6	360.8	4769.9	224.2	-30.1	717.2	-38.8	1212.3	-57.5	-156.1	-202.9
	12	-4980.1	-1456.9	-150.9	126.8	351.3	-761.8						
12	0	-2228.1	666.1	232.6	-2484.9	-154.9	-131	532.1	-191.3	-178	-1060.9	-87.9	-809.3
	12	192	-923.8	987.6	-630.8	-1099.6	-811.3						
13	0	503.2	832.3	980.9	-4762.9	413.6	-697.8	-1516.1	201	-583.1	-189.5	513.3	-190.4
	12	781.5	-36.4	-455.2	-74.1	-128.7	42.4						
14	0	-255.4	-731.4	1010.3	1052.4	-4325.9	-863.7	1389.3	757.9	96.3	673	-1320.6	101.3
	12	-365.4	-785.5	-289.7	247.3	364.2	-454.2						
15	0	-1141.1	-681.4	960	743.5	-125.5	-15.4	3296.8	-290.4	-791.3	43.2	-226.7	331.9
	12	-554.3	233.8	758.7	-275.6	690.9	718.8						
16	0	-5704.4	2401.4	158.1	-752.6	-1016.3	1410.3	-177.4	-1506.6	58.3	-418.6	-3211	-659.7
	12	259.1	362	-1042.8	-482.9	-122.9	-197.6						
17	0	-1683.2	-71	1231.6	-1379	185	-1912.3	-225.6	-505.5	-708.8	123.2	1436.8	-488.1
	12	549.1	-392.8	672.3	390.9	-256.1	294.2						
18	0	-901.6	2564.7	1230	-705.7	301.9	-587.8	-186.8	-579.1	-1249.8	-269.5	-1970.1	-567.3
	12	47.7	-216.8	93	353.8	-114.1	508.9						
19	0	64.7	-328.3	3822.5	-790.9	136.7	-1221.3	-518.9	-715.7	-379.6	-826.4	890.1	290.2
	12	-665.5	-959.6	-808.4	316.5	332.4	2858.9						
20	0	1419.5	-5524.9	-23	119.9	198	-113.6	250.1	26.7	-516.3	654.4	147.6	-449.1
	12	293.2	601.8	174.3	-170.1	-69.4	71.9						
21	0	-780.5	179.4	2515.2	951.9	-506	-2612.9	-363.1	729.6	255.2	521.9	7.2	-316.6
	12	-303.2	249.6	-467.1	1289	-179	755.2						
22	0	-447.6	-195.5	583.3	256.3	-3050.1	480.9	-299.6	-1081.2	-1347.5	979.6	1858.4	-315.4
	12	-1009.1	615.2	1336.8	-74.8	-172.6	889.7						
23	0	708.9	-974.2	284.1	3577.9	-848.4	210.5	316.9	-976.1	-1480.6	166.9	1089.4	819.5
	12	152.7	-384.2	82.5	-774.8	8.5	493.2						
24	0	-1041.9	-508.2	-184.7	-1111	-534.7	-761.2	-555.1	7.8	1892	297.4	-108.1	-380.3
	12	-468.6	241	555.8	684.4	-148.6	-546.8						
25	0	-697.2	-484.1	-2023.8	-528.8	2761.4	480.7	2082.9	1216	-807.9	-566.9	978.3	-1032.1
	12	-480.7	1496.9	1188.6	-271.4	327	615.5						
26	0	-457.7	-1597.9	665.3	-618.4	1494.7	157	-719.1	475.6	-460.5	735.4	149.7	-352.4
	12	1402	-2084.3	-536.6	817.4	-358.5	-341.6						
27	0	6205.9	-250.2	1673.4	-380.6	1365.2	-166.6	-558.4	84.5	594.3	872.1	-197.3	33
	12	-455	24.3	613.3	589.3	128.8	-630.9						
28	0	-1760.6	-1898.6	-1638.7	-1069	1309.6	-90.1	-1383.9	-1749.7	-286	-507.8	-368	1194.7
	12	31	-816.1	138.2	-470.6	134.7	-479.4						
29	0	1447.7	-895	-1032.6	1577.1	2937.2	-134.5	592.6	415.2	-1087	138.8	-875.2	93.2
	12	463.8	633.1	259.1	140.3	-14.8	568						
30	0	-471.4	387.9	573.1	-474.7	1	2387.2	896.6	871.4	-300.9	-351	-1701.5	-1277.3
	12	1681.6	1366.3	-992.1	207.6	735.9	910.8						
31	0	288.9	1274.2	1515.8	574	-515.6	563.4	-588	-1288.6	-473.4	743.2	129.2	-483.9
	12	2606.6	-454.9	-101.1	-339.7	511.7	79						
32	0	-439.9	-1253.6	635.6	220.4	-887.5	-600.3	-422.8	82.6	384.1	-126.9	-518.3	383.4
	12	574.1	936.7	-898.1	-2089	442.3	167.6						
33	0	-1.1	773.5	-245.7	-1134	-2632.5	-2054	-1280.3	419.6	-697.3	186.9	588.1	-1139.9
	12	-541.4	105.9	-699.5	-906.3	-285.8	435.5						
34	0	736.3	3703.8	-833.9	-464.2	-823	532.7	-724.3	690.9	536.6	-253.3	-356.3	-535.7
	12	630.8	-766.9	-834.4	16.4	-875.9	967.9						
35	0	785.5	92.9	303.1	-2039.9	310.7	-708.2	-917.2	-17.5	-838.4	943	-1658	181.7
	12	30.3	1603.4	314.7	-117.2	-1020.1	371.7						
36	0	-803.9	1129.9	-10.8	1914.5	-234.2	3511	-116.3	-28.1	-483.3	661.4	604.9	-680.6
	12	-1511.9	450.9	-961	171.4	-1638.9	-1521.3						
37	0	-2674.3	2060	1176.3	-717.5	-411.3	-567.3	459	318.2	533.7	36.1	198.4	-456.2
	12	-804.9	656.7	527.3	150.6	502.2	1887						
38	0	561.5	-2493.4	2368.9	286.5	107.4	524.7	1428.5	-527	336.8	1007.7	-5.9	-860.1
	12	1035.7	-307.2	-1145.1	-192.1	-214	204.4						
39	0	-803.3	-475.9	1323.1	278.1	2060.5	-1301.8	-674	221.9	-2108.7	-207	-357	200.2
	12	41.7	-272.5	1690.5	-34.5	273.2	488.1						
40	0	-19.2	5079.3	-1672.9	1.6	2455.5	110.7	513.5	-53	-589.9	166.1	551.4	41
	12	-523.6	694.8	1037.2	-178.9	35.1	138.8						
41	0	677.2	257.1	514	293.2	334	-1094.6	131	4000.3	-261	-210.1	128.2	-383.5
	12	441.7	80.3	856.4	174.3	557.1	-430.1						
42	0	-72.3	355.1	1378	355.7	457.4	-326	-1208.4	-177.7	-790.7	119	990.4	-187.1
	12	861.8	3860.8	181.4	33.6	129.1	587.9						
43	0	223.6	589.3	1952.4	-603.6	-542.1	-69.7	699.9	811.9	-302.2	107.2	-973.9	716.9
	12	-144.3	-149	-640.7	-141.1	3264.3	-193						
44	0	-1778	-2609.1	-140.4	-248.4	-807.2	-113.4	2041.7	391.9	-110.1	-344.5	26.9	-2163.3
	12	932.1	980.5	314.9	-697.4	211.2	279.7						
45	0	88.8	-889.5	2397.8	-533.6	-14.5	878.2	-728.2	535	-686.8	535	840.1	-2787.4
	12	-617	820.5	-1297.5	222.6	90.8	-939						
46	0	-274.4	1177.1	-82.8	-753.6	163.6	-566.6	-102.8	-530.3	2043	-1931	779.5	-88.6
	12	-24.3	517.4	-66.4	-223	979.6	-611.9						
47	0	-44.9	30.2	-1940.5	-2419.9	-438.3	-1274.5	297.1	-305.5	627	-233.5	39.2	57.6
	12	216.8	-982.8	-2612.3	-81.1	26.6	134.3						
48	0	509	-420.5	-770	-71.9	-978.8	122.1	1634	916.4	-798.8	48.1	-249.4	-300.2
	12	61.8	-1343.1	2195.4	253.5	1328.2	-53						
49	0	2206.3	1202.3	1163.2	-1070.1	1715.4	1147.9	-309.1	-1004.5	560.5	-282.6	470.1	-949.9
	12	-293	127.6	-138	-890.7	304.6	-1074.3						

50	0	-1066.4	2144.8	-497	-2.6	-1856	-1793.5	356.6	-872.1	51	711.5	386.1	46.4
	12	607.4	-767.6	337.9	-363.3	-2031	-561.4						
51	0	-1892.5	-53.3	-130.4	348.5	1322.6	453	-172.7	772.8	2289.8	-33.8	-537.2	-730.2
	12	-784.4	-14.9	-656.3	-753.3	739.5	1485						
52	0	-30.8	331.9	-1351.6	200.8	1013.2	-1480.7	658.1	-757.3	-565.8	391	-1537.9	-1467.4
	12	289.8	-799.1	-408	-1256.3	-51.9	626.4						
53	0	-2726.9	4009.6	662.1	980.7	-1034.9	2889	-170.8	-1081.3	-1790.6	461.9	-1040.3	-763
	12	1634.6	-929.3	586.4	-140.3	632.4	-573.4						
54	0	-721.7	-1398.4	-339.8	183.8	-2054.6	330.6	-2112.2	234.3	-70	-532.9	1419.8	-527.1
	12	18	228.9	189.1	-75.4	-905.9	-121.8						
55	0	555.2	-1055.8	-623.1	-1279.2	468.3	1993.4	1269.3	-186.4	216.5	-1363.3	-1494	211.1
	12	-298.6	123.7	-808.5	-400.9	127.9	665.4						
56	0	2263.5	500.6	-801.6	-1160	501.5	-1042	-175.1	429	953.7	962.9	-564.6	-268.6
	12	-106.2	391.5	748.9	1691.9	1156.9	374.8						
57	0	1463.5	480.8	-122.4	709.7	-33.6	-4823.6	-205.3	26.7	325.5	-62.4	-449.7	-249.8
	12	325.3	13	90.3	-791.4	458.5	50.6						
58	0	-7572.2	-1701.4	1542.8	-248.3	-287.2	-384.2	853	1116.4	-295.3	-782.8	2170.8	290.7
	12	-1308.9	-180.8	756.1	-0.6	-290	-1765.8						
59	0	324	-21.1	161.5	-1685	1713.4	-1274.2	1366.8	267.9	255.5	1903	157.1	108
	12	360.3	552.2	850	81.2	175.2	1098.9						
60	0	608	-488.9	1430.7	1258	736.7	-197.7	752.4	547.4	1495.3	-80.6	-265.2	-256.7
	12	-363.1	-63.6	-327	-1003.7	-1621.6	-3593.1						
61	0	19.8	-3187.6	-1525.4	-468.4	-135	-166	190.1	278.9	629.7	-864.8	-2209.7	-673.9
	12	-455.6	348.7	201.7	1296.8	-169.7	727.1						
62	0	-342.8	241.5	-682.4	401.1	-647.5	416	-748.3	2465.9	555.4	-1049.9	-1115.6	624.6
	12	377.6	522.4	225.7	-536.5	-549.4	363.9						
63	0	4491.2	1240.2	33.4	119.7	-316.2	-782	-106.6	464.9	206.4	699.4	19.4	1721.2
	12	1766.8	408.3	-1008.5	465.4	486.9	452.1						

Table C. 10 Shape codebook 1 for MODE_VQ == 16_16, BLEN_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-3861.6	-1057	2473.3	301	92.4	-820.7	-897.3	-48	339.9	281.6	483.4	1904.4
	12	738.3	-809.4	-594.3	1325.4	-148.7	40.7						
1	0	-711.2	442.9	-4785	1116.9	-562.5	410.1	276.4	-435.7	520.9	70	835.4	-985.1
	12	-946.8	165.2	45.9	-222.3	912.8	-496.6						
2	0	-506.9	157.9	-179.4	-433.8	-561.5	297.2	-411.7	-381.5	-4064.5	298.3	-912.4	-58.4
	12	288.6	172.2	-377	191.4	942.3	-236.6						
3	0	931.2	509.6	-490.1	1136.2	-43.2	-367.1	1023.5	-4.2	481.6	86.8	314.2	1061.1
	12	403.8	-1353.5	710.3	25.6	1065.9	-789.8						
4	0	128.4	-672.4	651.2	402.6	-528.2	-615.2	-232.1	-3184.8	81.9	-438.8	-75.8	-539.9
	12	994.9	96.3	531.9	174.9	2501.4	-334.6						
5	0	1328.3	284	612.8	-512.8	-1130.2	-17.3	2788.3	116	-188.7	526.3	630.8	-610.3
	12	753	303.3	-10.5	620.7	392.7	-1353.4						
6	0	-811.2	589.2	2401.5	17.3	-105	539.8	300.9	-366.2	-505.8	-1573.3	-0.4	730
	12	-450.7	1050.1	233.1	-1029.6	26.1	252.2						
7	0	-2591.9	-304.3	-2785.5	-19.7	800.9	-99.6	941.5	-48.6	237.2	-711.7	1182.9	7.5
	12	-151.6	-367.8	-122.8	18.8	822.2	-1503.7						
8	0	-487.2	896.7	-282.5	546.4	-1934.8	1632	781.3	-179.8	479.9	2060.4	477.3	581.9
	12	1095.6	995.9	1040.3	-187.2	-61.7	-224.7						
9	0	1605.8	1190	435.4	-2171.3	-649.8	1125.6	-619.6	906.7	-60.5	-76	429.6	-222.7
	12	-712	-1237	318.1	1431.4	-359.4	366.3						
10	0	401.1	-1879.4	1567.6	454.6	400.4	-1278.4	2340.8	-557.8	-430.7	668.2	-203.9	-1.1
	12	-651.1	-1235.9	286.4	-688.1	-966.9	384.2						
11	0	-3308.6	779.8	-448.7	40.6	257.6	-246.4	439.3	-638.6	-400.8	470.6	369.3	90.7
	12	27.2	430.3	1038.4	250.9	-914.4	1190.9						
12	0	418.7	281.3	159.9	-2453.8	155.2	2121.1	-1242.4	596.9	344.9	311.7	1499	834.9
	12	648.6	567.6	-497.1	892	121.6	-1042.6						
13	0	-100.8	-3940.6	-797.6	-1937	1424.5	844.4	-583.4	508.6	1080.8	854.5	160.6	435
	12	-610.9	109.9	812	-514.6	824.3	-225.4						
14	0	282.1	116.2	-1814.4	-268.7	-42.2	-873.7	-2121.9	-728	438.9	-1117.3	-851.5	-429.1
	12	100.5	-201.1	-238.8	2141.6	707.9	-14						
15	0	763.3	-80.6	-896	-1505.2	222	326.7	1295.7	-387.1	-484.7	13.2	1229.8	-707.5
	12	161.4	8.8	-894.4	-1137	-2155	-711.1						
16	0	-2225.8	561.6	1071.1	1155	1170.7	-1018.1	-300.2	275.9	246.9	263.5	229.9	768.1
	12	465.5	-161	-259.7	205.7	212.2	-397.9						
17	0	531.6	2313.3	1287.4	-940.2	794.5	-1901.7	315	1.8	-468.8	-615.9	725.7	-59
	12	547.5	435.8	-508.3	-152.4	955.5	-80.3						
18	0	531.9	-1087	-1058.7	337	2172.3	216	-930.9	-388.3	793.2	1230.1	1231.5	-280.9
	12	-154.9	-1780.6	-861.4	-1250.5	136.1	226.2						
19	0	-685.4	-1671.7	286.9	-250.7	2862	82.8	925	-1014.1	-692.1	-440.4	-833.5	-420.5
	12	200.2	-47.1	906.2	896.1	-1509.2	288.6						
20	0	1081.8	-1830.4	599.5	-45.1	-196.8	334.1	-516.6	-708.5	993.9	112.7	153.4	-401.7
	12	-414.5	729.9	631.3	389.9	-143.3	2308						
21	0	1695.7	-1088.2	-3034.5	-547.8	178.8	109.9	201.9	-40.1	396.1	48.5	-226.1	-448.4
	12	-442.6	-281.5	5842.2	452.7	-317.7	-1167.9						
22	0	253	711.5	276.3	4785.7	885.6	-346	-870.5	558.1	246.3	-764.9	-493.4	-610.2
	12	-521.5	-15.5	-209.9	504.9	-525.7	-929.4						
23	0	-1604.2	1466.3	-2707.5	440.3	793.3	878.5	-635.7	348.1	-462.3	-161.3	-355.4	-389.3
	12	1295.7	1719.8	192.6	-355.7	-451	466.7						
24	0	-414.7	-383.6	550.8	609.9	-1079.7	-158.8	365.7	954	-1105.6	439	247.1	-1503.3
	12	-3186.1	428.5	45.1	-303.5	-11	793.7						
25	0	-745	1109	1035.8	-271.7	827.8	-951.7	1925.2	-129.2	-29.8	-207	1111.1	-92
	12	-46.4	-223.8	-1229.5	189.9	-522	222.4						
26	0	1384.4	-710.7	-251.8	77	-2333.8	121.1	-61.2	-172.4	-460.6	67.7	-2227.3	-1189.4
	12	-150.6	-185.1	290.1	216.2	568.3	-654.7						
27	0	76.4	3258.7	-96.6	-462.1	205.5	947.8	458.4	-1179.9	-130.7	323.1	-950.8	566.6
	12	171.2	1296.3	-124.1	-871.6	-945.6	-309.7						
28	0	-1567.8	-230.5	-1454.8	-2401.2	-948.6	-340.5	617.8	664.6	75.9	2133.2	148.8	77.3
	12	485.9	-682.6	294.1	158.7	-33.5	-142.8						
29	0	-829.7	-2526.2	-508.1	-1966.6	-34.1	-701.1	-163.6	923.5	-1399.4	1184	-1198.6	200.4
	12	-670.9	-767.4	-1271.5	-441.4	1367.5	290.3						
30	0	297.2	-510	-797.6	247.2	399.2	-347.2	770.3	-274.6	-843.2	-3973.6	1294	236.7
	12	-264.6	-715	691.7	554.6	-570	945.2						

31	0	-662.4	-1325.3	-1370.8	-487.8	-502.1	27.6	1812.3	77.9	359.7	-22	28.2	-450.6
	12	-19.6	2597.5	-65.5	-61.8	55.7	-874.3						
32	0	308.7	-469.6	541.7	1985.2	488.9	1977.1	-956.2	536.8	-341.6	465.2	-1617.3	-558.4
	12	243.6	416.3	27.3	70.5	1117.8	727.8						
33	0	-687.6	-821.3	-334	3265	173.8	-413.6	376.7	659.9	47.8	-711.7	1335.9	134.5
	12	2177.8	369	460.6	283.9	884.3	562						
34	0	-5648.4	244.9	602.5	-333.6	-1.4	-441.2	-737.7	1145.6	387.4	906.2	130.9	-215.5
	12	384.1	-271.5	152.3	-507.9	215.1	-1835.2						
35	0	-1703.9	10.4	-214.1	193.5	370.6	-3172.5	-374.7	-761.2	430.6	27.6	-1020	28.3
	12	-1053.8	508.7	40.4	92.9	-680.3	-604.5						
36	0	907	-1958.7	-1693.3	337.3	159.4	-1496.8	520.8	1887.6	218.7	-569.5	-259.7	-249.5
	12	1543	298.1	-978.1	882.7	-859.6	444.5						
37	0	1351.6	1541.4	-571.5	-317.3	-441	-1051.6	-577.6	-249.9	-617.6	265.9	-75.9	498.4
	12	-192.8	669.2	2317	-985.1	-339.2	0.1						
38	0	-37.3	902.6	-1334.1	94.4	139.1	-812.1	687.7	312.6	-99.8	-3168.9	-653.1	-658
	12	1016.3	150.6	-970.8	-154.4	-1338.6	-750						
39	0	-116.6	1151.2	759	-791.5	-80.1	322.8	-458.6	1646.2	332.4	170.3	-638.9	-868.5
	12	9.4	-167.2	474.5	-1053	31.9	461.6						
40	0	-27.8	-850.9	-1755.1	-1364.6	-4682	336.6	-490.4	-189.8	114.7	-21.9	-86.2	-576.9
	12	240	-510.1	2207.4	777.1	-121.7	636.9						
41	0	120	-1812.1	-995.6	2009.3	-1916.5	-1440.7	348.4	-906.1	-543.7	-3.1	-16.2	59.4
	12	-87.8	-265.2	420.6	-108.2	1216.5	-435.9						
42	0	301.7	-483.1	-438.8	-166.4	-796.9	482.1	-501.3	2208.1	-98.3	1228	1035.8	932.8
	12	-724.5	1523	-73.9	424.6	145.1	181.3						
43	0	478.5	385	-965.6	213.3	359.5	-159.5	-1776.7	-873.5	189.3	1969.4	-817.2	81.2
	12	-671.7	-636.1	527.6	-810.4	-896.7	315.2						
44	0	2472.7	-229.3	781.7	83.1	-885.3	-629.4	555.6	-449.4	-604.2	-1295.1	2900.9	397.2
	12	553	-208.7	-41.7	-452.7	267.5	732.5						
45	0	893.4	-2663.4	901.7	-511.9	-2680	37.1	-242.3	258.4	396.3	-540.3	727.1	1651.9
	12	-919.4	909.4	-634.8	279.4	-1926.6	-707.1						
46	0	2746.2	895.9	-1489.4	484.2	193.2	-1406.1	98.1	-1317.4	835.7	168.4	281.7	-262.7
	12	-762.5	239	-908.7	-1114.5	-101	641.6						
47	0	1553.1	975.6	1267	390.9	337.6	1431.3	142.6	-971.7	385.6	246.8	-27.8	-3179
	12	-256.2	-104.7	595.3	-178.6	-509.2	127.3						
48	0	1842.9	-1117	-207.4	-1094.9	504	-608.3	-1333.7	-625.1	-1048.2	-189.4	-102.5	-376.2
	12	15.8	359.5	-337.7	-290.9	64	-415.2						
49	0	477.9	157.1	2124.1	-2943.8	-484.2	146.4	835.4	-666.6	2349.5	24.6	400.7	403.2
	12	-237.5	442.3	-78.4	766.9	804.8	1345.2						
50	0	-220.2	-898.4	275.3	-1157	-359.6	614.9	1821.9	463.6	-69.6	198.8	330.1	2327
	12	538.6	638.5	-470.9	-324.1	402.3	-40.7						
51	0	-887.7	-194.7	4433.4	566.3	314.3	1589.7	-359.2	1275.6	105.4	-7.2	-113.6	642.4
	12	2182.7	-401.8	632	-723.8	-599.7	684.8						
52	0	-377.1	-748.9	-1231	-3103.7	64	1878.2	-184.3	-1011.5	-519.1	-1056.9	611	-292.5
	12	-600	440.2	789.6	-634.5	-30.6	152						
53	0	-635.4	467.9	139.9	131.3	177.5	-111.4	-38.7	-149.9	1256	2614.4	-733.6	902.8
	12	-431.9	597.3	-700.8	775.1	-462.2	445.6						
54	0	-319.9	167.1	1670	777.4	422.2	-576.1	-705.2	1918.9	-784.4	-494.8	-534.2	124.2
	12	745.8	93.6	379.8	391.3	653.1	25.1						
55	0	3173.5	-3917.2	1348.8	622.9	1374.5	-953	522.8	-1344.8	3.8	-898.1	-922.8	-99.3
	12	754.8	762.5	-113.2	-33	869.8	546.2						
56	0	-887.9	-2946.6	1154.5	1188.1	80.7	108.1	-1106.5	-352.2	20	298.8	-518.3	-188.1
	12	225.4	244	207.6	-816.4	-826.4	16.5						
57	0	-408.1	-197.4	1133.2	665.1	227.5	1312.3	545.8	761.4	2271.3	210.3	3547.4	-340.8
	12	-619.7	473.6	-158.3	-1240.8	23.6	2894.9						
58	0	-200.9	-174.8	604.6	3296.7	-319	604.6	10.9	-1740.1	-208.3	-142.2	-283.9	-689.6
	12	721.7	-1792.2	-803.1	1398.5	185.3	-1633.4						
59	0	-523.9	-136.9	338.3	-1111.9	2461	1553.1	-1059.8	-643.2	545.2	-63.2	591.9	719.7
	12	-262.7	-59.9	2738.8	22.4	-89.8	-209.8						
60	0	-338.9	-516.9	2049.2	-1480.5	201.1	229	-1191.2	-442.5	-190.4	-1221.9	1142.7	123.4
	12	-704	-419.3	-1818.2	124.6	393.8	384.6						
61	0	-380.2	84.3	-765.6	-652	-3144.1	1314.4	-25.9	164.3	140	-721.8	-499.3	181.4
	12	253	-891.1	-420.5	-1842.3	-99.7	1017.8						
62	0	249.7	2196.4	-777.7	863.3	-63.8	1596.6	888.2	47.9	-388.7	-619.5	204.9	890.9
	12	76.2	-1355.5	-1695	358.5	-297.3	252.9						
63	0	-1846.1	-1667.2	856	-1696.2	-527.6	690.4	54.2	161.8	1122	-468.1	-2858.9	225.9
	12	-162.7	-18.5	509.7	126.1	-335.3	-530.1						

Table C. 11 Shape codebook 0 for MODE_VQ == 16_16, BLEN_TYPE == MEDIUM

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-3641.6	-3365.2	395	129.7	293.5	247.6	-191.9	57.6	-560.6	269.1	136.6	-104.9
	12	163.7	-635.6	-296.7	241.9								
1	0	903.1	-78.2	-273.5	15	-4809.2	96.7	56.7	-122.7	177.2	-127.1	-129.3	-0.1
	12	-70.1	205.7	-472.4	-3.1								
2	0	-136.3	51.4	1986.7	-596.7	-2863.7	-410.4	-313.4	-367.6	-56.7	-159.8	-1069.9	595.7
	12	-1448.8	311.4	547.4	-69.3								
3	0	74.2	73.5	-326	109.4	63.3	474.6	43.9	93.4	-122.2	4435.2	-287.7	-185.5
	12	249	233.7	67.3	56.2								
4	0	-36.2	2422.6	1653	-205.8	987.7	25.7	134.5	-538.1	325.7	-561.5	-15.4	-131.4
	12	-549.9	-1.9	409.9	-2618.3								
5	0	244.3	-140.2	186.6	37.1	-140	-22.8	114	55.8	2453.8	146.9	650.3	-106.2
	12	-361	104.7	-627.6	-290.3								
6	0	6527.6	-149.3	-82.3	-123.2	12.3	-67.4	85.5	-80.1	-123.2	-85.7	33.7	22.3
	12	-108.4	-526.4	-153.5	9.9								
7	0	-64.8	877.8	728	376.7	-58	-2.6	-423.4	-108.1	120.4	262.2	-1971.9	28.9
	12	-98.9	-154.3	-1339.4	37.4								
8	0	-4257.9	20.7	88.9	-266.6	-222	168.2	16.1	170.7	-106.1	-91.5	88.2	-73.4
	12	52.7	101.1	-220.9	308.5								
9	0	-37.4	1096.4	-69.1	803	-3934.4	-335.2	155.5	100.2	656.4	496.7	12.4	42.6
	12	-73.1	-350.8	302.6	-101.8								
10	0	-39.2	-45	3978.6	1828.1	520.2	-335.9	-356.1	-239.3	-154.1	456	114.3	54.9
	12	-77.5	258	-32	-329.5								
11	0	86.9	2328.1	-233	-2525.4	15.4	-9.4	256	-112.5	89.9	3.3	-789.6	-70.2
	12	304.5	41.5	248.6	26.1								

12	0	-1448.6	-104.3	68.3	520.5	-78.3	91.8	-3200.6	19.9	10.1	-56.4	-167.8	-40
	12	570.8	-5.3	-241.9	24.7								
13	0	-46.7	-231.4	-18.1	212.6	788.3	-47.2	-4.1	100.5	-4.1	-177.9	-46	-1625.8
	12	57.8	1483.2	-407.3	615.6								
14	0	-258.1	-283.4	2109.4	192.2	1538.4	650.4	-227.7	840.7	202.5	-343.4	54.6	-698.4
	12	-852.9	-1753.6	253.3	-380								
15	0	513.6	155.5	717.9	151.9	269	1922.1	1908	-536.4	246.6	236.3	-129.2	-18.4
	12	282.5	154.7	-288.4	187								
16	0	-6.9	-177.5	-2968.2	419.9	-4.6	270.8	-645.5	116.9	-125.7	-307.6	115.9	130.5
	12	-82.4	717	323.8	-1464								
17	0	-2487.6	-1498.9	-61.9	-55.4	-159.9	148.1	-14	-1392.5	-68.8	-124.5	343	-307.9
	12	308.2	812.7	-318.7	-172.6								
18	0	-3158.8	407.8	1447.9	-32.8	2080.3	233.6	3.4	-417.9	-153.3	194.8	-1088.7	50.5
	12	-202	774.5	-234.5	-193.1								
19	0	336.6	16.4	212.4	4571.9	618.6	-27	-199.8	-312.3	-119.8	-26.8	-324.7	51
	12	-194.1	128.4	48.6	529.2								
20	0	11.2	-127.3	162.7	-33.8	50.2	-10.6	-4934.5	-51.4	-61.1	-51.7	-28.7	90.8
	12	-214.3	142.8	27.7	110.3								
21	0	342.5	-1457.8	-2330.8	-190.8	112.4	-238.4	-74.9	-2267	196.1	-216.6	569.1	-425.9
	12	161.1	-393.6	-137.5	-251.9								
22	0	-1990.4	-143.3	382.1	-81.1	567.9	90.3	121.2	2974.3	-360.6	-204.2	132.5	-158.1
	12	135.6	-30.4	375.1	-317.3								
23	0	-117.5	80.9	271.1	-53.5	155.7	2471	-440.9	37.8	278.7	155.9	1133.5	-364.9
	12	-1564.6	149.2	950.2	35.2								
24	0	257.1	229.1	-52.7	1709.1	1.9	-490.6	-167.8	2677.3	256.8	-92.2	-25.8	42.9
	12	179.2	-422.6	326.4	154.3								
25	0	1559.6	-801.4	952.6	91.3	-1349.6	-43.2	16.8	98.6	-490	-748.7	188.6	237.2
	12	-183.6	538.1	-57	1340								
26	0	-2604.7	-81.5	765.7	2066	-318.5	460.4	48.7	45.6	905.9	-185.1	248.6	-228.1
	12	-166.2	-208.8	975.9	-42.5								
27	0	-180.6	42.5	-109	1314.7	-999	-64.4	0.3	-125.5	-74.5	114.3	-401.8	-496.1
	12	-2024.2	186.6	-1969.5	229.5								
28	0	-267	536.9	-1204.6	1678.7	1123.2	-137.4	631.3	-219.1	159.3	12.6	42	160.8
	12	-366.1	95	60.1	-356.4								
29	0	-220.4	-2665.5	-119.4	-34.2	2291	137.1	-80.5	120.5	166	119.3	388.1	314.4
	12	656.9	-110.1	-959.1	-1575.3								
30	0	227.4	-522.9	58.3	-9.5	47.8	18.8	33.5	7999.5	10.3	43.4	-0.3	-85.7
	12	-124.6	73.9	35.7	0								
31	0	-1044.2	465.3	53.3	57	-977.3	-357.2	84.7	110.3	-128.4	-37.7	-75	-3528.7
	12	-212.2	-186.4	50.5	18.2								
32	0	123	1903.8	17.5	529.8	333.7	-1682.6	84.1	-63.2	-300.6	705.6	-73.7	-328.7
	12	136.7	-2712.6	-138.5	647.2								
33	0	-806.3	-45.2	-1807.7	530.4	-2668.7	812	-715.8	85	378	-435.4	221.8	-136.6
	12	163.8	143.3	113.5	-9.1								
34	0	1191.3	41.4	-67.7	-20.1	211.4	-8639.5	20	-306	121.2	-11.5	-761.7	89.1
	12	17.6	-9.6	229.1	-277.8								
35	0	-1632.7	-419.7	-2494.2	428	-47.9	-550.6	390.4	482.6	-244.8	51.1	-149.8	480.9
	12	60.2	129	-25.7	88.9								
36	0	55.8	11	-623.3	3033.4	557.6	2680.1	-459.9	-243.3	-69.2	281.3	-489.4	0.4
	12	-283.6	189	-193.1	27.9								
37	0	2383.7	-298	-52.7	-84	-37.8	3315.8	-75	-33.3	-8	-87	-74.4	187.5
	12	26.7	61.4	63.4	102.7								
38	0	-80	37.7	-76.7	-49.3	-8797.9	-1.7	64.4	50.1	-156.7	-45.9	-13.1	-37.5
	12	-186.7	193.4	153.1	209.3								
39	0	713.6	70.9	-58	17.8	-70.7	-76.9	-363	-192.6	56.2	-40.2	81.9	-191.2
	12	114.8	114.7	-294.8	147.1								
40	0	-195.2	-3765.9	-3421.8	-29.1	327.1	23.5	-62.3	44.9	99.3	-15.2	-275	152.6
	12	59.1	435.6	-191.9	-48								
41	0	724.4	-1506.2	24.4	-1134.4	-20	-1399.7	-165	96.3	-705.6	58	1165.2	157.3
	12	-502.9	-8.8	-1311.8	108								
42	0	-117.4	8	-1370.9	-90	-64.1	146	-303.7	-572.8	1957.9	-530.2	-654.9	165.8
	12	743.6	85.7	-306	-7.4								
43	0	-98.7	-350.6	345.7	-82.8	-193.9	-6698.4	66.5	125.6	108.8	58.2	574.3	94.3
	12	201.4	29.5	-349	6.5								
44	0	107.3	-571.1	-153.2	254.1	-414.6	1412.2	24.5	-974.7	-346.1	-83.1	-176.8	-131.4
	12	346.7	-2303.7	234.5	-833.5								
45	0	83.7	-988	-5865.7	16.5	-554.2	124.5	-312.1	326.5	132.6	498.9	-224.2	-128.7
	12	-101.7	-525.8	-38.1	167.5								
46	0	-1699.5	54.6	-120	-1900.2	-540.8	347.5	149.4	-192.8	162	-85	53.2	61.3
	12	-108.8	136.6	135.5	349.5								
47	0	-408.1	-1281.4	-139.2	171.7	29.5	21.8	-100.7	142.9	-66.1	-16.3	-141.3	-113.5
	12	3158.6	469.3	121.2	233.9								
48	0	-980.1	3431.7	257.8	16.5	128.9	164.2	909	-124.1	98.2	57.5	176.1	612.8
	12	-691.3	-190	-590.3	1153.4								
49	0	-625.5	298.3	1315.9	-2048	706.4	-348.1	-664	165.2	1968.7	237.6	-208.9	295.4
	12	-549.7	-315.3	1014.5	442.5								
50	0	544.9	-2441.7	3510.9	-10	-199.1	170.6	-17	-10	145.3	-52	109.4	-306.4
	12	393.3	-526.5	-2322.2	-146.9								
51	0	-331	-457.7	-52.6	69	133.7	4141.7	113.4	460.1	-234.9	-19.5	-87.3	-38.2
	12	430.5	-257.8	-531.1	68.9								
52	0	232.5	-18.8	44.6	-7690.7	298.9	-41.6	-370.2	-175.6	-89.1	-112.4	3.9	-37.6
	12	-135.8	79.9	223.2	-105.7								
53	0	1081.8	971.2	-438.3	502.6	-22.3	140.5	61.7	4.4	43.4	-5828	-686.2	-296.5
	12	172.3	194	207.4	-98.3								
54	0	2015.8	-1200.6	-1305.6	172.3	3050.5	58.6	-70.6	-67.5	65.9	7.8	-160	-251.9
	12	204.3	1161.4	696.8	233.4								
55	0	90.5	123.7	113.5	117.7	2741.8	-532.9	-207.2	-1490.4	315.2	-61.8	95.4	63.5
	12	-414.9	220	-283.4	-210.3								
56	0	-449.8	173.2	768.9	295.5	-28	-112.8	59.6	-490.1	-139.9	79.1	3689.5	147.9
	12	360.6	-101.9	-171.4	-3.5								
57	0	71.3	274.1	-51.7	-10.9	60.2	22.6	86.1	48	7154.8	-15.3	-101.5	25.1
	12	22.6	-14.7	-107	-104.7								
58	0	-70.6	1520.9	-399.8	-147.1	427.5	156	-2629.2	-92.1	445.9	-264.7	88.9	-158.3
	12	100.2	-188.2	194	-381.2								
59	0	-16.6	-3388	570	-758.8	-201.1	-245.9	100.6	-688.8	447.8	142.2	-207.5	-219.2
	12	303.1	219.8	238.4	233.7								
60	0	-154.9	-44.9	240.7	2966.4	-13.8	83.2	416.7	142.7	-85.5	-2.6	-225.9	172.7
	12	221.8	131.9	1041.5	94.8								
61	0	575.1	-1346.2	-642.8	5094.2	-105.5	-16.1	75.6	399.3	395.7	-72	71.4	234.4

	12	336	246.8	-178.6	101.3								
62	0	3723.5	-3257.4	255.9	-1.7	-178.8	-136.1	227.4	-27.6	-351	-234.7	718.8	286.1
	12	198.3	493.7	-1061.4	-23.7								
63	0	-446.3	-6147.5	410.7	-135.8	-117.2	81.2	35.7	-135	27.3	-71.7	38.7	-27.5
	12	-76.2	275.6	-128.5	38.6								

Table C. 12 Shape codebook 1 for MODE_VQ == 16_16, BLEN_TYPE == MEDIUM

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-43.5	126.2	-8836.9	35.1	66.4	-219.2	46.5	-29.5	-231.6	-77.2	164.6	65.6
	12	-23.9	135.3	132.2	-315.8								
1	0	64	-58.4	-643.3	-3894.8	400.1	2082.2	-98.4	-146.7	-2.4	134.8	-1743.3	-174.5
	12	-158.3	470.9	-10	-146.9								
2	0	527.4	3.2	-1395.6	-152.6	196.3	-146.3	333.6	-454.6	382	169.4	34.1	-1329.8
	12	-1537.4	-60.5	443.1	-98								
3	0	-199.1	1748.8	345.9	-233	-160	-31.8	83.4	-369.9	138.7	134.8	95.7	-42.6
	12	3673.3	-301.1	-314.8	-11.4								
4	0	-1077.7	-2250.1	341.7	752.1	795	274.7	541	-47	-19	-1027.7	-85.6	-202.9
	12	324.1	213.4	1520.3	-147.2								
5	0	-148.2	-629.4	528.1	-401.9	108.7	-695.2	-149.3	-124.6	-496.2	-744.5	-248.4	-169.1
	12	-304.5	301.1	-101	-226.8								
6	0	-91.3	-28.6	27.7	-7032.4	-4.4	-156.6	195.1	198.1	119.8	40.2	210.2	112.6
	12	132.6	15.5	-291.8	185.5								
7	0	45.2	207.1	2128.1	-216.2	-300.6	116.4	4.5	-2638.3	428.9	-55.1	85	-67.2
	12	546.9	-328.6	413.9	-19.9								
8	0	-1607.4	137.3	-59.1	-99.7	-72.4	2668.1	-1.1	-178.3	-159.3	-21.1	-227.5	185.5
	12	-363	-261.3	-111.8	87.3								
9	0	-11.8	1553.1	14.3	3507.7	32.1	190.8	-146	-275.2	31	16.3	-63.9	174.1
	12	10	276.8	43.5	58.3								
10	0	265.6	-53.8	109.9	-70.1	3875	81.7	-101.5	476.2	418.1	15.4	-187.9	290
	12	-424.7	-474.6	58.1	-177.9								
11	0	181.9	-838.4	-56	2667.2	-1894.8	417.3	202.7	-94.4	281.9	871.2	253.2	-526.5
	12	1211.6	-251.4	-106.1	-26								
12	0	-809.9	-292.2	3347.6	205.7	-1525.9	336.3	1904.2	-94.3	-117.5	9.5	40.6	-12.9
	12	71.5	393.5	131.9	-142								
13	0	795	1659.8	-4765.7	365.4	360.3	202.2	24.1	-237.9	269.6	534.6	-170.5	-1412.1
	12	915.3	-231.4	-1132.7	295.9								
14	0	-316	-2738.3	-329.2	-140.3	-5.4	-177.7	-129.6	56.1	132.1	-145.4	33.5	148.3
	12	-24.8	-4722.9	164	37.3								
15	0	-35.5	-5.1	-354.1	84	157.7	51.5	125.4	140.2	-536.1	-40.4	-88.3	128.4
	12	-269.7	-221.9	1195.8	142.3								
16	0	-4742.1	-10.3	-32.5	46.9	475.8	-318.7	-233.7	49.7	28.6	-0.8	64.1	618.5
	12	174.1	-298.9	-209.5	-576.1								
17	0	68.9	2105.8	-273.8	-301.4	3255.6	606.9	82.1	112.8	-538.1	179.9	109.4	172.5
	12	429.3	174.4	-523.1	-116.1								
18	0	-8451.6	88.5	-64.7	-141.7	45.3	-196.1	41.4	-170.9	12.9	-82.8	169.1	109
	12	-290.2	-349.3	-313.5	-244.2								
19	0	-495.2	-368.4	-2767.5	-2421.9	1471	-1008.8	543.8	-151.1	-224.8	52.7	43.4	114.8
	12	-158	329.9	-363	347.9								
20	0	11.9	-139.5	755	26.3	40.3	5306.5	15.6	-155.9	507.8	40.2	55.6	198
	12	-199.3	131.7	-74.4	-78.1								
21	0	-197.8	60.8	19.9	75	-56.5	28.7	44.2	137.3	3922.7	6.5	129.9	71.7
	12	107.9	-28.9	47.2	136.7								
22	0	1218.8	-118.8	-638.4	-1563.7	-362.3	257.2	-190.1	2130.1	771.1	-324.5	53.8	-127.7
	12	48.2	352.3	280.5	128.8								
23	0	300.9	265.5	4.5	78.8	-1091	61.7	19.1	-29.1	-69.7	35.2	3.5	-117.3
	12	282.1	61.7	-140.1	-3886.1								
24	0	-414.6	30.7	144.3	-1790	-521.2	191.5	458.3	-30.5	-127.8	1240.3	203.5	1081.3
	12	49.3	949.4	499.7	-162								
25	0	-2845	117.4	9.2	-3046.7	44.6	-219	-194.7	-273.5	138.5	-318.4	506	-124.1
	12	138.1	-128	56.3	-57.1								
26	0	-34.2	3342.1	-1723.3	80.6	-609.9	-33.8	-29.1	-191.6	-31.2	894.1	350.8	-198
	12	396.8	75.7	-316.4	459.2								
27	0	-1403.6	-593.1	-35.8	-320.5	-631.8	-1099.3	-83.9	-674.1	726.3	-72.2	-80.1	709.4
	12	-109.9	0.2	-90.6	248.9								
28	0	-164.5	-397.7	-1170.5	-566.5	-1871.3	-499.8	1738.4	-220.9	751.8	223.8	515.8	-307.9
	12	-517.1	-971.7	275.2	-116.7								
29	0	158.5	26	3621.2	45.4	-113	-1703.7	-348.7	51.3	243.9	-259.8	-1479.3	109.2
	12	463.8	560.4	636	-6.1								
30	0	1264.1	-4161.9	-589.6	759.5	766.7	78.2	826.3	-112.9	29.7	624.6	-65.3	234.5
	12	-89.6	-25.8	-42.4	193.2								
31	0	-332.7	-1.5	-2339.5	3551.9	91.7	-294.7	61.4	-205.1	163.8	-270.7	-196.4	84.9
	12	-199.1	-135	-408.1	-690.1								
32	0	-685.2	249	-680.6	480.4	-96.9	-92.2	-607.1	-496.6	-117.3	-11.5	39.7	-375.9
	12	-211.6	1582.8	382.9	-24.2								
33	0	44.1	-109.7	359.8	-92	-6138.1	96.7	-57.7	48.9	-78	22.1	169.2	54.3
	12	-223.6	45.9	-20.8	-114.8								
34	0	-112.3	8451.3	164.5	1.1	1.9	-19.3	118.4	-3.7	-50.1	48.5	80.8	-25.7
	12	-115.4	212.7	-108.5	-145.3								
35	0	241	1076.8	301.9	17.6	19.9	-194.2	-18.4	85.7	64.6	14.5	6795.7	60.3
	12	-96.9	102.2	-132.6	-56.2								
36	0	-178	-937.1	1468.2	-64.3	-765.3	551.3	29.1	1480	1141.4	470.5	-602.3	-246.9
	12	187.5	1202.7	-161.1	-858.1								
37	0	-36	-657.5	13.7	973.4	-20.8	35.1	-2155.9	-911.2	278.8	87.4	-24.1	404.9
	12	0.9	-276.4	-39.5	161.1								
38	0	372	-2816	-40.5	218.1	127.7	2919	-304.3	190.1	246	293.7	-84.5	-173.5
	12	9.8	-550.2	537.2	180.3								
39	0	137.8	46.3	191.9	487.1	-63.8	-216.1	2583.4	-373.7	145	-1291.8	-128.2	-143
	12	-17.8	-60.6	-145.8	116.5								
40	0	-2152.2	39.7	-360.7	3628.4	3.8	-86.8	196	356.8	-166.6	-14.9	64.2	45
	12	61.7	287	10.8	-63								
41	0	928	665.8	1073.3	114.3	-1883.4	-592.5	84.7	-350.8	-1.6	-32.6	-203.8	-87.5
	12	290.1	163.4	55.2	118.5								
42	0	800.8	164.3	-21.1	52.7	64	-15.9	-228.5	-69.2	19.8	51.1	37.2	-180.4

	12	-23.8	3.6	3892.4	16.9								
43	0	-84.4	-79.7	5299.5	-82.6	13	458.6	-557.7	109.3	128.4	326	-105.7	-31.5
	12	164.2	-372.9	-29.4	-157.2								
44	0	1973.5	24.8	801.2	91.7	-142.1	48.9	-3207.7	223.8	-218	-177.8	83.1	90.4
	12	119.7	-133.1	6.7	-128.4								
45	0	-122.3	4537.1	102.5	935	2206.7	-461.1	-150.2	198	553.4	235.9	110.9	-604.2
	12	52.6	174.1	193.3	-78.4								
46	0	1077.7	-1166.6	239.3	398.1	-57.2	130.2	-1.3	-69.1	213.9	2770.4	-90.2	-295.5
	12	-153.6	-242.4	-147.3	-97								
47	0	1951.9	-72.1	-2869.6	398.8	-72	11.8	266.2	75.5	172.1	219.1	-260	-53.8
	12	-33.9	74.3	-50.2	-95.5								
48	0	686.2	4838.9	12.7	-363.7	-335.9	1086.2	-70.8	35.6	-52.2	75.1	149.9	126.3
	12	109.7	320.1	-111.9	26.6								
49	0	-80.5	-143.1	-5.5	-43.1	-0.3	41	8255.1	151.6	76.5	18.9	105.7	24.1
	12	-178.3	206	31.4	53								
50	0	-47.1	82	141.4	-1192.7	1120	640	-120.5	-129.8	-25.2	-17.7	28.3	-2869.2
	12	679	-540.7	531.8	848.2								
51	0	-545.8	798.2	-13.3	98.3	-95	-160.5	-81.6	3525.9	91	366.1	408.5	-642.2
	12	-29.1	-395.9	-453.6	79.5								
52	0	81.6	-1043.4	49	187.5	1429	-3489.9	61.7	369.8	678.8	-442.6	-94	-30
	12	-2.7	-470.2	-53.9	-96.3								
53	0	-1975.2	-2190.2	-874.6	-891.1	-53.2	15.5	-333.1	556.9	-317.9	-54.3	387.6	-242.2
	12	-457.1	608.5	-677	-317.7								
54	0	116.3	-129.1	26.1	-52.8	-4.4	38.9	7.5	4941.5	-19.6	-293.6	-169.8	224.1
	12	207.5	19.9	85.7	-69.3								
55	0	3453.8	242.5	1153.4	-32.1	1722.1	81.3	158.7	195.9	57.8	66.7	291.2	-323.2
	12	130.7	338.1	-415.5	464.1								
56	0	88.9	49.3	-178.1	-138.1	-259.1	2093.2	-134.7	206.9	-309.9	-1604.1	-169.3	792.1
	12	352.9	-253.2	-791.3	171.1								
57	0	227.6	-308.6	-269.2	-802.6	-2674.1	-3.5	-2514.4	108.3	116.5	-289.2	-63.9	-436.6
	12	-24	65.5	-121.2	297.4								
58	0	-355.1	238.7	-1684.7	-1221.4	11.8	-400.2	-46	141.5	-395.2	443.2	-1642.2	-186.4
	12	307	-975.8	758.7	-217.8								
59	0	-138.1	1111.1	-2744.3	1.9	-648	-315.1	-29.6	-20.1	-133.6	127.2	-84.7	3054.3
	12	-284.6	-740.2	88.5	739.8								
60	0	2373.5	-1907.4	231.3	-154.6	-84.4	39.7	158.4	235.2	918.6	-274.8	-289.8	-278.2
	12	-224.2	-195.7	152.9	604								
61	0	-2729.9	142.3	-4717.2	37.8	329.1	174.3	-60.1	-140.1	76.2	-141.2	-72.3	-101.7
	12	-241	-181.4	-96.3	543								
62	0	-581.1	-1502.7	1685.8	1420.9	53.3	-1447.6	-221.7	-419.4	-406.4	307.6	-345.9	104.7
	12	-740.2	-353.8	-16.7	533.1								
63	0	2634.6	541.1	-261.1	-244.1	83.8	-284.1	-1.1	-350.3	-76.5	-73.1	361.8	-165.2
	12	-165.2	-387.2	-144.6	-1350.6								

Table C. 13 Shape codebook 0 for MODE_VQ == 08_06, BLEN_TYPE == LONG

VN	FN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	148.1	-16384	-93	-230.7	-23.8	-6.5	66.7	2.7	-46.9	-86.6	24.8	-100.3
	12	12.6	-265.4	136.3	-7	-28.1	86.8	-14.3	-424.6	-36.6	-229.5	-4.7	113.5
	24	-30.4											
1	0	90	-59.8	3216.1	-114.7	-3167.6	-113.3	142.1	100.1	196.2	52.3	-101.7	-180.1
	12	-156.8	22.3	-64.5	-297.1	110.4	91.6	39.1	-74.5	165.9	172.4	-167.7	46.5
	24	63.1											
2	0	95	-12.8	62.4	30.4	-57.2	13.8	-1482.5	-18.4	9.7	-6.1	18.6	12.5
	12	-39.9	-67.3	18.3	17.3	-107.5	89.1	-54.6	-228	231.7	-76	165.2	5964.2
	24	270.4											
3	0	103.5	-48.8	-49	143.6	22.4	-16.2	28.3	4448.7	-161.6	-159.7	59.9	24.4
	12	82.4	-52.6	7.8	-72.4	245.3	23	-91.2	-12.1	-192.3	-106.3	-128.5	-10.8
	24	1762.1											
4	0	68.5	-51.4	78.4	-14.5	1.4	-6841.2	-105.7	-9.3	44.2	-17.4	-15.8	91
	12	63.9	-100.9	-51.4	-15.3	-75.7	37.4	46.7	-21	-135.8	-322.4	423.1	18.5
	24	179.2											
5	0	176.7	-222.2	49.3	-152.1	259	-74.7	17.3	-79.3	11.3	68	83.1	56.2
	12	-59.7	7567.2	-61.4	-59.4	-22.5	26.1	-29.9	-11.8	-96.3	281.2	-105.9	329.9
	24	35.8											
6	0	-68.5	-3528.6	3342.6	187.2	24.9	-34.2	243	-45.4	138	-150.1	-20.5	55
	12	-32.6	-164.7	-322.9	183.7	224.7	111.9	140.6	41.4	61.7	98.4	-236	-3.3
	24	393.4											
7	0	-25.6	2051.5	15.4	173.1	-167.6	171.2	2715.3	-2	-10.6	-150.5	119.5	-275.1
	12	72.7	73.8	-498.5	-189.6	85.4	65.4	-37	-125.4	1180.6	62.6	48.7	164.1
	24	-907.2											
8	0	-40.3	-8133.3	25.5	-37.8	-38	87.1	-16.3	-25.9	140.3	16.5	32.5	28
	12	-33.2	-115.8	11.8	-92.4	12.3	128.8	135.3	-449.2	6.3	-262.9	-45.9	-37.2
	24	-796.8											
9	0	565.3	227.2	-1043.3	246.9	187.3	158.8	-128	3	-10.9	110	14.5	-35.7
	12	115.7	-154.1	-84.1	6834.5	-17	129.8	-47.2	-146.1	-87	157.9	20.4	-203.3
	24	542.6											
10	0	258.7	-203	-3540.1	-87.8	-3166.6	-29.7	-21.2	111.6	156.2	164.3	135	53.4
	12	-0.3	-46.9	122	-252.8	60.4	88.9	-132.8	273.9	45.9	341	25.6	-96.2
	24	-214.6											
11	0	930.7	124.5	106.5	-47.2	-15.6	1.2	31.5	10.1	18.6	-32.9	-776.9	34.2
	12	15.6	-90.3	-43.9	-148.3	8151.2	-22.4	1.9	-115.7	-84.6	-21.2	-44	64.8
	24	-121.2											
12	0	-13.5	30.1	-18.5	109.4	22.8	72.4	38.7	-98.7	-242.7	-5970.3	-41	32.3
	12	-96.5	-46.6	91.7	80.7	28.3	-17.5	-3.1	-122.7	69.3	-154.4	-38.8	81.6
	24	-543.7											
13	0	-16384	-80	-359.7	-82.8	99.5	17.1	3.3	-97.1	15.7	-40.7	128	-1.7
	12	-69.7	64	-10	-68.3	-7.1	-20.3	-72.2	104.6	-0.6	115.8	-36.4	23
	24	-99.5											
14	0	23.7	79.8	-58.5	23.3	1.9	-1123.7	-50.8	119.2	21	-5.9	-82.9	9.7
	12	-23.5	-27.9	-90.5	94.6	48.1	-7276.8	87.2	44.5	108.5	-207.4	40.4	-24.7
	24	119.5											
15	0	-54.3	-44.8	65.4	22.9	-15.5	9.4	-60.2	-3.6	6528.1	49.3	-13.8	72
	12	106	-89.1	185.1	-10.1	27.1	20.4	-27.4	-4.2	-185.3	-137.3	-9	-101.4

	24	282.4											
16	0	-8818.4	-123.8	-25.7	-73.7	-14.2	11.9	-82.7	-81.6	-55.5	-133.8	-7.1	13.1
	12	-104	-94.1	259.2	-1.8	14.9	48.7	-84.7	-96.5	-71.5	20.4	19.3	-57.2
	24	376.7											
17	0	54.7	4.4	-11.3	-11177.3	-101.7	1.6	35.3	230.2	-21.5	-89.5	-8.3	63.1
	12	78.9	-149.5	-41.6	102.4	86.3	65.9	-4.9	142.2	150.8	99.7	73.4	-46.8
	24	43.2											
18	0	3760	48.9	133.2	-3.1	39	81.2	14.4	-2650.6	-35.9	93.7	42.8	59.7
	12	47.7	22.9	137.1	273.9	-33.6	60.2	-21.3	-126.5	152.6	119.4	116	-11.6
	24	684.4											
19	0	-39.3	-89.8	350.2	245.6	-209.2	-16.3	-109.7	76.1	-3	233.9	46.3	38.8
	12	3120.6	-0.3	-40.3	-123.3	38.1	-14.8	-116.5	47.6	-21.2	-117.4	-16.2	17.6
	24	396.4											
20	0	295.7	-123.8	-39.3	429.4	-46	-91.4	21.8	-134.7	-238.4	-66.7	5.2	530.2
	12	-28.6	-87.5	-8383.4	-46.4	-47.2	30.6	10.2	132.3	-62.5	-101.4	191.5	97.4
	24	-385.6											
21	0	69.2	-118.4	77.9	7.9	-21.8	-0.4	5906.1	14.3	-41.7	14.8	14.2	6.5
	12	-94.4	-30.4	101.4	0.4	3.1	40	-22.3	99.7	-3.2	106.8	92.7	-94.4
	24	76.1											
22	0	171.1	51.8	-20.8	4299.6	73.4	5.3	128.5	52.1	3	-94.4	-16.5	1
	12	-68.5	-63.9	26.3	-107	11.8	-41.1	-31.9	136	-47.3	389.2	-164	-44.1
	24	5706.5											
23	0	-74.4	-107.2	-71.7	94.5	-9956.4	-60.7	-36.6	-27.8	11.1	-121.7	115.8	-7.3
	12	-75.2	-140.3	-92.9	-33.2	57.6	-10.3	-52.2	26.1	-47.7	30.3	-50.4	-7.7
	24	20.3											
24	0	-60.1	-108.8	29.2	-150.3	5	-23.7	13.3	3.2	89.6	2169.9	-19.8	-15.7
	12	-70.4	36.8	-29.7	-4.3	28.7	47.3	28.3	-48.3	91.3	7292.7	70.3	10.2
	24	-102.6											
25	0	12.1	25.4	-10.9	77.2	101.2	95	105.5	10320.9	-64.7	-352.9	-4.6	-160.8
	12	-18.7	-71.8	-68.2	26.9	105.2	4.5	73.1	29.8	-5.6	-51.8	65.8	6.7
	24	-838.9											
26	0	226.3	164	15894.5	-48.5	59.1	-92	42.9	-63.4	77.2	-70.8	62.9	-46
	12	-27.3	175.3	126.6	-115.7	185.4	16	-3	129.6	50	-138.6	191.7	37.3
	24	-262.7											
27	0	-148.3	-168	-239.4	133.6	-12.8	-226	87.6	-69.5	2776.7	-2927.2	158	-92.2
	12	-75.7	-123.7	15.2	-17.3	64.8	116.9	28.7	-124.1	-105.5	-62.4	43.7	-26.4
	24	24.4											
28	0	-50.6	28.4	72.2	-55.2	-6.6	44.7	3.1	-1541.4	-16.7	13.2	-96.5	62.9
	12	-43.6	-3157.3	-46.1	-34.7	-34.3	44.9	70.1	-38	-79.6	181.1	98.3	-29.3
	24	-250.3											
29	0	-148.3	-106.3	374.4	59	-51.7	159.4	196.9	2515.7	2566.1	231.7	-53.6	-225.6
	12	117.8	-194.5	-261.7	-292	-7.2	-51.1	65.3	-371	-374.5	399.2	-6.3	-85.3
	24	-509.8											
30	0	27.2	-20.7	-56.5	114.8	-113.4	-2643.5	2735.6	146	74.4	-336.3	-59.3	-7
	12	116.5	-94.1	269.8	-46.8	-2.3	-130.9	299.2	-72.6	-48.1	-10.5	-61.8	-79.2
	24	317.3											
31	0	5156.7	-2128.5	22.9	210.5	-7.9	15.7	119	105.3	134.5	-86.3	-52.4	-0.7
	12	-25.9	40.5	-28.4	-84.4	-50.6	-107.2	-78.5	177.7	-105.3	-40.4	232.5	106
	24	-288.3											
32	0	-2616.6	-157.5	75.4	224.8	-222.8	96.1	3041.5	49.2	-17.5	-106	-15.4	217.7
	12	82.5	104.9	25.3	365.3	-226.8	-26.7	-75.3	-180	-86.9	-6.7	280	-49.7
	24	331.5											
33	0	-164.7	-44.8	182.6	116	148.3	-44.1	46.8	-34.8	105	-186.9	19	8570.4
	12	28.8	-10.7	218.6	-47.7	7.6	-16.2	108.6	-101.4	18.6	-88.2	187.5	-102
	24	-261.4											
34	0	100.9	-1.6	63.9	-1514.8	-46.9	-59.6	-39.1	11.6	55.2	-0.9	-7.8	53.2
	12	68	-3.6	18	17.5	-6.5	117.4	-81.3	5596.8	151	-276.5	58.4	-89
	24	24											
35	0	-109.1	-105.8	2218.7	-28	68.8	2337.8	197	-245.2	75.8	-493.4	-24.1	41.7
	12	98.4	125.8	-284.7	-40.4	54.1	-0.1	-81	59.9	115.1	-236.8	42.3	32.9
	24	-591.3											
36	0	107.9	649.7	95.7	-9.6	50.5	-0.5	78.2	32.6	-27	18.6	59.3	16.5
	12	-8761.2	-72.1	40.7	33.9	-141.1	-112.9	-16.5	349	-3.3	24.1	0.5	-71.2
	24	516.7											
37	0	51.7	-273.4	-2299.4	-2561.1	-99.7	95.7	17.1	-236.2	24.8	320.6	149.2	176.4
	12	-131.4	9.5	86.3	-65.9	27.4	1.5	-33.6	-78	7.7	-245.8	-149.6	82.4
	24	71.5											
38	0	99.1	-101.3	-164.6	-50	-45.8	3317.2	304.6	-35	1.4	61.4	50.3	20.1
	12	159.9	148.9	59.6	-2.9	-167.2	-66.3	173.9	-123.9	50.5	-252.3	73.4	29.8
	24	-307.9											
39	0	-2634.6	-1478	-196.7	51.3	11	6.3	-119.2	-42.7	27.4	41.2	24.8	-53.2
	12	51.9	0.6	-57.5	2265.4	77.2	28.1	-23.3	66.8	122.8	-132.1	23.1	-373.8
	24	474.9											
40	0	124.9	-16.1	108	-37.6	-22.7	23.9	11.6	25.4	-31.3	-12.8	38	-2893.7
	12	48.3	-77.1	21.9	47.8	-3.3	30.9	-50.9	78.7	86.5	12.3	-71.9	97.5
	24	162.9											
41	0	147.6	3470.1	-83.5	-3580.3	104.2	-9.2	75.1	-87.3	12.6	113.6	-72.1	164.6
	12	-27.2	18.2	140.4	27.3	-56.3	-95.1	0	213.2	-65.9	-186.8	178	-135.1
	24	464.6											
42	0	-83.7	-27.2	-92.8	-64	5.2	-4	-9.2	-136.4	1715.8	80	18.9	-5.3
	12	16.3	-10.6	149.7	73.6	226.2	176.2	-9.6	117.5	6346.4	-20.5	-230.9	-101.4
	24	38.2											
43	0	89.5	-113.2	-3277.1	89.6	22.3	62	2995.3	-249	18.8	-209.8	191.9	91.7
	12	26.7	163.3	-348.2	-361.4	230.2	21.4	-30.2	-191.4	99.8	-19.3	-74.2	-76.7
	24	186.9											
44	0	-155.9	48.7	7756.1	-16.6	35.4	-49.3	-18.4	-18.6	79	97.1	42.8	-2.8
	12	46.4	105.8	104.1	34.9	32.7	-87.5	-9	131.1	75.8	-58.5	91.8	-57
	24	-143.9											
45	0	1374.4	-128.9	-62	158.8	113.7	-53.3	-65.4	-1.9	3085.4	214.8	-25.4	44.6
	12	89.7	-38	-113.6	35.7	-77.5	-0.4	123.2	-42.8	-61.2	-33.8	-84.7	24.1
	24	-367.9											
46	0	1942.3	-5169.4	-112.4	161.9	-27	-12.4	27.1	-47.3	-5.1	288.2	-87.9	-122
	12	-121.6	116.8	-290.6	-195.9	124.9	-45.8	-132.5	0.4	-151.4	-84.4	425.7	122
	24	6219											
47	0	17.9	-31.7	-71.5	-2459.1	-146.4	108.1	2861.7	-195.9	-120.6	256.6	25.6	-20.4
	12	-45.3	83.8	52.8	-10	-79.5	76.7	-8.7	29.3	-13.4	-133.3	-167.4	81.6
	24	39.3											
48	0	-3631.4	-11.3	2298.5	-15.1	33.7	-61.2	-51.6	13.8	-25.3	103	-91.8	-37.4
	12	-18.7	-16.9	-295.8	-82.3	41.6	45.7	174.7	-25.9	115.3	-78.6	118	-93.7

	24	135.7											
49	0	44.2	3424	3065.7	-137.6	31.7	-2.6	-13.4	16.4	-15.1	38.4	48.2	32.6
	12	-139.5	-111.3	-29.9	106.1	-73.3	194.2	84.9	408.9	-248.2	-348.2	40.1	96.6
	24	254.7											
50	0	-112.4	135.1	-264	-289.9	-2350.2	-350.1	-2607.9	52	173.2	-339.4	0	104
	12	99.1	6.1	28	271.5	78.9	-34.9	18.5	57.7	47	145	272.8	-91.4
	24	126.2											
51	0	4439.8	-40.4	165.7	-3218.5	-80.3	-71.3	-17.6	114.4	248.5	5.1	60.9	-120.7
	12	-51.5	-130.5	42	95.5	-68.9	-55.2	151.2	-219.1	-67.7	6.6	-36.3	-62.3
	24	-214.8											
52	0	-0.4	906.1	175.2	-2181.6	-49.5	97.8	-304	2607.9	-192.1	87.2	-171.5	12.2
	12	-97.8	-262.5	71.4	-111.2	40.2	-127.2	-44	-95.8	-51.3	-13.2	-218.4	-87.9
	24	162.8											
53	0	-374	-4164.2	13.2	-148.2	3.2	-18.5	-16.1	52.4	-32.3	-15.4	-33.8	-41.8
	12	-34	56.2	53.5	23.2	-6.7	6.1	36.9	-102.6	10.3	120	-65.5	130.6
	24	-241											
54	0	-166.2	1815.4	97.2	-75	-85.9	-17	-63.5	-45.1	19.3	25.6	43.4	4.4
	12	22.6	53.4	27.8	-21.5	-77.5	17.4	-24.1	-0.2	-45.3	-74.7	-588.5	-130.9
	24	871.2											
55	0	-97.9	551.6	70.8	2024.9	-214.3	36.4	-278.7	262.9	88.2	3002.7	-64.2	-237.2
	12	-103.6	10	28	55.3	-2.3	-92.7	-79.4	92.5	-27.5	-184.4	-160.1	-12.8
	24	-175.7											
56	0	-437.6	-170.3	-1638.6	-13.7	-104	24.7	-6.2	-102.3	-60	-57.8	-37.4	-13
	12	-47.1	35.3	-3043.5	26.9	-9.4	39.9	40.5	15.6	29.7	92.1	-53.9	-41.9
	24	249.2											
57	0	38.2	-7.7	-29.8	-18.2	113.3	-80.5	-161.8	-32.2	-24.1	-37.6	5134.9	-33.1
	12	-18.8	-61.8	20.2	-90.1	-24.1	101.7	-10.5	-54.9	-12	-142.1	-21.6	35.4
	24	441.1											
58	0	86.2	101.3	-47.3	-59.4	-1634.2	21.9	78	113.5	-1.6	11.2	96.4	-49.9
	12	7.7	-51.3	-5.5	164.1	-23	124.3	4146.9	-78.2	-102.5	33.1	-96.3	-39.1
	24	166.2											
59	0	-73.9	-2504.6	197.4	-72.2	-2554.6	-1	148.5	-114.6	-10.6	145.3	-47.6	77.9
	12	99.2	-9.7	65.3	-38.5	-117.6	-161.5	-271.2	-45.6	116.5	201.9	-133.5	-81.7
	24	-5806.6											
60	0	17.8	-96.7	4395.8	-81.3	-18.8	-52.4	173.9	29.4	-30.3	56.9	145.6	106
	12	81.4	-29.8	42.7	-86.1	-68.7	-13.4	-91.3	49	-84.1	279.1	29.3	-169.5
	24	-229.6											
61	0	34.7	123.6	225.1	-2600.3	2494.7	130.5	27.8	-52	-0.5	228.8	134	74.6
	12	16.1	53.6	-63	-38.4	123.1	-172.9	-61.1	221.3	133.9	2743.9	41.4	-254.3
	24	-306.9											
62	0	2169.8	-319.2	9.9	2.8	-41	34	45.1	-48.9	-65.2	13.2	16.6	36.9
	12	2.6	-4.4	-56	76.2	11.1	30.3	-17.1	75.1	7.6	-162.1	4.7	-16.1
	24	326											
63	0	5.5	-23.4	-23.5	-77.2	-4904.6	-8	11.6	-78.5	41.6	-29.8	29.1	34.7
	12	-11.8	13.6	64	183.6	143.6	73.3	81.1	-51.1	-117.9	-13.7	122.2	81.2
	24	-494.6											

Table C. 14 Shape codebook 1 for MODE_VQ == 08_06, BLEN_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	98.4	25.2	130	18.2	-17.8	15582.7	-105.4	247.7	5.1	-9.2	-85.1	111.8
	12	-41.8	-75.8	-2.8	-77.5	55.4	-115	76.1	-110.7	-26.4	55	363.3	-73.2
	24	-200.5											
1	0	-209.4	259	-352	-2247.6	116.5	17.4	-179.1	-137.4	-98.9	-17.8	23.8	-2880.6
	12	-15.5	-25	-243.6	105.7	-97.5	124.1	-74.4	-237.9	-157.8	-59.6	-67.7	354
	24	-95.9											
2	0	-310.8	-184.7	-85.8	198.3	-3453.3	-5.7	111.4	227.9	-85.8	-107.8	23.7	-23.8
	12	230.9	113.7	-79.3	-6	-103.5	50.9	45.5	25.4	-1.7	196	11.3	-18.6
	24	6459.3											
3	0	-117.6	1425.4	67.1	1187.4	24.4	-63.2	-110.7	-223.4	-41.9	-51.2	-74.3	194.2
	12	67.6	-24.8	19.5	-42.9	-113.8	131.1	57.4	9624.8	-321.2	96.5	142.2	92.5
	24	745.7											
4	0	-148.4	-49.1	276.5	-63.3	24.7	-56.2	70.3	46.6	18.7	80.9	-3034.5	-14.7
	12	22.6	-48.6	-11.4	-112.8	-57.6	-2.2	42.5	25.1	24.9	-287.5	-25.2	8.1
	24	124.2											
5	0	-36.1	-135.3	-1121.3	-205.6	-47	-9.4	98	113.6	53.5	-11.9	45.4	35.7
	12	39.4	94	-25.2	-120.8	-32.9	3983	66.2	84.9	88.1	-215.9	-130.3	17.7
	24	399											
6	0	16384	-52.8	-128.3	-25.1	203.9	84	24.3	-183.1	0.2	15.9	-67.9	-44.1
	12	150.8	235.8	128	-299	7.7	-76.6	-42.9	17	67	-161.8	29.2	-58.6
	24	-48.1											
7	0	-432.1	-61.2	-116	-16.3	56.1	-9181.1	72.5	-19	-79.4	1.6	25.3	35.9
	12	-18.2	39.3	78.6	-15.2	0.2	131.2	133.3	-50.7	118.5	80.5	-159.5	-120.7
	24	-773.2											
8	0	-23.5	56.1	24.7	-78.3	6884.4	13.1	36.2	5.7	-16.5	-54.5	-106.7	-61.8
	12	-29.2	22.9	-46.1	91.9	56.8	8.8	-103.3	78.5	-31.6	-6.2	0.5	65.4
	24	189.2											
9	0	285.8	-459.3	15.4	52	-62.4	-41.7	104.7	404.4	12.2	-185.9	11.3	10.4
	12	-78.6	-4238.5	12	-17.8	26.8	-36.9	93.6	-28.9	78.8	35.5	136.9	164.3
	24	37.9											
10	0	-201.3	-109	-313.4	-47.6	58.1	-0.5	-4270.9	-90.2	42.6	9.5	103.3	51.2
	12	-77.9	3	36.5	-72.7	-92	142.7	142.6	119.7	-36.5	199.5	10.9	-25.5
	24	-439.5											
11	0	119.4	54.2	445.9	4.9	1.9	311.7	-19.6	156.5	494	62.5	32.1	-69.3
	12	2.6	38.4	-4393.5	72.5	-53.3	24.3	-5.3	184.6	-26.4	-111.6	24.8	-128.8
	24	138.4											
12	0	20.3	-302.2	-36.3	41.8	377.3	57.5	-298.2	-24.7	-22.6	173.3	-28.6	33.7
	12	-5106.9	-1	-31	-85	-59.5	-36.4	-8.2	8.2	-4.5	-57.6	-30.5	-29
	24	48.1											
13	0	-140.4	96.3	-380.2	44.4	24.1	64.8	-13967.6	33	-26.4	31.5	22.7	-60.4
	12	-45.3	-129.2	12.9	-45	-1.9	427.7	172.1	233.6	110.1	104.8	262.8	-95.5
	24	373.6											
14	0	-270.4	1313.5	-2100.2	-25.6	40.3	-13.5	143.3	19.7	90.4	-135.5	91.8	116.4
	12	200.5	-2170.8	122.6	205.6	136.3	-242.1	-45.6	83.3	291	-338.5	81.2	-51.5

	24	-215.6											
15	0	-100.9	145.6	-16384	-109.5	97.4	-223.7	30.2	-96	-91.8	188.8	82.2	-65.1
	12	25	138.4	36.5	-53.2	-279.9	-184.4	-170	6.9	-33.6	-61.9	-333.1	5.2
	24	109.9											
16	0	26.8	-22.2	-43.2	2854.2	2677.3	-274	-45.9	99.9	61.8	24	43.2	-25.9
	12	-37.9	28.7	84.8	-32.6	-74.5	-30.7	-4.8	-36	136.5	-211.9	-94.8	-7.7
	24	194											
17	0	145.3	-115.2	-2465.9	2887	-114.6	-58.7	10.5	76.6	-345.3	24.9	52.8	-191.4
	12	-26.3	228.6	80.7	138.1	47.4	-16.8	-66.7	-152.8	-75.9	168.3	126.4	104.6
	24	149.7											
18	0	211.7	80.1	80	49.6	72.1	23.1	-9.5	202	-4445	-112.6	-77.3	-22.8
	12	117.7	88.5	-21.4	-91.1	-115.3	78.4	-33.1	-68	-80.6	46.8	229	80.8
	24	-642.8											
19	0	2.2	39.3	218.7	16384	192.8	-38.8	58.1	57.3	89.2	81	95.4	-21.5
	12	143.8	-17.5	4.6	111.2	2.9	-44.5	5.5	-17.8	125.3	-318.6	241.6	-51.8
	24	126.1											
20	0	945.4	4.4	122.8	7.3	-1.7	34.5	-25.4	1	16	24	133.5	-6.5
	12	25	19.2	6.2	-13.4	12.5	-105.2	34.2	-69.1	109.4	27.1	-5665.1	29.7
	24	-145.6											
21	0	-1817.3	16.7	73.7	-40.5	-103.3	118.5	48.8	49.7	51.7	30.8	-51.7	72.8
	12	-27.5	-12.8	-17.2	52.2	-31.3	-47.5	-31.1	3725.3	77.4	236	-42.4	-16.2
	24	-221.3											
22	0	165.1	-8.3	-43.5	4.5	7.4	137.7	19.8	-2800.3	121	-130.3	-13.1	1.6
	12	134.4	0.8	-35.7	-65.2	54.1	-3.4	-29.3	-152.7	58.2	56.5	-116	-62.8
	24	367.5											
23	0	-74.4	-3906.9	50.2	-3115.6	20.4	-67.8	-32.7	234.6	20.6	76.9	59.6	86.9
	12	-37.3	-165.7	-98.2	43.5	9.8	-114.2	-5.5	7.5	96	-150.6	182.8	62.6
	24	-488.9											
24	0	4220.6	89.9	77.8	-21.2	-162.5	-2914.7	-48	11	-115.2	122.4	18.7	175.1
	12	53.4	307.8	143	54.2	95.9	133.8	-56	194.1	196.2	-209.1	189.7	46.1
	24	-996.3											
25	0	-134.9	10.6	17.4	-13.2	-17.2	21.5	8396.9	12.1	13.5	-85.4	42.6	-27.9
	12	-49.4	-84.5	-23.1	92.2	124.8	-15.3	-103.7	77.7	-97	-103.5	88	121.1
	24	-141											
26	0	-216.4	-1072.1	70.1	139.1	-25.1	-18.2	55.3	-237.8	-1.1	-28.6	6.8	-15
	12	154.7	-14.6	23.9	-65.6	4231.7	-19	-37.1	175.3	1.5	-77.6	-45.6	-3.1
	24	172.3											
27	0	123.2	45.7	-59.7	47.1	345.7	3.2	97.2	92.4	25.9	-2	-8825.7	82.9
	12	-8.1	-55.8	32.4	73.1	80.2	-54.3	31.3	80.9	27.7	-33.3	-84.9	-42.5
	24	35.8											
28	0	-801.6	-69.5	22.3	-2503.6	18.1	-2525.8	-193.1	146.4	0.6	73.6	-92	229.3
	12	-212	298.2	-24.1	-22	-0.5	163.2	55.4	-228.1	46.5	-882.5	229.9	324.7
	24	60.7											
29	0	11.8	-88.9	17.8	11.4	-46.5	-6	-49.9	6553.4	148.4	187.2	-47.8	-67.9
	12	4.2	5.6	25	-27.2	-8.3	3.2	-99.7	-133.9	79	114.1	15.6	-77.3
	24	-189.9											
30	0	-118	2408.7	58.3	-105.4	-35	2966.4	56.8	-179.2	58.8	190.8	-87.3	233.2
	12	95.9	-70.6	271.1	58.1	-100	-4.3	-68.2	55	31	120.7	66.9	57.9
	24	510.3											
31	0	-43.1	40.1	-134	31.5	-144.4	-22.4	-20.8	124	-9869.6	11.8	50.5	78.4
	12	113.7	-13.1	-103.3	34.4	46.6	-23.4	33.6	-22.5	-115.6	-79.1	-190.2	-213.2
	24	788.9											
32	0	248.9	133.4	61	7767.3	-34.1	1.4	4	-27.4	-39.2	-46.6	65.6	143.8
	12	43.6	8.8	13.5	97.7	-103	94.2	2.1	160.1	15.9	-13.4	6.1	-61.9
	24	-862.2											
33	0	190.1	2078	-146	300	-7.6	-58.8	99.4	224.2	-2618	-101	37.5	121.6
	12	-235.4	31.4	35.1	-35.7	-57.4	-217	-83.9	-371.9	-206.9	268.8	42.2	157.3
	24	-1026.4											
34	0	-68.7	-12106.7	30.9	61.7	-101.4	-2.8	20.2	-49.9	-57.7	-124.2	101.5	62.9
	12	76.2	-19.4	-18	21.6	16.4	-68.5	-73.8	22.8	104	362.7	-10.1	-60.9
	24	-1240.5											
35	0	-6265.2	-66	64.7	35.6	-88.7	-104.9	-30.1	-19.1	52	-14.8	32.8	102.1
	12	108.3	-65.8	-114.7	-63.3	-17.4	93.8	248.2	182.1	65.8	-18.1	-34.5	-46.7
	24	-264.2											
36	0	-11497.9	158.1	10.7	15.8	-48.8	40.2	22.3	64.5	57.4	37.6	34.8	54
	12	137.1	118.3	-66	-105.3	-13.2	53.5	-303.6	210.2	17.7	-158.4	78.6	-184.1
	24	-325.6											
37	0	-62.5	38.7	133.8	-880.4	0	-17.8	-2.2	74.4	-50.7	-59.8	37.4	306.9
	12	-128.8	85.6	-26.8	3952.5	-44	56.8	-188.3	-149.1	-16.3	-23.2	-37.3	170.2
	24	-201.9											
38	0	-207.3	-16.7	99.5	355	95.4	-100.5	-129.8	56.9	2.6	-205.5	-52.9	-4992.6
	12	23.7	6.4	94.3	-107.7	13.6	-24.3	93.3	99.6	105.5	-35.4	205.4	-423.5
	24	101											
39	0	-49.7	237.2	63.2	-38.4	1156.7	49.6	165.3	-29.1	-17.1	-138.6	12.8	-160.1
	12	-102.8	-68.6	84.8	-253.1	121.8	-64.4	6662.5	38.2	-2.9	-61.9	4.7	63.9
	24	48.4											
40	0	-1.4	123	25.3	98	83	-3.3	125.4	6	-79.2	9567.4	-72.7	6.8
	12	-32	16.9	15.2	-2.8	144.6	32.1	-39.2	-59.2	-69.1	-255.2	150.5	91.2
	24	-448.6											
41	0	-4117.2	14	-3114.7	-40.8	-25.9	-58.2	-79.1	37.3	34.1	-36.6	106.1	31.7
	12	-40.2	47.9	183.8	-49.1	-189.4	-61	-114.6	82	94.6	-14.3	105.2	43.8
	24	22.4											
42	0	-61.3	74.9	32.4	-68.4	-15925	99.2	48.2	166.1	2.3	-55.3	58.9	-73.5
	12	53.6	26	103.7	-246.3	-60.7	-69.3	56.5	-86.7	-24.9	-118.8	-53.3	-64.9
	24	-436.6											
43	0	93.1	21.8	10337.4	-48.2	-56.9	-8.8	12.3	-51.1	65.9	-97.1	-240.2	-103.6
	12	65.6	-136.5	105.3	99.2	-27.3	193.6	76.3	-274.1	-34.9	175.3	33.9	-32.3
	24	61.8											
44	0	-2880.5	112.6	-0.3	-136.9	-248.1	-17.1	-47.9	-2812.4	47	-145.3	-167.4	-175.7
	12	-10.7	16.3	-110.7	-127.2	116.2	20.1	-70.7	-166.3	51.1	185.7	-110.8	53.3
	24	-40.1											
45	0	246.4	84.9	-15.4	-5508.5	118	-11.5	25.2	44.4	-110.4	27.9	-21	221.3
	12	58.4	-136.6	-107.8	-75.2	-82.9	78.1	-41.6	195.1	29.5	473.5	-32.1	9.6
	24	-335.1											
46	0	18.2	6081.2	-69	-119.9	23	71.6	-25.9	87.4	66.9	52.4	1.1	-91.1
	12	0.6	-6.4	-106.6	-99.1	-34.4	58.5	-73.7	-390.2	51.9	-0.7	-84	-16.6
	24	-10.1											
47	0	39.2	-55.4	5701.5	26.8	-4.1	38.1	-13.9	-50.9	24.9	-111	-41.8	-153.8
	12	-4.9	-109.9	202.5	156.4	102.5	19.9	-86.4	-238.7	-5.5	-92.8	-40.1	120.2

	24	26.8											
48	0	137.2	-34.3	-30.6	28.2	-6.7	-1924.7	87.3	-44.8	11.9	-0.2	32.6	8.5
	12	96.4	40.4	22.5	-19.8	-29.8	-36.3	-52.5	-9.8	-92	-138.5	-24.6	91.3
	24	-132.9											
49	0	-2682	-61.1	-84.5	-192.6	2423.2	4.7	-145.8	-16.8	11.4	-10.2	-200.5	34.2
	12	-126.6	2.3	162.4	-72.7	141.2	87	-61.3	-10.8	34.8	42.1	89.9	-253.6
	24	187.9											
50	0	4905.3	3536.6	-49.3	-8.8	-121.6	29.7	4.8	-139.2	13.8	77.8	-23.1	42.8
	12	109.4	-59.1	80.1	214.9	18.9	63.9	14.8	-94.3	187.5	-271.8	135.5	-144.8
	24	69.7											
51	0	-7.2	-12.7	-45.8	100.4	-16	-4897.7	101.4	-11.5	-22.9	-11.9	7.9	-6.4
	12	192.3	167.5	-46.8	-52.6	-132.3	115	63	-67.2	-28.3	810.1	23.7	166.5
	24	-100.7											
52	0	-223.1	-47.7	1672.5	-210.1	72.8	-49.8	102.6	207.2	16	-52.7	2788.2	-23.8
	12	183.1	-22	-37.5	24.7	109.7	117.5	111	37	-33.5	-4168.7	-83.1	-29.7
	24	-984.3											
53	0	66.2	-14.5	-103.1	-109.4	19.6	41.6	-4.2	-146.9	195	3840.1	69.4	23.2
	12	-5	18	103.8	-17.3	-15.6	-56	55.9	-33	25.6	-149.2	75.7	29.1
	24	315.3											
54	0	-5.2	135.4	134.7	-38.2	33.6	-9.5	-2279.6	-46.2	-66.5	58.1	-17.8	34.6
	12	118.3	-3.5	-176.1	9.6	9.4	79.3	34.9	57.4	261.7	-26	47	-41.8
	24	864.1											
55	0	83	-2952.5	46.2	150	-73.8	-14.3	-2.7	7.8	-80.7	19.7	-83.5	103.5
	12	-12.7	-90.2	-42.6	33.4	-3202.8	-43	161	749.3	-320.4	-143.4	-222.4	254.2
	24	232.8											
56	0	1688.5	-8.8	-44.8	-2048	52.5	27.4	78.1	-29.4	-47.2	73.5	-13	68.9
	12	51.4	63.5	20.4	-10.9	-26.7	51.9	-18	-10.4	62.3	65.9	14.2	-22.4
	24	9656.5											
57	0	-102.3	-154.3	-1172.3	124.8	-508.1	2456.3	21	2157.9	-82.1	-1221.5	-40.4	182
	12	114	95	36.3	-98.1	-2.5	-56.5	-56.3	-48.2	138	198	-118.1	225
	24	33.6											
58	0	-136.6	182	2910.7	127.9	-81.6	-6.1	63.6	40.6	-15.7	-40.4	-140.3	-57.6
	12	-21.7	-44.6	11.1	-64.6	-32.5	96.3	16	74.9	31	125.2	28.4	64.3
	24	-110.1											
59	0	3569.4	-32.2	21.5	175.4	99.4	93.9	30.9	4.6	27.7	14.4	18	21
	12	67.7	6.8	-45.1	-14.6	64.7	28.7	111.5	131.7	80.5	-119.2	-93.4	-103.3
	24	-68.1											
60	0	221.5	-2896.5	-78.6	93.4	3070.3	-63.9	-28	-121.4	-92.5	-135.9	29.1	-198.3
	12	-17.7	-10.3	-18.7	15.8	123.4	-136.3	28	-254	-67.9	-164.4	-46	-22.3
	24	854.6											
61	0	171.2	377.7	258	-48.9	-2050.3	-19.9	143.2	-767	48.4	-61	-43.6	-55.1
	12	-2587	-45.9	-35.4	94.7	70.5	129	-222.5	89.5	41.8	85.4	-58.3	100.6
	24	2427.1											
62	0	-64.1	1168.3	37.5	-22.4	-15.1	-36	-17	159.6	2690.9	88.6	-72.2	23
	12	106.6	191.1	72.4	59.7	-67.2	64.2	69.4	-130	52.4	-82.4	43.1	-41.6
	24	-693.1											
63	0	-168.7	2966.3	66.9	115.7	68.3	-2589.6	-51.9	72.2	128.9	-223.1	-76.7	-143.3
	12	-24.6	-168.6	-109	59.8	27.1	-42.5	-21.1	143.4	-209.7	203.2	305.8	-141.3
	24	413.6											

Table C. 15 Shape codebook 0 for MODE_VQ == 08_06, BLEN_TYPE ==SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	339.7	-841.5	-5098.1	-888	343.9	27.1	94.3	210.7	-350.9	706	-46.9	472.6
	12	349.2	-219.9	-541.4	-253	-553.9	53	518.9	-54.6	81.4	196	-269.4	225.5
	24	-399.2	-342.8										
1	0	774.6	-100.2	2345.1	-806.2	252.9	-156.9	167.8	399.3	-2521.9	97.4	84.2	54.7
	12	712.3	-485.5	523.9	209.2	-304.2	718.6	-4785	114.7	57.9	-41.4	-42.9	-26.8
	24	-94.7	56										
2	0	34.9	-287.1	461.2	164.7	-261.3	-110.7	-70.9	374.5	5766.2	-20.6	-164.8	215.2
	12	420.4	344.4	435.4	-224.2	245.9	181.2	156	31.7	825.9	145.2	232.2	323.2
	24	344.9	448.4										
3	0	29.2	-418.3	-1053.4	-2938.9	867.5	-989.1	-689.3	16.3	1859.2	-170.9	-1914	113.3
	12	-1206.8	-913.7	-180.8	298.9	692.6	720.6	199.9	213.7	713.6	-1210.8	-366.1	317.3
	24	-360.3	-614.7										
4	0	-78.5	2375.3	427	265.6	-929.4	3.1	820.7	-1888.1	480.2	72.4	1712.9	-167.5
	12	-230.9	1055.1	481.2	-219.2	-67.7	122.6	-352	-242	4365.6	-661.9	-212.8	151.6
	24	155.4	-83.8										
5	0	-206.1	43.3	-436.3	-137.5	169	-5403.8	51.4	-405.8	84.7	-86.3	16.1	-156.1
	12	-12.2	32.5	-300.3	134.1	12.9	243.9	508.1	-363.8	266.7	-33.9	149.9	-520.5
	24	468.5	-13.9										
6	0	-206.2	-1436.6	465.5	459.9	-1704.5	1488.1	576	-122.9	2757.8	-16.3	-211.5	-670.5
	12	122.7	-289.1	-247.7	592	-3342.3	-542.6	-117.5	-414.3	587.8	-44.4	-588	95.6
	24	164.7	386.4										
7	0	-222.9	2421.2	187.2	79.8	1206.1	361.6	-717.5	-33.6	3436.7	-261.8	147.9	-544.8
	12	128.8	-369.1	-133.9	-863.7	626.8	-273.7	-241.6	1416.4	-2164.8	30.1	-737.8	652.2
	24	-390.6	-423.8										
8	0	-275.3	-562.4	-853.8	69.5	68.3	100.1	-547.7	-37.6	-803.5	470.5	-209	-4924.5
	12	-378	343.3	-556.6	41.5	188.5	-246.4	104	-10.5	422.8	-826.2	-83.9	-65.9
	24	582.9	-1464.4										
9	0	530.3	165.1	-212.6	-5560.3	391.3	-218.6	-55.3	200.4	-515.2	-147	325.2	-258.5
	12	178.5	215.2	35.6	111.1	480.2	670.6	28.5	-113.7	197.9	607.5	25.6	503
	24	-228.7	20.9										
10	0	1136.1	-179.2	88.9	842.1	-495.4	-5.5	220.3	175.2	262.4	463	-300.9	-1121.4
	12	-297.4	749.5	292.8	504.7	5114	20.6	-206.2	-314	-232.8	-44.4	-344.6	166.8
	24	295.7	-500.5										
11	0	2441.4	1452.5	-2237.5	488.6	219	-994.6	-299	331.9	32.1	-57.5	181.1	425.8
	12	-257.5	-807.9	352.6	-1726.2	-649.6	-2214.1	645.2	519.5	259.9	-213.2	490.6	294.1
	24	650.7	-3352.9										
12	0	-114	-594	938.5	1622.5	492.3	-275.3	1801.3	-1526	-1590.6	-2129.3	485.2	443.7
	12	102.2	2104	-162.7	543.8	224.7	-845.9	-351.4	-354.4	82.9	198.6	-3162.2	133.6
	24	-132.7	586.8										
13	0	254.1	2483.4	-897.7	92.8	390.3	-280.9	1889.5	222.7	-390.9	-2000.7	-1047.4	493.1
	12	-791.4	-44.5	-179.4	58.3	208.8	271.8	-763.1	-134.4	-55.8	-247	-315.6	-4827.5

	24	-161.8	610.5										
14	0	253.3	1199.4	2062.5	143.9	-438.4	1780.3	1526.7	-644.7	-139	7	-1090.4	-693.2
	12	254.9	754.8	-2415.6	-719.5	152.4	770	335.7	-632.6	519	13.1	468.2	510.5
	24	450.6	-672.5										
15	0	2644.7	446.1	-23.9	2701	994.1	220.8	79	-665	128.7	-773.1	-1533.9	126.1
	12	-582.3	71.3	642.1	437.8	213.3	474.3	706.4	-39.7	-775.1	48.9	-381	705
	24	179.8	205.6										
16	0	-188.4	5918.5	-781.5	-67.6	294.6	634.1	-225.4	7.1	-415.5	-98.3	-96.7	288.3
	12	-184.4	-333.7	518.5	-51.1	175.7	391.6	513.6	67.8	-484.2	-911.7	317.1	121.1
	24	700.9	642.7										
17	0	-104.3	-1085	45.1	1256.3	113.4	-62.1	-674.6	-1103.7	-590.9	-792.4	-796.3	-1203.3
	12	690.1	-56.3	3852.4	-899.3	-193.7	-129.1	-346.7	94.2	918.1	727.6	-18.8	-7.1
	24	765.6	555.6										
18	0	124	-1300.5	1775.5	-1110.2	741.3	796.5	-1.4	-64.7	335.5	-98.5	272.2	123.1
	12	-149.4	-4003.6	56.6	-123	-113.1	62.4	395.4	444.6	518.5	-178.9	524.6	412.1
	24	456	-668.7										
19	0	-684.9	-757.9	-314.8	-99.4	-1092.7	17.5	95.8	197.9	365.3	248.8	-462.5	94.5
	12	-5040	-63.3	-43.7	174.4	129.4	-465.1	-54.4	-274.4	270.6	1067.6	108.4	5.9
	24	-156.5	-111.9										
20	0	-158.6	-452.6	15.6	136.1	-36.8	187.4	5496.2	23.8	37.9	-93.6	-7.2	156.3
	12	240.8	121.7	69.5	103	62.1	-298.8	-133.3	135.8	6.9	175.8	-141.8	-189.5
	24	353	145.3										
21	0	724.9	228.8	-1577.3	-1183	-2040.1	-227.9	1049.5	2732.7	-837.1	354.9	129.5	-104.7
	12	-241.2	679.8	654.9	-145.6	1430.4	-584.4	-231.9	910.3	679.1	46.9	-412	-253
	24	3084	714.9										
22	0	104	-1425	539.3	93.5	291.6	-1363.7	744.2	28.4	-25	-29.9	121.9	449.4
	12	-308.5	-145.8	-339.9	648	-82.3	-643.6	-104	-5020.9	-330.2	-361.4	-510.6	303
	24	-58.6	-148.1										
23	0	-3152.5	-1935	-1208.5	818.3	-247.5	389.1	-945.6	-159	-319.8	71.5	-774.9	1121.4
	12	194.6	789.1	1354.8	465	341.3	266.7	-154	420.6	901.7	-407.7	381.9	-693.8
	24	-906.6	-88										
24	0	-3.4	162.9	104.6	287.5	-794.5	-86.7	-2.4	311.3	-412.7	-5177.9	-21.8	-258.6
	12	64.2	-283.2	2.6	88.5	264.4	-23.2	502.7	889.2	394.6	38.9	-203.8	-223.9
	24	3.8	239.1										
25	0	-31	3170.1	-58.7	2751.6	35	-1087.5	-864	-53.1	-789.9	-206.8	34.3	-365.9
	12	-235.1	338.9	-1051.4	-837.9	-1238.8	609.6	232.3	44.3	-46.3	289.9	-762.5	706.1
	24	44.7	-122.4										
26	0	-80.2	-2181.3	-1172.9	-177.7	300	275.5	711.2	466.3	563.8	-3754	95.3	367.9
	12	116.9	47.3	777.9	-657.3	646.8	-182.5	-533.8	-796.5	-165.2	-28.5	58.5	-112.3
	24	190.5	-492.3										
27	0	-2729.4	357.6	1148.1	32.6	1953.5	599.2	-1308.4	44.6	-538.7	428	-918	-255.7
	12	33.8	1076.4	729.6	-27.2	259.6	-26.4	511	1853.1	-484.3	661.1	-400.9	262
	24	-214.6	37.2										
28	0	-164.9	-373	-699.2	-10.9	-2598.4	-2393.5	83.5	462.2	-1093.3	159.9	524.4	-1108
	12	422.5	-123.1	-117.5	-242.1	1273.2	3739.6	691.4	315.1	175.1	-160.3	399.2	-123.4
	24	-254.3	-43										
29	0	-272.6	-31.6	650	-365.7	-325.7	837.1	802.1	-328.7	-1027.3	1297.6	-368.1	163.9
	12	-231.5	173.8	459	5150.9	208.6	93.7	258.2	245.4	39.3	288.8	1118.3	371.5
	24	36.9	85.4										
30	0	68.5	-787.5	164.6	223.2	116.7	315.4	185.8	1081.3	-164.9	21.8	5334.7	-221.8
	12	-177.6	129.1	420.8	78.7	-354.2	197.4	330.3	-54	169.8	-186.1	-595.9	-177.4
	24	76.5	-391.8										
31	0	48	-1052.5	625.4	162.6	-351.1	-118.6	2000.6	-1590.4	1825.5	1358.2	-527.6	-165.8
	12	-893.8	-1134.7	864.1	-2400.5	148.9	1071	310.2	1148.9	196.1	126.4	-3.3	383.6
	24	-694.6	240.1										
32	0	-113	-951.1	143.2	-529.5	2736.9	-2550.4	432.3	964	-322.5	-578.8	-530	-296.1
	12	490.7	1957.2	-524.4	-16.8	-1618.5	-543.5	-238.1	160.2	-115.9	334.4	-901.2	978.2
	24	681.7	241.5										
33	0	-141.5	51.7	-3275	-321.8	-1121.8	471.2	-1546.6	-1635.9	-1551.1	270.4	-1835.8	-251.1
	12	843.4	181.1	35.8	184.3	-673.2	-1079.6	180.4	-122.6	635.5	-103.4	-67.4	648
	24	466.2	769.7										
34	0	277.7	-1491.6	2876.5	1234.2	-538.7	-2628.6	-287.7	-544.8	-215.4	289	414.8	-811.6
	12	1213.4	-11.5	617.4	-192	-218.1	883.3	498.7	-434.7	498.1	-527.5	128.2	-700.1
	24	194.4	242.2										
35	0	1059.6	435.6	2287.8	-1274.1	-714.6	-891.2	-1553.1	-2407.7	-501	94.9	-370.4	271
	12	1186.1	162.5	-104.5	-433.1	1537	-1288.6	602.2	955.9	-338.1	231.5	-426.3	-1229.7
	24	-358.2	61.7										
36	0	-465.4	308.2	-3184	239.6	2305.3	-93	258	-927.3	-382	-218.4	1087.1	-589.8
	12	793.5	-994.8	830.5	-1026.8	-373.5	784.3	-213.7	-310.9	1180.6	-831.2	465.4	-774.7
	24	254.6	1272.1										
37	0	-3150.7	2387.2	-397.6	52.6	-261.8	462.6	190	822.6	170.1	1030.3	-70.4	-856.3
	12	-1325.5	1564.5	-83.8	-177.3	-80.4	-426.9	566.4	122.6	-532.6	52.5	818.7	-285
	24	-1061.2	-1173.8										
38	0	-220	582.3	1560.6	-1433.5	-902.5	-1696.3	-19.1	1206.8	374.2	-565	-365.5	-3467.2
	12	56.4	-130.2	-54.7	1171.1	-1072.7	-115.6	351.7	-334.9	-399.9	374	-333.4	-209.2
	24	209.4	-577.5										
39	0	-85.3	517.6	663.3	-367.9	-1220.3	-370	-802.3	489.6	60.4	117.3	59.8	-519.7
	12	-62.6	-251.7	-55.9	119.9	383.3	-5662.1	200.3	11.6	130.7	154.2	-330.9	-96.9
	24	536.7	-126.7										
40	0	-47.5	-864.6	-116.5	242	-5397.5	114.4	28.3	364.1	498.1	-103.6	-198.1	221.1
	12	-129.8	-80.8	258.3	651.5	557.5	-481.2	309.7	180	185.7	-119.6	-255.1	-53.8
	24	-246.5	12.9										
41	0	6353.3	41.9	264.9	86.7	593.7	85.2	152.9	-48.4	82.9	147	-492.5	152
	12	-145.7	-213.8	48.3	-2	-43	488.3	904.7	59.3	-24.4	-46	446.4	-469.9
	24	770	194										
42	0	-159.1	-959.6	-609.8	622.5	-1463.2	-1912.4	184.6	-3844.2	102.1	-345.3	-1164.2	190.1
	12	-405.8	-813.2	-556.6	272.7	-233	1215.8	728.9	190.8	-553	110.4	-171.3	1380.7
	24	361.9	78.6										
43	0	-161.6	708.6	-888.1	623.4	-439.1	1109.2	2676.6	-139.1	-622.2	2783.4	-832.3	591.4
	12	694.4	-20.6	-225.5	417.2	1763.9	139.5	-666	1244.3	-2391.3	165.6	-2120.1	204.8
	24	258.8	27.3										
44	0	-263.7	3509	2638.9	565.6	1617.9	78	945.5	-763.1	62.8	280.9	-438.2	155.8
	12	518.2	-250.6	-85	-122.8	-789.2	734.3	-539.2	-328	-477.8	56	-1391.1	189.5
	24	-978.6	1207.3										
45	0	-345.7	-2392.4	1182.7	630.1	1955.3	917.6	-334.9	653.3	-2152.6	-198.8	973.7	-701.5
	12	-173	-312.8	-730.4	-660.4	83.2	688.9	-162.4	396.3	-872.8	-672.9	1977.1	-764
	24	-157.4	-480										
46	0	-545.7	956.5	-1982.7	-2187.3	-992.3	-1286.2	-220.5	-896.8	-742.7	-2595.4	468.5	-277.3
	12	-904.9	113.2	-103.7	741.4	-585.3	266.7	-695.5	150.2	-1393	1706.2	-355.4	74.7

	24	301.5	245.6										
47	0	-3129	294.3	-578.3	6.5	99.3	-620.2	1187.6	-1799.1	-26.7	567.9	858.2	109.7
	12	12.7	-1769.2	-384.5	379.8	134.3	-1423.2	2293.4	-353.9	211.8	776.6	228.5	72.3
	24	-12.6	-648.4										
48	0	-149.2	-772.1	-203.9	-1636.9	-152.4	1082.4	-411.5	279.8	-2319.1	-1445.4	131.2	1343.1
	12	-2745.8	-335.2	1255.5	-606.2	-209.9	302.2	544.3	582.5	560.1	-1254.3	590.2	1034.2
	24	458.8	534										
49	0	166.1	-140.4	-732	2371.2	3091.8	-13.9	-1436.7	897.7	412.9	482.6	233.9	-363.5
	12	-78.7	708.8	398.4	647.2	355.9	1267.5	-227.7	-64.3	379.3	-38.4	984.2	494.1
	24	49.5	-194.7										
50	0	3004.8	-68.5	-775.2	-583.4	215.2	1036.6	939.3	-1267.8	-1711.2	1166.6	169.9	-621.5
	12	-93.7	-1015.5	236	-511.1	605.6	-490.4	-1260.7	-344	758	1656	542.2	-814.1
	24	-924.6	1154										
51	0	287.3	2509.8	294.8	1562.6	1906.6	549.1	1110.3	554.3	-146.2	863.2	600.1	519.7
	12	-2485.6	-1287.4	-59.5	566.7	420.1	-813.4	22.8	-240.8	333.6	-263.9	332	-0.3
	24	-90.5	252.8										
52	0	-270.5	-2324.6	-1175.9	-1368.6	3218.6	324.3	1033.4	-191.1	574.2	373.7	243.6	602.5
	12	429.4	334.9	700.1	-499.4	72.7	449.3	1590.4	-886.3	131.1	433.5	-203.1	-945.6
	24	81.9	-147.4										
53	0	-66.6	-1137.7	-373.5	2857.4	-958.3	411.9	-1729.2	1132.1	-1907.6	635	-76.7	-809.7
	12	487.8	-966	-1064.7	-413.1	484.4	-424.7	-121.3	282.3	386	-7.3	-1428.4	-407.1
	24	-665.2	1461.8										
54	0	1512.6	-1120.9	472.7	-1319.3	2203.7	2.9	-909	-1019	231.3	817.4	1231.1	-1809.5
	12	-1944.4	-68.5	-138.9	496.6	-202.4	-207.4	-74.3	-54.1	1126.5	-802	-585.5	150.8
	24	656.3	-105.2										
55	0	-88.9	1614.4	-486.5	-962.3	1351.3	165.6	-482.5	-3301.8	-641.9	103.9	172.3	-2.9
	12	107.6	-274.2	-312.1	1576.8	-576.1	83.5	1537.6	1338.3	568.4	288.2	-802.1	-765.2
	24	-342.8	-285										
56	0	32	-444	1798.7	1300.4	420.7	-1272.7	199	1946.4	-398.8	502.3	-2292.3	824.9
	12	145.9	368.8	1063.2	-2153.1	-1020.2	-294.9	2802.4	-294.6	428.3	-634	-871.6	189.1
	24	-62.6	-223.4										
57	0	-502.7	838.6	-1693.4	-948.4	420.4	-419.1	2043.6	377.3	-415.3	-89.9	-920.4	-2115.9
	12	1057.9	120.4	1712	-77.3	-18.5	-527.6	1344	-91.7	-126.5	234.4	367.6	867
	24	-3676.7	2343.4										
58	0	-31	-230.1	1458.2	-1232.7	1078.8	-204.6	2792.9	1024.1	-319.9	513.1	-641.7	-486.4
	12	349.2	1881.7	-68.6	18.9	0.3	-68.2	-466.1	171.9	370.9	4459.3	75.1	129.5
	24	-106.9	-52.5										
59	0	-473.6	-393.4	370.8	-294.7	83	2415	-463.6	1007.5	-327.8	-1324.5	-3338.9	-1279.7
	12	447.4	309.4	-11.7	379.5	390.3	-239.3	-604.5	-463.2	557.4	143.5	135.2	-672.4
	24	-624.4	198.4										
60	0	13.8	-1748.9	1036	363.9	23.8	2020.4	697.4	-1788.1	142.8	-791.6	707.9	-348.1
	12	-2571.8	2368.5	198.9	-16.8	150.9	822.5	-750.6	71.7	-879.7	-1276.1	856.5	151.4
	24	313.2	167.9										
61	0	78.8	1990.8	1920.8	60.8	616.3	-2498	187.4	-230.4	198.3	-188.7	630.6	207.1
	12	294	780.4	2427.9	245.9	512.2	-310.2	-983.7	-684.7	-1349.8	-619.2	-244.2	1022.1
	24	246.3	148.2										
62	0	2680.7	-717.2	-1468.3	-418.9	1351.7	298.2	140.9	275.1	2849.4	-54.7	-215.4	949.3
	12	401.8	686.6	236.6	-237.3	-732.2	73.9	-4.4	-434.7	-226.4	556.2	-85.6	-451
	24	953.4	-212.3										
63	0	-378.2	2946.9	899.6	-2873.7	318.1	850.7	-442.3	526.1	-557.5	118.5	-1084.8	-71.6
	12	556.3	-83.7	27.8	-16.8	-118.8	-767.3	-932.1	-58.6	607.8	-651.9	-0.2	-234.5
	24	711.8	106.3										

Table C. 16 Shape codebook 1 for MODE_VQ == 08_06, BLEN_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-3998.1	842.7	1612.8	-818.1	-179.1	-594.8	-40.7	-494.6	206.4	679.9	212	1311
	12	-556.8	313.6	1359.2	872.6	-168.5	327.7	-321.8	-934.1	-175.3	230.2	-83.5	465.1
	24	859.2	1489.8										
1	0	-1013.6	726.1	463.1	442.9	-1072.5	489	-1338.7	-564.5	339.7	-42.4	-259.6	-467.7
	12	40.9	389.5	-313.2	-414.4	-11.2	-955.1	-5468.4	-51	-391.1	239	34.1	-359
	24	721.7	110.9										
2	0	1158.3	-598.5	270.2	-1813.7	-1377.6	2964.4	-796.8	-683.9	526.3	-175.1	777.6	-1008.5
	12	1756.7	627	120.1	-487.5	486.5	-413.7	-299.6	-248.1	-180.3	-182.5	-500	-613.2
	24	-368.5	-164.1										
3	0	-157.5	-7550.7	-610.7	381.7	387.4	434.4	-77.4	79.8	-118.3	47.4	101.8	360.2
	12	175.5	-367.3	31	252.1	26.3	-57.6	143	544.7	-83.5	-450.2	-1059.7	186.9
	24	-71.6	1022.9										
4	0	23.1	2338.4	186.8	-781.8	81.3	1185.7	3094.8	288.4	1079.1	-183.3	280.3	54.8
	12	875.5	-1693.8	293.1	291.6	-316.7	303.6	1236.7	-256.3	107.1	-720.8	136.8	115.7
	24	-858.2	-699.6										
5	0	2989.2	-289	1101.7	995.6	1344.8	-1870	-68.6	-347.4	-354	582.8	809.4	4.3
	12	-281.2	248.2	-525.5	398.1	-402.8	-459.9	2016.8	1692.9	261	-31.5	641.1	-329.7
	24	-563.1	-156.5										
6	0	362	1574.2	-3.6	-1202.9	323.4	584.7	-438.7	557	-274.1	-3180.5	454.4	428.8
	12	1991.4	6.2	-675.1	-60	-367.8	-193.7	507.7	151.7	69.6	530.9	-1984.7	147.3
	24	659.4	-199.1										
7	0	777.3	1920.7	-1554.4	-465.5	2177.5	-2342.9	55.2	-351	401.3	333.7	29.7	-924.8
	12	0.9	-569.6	-181	658.6	-2601.6	125.5	-1211.4	12	-506.9	-96.2	-34.1	230.8
	24	-161	223.5										
8	0	143.6	-850.1	122.4	-194.4	1567.6	1268.9	404.4	932.7	2255	136.6	391	-2843.3
	12	956.3	214.5	366.4	274.3	-435.3	168	2206.5	301.4	-209.3	900.1	153.5	-309.8
	24	488.8	723.8										
9	0	-211.3	-3572.3	2669.7	-966.7	279.6	488.5	-1044.4	185.1	-269.8	-179.8	-311	-465.6
	12	-367.4	170.8	327.7	115.9	-42.6	535.7	1017.3	-623.6	-818.2	-649.6	-245.2	-57.4
	24	28.3	326.1										
10	0	-111.1	-1904.1	429.3	-1051.7	-223.8	446.2	222.7	3806.1	-34.5	303.4	462.1	-144
	12	-137.6	-677.1	-365.6	369.2	-214.4	1008.3	-664.7	532.9	-16.9	-274.9	351.1	245.9
	24	1224.7	414.1										
11	0	-4585.8	-347.4	-486.6	531.9	825.9	509.7	570.9	1040.3	-48.2	432	-360.6	127.6
	12	-122.9	-1436.4	661.4	89.6	319.7	763.8	234.7	583.4	-285.4	1244.6	-806.7	-656.9
	24	413.3	259.6										
12	0	86.9	-251	-1477.3	-392.2	-231.5	49.2	109.8	670.1	1243.1	140.4	3148.9	374.4
	12	322.6	337.6	137.3	328.7	1886.3	-2357.7	1032.3	85.8	139	434.4	890.9	122.2

	24	-1100	107.4										
13	0	284.7	-2272.2	572.3	381.1	127.2	-1365.6	-1028.4	-389.4	3082.1	296.7	157.5	445.3
	12	-255.9	-245	53.9	-303.5	-244.3	-2080.4	-123.8	690	-386.8	-559.8	979.7	48.8
	24	86.9	214.7										
14	0	205.3	-648.4	-431.6	88.8	562.1	745.2	147	-228.8	-118.3	-323.9	103.6	154
	12	-12.6	5037.5	-166.3	-168.5	41	-35.1	225.4	277.8	387.6	161.2	610.9	397.8
	24	-95.7	511.4										
15	0	-14.6	-197.3	6029	-378.1	-297.3	73.4	-394.5	280.8	-405.9	464.7	21.2	269.7
	12	210.7	-164.8	-349.3	-179.5	250.9	-184	-49.1	295.7	559.2	457.9	-115.6	-125
	24	420	-428.9										
16	0	-2646	-145.4	547.9	2376.7	347.2	-997.5	-105.6	-688.4	-607.8	-1083.3	1337.2	178.7
	12	-505.5	-353.5	-1238.5	-1299.7	795.5	-578.3	705.8	-730.2	692.7	1580.6	90.6	-369.4
	24	285.6	129.2										
17	0	346.1	-295.1	-3494	2661.5	306.8	-72.1	218.6	596.4	-140.5	1296.1	503.7	45.1
	12	313.8	889.7	594.7	-687.2	1262.1	185.1	52.1	592.3	712.1	-283	41.5	-594.1
	24	-200.8	-222.4										
18	0	91.9	-1265.6	-822.6	-126.9	-1099.2	-1010.1	678.6	1119	108.9	-324.8	-3859.4	-347.8
	12	-413.4	-772.7	124.5	-44.2	-399.5	-201.7	955.2	61.5	175.1	153.9	250.5	493.6
	24	298.3	-1506.3										
19	0	-2.5	297.6	-2597.1	534	-693.6	232.5	556.2	-685.4	-647.5	-1957.2	387.7	389.9
	12	-1076.2	-3664	-156.4	227	42.4	-804.8	-434.1	317.8	478	-141.1	165.1	-510.9
	24	31.5	561.1										
20	0	234.9	445.9	-608	-2239.2	321.1	-932.6	-450.3	157.8	48.2	91.7	-1154.6	3292.7
	12	3.6	260.4	-159.7	794.2	380.9	211.6	-64.1	354.5	305.8	-68	243.7	352.7
	24	63	-746.4										
21	0	-84.5	-114.2	496.3	-4554.1	184.8	96.6	-87.1	446	161.1	-18.3	-185.6	-270
	12	-270.4	-85.5	-228.2	-873	481.8	-1149.7	-538.7	-135.4	-604.8	287.1	153.2	71.6
	24	-662.5	446.2										
22	0	-710.4	67.6	620.4	-1071.1	362.3	-1778	1044.5	1534.5	-845.3	2742.1	18	-623.4
	12	-443.7	-1179.8	-424.5	155.6	-351.5	-503.3	-489.3	-229	-472.9	46.4	-332.6	-850.1
	24	-126	-775.4										
23	0	-486.3	-236.2	94.2	313.2	463.2	-248.4	225.8	-456.3	772.5	-334.2	355.6	165.2
	12	-65.7	292.5	-4874.1	85.3	175.8	136.2	-459.2	168.6	21.4	324.7	-351.2	-171.6
	24	160.2	406.7										
24	0	696	1457.4	606.4	325.5	129.8	-373.4	224.4	-427.1	392.6	78.4	-106	-4584.3
	12	-390.6	-443.6	266.8	-11.9	686.7	640.4	-391.6	210	-237.1	150.3	-1459.4	175.7
	24	-204.9	349.3										
25	0	-512.3	104.7	-128.7	176.1	869.6	4201.8	200.5	-254.9	-683	52.7	502.9	104.8
	12	-479.6	307	-998.1	293	-355.1	-39.9	781.3	-657.5	524.1	-231.8	-110.8	-231.5
	24	-142.8	-133.9										
26	0	-27.9	-823.1	794.7	-247.2	4214.1	-312.8	158.7	477.1	483.6	517.8	-440.3	484
	12	-224.2	-601.2	454.5	957.1	453.3	-307.4	-476.7	40.8	275	-2775.2	-478.1	-430.6
	24	-60.4	3										
27	0	-728.8	2146.1	671.1	292.3	-197	-224.5	237.7	2949.4	547.9	-255.7	-416	746.2
	12	-110.9	-184.4	104.4	105	2197.5	1194.7	-79.3	160.8	1148	-293.8	-2817.3	-498.4
	24	69.5	-324.4										
28	0	312.8	-938.2	-2559.1	324.5	133.7	-1183.3	-447.2	317.5	-2123.7	-160.9	-141.1	-529.9
	12	-2458.4	288.9	-890.9	-1500.7	851.1	409.5	-152.8	-581.9	-1898.1	-490.4	14.7	445.1
	24	341.8	-139.8										
29	0	3979	-2169.2	-9.3	-959.4	-589.9	-713.8	423	248.9	-129.7	493.4	-61.1	-139.3
	12	-1046.8	135	723.6	-91.1	1100.1	494.9	68.1	113.4	-892.2	666.3	-1227	30.6
	24	372.5	-758.1										
30	0	-1980.7	-636	210.5	881.6	-1262.3	-239.9	213.3	124.9	99.2	1004.7	-476.3	-264.5
	12	3651.5	-229.8	-121.1	-263.7	80.2	1167.5	-858.9	16.7	41.4	297.2	1988.2	715.7
	24	-163.3	-220.6										
31	0	-96.2	290.5	-515.9	158.6	244.1	-2255.3	2159.8	-1142	-1593.4	-395.9	580.2	801.9
	12	1767.9	719.1	559.3	1682.2	102.2	-267.8	55	-72.3	-619.2	-194.9	-387.9	134.7
	24	-716.5	-528.7										
32	0	-2378.8	-249.8	-1982.2	-1589.5	1541	402.3	-266.8	-215.5	-901.8	-290.3	-323.2	314.3
	12	-171.2	410.5	411.3	824.5	623.4	148.4	571.1	-290.3	180	-705	-221.7	35.6
	24	15.8	-5003.7										
33	0	1289.8	316.8	656.7	-191.3	-522.4	-975.6	3765.3	137	271.9	97.8	-331	241.5
	12	-738	594.9	-419.8	-843.7	377.9	594.3	51.1	1303.1	-1575.5	772.3	197.8	-620.7
	24	1297.9	979.3										
34	0	-58.5	-2722.8	576	760.8	687.5	-518.1	3021.5	1.7	-61.3	-461.3	132.7	-0.5
	12	624.6	-899.4	-246.7	353.4	107.7	1299.8	1182.2	135.9	831.8	309.1	-292	493.6
	24	-515.9	520.2										
35	0	-100.6	4026.8	80.5	223.5	-272.7	-139.1	-361.6	173.2	126.2	-631.2	2452.9	-620.2
	12	528.3	-1232.3	413.5	356.3	-182.7	82.1	241.4	-560.3	-200.6	-369.4	219.3	125.9
	24	587	-278.8										
36	0	87.1	1885.8	-2390	-2513.1	-106.8	166.7	442.8	334.4	-488.7	127.1	690.3	-483.6
	12	-864.5	1040	1326.2	-1904.3	-216	68.7	-399.8	-40.1	503.8	-188.5	810.4	-788.7
	24	5.4	-839.7										
37	0	2990.1	2685.1	-513.6	212.4	-1146.3	313.8	-645.1	-476.2	-1043.6	32.6	-584.2	484.8
	12	622.5	551.6	868.8	71.3	-219.6	880.4	371.3	-115.9	20.2	279.7	139	682.8
	24	-1090.3	-233										
38	0	174.9	-327.8	2188.7	-2189.8	194.8	389.8	1685.2	-450.6	-804.9	174	1676.6	608.8
	12	532.7	724.3	154.1	535.1	-836.3	-23.5	-178.1	870.6	245.8	-994	600.1	488.8
	24	-278.2	195.7										
39	0	278.6	801.3	914.1	1018.8	231.8	-819.2	719	-550.6	1486.8	-3910.3	197.6	-326
	12	-147.3	26.2	439.5	1780	770.8	-114.2	114.8	-358.2	-768.7	-287.9	1253.2	337.5
	24	171.2	795.5										
40	0	143.1	934.7	-744.1	1797.4	730.8	-420.3	180.4	1802.3	-1332	-157.6	427	-482.9
	12	290.4	114	-1112.3	-581.5	-443.8	-119.1	-373.6	-30.6	-151.2	-4937.4	396.3	-127.5
	24	-19.8	541.8										
41	0	4132	1037.7	-80.7	931.4	224.2	766.5	600.2	1353.5	-23.7	149.5	946.1	435.6
	12	983	-352	-33.7	1649.7	83	-966.5	-411.8	-736.8	323.3	352.1	-27.3	387.5
	24	479.6	798										
42	0	-3.2	101	-2807.9	-646.9	-772.2	403.7	-1819.3	1696.5	659.3	-188.3	-28.7	1092.7
	12	1026.3	746.2	-848.7	460.1	-750.9	1135.1	-693.3	-477.2	-79.7	-27.8	80.2	-414
	24	-1304.1	515.7										
43	0	-87.2	1979.6	1340.3	2442.3	253.8	1788.8	-1236.3	238.4	183	-528.2	-846	-40.8
	12	-867.7	-1.3	383.7	431.4	442.1	-494.7	956.4	-445.3	-61.3	-516.5	490.3	-86.4
	24	-178.3	-243.6										
44	0	-11315.4	-29.7	-83.7	76.9	-101.9	-110.9	162.7	71.6	-113.5	141.2	110.9	-123.6
	12	-42.4	-25.9	24	-87.4	207.1	176.6	169.7	-62.8	-312	-38.4	63.8	-237.5
	24	-83.6	-60.9										
45	0	145.2	-817.5	2366.4	742.6	-361.1	814.3	121.5	1568.6	-58.3	-2465.6	-186.7	42.6
	12	-1020	-588.3	-249.4	-85.8	-1868.1	328.3	-219.3	482.6	470.2	1115	87.8	537.4

	24	-356.2	-221.8										
46	0	345.4	-364.3	102.4	-1682	-2557.6	762.7	914.7	-1544.6	-1345.9	735.2	-1018.3	-358.9
	12	-269.5	-305	-378.7	582.8	135.7	-413.8	1237.2	138.4	-271.6	21.6	-157.5	-581.4
	24	-154.1	-208.7										
47	0	-2338.4	1447	-816.6	-687.8	100.6	-1847.7	-496.3	-235.5	922.6	32.2	-420.2	-597.1
	12	742.2	-1185.1	34.8	-350.8	553.1	859.7	-273.2	630.8	-256.1	-302.8	-282.7	-433.7
	24	2902.3	3173										
48	0	440	1411.8	-603.1	-170.3	1339.6	431.1	566.1	362.3	100.5	418.4	-2837	-174.1
	12	-47.7	1570.8	-166.3	578.5	-320.6	-1363.1	-280.9	571.1	3542.4	-259.8	807.4	519.8
	24	-169.9	449.3										
49	0	-122.5	1325.1	-270.7	-167.9	-352.8	-108.9	-783.1	-2018.1	2147.3	-283.4	718.4	-287.5
	12	-2924.5	560.8	319.8	353.2	-653.7	1969.6	-977.5	39.6	55.3	-36.1	-221.3	-222.3
	24	-408.6	61.5										
50	0	130.7	440.1	2032.2	-780.5	1243.6	-783.1	-3089.6	77.4	574.2	-240.7	-3.7	975.3
	12	716.7	-998.9	167	-257.2	133.6	-960.9	629.9	284.4	57.5	51.3	384.9	-148.9
	24	-658.8	604.3										
51	0	391	167.8	-579.4	382.4	-2490.4	317.1	-558.7	204.2	365.6	2923.8	552.5	-526.1
	12	271.3	298.8	526.7	286.3	-2424.7	-627.1	15	-307.1	33.5	-310.2	233.9	-408.9
	24	221	-403										
52	0	198.1	176.1	-950.5	503.8	2141.8	1144.3	-1396.3	-425.3	-646.2	1955.5	176.9	245.6
	12	-733.6	-1052	138.9	-216.5	176.1	1116.4	-932.3	255.2	82.1	3047.7	302.7	45.9
	24	382.3	55.8										
53	0	45.7	-2692.1	-1156	1811.7	631	2126.3	-834.9	-379.6	160	-92.2	-6.5	117.6
	12	2216.9	-377.3	328	688	-120	-775.9	340.9	-429.3	-385.6	120.9	-863.4	-101.7
	24	256.2	-395.5										
54	0	11.7	-669	822.7	1259.8	-1373.7	-1793	-1675.5	1025.1	251.2	391.9	826	575.5
	12	-913.4	2260	-206.2	312.8	899.9	648.7	296.6	191.7	-40.5	-1079.4	-1141.2	-814.9
	24	-72	173.6										
55	0	-178.7	627.1	156.6	2726.6	-1921.1	-384	1763.1	487.9	1124.9	-49.9	-367.2	89.8
	12	-1006.7	-38.8	1698.2	-286.6	-645.1	207.2	-1398.6	-339.7	-836	1260.9	1216.9	-281.4
	24	-1272.7	-370.6										
56	0	-484.9	-1362.8	1243.8	-1155.8	908	-2681.4	27.4	-1269.4	-913	-40	-394.5	663.1
	12	-450	-232.3	-27	-1265.8	383.7	-188.3	-313	130.8	3156.7	-451.5	-930.4	668.1
	24	172.3	-22.8										
57	0	226.2	-199.2	766.7	764.5	141.7	309.3	-485	-178.6	-4046.6	-387.7	17.5	481.9
	12	-262.6	-264.3	758.3	-49.3	-449.1	-858.2	-141.5	163.2	-251.4	-122.2	-815.7	478.7
	24	-369.9	1205.2										
58	0	-73.2	1003.1	1153	-1883.9	-2970.3	70.6	-1416.6	1198.8	-107.4	286.9	1410.4	408
	12	-206.8	-300.3	-1.2	52.1	-673	2568.7	285	-146	207.2	920.9	214.8	-19
	24	-209.8	-548.4										
59	0	2165.4	-32.4	251.3	216.6	-293.2	-434.8	-2147.4	-868.4	-227.8	525.2	-2109.1	44
	12	872.8	-177.7	731.4	652.1	553.1	1643.1	897.9	-1625.3	-279.5	155	623	-408.9
	24	523.1	580.6										
60	0	-117.7	-479.8	294.5	-240.4	50.6	268	177.6	-5137.4	-30.5	307.8	206.8	-222.3
	12	202.1	116.2	-16.1	-251.5	-56.6	494.2	-876.6	151.6	-239	480.1	-57.4	162.1
	24	444.4	-6.5										
61	0	32.1	-1572.9	-1407.2	425.2	-474.9	268.8	1503.8	-1395.9	886.5	486.6	-992.4	1885.3
	12	-521.7	55.6	-613.5	1459.6	-437.2	-178.4	-3398.7	313.7	696.5	31.3	42.8	1150.3
	24	189.9	332.9										
62	0	133.5	1613.5	1871	-207.9	-583.6	1525	193.3	-1085.9	-330.3	720.4	-1244.5	1288.2
	12	-257.3	732.9	67.7	-3721.2	460.8	-96.1	-151.8	-618.5	-298.3	9.4	194.9	92.6
	24	58.3	929.6										
63	0	-245.9	-1906.6	-437.5	-817.8	-148.3	1.2	-230.8	-1373.4	408.7	-2150.8	-763.6	-2446.6
	12	-123.6	47.6	32.8	104.4	-404.5	-302.8	-1637.2	48.7	148.6	5.5	-88.9	-407.2
	24	-76.5	72.2										

Table C. 17 Shape codebook 0 for MODE_VQ == 08_06, BLEN_TYPE == MEDIUM

VN	FN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-4235.1	-2611.7	-40.1	93	414	235.5	579.5	343.9	-117.5	-51.2	417.4	-706.9
	12	-8.4	160.9	361.7	-256.6	-167.1	-772.7	-667.9	453.3	-306	733.7		
1	0	-186.9	-435.2	-1203.6	36.5	-650.5	-159.8	260.9	-257	648.2	844.5	34	-33.3
	12	465.9	94.5	-4940.6	-455.8	-190.4	-18.9	-225.1	42.9	-0.5	444.8		
2	0	-56	2275.6	305.4	-403.9	-245.7	286.4	-249.6	562.9	-316.9	101.4	-199.6	-331.1
	12	460.5	3671.9	-73.6	408.8	-76.7	-132.9	-417.3	474.2	140	1030.8		
3	0	349.9	-13.9	648	-175.8	-874.9	-258.4	235.5	641	-268.5	453.3	128.9	-75.8
	12	103.1	-5775.5	-127.1	-24.8	-199.5	181.4	28.1	-24.8	-300.5	32.8		
4	0	161	99.6	58	-122.6	34.1	-40.2	-1.5	91.5	6776.1	-21.2	-65.2	-230.9
	12	46.3	363.7	-39.5	87	112.4	201.8	9.6	-249.3	-162.3	400.3		
5	0	-119.2	-412.9	428.8	162	-61.8	1339.2	-539.6	-535.7	283.3	-171.6	166.4	44.7
	12	-4973.5	-42.2	151.5	-154.4	-162	148.7	-384.3	176.4	131.8	-233.2		
6	0	-1327.4	32.7	334.7	17.2	-12.1	-2829.9	1206.3	-592.9	452.8	-89.5	-86.2	407.2
	12	-427.4	759.2	288.6	361	-2431.1	520.3	917	-176.5	90.1	194.3		
7	0	-156.6	2813.5	132.8	88.3	-90.1	37.6	-141.6	-349.6	3400.1	292.5	-159.4	-277.6
	12	648	462.8	130.8	762.7	-500.3	13.5	297.1	-395.9	15.3	-486.1		
8	0	55.4	3525.1	-350.9	-3200.6	227.5	-123.3	178.5	101.7	1.1	297.6	-9.1	18
	12	457.6	214.3	-121.2	-241.1	290.1	338.3	399.9	150.7	39.3	-89.7		
9	0	-8822.8	72.7	54.3	182.5	-70.8	-182.5	183.7	95.1	-169.4	-44.3	171.2	-193.6
	12	154.2	195.6	279.7	303.1	-68.5	385.2	250.4	-3.3	21.2	127.9		
10	0	51.7	2659.2	-49.3	-310.8	-274.4	288.9	2870.4	-129.7	320.3	-584.1	582.4	206.5
	12	478.9	-738.4	450	-1216.1	-779.3	-508.7	-968.6	-584.2	-105.8	448.3		
11	0	109.1	-7.5	-236.4	3000.8	2739.4	-393.8	138.8	-266.2	-163.2	-175.1	632.9	-22.1
	12	102.3	-272.6	35.7	-633.5	-1466.6	-485	300.4	-196	1093.9	1001.7		
12	0	-417.2	309.2	-494.4	-335.2	304.9	-154.3	-1329	-614.5	-1702.8	681.5	85	498
	12	16.4	-190.4	-173.7	-5155	-282.1	-298	-138.9	130.1	70	-569.3		
13	0	38.7	-38.7	-6773.9	-305.1	367.3	-136.4	219.9	145.1	207.3	373.7	-75.4	771
	12	34.6	-374	-205.4	-156.5	-244.9	34.5	-101.8	473.5	-74.9	-127.3		
14	0	1.1	96.8	-1631	399.4	1920.5	758.4	-41.4	981.2	-2215.8	576.7	-64.3	-2.8
	12	-709.3	1367	876.6	116.7	-515.5	732.1	-2298.3	-371.2	-111.8	-123.5		
15	0	44.8	2367.6	136.1	351.9	61.7	-389.4	104.7	446.9	-3552.8	-111.9	-699.7	-210.8
	12	172.2	-301.8	180.2	86	-3.3	147.2	-496.7	548.4	-491.3	-495.7		
16	0	7.2	86.5	-117.4	76	-144.2	76.2	-80.8	7186.1	20.9	201.6	125.9	-292.1
	12	-171.8	3.4	57.3	13.4	-385.9	-242.3	76.1	47.2	273.9	-312.2		
17	0	388	670.8	-194.9	-948.2	185	580.5	-1229.5	-686.9	1904.6	-23.4	346.6	-223.4

	12	-101.5	203.9	-147.2	-308	50.8	5309.3	161.7	-229.5	-114.1	-291.4		
18	0	-13.2	-1803.8	448.7	-445.5	-323.1	-621.5	199	1810.4	-102.8	1041	542.8	55
	12	-239.8	-381.2	-214.6	431.6	338.9	138.4	-414	435.4	-4918.3	1435.6		
19	0	-177.5	5424.7	130.9	251.6	-848.1	-0.5	-438.2	276.8	-11	382.9	-552.2	-462.2
	12	208.1	199.7	-3.4	408.5	-301.8	355.5	-31.4	26.9	-36	-585.4		
20	0	0.9	9.4	71.2	-78.8	48	259.1	-6905	-140.5	84.3	64.4	-166.4	120
	12	-37.3	188.7	-39.3	-8.1	-19.1	-69.6	140.2	186.6	-150.7	684.9		
21	0	-132.1	-3.3	103.4	-3375.2	278.6	-652.6	2567.6	358.8	-262.8	580.3	875.6	471.6
	12	-55.4	327.2	-9.5	549.8	15.8	-558.5	497	575.1	1632.3	-243.6		
22	0	152.8	-86.3	-32.7	8056	184.4	-61.8	75.7	33.8	-165.5	-32.2	-96.5	-317.7
	12	99	-104.6	-152.9	-363.5	-38.9	18.1	-374.9	-105.3	-134.2	-335.6		
23	0	29.2	-9553.4	-231.5	6.5	-149.3	-47.8	-207.3	28.1	54.5	298.6	209.6	-927.6
	12	-56.7	-504.3	48.4	62.7	-168.1	-156.9	-27.5	-327.8	223	-282.6		
24	0	-3266.4	-2.6	-1676.4	-65.4	-2262.1	-4.7	-456.3	-207.8	-623.3	-493.1	-46.5	-634.4
	12	219.6	-251.5	128.1	1311.7	566.7	-721.8	25.3	113	231.7	-10208.7		
25	0	224.2	-259.3	478.6	-225.5	2205.2	-658.3	-1066.6	192.9	-159.4	157.7	-2430.6	-654.4
	12	-1722.1	91	-566.4	744.7	-104.2	562.8	290.9	-531.5	1353.6	-188.7		
26	0	211.4	317.7	26.9	288.8	345.9	187.3	255.8	1903.9	163.4	4028.4	132.5	391.8
	12	225.3	-382.5	-176.1	-486.5	528.1	29.9	-354	-244.3	157.4	698.9		
27	0	-194.8	-229.9	-109.9	464.1	164	-1271.3	-1254.1	109.6	1398.3	-852.9	312.5	-1308.8
	12	194.2	-57.9	774.7	207.8	-74.7	-67.9	-6959.6	-161.7	-383.1	-70.5		
28	0	169.1	-8.9	-2602.6	684.2	100.2	-302.5	228.2	80.9	3220.1	-236.6	217.1	232.8
	12	-988.4	186.2	-263.8	-639.9	100.6	-116	-123.8	-502.1	219.3	259.9		
29	0	2227.1	-108.4	-125.8	-3373.6	96.3	1142.5	304.8	-573.1	381.2	-354.6	119.2	-344.4
	12	-62.9	222.6	-241	-319.6	-431.3	-934.7	-1100.4	785.6	220.7	-246.9		
30	0	-201.4	-3407.4	236.6	-28	-2731.6	-110	444.5	152.8	476.9	-485.3	-185.6	-400.1
	12	322.9	687.4	102.2	226.3	-517.4	806.6	-1031.1	60.2	335.4	-246.2		
31	0	-204.5	355	121.8	105.6	-150.6	15.2	451	-139	117.8	71.4	-421.2	-5996.3
	12	166.4	-84.8	-10.8	-36	-140.3	-155.5	97.5	125.3	-314.4	-242.9		
32	0	214.8	55.9	-98.5	-143.3	-392	2534.6	321.3	-2913.1	-134.3	127.5	418.8	-421
	12	105.3	-682.5	-146.9	739.2	-330.1	66.3	-3868.9	332	-82.3	-123.4		
33	0	-254.8	-281.3	38.8	848.2	5.3	867.2	383.3	-204.1	-403.4	-182.2	-3572.9	-646.7
	12	-0.1	-1804.3	425.4	-263.8	269.7	-282.3	155.6	350	-115.6	-1.7		
34	0	-67.7	217.3	167.7	873	-322.1	-682.4	15.5	-1923.1	-438.9	147.2	-295.2	-301.7
	12	277.3	-359.7	147.3	251.6	5007.6	-65.1	-598.9	-52.4	255.2	-292.6		
35	0	-116.8	-54.2	1519.7	590	-102.9	282.7	894.9	-1054.2	144.6	3700.9	-240.6	-3.1
	12	-6.9	2229.8	266.5	259.2	286.2	-910.2	-58.9	-174.2	435.7	67.4		
36	0	5280	-420.4	122.5	173.2	-109	67.9	116.5	-449.8	380.7	-940.7	137.4	317.5
	12	-157.4	864.7	43.8	-240.6	-517.5	-70.2	-259.1	-260.7	700.4	-54.4		
37	0	2870	176.3	-237.2	-13.3	-250	41.9	185.2	2702.1	-120.3	-520.3	-34	-739.8
	12	-1053.4	-208.2	814.7	-951.1	-304.1	880	37.6	23.6	-304.2	-4755.3		
38	0	2077.2	285	-2031.5	368.6	461.1	13.4	479.9	-341.3	218.6	16.5	-259.8	1950.2
	12	571.9	531.6	44.2	790.3	-32.8	-479.2	500	-216.5	-3652	-3354.1		
39	0	-179.2	-1313.1	193.1	-301.7	527.6	104.4	168.8	-512.4	-656.4	-715.8	3841.6	-1039.4
	12	-356.4	-447.1	-889	-171.4	423	-287.8	33.9	285.3	49.6	94.9		
40	0	92.2	47.8	-419	-688.6	2790.3	-10.9	23.4	-2557.4	232.1	-0.5	183	105
	12	404.1	-72.3	-1763.8	-635.5	439.2	96.9	-276.4	81.3	492.4	-203.4		
41	0	151.1	-285.5	-287.5	-353.8	5593.7	-87.2	-73.8	404	-241.3	-61.9	-91.9	367.3
	12	257.2	-50	-281.5	-222.7	-77.7	109.4	-179.4	210	17.7	-714.3		
42	0	-32.5	210.2	-19	-158	86	7309.6	-20.4	-93	-342.8	136.1	-80.7	-54.3
	12	-78.4	138.6	-7.7	139.6	-360.5	-138.5	-142.3	-290.1	157.3	-0.1		
43	0	138.4	-176	-217.9	161.9	1550.3	-233.4	2928.6	-533.2	-226.5	325.2	235.9	-450.8
	12	-596.1	-211.8	624.6	173.2	252	3703.2	52.8	496.4	-327.7	-365		
44	0	57.2	-82.8	-30.2	-119.6	415.4	2318.1	1753.2	482.9	119.8	-2794.9	-169.7	353.8
	12	347.4	-47.8	850.3	-1416.8	262.6	36.6	497.2	-503.3	33.2	502.6		
45	0	-239.3	35.1	4585.9	-1018.2	807.2	-460.8	-275.2	66.7	216.1	1901.5	-164	248.3
	12	-119.6	-835.9	-167.5	13.4	22.9	-642.1	939.1	326.4	323.8	243.9		
46	0	-47.5	-56.5	-1916.9	247.5	-3031.3	-1855.2	-189.5	89.6	209.3	-214.3	-730	868.9
	12	205.9	-127.8	-57.4	471	-25.5	325.2	-416	-1181.8	440.9	573		
47	0	852.9	-1115	24.5	576.6	-2526.4	646.2	-842.4	-1119.3	-621.7	256.2	-487.2	444
	12	-567.9	55	-458.6	-2328.2	-860.2	1362.5	648.6	476.5	-224.9	-248.7		
48	0	35	-61.7	-291.5	-1882.9	228.9	777.1	-47	-804.6	383	2766.9	50	-954.4
	12	319.1	-99.6	2362.5	-140.6	82.4	-198.2	754.3	252.5	399.5	396.6		
49	0	-260.8	104.3	2993.7	-58.4	-10.4	-46.1	3228	434.1	-70.7	-98.4	-81.6	-393
	12	-192.9	51.4	475.6	-1070.4	-46.3	570.9	-271.8	382.7	222.7	-152.9		
50	0	113.6	-1823.5	312.8	-2373.7	256.6	-2296.6	-21.6	-970	-259	-785.2	-229.3	-748.1
	12	29.5	13.7	250.2	-113.8	26.1	-745.8	238.5	-482.2	134.2	-475.6		
51	0	-164	3282.2	2858	29.8	88.6	-262.2	-149.8	649.8	-136.3	225.2	242.1	109.2
	12	-531.9	-130.2	-360.7	590.6	366.4	-135.1	-429.3	-301.7	-320.3	13.9		
52	0	-277.1	3836.2	-3027	512.6	-137.1	-87.1	112.6	-314.5	-39.4	111.8	-9.5	-89.5
	12	490.4	167.3	-411	-670.9	-121.5	-248.7	-86.9	101.4	-647.1	-71		
53	0	338.7	-41.2	2928.8	-102.9	-90.2	-175.8	-118.4	-330.6	218.4	-293.5	-280.5	3045.5
	12	479.9	447.1	306.9	1538.6	-89.9	256	190.5	25.7	931	574.2		
54	0	-24.4	472.5	397.3	1088.3	756.6	800.1	-1984.8	90	-67.8	700.7	3013.1	-806.9
	12	-31.8	-449.9	1032	-291.8	15.9	356.9	651.4	293.1	-594	1086.9		
55	0	-2470.4	185	-752.3	-8	304.6	143.6	1458.1	235.5	-55.4	1158.3	-1442.2	42.4
	12	-1091.6	539.1	-510.2	-1115.3	-1573.9	-332.3	-944.8	-582	-2.6	389.8		
56	0	-199	315.6	133.6	-316.3	-813.3	3723.9	326.5	227.4	1341.9	111.1	-239.7	108.3
	12	-798.9	1168.5	-165.2	271.9	387.4	-237.3	-308	294.1	267.8	-362.5		
57	0	149.3	-117.2	-382.8	3296.1	-665.2	-296	1195.9	-953.4	391	573.3	-332.3	-198.7
	12	-463	1362.2	1069.8	-330	97.6	157.2	-529.8	628.8	199.3	-711.9		
58	0	3485.7	112.9	-174.6	2.5	-381.4	-718.4	1317.9	-307.8	-933.3	1500.9	127.6	-1841.2
	12	420.9	-544.8	208.1	-300.9	-48.5	-1028	-1042.4	-471.6	151.7	1716.9		
59	0	-252.5	157.1	-262.1	175.1	-136.2	757.9	2103.9	3432	-156.5	-9.1	-650.2	-457
	12	447.1	-637.4	-1239.7	-579	323.3	-253.4	148.3	-303.4	-121.4	-332.9		
60	0	45.2	2943.3	-31.2	118.8	-3.1	-3498.5	153.1	-166.5	80.3	577.4	-380.4	-158.1
	12	-159.7	1.8	694.1	793.3	95	504.8	-519.9	-6.3	-319.9	538.2		
61	0	-82.3	-207	-2693.4	-2921.6	99.6	-197.9	205.8	-418	-307.3	-338.9	-277.3	-184.1
	12	-1094.3	-487.4	1608.7	716.1	-1.8	183.3	-246.4	222.5	435.8	-89.9		
62	0	84.5	-1535.9	-104.5	1735.1	8.7	2049.8	77.8	576	163.6	1848.3	-550.7	-214.7
	12	682.2	-303.6	330.5	1199.4	-640.2	-183.8	386.2	-601.1	53.3	87.1		
63	0	-118.5	-106.4	-139.3	-92.9	1867.5	801.3	44.7	91.4	2790.4	378.4	-925.2	1223.8
	12	452.6	-2467.6	97.1	655.8	-372.5	-546.8	-14.3	66.4	-632	-743.7		

Table C. 18 Shape codebook 1 for MODE_VQ == 08_06, BLEN_TYPE == MEDIUM

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-101.7	798.6	-4817.5	-520.9	-116.1	84.9	-79.8	-486.3	197	230.1	-270	441.9
	12	372.3	-1055	176	-439.7	322	-1.5	987.5	-4.5	373.7	267.2		
1	0	12.8	-349.7	-156.6	-337.4	-468.4	-2306	-483.2	144.4	-390.5	-404	608.5	-92.4
	12	-3480.9	235.8	135.7	186.9	125	-140.7	-476.2	-55.1	224.1	-165.2		
2	0	-13012.8	-74.2	2.4	-111.1	122.8	-76.9	8.2	-71.7	20.9	-69.6	-27.9	33.7
	12	34.3	86.7	-148.5	15.8	-8.6	73.8	-219.7	-226.3	48.2	-585.5		
3	0	-71.1	-1160.4	-13.8	-228.3	73.1	613.6	-1006.8	-145.9	-6.2	-587.2	-141.6	-256.3
	12	4888.7	348	385.2	-373.6	-54.2	119.3	-398.5	-348.9	-423.2	79.7		
4	0	197.9	-1536.6	145.2	-779.8	-1424.8	374.1	178.2	-282.8	-322.8	425.7	-95.9	-3968.1
	12	-437.7	12.7	-617.4	-259.6	335.5	881.8	-626.4	28.7	283	302.2		
5	0	227.9	299	510.7	-661.9	2608.7	-260.5	17.4	105.5	-1439.5	-2385.1	-851.3	-308.4
	12	650.6	659.9	-115.9	-91	-692.9	-614.2	-966.2	622.4	-428.5	1906		
6	0	-38.3	4037.3	255.1	-379.9	-787.1	-215.4	190.4	-1800.5	-233.9	-145.3	-213.6	523.8
	12	-489.8	-48.2	211.6	1285.8	-25.1	-102.1	140	-131.6	-61.9	1182.7		
7	0	-82.5	424.1	165.9	60.5	-36.2	-182.7	-146.5	99	16.2	-52.1	6499.7	50.6
	12	17	-284.1	89.9	-55.7	62.2	98.8	322.7	36.3	-141.4	354.8		
8	0	-93.3	-160.8	-33.9	4021.2	-230.3	1071	1324.2	159.4	-342.9	-925.7	63.7	-612.1
	12	188.2	-787.5	-487.3	682.8	-138.5	-31.1	252	577.6	362.2	500.8		
9	0	3246	47.1	-302.5	678.5	8.8	907.8	-559.2	-1915.8	175.6	-428.9	370.3	-1131.7
	12	-1569.5	710.4	416.3	-218.2	0.7	147.6	1838.9	-521	375.3	-573.5		
10	0	-295.5	515	-48.3	-1241.9	662.6	169.8	-85.4	-928.8	293.9	394.3	473.6	-1563.3
	12	-452.8	54.3	133	-300.6	48.7	-4730.7	-260	-157.1	-178	19.3		
11	0	-98.2	2141.8	684.3	872.5	500.2	-509.8	1791.8	-441.3	-918.9	169.8	-646.3	375.9
	12	-778.7	11.8	-2919.2	-812.4	147.9	183	541.5	57.5	-618.6	18.9		
12	0	200	265.6	374.5	1203.8	191.3	-622.4	-265.1	-1457.5	288.8	1894.4	1146	42.9
	12	2766.4	-520.7	-193.3	-115.3	-612.4	1124.3	17.4	-271.1	-120.4	154		
13	0	-48.9	109	-8597.8	70	-491.4	-27	-144.4	188.8	-116.3	-152.6	39.9	-599
	12	-62.1	328.6	7.3	785.6	20.7	-180.5	149.8	-484.6	-115.7	465.4		
14	0	-15.6	623.8	3728.9	300.1	783.3	306.8	134.4	-651.2	575.9	-1671.7	-201.5	-682.1
	12	-283.8	-1704.7	371.2	-58.1	262.8	1986.1	157.9	412.7	49.5	-369.3		
15	0	375.1	184.4	-1407	199.3	-643	-132.1	374.1	-464.4	144.4	373.4	-272.9	-221.5
	12	179.1	206.6	4846.1	-138	122.8	-203.3	-101.4	137.3	-26	291.1		
16	0	192.8	-71.9	219.3	-904.9	-825.7	-177.4	74.7	4753.3	-38.5	-37.1	-189.1	532.8
	12	646.1	498.9	-438	-636.2	326.8	344	-493.7	-143.8	596.2	906.1		
17	0	-92.9	-41.7	177.3	59.4	70.7	144.3	519.9	-145.9	-183.3	-6814	45.4	54.6
	12	-257.3	29.1	88	-392.8	102.8	-83.9	88.7	-62.4	-131.8	444		
18	0	-106.8	405.3	-423.8	-649.6	-1605.4	9.5	-388.2	-3986.2	-447.2	914.2	-393.4	-368.4
	12	-65.2	369.2	-1474.6	-130.7	-520.3	14.1	613	37.6	365.1	104.7		
19	0	-3425.8	389.6	-276.2	385.9	2083.2	611.3	56	-200.8	-968.8	-581.6	89.3	187.8
	12	-314.2	-517.7	175.3	-698.9	-446.9	-260.9	-187.8	-672.6	501.4	-54.8		
20	0	25.1	-8.5	169.2	2443	1238.4	831.4	-349.9	603.3	782.9	336	-589	227.7
	12	-1822.2	108.8	-511.4	-262.3	2578.2	462.6	37.8	241	-733.8	191.7		
21	0	13.7	-1397.9	184.8	-493.7	3276.9	-701.9	533.1	-342.3	-64.3	757.5	756.1	762.6
	12	804.4	-174.4	485.8	546.1	12.8	1391.8	-156.3	-551.8	241.2	64.1		
22	0	-910.2	31.2	308.3	517	-775.3	260.8	-717.3	-637.4	-564.7	517.4	715.7	71.8
	12	272	-140.7	-149.9	5495.7	-213.2	-95.7	-90.9	84.5	-90.4	-96.1		
23	0	73.9	187.8	-1890.3	239.8	1287.1	-2819.2	122.9	187.2	196.7	173.8	-649.2	-199.4
	12	714	131.6	-182.1	-32.6	32.6	324.2	-358.6	3169.7	374.9	-223.7		
24	0	117.2	-400.7	-112.3	-608.8	-4578.3	244.9	-329.9	425.6	-373.3	-97.3	227.3	546.8
	12	244.4	185.4	-156.2	483.6	-286.6	118	806.7	-41.2	280.9	14.8		
25	0	176.8	2756.4	-359.7	339.9	-413	-181.2	-229.3	1782.9	330.1	-184.7	1715.4	-296.9
	12	-411.2	-419.9	346.6	588.5	677.4	38.2	600.5	190.6	2539.6	1238.1		
26	0	-344.6	-61.9	-174.3	-338.1	-764.7	431	1724.3	79.5	-350.4	71.1	-71.6	4322
	12	113.5	271.7	-437.2	-380.8	787.1	-392	-943.1	599.9	-234.1	53.9		
27	0	-6.6	231.1	1139.7	269.5	-2374.9	-876.8	47.9	-204.7	-181	-1231.2	168.7	-312.1
	12	591.1	-152	328.4	-4071.4	257.6	-47.9	215	420.4	686.4	-144.7		
28	0	-57.8	202.2	118.1	-632.5	-1052.8	-545.6	-2019.6	483.6	250.7	309.7	-0.8	3040.9
	12	-803.4	-435.1	327.7	-1896.2	-54.6	-192.7	238.1	121.3	-764.9	-39.4		
29	0	-24.6	-112.6	129.4	181.5	-0.7	5166.1	177.3	-236.2	25.1	-165.1	141.9	18.6
	12	145.1	-753.7	33.3	-200.9	-204.4	13	-688.2	-114.4	-356.3	193.6		
30	0	-4459.5	-226.9	294.2	338.3	-731.9	-730.2	417.2	-584.8	727	-9.3	-483.7	-1748.6
	12	-155.2	681.8	404	810.1	-88.1	181.7	64.6	188.2	-984.7	-1196.2		
31	0	71.6	-63.8	141.8	397.4	8153.1	83.2	8	-204	13.6	-470.2	298.7	360.3
	12	-157	173.1	100.6	165.3	-83.5	-229.9	-241.1	88.1	-277.8	632.3		
32	0	-70.1	-178	-133.4	42.2	-215	-347.5	-4860.7	108.6	-185.8	-227.1	392.9	-418.5
	12	-377.3	-966.5	-120.7	-292	61.7	355.8	-313.3	-162.9	114.1	82.8		
33	0	145.6	127.1	4127.7	-54.1	-2547.5	146.2	219.6	-192.5	-309.5	675.2	213.8	221.8
	12	-35.8	-400.4	-32.9	525.6	-341.3	-452.5	-497.8	426	66.8	-179.6		
34	0	-90.4	-70.5	-39.5	-252.8	2779.1	70	-3143.5	129.5	469.5	173.3	-332	111.6
	12	228.9	1032.1	315.8	-415	-468.6	-665.4	-255.3	-347.1	-26.3	586		
35	0	-925.1	-1903.2	-1980.2	1451.2	257.1	154	-168.9	48	-391.9	-323.3	154.6	777.9
	12	-1040.6	-304.1	203.2	77.1	-288.1	411.6	-151.7	-979.5	3183.3	364.1		
36	0	127.4	6.6	1191.1	0.9	533.8	-937.7	1073.1	373.5	-2977.8	714.4	352.5	-390.2
	12	-514.1	-99.5	-416.3	-17.5	147.9	279.8	171.6	-4023.2	441.2	-644.9		
37	0	-15.4	-272	-644.9	15.9	318.4	-389.4	398.7	-522.9	-1462.2	4245.2	-293.5	-102.4
	12	-590.2	-800.9	430.3	-89.1	-164.8	214.8	43.5	-115.8	-295.4	254.1		
38	0	-16.5	-5617.8	258.9	330.3	-357.3	140	-416.8	-584.1	79.3	222.8	-424.9	-415.1
	12	-72.8	-33.9	-87.7	431.9	274.7	-159.4	638.6	76.7	-253.3	884.1		
39	0	316.2	-937.5	553.8	-181.6	33	-599.1	571.2	-181.9	520.8	176.4	61.4	-65.2
	12	32.7	4371.9	40.7	-164.5	-86.7	39.7	663.5	-426.5	42	60.1		
40	0	6506.2	201.6	28.4	20.1	-141.2	-442	477.5	417.4	-228.7	-37.6	1.2	-303.9
	12	125.8	-74.9	136.2	732.6	32.8	40.7	313.5	-127.2	-788.7	-1005.3		
41	0	59.5	83.9	4.5	-5654.5	-184.4	229.8	238.5	125	-332.2	-133.9	-95.2	-566.1
	12	484.3	-159.6	-603.6	-296.6	210.9	-151.2	-360.9	-557.6	-5.8	174.8		
42	0	1870.6	-158.1	-36.6	-716.5	311.2	-532.8	-817.8	518.2	-477.3	1108	-70.8	13.7
	12	-510.1	362	-2534	338.2	-409.8	-707.6	-866.5	-137.9	483.9	621.7		
43	0	-2156.3	-228.1	-566.4	-267.3	220.6	-262.5	-783.3	163.5	677.5	2134.2	1191.1	-18.4
	12	-21.9	536.5	17.2	-2132.9	215.2	-248.4	388.5	-78.8	414	67.4		
44	0	57.7	160.8	-362.5	-372.7	-1178.3	1135.6	-1197.1	160.9	-2389.5	1.9	-2688.2	330.7
	12	177.6	-99.6	325.6	1088.9	391.6	174	194.3	635.7	329.6	169.8		
45	0	-108.5	1491.4	-480.4	-2227.6	-255.8	1356.3	233.2	651.8	161.2	280.2	163.4	402.2
	12	100.5	-117	-660.6	117	15.4	807	4063.2	544	-348.9	-214.5		
46	0	-168.3	340.1	-6.4	697.7	-72.6	-1738.2	438.8	2550.8	-2.4	150	-114.6	-2280
	12	585	81.4	259.2	30.4	-314.6	-765.5	12.7	174	-223.3	45.1		
47	0	3348.8	-2621.9	-298.3	-39.4	532.9	-134.9	113.4	9.4	-228.6	336.6	-691.1	115.2
	12	745.2	-276.2	-99.4	-588.8	449.2	73	-655.5	-359.7	-263	773.7		

48	0	126	3252.8	304.6	-108.8	497.7	2773.9	-697.3	87.8	-88.7	-453.1	-151.9	-251.6
	12	-469.6	-238.8	-7.5	360.7	-1380.7	-949.3	-201.2	2.8	20.2	-261.1		
49	0	-217.5	-2204.6	65.7	-65.5	469.1	-851.6	67.6	894.5	452.6	-2911.3	199.9	321.1
	12	500.1	50.4	415.5	1709.9	-32.5	-410.3	386.4	94.1	-234.9	623.1		
50	0	-438.6	-100.1	530.8	993.1	-92.1	-211.3	-1221.7	271.1	4240.1	98.7	143.1	428.6
	12	-487.4	-408.9	-705.9	103.5	-290.8	-330.7	-725.6	-6.6	454.6	181		
51	0	-63.4	-2260.8	203.6	153.2	-140.8	401.6	3081	107.8	342.7	666.7	-0.5	18
	12	-887.4	-312.1	-777	471.1	248	-525	605.3	320.5	777.6	609.9		
52	0	22.9	-59	-2909.9	-37.4	30.5	-137.6	3263.1	423.8	41.7	-31.4	484.5	-184.7
	12	126.2	56.6	-540	-560.9	-320.3	-142	-307.7	349.4	-1707.8	-352		
53	0	-98.1	-287.7	1477.3	-2882.8	727.1	561.9	616.8	-531.2	1322.3	-422.5	265.1	571.6
	12	-1179.6	-406.4	1036	-463.1	-551.5	730.9	582.5	105.7	-373.9	-977.1		
54	0	-84.2	-158.8	2385.8	248.9	-43.2	241.4	-60.8	1619.4	174.9	525.2	-2975.9	47.2
	12	455.5	445.9	-59.4	-221.6	-17.6	114.9	-1626.7	184.6	141.7	90.1		
55	0	-251.5	-3277.4	-228.4	-3015.5	96.9	-34.1	-291	695.5	-466.3	416.2	-207.5	336.9
	12	-104.3	607.9	72.3	747.2	-121.9	-6.5	-146.5	-486.9	-250.5	-289		
56	0	-25.4	216.4	-2663.5	10.8	95.5	305.6	-325.6	3049.2	207.9	-58	244.5	-24.1
	12	-692.9	-223.9	-64	1042	-127.6	-92.7	9.9	158	-239.5	-361.4		
57	0	1784.1	360.3	87.7	2328.9	-489.2	-1521.2	407	85	373.6	-344.7	441.2	661.1
	12	-196.7	-1562.6	620.3	323.8	-1587.2	-461.8	-889.6	293.2	483.4	-998.9		
58	0	-483.8	-60.6	279	-62	694.2	2792.3	-286.9	2782.2	-5.7	484.9	453	-758.7
	12	135.9	472.3	-58.3	143.9	-626.7	2526.1	-910.3	-279.8	-183.9	-52		
59	0	-30.9	-159.8	334.3	1816.4	349.3	1625.5	-291	-1285	-2574.7	-626	195.7	369.4
	12	374.3	1096.6	172.6	-75.1	87	-1316.6	1192.1	-8	171.1	-595.5		
60	0	70.5	-6.3	166.5	-42	-37.4	512.5	2138.4	412	3488	242.8	-268.9	-235
	12	541	477.2	833.5	-704.4	991	-893.9	-669	-946.3	-243.4	-166.4		
61	0	-223.1	112.9	-86.4	-162.6	200.5	-791.1	-693.8	-1486.8	1694.7	-228.7	-3094.2	-704.6
	12	9.6	37.7	-339.1	-585.2	-249.9	162.6	638.7	-150.3	-2114.8	1110.2		
62	0	-44.1	38.6	-248.3	-96.4	246.8	-499.9	2575.2	-2983.2	105.5	-675.1	426.6	321.9
	12	-227.7	40.3	175.7	357	703.5	194.8	-1571.8	-13.9	154.7	551		
63	0	-57.6	-25.8	2653.1	352.2	323.7	-3558	29.7	327.4	288.3	46.1	351.7	-175.3
	12	-286.9	-190.7	-311.7	-336.4	33.5	-214.5	-143.3	-190.6	131.9	72.7		

Table C. 19 Shape codebook 0 for MODE_VQ == SCL_1 / SCL_1_960, BLEN_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-116.4	-379.5	688.5	-58.8	149.6	500.8	-58.8	-8192.0	198.8	-164.0	-54.1	25.5
	12	6.0	-12.6	13.6	66.7	55.1	-95.9	-131.3	62.4	76.6	-178.5	-189.1	-189.0
	24	-177.3	-75.4	161.9	34.5	-142.5	94.5	106.1	-185.1	-34.8	27.3	26.4	-35.3
	36	117.5	168.4										
1	0	-490.7	-2041.4	-1560.4	8.1	90.4	-465.1	298.7	-402.5	94.4	-4241.9	-519.4	-368.4
	12	-176.9	-170.6	282.2	-116.3	-445.6	-75.0	-91.4	-179.7	-2.4	-67.1	745.9	15.1
	24	-86.7	259.8	174.0	-19.1	-2.4	180.2	103.0	-166.4	31.4	147.4	67.7	-115.2
	36	22.0	-381.3										
2	0	-460.2	-274.7	-88.7	486.6	586.7	-744.1	402.3	512.7	555.7	127.1	88.8	184.5
	12	130.8	172.0	231.0	-67.3	99.5	-208.7	16.6	88.8	24.4	-381.9	-19.8	26.7
	24	59.6	-105.4	63.9	-114.3	-191.8	96.5	-158.3	-7.7	-100.7	-7833.2	-23.4	60.9
	36	34.0	155.3										
3	0	-2031.3	25.5	1041.7	-5.3	-15.8	-430.7	236.7	83.5	-271.2	-172.2	-218.8	-734.8
	12	535.7	15.3	-376.2	4400.8	633.3	93.3	-379.9	276.0	-73.7	278.5	-145.8	-230.8
	24	27.3	46.0	-79.9	36.6	-23.9	71.0	53.5	93.1	-144.8	27.5	110.5	268.4
	36	-68.8	13.3										
4	0	415.2	417.0	-542.9	-664.6	446.1	-572.3	268.6	1323.9	30.1	721.4	2.4	71.5
	12	-57.1	-136.6	-604.8	152.9	9.0	-239.1	312.4	-135.3	-76.4	140.7	183.9	-191.7
	24	-17.2	202.5	372.5	7247.3	125.2	160.1	282.8	106.9	126.4	279.4	151.1	-223.1
	36	119.7	15.0										
5	0	-719.9	-894.4	388.2	-995.5	-910.0	-250.9	-43.1	-26.1	371.5	-129.9	-94.3	-258.4
	12	108.5	185.4	132.8	-17.8	-216.3	103.1	-99.6	-19.9	7.3	-35.6	55.9	-7066.5
	24	39.4	-19.9	-192.6	57.1	-134.2	-6.6	50.1	-481.7	50.9	-2.0	-22.5	342.4
	36	120.0	44.3										
6	0	3127.0	-283.3	180.1	-68.9	187.5	104.1	-111.0	4104.9	130.0	-97.5	-121.3	-573.4
	12	122.2	1.8	162.7	-52.2	34.4	373.5	-548.0	208.1	126.9	-389.7	-39.5	52.8
	24	116.7	-39.8	-372.9	85.9	-56.3	62.9	57.6	58.6	87.8	24.9	-55.1	-216.2
	36	86.0	69.7										
7	0	-871.2	301.9	-250.1	-2261.0	-3120.8	-3059.7	-652.7	-188.7	474.2	-18.5	29.0	-127.3
	12	163.2	-36.6	-45.8	-118.2	875.8	101.4	88.3	96.6	-24.9	49.4	103.0	291.5
	24	103.9	41.3	-481.4	23.7	-355.8	-109.5	-158.2	57.5	-115.4	15.5	-13.0	365.2
	36	-127.0	9.0										
8	0	-48.3	-845.2	109.3	154.5	-664.0	-21.7	8192.0	-239.3	242.6	-78.0	-148.6	-87.2
	12	-178.7	-104.2	-157.1	-21.0	191.9	-259.7	18.1	167.2	29.5	-94.7	43.1	67.2
	24	105.6	312.2	-30.4	9.8	151.5	-57.0	19.7	-219.5	44.9	55.6	71.5	325.3
	36	104.9	-210.3										
9	0	621.1	64.7	-46.8	-427.5	135.7	-14.9	-0.7	481.8	347.2	149.9	-15.4	-114.6
	12	28.6	97.6	51.3	8192.0	-408.4	-181.6	54.3	95.8	22.7	-89.6	23.8	-119.6
	24	115.9	-54.0	98.6	101.9	86.0	-3.3	-5.5	82.8	-258.3	-154.1	150.9	-116.1
	36	-47.8	102.7										
10	0	-116.5	-2814.5	1684.6	243.4	-217.0	-952.5	285.5	2291.6	98.1	-25.7	-100.0	117.1
	12	-357.5	-248.6	-196.1	59.0	-712.3	173.8	-385.9	-123.3	-61.8	471.6	-61.0	104.0
	24	13.4	-165.1	-364.8	124.6	264.2	-844.3	99.1	-1.7	175.8	-92.1	45.3	58.9
	36	-33.9	157.2										
11	0	687.3	-566.0	-22.9	-375.2	-71.9	640.7	882.4	801.2	-66.6	-514.4	-285.5	466.8
	12	50.6	-576.6	101.7	186.8	7340.4	-70.7	65.3	81.7	-133.3	77.5	-75.2	-122.1
	24	124.6	-134.3	52.9	-50.7	134.5	-73.0	3.0	39.3	48.5	-7.3	-88.5	11.0
	36	6.0	66.0										
12	0	-551.6	-672.1	24.7	2023.0	-4.8	200.7	-261.4	3.5	45.5	26.3	108.4	24.4
	12	191.0	300.0	-299.7	-259.0	59.8	83.2	12.1	69.6	6416.6	180.4	221.0	-90.9
	24	-56.2	-135.8	180.7	-177.9	-45.7	35.2	82.3	-139.4	200.6	-153.0	-117.0	86.2
	36	6.6	-282.1										
13	0	256.6	972.7	298.7	32.0	-1574.3	-523.8	-393.5	-124.1	537.0	-414.9	-74.0	660.9
	12	8.3	-76.6	-119.5	-44.5	-14.6	-77.7	-70.5	53.7	-40.7	-576.6	7036.5	-63.0
	24	-36.0	-399.3	99.1	40.0	-101.2	-70.6	77.6	99.4	30.2	-235.6	-201.1	-39.9
	36	72.1	-102.0										
14	0	63.3	-2452.7	-681.3	338.2	-275.6	-355.2	10.4	-288.6	675.9	4450.7	-59.6	-257.3
	12	236.7	46.7	1.0	23.7	283.7	-121.7	35.7	252.5	16.8	172.4	-85.2	-112.6
	24	38.0	31.6	119.1	9.9	-284.0	120.8	-75.4	182.0	-45.6	126.4	65.8	-67.3
	36	-48.0	0.5										

15	0	-493.4	-629.2	-292.1	-89.5	244.3	331.1	125.5	-114.2	-61.1	-208.3	-55.5	55.1
	12	92.1	8192.0	-126.2	80.4	-272.8	133.2	122.8	-127.2	65.5	181.1	91.6	-85.0
	24	-101.9	-38.5	24.3	-37.0	119.7	-68.8	-31.2	12.1	-60.8	-94.7	10.3	-93.7
	36	-37.4	70.4										
16	0	770.4	-1030.7	-715.8	-166.9	-107.1	-245.6	-706.7	-394.6	-197.2	-118.9	307.9	-158.8
	12	77.2	134.1	255.0	-149.2	51.3	-111.0	-376.0	196.9	-183.4	83.7	-29.7	-15.5
	24	-83.7	62.0	-305.8	-110.8	116.4	31.0	7552.2	31.4	-269.0	232.8	143.6	-130.8
	36	-75.8	-62.0										
17	0	6.6	-870.3	568.5	100.5	148.0	-6.3	-67.9	-68.9	202.1	-25.2	-222.4	102.5
	12	-8192.0	173.9	-76.1	4.1	13.1	32.4	188.0	-125.3	-69.3	227.3	146.1	-28.1
	24	71.1	69.5	157.8	-52.0	-90.8	71.6	29.1	0.3	-74.2	59.5	212.5	-116.8
	36	48.2	65.1										
18	0	500.1	2125.1	-932.8	261.7	-157.7	20.9	-123.1	-185.1	158.4	-250.8	-4827.2	-286.4
	12	-56.7	23.6	155.9	129.0	-104.2	255.7	115.1	-14.3	-40.1	-104.4	15.0	92.2
	24	-189.0	-150.7	539.4	20.8	-185.6	-119.9	148.2	19.8	217.1	104.2	27.3	606.2
	36	-13.2	-4.5										
19	0	237.8	-3668.8	-840.0	-4148.7	-203.7	-109.6	-286.3	634.1	-10.9	-108.8	-1.4	78.5
	12	-90.5	-19.8	-265.7	-189.0	-25.6	296.8	150.1	253.7	29.1	105.6	-83.6	-16.4
	24	-102.5	-197.2	-227.4	-32.4	-29.6	-7.3	73.7	266.2	-12.1	-125.9	-156.1	-98.3
	36	-77.2	-274.4										
20	0	1544.5	-8192.0	-1259.2	479.4	290.8	-407.3	0.0	450.4	-325.8	194.6	-81.4	200.1
	12	116.4	-62.5	267.0	-7.3	-403.8	256.2	25.4	108.9	-113.8	-299.1	169.5	-66.8
	24	12.5	62.7	66.2	15.5	-221.1	39.6	165.0	-28.6	-32.0	-201.1	-21.4	-59.6
	36	-89.6	59.3										
21	0	802.2	124.5	910.8	-100.5	-25.3	-167.2	306.8	66.8	-225.6	-6.1	-19.6	-208.0
	12	107.9	3489.8	-142.2	52.4	88.1	189.6	-126.2	72.5	-26.3	7330.9	-15.1	-28.7
	24	75.3	-77.5	-139.8	90.2	286.8	70.2	35.2	-226.9	237.9	31.6	-6.4	101.2
	36	-126.7	-193.7										
22	0	317.2	-210.7	-93.6	-266.3	-261.4	-89.5	-68.0	78.1	-8192.0	110.6	-57.6	-16.2
	12	4.7	26.3	-45.7	82.6	78.5	156.3	336.6	60.5	42.0	261.2	-50.2	-80.7
	24	-105.6	120.2	-61.8	39.6	103.4	-36.5	90.8	-294.0	-44.6	-8.9	26.6	-85.8
	36	13.6	134.6										
23	0	-1713.3	-5239.4	-57.7	-190.4	-492.7	829.1	199.5	-115.3	364.4	52.6	-163.8	56.6
	12	-276.2	-117.5	74.2	204.4	116.8	-787.9	-113.8	42.5	75.2	-186.2	110.6	92.0
	24	152.8	-18.3	75.9	-204.5	235.8	81.4	213.2	501.9	73.5	-184.7	130.6	154.2
	36	191.8	-76.8										
24	0	-1769.9	-953.2	1624.5	-974.9	239.1	-209.9	24.7	50.5	-25.4	48.8	-4762.1	11.0
	12	89.4	-60.9	-87.7	-262.6	301.9	211.7	-124.4	-190.8	-29.2	23.5	111.8	29.6
	24	-70.7	3.2	-141.5	-137.9	-343.5	221.6	-73.6	-32.1	-136.4	-41.4	134.2	-143.2
	36	143.4	84.3										
25	0	3661.1	-3009.3	775.3	-125.9	-351.1	193.9	434.9	-189.7	137.5	36.3	55.1	-40.9
	12	-52.7	42.7	7.0	-199.1	372.6	46.8	95.1	67.6	-45.0	454.9	145.5	72.2
	24	-91.0	-239.4	121.9	55.0	161.4	35.3	-34.4	-21.5	-17.0	20.6	-49.4	24.5
	36	-21.9	-91.9										
26	0	1140.3	-552.1	464.1	831.0	325.8	-197.6	-29.1	-190.9	-18.8	-23.3	-8192.0	-186.3
	12	-435.0	-98.7	-75.9	-74.6	63.2	624.4	60.1	98.0	15.8	-94.9	23.1	-75.4
	24	137.6	41.9	-847.0	122.5	184.7	-102.5	-97.3	-165.9	-15.1	-107.9	-244.6	-91.4
	36	-25.3	93.9										
27	0	514.9	-3436.3	-798.7	359.2	-4073.8	-146.2	-590.2	-477.5	-72.1	-232.7	71.4	-138.4
	12	-26.0	143.7	115.8	-48.9	-380.4	262.3	-184.8	-61.5	74.5	115.6	-85.1	160.0
	24	45.5	-131.8	-41.7	-163.6	139.8	24.5	47.3	519.5	-123.0	70.7	180.9	407.5
	36	13.9	-38.5										
28	0	161.1	-3814.7	-160.0	-78.4	4144.5	72.4	97.1	-223.2	-291.2	-119.7	-209.6	24.6
	12	-82.3	-394.7	57.7	-179.4	-222.9	420.0	-521.8	-45.4	53.2	450.6	74.3	-28.7
	24	17.5	-66.8	-620.3	-37.8	-11.8	131.7	-117.3	-68.1	43.6	198.8	-14.3	295.7
	36	138.1	-296.1										
29	0	829.1	91.3	724.2	331.4	107.6	519.1	-489.4	72.9	546.5	541.7	-82.9	-22.8
	12	212.4	-66.1	363.8	-124.0	-33.6	126.8	86.5	-7453.2	20.8	-20.1	-93.3	154.3
	24	20.7	129.6	153.3	-117.0	42.6	9.3	206.5	115.0	-20.1	12.4	2.6	-47.6
	36	-30.3	-80.0										
30	0	1380.3	-193.1	540.4	436.0	-20.8	422.1	86.2	420.9	345.9	-8192.0	-16.1	-55.7
	12	-183.5	191.2	70.2	-205.2	886.9	-212.7	198.1	-54.1	107.3	-221.6	53.2	-306.9
	24	14.5	172.2	3.6	27.6	118.8	-159.4	-123.8	-178.7	203.5	202.4	-215.9	103.4
	36	-190.7	83.8										
31	0	127.9	-689.1	877.1	-857.8	-1229.2	3160.6	1797.6	-2800.4	647.2	188.4	283.8	-791.6
	12	126.3	-236.6	-213.8	-351.3	-8.5	10.9	-192.7	193.1	28.3	227.8	-110.4	-137.4
	24	-23.3	-409.8	230.0	-41.3	-270.9	-45.7	-27.6	50.8	-15.5	-105.1	171.4	192.5
	36	-92.6	-230.9										
32	0	-1280.3	352.7	-3567.4	-768.9	-40.6	-670.0	-3088.9	44.8	93.1	-58.3	-304.8	-627.0
	12	33.9	-342.2	180.7	96.4	389.2	-105.6	397.6	16.2	25.1	991.6	-112.5	65.4
	24	-179.4	-267.3	4.1	71.3	-158.3	99.2	104.7	-106.1	25.4	421.9	255.7	320.3
	36	-246.0	135.1										
33	0	-161.2	-316.4	2158.5	-4209.4	-642.9	867.6	893.8	492.2	-145.6	242.3	348.2	-483.3
	12	434.7	99.8	-177.1	-135.2	-456.9	-106.2	23.4	-45.6	207.7	-230.4	-58.1	147.3
	24	112.1	221.3	-64.6	92.9	-709.2	-77.7	125.3	89.3	-3.5	57.7	-11.4	-3115.6
	36	33.3	-319.2										
34	0	969.1	-20.7	838.9	-403.7	69.7	-538.6	-3215.0	-118.3	-2007.0	-316.0	660.3	-1342.7
	12	-1080.5	-306.8	-163.8	-295.2	395.9	61.1	-1315.5	-108.3	-102.4	-99.5	-124.4	-113.4
	24	59.6	-175.4	-230.1	-132.3	-234.2	-117.9	-13.5	26.7	-95.2	-55.9	452.0	210.0
	36	-221.2	-310.5										
35	0	74.5	809.5	-935.6	199.9	-148.4	2621.1	142.4	129.8	-1220.9	172.5	-186.4	-4731.0
	12	289.8	114.2	6.2	81.6	-335.6	83.0	277.8	-186.2	14.7	-278.4	102.4	110.0
	24	37.4	116.9	-516.6	55.4	-8.7	-141.1	-127.8	46.8	227.0	-69.6	155.5	9.0
	36	43.2	-92.4										
36	0	-332.3	-168.8	-1408.0	-4171.9	-657.6	389.2	-675.0	-3473.9	347.2	69.3	-329.4	101.9
	12	-211.8	87.2	48.3	98.4	340.3	-24.5	257.2	-67.9	-85.8	176.1	-46.0	132.3
	24	105.3	506.0	-85.9	102.8	-429.6	144.1	270.2	109.1	-193.6	-85.0	423.3	-387.9
	36	39.6	-202.1										
37	0	86.6	205.4	-1164.2	219.6	186.1	496.4	-866.5	125.9	2484.9	-114.8	-214.6	-0.2
	12	79.8	255.6	245.3	224.1	111.5	7071.5	-118.3	14.8	27.5	5.4	63.2	77.0
	24	-38.3	146.9	324.1	60.4	217.0	-152.5	-69.9	372.8	-190.6	113.5	256.9	-86.6
	36	48.5	-48.7										
38	0	734.3	-254.1	704.9	-4201.9	543.9	-4259.7	-513.1	-120.1	-30.2	25.8	-126.2	21.2
	12	-233.8	156.7	58.7	1.1	79.6	-0.0	100.2	-40.3	-240.9	220.3	-18.2	-224.8
	24	-51.8	211.1	-7.2	64.8	344.4	-11.1	179.1	31.4	235.2	22.3	-15.2	51.1
	36	59.8	-106.7										
39	0	2102.3	2122.6	199.2	303.4	40.5	-856.2	3995.2	-44.7	248.5	134.5	262.2	-116.7
	12	-43.2	110.8	226.1	111.1	182.6	-250.8	-51.4	168.2	-22.0	83.8	491.3	-110.6
	24	-108.8	282.9	-378.4	-107.4	42.5	-62.5	73.4	167.3	-0.1	4.8	529.8	246.2
	36	101.9	73.4										

40	0	645.7	109.3	-281.1	-8192.0	67.9	98.6	-142.0	13.1	130.4	66.1	-5.5	55.5
	12	239.2	-149.0	-239.6	-209.3	127.1	-209.0	-245.4	-67.1	125.7	-293.1	43.7	189.6
	24	134.9	78.9	-107.8	-53.4	-260.1	160.2	-82.8	118.6	-21.5	-7.1	268.6	55.5
	36	43.4	-109.1										
41	0	365.5	3471.7	-1381.0	-4276.9	130.7	10.1	182.2	623.1	-86.6	-18.6	22.2	-14.4
	12	-34.6	-79.9	-4.5	114.8	-48.9	-53.3	196.1	78.9	94.3	111.9	32.8	81.7
	24	50.4	-62.9	200.2	-64.0	62.1	-218.8	318.6	-78.7	151.3	60.1	-121.9	458.3
	36	15.5	-104.6										
42	0	-2217.7	-110.8	1283.9	1818.0	3964.6	-218.6	-407.1	-195.3	-36.1	140.7	74.5	126.8
	12	-88.1	84.9	259.9	22.0	273.2	-121.9	921.3	-178.3	14.5	-139.7	-187.6	27.3
	24	67.4	-120.2	-277.9	50.1	278.0	-1.8	-100.0	-183.9	-357.7	203.5	174.3	-439.3
	36	-20.2	76.9										
43	0	-569.0	64.0	-426.3	2649.5	-1644.1	-4403.7	1175.4	114.7	-94.6	111.5	192.4	45.1
	12	117.8	105.2	49.4	5.7	-187.6	15.1	-384.7	56.7	-1.0	52.4	22.6	-85.5
	24	24.6	10.4	-213.1	-95.0	85.5	213.7	63.8	-82.0	-97.8	97.4	-282.8	-30.4
	36	50.4	-225.4										
44	0	923.8	209.6	-39.8	364.6	82.4	439.5	237.2	433.7	101.5	-100.3	-123.2	8192.0
	12	-77.2	185.7	81.6	443.3	-45.7	67.0	170.9	-187.1	-135.1	307.1	47.2	-26.4
	24	231.5	-27.0	-92.3	52.1	109.9	-6.6	-26.6	-81.8	111.7	-7.7	-83.1	415.1
	36	-58.3	-101.6										
45	0	290.3	293.0	1749.9	2809.9	-502.9	1673.2	-3799.0	254.8	387.1	-337.2	253.2	-23.6
	12	280.7	-181.4	-22.8	-118.4	136.2	12.7	121.4	171.0	-139.7	491.5	24.2	110.5
	24	-170.4	59.6	-160.0	-110.4	119.1	10.6	-44.3	-122.0	100.3	-74.0	-202.3	65.9
	36	-66.4	244.9										
46	0	3235.0	175.2	-674.7	-895.6	4486.9	-97.7	400.7	-73.7	-48.1	144.5	-148.4	-79.6
	12	1.1	84.9	-85.1	-102.5	-116.2	-51.7	-20.3	-121.6	-20.5	-22.6	99.7	-46.7
	24	106.7	140.1	-883.4	-44.3	-177.7	161.6	-142.1	76.8	58.4	-8.8	-221.2	79.2
	36	-135.4	-245.8										
47	0	-3553.5	392.7	3314.5	-382.5	-124.7	144.3	-39.4	26.8	412.2	56.0	154.8	-390.7
	12	74.3	87.2	-44.0	-187.5	-101.8	263.6	149.0	-74.6	-35.4	-135.9	-10.3	439.7
	24	-85.5	24.7	-30.3	71.8	-12.5	-8.5	151.3	88.4	282.0	165.3	290.5	-309.7
	36	113.8	217.0										
48	0	475.5	-121.0	87.1	278.3	-646.2	-157.1	334.9	822.3	838.4	-587.9	-519.1	-1194.5
	12	-805.6	169.2	-105.0	-193.9	-146.4	-147.9	-150.7	-56.3	42.1	-143.8	-111.9	-37.7
	24	7815.7	87.4	88.6	-170.1	-41.9	-85.2	40.2	-320.5	20.9	-18.5	70.4	-109.9
	36	-34.4	163.3										
49	0	309.9	587.1	3922.7	954.8	-220.4	-3815.0	-185.9	-16.4	147.1	81.3	263.2	-88.1
	12	117.4	134.2	-81.8	4.2	27.9	-100.9	56.3	121.6	-46.3	-634.5	-39.1	-29.3
	24	-175.7	-3.6	-194.0	-54.1	27.6	-26.3	240.7	-61.5	-151.0	179.6	-120.2	68.5
	36	61.6	4.4										
50	0	1376.3	-8.5	-99.5	3970.6	-3834.3	248.9	367.5	-72.6	123.9	316.2	258.5	-12.7
	12	171.4	100.4	98.3	54.4	204.0	-280.2	-31.0	-10.4	74.3	60.8	-99.1	-27.7
	24	-108.4	457.1	-126.4	-57.3	126.5	-39.5	-37.0	4.0	105.4	295.9	74.6	210.4
	36	-13.4	-174.6										
51	0	335.2	-747.8	91.8	114.0	2682.8	-3919.6	764.9	51.4	783.6	-97.1	-131.2	-593.9
	12	782.8	133.9	-58.9	-270.2	91.3	228.0	-135.0	-238.3	14.5	2027.2	31.0	22.9
	24	70.9	-9.6	-7.0	-77.4	120.4	-132.6	-214.9	461.4	168.2	72.0	303.3	-48.8
	36	12.8	-345.2										
52	0	-3482.8	-1187.4	-1718.6	1566.3	-193.5	-344.8	315.7	359.8	103.0	-21.9	138.5	-232.9
	12	90.9	3.7	-79.4	-55.7	960.4	-102.1	-152.3	-168.3	-118.0	-734.8	-266.6	-89.6
	24	74.6	-16.5	-469.4	-33.0	-59.4	-158.5	-61.6	-296.5	182.2	23.9	-88.9	-265.8
	36	-171.1	-92.3										
53	0	-471.1	-555.5	-209.8	-60.4	206.0	-8192.0	-212.9	92.8	-120.1	185.7	75.2	-84.6
	12	-132.2	-57.0	-44.1	2.7	89.6	-202.7	244.4	122.9	50.1	113.5	-149.5	73.3
	24	36.0	55.5	-47.8	111.5	104.9	12.9	-32.0	-18.5	26.5	-143.2	34.8	44.9
	36	236.9	19.5										
54	0	77.5	79.0	144.4	208.9	8192.0	-80.7	-109.1	125.3	-70.3	79.0	-323.4	143.8
	12	-128.7	-114.3	25.8	176.4	-72.5	352.7	-112.8	-112.6	-36.0	210.3	-126.2	103.8
	24	-194.4	-31.8	306.7	-71.6	148.2	32.0	3.8	-21.3	-116.8	159.1	4.9	321.0
	36	-12.8	206.5										
55	0	-8185.5	-377.6	165.7	-325.6	155.6	-277.9	-119.0	193.3	69.6	-164.4	-216.4	-126.9
	12	223.6	-68.9	-177.6	286.8	107.9	57.5	-88.0	65.3	-40.0	807.2	-147.0	-154.4
	24	28.3	-221.5	123.5	26.0	155.1	141.4	-41.9	203.6	-44.3	-13.8	8.7	211.8
	36	-127.4	107.1										
56	0	2370.1	117.0	142.2	515.3	-66.2	-968.5	-55.0	-4076.4	531.2	-41.1	-63.5	-32.7
	12	-99.1	181.4	6.4	-50.0	-26.6	-93.7	-327.6	147.5	106.9	-193.4	-48.4	76.8
	24	-82.6	-159.7	62.4	-31.2	-271.1	-156.2	-213.0	92.5	-51.4	79.5	9.4	-331.3
	36	-32.6	-60.8										
57	0	-122.9	175.9	-1498.1	-169.1	-347.0	-250.0	-259.3	61.2	48.4	-134.4	146.5	-115.6
	12	209.6	81.0	232.2	259.7	104.2	-252.6	-100.2	75.2	7.1	73.7	3.0	306.7
	24	-70.7	-25.0	294.9	47.9	-81.3	69.3	7.7	8192.0	-96.4	141.2	-79.3	126.0
	36	29.8	368.4										
58	0	448.8	-2258.7	-1010.0	569.6	41.7	183.4	-105.5	-122.5	-102.8	-24.4	285.2	133.8
	12	-11.3	299.5	-4384.3	222.7	71.7	-241.0	-38.4	-120.9	-71.4	-4630.9	97.7	-91.4
	24	-46.4	-207.3	-5.6	-99.1	103.5	-187.7	114.3	97.9	28.9	6.1	-33.2	-269.8
	36	-128.1	104.8										
59	0	-137.2	898.6	-4505.3	963.7	-1909.6	-702.8	799.0	324.4	-64.5	71.9	-228.2	207.7
	12	9.0	-39.6	320.4	-143.2	-542.3	-102.6	15.5	89.6	83.6	-29.8	-135.5	172.2
	24	125.5	-26.7	-123.2	-20.3	375.9	-18.5	-35.3	-353.5	-137.9	95.1	294.7	-418.4
	36	90.2	-340.3										
60	0	1060.9	100.4	394.7	347.2	-226.6	-314.8	-339.4	-982.4	-88.8	-83.0	123.3	194.6
	12	-20.9	-106.0	-8192.0	106.2	47.1	-559.6	593.7	-32.5	-66.2	685.2	-174.6	-52.8
	24	39.1	9.7	115.7	16.7	180.5	-74.2	72.6	-279.3	-47.3	-78.7	72.6	-159.8
	36	-83.0	-28.4										
61	0	644.2	-1092.6	-8101.8	-69.7	96.2	84.4	-62.0	-363.2	323.9	-62.5	-22.0	-11.4
	12	140.6	30.2	48.0	-42.3	-72.3	-189.0	123.6	-167.1	-149.4	-272.7	-179.3	-166.8
	24	234.8	-1.4	-274.6	110.5	89.9	256.8	86.7	-217.7	37.4	134.0	52.4	-163.4
	36	96.4	-339.9										
62	0	-836.7	1897.9	-929.4	-235.7	124.0	-329.0	304.4	920.5	79.1	-56.1	216.1	-300.7
	12	69.1	274.7	-4674.2	-43.5	-29.7	127.4	-135.0	10.9	-29.0	3089.0	-85.4	64.8
	24	112.7	-69.9	-0.7	-105.6	45.8	-28.7	201.6	-21.6	277.5	-134.4	-158.4	-43.2
	36	-57.3	162.8										
63	0	-603.2	285.9	-3963.7	623.6	4057.4	-55.6	579.9	-55.4	8.3	83.6	112.8	67.8
	12	-78.0	-170.9	40.6	-40.5	228.1	40.1	-226.0	-141.8	1.5	101.6	-139.4	212.5
	24	106.1	-33.0	526.3	-172.2	61.4	-39.6	-31.8	77.5	-7.6	192.5	115.1	-356.4
	36	207.9	157.4										

Table C. 20 Shape codebook 0 for MODE_VQ == SCL_2 / SCL_2_960, BLEN_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1307.2	1337.7	-623.5	842.0	189.8	1113.2	-451.9	-6609.5	-358.2	1057.5	-96.0	350.3
	12	-261.7	448.8	-193.4	816.1	251.8	-216.4	109.4	-911.8	334.2	-683.0	115.3	-250.9
	24	572.5	480.8	-60.0	-484.2	38.0	-424.4	93.8	-413.7	-89.4	107.6	393.1	339.6
	36	267.9	47.0										
1	0	-465.0	-647.6	-3223.9	166.4	-869.1	-1219.8	1.4	-410.1	-1297.1	-4106.8	255.2	-13.3
	12	-840.3	-1277.8	5.5	360.7	-344.0	58.6	72.5	179.6	122.7	-427.8	201.4	-189.7
	24	35.8	-246.6	-72.6	195.9	0.6	61.4	180.5	186.4	-19.8	39.0	-327.0	279.3
	36	-105.1	161.8										
2	0	4257.9	566.6	958.4	-753.1	1770.2	635.4	870.0	150.4	-372.1	-618.7	-54.1	-1354.9
	12	1465.7	45.0	-268.0	132.8	-68.4	453.8	759.3	-792.1	660.1	-451.2	-452.4	-95.8
	24	54.9	-454.3	547.5	-220.8	280.5	240.8	-3.1	494.0	-76.3	57.0	-398.8	54.7
	36	-349.6	97.6										
3	0	-297.8	-696.1	828.1	-602.5	-2370.5	-155.7	-984.0	507.3	-1344.4	337.1	589.5	-464.0
	12	499.8	-268.1	712.0	4453.4	1036.9	1093.7	115.0	-777.1	-543.5	403.1	-268.2	-71.2
	24	-53.1	729.8	-38.0	-235.3	52.5	131.5	4.7	314.2	-20.8	143.3	5.3	-79.4
	36	-34.9	81.6										
4	0	449.6	1226.2	-617.2	-181.8	639.8	2587.3	1242.2	-2811.1	-1921.6	264.6	1601.1	860.0
	12	-190.8	-1982.7	255.5	685.3	1167.5	1998.1	358.7	845.5	381.9	-1192.3	-443.8	250.4
	24	-330.3	243.3	-160.2	144.3	-144.7	-55.5	185.3	-196.5	19.1	-89.2	-62.0	-201.4
	36	-238.1	-82.1										
5	0	-371.0	92.1	-449.5	-479.5	582.6	232.5	268.0	-209.3	-201.2	566.3	-171.1	-350.2
	12	-191.0	147.3	31.1	7329.3	24.5	-469.5	509.2	393.3	-155.5	-71.2	43.8	-76.1
	24	505.9	-437.8	183.3	-40.3	-104.0	197.1	-148.1	-75.4	48.9	147.2	283.1	-362.9
	36	367.3	-76.0										
6	0	884.6	384.9	-186.0	641.6	-107.7	320.6	-106.4	20.4	172.8	407.0	63.6	225.4
	12	-30.8	665.6	417.1	296.5	13.5	115.6	-57.2	-22.0	418.9	150.2	-12.0	233.3
	24	293.1	-117.9	-16.7	6517.2	250.7	239.4	-106.1	216.8	538.0	327.2	331.7	-152.8
	36	-139.2	-87.2										
7	0	110.9	2460.6	3471.5	362.1	-170.5	646.3	-707.5	-1014.9	-917.1	-1644.1	224.4	555.1
	12	659.2	1752.8	239.0	-929.6	-590.7	421.8	390.8	306.1	-1518.7	352.8	-160.5	336.5
	24	-639.2	-909.7	-1291.0	-363.2	183.0	392.6	51.5	-474.7	-381.8	-392.1	306.1	-135.5
	36	60.3	300.3										
8	0	105.7	-943.8	150.6	-1346.4	562.8	133.6	6010.0	-114.6	-955.3	-271.3	-787.1	-165.7
	12	-1448.0	-22.6	-800.3	10.9	189.2	287.0	98.9	137.3	701.2	-271.8	455.5	164.5
	24	235.2	-366.8	145.5	-45.2	333.4	-785.5	-180.1	-171.9	-168.8	8.1	72.8	130.6
	36	33.4	-221.6										
9	0	82.6	1911.1	338.5	-156.3	1561.0	829.3	297.4	-1090.0	-37.0	-1979.1	-335.9	687.7
	12	-2042.5	126.2	725.9	3711.1	-996.4	-1063.5	-938.7	-30.5	1235.9	807.8	-10.6	-44.7
	24	246.0	-311.0	251.4	241.4	-190.1	-51.6	317.8	418.5	-314.5	-78.1	39.0	234.7
	36	61.8	83.9										
10	0	94.3	467.2	358.6	100.8	71.0	-198.0	-80.6	305.3	14.2	-91.6	332.3	84.7
	12	263.0	303.3	183.9	317.7	-127.5	178.5	-6651.3	151.7	38.3	-593.7	290.7	309.9
	24	241.3	-129.9	168.6	-304.1	144.6	141.0	-66.6	-107.1	197.9	197.4	-164.5	329.7
	36	-270.4	10.2										
11	0	413.4	-87.4	-416.7	-25.7	254.8	-55.0	-3.6	176.5	-128.8	-36.9	-128.0	-190.9
	12	-211.9	-71.7	-14.4	-191.0	-201.8	6901.9	127.9	-74.6	25.5	270.3	81.8	-135.9
	24	-178.9	188.5	-203.0	-25.4	-6.2	36.7	-100.3	167.6	-33.6	3.7	167.3	8.6
	36	-20.4	-100.9										
12	0	34.2	-176.7	-188.2	-150.7	22.2	102.6	147.1	-241.1	-371.2	399.7	267.2	-77.6
	12	92.7	-66.0	-251.8	-61.9	100.4	-31.7	-3.8	165.1	100.8	218.2	22.0	14.9
	24	-87.7	6.5	196.9	-182.9	7452.0	-170.4	44.6	-113.5	-59.7	-48.1	18.9	23.3
	36	148.4	146.0										
13	0	563.4	-257.3	-44.9	1122.4	-482.9	63.0	-1104.6	373.6	270.9	-367.1	1054.3	322.2
	12	-85.3	-633.7	35.3	75.4	85.2	310.1	108.0	-74.9	-84.2	1810.3	475.4	-276.0
	24	175.6	-1804.0	24.2	-416.6	-161.5	358.4	867.2	387.4	-324.1	221.8	-177.1	1104.5
	36	4295.5	-197.5										
14	0	-676.7	-1019.8	-588.4	1802.9	-685.8	1040.7	1576.5	-2299.7	114.9	3809.8	-638.3	-1171.6
	12	-1714.0	129.7	340.4	-371.3	-1332.5	-217.9	-169.0	731.8	595.7	-342.4	442.2	-200.4
	24	13.1	176.3	-376.6	-266.0	-47.0	-374.4	172.8	-204.2	-149.5	926.3	214.3	-18.6
	36	-106.3	-210.1										
15	0	-46.8	-328.7	525.3	-452.1	-511.4	646.3	-178.3	120.6	520.6	-741.6	-31.7	-90.5
	12	582.2	6718.6	206.2	230.0	-18.6	38.3	-17.9	-31.7	153.8	455.4	-402.7	55.1
	24	-103.8	-19.5	-77.0	-430.7	-108.3	-299.4	-216.2	50.9	-272.8	-357.8	-143.6	133.1
	36	263.1	125.6										
16	0	1487.2	-1367.6	530.2	-466.1	-276.5	-867.1	-1227.3	486.6	-1282.7	1618.7	-1085.6	4436.4
	12	1033.3	350.1	1601.4	-886.6	-355.9	429.1	407.0	-361.5	57.8	504.4	179.4	-14.4
	24	690.5	-136.5	-252.8	-164.6	248.9	-180.0	-1077.2	-855.6	719.1	112.8	431.0	-324.2
	36	294.3	363.4										
17	0	457.6	635.8	-743.8	-172.9	432.9	-580.8	-67.1	92.8	456.1	469.3	700.9	-295.4
	12	-6327.6	-198.3	-41.2	-56.9	49.7	239.0	200.1	73.1	-383.2	-638.1	238.2	-68.9
	24	-60.9	261.1	-226.4	327.7	-3.5	-40.4	-232.8	-4.3	140.4	22.8	-393.3	-68.1
	36	-21.8	337.6										
18	0	58.8	238.9	-255.5	-1004.0	-414.4	169.3	570.4	-211.6	484.4	493.2	-6631.0	-190.7
	12	-252.1	235.3	-224.8	334.5	412.7	637.2	-349.5	-43.9	189.6	-479.7	154.8	82.6
	24	55.6	-294.0	407.0	143.6	-184.1	39.9	-82.8	-95.2	-174.5	-155.3	66.4	414.6
	36	-50.7	-68.5										
19	0	373.7	-3323.6	384.5	-3826.7	-602.5	864.1	-1589.7	-304.3	-109.6	-1342.0	1065.7	-172.1
	12	-689.6	-273.0	-26.6	-194.1	129.0	475.7	109.8	511.0	-64.9	-214.9	72.9	-304.3
	24	-216.0	300.5	-236.0	-74.8	317.9	-14.9	164.5	263.3	41.5	53.2	60.3	-410.9
	36	290.9	246.1										
20	0	-903.0	-1815.2	-561.7	-497.4	99.0	756.0	-193.8	-5638.0	374.3	-703.8	598.3	-574.9
	12	217.9	-35.3	-21.6	-928.8	-191.7	276.4	212.1	58.1	468.3	-492.4	66.9	-436.7
	24	146.6	-594.7	53.8	388.4	-93.8	288.1	199.7	-166.7	419.1	-369.4	-8.2	49.7
	36	-11.2	-23.8										
21	0	-254.8	-506.3	-376.1	557.3	853.0	-504.3	-158.2	-506.9	967.2	540.1	252.4	-723.2
	12	-1344.0	2985.7	399.5	160.1	195.4	1777.4	-702.7	-80.3	82.3	-743.4	218.1	-277.0
	24	282.9	476.7	-384.1	133.3	38.1	3687.9	-464.2	-46.3	-74.3	-1034.3	127.4	412.8
	36	125.9	-160.7										
22	0	321.0	-589.5	-665.7	745.5	-1006.9	-5.9	96.8	-77.8	-6094.5	-664.2	493.9	-6.9
	12	325.9	-224.9	-206.4	-143.1	183.0	-253.4	-229.5	32.5	-79.1	372.4	-173.3	-228.3
	24	75.4	-365.9	58.6	-575.9	19.6	-327.1	-37.4	-145.1	-65.2	-229.5	912.0	66.7
	36	-390.6	45.8										
23	0	-182.1	-6967.7	-115.4	-197.2	-224.6	245.6	378.6	313.7	601.8	72.7	-213.9	470.5
	12	370.4	327.7	-150.2	-317.8	1.9	712.3	-83.8	-253.9	294.9	-157.3	16.5	86.1
	24	-206.2	205.5	-161.2	28.9	77.4	53.5	70.0	-104.8	145.4	-8.8	7.8	-199.8
	36	267.5	-114.7										
24	0	-958.3	-965.2	1512.2	-1815.1	527.1	-625.5	-144.5	-190.7	-1893.7	-693.8	-2701.2	-219.8

	12	-356.6	-14.8	-165.3	-363.1	-1441.8	-474.9	205.5	315.0	-992.3	230.7	121.8	14.1
	24	3465.3	59.5	170.4	421.2	561.4	-165.0	152.7	171.7	-564.2	-46.9	-62.2	383.4
	36	-270.3	66.6										
25	0	2819.0	-2811.6	-39.8	373.6	-229.0	-60.0	307.2	-53.7	-683.0	788.8	245.6	-391.4
	12	-311.1	390.0	394.5	275.3	-3851.8	83.0	-921.5	-367.7	-239.1	136.5	40.3	125.9
	24	-548.4	871.4	-279.6	-281.9	-342.8	-263.8	-323.6	330.0	117.3	187.7	31.3	-203.2
	36	-2.2	-113.9										
26	0	851.8	76.6	540.6	1600.6	-1048.8	21.9	-489.2	-99.7	1456.3	-162.4	-3574.5	-426.2
	12	-2457.5	-2109.0	-637.9	-414.4	73.1	-227.7	508.2	-280.4	-117.9	16.0	-15.6	-1134.1
	24	-373.0	-755.6	83.3	8.4	356.0	-274.3	292.2	246.9	-727.7	-90.3	-30.9	-639.1
	36	-208.0	-42.6										
27	0	-681.1	-2815.3	2273.7	479.2	-85.4	2533.8	839.0	-390.5	-743.6	246.1	1318.4	-2339.5
	12	-324.3	-768.8	-674.2	-26.1	1084.8	-56.6	113.9	-830.2	471.4	1050.3	728.4	-304.3
	24	710.6	133.4	-259.6	327.7	-39.8	-867.5	-318.4	-190.8	9.4	-534.7	100.6	402.7
	36	-422.7	-145.0										
28	0	-573.3	-3139.9	-260.9	-102.6	3289.5	-202.0	-265.1	-736.2	606.1	-11.1	-376.8	230.0
	12	-471.9	-155.2	69.5	494.6	3487.8	353.7	12.6	-173.6	-334.2	-276.5	-119.0	17.2
	24	0.7	64.1	-666.9	260.8	-407.0	-187.4	257.7	-447.4	83.4	-131.5	295.2	-94.0
	36	-44.7	-393.3										
29	0	83.8	-124.3	-277.1	72.5	-99.8	-70.7	-0.6	-95.0	-30.9	-165.9	-220.2	173.0
	12	27.8	266.5	20.4	158.4	189.3	254.3	36.1	-84.3	-190.1	-9.4	352.9	-281.0
	24	224.7	267.3	3.2	-98.8	-199.8	86.3	-26.9	-7525.0	-85.3	38.2	-36.6	-34.9
	36	82.5	123.2										
30	0	-45.1	230.9	824.4	247.5	158.7	-241.2	426.9	-518.8	394.4	-6771.5	-86.5	25.8
	12	-818.2	265.4	-74.8	-112.0	266.7	-700.4	61.6	200.0	99.8	-698.2	-61.4	-61.1
	24	77.1	-386.1	284.0	202.6	-107.4	-72.0	-120.8	153.8	206.4	301.0	-212.6	-314.2
	36	41.7	120.9										
31	0	16.9	372.2	-262.4	208.9	-96.7	323.3	-506.2	-202.2	301.3	728.2	77.1	69.0
	12	752.8	-72.9	249.8	-230.6	223.3	4.0	-80.3	53.7	-20.1	235.2	48.3	-6495.3
	24	-25.0	347.1	-145.0	477.2	-22.5	-23.0	16.3	-446.1	0.7	222.6	207.9	-63.8
	36	-472.4	128.6										
32	0	-1013.1	-513.1	798.0	2344.6	936.0	865.9	-1788.4	-2115.6	-2241.4	63.8	-1707.4	62.3
	12	-317.2	-1489.6	1225.3	-1051.9	-439.4	1073.3	59.0	-417.0	356.4	-569.9	-690.3	398.0
	24	203.9	276.6	198.3	-63.3	-499.7	-385.9	167.8	-10.2	-61.0	86.0	-1611.3	99.2
	36	105.5	713.6										
33	0	-2013.8	91.4	-572.9	-3815.9	-1349.3	1076.3	-26.1	2468.1	45.6	883.1	-1346.0	706.7
	12	-1002.3	641.5	74.8	253.2	187.6	1159.6	808.8	438.0	301.6	-324.8	-87.3	-185.4
	24	-30.4	754.0	-271.6	157.8	-448.2	-188.9	511.8	-116.7	92.6	148.2	-263.9	-196.2
	36	-16.0	612.2										
34	0	1473.4	420.6	390.8	-1565.6	1895.3	-1959.9	-2041.5	-2608.4	1331.7	-81.5	1120.0	410.5
	12	-490.6	778.6	-2222.4	248.0	1176.4	-563.9	556.0	-186.1	119.0	619.9	-4.0	-599.0
	24	144.9	582.9	348.3	-20.2	-828.5	-390.7	232.0	-15.8	173.0	3.3	-285.3	513.6
	36	109.1	537.6										
35	0	390.6	-859.5	-432.0	199.7	205.4	-676.4	200.7	-195.0	-366.3	217.0	-45.2	-7328.2
	12	-43.0	112.5	-47.6	114.0	-458.6	44.2	361.5	-173.2	-484.1	-3.7	128.9	-84.1
	24	-103.7	16.5	-152.2	-322.8	106.1	-274.6	577.7	-75.7	264.3	210.5	170.7	-127.0
	36	-219.9	189.9										
36	0	-2172.3	-151.5	1751.3	-3410.1	632.2	307.3	-560.1	-2036.2	176.0	1352.0	-668.5	-158.4
	12	812.8	-579.2	1320.5	621.2	108.6	-178.1	221.1	-319.0	480.3	2132.7	-444.7	518.3
	24	-461.1	-499.0	-147.6	-1192.9	-208.1	454.5	-604.5	514.7	51.0	-173.2	42.7	-18.2
	36	307.1	664.1										
37	0	1435.2	-1074.0	-1391.8	-726.1	304.6	442.1	-377.8	-2539.7	741.5	-984.2	-2921.8	-1646.7
	12	1995.8	832.1	1530.0	-45.4	901.5	-1249.9	-598.5	-96.3	765.7	438.7	-6.2	-517.4
	24	-114.8	-318.9	-541.8	-298.0	274.9	78.1	474.7	230.8	-60.4	-180.4	260.4	-37.4
	36	-205.8	-42.0										
38	0	512.7	105.7	196.7	-8140.2	-851.3	-678.7	-294.5	-81.9	228.9	36.4	-506.6	-223.2
	12	395.8	79.1	190.5	99.5	114.8	-21.7	-153.9	21.8	38.5	-114.9	71.7	93.5
	24	-333.5	356.8	-136.6	-52.4	486.6	316.5	-279.4	-25.2	-72.7	-111.7	65.1	246.9
	36	58.5	-215.0										
39	0	41.9	202.6	1117.0	-1901.6	-612.8	-1611.9	2232.0	-735.4	-4113.9	1704.9	176.6	-880.8
	12	598.0	-85.8	788.6	-1003.4	163.9	139.1	-443.4	-325.3	-489.0	-1287.6	-668.7	-204.2
	24	-383.7	-773.2	154.4	110.1	-11.2	454.7	494.7	243.5	207.5	185.7	121.3	-180.2
	36	7.9	-113.3										
40	0	300.7	430.4	82.2	304.0	-48.7	-491.4	242.4	-679.9	-673.9	520.7	-247.2	-136.7
	12	-73.2	-629.0	60.9	-566.3	195.2	338.3	153.0	-135.6	158.6	-303.3	8192.0	-69.1
	24	-176.4	217.7	326.2	225.8	225.4	316.7	-724.6	-53.3	492.3	-78.3	-404.5	-292.8
	36	152.4	233.3										
41	0	271.1	3498.3	-284.2	-3779.0	54.5	774.5	766.0	-560.8	-238.4	372.6	630.8	-103.2
	12	-389.9	-775.6	-1348.2	-1095.0	61.6	-678.4	-518.5	149.2	472.1	-427.9	-134.3	-154.6
	24	358.9	-332.3	-481.6	311.0	385.6	228.2	59.9	-311.2	38.4	205.7	-425.1	-74.2
	36	306.3	-17.1										
42	0	354.5	1634.4	1014.1	-495.0	4903.4	-1701.2	1115.8	-540.2	-385.4	59.2	-367.4	567.6
	12	1131.6	-276.1	-496.8	303.1	-438.6	917.9	196.8	48.5	-258.6	-179.8	-289.8	-100.8
	24	388.2	66.4	-113.7	-157.5	206.9	-311.7	-351.2	175.7	-87.9	199.8	-199.9	103.3
	36	41.9	-537.6										
43	0	21.0	751.9	-167.5	-587.6	-585.8	-5113.0	-844.4	-239.0	1764.2	-319.0	931.6	-1789.1
	12	977.2	-913.7	356.9	-392.7	-827.5	88.6	-65.6	413.3	-633.9	-309.3	497.6	26.4
	24	13.5	-234.3	-148.0	285.2	319.3	118.7	2.2	-386.6	6.8	5.0	34.3	124.5
	36	22.4	201.5										
44	0	2972.3	-668.6	-574.8	-1676.6	-3566.5	-1230.2	599.6	-1404.3	-206.7	-21.6	60.6	356.4
	12	-745.8	1289.8	-1007.4	-506.6	273.4	1000.5	554.3	-625.9	167.3	674.6	71.5	-199.4
	24	-56.5	196.7	-1.5	255.3	-408.1	-80.1	-1.0	169.4	70.0	-246.2	190.0	-95.6
	36	-620.7	-73.8										
45	0	-788.4	1272.6	-205.6	-1619.3	-2666.9	27.7	-3228.7	-2748.4	-100.8	-650.5	-954.5	1063.5
	12	-940.6	841.4	-59.7	-89.8	181.3	961.8	-396.0	-235.2	16.6	170.8	45.1	363.0
	24	23.9	935.6	-90.6	-261.7	1031.5	476.9	-371.6	-77.1	-14.1	-4.7	107.5	116.5
	36	-255.7	-51.9										
46	0	-168.7	416.6	583.3	278.6	1234.4	473.8	147.2	-284.1	-1278.6	118.6	461.3	-751.3
	12	-195.9	-282.6	-400.2	-134.8	188.3	150.8	400.1	-65.9	395.0	89.6	271.9	-81.9
	24	23.3	17.2	-57.7	-175.5	224.8	106.6	-37.6	-57.9	5861.1	300.7	-394.7	97.4
	36	-101.6	2.7										
47	0	-259.1	-183.3	688.2	175.6	-109.5	362.8	308.3	145.0	83.7	-414.2	-151.4	-39.1
	12	-196.1	98.6	-186.0	-546.9	346.7	7.3	93.9	23.9	102.4	-720.4	154.4	127.0
	24	103.2	526.3	-221.8	-498.1	97.7	175.4	-142.4	86.1	-198.0	6408.1	-0.5	411.4
	36	157.4	-124.5										
48	0	-28.4	109.4	-51.8	-186.2	79.4	-128.5	26.4	158.8	241.8	-161.2	134.1	74.5
	12	-174.0	-234.8	-86.0	-164.1	-68.1	-65.8	-72.7	-6793.4	-147.6	-19.3	-4.8	-157.7
	24	-102.5	182.0	-135.3	-169.9	59.6	79.7	-163.0	-18.5	2.1	13.3	139.8	-100.2
	36	25.3	-68.4										
49	0	-1619.4	-1116.6	3160.2	634.5	1355.8	-2860.6	493.2	-531.3	0.5	284.3	164.2	344.0

	12	-1322.0	354.2	532.0	-100.7	484.0	-315.7	214.5	-328.7	2622.2	-635.2	521.2	254.1
	24	-92.3	3.2	33.2	-354.3	79.5	247.4	67.9	213.2	112.5	-60.8	34.4	-252.4
	36	39.2	72.6										
50	0	-189.1	-1063.9	397.4	-1068.3	1231.6	-289.3	-6340.0	-528.0	-583.5	143.0	-260.4	-254.5
	12	-938.1	402.2	303.3	738.0	-181.8	635.5	-19.4	90.9	364.6	-418.1	118.4	-220.2
	24	134.6	-42.4	-157.6	17.8	-3.9	-583.1	-312.0	87.6	119.8	157.1	209.5	55.4
	36	15.8	271.3										
51	0	850.1	-135.9	-1614.0	-22.5	-83.0	-446.0	2154.9	-2409.6	849.8	476.0	358.4	3255.0
	12	-1619.1	667.1	651.7	-771.0	1191.9	-640.5	-27.5	-268.0	-74.4	2578.9	-413.5	768.9
	24	23.7	-44.6	-144.3	76.7	755.9	-231.0	584.5	7.4	-128.4	109.6	63.0	-346.4
	36	85.0	168.7										
52	0	-771.9	-1168.5	-47.8	-228.9	177.5	-779.1	2114.2	-381.0	3591.9	-1284.1	278.4	-974.1
	12	3041.6	-294.2	276.0	85.1	-110.7	1306.0	321.1	279.9	52.8	-242.8	104.6	-24.7
	24	544.9	-299.3	-735.8	404.9	-109.5	-241.1	-233.5	-206.6	-56.5	-94.1	-249.9	-97.5
	36	-81.1	-161.4										
53	0	186.5	-1143.2	-635.1	1015.2	-259.1	-5831.5	881.7	-256.6	-403.0	315.2	-752.0	1653.3
	12	-433.5	133.1	-497.0	758.5	-20.9	-1.5	277.2	-38.5	242.1	-571.9	231.6	187.2
	24	-82.3	467.6	0.1	4.6	-346.1	-111.6	53.2	69.5	-233.0	315.7	77.8	310.8
	36	-469.0	-316.1										
54	0	144.1	-834.0	-262.0	297.4	8192.0	410.6	-123.5	-593.5	-378.1	153.7	-43.1	-329.6
	12	-283.6	-154.1	47.2	-91.8	-261.1	-325.4	-199.4	-165.9	-373.0	480.6	128.3	163.8
	24	-266.4	-124.5	349.6	-70.2	-121.6	34.9	108.9	-142.5	-257.5	97.4	-23.9	87.2
	36	18.5	-396.5										
55	0	-6809.0	57.9	235.4	-627.2	209.7	40.9	-247.1	-66.8	-113.6	-132.0	-117.2	-555.5
	12	245.7	228.1	35.2	121.8	-492.4	282.5	9.9	39.6	-81.4	-189.2	-69.6	162.2
	24	-1.4	22.6	-81.5	-169.6	-226.2	-44.2	15.3	194.9	171.9	-124.3	-35.3	-62.1
	36	-120.1	-172.5										
56	0	-653.9	1806.1	1455.5	908.6	-2989.2	-573.2	1437.5	-2944.1	2092.1	-343.1	-486.2	-675.2
	12	278.9	248.4	-606.2	1276.8	367.6	27.4	-76.4	-578.2	290.6	1490.9	467.0	-102.5
	24	38.3	-761.9	90.9	67.0	-423.8	129.8	80.9	-418.4	-234.9	-244.1	-97.8	376.3
	36	-141.0	60.7										
57	0	502.8	753.1	-429.2	-501.9	376.8	-92.8	-1367.1	193.4	-23.9	815.7	-1992.1	227.5
	12	133.0	-239.9	-1665.9	-211.2	891.0	363.0	-833.7	-215.2	-525.2	286.1	-149.4	382.0
	24	-28.8	48.5	-4889.8	158.4	49.3	-348.7	-295.5	-10.9	119.1	-2.9	92.8	34.8
	36	-4.5	-472.9										
58	0	-265.5	-1674.1	457.9	159.4	-529.4	-14.8	-1137.4	-1253.2	627.3	-681.0	1431.8	3576.2
	12	990.6	311.6	-2281.8	403.1	-1964.5	-36.4	79.9	10.5	274.0	-1604.3	571.9	-255.8
	24	154.1	95.0	-428.1	615.1	-450.2	-341.2	75.1	280.6	236.5	51.1	-328.2	742.7
	36	-76.5	18.3										
59	0	-777.8	332.9	-3644.5	1758.0	-1425.0	-731.2	-728.8	-242.4	-1699.9	450.1	-824.9	-1605.4
	12	1012.5	1911.6	-274.4	477.6	91.7	-260.8	881.8	-594.7	481.8	-1235.6	-534.9	598.8
	24	-195.8	-76.9	-170.2	-274.4	-293.6	147.2	9.0	89.7	-139.7	142.6	352.9	295.5
	36	332.5	-29.1										
60	0	-21.8	-769.6	-150.9	-650.9	265.6	-425.7	998.9	576.2	-83.8	349.9	-776.6	-172.8
	12	330.0	594.7	-4266.1	1112.7	-3896.4	-382.5	866.2	350.9	-44.7	103.6	-454.8	262.5
	24	-337.2	-344.2	47.5	-305.3	-302.9	125.4	448.3	-301.2	26.6	878.8	298.2	-94.3
	36	163.3	-353.5										
61	0	529.2	-320.6	-7180.9	-49.7	-168.9	-202.6	593.0	33.3	-313.0	-18.4	433.2	-14.9
	12	-14.6	321.0	902.7	-235.4	574.4	154.2	236.5	189.1	55.7	-40.5	-204.0	-173.1
	24	-91.8	121.5	30.5	268.3	166.7	74.3	310.2	83.0	85.7	162.0	155.5	200.2
	36	121.3	553.2										
62	0	-378.2	-203.5	893.9	1501.3	-154.9	-510.4	-259.1	-575.4	-714.5	662.3	-527.1	-410.1
	12	680.6	390.3	-5496.6	-444.0	416.0	-837.8	-133.8	-457.3	502.3	286.6	-376.1	-222.4
	24	-241.5	-510.0	330.7	-4.4	-357.0	-28.4	9.1	-179.9	-366.2	-201.1	23.9	-57.6
	36	-47.2	-25.7										
63	0	-1526.9	1118.0	-3911.4	-773.5	3142.0	759.4	-1211.7	-641.4	-22.1	331.7	91.7	71.7
	12	592.7	-221.7	-208.2	-92.4	-271.2	-117.9	-512.0	-592.8	321.6	1468.1	-55.2	8.7
	24	516.2	589.2	-637.2	-25.0	120.5	71.7	23.2	-63.7	-370.9	244.2	133.9	190.1
	36	-233.7	-302.0										

Table C. 21 Shape codebook 1 for MODE_VQ == SCL_1 / SCL_1_960, BLEN_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-3187.0	689.6	-55.6	-410.6	143.5	-4328.0	211.2	-145.3	-323.0	91.2	-40.5	29.3
	12	89.9	6.0	61.3	-30.8	32.1	-178.3	144.3	-117.7	-53.2	50.2	100.3	-80.9
	24	84.2	78.6	482.7	-30.8	104.1	55.0	-24.1	242.9	-165.5	-263.0	-38.6	-32.6
	36	53.6	42.4										
1	0	3205.2	-682.3	-771.5	403.0	195.0	150.8	-4894.5	65.0	82.6	262.5	368.2	48.8
	12	479.8	33.7	-168.8	306.2	430.4	-131.5	259.4	-121.3	-62.4	-464.6	-27.6	-59.6
	24	146.3	44.8	158.8	-20.8	-102.6	28.6	-46.7	-100.4	-120.0	-215.5	126.7	-281.5
	36	104.1	629.9										
2	0	339.7	323.5	3373.0	273.4	-239.5	407.7	-31.4	-4351.3	-308.1	54.2	-16.2	-33.9
	12	-117.2	-36.8	300.5	191.0	119.0	-164.9	151.2	185.0	-99.9	45.3	-257.8	-329.4
	24	17.5	-26.4	372.0	110.2	42.8	114.7	-69.0	-3.3	62.4	28.8	43.1	41.0
	36	50.8	-53.0										
3	0	554.1	2071.0	111.7	-67.4	-105.0	-202.9	49.9	175.4	-91.7	269.9	160.8	-36.3
	12	-7175.7	-196.4	-12.7	-54.4	-389.7	-56.4	-28.9	-156.3	176.2	-134.9	-40.1	150.0
	24	-55.3	209.4	77.2	64.4	-130.2	-96.8	151.8	142.4	-194.3	-3.7	-137.6	126.3
	36	-74.2	-297.3										
4	0	-66.4	-785.3	273.7	-312.2	524.9	404.0	-47.7	14.4	-489.1	-8192.0	195.0	224.9
	12	86.7	-490.8	-122.8	118.4	-341.9	473.0	-147.5	193.5	34.4	18.9	79.4	-170.5
	24	-9.6	215.6	147.8	-94.1	-149.1	346.8	71.2	-209.0	-310.5	98.3	-87.6	193.2
	36	128.2	536.4										
5	0	-1435.6	-946.6	-604.0	112.2	-211.2	-20.8	106.7	20.7	-425.9	17.6	29.1	7789.5
	12	39.5	-188.7	-58.5	-97.4	-17.0	-339.2	712.3	-215.7	-51.9	-533.0	61.8	-198.5
	24	-198.5	153.5	-54.1	-105.0	-367.5	-61.3	92.5	-41.3	-61.9	-179.6	16.0	-726.8
	36	-23.8	31.7										
6	0	304.9	989.1	-1703.5	440.8	-382.0	117.0	184.0	-7257.2	-177.3	-15.6	89.0	105.8
	12	-154.0	7.5	131.3	38.7	-91.7	189.9	69.7	-70.3	10.3	-140.6	89.2	58.7
	24	221.3	13.8	86.3	24.0	-42.3	88.8	64.1	159.6	30.6	-279.5	-104.8	393.4
	36	-0.2	7.0										
7	0	2098.8	-535.1	2464.6	-795.3	-346.2	-301.7	468.8	-9.2	1567.6	38.9	461.4	294.3
	12	239.8	-2723.4	-322.4	143.5	184.1	-303.3	320.0	65.3	-43.4	-224.4	-114.6	-148.1
	24	-69.9	-109.6	-21.0	104.8	-117.8	14.9	-48.8	-241.7	-199.6	-25.7	216.5	270.6
	36	-68.9	126.2										
8	0	-517.5	-32.2	-702.5	-439.8	-102.3	352.2	64.8	263.6	-48.6	41.6	-8192.0	200.3

	12	300.4	182.1	79.1	20.2	-118.3	-295.2	104.3	-85.3	94.5	44.3	-56.2	-126.8
	24	-118.7	22.8	554.6	-186.8	17.7	67.2	78.8	-235.9	-72.5	-15.5	-15.7	-69.1
	36	63.8	-14.8										
9	0	-136.5	387.3	520.6	-565.5	24.6	-40.1	-129.5	654.1	184.4	11.3	-250.5	-322.7
	12	224.7	-93.1	-17.5	-6287.0	100.4	-319.6	483.9	97.1	1.9	108.7	165.8	-236.2
	24	-6.1	6.7	30.4	27.9	70.1	95.3	61.2	272.7	-90.0	-120.5	85.6	37.1
	36	-41.0	34.7										
10	0	955.4	1530.6	68.1	-1304.7	-1616.9	915.7	3695.6	222.3	-397.6	163.8	-768.4	-66.1
	12	-573.7	-18.4	-362.6	248.3	-70.8	-350.8	738.6	-269.3	47.2	87.6	-758.0	-221.3
	24	-147.0	88.9	267.6	77.7	-35.0	-71.5	-163.2	199.4	139.7	-162.9	222.3	60.9
	36	-54.0	297.9										
11	0	355.6	-3564.6	-465.6	-268.7	371.9	184.8	-56.8	-4404.6	77.1	204.8	-46.3	-105.1
	12	22.8	-126.4	-211.6	-276.3	-213.8	-146.4	37.7	-38.5	-70.7	-217.7	38.7	-40.5
	24	-1.1	189.5	-78.0	73.9	-23.4	-49.2	-130.4	-153.3	192.0	17.3	206.5	114.4
	36	76.4	41.5										
12	0	2938.6	-111.6	-299.4	139.2	-377.0	-435.6	-141.5	-466.0	-357.0	-4243.7	-17.0	-110.7
	12	565.8	218.1	-85.0	318.7	260.0	-143.5	180.7	120.8	156.3	-14.5	-133.8	35.8
	24	320.5	58.0	-54.8	205.8	-168.0	-30.8	-23.8	-22.3	-145.1	83.0	135.8	-352.1
	36	14.2	-371.0										
13	0	-103.9	592.7	519.9	104.5	188.0	50.9	-444.5	914.8	165.7	-28.2	250.6	562.2
	12	-579.0	-6.4	-219.6	-302.3	130.6	132.2	108.9	199.9	-48.9	648.1	-67.5	-3.1
	24	-102.7	-7844.5	-8.0	-163.3	191.3	-64.3	115.4	-33.6	140.7	53.9	32.2	-194.7
	36	-265.4	-64.6										
14	0	1706.2	-1136.6	1024.1	7.2	335.7	637.7	191.0	254.9	-244.4	7827.8	41.6	-60.8
	12	-36.1	-147.5	173.6	-31.6	39.8	-49.5	198.1	137.3	-113.0	-9.2	49.5	-293.6
	24	-303.9	134.1	46.7	-138.2	-282.2	-204.1	237.1	-67.3	32.6	334.2	-50.5	166.3
	36	-94.9	288.4										
15	0	1189.2	-378.6	-888.1	-199.0	-20.9	-4208.4	1450.9	504.4	318.5	-127.6	136.9	-32.2
	12	-272.5	-518.2	-170.7	128.3	-727.2	-312.0	373.5	75.1	47.0	-119.7	112.0	46.7
	24	-54.1	189.9	535.8	-709.2	-31.4	194.4	-251.0	-47.9	277.8	137.8	-252.3	176.3
	36	60.4	65.8										
16	0	-540.4	223.5	-808.3	362.2	36.3	15.0	600.3	320.0	188.8	99.3	-92.5	-179.3
	12	104.8	-12.9	50.4	-208.5	-13.8	61.1	-7440.4	-63.2	-0.6	7.4	22.6	32.7
	24	41.3	87.1	-28.1	47.3	-3.7	-51.8	140.1	215.9	-37.6	-69.6	-12.6	-87.5
	36	-29.8	-85.1										
17	0	1902.5	-490.9	3027.3	630.2	3604.0	-527.1	-121.0	-132.1	-2.4	47.8	125.0	-144.3
	12	-7.9	212.0	58.6	-99.6	-31.1	-114.9	234.2	201.0	-126.4	-148.1	205.3	-29.7
	24	-185.1	4.6	-433.7	-32.8	83.2	-171.0	-14.8	235.4	-252.5	-207.9	146.4	-164.7
	36	-49.7	-410.1										
18	0	-563.4	-1282.6	414.7	-209.5	-4934.3	-81.7	983.8	-620.6	-122.8	193.4	-181.0	286.4
	12	-108.8	-13.7	43.7	56.0	190.4	-42.0	193.5	-107.2	61.0	212.1	84.0	-28.1
	24	46.2	83.6	67.4	136.0	58.2	-9.1	-43.1	-301.0	-193.3	-121.5	110.0	-665.2
	36	-99.3	257.7										
19	0	276.2	1020.7	262.5	-52.0	-1316.3	169.9	-359.6	296.2	-35.4	-435.6	24.4	98.0
	12	-216.1	-149.2	198.4	71.8	2.3	-295.5	62.0	101.1	60.9	-640.6	-7205.1	-65.2
	24	-135.4	-81.8	335.2	193.9	-46.7	-6.1	-18.7	116.4	1079.7	-51.6	13.4	-121.2
	36	8.1	-66.1										
20	0	-335.5	2382.8	826.1	256.2	-209.7	317.0	219.6	-68.2	-6478.8	-229.9	-126.6	-272.4
	12	-199.0	126.0	-110.6	-409.8	-214.7	225.3	216.6	36.6	-3.2	190.2	-73.9	132.3
	24	256.6	-25.5	-83.1	23.4	-162.8	-92.5	-28.8	363.5	6.0	76.4	-6.7	16.1
	36	-23.6	-178.2										
21	0	-1592.1	4597.1	380.9	-339.0	-15.3	-485.0	329.3	-504.5	261.8	403.5	112.8	-113.7
	12	-336.6	113.0	22.8	-104.4	220.0	-244.3	-102.1	-376.0	-104.7	-176.8	61.2	-272.1
	24	90.6	92.9	-403.1	-107.0	-15.1	65.4	99.7	232.5	125.0	277.4	163.7	-173.1
	36	85.0	92.1										
22	0	92.1	-1536.1	-311.9	1699.6	-1146.5	1439.9	-290.9	-87.9	-347.7	184.4	-404.3	-411.0
	12	135.0	-149.2	-44.0	-83.9	-120.7	-59.5	-86.3	22.5	-5048.3	16.3	8.3	5.9
	24	-110.4	-9.6	150.2	-332.8	261.1	22.5	427.6	59.9	423.0	-30.8	-173.8	-120.7
	36	48.9	-30.5										
23	0	-319.6	-4593.8	4132.5	-854.1	202.6	47.7	49.8	-47.4	-181.0	159.1	-77.5	298.3
	12	265.0	36.9	157.7	31.3	-22.2	-175.9	-312.3	-176.5	3.3	209.3	170.6	-15.9
	24	-19.4	-89.3	87.2	-36.1	-47.2	70.7	-28.8	207.8	210.4	-104.7	-179.5	119.4
	36	101.1	166.2										
24	0	-187.4	-266.5	-122.2	-113.7	-715.7	916.1	-65.2	-475.5	38.9	386.7	541.4	81.2
	12	32.8	238.2	-351.8	-194.8	116.7	-112.4	229.8	-20.1	19.3	-25.6	-55.7	-91.7
	24	-22.6	1.1	-8192.0	2.3	6.0	29.7	-194.6	-151.8	-159.1	57.1	-22.7	81.7
	36	246.5	127.5										
25	0	-2142.3	-528.2	2495.9	124.0	-286.0	-234.2	-14.5	-391.2	-283.0	-3719.2	-13.8	-435.1
	12	273.1	122.7	-43.2	67.7	192.3	-27.4	72.8	-252.6	-38.3	49.9	-61.1	-289.3
	24	-119.1	-86.7	196.8	-27.7	-184.1	89.6	-98.4	399.2	-97.1	-4.7	64.4	164.1
	36	246.5	-12.1										
26	0	8.9	96.4	350.0	-1098.7	385.4	388.4	-116.7	-488.3	154.4	47.3	-54.0	-352.7
	12	206.1	-190.7	16.9	-52.8	-21.7	77.9	1.1	17.2	-37.6	-649.3	0.6	-66.0
	24	11.9	-90.6	88.1	35.1	8192.0	-58.0	-15.6	-38.7	30.4	-10.2	55.3	-22.5
	36	35.5	290.8										
27	0	-125.5	2087.6	3753.7	-3303.4	-33.7	-409.9	121.8	-44.3	302.5	-112.6	-186.2	180.5
	12	-46.5	71.0	-137.6	98.5	-143.5	-281.6	-8.4	143.7	-95.9	-139.5	-183.3	-70.0
	24	-38.6	-58.1	776.2	241.0	142.3	150.9	236.0	-218.8	-140.0	-243.3	33.9	337.8
	36	-101.4	533.9										
28	0	1160.1	8192.0	-912.5	264.5	236.9	197.6	-496.5	24.7	-42.2	-54.1	-34.1	113.8
	12	44.1	-313.5	119.2	61.8	-319.6	260.8	-129.1	355.8	25.8	-1320.4	90.9	-280.2
	24	-59.4	-19.2	-263.6	-51.3	68.5	150.2	0.3	-98.1	91.4	-217.8	-23.6	99.1
	36	-13.7	-26.4										
29	0	-397.6	-2179.8	972.4	240.7	549.1	298.6	-502.3	710.4	-178.2	61.0	289.4	632.5
	12	-261.9	-65.0	126.0	265.5	-132.9	-322.1	118.6	-121.4	20.4	95.1	-71.9	5.6
	24	-112.0	-52.1	225.7	17.8	-185.7	-87.2	68.1	-218.6	89.6	-103.0	37.7	-271.6
	36	-4904.8	-225.1										
30	0	4115.2	2696.9	-336.0	135.2	-484.9	93.2	-154.1	51.3	-212.7	118.1	140.4	-71.1
	12	78.1	-143.9	26.7	418.5	53.5	138.6	157.3	-112.7	164.2	26.2	-7.6	-45.7
	24	-53.1	-11.4	308.2	-18.7	-16.2	69.3	26.5	-123.0	-110.1	-168.8	149.1	170.6
	36	98.4	195.3										
31	0	-994.8	255.7	-6816.0	-255.7	-25.2	269.4	-18.5	200.1	-160.9	176.6	117.5	154.3
	12	-34.7	12.3	-380.7	-45.6	485.3	-207.7	185.8	-161.0	-101.3	-155.5	112.7	-223.8
	24	-170.4	-251.5	244.0	239.1	24.0	-171.3	-254.5	-152.7	-132.8	61.4	-377.2	-211.9
	36	-134.2	-456.1										
32	0	-1121.3	-455.1	48.8	-634.5	6364.8	-9.8	372.8	38.4	184.4	26.7	251.7	-178.9
	12	-54.8	21.1	-121.4	4.3	-106.6	-66.7	374.7	-291.1	-84.4	432.5	58.7	-296.2
	24	164.6	54.2	3.2	-38.4	-226.1	-36.1	-167.6	79.0	-202.2	-129.1	98.3	-0.2
	36	-25.5	252.8										
33	0	-192.3	156.1	3962.5	418.9	360.0	4098.3	403.1	1024.1	325.3	84.9	-172.1	-141.4

	12	-232.7	-141.5	-33.4	-139.3	1.1	-41.4	113.7	44.8	124.9	330.3	200.7	209.7
	24	-3.9	198.5	310.7	-1.0	175.0	98.8	113.0	209.5	-66.6	301.6	122.8	-123.3
	36	106.9	262.3										
34	0	238.0	-2693.7	116.1	81.7	175.3	191.4	-72.3	197.1	326.1	180.7	9.6	-5291.9
	12	36.5	72.7	59.2	197.3	115.9	-106.9	391.5	113.0	136.7	817.5	14.6	287.4
	24	152.4	-30.6	167.8	-6.7	-89.3	151.5	-200.0	120.5	10.2	-105.4	-116.7	211.4
	36	-65.6	147.7										
35	0	351.2	-619.5	224.3	-547.3	-2715.9	-31.2	-321.3	4557.7	-602.2	-1450.5	-114.6	459.0
	12	-190.0	48.2	179.4	1.5	147.4	-90.0	-156.9	-322.8	156.3	-108.6	-129.3	140.9
	24	7.1	24.9	194.0	-37.8	108.7	13.6	-232.0	-12.0	-138.8	-118.2	109.4	246.6
	36	-121.8	-30.0										
36	0	1060.7	-7.7	-3827.4	-1332.8	767.5	121.0	-146.7	-304.8	1012.3	-376.5	378.2	-1481.2
	12	221.1	-88.5	-171.5	118.8	-624.6	-17.7	343.8	191.9	82.7	-49.8	-24.2	-165.9
	24	96.6	-78.3	-354.1	-129.4	-360.0	67.7	124.5	-234.8	-384.1	267.4	-47.5	-357.0
	36	-193.7	1847.5										
37	0	448.8	1390.3	1413.3	246.4	97.4	-117.3	-70.6	-10.9	-4.8	272.3	-35.6	209.3
	12	5394.6	106.2	-139.2	-170.0	-267.9	110.8	-60.5	-19.8	13.3	-91.9	176.6	-22.9
	24	9.1	68.5	60.8	-31.3	-457.1	8.2	-41.2	66.2	-36.0	-65.4	32.9	73.2
	36	32.1	-141.0										
38	0	-532.8	1234.3	-428.8	2500.7	461.6	-1289.2	279.2	128.9	252.3	-395.4	20.8	-4485.2
	12	131.8	88.7	92.9	17.5	472.5	-84.8	375.0	-143.3	-19.1	-427.5	-0.7	-339.2
	24	-75.3	-202.6	171.3	142.8	-263.0	-115.6	138.8	-139.3	-89.0	-237.8	59.6	-109.3
	36	45.9	-298.3										
39	0	732.5	384.0	-398.0	401.3	-202.3	-193.6	-158.9	-361.1	-15.4	-45.5	38.5	140.3
	12	-45.0	-85.8	8192.0	87.2	58.6	-269.1	290.7	-224.3	-151.0	289.5	-126.5	-221.7
	24	37.1	-98.0	-32.1	-180.0	90.9	161.9	-7.9	156.2	-221.8	-52.7	37.7	-102.5
	36	-84.2	-26.7										
40	0	293.2	-117.9	-2330.9	158.5	1465.0	-256.6	-12.7	-369.1	-4852.3	-360.6	221.9	-29.8
	12	139.8	-208.9	-20.9	216.4	-35.7	-108.4	287.9	-59.4	51.9	-91.6	72.7	-139.9
	24	-54.5	-114.4	244.7	140.6	102.8	21.5	0.3	152.3	-63.3	3.2	43.7	286.0
	36	81.0	70.6										
41	0	-986.6	-90.2	2462.6	-610.2	706.6	-720.8	-4937.4	-103.4	815.0	222.8	-193.9	61.6
	12	-306.6	108.6	215.5	-156.5	-183.8	-16.2	95.2	-60.4	84.6	579.9	-76.5	-67.2
	24	79.2	123.1	214.7	-53.1	0.7	216.7	88.8	233.1	63.0	120.5	19.5	187.3
	36	155.7	28.9										
42	0	1723.5	1382.9	-392.8	-25.7	-108.6	-209.4	-114.5	-299.1	6172.1	-135.1	142.6	-370.4
	12	-243.2	27.8	101.4	77.2	45.2	-789.1	-3.0	-197.7	-18.7	0.7	23.7	170.9
	24	81.6	272.7	-3.4	-39.2	121.4	93.6	151.1	134.6	-96.2	-62.6	-140.0	-377.8
	36	-2.0	-70.0										
43	0	-3388.8	478.5	52.8	-2436.0	115.4	2099.8	601.9	-241.5	-490.3	73.7	262.6	2.1
	12	37.1	-64.3	-16.1	-274.8	247.1	-121.9	-120.1	-22.6	78.9	19.7	52.9	-174.9
	24	174.1	-48.2	-256.9	-47.9	-442.8	16.4	56.2	-333.3	73.5	269.3	-220.4	2647.4
	36	-4.9	-171.7										
44	0	-1242.3	-3372.9	-2607.6	-143.9	-324.4	-299.4	1969.4	104.5	-109.3	176.7	-132.5	181.0
	12	116.1	71.5	212.9	40.7	-650.7	-445.8	197.4	296.3	-4.9	786.7	-4.7	-32.7
	24	54.6	-193.6	-14.2	-37.7	-81.4	33.3	106.7	-132.6	174.2	-0.1	-338.2	175.4
	36	64.5	127.8										
45	0	1729.5	-2287.1	-1157.8	-1509.0	512.0	201.9	-179.9	372.6	-5.8	535.8	-271.7	70.2
	12	-88.1	2015.7	55.6	368.1	1755.1	-21.8	341.9	-198.6	32.5	-108.6	99.2	-112.3
	24	-193.5	131.8	1416.2	-86.5	-230.7	-62.9	-43.7	-5.6	112.3	165.8	-76.7	-213.3
	36	-115.0	-126.8										
46	0	-2946.6	-11.4	-530.6	-297.1	136.6	-53.6	-61.0	54.5	4671.8	-33.9	33.4	6.6
	12	52.8	140.0	69.4	383.9	-11.9	8.0	809.9	244.5	120.1	-7.4	113.1	17.4
	24	-48.1	32.8	264.0	-90.4	-3.7	-83.7	94.0	190.0	-159.8	14.4	320.2	-41.9
	36	-92.8	226.3										
47	0	-3506.2	398.9	203.7	3714.8	82.0	407.0	224.3	146.7	-103.8	145.4	19.4	308.4
	12	-271.0	161.4	-193.2	153.5	-29.6	-356.6	-76.2	197.9	-55.0	56.3	-76.3	99.2
	24	-91.2	58.7	-47.5	-57.4	-69.7	3.6	47.8	150.0	-80.8	25.9	123.7	-92.6
	36	-24.4	4.6										
48	0	-230.7	232.1	34.2	67.2	1766.3	313.0	6308.5	415.4	-145.8	117.2	531.4	-84.6
	12	75.0	-193.7	180.5	-51.7	-139.2	125.0	239.1	-107.8	64.1	326.7	78.0	-0.7
	24	100.9	76.3	182.5	65.5	171.7	-161.1	159.0	420.6	-32.6	382.4	133.1	254.3
	36	-75.0	144.9										
49	0	644.7	424.9	-624.7	761.5	650.0	1955.2	-142.6	5245.5	837.6	578.3	180.8	172.0
	12	-29.3	-34.0	74.2	-25.3	37.6	-99.3	229.0	63.5	34.4	-72.1	125.2	-51.1
	24	-48.0	214.3	-262.6	83.8	-113.6	-38.3	-27.7	-162.2	143.8	-52.8	-174.9	-434.9
	36	69.4	-46.0										
50	0	-648.6	-2391.0	-570.3	-221.6	17.6	6084.7	314.9	37.9	117.9	233.2	491.5	433.4
	12	58.4	-22.7	-146.7	45.3	12.7	-218.0	250.6	18.0	20.7	431.9	176.0	-108.0
	24	63.4	-41.3	-37.7	-79.3	-58.2	61.5	120.0	-80.5	-106.7	-28.6	76.2	82.8
	36	-2.4	39.8										
51	0	320.7	-3519.7	403.0	262.5	-473.5	-4641.2	-847.3	-0.7	-82.9	-19.1	-16.5	-6.6
	12	135.5	102.8	69.2	-85.8	298.9	-173.0	112.4	70.8	-238.5	-168.7	12.1	122.1
	24	-92.0	111.4	151.3	69.7	92.7	-151.1	-120.6	-9.4	-19.2	-158.6	252.0	-95.2
	36	61.3	117.3										
52	0	-866.1	-898.7	-70.0	-1149.7	-1103.1	-440.5	-160.7	137.5	152.4	-107.7	-59.3	-189.2
	12	11.3	-107.5	310.2	-162.5	14.7	6.7	390.1	27.2	30.6	24.2	23.2	6896.4
	24	29.3	158.5	-35.1	-166.0	42.3	-142.2	4.6	-875.8	224.0	-128.9	-135.3	481.3
	36	24.3	-37.3										
53	0	-253.8	668.6	586.2	441.7	-391.3	-772.2	-333.5	-113.0	294.0	116.2	285.1	-47.7
	12	-112.1	5463.0	34.5	42.3	118.2	120.1	-41.1	-167.5	-50.9	-5211.1	115.0	-14.1
	24	-36.5	-56.8	-305.2	-90.2	-266.4	-89.0	58.4	-116.9	-19.9	144.3	136.8	-83.8
	36	52.0	163.3										
54	0	213.0	-403.8	1686.7	199.6	-507.3	-1004.2	1371.2	-247.6	-762.4	-42.2	231.7	115.2
	12	-15.1	-81.9	-160.4	-291.3	-7.2	5801.0	250.2	1.9	18.5	-21.5	13.4	-181.0
	24	-81.2	204.3	189.3	-27.2	-81.1	-76.4	-105.9	-202.4	99.2	-119.3	-474.6	141.4
	36	123.7	150.6										
55	0	-7493.1	501.5	-839.5	-195.7	84.0	118.7	-339.8	138.4	-146.5	-14.4	11.7	108.0
	12	-57.2	16.3	326.8	-277.9	-344.0	168.0	-33.4	-319.0	-22.7	-975.7	17.4	39.4
	24	36.0	-44.1	-1.1	76.0	95.1	-81.8	-139.5	-325.2	41.6	-47.6	-12.5	62.6
	36	-39.8	47.6										
56	0	1090.9	35.1	-45.6	6673.4	-11.1	104.2	-58.1	-150.8	206.7	13.5	330.9	152.5
	12	156.3	-125.5	-25.9	-30.2	-100.0	-61.8	-26.2	160.7	31.8	-90.3	-30.1	12.4
	24	-104.9	143.3	423.0	78.4	238.6	-34.8	73.8	-48.0	-191.4	114.8	-26.1	156.8
	36	-12.9	89.0										
57	0	1388.4	159.9	167.9	-352.2	-291.8	715.6	-214.6	446.2	318.2	-73.0	-211.0	-248.9
	12	313.0	43.5	80.2	293.4	255.3	-20.4	-167.5	-4.4	114.7	-396.9	237.7	-314.1
	24	124.3	430.5	273.2	500.8	-177.9	-7879.1	-261.2	287.0	165.0	77.9	-369.6	-209.7
	36	63.5	-93.7										
58	0	1851.7	-802.6	193.8	1023.7	-800.1	-298.2	457.2	72.9	9.7	367.8	-128.7	419.5

	12	18.9	323.1	216.6	158.2	77.5	284.5	375.2	345.4	138.8	-892.6	-153.4	351.1
	24	308.9	3.1	473.1	15.8	-17.5	6314.5	-290.0	-48.2	61.4	-6.6	-239.0	-240.1
	36	-21.9	-117.7										
59	0	-9.6	537.3	-868.0	-3807.4	-795.6	1234.9	-3011.5	609.6	53.9	130.8	494.2	147.9
	12	-101.4	-67.0	-38.2	156.8	-882.0	264.6	-34.5	52.6	128.6	235.0	109.9	-13.4
	24	-81.9	-81.9	-19.9	-240.0	-171.1	31.2	-271.3	70.7	101.6	15.3	308.4	175.0
	36	-52.5	44.3										
60	0	638.8	136.9	357.4	2655.6	396.1	655.9	-1195.3	-382.8	543.8	4.6	-1929.9	207.9
	12	-401.3	-97.0	-2462.8	353.9	2.5	-75.0	526.1	124.3	-1.6	-249.7	98.3	18.7
	24	-205.1	141.0	-176.7	-97.7	16.7	-21.0	15.4	-86.9	389.3	-290.2	-460.2	329.8
	36	-56.0	83.1										
61	0	-68.1	504.0	-413.4	263.7	160.7	-1490.2	-1790.1	-71.2	-2315.7	3528.1	-924.9	440.7
	12	-48.8	-420.8	25.7	-138.5	35.3	-125.1	32.9	15.4	55.7	0.7	818.3	79.3
	24	-92.5	149.2	354.7	-193.7	-108.4	16.8	75.9	33.2	-14.4	96.0	219.1	183.5
	36	-46.8	-308.6										
62	0	184.3	-84.1	-878.6	199.5	2602.7	505.1	-187.0	-1300.1	392.3	-1064.3	214.2	2929.7
	12	172.4	61.1	232.3	-262.3	152.8	407.1	-169.4	0.7	70.0	1168.9	70.1	247.7
	24	28.7	-55.9	252.7	-120.4	-275.0	-113.9	5.1	410.0	-84.1	-284.1	95.8	-46.6
	36	110.7	335.8										
63	0	-650.7	993.5	-774.8	1549.2	-1301.2	-452.8	-1630.6	-327.3	160.3	-2.0	404.4	287.8
	12	-58.0	-235.6	-122.4	63.5	56.4	-24.8	403.4	-141.6	33.5	6028.2	190.6	4.7
	24	32.2	138.3	152.4	-161.1	384.0	-170.2	-51.1	30.7	-192.5	-66.9	-6.9	13.9
	36	14.1	-725.2										

Table C. 22 Shape codebook 1 for MODE_VQ == SCL_2 / SCL_2_960, BLEN_TYPE == LONG

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-676.4	1543.7	-626.4	-1180.0	-95.5	-4117.2	-961.1	-73.0	-707.3	1811.5	589.8	-854.3
	12	-1053.8	2094.3	-514.9	17.4	1036.9	-587.7	-370.1	129.8	357.6	917.4	-271.5	-26.7
	24	120.2	-170.9	-167.9	1.5	100.6	-326.4	141.2	-31.5	-10.4	-304.8	62.5	-41.1
	36	-41.1	200.4										
1	0	4292.6	-466.6	-841.2	136.5	159.3	252.8	-3214.3	-48.4	334.1	-1476.3	-683.8	39.4
	12	679.6	-394.1	-272.9	-208.5	-219.9	342.0	50.9	401.1	-400.9	-401.2	266.4	-283.9
	24	-160.8	-423.5	-143.9	210.8	-76.0	-161.3	113.1	-546.8	-32.8	63.7	144.3	-159.9
	36	300.2	156.7										
2	0	313.7	-208.3	321.5	546.4	919.2	594.5	198.8	-404.3	-319.7	-634.1	-61.3	446.3
	12	457.9	-513.0	335.7	-679.2	-6711.2	109.8	238.3	148.4	-91.1	40.2	-109.3	26.6
	24	18.3	149.4	544.8	-192.2	-104.2	124.1	110.7	-10.3	343.2	67.2	-175.9	-207.7
	36	-45.1	67.9										
3	0	-682.0	-377.1	1428.7	-213.0	-215.9	277.0	750.5	-582.6	-408.6	-336.1	-93.4	266.7
	12	-5807.4	-29.5	23.9	118.9	-161.6	-627.3	-267.9	228.9	-137.3	287.6	-86.7	170.2
	24	-85.9	-417.1	61.2	-513.7	-264.5	-143.8	286.1	56.8	-398.9	162.1	545.1	-185.5
	36	-76.9	-407.4										
4	0	-350.2	2010.3	-622.2	2453.1	-258.4	-658.5	309.6	1032.6	384.6	-3416.8	-933.8	337.2
	12	872.4	1904.6	598.1	-1246.4	2184.4	-75.3	136.7	-226.8	135.6	-302.0	282.5	152.0
	24	159.7	204.1	-98.6	89.1	264.3	-81.0	-261.4	116.2	-98.8	285.7	-314.4	311.6
	36	-285.4	473.4										
5	0	429.4	-51.5	-489.2	759.8	1315.2	-139.7	686.1	1496.8	1633.6	-102.1	-11.8	4348.3
	12	-239.9	62.9	533.5	1203.6	-228.2	-497.2	-668.0	821.5	-856.8	239.7	461.5	-999.7
	24	-181.1	229.0	-540.8	-289.1	-111.3	-295.3	670.4	-438.1	-179.3	-231.4	131.6	-340.9
	36	280.2	693.4										
6	0	-271.9	522.7	646.4	-440.8	245.5	-1214.5	129.5	-6143.1	94.3	-711.9	-351.4	850.0
	12	199.1	-516.7	306.6	-258.8	-418.1	767.5	-284.0	268.1	-137.2	357.6	166.1	197.5
	24	-359.3	-290.9	63.1	-28.6	-88.7	22.3	11.9	302.5	-87.5	217.9	70.7	-354.3
	36	-447.1	29.9										
7	0	-620.5	179.6	-849.6	178.5	-24.1	544.8	-432.8	2266.4	574.8	-50.8	890.8	312.5
	12	440.8	-437.3	486.7	-389.8	-281.1	804.2	204.3	-322.7	5307.3	517.6	172.9	-145.2
	24	501.9	-116.2	-98.3	-482.4	299.8	21.8	-577.5	224.5	-547.8	278.3	237.3	-249.0
	36	-270.9	-73.1										
8	0	97.7	125.6	297.7	2016.1	487.3	-136.4	-405.6	-111.1	-843.7	-2.7	-5464.7	236.4
	12	264.9	452.9	-186.7	768.8	-32.0	-316.3	420.9	191.1	-69.3	174.3	-26.4	63.7
	24	121.8	-18.5	-420.5	368.6	52.7	167.0	-166.5	398.1	-25.9	78.9	202.8	-196.5
	36	31.7	-113.3										
9	0	-597.2	105.5	140.4	-630.1	-93.3	549.1	-165.7	40.7	-515.5	487.1	105.7	231.3
	12	-98.7	203.2	292.1	-5907.4	556.6	-332.3	3.9	-174.2	50.0	369.9	-34.6	-209.5
	24	50.4	-756.8	-69.8	-166.0	79.9	572.6	57.3	-53.9	172.1	163.7	124.4	-179.2
	36	-23.4	-108.5										
10	0	335.9	717.1	84.0	592.6	61.3	721.6	-744.7	18.9	599.8	818.3	196.8	-336.9
	12	491.0	-496.8	327.7	309.2	-9.0	124.3	302.9	-153.6	94.4	301.5	408.8	5402.7
	24	-111.6	685.4	-243.6	449.2	366.0	198.6	99.9	-1156.5	-484.5	453.6	114.0	-144.7
	36	-176.1	152.2										
11	0	265.8	-3447.7	-2144.1	528.3	-1060.5	-510.0	-898.3	-1468.3	-524.1	30.6	1342.5	1087.6
	12	2333.6	6.9	-247.9	173.5	8.8	436.0	-484.4	834.2	-339.2	553.1	593.9	-85.4
	24	-85.3	331.3	-72.6	-139.9	1055.0	390.0	-211.3	-828.8	12.1	30.9	-14.4	-225.4
	36	-301.7	514.4										
12	0	346.4	74.0	102.0	-2663.2	-2308.4	280.7	2368.9	-35.1	296.2	-4130.9	29.4	221.3
	12	-96.1	-148.5	271.9	-423.3	-302.8	198.0	-341.3	-64.2	-146.8	251.3	-483.7	293.0
	24	-227.7	26.3	-20.4	344.1	-239.3	18.2	218.7	-88.6	-36.5	82.2	93.9	-31.4
	36	49.5	-4.5										
13	0	-106.1	435.4	242.4	304.0	377.1	-1210.0	6.9	1063.0	-288.1	-118.1	135.1	107.7
	12	138.2	934.3	602.4	1347.9	-520.7	-161.9	5227.9	-312.3	-173.1	-372.6	73.0	-387.9
	24	193.3	-44.8	289.7	34.5	312.7	293.3	449.2	172.0	-11.4	-177.0	-96.5	414.9
	36	-48.9	-46.8										
14	0	175.2	-117.2	387.1	-396.3	-361.5	-792.2	648.0	19.8	-404.8	5802.6	481.5	-315.4
	12	-405.6	-314.1	48.6	-364.7	637.9	-709.7	-491.9	-227.9	720.0	75.8	-516.4	43.7
	24	276.4	244.0	-106.3	-266.1	171.2	241.3	-170.2	251.9	153.0	889.5	-256.2	-76.7
	36	-136.5	-96.8										
15	0	-618.6	-2158.5	244.7	38.0	-1891.5	-3654.3	2480.8	217.4	521.2	633.9	11.4	90.4
	12	-1001.1	-1631.2	810.8	-297.9	-332.7	920.1	403.3	-173.0	214.0	1087.3	-409.8	-285.5
	24	496.6	-171.3	-600.7	-517.9	-320.9	122.3	-147.3	-27.3	-229.0	-313.7	12.4	16.4
	36	318.5	36.6										
16	0	-92.8	457.6	570.0	-73.6	1619.6	-426.3	4474.2	-972.1	1408.7	-209.4	-87.8	-1051.2
	12	-567.7	-58.1	1329.6	811.9	81.8	768.3	-181.5	-318.9	-185.5	-1633.5	-238.9	-349.5
	24	-780.3	306.9	316.2	-778.4	707.1	-185.5	-1116.1	157.5	-79.4	227.8	-110.5	-27.0
	36	-454.0	433.2										
17	0	897.5	959.7	-631.8	227.7	8192.0	107.6	633.7	1023.5	547.1	-259.8	437.0	871.2

	12	-380.1	-371.4	-85.3	-2.7	513.9	-277.1	663.8	-363.3	465.6	-408.4	385.4	248.1
	24	311.6	-86.3	-376.6	25.2	-329.4	138.9	40.0	745.7	-368.4	-197.3	177.2	124.0
	36	-71.0	320.1										
18	0	235.0	156.3	-323.3	-84.9	-493.3	-112.0	324.5	9.6	-254.9	-109.8	121.2	-251.0
	12	123.8	374.0	-41.4	439.3	78.4	-490.8	119.0	169.4	311.9	-134.2	235.0	-62.8
	24	70.1	220.1	113.2	-179.1	-187.4	-315.0	152.3	91.0	134.2	-122.4	-6338.9	282.4
	36	160.0	-247.0										
19	0	-111.9	310.9	108.3	589.0	82.3	-11.2	257.8	261.2	-102.2	-96.0	719.0	-223.8
	12	-126.9	-449.1	367.8	184.6	101.5	251.6	533.0	-172.4	175.4	45.2	727.5	254.9
	24	-55.5	-113.2	-6142.5	-17.4	241.6	376.3	370.2	92.6	-157.8	98.8	206.9	74.2
	36	-54.5	306.0										
20	0	1138.7	3044.5	-785.3	1181.0	-209.6	16.0	157.7	934.1	-3829.4	186.1	325.9	417.9
	12	1353.8	-660.3	-1.9	487.4	-175.5	120.2	-264.4	-182.5	-39.3	69.3	487.6	-106.4
	24	43.8	-83.6	730.2	509.9	-185.0	28.7	-22.7	-125.9	-188.9	-83.3	102.5	358.4
	36	-140.6	-72.4										
21	0	-228.5	5733.3	-745.5	-491.8	322.8	513.9	473.4	3.8	515.9	-338.9	-16.9	-642.8
	12	291.7	524.1	619.4	-238.4	-13.8	493.5	136.6	-94.8	88.3	-301.6	170.4	-59.1
	24	60.6	-224.3	-11.1	-375.5	-23.7	487.5	-118.8	-120.7	50.7	22.6	-58.9	-60.2
	36	106.8	364.1										
22	0	366.7	-66.8	1507.7	3100.5	-2267.1	1138.1	-1113.2	-581.2	1310.0	1903.0	265.9	2799.3
	12	798.9	-441.7	-82.6	24.7	517.5	359.8	113.1	-24.8	-266.7	-177.0	-219.6	31.3
	24	97.9	-463.0	257.5	75.2	331.6	-128.3	-188.7	684.4	-168.3	12.7	-9.2	10.8
	36	-118.6	203.8										
23	0	465.6	-1047.1	1191.8	-1131.8	-234.5	-266.7	51.2	90.7	63.6	387.3	55.4	-305.0
	12	320.9	-296.3	337.8	472.2	-86.2	95.3	-41.3	143.9	266.5	211.4	-364.5	-313.9
	24	9.5	305.3	-334.4	-16.2	-221.5	73.0	-223.7	294.8	44.6	281.4	-21.0	5968.6
	36	-338.9	3005.8										
24	0	1783.3	121.2	130.0	756.8	770.2	-1314.6	1689.7	-108.7	-1148.6	318.0	179.2	1128.1
	12	584.5	-550.3	-1372.7	1005.2	4665.5	545.8	-57.6	66.2	-196.8	130.5	247.9	87.3
	24	-147.9	-239.0	265.8	-162.4	132.6	289.3	67.5	-34.5	151.5	186.8	-65.4	239.7
	36	115.3	-31.2										
25	0	-38.4	-1751.3	2190.8	709.7	60.6	-1317.8	-398.8	-485.5	-1332.4	-3852.3	1270.8	-1061.8
	12	-837.6	-834.5	-556.5	739.8	117.7	112.3	-632.0	562.0	-70.8	-31.5	-1107.7	67.5
	24	168.9	111.9	-388.3	12.9	1173.6	326.9	165.6	-143.6	-404.8	124.5	229.0	-202.8
	36	105.5	-0.9										
26	0	241.5	-189.2	-567.6	-5879.6	1811.0	734.4	-131.0	-102.5	295.2	60.6	30.9	758.0
	12	150.1	-328.6	187.9	352.7	-6.7	-211.1	-202.6	-301.5	-139.8	161.8	-1.4	-107.0
	24	167.4	-703.9	-103.9	144.5	367.6	21.6	56.0	79.9	-222.5	74.3	-281.6	54.2
	36	50.2	-237.1										
27	0	-46.7	-396.2	313.9	-286.0	-471.5	160.1	-78.8	67.6	-324.6	584.8	-358.6	-143.1
	12	91.8	-213.0	387.9	-104.8	-97.2	235.6	-61.7	-148.1	-291.4	-680.5	-93.4	250.7
	24	463.5	-417.5	44.7	70.8	-683.5	328.1	6060.9	140.5	171.4	-566.6	241.5	-37.9
	36	107.9	-107.7										
28	0	-193.1	-114.2	294.9	-207.8	-223.1	-83.9	-331.4	102.3	-606.2	-311.1	-233.4	426.8
	12	-252.5	381.2	-187.4	-445.6	-352.2	-167.8	47.5	22.3	72.0	333.2	-82.3	-226.8
	24	-6305.7	-10.0	313.2	366.1	-106.4	-24.3	-253.2	-332.5	-486.9	-45.1	-201.0	269.0
	36	-730.7	-494.1										
29	0	-355.4	-1902.8	462.0	381.6	-52.0	671.7	523.4	641.9	-5401.5	-474.0	-642.4	-95.5
	12	-196.6	192.2	-35.7	1171.3	523.2	-398.0	-13.6	60.2	-547.5	309.4	318.5	89.7
	24	-92.2	79.9	-353.4	-469.4	330.2	-45.0	211.7	-461.4	1.9	-96.9	62.9	-39.9
	36	22.1	319.5										
30	0	3474.8	343.8	-604.6	-639.9	-1470.9	768.1	997.9	-1202.1	-1869.2	21.5	-861.9	-2451.0
	12	-1345.6	-693.4	753.2	-483.9	89.7	237.5	345.8	897.7	-391.1	-678.4	189.1	325.7
	24	208.6	364.8	-709.1	46.9	31.2	82.3	22.7	-418.5	35.2	39.6	124.2	-364.9
	36	151.1	103.4										
31	0	239.1	454.3	-5796.7	151.5	-84.9	18.5	-611.4	272.1	566.4	84.4	-41.0	112.5
	12	-310.6	-169.1	-989.9	-126.2	-169.3	85.8	-261.9	131.1	-121.4	248.6	178.7	87.0
	24	-57.8	-280.9	374.9	-342.7	45.7	-266.6	-19.1	184.6	-4.4	127.9	-69.2	-113.9
	36	3.4	268.5										
32	0	-366.8	-728.3	327.5	-442.2	4431.8	-158.4	-1452.9	444.5	321.9	-1135.5	619.4	-368.8
	12	-153.7	450.5	-393.1	-182.0	-331.7	348.2	14.0	-675.5	-529.0	664.5	-173.0	-39.3
	24	-344.8	367.5	394.7	74.7	225.4	-12.4	35.9	309.4	-142.3	30.8	184.9	-167.6
	36	3.5	343.2										
33	0	570.4	1045.3	3575.6	275.0	-1381.8	3667.7	-511.1	376.8	297.3	307.7	140.1	-46.3
	12	-1135.0	221.0	343.4	69.3	324.4	-54.1	-623.1	-249.8	-5.7	144.4	250.8	-362.8
	24	110.9	781.3	-171.3	-70.3	168.9	-63.2	120.6	116.1	-306.3	-121.8	88.4	-49.0
	36	292.3	30.7										
34	0	-78.1	-4250.0	-1041.8	-382.4	214.4	311.5	402.8	-41.0	-81.2	-321.4	-227.2	-2398.7
	12	24.8	-144.6	454.0	-436.0	-340.9	-828.2	-423.4	-755.1	-380.1	258.5	-354.8	308.6
	24	265.0	-2819.9	434.0	-9.9	-182.6	355.0	-153.4	-76.4	-253.0	154.4	126.0	11.5
	36	17.1	35.4										
35	0	-81.5	-94.9	-391.7	-239.5	3072.3	-1784.7	35.6	-540.6	659.5	2035.0	-2810.3	-153.0
	12	-1354.5	-912.3	904.8	-968.7	272.3	1191.6	-741.4	1145.4	144.8	191.3	705.0	514.1
	24	-653.9	10.5	67.1	167.4	385.1	195.7	-104.5	-5.1	-52.7	-263.1	27.5	68.2
	36	55.4	-78.3										
36	0	358.5	453.6	-462.1	444.8	235.3	-0.5	-1103.1	392.9	2799.9	420.7	-473.5	-4837.8
	12	800.1	188.8	433.7	-35.8	199.2	419.8	228.3	309.0	1.6	-117.7	-207.8	-90.6
	24	121.0	-114.0	231.0	245.8	-157.8	-364.8	-237.7	2.8	-111.5	-125.0	-113.2	64.7
	36	-261.7	88.0										
37	0	317.5	2161.0	2933.5	-1071.3	169.4	-765.9	856.9	235.5	489.3	239.9	750.1	-302.0
	12	4019.3	-520.6	-419.5	-69.1	-1035.6	121.5	-75.2	-119.5	-167.8	227.8	-124.2	94.2
	24	349.6	-1.7	35.4	-68.4	149.6	-48.9	69.1	33.6	201.1	222.1	-447.6	-77.3
	36	127.8	308.8										
38	0	-2471.6	1059.1	-2533.7	-2120.0	-2270.5	-677.0	-59.3	-186.5	237.3	399.0	442.3	493.3
	12	813.5	25.9	969.9	796.4	1131.0	-1811.1	-9.1	-294.4	-301.8	-708.8	689.2	246.3
	24	-255.4	-248.6	-61.4	-222.5	630.1	520.7	-162.1	148.4	-147.7	459.5	340.8	239.1
	36	174.8	-257.1										
39	0	226.2	-716.5	143.4	-570.1	169.8	-344.2	-635.9	37.1	-100.1	529.0	-339.3	-292.9
	12	54.8	65.9	6711.6	389.7	-305.9	-873.9	247.4	-95.4	308.8	43.6	-188.7	-240.4
	24	-67.2	57.9	-215.7	-72.7	-702.7	-109.8	-134.0	-379.8	-205.2	-222.7	-12.8	-409.5
	36	172.5	-15.0										
40	0	246.4	502.8	-394.3	187.0	326.6	-241.5	-136.1	-715.2	-300.7	644.1	-675.8	-275.5
	12	96.2	-382.3	948.8	-917.0	176.2	897.2	156.9	-257.3	417.8	278.6	-8192.0	639.9
	24	-5.2	-832.9	174.8	191.7	-283.9	312.2	6.7	-173.2	-513.2	-86.5	-279.2	-258.5
	36	123.5	0.4										
41	0	469.2	24.7	3596.9	196.1	-712.7	-1315.3	-3717.8	-408.5	359.8	263.5	-487.8	-768.0
	12	593.7	-42.3	213.3	-183.7	1309.4	-588.0	798.5	78.2	349.5	605.0	-101.0	-127.3
	24	10.1	-156.4	572.8	-334.0	-112.9	-59.9	103.4	-251.0	-238.4	-29.6	235.8	-470.5
	36	-27.9	126.3										
42	0	-474.4	932.3	-312.3	194.5	-2653.5	-609.7	-732.6	876.1	2360.4	-803.3	-1004.6	650.4

	12	-646.6	-823.4	604.7	369.9	-140.5	466.2	-316.3	-368.8	-352.2	-205.5	69.7	58.8
	24	309.3	185.7	329.6	-581.5	180.6	-153.1	-46.8	-140.4	3440.3	178.5	524.7	-38.4
	36	56.6	-107.1										
43	0	-3849.7	417.4	-660.3	-729.1	653.9	2705.6	1116.2	-729.0	-156.6	-1952.1	-1022.2	-571.2
	12	-579.2	503.0	-1295.5	235.5	470.8	-437.9	189.0	-108.7	13.2	497.1	-558.1	-194.5
	24	-165.8	-582.8	663.6	-207.0	-484.5	43.3	-245.0	-525.1	-10.3	-174.7	5.3	-366.9
	36	163.0	-7.7										
44	0	-218.8	-401.6	-1162.5	-1230.2	-268.6	366.5	-122.5	-365.6	272.3	593.1	-2523.1	1297.1
	12	2178.9	-4048.6	-750.3	760.3	-646.9	-115.7	172.9	-376.3	248.7	-388.1	-294.7	338.1
	24	92.5	-60.5	367.3	-70.0	-133.0	-26.5	-207.9	206.0	-57.5	-53.9	-22.7	-379.6
	36	189.0	-148.8										
45	0	119.4	-1189.6	-914.0	2485.0	-414.8	1582.6	1615.0	-58.3	69.2	358.1	-327.1	-562.1
	12	929.3	3461.0	-595.5	419.4	-580.5	598.1	-429.4	-150.5	77.3	1910.1	113.3	-472.4
	24	421.7	-682.4	218.5	-297.8	484.1	25.8	143.4	-218.2	90.8	-298.9	-307.5	-483.8
	36	25.5	-8.2										
46	0	96.9	-466.1	-425.2	464.1	-270.6	-258.0	-243.6	164.5	732.3	306.5	34.9	-80.6
	12	24.6	372.4	301.6	399.4	160.2	71.8	187.3	52.0	-335.1	-776.7	-145.9	-47.4
	24	122.1	-29.2	-526.8	671.9	230.5	-6279.9	53.0	-172.1	-24.1	-22.8	-21.0	12.5
	36	47.5	-110.7										
47	0	-4456.5	-870.2	-1012.5	3282.1	-104.2	-491.6	-533.4	-15.0	-791.9	-157.8	219.1	-197.9
	12	18.5	-193.5	-29.0	-479.7	353.2	248.7	171.5	-251.2	-288.1	113.2	175.8	68.9
	24	298.4	-290.2	-81.7	-172.1	92.1	93.7	147.5	-243.5	-78.4	-1.1	-30.1	-96.6
	36	-166.8	-14.4										
48	0	2079.3	-2856.9	1445.0	-1588.5	290.7	-360.3	1848.8	1338.5	-389.0	24.2	-383.9	1219.1
	12	452.3	2545.2	-48.9	100.4	1160.6	994.2	-775.5	-454.3	-294.0	-629.5	-535.8	165.5
	24	-92.8	-473.1	-11.5	-489.6	97.1	-151.1	-240.4	-68.6	25.5	83.2	18.4	-27.6
	36	-53.2	148.4										
49	0	276.5	-525.0	-176.6	-2597.0	-164.4	250.6	-583.5	1217.0	-88.9	215.7	-552.9	-966.7
	12	-108.7	-407.1	-4961.1	210.0	158.0	-472.5	11.6	257.8	500.2	-1128.8	474.7	-310.9
	24	215.0	202.1	313.1	-42.0	-90.6	123.9	-245.0	176.7	-287.1	-46.1	-492.7	85.2
	36	209.6	166.6										
50	0	-271.0	-430.6	-1121.0	-346.6	-337.3	6256.0	45.6	-259.1	479.7	966.8	-176.3	-105.9
	12	-300.5	-118.2	157.1	90.4	-425.9	-159.6	22.7	-47.5	504.0	-472.0	276.9	-91.2
	24	192.1	-260.4	220.3	-115.6	77.6	457.7	138.3	-59.0	272.9	0.6	-68.1	178.3
	36	70.5	293.6										
51	0	-2651.8	671.5	1447.2	-226.4	-710.3	433.5	-640.0	-412.1	-74.0	331.4	-121.8	55.7
	12	-303.9	453.3	-623.9	639.5	-158.7	4651.5	-414.1	248.6	202.3	-357.6	511.7	202.6
	24	36.8	-195.4	161.8	-0.4	-11.5	-137.1	121.3	-204.9	-14.4	-204.7	-51.7	47.7
	36	104.4	-44.3										
52	0	682.2	458.7	-185.2	984.2	-530.6	-423.0	-1014.4	-2208.1	443.0	635.2	4580.9	-712.4
	12	-905.3	-92.7	-1110.5	725.7	-698.4	-120.3	31.6	-538.0	635.5	-1005.6	-250.9	-130.4
	24	302.1	-453.0	-13.3	-67.9	-123.4	82.2	-158.8	-43.6	-71.1	-0.2	484.2	-170.6
	36	67.4	-279.7										
53	0	512.3	349.0	-570.5	-146.2	534.0	-688.2	-114.7	-225.3	-707.0	774.5	-655.3	550.5
	12	-371.6	6419.7	-220.0	3.1	1.7	-364.5	-109.6	100.1	-385.4	-613.8	117.6	349.0
	24	336.2	274.4	48.5	194.1	282.9	-116.4	68.0	-420.6	-82.6	561.5	285.9	-173.2
	36	180.1	-163.2										
54	0	366.3	-1008.4	-75.8	522.9	-1093.2	281.6	811.2	363.8	6385.1	-203.8	-107.9	497.0
	12	-46.1	-88.9	-381.6	-116.3	238.6	172.3	-464.8	213.0	347.3	524.3	99.3	149.7
	24	124.0	-48.3	105.0	283.1	-150.0	-1.8	-163.1	74.4	-130.5	153.9	451.4	301.2
	36	-122.4	-28.4										
55	0	-3298.3	-1559.0	-161.5	-1727.8	291.6	-933.5	-151.1	350.1	-796.7	-221.9	685.5	-315.1
	12	350.7	-1169.4	-60.0	-7.8	-329.3	62.9	-539.8	91.9	315.3	260.2	-19.8	-344.0
	24	-582.7	151.4	182.9	3600.3	-117.6	241.3	-104.3	397.7	-458.1	50.5	-88.0	-68.8
	36	562.6	555.3										
56	0	407.0	987.8	-1054.8	-12.8	96.7	591.0	602.6	-184.4	-374.3	68.2	632.1	-383.7
	12	-654.9	-880.8	58.8	-461.7	175.1	571.3	531.0	20.0	-510.6	5958.7	756.9	41.7
	24	152.8	244.8	63.9	-123.3	-573.3	-182.4	107.1	43.5	179.6	1088.4	119.6	422.9
	36	-448.9	-82.2										
57	0	-353.7	-392.4	-521.6	1253.1	-467.7	-692.1	-544.6	787.9	-66.0	1400.6	2312.6	-176.8
	12	-420.0	-1124.6	120.4	-898.7	1139.9	720.4	-1197.2	-722.4	-904.1	-938.3	883.0	-105.5
	24	338.9	2234.6	1384.4	141.4	-964.3	-39.4	555.2	472.1	233.7	-980.5	-881.8	-889.2
	36	-506.3	586.5										
58	0	-199.0	12.6	-561.6	1003.2	159.1	315.9	925.3	-201.3	1077.8	-305.7	-613.8	-364.3
	12	-493.1	-256.5	-1096.3	-301.5	-95.0	-747.1	845.5	1078.6	259.3	406.4	-912.8	161.7
	24	-19.7	4489.5	-70.4	-603.5	1253.8	693.4	467.0	690.4	254.8	-737.3	294.4	141.5
	36	595.8	2225.3										
59	0	269.5	-456.8	-91.7	-1254.0	375.2	442.3	880.2	-1011.0	-286.6	-1270.6	-136.0	283.0
	12	1172.8	971.2	505.5	-1424.7	-14.8	-464.8	907.8	442.4	160.4	-681.8	1444.6	1636.4
	24	608.0	898.8	-39.0	-81.5	-1436.4	-421.8	-998.0	1604.7	-344.4	-1764.6	1155.3	-437.3
	36	32.0	-2192.0										
60	0	-1328.4	64.9	-136.3	-956.9	-235.5	1363.2	-279.2	-421.4	24.7	1389.6	-68.4	531.3
	12	-758.1	671.8	-698.0	-976.9	-789.2	-367.2	484.9	52.8	-2952.0	-2497.1	-873.9	-519.9
	24	303.7	377.7	-42.1	-261.8	167.7	697.4	-1119.3	282.2	70.6	667.9	-63.3	-394.8
	36	-341.1	856.6										
61	0	-668.8	770.6	162.6	298.2	-20.7	-1185.6	-456.6	-549.9	-1080.0	-1993.4	-1432.0	-20.2
	12	1056.5	-1201.9	25.0	-1290.6	-807.7	-904.4	-862.9	856.6	734.8	-123.8	-634.8	-853.0
	24	186.4	2152.9	-76.6	106.8	-886.6	20.7	-415.3	-618.5	-123.8	679.2	834.5	-963.3
	36	-156.2	157.9										
62	0	-1191.3	1276.2	-647.9	142.0	534.3	-2367.5	264.5	-179.3	-1083.7	585.1	217.6	-120.5
	12	-1120.0	1045.0	-262.2	276.0	-4160.4	-273.2	-641.6	-562.9	472.6	1066.2	685.4	-404.4
	24	95.1	-104.4	-412.5	392.8	338.8	-135.5	-47.3	-237.8	-133.5	15.2	-381.2	152.5
	36	-97.8	193.7										
63	0	1741.1	-142.1	-359.3	-88.8	-513.5	598.9	-2388.6	2905.1	55.5	-1254.4	186.8	-202.4
	12	-2957.1	57.4	531.0	-151.5	-444.5	580.7	64.7	503.7	842.4	645.8	-395.8	352.3
	24	329.6	-515.4	-104.2	-479.5	-394.0	298.8	-222.6	93.7	191.7	50.9	-209.1	-204.1
	36	-45.1	-258.6										

Table C. 23 Shape codebook 0 for MODE_VQ == SCL_1 / SCL_1_960, BLEN_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	4137.3	-1201.1	442.7	4502.6	826.3	2471.4	282.0	-379.0	-412.4	-558.7	70.6	51.3
	12	-245.3	-288.6	-177.1	254.7	4.5	403.2	164.1	-112.4	171.4	-427.4	93.4	66.3
	24	71.7	467.7	-531.2	-393.5	-372.6	-218.2	142.2	231.6	264.4	150.6	-391.0	452.5
	36	140.6	-65.8										
1	0	619.5	784.4	-740.8	-309.8	1158.7	547.3	1719.5	445.2	-1274.1	393.8	368.9	128.8

	12	198.5	-414.3	-485.7	-194.5	731.5	5367.4	-141.8	92.2	185.6	-58.6	148.1	-309.2
	24	-207.3	55.2	193.1	-118.8	-87.0	-314.9	506.8	-69.1	-306.8	100.6	-496.8	-305.8
	36	216.6	-1217.6										
2	0	1407.7	1354.2	813.0	1180.9	2898.0	-345.1	-335.7	-1539.3	141.3	-1734.2	1565.2	-137.4
	12	1050.3	-1668.1	356.9	-694.3	-467.4	60.4	-449.7	-136.0	-167.3	-350.0	1505.3	536.3
	24	-199.8	-145.3	395.7	1402.4	-680.8	505.6	182.9	820.8	-1221.9	155.8	-61.8	208.8
	36	177.1	-274.4										
3	0	-842.5	-738.1	-264.6	-3836.4	322.8	1877.1	-12.5	-4046.8	-1198.2	-187.1	696.5	139.3
	12	-90.5	697.3	449.3	-138.9	-63.4	818.0	291.6	-22.0	-248.2	205.9	-586.1	104.6
	24	18.0	61.2	-298.3	3.5	830.4	-89.4	52.4	-304.3	-271.3	421.7	178.7	228.7
	36	529.8	489.1										
4	0	-2580.3	-2874.0	-798.6	-653.7	2196.9	-901.8	1639.9	-1361.0	963.9	2241.1	-259.2	415.3
	12	-152.8	-75.8	-269.3	52.0	-215.6	-85.1	26.5	1180.7	-24.0	100.2	303.8	-198.8
	24	125.7	-227.1	116.1	566.2	-123.7	-242.9	-431.6	-72.6	248.8	-411.9	-193.8	-542.2
	36	131.8	-1414.2										
5	0	-195.8	-150.5	-998.8	-7823.1	766.7	-739.4	-5.0	-312.8	-59.8	827.5	153.3	122.8
	12	-116.5	146.9	-189.4	-185.9	391.8	35.0	-536.8	-100.0	62.3	458.7	-1.3	-235.8
	24	325.4	134.4	154.1	-33.6	12.1	359.5	-299.9	45.5	21.7	162.4	-193.8	-349.7
	36	314.5	-68.5										
6	0	-1093.7	2327.4	330.8	34.6	604.5	-746.3	1379.9	168.1	1605.6	3659.5	913.3	84.4
	12	-1233.7	919.7	112.3	343.5	420.0	300.2	1373.1	-202.2	-27.0	62.8	553.2	-136.2
	24	213.3	186.9	233.2	-243.5	-216.7	-121.5	-584.2	162.1	324.2	-138.4	56.9	-1.0
	36	525.2	-351.5										
7	0	1351.5	813.2	1155.9	-1173.5	-418.8	-473.1	-504.3	747.4	-162.0	-22.8	-490.5	1488.5
	12	849.0	-276.5	237.7	-519.5	84.5	308.1	-137.7	-9.2	-48.5	-3317.8	717.2	113.2
	24	465.3	460.8	381.9	-68.1	-123.3	255.2	-142.2	-681.6	-425.0	57.3	452.3	-307.2
	36	892.0	4.8										
8	0	-1041.8	-440.6	27.1	-508.9	935.5	-837.3	-742.2	-513.4	-14.4	2469.6	1980.4	-47.1
	12	3278.7	207.7	105.2	2162.9	-525.5	-311.0	-641.4	-477.9	-320.3	235.5	897.9	-20.1
	24	7.6	521.0	70.2	-314.3	321.9	-10.1	41.5	-9.3	-369.6	-48.2	45.0	306.3
	36	426.7	-1524.5										
9	0	1567.1	-1483.2	-1135.2	-359.5	-43.9	498.1	-501.1	-300.0	398.2	943.0	1512.7	501.0
	12	-535.3	-478.1	-554.3	884.3	-112.9	5.9	-6.7	207.6	349.4	241.3	-333.9	-554.0
	24	-395.6	635.6	-191.3	1050.4	-482.5	325.0	5310.4	925.5	-302.3	322.2	-350.4	-481.6
	36	-234.6	-261.2										
10	0	1690.9	-4790.1	-185.6	960.6	-2416.8	-259.0	630.5	-2171.4	-178.5	424.9	-324.7	-168.3
	12	307.9	-281.4	-213.8	233.0	-409.1	309.8	215.1	381.8	-327.3	-204.3	-463.4	-789.0
	24	-216.7	-218.2	-948.4	575.6	115.2	-209.8	-470.5	-77.1	-606.8	-885.3	-190.9	-153.1
	36	-543.7	442.5										
11	0	44.5	660.3	-1671.2	-70.4	-321.5	-61.3	1404.5	-983.3	84.4	1011.2	-1316.8	1017.1
	12	1453.2	774.2	927.6	-349.6	396.4	-21.1	-72.5	65.3	-183.0	-261.1	160.0	289.9
	24	285.2	74.3	-18.2	-6.6	169.9	441.0	319.4	5893.4	200.7	-325.2	290.0	-268.0
	36	-185.8	-532.3										
12	0	-372.1	-4633.5	-459.6	987.7	1622.6	952.0	-1141.5	-1029.8	79.5	704.8	265.9	-785.4
	12	-356.4	-767.8	281.2	105.0	-146.0	-979.6	-12.7	-241.4	-95.5	-222.8	481.9	-27.9
	24	494.8	411.0	-663.5	374.7	-191.6	-45.9	682.5	-102.2	-398.7	-129.8	423.9	-198.8
	36	-141.9	1216.2										
13	0	586.8	296.3	-231.8	1117.3	683.8	465.5	-704.3	842.5	-346.0	4010.9	30.0	254.8
	12	-285.8	-786.5	3036.4	200.9	-376.8	-229.3	-1466.2	-609.1	212.4	-3.6	486.8	-748.7
	24	697.4	217.1	-859.7	-25.7	-677.7	-488.9	4.4	-267.8	205.5	156.2	-57.5	530.6
	36	614.5	-311.7										
14	0	4053.2	1250.5	-3018.9	2184.2	-2140.3	-56.3	341.4	-1475.2	-155.0	96.5	-53.0	-288.9
	12	12.2	-108.5	109.8	150.7	182.1	-70.1	900.7	-38.4	21.8	286.1	54.4	352.6
	24	27.2	-241.3	47.7	136.5	-425.1	-36.5	220.7	-271.2	-15.1	-189.7	141.7	134.2
	36	82.3	470.3										
15	0	-20.0	539.8	319.3	134.2	-666.7	-349.9	412.1	107.2	-113.7	-239.9	524.2	-521.9
	12	227.7	6105.7	269.7	-18.5	-109.1	151.0	273.8	302.8	-431.3	-15.1	-315.5	6.9
	24	-282.1	-113.2	411.2	151.5	175.4	237.7	450.1	1134.0	-87.8	150.9	285.3	-20.9
	36	499.7	763.8										
16	0	-1283.8	1098.0	734.6	-1276.9	86.7	125.1	-678.9	-1.6	-1215.9	894.9	-257.7	4395.2
	12	-513.7	-291.5	-1105.4	881.1	-105.0	-184.6	-27.9	-619.3	97.3	263.5	-14.8	374.6
	24	-259.7	370.7	-54.6	42.1	555.3	-22.6	740.8	659.9	-170.3	15.6	11.4	380.8
	36	-846.6	-291.9										
17	0	159.7	-197.8	-5421.4	-2775.6	-876.1	419.0	-1021.3	-833.7	420.8	-962.7	50.2	-162.4
	12	-294.3	-53.5	607.9	779.2	-179.4	69.9	-273.5	-54.7	-203.0	-343.7	-260.4	576.4
	24	-480.6	-63.1	440.5	-691.5	706.6	-252.9	-416.2	-129.7	696.0	-483.4	-152.8	-130.9
	36	133.2	577.0										
18	0	3701.1	1301.2	796.0	685.1	2746.7	-1628.3	1301.8	-475.3	-1824.5	82.5	-14.7	-748.2
	12	-395.2	-68.7	35.5	-24.6	279.3	211.2	398.5	-41.3	-170.0	-82.3	773.5	160.2
	24	-181.2	-519.0	-303.9	-281.3	52.7	-653.2	1180.9	104.7	356.7	115.7	113.8	67.3
	36	250.4	-674.6										
19	0	1311.8	12.6	-2985.2	-189.8	-1956.9	-1040.0	1161.5	-952.8	-772.8	2744.3	-1005.4	-1107.9
	12	-799.3	-701.1	119.6	394.8	-147.8	-507.3	-240.9	1457.8	-118.2	24.6	38.7	-138.4
	24	565.6	157.1	308.0	456.7	-268.3	-273.8	-341.8	-229.1	-539.3	167.4	-74.1	-203.7
	36	-258.0	-389.8										
20	0	4369.4	685.0	323.9	325.8	-2928.8	-3717.3	1379.0	83.5	468.3	2.0	320.6	285.4
	12	-542.0	-276.9	43.3	565.7	422.6	-259.7	-29.5	-590.4	-174.8	278.2	189.4	192.2
	24	-16.8	246.3	-56.8	217.2	-169.1	110.8	-140.5	-695.3	314.2	-287.5	-3.5	-711.7
	36	758.2	-802.6										
21	0	-941.0	649.7	-7974.3	-189.6	1621.7	-264.4	141.2	-275.4	127.4	191.8	-119.5	61.8
	12	-266.2	301.2	-511.5	-467.2	-37.0	-74.1	-154.7	13.6	301.8	872.4	284.7	246.9
	24	-76.7	-120.9	-57.0	-61.6	-707.5	226.6	64.3	318.2	14.0	-36.9	262.5	-29.7
	36	8.9	558.2										
22	0	-584.9	-402.8	490.0	775.1	1384.8	1111.0	-77.0	-671.7	728.8	546.0	-280.0	-11.8
	12	-2.5	865.5	-2.8	-1843.4	-5024.8	61.8	-28.7	-71.5	77.9	318.1	193.0	444.8
	24	-144.4	-100.2	-462.0	347.9	-410.1	142.2	197.6	58.5	44.0	85.3	76.7	-252.2
	36	-155.1	-508.9										
23	0	-213.8	91.9	-119.4	173.3	-568.9	-45.1	-767.7	442.4	1945.3	-492.8	488.2	584.3
	12	555.6	-1246.6	399.6	48.1	-15.6	-196.8	16.8	348.4	-137.5	203.3	-265.0	-322.8
	24	309.3	-215.0	149.0	-307.0	307.2	-220.4	-471.7	177.1	136.1	6179.0	156.5	106.6
	36	166.5	1545.8										
24	0	-3913.3	1235.3	-474.2	2000.5	-2779.8	-823.7	972.2	1829.2	-987.5	903.4	391.8	356.6
	12	590.4	10.1	163.0	-870.2	124.6	159.0	-96.7	-474.6	-309.0	-228.0	280.1	-267.7
	24	-57.1	237.1	464.4	-25.2	-276.2	-8.0	508.6	-720.1	89.7	4.4	-289.1	-164.0
	36	-541.0	239.4										
25	0	1106.1	-1347.6	1053.5	-117.2	-130.8	-3470.6	881.2	316.8	2544.9	-2205.0	96.5	2111.5
	12	-70.8	-144.6	909.4	277.4	-494.0	662.8	456.8	39.9	-389.7	-500.0	-157.7	-859.7
	24	-58.6	512.7	-822.5	-83.3	121.3	174.5	279.0	1106.8	-269.8	305.8	20.4	335.3
	36	9.9	-508.0										
26	0	283.3	1612.2	-398.8	429.9	-377.9	1452.5	-74.6	95.0	-535.6	-587.6	155.7	139.7

	12	-357.9	-620.6	-166.1	1234.4	58.3	-409.7	51.1	-101.4	295.4	-1.5	-285.5	260.0
	24	-55.1	7303.7	1654.1	98.7	-237.8	162.6	-842.2	213.7	521.2	-133.1	-98.1	-84.6
	36	1212.0	-598.7										
27	0	1239.7	-2805.8	210.8	58.5	1641.3	-4007.8	2484.2	400.5	-966.3	806.2	732.7	-207.8
	12	-716.8	-82.4	-446.2	116.1	-45.6	-353.1	-156.8	-561.0	86.1	368.0	91.0	-8.2
	24	-268.6	-747.4	86.5	219.8	444.9	167.9	-312.7	-167.9	222.6	94.0	219.0	176.6
	36	515.7	-253.7										
28	0	-4015.5	3367.7	2401.1	1030.5	1399.5	310.1	531.3	-542.3	35.6	172.5	401.6	62.7
	12	503.7	-395.8	-50.3	-750.5	392.7	735.8	-122.7	458.6	-77.2	-179.6	318.4	-5.9
	24	-141.4	-522.7	92.2	415.5	-305.8	-237.6	156.4	-1066.5	-206.8	-588.2	-203.1	-409.7
	36	438.3	-376.6										
29	0	1406.9	1373.9	-1906.9	414.4	1275.3	257.4	-976.6	1205.0	1105.9	-1406.7	-3861.0	-69.1
	12	69.1	189.4	1001.1	742.7	414.3	-692.0	-91.2	1182.0	91.7	6.9	718.0	314.9
	24	-371.5	-262.9	-129.1	511.2	-328.3	-154.5	-337.3	196.9	92.0	122.9	339.0	-90.5
	36	-14.1	-3.5										
30	0	-360.0	-301.1	5.4	658.2	-708.0	236.0	153.8	278.7	797.3	1093.3	-600.6	1403.0
	12	-227.4	-474.7	-730.4	-70.0	-200.0	-51.9	176.3	-142.5	-73.6	30.1	626.5	752.3
	24	150.7	-184.7	138.6	-465.9	6323.6	283.5	437.9	-295.3	110.2	-239.0	878.3	-312.2
	36	147.7	-890.1										
31	0	-3275.9	423.8	4921.8	-680.1	-1414.3	-1190.8	-903.6	-336.5	-255.8	-262.3	-878.1	-239.7
	12	-204.4	-83.5	409.4	375.3	571.8	147.0	158.7	22.0	-323.1	113.0	-10.0	-281.1
	24	14.8	119.5	1.4	-386.4	-188.6	643.2	44.5	300.5	-282.2	155.7	-211.4	-193.1
	36	-89.4	754.5										
32	0	-462.8	966.4	-358.9	-545.4	-911.3	-443.0	-118.0	395.4	-191.8	278.6	338.8	-6389.5
	12	-93.3	-374.0	-934.9	23.2	-795.6	29.7	11.0	177.5	-323.3	-299.9	302.7	147.3
	24	737.5	-166.1	-583.2	237.7	366.2	-343.9	87.2	17.6	-378.7	317.2	-118.0	-71.3
	36	-299.6	171.7										
33	0	346.4	2391.9	2795.2	-1534.5	214.0	-1512.0	-285.0	-2809.6	-306.8	488.6	-358.0	-878.9
	12	-487.2	277.0	-103.3	-756.0	-426.8	107.1	-524.7	3.2	334.2	-407.4	1003.4	476.1
	24	-334.2	399.8	20.1	28.9	42.0	129.1	445.4	778.6	295.1	-63.1	242.7	236.8
	36	-391.9	53.7										
34	0	-545.1	869.8	2009.2	-646.5	-124.5	-1590.8	3088.0	4934.1	-906.7	-560.6	810.8	-300.5
	12	166.4	-618.1	25.5	826.9	32.6	-221.9	-234.7	-209.3	437.4	13.3	-243.7	616.9
	24	131.9	-53.9	162.8	156.7	-167.5	350.5	286.5	-24.4	-61.0	-224.8	287.1	-60.7
	36	15.0	-105.4										
35	0	-1384.1	-17.1	700.9	410.9	-805.6	-7709.7	214.2	388.2	-282.5	235.0	-776.3	222.0
	12	-98.9	-66.1	407.3	-1188.6	-50.9	-26.2	231.0	84.0	-208.8	147.6	238.7	126.6
	24	612.7	445.9	-156.8	13.4	-89.0	-170.4	55.4	-281.1	137.0	-530.7	113.5	-17.3
	36	529.7	138.6										
36	0	1096.8	615.0	173.9	187.0	237.4	1077.0	-485.9	770.8	138.9	367.8	6923.0	24.8
	12	236.8	-102.0	440.6	-113.7	-195.1	-99.0	598.1	263.0	202.9	-183.9	-424.6	408.2
	24	-301.6	49.1	450.8	-416.6	-261.0	197.3	637.6	-50.0	-301.0	510.8	-591.4	-261.8
	36	473.1	280.8										
37	0	-84.7	-1644.8	-192.9	1009.3	377.9	254.0	-733.4	1302.9	-971.2	369.1	148.5	179.4
	12	508.7	-346.4	-655.5	-1275.1	-239.3	-20.0	-256.1	-260.0	102.5	-213.7	261.9	-168.7
	24	641.0	141.0	-39.7	134.6	-452.4	5913.1	-798.1	-150.9	24.1	72.1	660.1	-64.3
	36	14.5	-844.9										
38	0	198.0	521.8	-1213.5	-3668.0	1138.9	705.7	733.5	1644.7	741.7	-675.7	1168.7	915.4
	12	-2346.0	-245.3	307.5	768.9	24.1	-659.2	-295.5	516.6	258.2	-262.7	299.9	178.4
	24	-218.7	371.5	241.3	247.0	-515.1	889.8	320.0	-1746.9	-228.8	-37.9	165.0	-306.3
	36	310.3	-429.3										
39	0	1164.4	1885.3	189.3	477.6	1198.7	2313.5	-209.5	261.6	4259.9	1204.4	670.2	-837.0
	12	226.3	-751.1	-947.2	-1250.1	-131.0	121.2	590.2	359.8	58.3	138.8	434.9	375.9
	24	-103.8	-182.3	-345.9	385.7	420.5	91.4	-99.2	-58.1	56.3	-161.9	-335.6	-149.1
	36	401.7	-540.7										
40	0	3059.6	-519.4	1342.5	624.2	837.3	379.9	441.0	-59.3	-253.3	902.1	-3957.8	559.3
	12	498.4	511.5	508.9	1412.4	-76.9	-245.8	-122.6	-172.5	-92.8	-266.8	-42.9	1323.7
	24	176.0	26.3	637.6	-129.7	108.2	184.9	225.2	-467.8	-182.7	6.4	-164.3	-359.3
	36	109.3	673.2										
41	0	-1355.9	288.3	66.2	-146.4	746.8	-350.2	-458.3	-160.2	7037.2	-739.3	-150.6	357.7
	12	-418.9	-600.2	-213.4	173.6	162.6	-39.8	-179.2	-265.8	48.3	137.2	-264.9	-6.1
	24	99.6	453.9	-308.7	54.9	-173.7	179.2	-32.9	414.5	280.6	207.3	57.4	596.9
	36	206.6	-851.6										
42	0	517.3	-559.3	267.6	-163.8	200.4	-256.6	205.9	214.4	215.6	8192.0	-38.4	316.0
	12	-479.9	-220.9	69.5	-881.9	-106.3	-75.9	-83.2	-777.8	-160.5	-56.5	-432.0	-307.7
	24	-313.2	-313.4	92.9	112.5	290.6	-264.3	-165.6	147.7	424.3	-298.7	-526.3	905.1
	36	36.8	766.7										
43	0	-115.5	-230.9	-232.4	-565.7	120.1	371.1	109.2	189.5	108.8	508.0	-110.6	475.3
	12	-187.3	-318.8	-7079.4	212.2	-455.9	432.4	-514.2	42.9	225.0	201.7	-573.1	-398.7
	24	-463.7	-161.6	-207.9	177.6	-202.6	-328.4	93.3	241.0	391.4	72.1	-178.2	35.0
	36	427.7	601.1										
44	0	2408.5	708.0	352.8	942.1	-1499.4	-658.0	-1015.7	-913.4	276.8	892.9	286.7	585.2
	12	322.8	82.9	-1295.4	296.3	-67.0	-290.9	18.7	-317.9	-29.8	19.8	-191.3	2299.4
	24	4483.3	708.1	129.5	-20.2	396.4	143.1	313.2	-723.9	-19.4	23.9	-60.3	40.9
	36	-1108.6	-507.9										
45	0	-309.3	1463.8	111.1	-413.7	-1371.1	-1103.1	-4154.3	572.9	-3076.4	754.1	973.0	-227.6
	12	-425.5	-978.7	240.9	-610.2	135.5	-204.0	-273.0	-98.1	-86.0	-153.8	-141.3	-1015.8
	24	-46.1	94.1	-586.1	-325.4	-381.3	525.7	-209.7	-337.3	270.0	46.4	-6.3	35.5
	36	458.2	8.5										
46	0	432.6	-1092.0	-1625.5	-2192.0	597.3	-5444.7	-1093.0	1151.0	672.0	1171.3	-758.7	194.7
	12	-25.0	-145.0	-360.6	647.8	-667.5	128.3	-500.4	159.8	95.7	-7.2	117.3	-29.1
	24	128.6	-201.3	-227.9	-89.7	-379.2	99.6	-207.8	166.9	114.0	47.7	-372.5	-58.1
	36	-347.0	-292.1										
47	0	-881.9	-5073.9	2882.9	-45.4	-804.0	132.0	1158.8	994.0	205.5	-346.4	220.9	866.0
	12	-89.7	-111.6	55.5	73.6	-237.3	-37.7	-311.6	-140.5	-54.9	-321.1	-297.5	337.7
	24	-242.2	174.3	1164.3	-805.9	-151.3	248.8	128.4	424.0	451.3	162.3	103.8	-357.5
	36	-325.9	-10.2										
48	0	-1206.5	856.0	-3878.8	1462.4	-388.2	-3068.0	-217.2	1568.0	-873.5	-767.5	240.1	290.7
	12	300.6	63.9	-179.1	495.1	-161.3	335.4	389.7	-99.9	102.6	-728.4	80.7	457.5
	24	69.0	226.3	-763.6	-148.3	-397.4	99.3	-51.0	72.6	-117.8	-291.2	-335.0	395.4
	36	-219.1	810.8										
49	0	-323.0	-8192.0	-725.4	-356.9	194.2	-376.5	234.4	141.9	-564.0	126.6	-655.2	176.9
	12	448.7	-156.0	157.8	108.6	-17.3	357.7	-363.2	-175.6	-134.3	-140.7	2402.0	-81.0
	24	-100.9	-126.6	-0.3	412.3	209.9	-107.0	33.6	811.0	535.4	101.7	359.4	-38.6
	36	-191.8	-31.6										
50	0	-454.5	-718.4	3107.7	-4161.7	-770.6	-351.0	1000.9	813.9	711.4	1550.2	-101.9	-413.4
	12	641.8	-99.5	-17.3	-458.0	199.1	-346.1	345.6	-373.8	342.4	-197.0	-240.6	13.6
	24	-262.6	112.5	-527.4	229.0	-271.4	764.3	425.7	-161.3	64.8	433.4	9.2	182.4
	36	664.8	193.5										
51	0	1005.9	-537.4	-371.6	1333.8	460.9	-763.9	2243.6	458.4	1105.6	-220.0	651.0	-358.6

	12	56.0	-1024.9	-439.6	-811.1	40.6	-119.3	-123.0	-332.7	-465.6	95.8	-120.1	717.2
	24	-1253.0	400.7	-12.2	-4557.7	-498.4	-119.6	337.9	-48.9	-547.6	520.5	-108.6	-1183.7
	36	-756.4	2213.8										
52	0	-2469.8	1033.4	-12.2	2.8	660.8	236.7	-584.8	713.8	564.1	-470.8	-585.2	-315.1
	12	280.8	-1105.9	-602.1	4101.7	439.4	244.7	109.2	265.8	205.7	-48.0	-210.9	4734.3
	24	-38.2	353.1	-17.9	207.3	142.8	87.6	168.5	369.7	203.9	118.2	228.5	162.2
	36	-417.6	785.4										
53	0	-5062.4	683.5	47.5	-241.0	27.3	1247.4	2805.8	-1616.4	-66.6	-944.7	-474.1	197.9
	12	-612.7	359.3	74.3	-502.3	-158.7	82.7	-86.4	2109.0	-1219.9	-445.6	34.1	70.2
	24	373.5	-181.3	247.7	88.3	149.3	87.6	127.1	153.1	17.5	-199.5	118.9	963.9
	36	-170.8	-197.5										
54	0	-3599.9	881.2	-1040.5	-525.0	1873.1	-1729.7	-908.3	-3745.8	-1063.3	290.8	-441.9	222.7
	12	-202.4	109.8	93.3	481.0	270.4	-127.5	-306.9	-114.1	-391.2	-382.5	-195.4	-382.6
	24	-246.4	513.2	-620.9	11.5	374.0	16.6	-433.5	-82.9	-558.3	481.9	610.7	-286.1
	36	-309.2	-139.0										
55	0	-372.0	596.9	920.2	442.0	-6547.6	588.3	849.7	-1089.5	67.7	-180.7	558.3	592.5
	12	-375.4	315.2	-247.1	-248.7	187.0	-217.2	-16.9	338.5	45.4	66.9	-24.2	-707.5
	24	289.5	277.4	68.2	-165.4	-431.6	33.0	-95.0	-64.6	167.3	-1.9	227.7	-229.0
	36	80.9	-47.9										
56	0	-125.4	973.7	579.8	50.9	12.3	-682.5	-729.7	-1382.0	-320.2	79.4	822.3	304.4
	12	-6402.1	-314.5	824.8	1304.8	-223.1	29.1	-256.3	135.0	141.0	175.4	-201.6	-6.6
	24	297.2	89.0	24.6	194.5	107.5	183.6	4.1	-227.1	-387.5	-208.9	-214.8	-331.7
	36	42.5	-1039.8										
57	0	-8192.0	819.4	17.5	144.0	1028.8	-95.1	-723.5	1077.8	-7.7	677.6	76.8	-152.0
	12	-84.5	-259.8	228.4	-1028.4	-272.9	-212.4	-32.5	-241.3	-310.4	-124.0	20.8	482.0
	24	393.3	-332.5	386.4	-374.5	104.2	-32.0	-586.7	-66.1	62.4	-148.3	36.6	-203.0
	36	197.6	-1268.5										
58	0	125.1	443.4	844.1	1218.0	1222.1	-1560.1	-1914.6	554.9	-110.3	165.2	-144.7	326.8
	12	-128.4	348.6	-694.7	709.4	-237.9	183.1	-124.0	-574.5	-3739.0	4.7	47.3	-19.0
	24	-1.6	-427.9	357.4	-195.1	-181.6	139.4	-189.0	-223.3	-142.8	46.3	71.0	27.0
	36	483.0	224.2										
59	0	-688.5	470.3	590.8	-333.6	787.2	-126.8	1683.5	555.3	922.3	650.4	-491.2	168.6
	12	-197.2	-319.2	-83.3	1105.2	-210.1	128.7	23.8	6.1	222.1	92.2	-7.1	-71.7
	24	-107.0	129.2	136.8	-467.2	-92.6	377.5	156.3	-117.8	-6142.0	121.7	117.5	97.2
	36	-1010.1	519.4										
60	0	-551.6	-433.0	453.8	3998.1	4593.4	670.1	-121.7	-816.8	79.6	528.4	554.3	13.6
	12	-506.3	-539.6	-63.0	537.4	457.3	581.1	65.6	-198.3	162.2	242.8	-1430.6	-342.0
	24	716.9	-82.2	205.3	32.9	-517.8	-605.9	-252.3	301.1	94.7	-477.0	204.2	-1095.2
	36	279.0	-394.7										
61	0	328.7	-737.1	38.6	-3046.5	446.0	1138.8	-5263.0	793.5	231.2	163.3	-700.0	513.0
	12	-143.7	150.0	-50.8	-384.0	628.1	-111.0	-173.4	-400.5	638.1	-195.8	66.7	493.5
	24	286.3	-227.2	-205.6	156.3	-162.7	-874.2	0.6	429.1	-22.9	-40.8	-339.0	-186.2
	36	210.1	1921.0										
62	0	-4066.0	-1745.3	190.9	3742.9	1387.3	-1754.2	-67.1	-21.0	-206.2	-375.1	16.6	289.9
	12	-918.0	-66.1	155.6	209.3	607.1	-682.4	-412.4	-196.9	-45.6	51.9	41.3	49.5
	24	92.6	437.7	-164.1	-83.4	110.7	236.3	83.0	130.9	147.9	284.8	-351.4	95.5
	36	-58.0	548.0										
63	0	-246.2	386.5	-365.6	-234.5	-294.5	-227.2	-767.5	8085.3	-616.7	516.1	-584.5	-18.1
	12	128.9	93.5	180.8	-586.0	-417.5	-808.5	-237.7	-158.9	360.0	-344.3	732.2	41.4
	24	357.0	20.1	166.3	-48.9	37.8	-256.6	445.2	-554.4	-27.0	-136.7	328.3	69.6
	36	481.2	-740.5										

Table C. 24 Shape codebook 0 for MODE_VQ == SCL_2 / SCL_2_960, BLEN_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	345.8	-221.6	1319.9	7462.6	1799.3	483.7	80.1	122.2	-893.8	346.7	80.4	-474.5
	12	364.9	-97.8	-161.9	-78.4	54.2	610.7	381.2	-269.1	457.7	-70.6	516.6	528.5
	24	126.6	90.2	-74.0	-65.8	-260.0	-254.2	547.9	-86.7	241.5	48.1	59.9	109.3
	36	22.4	817.6										
1	0	361.3	451.8	347.0	-459.3	372.3	-404.8	-207.5	810.9	872.7	-781.7	-405.5	-690.8
	12	78.9	-173.4	43.4	-518.7	-7.4	1602.1	-2439.7	365.4	312.5	473.6	-398.0	-579.8
	24	-435.2	719.9	-4826.3	-274.3	456.6	-579.9	151.7	509.3	162.9	-285.4	-451.1	108.0
	36	1010.2	-429.7										
2	0	-193.0	280.3	-295.6	-124.2	-26.6	-210.1	513.0	375.2	159.6	-216.7	-40.6	-235.7
	12	-237.1	-138.0	-416.5	-354.7	-190.3	618.5	348.7	-324.7	191.9	941.8	-581.5	-277.4
	24	-89.6	-26.0	1265.6	-84.2	159.6	70.9	-71.3	97.1	5760.9	-615.7	-140.6	840.8
	36	-186.2	-169.1										
3	0	-336.6	131.5	-103.1	270.2	-316.1	39.6	229.4	-509.7	481.9	362.7	-362.3	259.0
	12	-837.8	-10.2	83.5	107.9	-145.8	154.1	-717.2	-514.8	-139.6	217.1	-699.4	15.3
	24	-2905.7	4887.3	785.2	548.2	198.1	556.3	319.6	104.8	186.9	402.5	-211.4	-606.3
	36	-691.0	1044.6										
4	0	-32.4	935.6	-236.4	-392.8	212.0	407.6	-1218.1	-834.5	482.3	-38.3	-803.0	-887.3
	12	613.0	290.7	405.4	-458.8	20.1	-1240.1	-365.6	164.3	4480.4	719.2	1135.8	560.8
	24	605.0	290.1	300.7	287.7	-79.3	364.6	-642.9	1706.0	-761.5	529.0	-724.3	-204.5
	36	224.3	-765.6										
5	0	-305.1	-724.9	2894.4	3135.8	414.9	1744.2	1070.8	-167.5	-579.4	257.0	-76.2	-1112.1
	12	12.6	-474.4	10.7	-165.9	-1361.0	-323.3	695.4	287.4	-359.3	545.8	135.4	-513.3
	24	-188.6	-357.6	-544.5	1282.3	-2829.6	-211.2	-1098.5	556.2	-261.2	-638.8	-1314.4	566.7
	36	590.7	1245.5										
6	0	-1463.3	644.3	-546.8	-796.9	-845.0	-735.1	375.9	-659.3	6150.4	-1031.5	-311.7	-401.4
	12	352.7	293.7	68.0	-358.0	-578.4	862.7	-728.8	13.7	43.2	-229.9	-47.2	446.8
	24	129.1	-234.2	84.5	316.4	157.1	-672.7	315.6	195.5	198.1	-312.1	1667.2	78.1
	36	732.5	-1157.2										
7	0	-1415.0	1263.3	-446.0	-326.2	50.2	437.1	-200.4	663.1	813.5	431.3	-776.7	178.9
	12	-1056.8	603.4	-195.3	627.4	-5483.1	-1376.8	-613.5	-31.4	-229.5	-10.2	90.8	84.9
	24	-68.0	782.1	856.9	-540.5	106.7	-543.0	-533.6	16.6	304.7	-168.6	-90.8	-315.8
	36	309.5	-125.2										
8	0	367.5	264.1	-658.1	265.4	-49.8	-262.0	134.7	194.3	-521.7	351.2	12.5	274.8
	12	-78.4	-228.2	-379.4	635.9	366.9	-193.0	954.4	5916.2	-798.7	293.1	-0.9	289.2
	24	551.3	387.1	22.7	-310.4	-156.9	-473.3	-320.3	-46.1	-40.5	734.3	972.7	-359.5
	36	-1244.0	-444.7										
9	0	-1151.0	1908.1	1567.9	735.1	537.6	1457.0	-1156.7	1707.4	-1225.8	-898.6	1827.6	-1202.4
	12	-256.9	-44.7	1177.2	1449.0	774.9	2687.3	-228.0	-696.9	-259.4	-579.5	-362.2	443.0
	24	903.7	867.6	722.0	470.1	451.0	289.5	-724.3	2063.7	291.1	-286.8	-448.1	-70.3
	36	206.0	417.0										
10	0	4164.6	-1440.7	-518.8	-3348.1	514.9	-166.7	2333.8	1383.0	-461.9	-365.7	-775.3	-148.3

	12	1337.6	553.8	-1053.7	70.3	348.9	-903.7	205.9	63.7	-88.2	30.1	-105.5	-124.0
	24	38.9	205.2	-184.4	243.0	-173.0	-345.1	-126.8	224.9	19.6	379.2	-10.4	118.2
	36	181.8	-1142.9										
11	0	-622.0	-595.9	-26.1	625.6	2220.3	-1578.7	-370.4	803.7	-1933.1	2011.3	2250.5	1177.4
	12	-1055.7	2949.7	1354.1	-2315.0	-287.6	-896.5	1326.0	917.6	296.2	560.8	301.4	505.4
	24	203.7	467.0	-167.7	43.6	-174.8	-247.4	-481.6	-22.9	202.6	231.5	274.0	-613.5
	36	-4.0	-144.6										
12	0	-92.8	-2749.2	-716.0	-8.3	647.5	333.7	-2894.1	-2788.9	-1456.7	-517.8	-838.2	-2178.3
	12	-8.1	1505.1	1772.7	935.7	936.2	739.7	-529.3	-46.8	-215.3	577.3	772.7	259.4
	24	121.2	-447.7	699.4	136.3	754.4	-35.6	599.2	-86.2	-302.4	202.1	95.2	-190.9
	36	-866.0	95.9										
13	0	1213.3	-666.0	374.4	-2314.6	845.7	-3792.8	-451.7	3116.6	-1346.4	-197.0	1856.9	-75.6
	12	-1997.5	423.2	1005.3	876.4	-1430.6	383.1	-154.4	316.2	55.0	27.8	-17.0	-128.3
	24	-324.2	95.2	138.6	-160.6	306.3	107.5	-392.1	171.2	-78.9	133.1	-523.3	97.1
	36	222.7	-373.6										
14	0	6780.3	-1272.7	-453.3	258.3	922.6	-1172.7	-779.1	682.3	683.6	-161.0	69.1	326.9
	12	338.6	-472.4	694.0	507.0	-72.6	633.8	492.2	50.4	107.3	149.3	-503.0	69.2
	24	-796.5	-74.9	-162.1	-91.5	-104.7	-42.6	-596.7	338.7	115.1	-127.0	84.0	330.9
	36	-357.2	-14.4										
15	0	-44.5	538.2	-739.4	386.8	429.3	985.4	247.3	-265.5	556.7	1084.6	-34.8	341.0
	12	70.2	1891.4	1681.0	1074.2	332.2	2518.1	5158.0	35.0	-109.9	-97.9	109.2	-116.0
	24	-278.3	631.2	413.4	385.1	-210.2	220.7	-264.9	-369.9	-34.1	694.9	525.5	-657.4
	36	-220.5	-185.9										
16	0	38.0	-280.8	1083.8	1642.3	3238.2	1252.5	121.8	1885.1	4583.4	-1465.4	753.2	553.0
	12	138.1	75.8	-882.3	29.0	437.3	-104.9	-611.7	469.1	27.4	-57.7	143.0	231.6
	24	-81.4	289.8	304.7	-728.5	522.6	-195.0	201.3	-152.2	-530.2	-204.1	-328.1	254.3
	36	-275.9	-1742.9										
17	0	3667.4	1950.7	319.9	875.1	-802.3	1414.1	561.3	2248.4	621.9	3349.5	888.1	-1251.3
	12	-660.9	537.9	1407.0	453.7	-34.9	247.3	661.9	202.3	21.5	479.8	441.5	670.5
	24	16.6	-62.0	-8.8	-257.7	-783.5	-377.4	-449.0	-11.5	135.8	-75.7	-0.6	-151.7
	36	4.5	-123.2										
18	0	-574.7	-1741.4	-324.1	-3330.1	1024.4	1046.6	1807.3	-226.2	-1843.3	-72.8	-855.4	2361.4
	12	-610.4	1407.6	-2028.7	315.7	1579.9	-135.0	386.1	5.1	833.5	-458.4	-41.9	427.4
	24	-8.6	-273.7	173.0	230.8	499.7	-774.0	-277.7	172.3	330.8	-137.9	-256.5	81.8
	36	-77.6	-267.1										
19	0	1.2	584.5	-463.8	2341.2	1858.4	1325.4	-108.6	55.3	-1151.3	346.4	-401.1	179.9
	12	227.9	48.6	-1394.1	-5054.1	-357.6	-1232.7	-802.1	345.0	-74.7	385.0	-149.4	-467.9
	24	-61.0	220.5	-3.2	-65.0	-223.8	-165.3	412.0	52.9	-233.3	316.3	535.8	-388.0
	36	69.5	162.7										
20	0	1117.9	-4348.0	-4001.4	605.7	2527.6	-1173.2	21.0	-14.4	-89.6	929.1	-369.0	458.1
	12	321.8	91.8	228.5	354.1	-319.6	-97.4	-55.4	-575.7	262.3	31.5	-284.1	-135.5
	24	319.6	-208.4	-314.2	-265.2	139.3	-366.1	233.9	-351.1	21.6	388.7	-77.9	22.6
	36	-305.6	-912.5										
21	0	166.2	145.9	136.1	-1162.4	-1395.5	138.2	1827.1	-469.3	-1645.5	380.1	-635.0	-624.3
	12	-930.2	5405.9	1787.7	251.9	-328.7	-336.9	-1110.0	-290.0	-303.3	252.5	-7.4	-186.3
	24	260.2	149.2	-172.5	-109.2	-264.7	304.3	-293.6	-474.6	267.8	239.7	-41.5	-144.8
	36	28.7	-1071.9										
22	0	340.0	-280.8	-47.7	580.8	-861.7	-2165.7	263.8	33.4	-63.2	271.8	2012.0	-1610.2
	12	-298.8	806.4	-1985.3	-2338.0	-781.6	11.2	-601.4	-722.9	701.8	-2021.7	-848.3	-1911.0
	24	-376.8	386.6	1182.5	333.6	-2224.6	-3.7	-500.8	-350.6	128.6	760.2	1359.1	-330.1
	36	-303.7	1423.5										
23	0	611.7	-169.6	576.6	-1326.8	-250.0	570.2	304.5	-1231.8	-124.6	-370.6	-942.8	-282.2
	12	-365.7	328.9	167.8	16.7	-1004.3	42.8	430.7	266.5	-558.8	-353.9	153.8	-888.2
	24	-384.8	-524.5	-1595.4	161.4	-84.9	2078.4	-1424.0	858.3	520.4	-368.1	-289.1	-5062.4
	36	833.9	-2842.7										
24	0	-298.9	-105.6	-822.3	123.1	4.2	177.2	-317.7	249.7	-733.3	1353.3	-56.4	386.7
	12	-331.3	-198.6	-939.0	1652.7	969.1	-829.7	-1082.6	-2711.0	-34.6	-279.0	-2674.6	1891.9
	24	-1009.4	-577.7	-847.5	-1897.4	-1455.3	-117.4	-1248.3	821.6	-410.6	-173.7	890.6	426.6
	36	128.7	-1098.4										
25	0	765.6	-1207.8	807.8	-416.1	-591.9	225.0	377.2	263.7	-12.7	-809.1	796.4	-109.0
	12	-225.3	-271.0	383.4	-500.5	197.4	-375.2	761.9	1224.7	-87.2	163.9	-12.8	1360.1
	24	-159.3	2334.0	-354.3	-198.5	-221.5	2288.0	621.8	249.1	360.9	-1200.0	1804.7	4143.8
	36	1796.6	106.2										
26	0	33.8	281.1	-211.8	-864.0	368.8	-478.5	314.7	-968.3	984.7	520.5	-832.0	0.7
	12	70.6	73.5	217.5	104.9	214.4	272.7	698.6	355.9	-440.8	21.1	277.1	-376.5
	24	-282.0	-15.7	-396.9	1414.9	-5903.4	818.7	1782.0	-180.2	349.9	535.7	-357.5	-458.1
	36	-846.4	334.0										
27	0	-220.1	1091.0	2236.2	1148.6	-1739.2	-748.8	-1384.1	2474.2	1542.3	306.5	-676.5	3264.6
	12	1152.4	2511.4	787.5	-265.1	351.8	265.9	54.5	-170.8	-412.8	-437.8	-21.8	-640.4
	24	-111.4	-304.6	-381.6	239.1	-276.3	241.5	-9.4	-364.8	77.7	587.6	-148.1	1557.2
	36	-340.3	-93.1										
28	0	-4258.4	-222.7	1.5	479.6	440.7	-1451.3	1652.7	-464.8	2077.3	451.6	-1472.6	271.0
	12	-2246.0	-85.1	436.0	2122.1	-393.1	251.0	908.6	107.2	220.7	-36.6	-270.6	-229.9
	24	275.8	-366.7	-1720.4	548.7	321.5	-582.8	-142.1	-279.9	235.2	-3.3	76.9	-29.1
	36	378.6	517.9										
29	0	-276.1	-480.8	-50.3	1230.6	646.9	-1022.8	92.1	-32.2	496.4	1231.4	107.8	-1990.7
	12	-403.4	530.6	565.1	3501.3	-2064.2	-2136.3	-963.7	545.2	-1279.6	-995.4	954.0	-1880.7
	24	-295.8	94.1	154.4	-100.8	-249.9	356.7	-105.6	158.4	160.5	84.0	277.9	94.8
	36	253.9	1000.6										
30	0	-391.7	46.1	150.8	31.3	-797.8	-220.4	-333.0	-368.7	233.4	-668.4	505.7	1089.5
	12	-8.7	-7.8	267.5	458.5	512.3	77.1	-513.2	-504.4	242.0	1175.8	702.6	-219.4
	24	1300.4	1068.4	256.6	2357.5	-1975.1	92.4	-1859.9	-3530.9	871.3	-921.3	-889.8	-785.8
	36	2755.4	-1959.4										
31	0	-1493.0	-3249.4	-2242.6	796.8	-2638.6	1643.9	3172.9	-131.4	-468.6	-26.1	331.8	189.3
	12	406.9	-181.8	-594.9	-1918.8	-173.2	452.6	272.1	-430.3	-333.2	218.3	800.8	526.6
	24	47.5	435.6	-556.8	-168.2	289.2	-196.8	223.3	564.7	-563.9	148.2	-42.2	-16.0
	36	434.4	-84.3										
32	0	598.1	263.2	-921.1	1519.0	3803.2	835.9	2965.0	-675.2	-930.5	-1342.5	52.2	681.7
	12	2361.4	-409.3	1781.3	1341.9	530.7	137.0	-1621.9	549.1	48.3	391.1	-201.9	202.2
	24	-66.3	-263.5	418.1	461.7	-94.9	255.0	-216.7	180.3	14.1	295.1	338.7	-442.5
	36	-115.6	-16.7										
33	0	-568.0	302.2	-97.4	706.9	7.4	220.6	1047.6	831.3	-1525.9	1932.5	616.6	1929.9
	12	1729.1	-1868.3	2371.8	-472.6	1318.2	-243.7	-4.2	606.1	1239.4	-1777.8	-386.3	-1728.8
	24	-29.9	14.5	-1171.4	394.9	277.5	-505.9	860.6	-588.5	-416.6	-129.9	-179.1	447.4
	36	873.5	529.6										
34	0	1981.2	-455.5	-1302.2	1321.6	1413.6	1922.3	-1167.2	-1017.8	493.8	-1040.2	-73.1	5029.9
	12	390.3	788.5	209.6	67.9	-895.7	872.6	146.8	-237.6	-352.8	-382.3	157.1	-8.3
	24	463.1	63.1	332.6	220.4	-17.6	-359.9	146.0	-246.9	-100.3	-228.5	71.6	-277.5
	36	-263.0	287.6										
35	0	-1000.7	-6639.9	81.6	98.2	256.9	-83.7	-42.0	1223.4	1223.3	-898.9	107.3	-403.7

	12	-81.9	206.3	-688.5	145.0	453.8	-519.2	-432.0	385.1	-361.0	-353.5	394.3	129.9
	24	192.5	95.7	-37.7	775.6	133.8	51.6	-122.4	-597.0	-137.0	-102.9	-213.5	-251.8
	36	-373.2	-127.6										
36	0	134.5	1113.7	1153.9	55.1	-845.3	-989.1	4240.7	-3669.6	14.2	-2639.8	-142.4	-970.1
	12	-39.7	-134.3	810.5	71.2	994.4	-244.4	-663.7	132.6	866.3	-132.9	311.2	-120.3
	24	338.2	323.2	-323.3	-611.5	198.0	-223.5	-198.0	-314.3	-287.2	-28.7	-371.7	52.4
	36	-325.1	254.2										
37	0	1116.2	864.8	-459.2	-283.0	-951.3	-1746.9	1022.0	-1179.0	981.5	763.9	289.4	693.4
	12	428.5	237.3	465.5	-314.8	464.3	-65.0	-1128.3	483.3	373.8	-308.0	2428.7	553.6
	24	235.2	1576.5	-110.6	-3702.2	-149.0	347.0	255.6	1019.5	776.6	-394.0	-11.3	1533.0
	36	-184.2	-3.9										
38	0	355.2	-406.1	-361.4	574.5	-623.9	-21.2	-245.2	-382.6	-1381.5	-2760.1	-711.8	725.7
	12	1796.3	2336.4	-870.1	2459.3	640.1	-3977.0	195.3	551.7	59.0	577.4	-478.8	-122.9
	24	-478.9	-101.4	47.2	-478.0	-529.9	26.5	-642.7	59.4	-100.7	-62.9	201.8	36.4
	36	-439.7	584.2										
39	0	-1506.3	1900.7	-1655.4	-411.9	-638.6	3141.2	4286.8	142.3	-2246.6	243.5	1421.5	310.5
	12	651.5	795.6	309.3	-348.4	-123.3	33.5	300.5	-550.3	-13.8	909.0	236.2	56.6
	24	188.5	625.0	372.7	-277.4	237.0	79.4	166.4	-687.9	-62.8	-38.5	27.4	-58.5
	36	-430.7	507.2										
40	0	142.2	-263.0	-1120.0	1147.2	7551.1	1032.7	-504.3	337.8	-384.5	597.7	-63.9	-280.7
	12	20.8	583.1	261.1	181.3	301.5	252.0	-201.0	-600.4	285.6	441.9	-657.5	-124.0
	24	-420.8	358.0	53.0	7.1	-234.3	-292.7	676.7	-215.5	-66.2	576.4	45.4	244.2
	36	-270.9	221.6										
41	0	2180.2	-5126.0	-658.4	-2465.0	-1307.6	510.1	178.9	-734.9	464.3	1130.1	-24.7	404.3
	12	258.9	-661.1	831.0	-1090.8	1285.4	434.6	1109.7	-508.9	-52.9	269.7	-385.0	-89.6
	24	205.3	337.8	-117.4	-671.1	-76.1	50.9	160.0	79.7	562.7	-270.9	116.5	-73.0
	36	369.8	-1324.6										
42	0	-399.2	398.0	81.0	-399.1	-1510.0	-7666.9	-150.7	-596.2	366.5	113.8	105.1	76.7
	12	-431.2	142.9	-173.0	-229.7	44.7	517.6	855.4	370.1	-522.1	-98.9	410.1	94.1
	24	329.2	36.8	-308.2	-71.6	-39.3	379.0	121.4	-41.0	-330.0	-285.8	374.0	49.5
	36	208.0	-273.2										
43	0	-322.2	11.2	-436.6	-12.4	-102.6	-293.5	167.4	763.4	715.7	-3192.1	2839.4	-3922.9
	12	1137.0	337.6	1272.1	812.2	141.7	-503.6	1442.7	-509.4	-378.5	736.0	-586.1	973.3
	24	-376.1	-23.2	-219.3	-507.0	-202.4	-284.4	119.5	-503.4	104.0	-67.8	-715.8	318.4
	36	29.8	-302.4										
44	0	-486.6	543.3	434.8	690.3	69.5	-90.0	-434.1	258.0	-390.7	-308.5	-299.7	-245.6
	12	-335.4	455.0	796.1	-114.2	-181.5	-516.1	-230.1	199.7	166.7	113.5	-598.4	-262.7
	24	5888.4	1170.1	394.4	377.3	-125.8	12.5	74.8	237.1	55.4	-64.4	466.0	-1609.0
	36	-424.4	172.6										
45	0	-908.9	-265.5	-20.8	412.5	319.2	-10.0	-190.3	-370.2	96.3	-411.3	-1696.8	-1043.0
	12	720.1	638.5	884.8	-1110.2	1025.1	-380.3	7.9	970.6	-645.6	-571.2	-876.5	-157.6
	24	-250.0	-618.2	-221.2	-698.9	108.9	-512.5	-105.5	578.3	672.1	-596.6	-396.9	-249.3
	36	6161.8	1599.5										
46	0	-147.2	148.8	528.1	661.6	170.9	455.4	732.8	201.4	-36.0	-567.0	606.2	28.1
	12	1023.6	797.7	306.7	-353.1	249.8	-119.0	-225.1	-611.0	-53.8	-236.9	-425.9	92.2
	24	1052.6	184.0	-334.5	-452.7	721.9	5271.0	-766.7	-1100.0	-315.3	-326.6	-440.0	125.6
	36	-1526.6	778.8										
47	0	-382.4	366.1	436.1	-882.2	-331.3	-396.6	922.7	446.7	0.5	-235.1	208.8	504.8
	12	-202.0	-964.7	822.8	431.5	-94.3	-890.4	28.3	-1.6	452.7	248.2	-727.8	-657.8
	24	-126.5	248.7	-72.9	904.7	-671.5	1687.5	-986.6	521.7	-65.7	5318.7	-430.7	487.4
	36	-204.7	-1723.5										
48	0	3334.5	2584.4	-50.9	-1498.0	2408.5	2990.9	1324.1	-1394.4	762.0	356.8	-382.2	234.1
	12	-1364.8	-942.7	-460.8	208.1	-504.0	-38.5	-201.3	-29.4	-68.6	4.6	-23.7	-55.4
	24	36.6	392.8	-205.0	8.3	-125.8	-320.0	-360.0	-289.8	-78.1	-23.1	49.5	921.0
	36	164.3	138.8										
49	0	2502.9	-2744.8	1126.0	1171.6	227.8	580.2	-30.4	-448.6	97.7	824.3	-366.6	-103.0
	12	-1261.0	1113.2	-899.5	502.1	593.7	119.9	554.4	267.4	555.3	-982.6	-823.1	-3836.4
	24	447.5	-133.7	223.0	-262.4	43.9	654.3	417.5	613.7	216.2	171.6	-229.9	-194.3
	36	150.9	-364.1										
50	0	39.0	746.6	2646.7	-5645.3	533.7	816.7	-148.8	-496.9	-1628.6	321.6	157.3	-959.7
	12	332.9	-305.4	439.2	602.3	-452.5	251.2	-673.1	-65.9	96.5	-149.5	1029.8	580.8
	24	-19.1	18.1	-236.9	157.8	-538.4	-103.1	296.5	-47.1	-1.4	-5.0	-211.8	756.5
	36	295.8	307.5										
51	0	74.7	235.1	659.3	-75.7	316.9	256.3	-265.6	-892.9	1157.9	-198.9	648.5	711.0
	12	767.6	1022.3	451.7	-944.6	-137.1	391.2	-1418.7	893.7	-2610.9	102.1	-4072.2	-884.5
	24	1292.5	447.4	-1235.8	1064.2	-272.9	-908.6	606.4	1317.7	-542.7	1325.1	-992.2	440.2
	36	57.7	1607.3										
52	0	-702.9	1649.4	-517.8	-940.4	-115.5	-651.9	-533.0	-4638.3	388.5	1788.4	2330.6	-250.6
	12	-1502.3	-36.9	-831.5	279.1	433.1	1449.8	-520.7	615.6	-201.0	81.3	-587.7	-349.8
	24	130.0	124.0	-55.8	267.6	345.3	-427.4	488.4	-133.1	37.4	809.7	-195.8	1127.3
	36	49.5	140.3										
53	0	269.8	-356.1	1724.9	3732.6	-1134.7	-1561.8	-943.3	-2791.3	-1894.4	-2202.8	-606.2	646.2
	12	-1281.5	1014.3	-765.9	-15.3	-499.5	-313.6	630.0	-656.0	62.1	-790.9	534.6	-257.2
	24	844.1	-133.2	185.5	-210.7	963.3	-314.1	43.0	177.8	-221.5	-209.5	-174.9	342.2
	36	473.7	95.1										
54	0	-820.6	-193.6	-1801.6	449.1	-457.6	1630.9	-898.7	318.4	867.7	506.5	-877.9	-1298.5
	12	339.6	-1135.1	237.6	65.6	1080.7	-293.6	36.7	-888.6	-69.6	389.5	-360.1	-76.0
	24	1789.8	1721.8	-626.2	-956.7	212.9	-1057.5	-1243.0	-182.6	2926.9	2455.6	366.5	-261.1
	36	11.2	-18.2										
55	0	-777.1	-269.9	274.6	2314.9	4103.5	-4348.9	750.9	-276.9	-245.2	-342.1	-355.9	185.2
	12	-712.5	-223.4	-82.8	320.4	-184.5	-993.2	221.1	628.7	283.2	-487.1	389.0	347.6
	24	145.0	257.8	-530.3	-143.3	-353.8	-192.1	213.7	49.3	262.7	288.6	272.1	-212.4
	36	387.4	789.7										
56	0	2779.3	3175.0	-3142.2	535.7	66.5	-657.2	-689.3	-7.4	-520.0	-1312.1	-1426.1	-969.1
	12	1658.4	-87.3	-385.5	749.4	887.1	683.4	507.5	-60.3	-538.5	-85.3	240.1	147.5
	24	-200.0	-664.3	286.0	249.1	261.2	-1.0	22.8	71.8	116.3	252.1	231.0	-331.8
	36	264.9	-164.2										
57	0	1861.8	305.8	-495.8	1159.6	-539.0	2858.6	-5922.1	-1716.3	350.7	35.0	471.5	-427.6
	12	-902.8	-378.6	7.1	-98.7	-244.4	-13.6	-53.6	353.4	310.6	-160.6	-271.2	189.6
	24	230.0	-210.2	41.9	-17.0	-168.9	-100.1	2.4	-228.0	183.6	645.0	224.4	359.0
	36	201.0	475.2										
58	0	-492.6	-884.3	631.7	330.1	1258.0	-62.4	613.5	626.7	-375.6	-322.1	-1060.5	-309.9
	12	178.6	197.3	-394.9	-130.5	203.4	5949.8	-839.4	-246.7	-336.4	306.5	305.7	-418.6
	24	528.5	89.1	26.8	-432.7	-1460.4	-133.3	34.3	307.4	-328.1	933.7	748.4	-217.9
	36	719.0	-510.5										
59	0	876.9	251.5	385.3	1155.5	-1554.3	-2375.0	848.9	423.0	-319.9	1918.2	-4610.2	-693.7
	12	-140.1	-370.5	591.3	729.2	-516.2	-107.8	-7.5	-425.6	115.1	78.9	309.2	71.2
	24	126.9	188.7	608.3	-326.9	399.3	-266.7	134.7	302.5	32.7	-987.2	-2166.5	1035.6
	36	90.0	521.1										
60	0	-885.9	176.9	-1068.4	-664.1	-561.5	993.4	503.3	665.5	522.5	-2006.8	87.2	1054.1

	12	-5896.2	-1042.9	589.0	-68.9	49.0	-373.4	827.6	355.5	-190.1	-519.2	578.8	-141.5
	24	140.4	94.2	435.3	-173.2	-122.4	-130.9	-61.3	-13.9	117.0	-96.2	-534.8	-154.1
	36	85.5	-145.3										
61	0	-164.2	-179.0	71.3	845.2	-1151.9	-146.9	469.6	-791.7	-718.5	240.8	-144.3	400.0
	12	107.1	-343.9	-661.5	-223.1	158.6	-29.9	364.4	-156.9	443.7	5578.6	-693.9	2.8
	24	-163.8	106.4	339.7	130.0	-6.6	468.1	-204.2	293.8	134.6	71.9	309.4	79.0
	36	320.6	-999.5										
62	0	-283.6	113.2	-6939.0	-560.7	108.0	-180.6	-410.4	679.9	-225.6	-235.5	922.9	301.8
	12	-58.6	156.4	682.0	-151.0	28.6	683.4	-77.4	-186.9	27.7	-575.3	-276.9	184.8
	24	165.5	147.5	-583.8	509.4	162.3	271.2	112.7	160.6	104.3	-148.6	90.9	165.3
	36	70.7	162.2										
63	0	-54.2	442.3	103.5	-1044.1	-197.0	-1408.9	-3019.2	-1028.5	490.1	-723.0	79.4	959.4
	12	-612.5	-57.0	3317.8	-3996.1	-819.0	-507.9	48.3	125.2	-26.5	44.6	-21.3	396.4
	24	-502.4	-33.5	-65.4	347.5	-112.2	2.3	-322.9	3.3	345.5	-139.3	70.4	485.0
	36	410.8	813.0										

Table C. 25 Shape codebook 1 for MODE_VQ == SCL_1 / SCL_1_960, BLEN_TYPE == SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1038.6	-7047.6	410.4	-214.0	-3.8	816.0	-406.1	193.2	330.6	-732.2	32.9	-47.3
	12	265.8	276.9	58.8	-61.0	799.7	-484.6	-11.5	284.7	280.3	39.2	-42.5	436.7
	24	-208.2	73.8	-496.8	-325.7	20.0	373.5	-164.8	-672.3	430.0	-373.4	-182.8	282.6
	36	215.6	-650.3										
1	0	-605.8	138.1	-1772.8	-648.6	1115.9	823.8	919.8	-1648.9	346.5	165.7	-798.7	-200.5
	12	-347.1	162.3	-493.5	-337.6	213.8	12.9	-126.4	-3904.2	-99.4	148.4	10.2	157.8
	24	150.9	-25.4	392.4	90.6	-234.1	85.5	-27.7	-416.7	-348.3	26.0	68.6	-445.1
	36	181.5	190.2										
2	0	1554.1	-650.7	-557.9	-619.1	462.0	9.2	-6071.0	-997.1	401.4	-1367.3	902.1	-314.7
	12	466.8	447.8	371.6	-249.0	15.8	-48.4	417.8	-181.4	-102.9	142.9	90.1	132.6
	24	-526.5	394.1	-73.8	88.8	-188.8	174.1	-43.9	108.3	161.3	-326.5	264.0	36.6
	36	-98.1	213.7										
3	0	2818.1	-433.5	1663.2	-2289.0	-4666.9	238.9	-379.8	-96.5	-59.6	144.7	87.4	453.7
	12	138.5	-427.0	20.6	-200.6	305.4	114.3	14.1	-251.9	-253.8	-38.0	-91.2	-9.9
	24	257.7	207.3	-38.9	369.9	-228.7	-49.1	-466.2	-50.8	145.7	154.9	-463.6	-194.4
	36	246.3	49.8										
4	0	-329.8	-3014.0	467.0	-1090.7	-424.5	-1185.5	6.0	1174.1	-503.2	1368.1	-130.4	-3102.7
	12	136.3	-540.3	1363.0	-19.4	25.9	743.0	58.4	-154.7	-167.9	227.8	-1318.7	182.7
	24	-197.6	282.3	-71.2	122.7	57.4	-204.8	332.8	-364.9	-65.6	25.3	-215.9	247.9
	36	238.9	-602.9										
5	0	717.6	-1800.6	-495.6	3087.7	-4431.2	503.6	-1267.3	-343.9	377.2	268.5	-122.2	-218.2
	12	-126.7	354.9	680.1	-398.4	536.6	424.3	127.2	-35.8	-15.7	61.9	1786.4	-65.9
	24	-119.9	118.5	277.4	-242.1	-28.6	-520.7	262.0	-73.5	626.0	-223.7	-14.0	399.7
	36	-249.4	580.5										
6	0	40.1	434.6	-271.5	118.5	127.4	-237.9	-347.2	29.6	-681.8	6741.8	-566.5	-655.5
	12	792.5	339.7	-611.1	781.3	-146.5	45.3	136.4	39.0	481.3	111.7	228.0	-250.7
	24	316.9	-49.3	-378.1	451.7	165.6	91.5	-47.1	126.1	3.5	334.4	56.1	151.6
	36	-218.3	-64.3										
7	0	2171.7	678.6	5400.6	-2460.3	1073.0	-737.7	-766.0	276.4	-65.3	-701.8	-80.9	248.3
	12	-467.3	380.4	-185.4	-274.7	18.6	-55.7	-202.0	267.0	-140.7	-141.9	170.6	22.3
	24	-105.0	15.9	-133.4	-226.0	-756.5	-12.6	28.1	-109.5	228.2	129.5	-86.2	-56.1
	36	-406.2	-810.5										
8	0	-637.9	-555.9	1138.8	-2427.0	96.9	1051.5	-1037.6	1583.8	-1313.7	-2961.3	-1076.7	-425.9
	12	73.7	-790.5	-1349.3	2642.9	988.9	730.2	-463.2	-35.1	202.6	389.7	1370.8	-316.8
	24	-87.9	-138.7	225.3	-73.2	-5.4	762.1	64.8	-778.2	-496.3	-383.8	18.3	-33.8
	36	-49.0	-896.5										
9	0	-872.2	691.4	1253.4	-1402.1	591.8	875.9	-3438.5	340.4	-482.7	1049.2	1172.6	-516.0
	12	-204.2	-291.1	-163.8	-705.8	879.9	543.5	301.0	16.5	241.4	-92.5	132.8	-643.9
	24	-599.5	-125.8	-601.6	283.6	63.1	-1091.0	213.1	461.5	-2903.9	378.2	-566.2	-31.1
	36	14.5	911.2										
10	0	251.7	704.4	-1129.3	-6706.9	-235.4	1882.6	754.6	596.0	32.2	-170.8	-81.5	-473.4
	12	-272.1	-157.2	37.6	-257.4	-265.9	-291.3	129.8	-44.5	-35.1	565.0	108.8	-361.9
	24	-172.1	171.0	-193.5	523.7	175.7	-337.2	-32.3	264.3	545.9	-463.0	-643.2	8.8
	36	-135.0	-573.4										
11	0	181.5	786.1	-515.5	1429.6	-627.2	-424.9	539.0	332.3	-2657.2	228.3	-20.3	110.6
	12	-233.9	-4505.7	-12.7	-572.3	-193.9	-2.4	-299.8	-465.6	310.5	147.9	655.7	-1062.3
	24	-39.5	-18.8	71.0	363.8	791.3	478.6	218.7	198.8	392.8	-495.9	-385.7	596.5
	36	-246.2	-3159.1										
12	0	-285.1	-1859.6	-165.6	1029.0	758.9	-222.3	-45.7	-257.3	-869.5	35.6	-168.3	-1002.5
	12	-369.6	1288.3	-609.0	98.3	-504.6	128.8	52.7	-455.5	238.7	-3929.9	59.8	-386.3
	24	-76.4	-492.1	-235.7	493.7	190.1	-94.3	458.4	-18.8	277.0	-3.4	98.7	66.4
	36	679.2	-693.7										
13	0	-2495.0	-122.8	1315.1	4007.1	-1865.5	1629.7	786.4	158.8	897.0	618.5	235.4	693.0
	12	-235.6	-260.7	-402.1	-170.6	-432.6	-374.6	-335.4	11.3	-139.3	65.8	536.9	-200.8
	24	-187.7	-24.3	113.2	493.0	-51.0	-136.0	-319.3	-496.9	13.8	-55.5	50.5	-39.5
	36	-134.6	-537.1										
14	0	-505.3	2035.8	-798.0	307.0	3310.0	2820.7	3719.7	488.1	-77.5	482.0	-606.3	-142.3
	12	1.9	183.0	-545.1	-144.8	-724.5	-198.7	-449.9	453.7	-201.3	216.9	241.0	67.5
	24	-1.0	341.2	135.0	-566.7	381.7	-545.2	317.6	-150.3	359.6	-111.8	-151.5	69.1
	36	-418.5	734.6										
15	0	1385.9	195.5	-74.5	-736.0	580.6	634.6	115.2	-339.7	365.1	182.9	190.8	592.3
	12	-462.5	-761.6	5073.1	-490.0	-295.8	955.9	260.3	414.0	-404.6	41.6	-408.7	11.1
	24	-462.1	143.9	290.1	283.7	514.3	889.7	84.0	359.8	-94.2	130.8	-289.5	-707.2
	36	-103.5	-102.6										
16	0	-214.3	-130.7	-1311.1	-189.6	-3194.3	631.1	18.5	1598.7	-2414.2	-550.2	569.8	130.3
	12	-377.7	-49.6	-1258.1	-617.8	281.1	497.0	441.8	-332.8	-31.8	61.9	279.0	-482.7
	24	-327.2	-10.5	85.6	613.5	-132.7	-228.7	-36.9	3406.2	667.5	221.8	349.6	33.2
	36	393.7	1366.8										
17	0	-1397.8	-1403.5	-1041.4	336.9	-1012.0	6166.7	-342.5	104.5	-31.8	79.9	-539.2	-170.1
	12	-102.6	-82.0	78.3	-551.4	-284.4	-674.4	-118.9	199.3	163.5	239.7	-40.4	506.8
	24	834.5	-153.1	-130.3	369.7	-311.3	-126.5	207.8	105.9	-229.6	-30.8	718.8	-119.7
	36	463.2	94.6										
18	0	-3116.7	-1226.7	-1752.1	-2554.4	1791.0	-103.1	-901.3	34.9	-992.3	-1362.5	308.3	-429.2
	12	-86.5	272.5	-121.0	-907.1	-221.7	-241.3	-284.1	-233.0	346.6	377.1	648.8	-106.3
	24	-345.7	188.5	-155.9	-174.4	727.5	-669.1	259.9	-613.8	487.8	175.2	56.9	826.5
	36	-855.1	-384.7										
19	0	-2661.4	693.8	198.0	39.9	1530.6	588.9	-74.0	714.9	342.3	-291.3	-3945.4	-36.8

	12	-307.0	202.2	187.8	-249.8	-178.6	204.1	197.8	116.7	176.0	-35.5	-161.3	-286.2
	24	49.9	122.3	-261.6	-272.1	151.7	-4.1	28.0	521.5	-223.5	-172.6	-236.7	-547.7
	36	-138.2	373.8										
20	0	-4023.9	917.1	-1070.5	950.6	-1636.9	-946.8	-2488.9	-649.6	320.0	-1548.1	-436.9	-270.9
	12	-326.2	120.9	344.2	353.4	-469.7	-132.7	-162.2	507.7	98.3	172.3	-265.0	63.3
	24	-195.8	276.5	136.4	383.0	23.1	-238.3	-91.7	-63.6	49.0	65.9	8.9	68.0
	36	-801.5	-404.4										
21	0	1638.8	443.7	1527.5	-107.0	3327.9	4260.8	-1036.0	250.2	-74.7	15.5	-631.8	-268.8
	12	403.7	551.2	655.0	85.8	-507.8	145.9	-135.0	-270.0	-3.0	-162.4	576.3	-905.9
	24	33.8	337.5	-453.8	70.1	71.0	142.7	-74.9	-156.1	-183.3	-156.5	-13.5	-362.6
	36	-244.1	447.2										
22	0	549.7	-190.8	79.1	-586.2	143.8	-338.8	-67.7	355.2	922.4	149.2	-482.7	-477.0
	12	-5635.8	68.8	-471.7	-1372.3	294.2	446.2	47.3	-344.7	-1138.7	-217.5	620.9	953.8
	24	-440.5	-761.6	-357.5	310.1	-641.3	155.3	38.5	184.3	-548.9	59.5	692.6	-366.3
	36	-863.3	-1833.5										
23	0	-315.7	822.9	68.1	-1251.7	-708.3	746.2	-534.6	264.2	-586.9	768.0	-205.3	-767.4
	12	105.0	928.4	-638.3	821.5	246.8	168.6	134.0	-210.3	499.8	340.5	59.8	555.0
	24	127.8	304.6	454.9	-5723.8	235.7	187.0	445.9	349.2	149.3	102.8	97.8	-576.1
	36	304.6	255.6										
24	0	332.1	-593.3	584.1	227.3	-400.8	-1.2	92.3	737.2	6539.5	-74.3	373.9	-96.5
	12	-112.4	367.0	-288.9	304.4	-191.0	-57.7	-508.8	44.6	318.4	40.8	241.9	-159.2
	24	-264.4	-80.9	148.7	242.3	230.5	176.0	14.1	34.0	117.2	90.3	-178.7	-97.3
	36	154.4	1163.4										
25	0	-920.1	-1118.4	-141.5	-178.5	2074.5	-4748.3	255.4	-1906.2	339.1	-930.8	-217.0	-276.1
	12	-186.0	298.0	401.4	106.0	-122.4	339.1	-124.5	-132.2	268.9	86.8	60.6	33.5
	24	711.1	308.5	365.0	101.7	-181.9	-60.4	94.1	85.6	-32.4	-16.6	86.5	104.5
	36	308.7	-624.6										
26	0	268.4	-1875.2	666.8	914.7	-834.2	1562.8	-332.0	-2353.8	-18.5	-1010.8	692.5	655.9
	12	-661.5	318.3	-1210.5	994.7	-740.1	7.4	366.8	-398.1	-289.3	-167.9	257.8	528.9
	24	79.4	-2529.4	748.5	-702.3	-236.8	773.9	-396.1	683.1	1515.5	2001.5	672.0	162.4
	36	778.9	-557.5										
27	0	-1743.7	413.3	3071.3	-366.9	-1151.4	-1698.0	2900.0	-1054.7	546.0	-325.7	-112.1	-1344.8
	12	88.3	745.0	-507.9	583.0	-500.7	-9.8	379.7	-15.6	-330.8	-353.0	-545.4	124.3
	24	753.4	-247.4	45.6	141.6	94.4	234.0	371.5	973.2	-62.6	262.6	19.7	34.7
	36	-969.7	957.1										
28	0	-370.3	1307.4	-1032.9	6399.7	-536.1	156.4	196.4	140.7	157.6	40.1	463.6	222.8
	12	-134.6	297.1	-197.7	444.0	734.6	-555.0	160.1	-56.1	6.0	222.5	-436.6	283.5
	24	-213.9	254.9	40.8	192.1	-145.1	-6.2	-28.6	66.4	-148.8	3.2	-81.0	-149.2
	36	-145.4	126.0										
29	0	-290.3	63.9	-1134.7	-350.2	-491.1	-308.1	604.8	476.0	423.0	85.0	637.9	5088.9
	12	554.9	-214.2	707.3	-1021.6	-1105.6	-357.4	1552.0	144.9	-169.3	106.3	220.4	-660.2
	24	924.3	166.9	15.7	-131.9	288.6	145.9	-134.4	-533.1	-220.5	-306.9	-340.2	803.7
	36	215.6	-129.1										
30	0	-1762.3	-886.5	-388.1	156.9	-215.0	-1943.2	-3093.8	21.7	802.0	1663.4	-146.1	458.9
	12	-139.6	188.3	-172.9	-308.6	191.5	-174.9	-57.5	205.9	124.0	302.3	656.3	-522.4
	24	264.7	274.0	4162.2	372.1	-380.8	553.6	-161.5	1558.4	553.3	164.2	-195.5	-39.9
	36	389.5	-45.7										
31	0	-2077.6	3303.7	-2120.1	-603.4	-2085.8	667.9	552.0	-48.1	718.8	723.0	115.4	-214.4
	12	109.1	408.6	289.5	-287.1	-114.1	-341.0	-703.6	-107.2	-337.4	-79.0	-624.7	-1188.4
	24	-64.0	-259.1	296.3	636.5	788.0	565.5	258.7	-169.4	294.6	157.9	167.0	560.4
	36	-45.6	642.7										
32	0	-330.0	-964.8	1069.4	375.2	1288.4	2114.7	830.4	-703.7	-877.0	2442.6	554.7	1603.6
	12	971.5	501.9	-787.2	-890.7	2163.7	-196.8	-201.4	488.2	147.4	49.5	271.9	143.3
	24	-847.6	586.9	-536.4	305.3	-84.6	549.3	288.0	-285.5	398.3	-259.0	-121.4	-801.2
	36	263.2	98.8										
33	0	1119.0	626.4	2739.3	2744.6	-10.2	-1211.8	-4525.3	11.5	522.6	851.3	-560.1	-388.1
	12	-248.4	5.9	-274.4	-371.4	494.1	176.8	227.0	1014.0	147.8	-347.6	528.5	137.7
	24	278.5	-283.4	-133.8	-323.3	276.1	-477.6	-398.3	124.4	105.5	-42.9	146.3	100.0
	36	-1185.3	-4.3										
34	0	-511.4	97.7	75.0	-311.6	670.4	-140.0	215.6	19.0	609.7	-149.6	5662.8	68.1
	12	-203.8	197.2	162.4	722.0	417.2	333.4	-244.3	81.4	-299.2	72.2	-114.1	214.9
	24	252.2	-148.5	-77.0	223.2	-56.7	14.9	19.7	172.2	391.2	-1012.9	-84.6	-79.5
	36	-960.7	1098.8										
35	0	-4359.0	-141.6	-886.5	-1288.0	-760.7	942.3	532.3	-491.7	847.4	1699.3	839.8	359.9
	12	-313.5	203.2	1498.0	1647.5	124.6	280.9	233.6	-367.0	-547.7	-382.4	161.5	-391.3
	24	-507.5	184.6	-240.2	-431.6	-341.4	1443.5	286.1	-36.3	22.7	-276.1	-50.4	-85.2
	36	59.6	27.1										
36	0	813.2	804.1	-161.8	-1239.4	283.1	-594.8	-917.7	-1887.4	695.1	-841.1	-455.9	428.4
	12	4724.5	-499.8	-30.9	-572.0	-227.5	-403.3	204.5	-216.7	-433.4	202.0	-134.1	-19.3
	24	-274.1	-518.1	343.0	187.0	45.7	-401.3	32.2	104.2	90.9	-170.6	-82.3	-758.3
	36	298.8	-396.5										
37	0	-554.8	1429.0	1378.6	-977.7	37.1	232.3	1807.8	-810.3	507.7	102.5	-621.3	854.4
	12	-1135.5	-1707.8	866.4	-24.0	345.1	-404.2	-234.6	-224.6	-124.0	225.6	-1132.6	-1065.5
	24	-328.6	-2415.7	140.0	-695.2	857.4	-1125.3	-1329.1	507.0	-198.3	-2885.9	437.5	-257.1
	36	-249.9	2485.8										
38	0	338.2	379.8	-1126.2	-1545.3	665.1	-7.6	395.0	-47.9	-1035.5	-399.3	220.4	-154.5
	12	194.4	-1004.1	-205.4	-6359.9	1173.3	-17.5	7.4	-481.4	-208.2	-359.9	-314.4	546.6
	24	-206.1	147.2	96.1	-60.6	328.4	62.4	660.6	-62.8	332.8	125.8	-571.2	37.0
	36	-1222.2	450.4										
39	0	-4387.7	-18.3	821.9	148.2	250.1	-1218.6	1226.8	2034.4	-70.7	-1995.9	29.4	-84.8
	12	1631.0	-243.4	538.4	120.8	83.8	212.3	184.1	627.5	343.5	-210.4	300.1	220.8
	24	366.7	-239.2	605.1	-107.8	-444.0	-788.5	127.9	27.3	334.5	-59.7	-287.2	-319.9
	36	-583.3	1725.2										
40	0	238.2	536.6	-812.7	923.0	250.3	255.2	-218.7	299.7	287.1	-515.2	409.6	-80.7
	12	696.6	-1187.5	32.4	477.4	38.1	-33.4	120.3	-11.3	-129.7	-122.6	378.0	418.8
	24	233.4	-1086.2	-6654.1	6.6	-201.2	-325.8	-234.4	931.1	489.5	128.2	-382.0	795.4
	36	-9.9	-1104.9										
41	0	-8192.0	-395.5	155.3	175.1	-653.8	340.5	406.8	-243.0	-247.7	207.4	20.2	251.7
	12	385.1	488.6	-289.5	58.6	95.4	-86.4	-99.8	82.5	341.8	323.2	212.9	69.5
	24	-322.3	-32.2	-127.9	147.0	229.1	-313.8	20.3	-44.3	77.8	63.6	-271.6	289.0
	36	83.5	-291.8										
42	0	-1190.7	-305.9	-933.1	-970.9	-7672.4	-325.6	148.9	-1170.9	240.9	250.8	407.4	-148.6
	12	69.2	596.3	637.3	-859.8	152.4	294.9	-380.3	-407.6	239.3	-200.6	-1001.6	-111.7
	24	-256.7	-134.6	-273.9	-263.6	213.4	-55.3	88.3	-17.7	56.8	172.7	-188.3	-379.8
	36	-535.6	-779.0										
43	0	896.5	-161.7	-573.3	638.6	-174.3	-111.1	-328.7	686.2	-481.6	-458.9	-166.1	770.5
	12	-852.9	5259.8	293.2	-482.6	-349.2	-41.4	-580.9	4.8	414.8	208.5	128.4	-735.6
	24	-124.0	-45.1	255.2	205.6	-370.3	173.0	-143.8	-305.5	-699.9	-530.8	-103.8	-168.8
	36	-547.1	-414.9										
44	0	-389.9	62.4	803.9	-295.7	1263.2	-130.7	478.7	2337.4	-237.8	-255.7	-278.1	729.6

	12	537.8	80.6	294.5	325.2	-116.8	-417.6	143.3	154.3	1659.3	45.3	-134.0	-365.9
	24	-219.7	-861.8	614.6	273.4	39.8	-869.2	-4416.4	66.5	-93.5	-91.2	-597.3	-392.8
	36	-705.6	-945.5										
45	0	-166.6	1098.0	-2296.4	-1325.4	2045.9	-1006.9	-771.9	1117.4	1704.4	697.9	476.2	-400.4
	12	286.1	-1137.0	-2476.9	-207.2	287.9	420.2	-661.8	-144.2	-287.2	-407.5	-291.4	-438.1
	24	-350.5	96.5	227.0	-730.8	-436.8	73.7	-586.9	-390.6	-300.5	-798.6	332.6	-243.9
	36	-239.7	-416.1										
46	0	398.3	1104.0	-1.0	416.1	146.7	-398.1	-104.6	-6088.0	-701.6	423.9	233.5	340.2
	12	-176.1	299.1	-98.2	40.9	-495.0	169.7	101.9	239.5	-262.2	-40.9	3.6	360.0
	24	196.3	535.1	104.5	-245.5	-374.1	68.8	116.9	-690.0	-629.4	25.4	268.5	-72.4
	36	5.4	766.1										
47	0	521.9	1779.5	917.1	1513.2	-1269.3	181.4	-506.9	-376.7	-415.6	475.2	-805.3	108.9
	12	494.0	-471.3	-303.2	-117.1	-684.7	104.4	300.5	-211.5	-101.7	-20.3	-145.4	221.7
	24	-5249.8	361.8	518.8	480.6	60.9	316.3	416.0	-502.9	236.7	-695.0	104.8	-154.1
	36	15.4	82.5										
48	0	398.8	2856.0	-1330.6	240.8	-1307.1	-4496.8	-334.4	-1326.7	495.7	-631.4	-158.0	-270.6
	12	-410.3	-717.8	-11.2	-75.5	13.5	-313.4	237.5	901.6	-40.1	221.0	-107.8	-468.9
	24	65.3	-9.4	223.8	109.1	240.5	59.7	-9.3	-66.8	549.4	223.8	167.2	542.2
	36	82.5	-1343.8										
49	0	2919.4	519.6	-515.0	-453.6	722.0	-352.7	-549.0	4838.0	323.6	289.7	235.5	-266.2
	12	-352.4	55.9	160.5	-85.3	-18.2	-284.6	73.1	-2.3	-590.6	302.0	-797.8	216.1
	24	207.9	-590.4	-328.3	140.9	46.8	133.9	-41.5	-367.3	-101.9	639.7	-189.0	395.0
	36	-1069.5	794.5										
50	0	690.0	828.5	109.9	-437.7	881.4	937.7	-2.9	-1642.4	-691.7	20.2	-575.3	-973.4
	12	1189.3	-532.3	1237.8	6749.7	-169.9	-641.1	278.6	-238.6	-333.0	-170.1	139.1	84.8
	24	-180.3	822.8	-447.6	-312.7	15.3	-252.1	740.1	101.6	370.2	26.3	139.8	634.3
	36	-4.8	187.4										
51	0	1997.5	-641.5	171.7	-1841.6	-42.2	-1002.0	1976.8	-862.7	547.8	-174.5	-723.0	-477.7
	12	798.8	56.1	-1272.8	-194.0	394.1	-57.2	-104.3	-257.6	-521.5	-137.0	-105.0	-3787.6
	24	-286.5	1500.1	146.6	166.6	152.6	154.7	657.8	-81.3	495.4	1270.2	-133.9	407.7
	36	317.0	-1255.5										
52	0	1417.2	1372.2	1320.7	-160.5	295.3	-1015.6	1223.0	-67.6	-4185.9	-102.8	66.8	622.8
	12	-1211.0	463.4	-1258.3	1478.4	-413.6	-32.6	-439.3	249.7	-105.8	238.5	-93.3	-907.2
	24	116.0	126.0	-557.9	55.3	598.9	109.9	53.8	-472.6	79.7	233.5	-44.5	-631.6
	36	83.7	-228.7										
53	0	1748.5	2560.1	-539.3	2332.0	3282.5	-329.1	-104.1	171.1	48.9	-20.1	-97.9	-220.9
	12	272.6	640.3	1.8	-279.1	103.3	442.9	22.8	357.0	1082.4	-26.5	197.5	426.9
	24	237.9	472.2	-136.5	-257.9	1138.4	-430.1	1.9	30.9	-117.9	-134.1	-467.2	235.0
	36	442.7	-640.4										
54	0	2998.3	-1592.4	-1775.5	1674.1	1140.8	-3093.5	57.7	358.1	259.8	-69.6	-506.4	-149.9
	12	262.0	-129.0	-33.1	-185.1	245.8	-512.6	274.5	124.0	404.4	-109.1	-516.9	471.7
	24	215.2	421.6	309.6	231.4	-20.1	437.9	-336.9	335.9	-492.3	-482.4	325.9	469.7
	36	-326.4	1327.0										
55	0	-129.9	-1257.4	276.8	-1647.2	6773.8	-213.5	255.8	-1002.9	-507.2	-579.2	218.4	311.8
	12	-156.2	661.5	-438.4	231.7	-105.0	427.6	180.7	823.1	168.6	-6.8	-328.3	-558.0
	24	-250.4	207.2	-143.4	-290.0	-117.4	508.5	453.8	-108.3	128.3	63.8	451.1	-672.5
	36	-10.2	-784.3										
56	0	38.5	-3156.7	4181.4	907.8	-714.0	-286.3	-597.9	-829.1	803.7	573.5	-130.6	-548.7
	12	-370.2	-69.2	198.8	-46.7	231.2	623.1	575.2	180.3	317.7	467.1	-242.4	-6.6
	24	-85.8	219.6	-36.8	421.6	85.6	-309.7	-280.5	-1088.6	-241.4	-462.6	462.2	-228.0
	36	294.4	-80.2										
57	0	4612.4	4094.9	756.0	-1000.6	270.5	363.5	644.4	46.3	679.6	-68.6	-6.3	486.0
	12	159.9	-104.7	480.8	195.5	-327.2	-826.7	-654.8	-447.3	-249.5	608.4	-2.7	97.5
	24	55.4	80.4	510.8	-256.8	-101.2	7.9	-376.4	149.5	264.8	-26.7	-148.9	-746.9
	36	-372.5	1019.0										
58	0	-540.3	202.4	632.6	119.6	-66.1	-1798.8	458.5	-629.0	-1103.5	-152.2	1264.2	711.9
	12	285.1	-5.9	39.6	204.3	-15.4	-235.0	-119.3	134.8	-59.7	206.2	452.5	6899.6
	24	-316.1	455.2	300.3	-480.4	-297.9	34.0	-14.7	-225.8	4.8	120.2	-179.9	452.8
	36	283.9	657.0										
59	0	-560.2	-1240.1	100.9	2684.3	1887.5	597.7	822.8	2767.2	-3178.3	-197.9	-674.3	86.6
	12	-125.9	118.4	483.2	32.0	89.8	488.4	-2.8	-370.9	-136.9	-40.8	-1806.5	-518.2
	24	820.4	9.8	24.2	695.7	318.5	190.1	597.5	161.7	-324.8	-422.9	163.5	-29.3
	36	-18.8	-1366.8										
60	0	-91.6	-3824.9	-4210.8	-333.5	23.8	13.0	438.2	386.9	-321.2	197.2	-245.7	273.0
	12	148.8	-16.2	-202.0	-192.8	-377.8	-331.3	159.9	-0.2	-1120.8	-496.6	-58.7	519.9
	24	-127.7	-101.6	114.8	-255.5	-646.4	847.4	206.1	-141.4	-989.7	-452.7	135.4	167.9
	36	17.2	-1495.7										
61	0	1217.8	-885.0	700.7	138.1	-3129.1	1097.2	5144.8	-566.7	213.6	704.0	-11.5	-674.8
	12	163.3	-469.4	226.3	225.1	-164.7	-914.8	-206.6	-469.4	-324.1	-135.1	-161.9	46.0
	24	-83.9	252.3	-34.4	-310.8	5.8	-108.0	-51.4	-292.3	-504.4	45.4	42.2	442.4
	36	145.4	-444.8										
62	0	825.1	105.7	-7459.9	209.2	-1041.3	-685.2	387.9	277.2	-251.7	-323.5	-97.7	208.7
	12	210.1	264.1	246.4	-12.3	-373.6	-356.7	-393.0	22.7	-192.0	367.4	-3.0	-69.4
	24	150.0	-361.0	97.3	927.2	202.0	188.4	151.7	-425.7	312.3	365.5	387.3	-142.7
	36	-223.8	-382.6										
63	0	594.7	443.9	-595.7	-1152.5	1570.3	-669.1	7645.9	-24.3	-287.7	216.2	364.3	-263.3
	12	-551.9	403.6	-199.0	92.3	-653.7	-376.4	-441.4	224.7	119.7	-119.8	604.1	-102.1
	24	131.2	329.9	51.9	-330.8	-36.9	-138.7	-353.2	-76.9	-153.9	309.7	-125.2	328.8
	36	-261.9	132.2										

Table C. 26 Shape codebook 1 for MODE_VQ == SCL_2 / SCL_2_960, BLEN_TYPE ==SHORT

VN	EN	0	1	2	3	4	5	6	7	8	9	10	11
0	0	-1149.3	-3702.9	2397.8	1459.3	-1730.7	3515.1	-160.0	-541.7	-1002.2	-1114.9	-704.4	-107.2
	12	511.2	358.0	908.6	-674.2	366.0	-57.2	-478.3	-295.1	-223.6	-401.2	-246.2	-252.4
	24	-169.3	981.3	132.8	-418.0	289.0	-388.7	-30.8	-294.7	261.0	-78.0	-300.2	52.3
	36	384.1	-222.5										
1	0	-206.6	-261.3	135.1	65.1	-74.6	-138.6	24.3	1040.2	159.1	-147.4	374.5	116.4
	12	-118.6	-476.1	-105.5	-269.4	-72.0	-1379.5	141.5	749.3	-403.3	205.8	5665.5	-665.9
	24	-412.4	-815.1	-452.3	407.3	-81.0	150.2	458.6	-295.9	290.2	80.7	594.7	381.8
	36	-60.6	1674.4										
2	0	-650.7	1329.5	1500.7	-933.8	-186.0	-614.4	-78.3	770.3	186.1	75.2	-400.3	-2505.1
	12	2071.6	-463.0	-131.5	-312.6	-452.0	75.7	40.7	-645.4	543.6	-340.9	686.3	-293.6

	24	-204.3	501.1	-100.0	159.5	121.5	-22.3	246.9	-193.8	-81.1	928.0	4726.0	-909.2
	36	227.3	-1374.6										
3	0	714.3	-724.3	-567.1	-47.7	-187.9	-387.6	355.0	911.0	337.0	92.3	552.3	453.9
	12	981.7	335.7	-273.3	95.3	-5431.7	2725.5	824.2	-247.7	-362.4	-129.8	-985.7	-0.3
	24	-481.5	-261.9	-493.9	315.4	-15.8	-195.1	-109.7	-88.6	-9.3	8.7	287.9	285.0
	36	-73.7	1016.7										
4	0	-404.0	-616.3	188.1	1776.8	-407.3	594.9	-101.0	14.0	-1245.3	-238.0	169.2	-76.5
	12	749.3	4.2	561.9	439.9	-481.8	-921.0	141.7	606.1	-830.6	-352.6	-198.7	1868.8
	24	-4345.9	828.1	-679.3	-29.5	394.2	123.4	228.3	-1005.3	260.6	5.2	160.5	-940.1
	36	-23.4	-499.9										
5	0	2090.1	-2770.1	-12.4	-2578.0	478.4	-804.1	-1025.0	118.6	-982.0	-25.5	456.1	807.8
	12	-160.8	-953.6	-304.4	423.0	400.7	673.1	-946.3	167.8	-221.8	-951.2	534.9	697.7
	24	493.3	1465.0	1594.1	1610.1	712.1	-8.2	1472.6	1133.7	-1244.6	747.9	444.0	-452.2
	36	404.4	1443.8										
6	0	865.8	3.1	-390.2	-473.4	-763.4	-1525.7	160.9	458.4	-24.8	-5595.2	1090.2	2930.2
	12	140.5	-3.8	124.6	-797.3	-805.9	-457.2	438.3	-83.2	84.1	-31.5	-433.7	65.2
	24	103.1	116.4	176.9	-475.9	-362.1	21.0	-64.1	-210.2	-251.2	-107.6	-163.3	-128.1
	36	171.8	-130.0										
7	0	1204.9	981.2	2863.6	4.9	-1350.2	-368.9	1367.7	-622.4	227.7	-450.5	1286.2	-234.5
	12	-1131.0	1777.5	1227.3	-239.8	372.2	230.6	-307.6	458.8	-301.5	649.8	-711.0	-437.6
	24	-539.5	-414.4	-172.4	-636.2	2803.6	-227.5	-102.4	-1122.4	75.7	-93.5	144.4	-1087.4
	36	-2097.1	3079.8										
8	0	329.4	-334.0	500.8	118.1	-507.3	-658.4	1386.7	-322.9	225.8	-780.6	-129.9	1041.4
	12	-87.6	-482.8	5091.1	-376.2	-394.4	-343.3	-149.8	-993.2	-862.6	-998.4	8.0	631.3
	24	-166.5	978.4	867.1	-337.2	-856.7	-491.4	832.5	-148.9	529.4	11.4	365.4	-169.7
	36	-389.9	350.8										
9	0	-7.4	274.9	-82.3	-92.3	-483.7	-368.3	-133.1	110.2	-331.3	-271.4	178.5	-219.2
	12	-120.4	-119.1	704.2	219.6	256.2	-104.7	-148.6	-146.6	231.1	-135.5	341.9	-154.8
	24	-55.8	-298.2	-598.0	807.9	-108.4	229.7	93.9	5787.7	399.5	-130.2	784.6	-565.6
	36	253.1	-124.2										
10	0	2113.6	470.9	-2857.6	590.5	-135.6	-2148.7	3376.7	632.3	474.1	174.5	-438.0	-1290.4
	12	-2387.3	-290.1	767.9	319.8	277.3	-939.2	7.1	-295.2	-698.7	-405.3	272.2	-114.3
	24	561.6	33.4	-145.4	791.6	346.4	131.9	-259.9	157.2	-241.3	25.1	-55.3	-111.9
	36	-459.3	375.3										
11	0	-1699.2	148.1	340.2	1198.0	380.0	-125.4	922.3	-49.0	-879.1	305.3	3976.9	-1132.9
	12	-1487.6	1404.8	-1330.5	-390.4	27.2	-1547.9	-452.9	-667.9	-860.2	-847.5	-623.6	515.2
	24	-161.2	-1.2	-293.4	-179.4	119.6	-223.6	679.5	196.3	-11.3	897.1	-733.7	648.5
	36	705.3	-1388.0										
12	0	485.1	5630.3	108.5	130.7	-200.4	1221.8	896.5	856.0	955.2	640.9	228.4	-1180.0
	12	-1080.8	549.0	-129.3	-899.6	394.5	152.9	-128.8	-140.0	287.2	32.0	-109.0	424.1
	24	-147.5	288.5	124.8	612.4	-78.5	49.0	85.4	-55.1	-492.8	93.0	202.6	382.2
	36	-133.6	-73.5										
13	0	492.2	95.3	621.3	-1959.6	-1021.5	-537.5	-1271.9	1230.6	-309.9	-473.3	4016.3	193.0
	12	1073.9	-778.1	-2079.5	1086.7	2614.7	-449.9	632.1	306.1	-72.7	96.8	125.6	861.9
	24	414.6	-173.9	-104.2	-56.3	70.8	3.2	-59.5	4.7	172.7	-98.3	43.8	-242.7
	36	-231.4	-747.0										
14	0	3265.0	-2630.3	-2130.6	323.9	903.5	2997.5	166.4	776.0	705.5	-641.0	-243.2	-474.8
	12	-788.7	30.0	566.2	59.7	-215.1	218.9	-409.8	1051.1	143.4	-294.9	134.7	27.9
	24	822.0	133.1	-457.6	295.7	38.5	17.0	-95.0	-51.0	-27.2	589.8	-439.3	213.1
	36	176.3	103.3										
15	0	166.6	-1060.1	512.9	-1007.2	1157.6	-12.9	-286.9	615.8	440.9	-156.8	713.8	337.6
	12	-731.4	488.2	-121.6	-2500.1	131.8	2993.4	-2731.9	851.6	-147.3	-325.9	657.3	504.7
	24	-745.6	-669.5	1322.9	-435.8	167.5	1680.3	-2442.7	117.1	-141.3	250.3	671.5	328.7
	36	-0.9	1087.0										
16	0	-765.0	1111.4	-4015.7	-2725.6	200.3	900.8	183.1	-24.3	919.6	-891.2	-409.9	570.4
	12	-253.3	278.4	-584.7	226.3	-384.6	491.7	383.9	-61.2	81.7	-369.8	-211.9	-693.5
	24	-260.0	362.8	28.8	-73.4	-538.8	826.4	283.9	389.5	413.1	-8.3	-65.1	495.1
	36	122.1	5110.7										
17	0	2147.4	2409.0	960.3	1548.6	-1486.9	-519.1	-2200.8	-1156.9	1510.5	-1530.2	-1827.1	1065.5
	12	-1221.6	-740.4	-1079.9	-764.6	583.7	-138.7	441.8	367.5	-6.4	-751.1	-94.0	-282.7
	24	-58.5	-392.6	-226.3	-122.7	-37.7	-68.2	-294.7	255.6	0.0	648.5	98.8	151.2
	36	421.0	175.5										
18	0	-717.7	-1922.2	-839.8	-361.7	188.0	-4998.4	1140.3	811.0	657.6	61.3	-932.5	283.7
	12	1730.4	-797.8	305.9	-1613.9	434.2	94.0	-1136.5	-731.5	701.8	486.2	-23.0	232.5
	24	-58.6	585.3	388.6	64.0	23.3	-234.1	-326.3	-433.0	-499.4	341.0	-151.1	-29.2
	36	-30.4	630.2										
19	0	-1454.9	108.3	-3466.3	3869.9	1527.6	-1053.7	690.7	-1982.6	-154.7	598.8	-291.8	164.4
	12	-400.2	1544.5	-747.3	797.7	579.9	-607.5	24.2	579.6	102.6	342.0	396.1	283.4
	24	314.2	126.4	-400.7	88.5	294.9	306.7	-186.4	127.8	-7.5	12.0	224.6	-32.9
	36	-204.8	142.6										
20	0	-5238.4	-3083.9	81.7	822.4	1228.2	353.1	-599.0	610.0	-698.1	760.8	-596.5	-1228.0
	12	1125.8	-217.3	-687.3	347.0	-213.1	219.3	214.1	104.7	19.2	59.1	136.6	800.9
	24	-146.6	-147.5	268.6	-370.0	14.1	614.0	-210.4	195.9	-7.8	210.4	32.5	-132.4
	36	-474.0	671.4										
21	0	-52.3	-278.3	-477.0	492.6	400.8	-906.6	-6757.7	197.3	-304.8	-432.4	351.9	-843.8
	12	974.0	527.0	-240.0	336.8	62.4	513.7	223.3	-144.9	-23.1	50.0	172.0	-391.4
	24	-349.7	-141.6	239.6	271.0	26.3	-444.8	-70.8	-150.3	181.7	490.7	-450.5	68.6
	36	93.1	-309.8										
22	0	665.1	-66.7	506.0	417.7	-1009.8	736.1	-9.7	-339.5	-663.5	-250.0	-434.6	-4769.4
	12	-212.2	269.8	-433.7	-3117.0	-1577.6	446.6	1277.8	177.0	-207.5	-894.4	106.8	149.4
	24	181.6	-164.5	-250.8	462.4	394.0	-51.0	858.7	-384.3	-224.2	284.1	-180.1	113.4
	36	-388.8	-299.8										
23	0	-436.5	39.3	-647.8	-529.7	861.0	1176.7	-90.8	225.0	-275.0	78.7	393.9	-94.6
	12	14.2	-835.9	1173.9	-71.2	-161.7	-1.2	-164.6	-133.7	762.6	43.0	-66.3	-1847.2
	24	326.2	326.2	3512.8	-3454.0	-461.6	-352.0	756.4	315.2	-113.0	-56.4	419.0	1243.1
	36	652.6	-3327.1										
24	0	689.4	-193.1	203.7	773.7	647.4	-82.3	-345.4	-950.8	-371.4	67.6	82.8	-221.7
	12	190.5	269.4	313.1	1404.8	1127.4	2191.0	-1819.5	2297.6	1954.1	-1435.1	-6.7	2579.3
	24	-958.0	-171.9	-42.8	218.4	-1114.8	-1009.7	269.0	-1816.4	27.1	522.1	-731.0	-70.8
	36	615.9	699.3										
25	0	-518.1	-88.5	433.4	-115.7	-112.6	-180.2	28.9	-542.4	-101.8	-649.8	-125.3	233.7
	12	820.2	41.4	-352.0	354.9	33.8	8.7	-267.3	-233.5	-125.0	59.1	-660.8	-3416.0
	24	-81.2	353.6	-690.0	445.9	779.9	-2634.8	255.0	-284.1	138.0	-1306.1	368.6	744.2
	36	-4502.8	1485.1										
26	0	-421.4	105.4	-292.9	75.4	-332.3	181.2	-550.8	-136.1	-608.8	315.9	140.7	344.1
	12	667.2	-184.2	-92.1	40.1	349.3	399.2	335.0	235.1	472.2	350.2	-513.6	-74.6
	24	256.5	213.8	57.5	40.0	75.7	3087.4	5067.5	1332.1	572.4	-854.7	-482.6	-945.7
	36	-49.7	-326.1										
27	0	-1800.3	-130.1	1551.1	587.4	2761.2	1168.0	-127.1	150.3	-2852.4	169.0	-2338.8	810.8
	12	-2246.3	1973.0	-399.1	-53.7	108.0	138.9	947.3	-45.1	-250.8	-191.6	359.0	-3.2

	24	372.3	-216.2	-238.9	-483.9	-506.0	699.6	11.1	624.0	232.1	-828.1	1141.3	1442.1
	36	-606.4	-2349.5										
28	0	-1058.6	2040.6	-1693.3	1520.3	-3196.2	-266.7	738.3	-138.4	-502.3	-1296.0	1564.6	270.5
	12	480.5	-574.2	-471.6	-322.6	367.5	3517.6	-463.4	1543.8	-58.0	531.9	287.7	-602.4
	24	-50.3	-573.3	14.0	-31.2	-78.7	205.4	405.6	-659.0	-134.2	9.6	-228.9	189.8
	36	825.2	580.4										
29	0	595.6	410.9	-233.6	813.2	-14.3	520.0	243.3	-82.3	2549.3	-793.9	-283.3	-60.4
	12	4622.7	1381.9	918.1	-547.4	847.2	-731.5	1475.7	1554.9	311.1	-773.0	-119.1	92.0
	24	-2.2	-390.7	-315.1	-668.7	-14.9	772.9	253.5	480.2	247.1	-13.0	-564.3	-340.3
	36	-567.3	-111.9										
30	0	-224.1	114.0	-102.9	401.5	-1361.5	305.1	332.9	176.1	665.0	-405.5	-557.5	-1110.1
	12	731.2	453.2	-930.9	164.4	392.8	-1050.5	-448.1	545.0	-632.2	-260.0	90.0	-883.9
	24	403.3	3116.6	924.9	-1397.7	-1188.2	592.2	-953.6	-918.6	-2215.3	-2761.0	1580.3	-1338.2
	36	752.5	-881.4										
31	0	1480.5	420.4	3070.9	157.1	2450.2	-742.1	1543.9	-1315.2	2900.3	1436.9	2367.8	1019.3
	12	-436.3	-612.9	529.5	261.0	78.6	-626.5	1487.6	-442.8	128.9	-249.7	424.4	-126.2
	24	25.3	-117.6	13.5	372.1	-294.1	-19.2	338.0	472.3	3.4	74.9	-12.6	75.1
	36	-200.5	532.4										
32	0	-1297.2	-784.7	107.8	395.8	-59.5	-163.9	189.7	-155.6	1419.8	-1322.0	-1204.4	-583.1
	12	-1287.2	-2746.9	695.8	-1059.9	-338.4	-617.0	-666.4	2043.3	-104.6	299.6	-3368.0	-622.9
	24	682.6	-229.7	116.7	-932.1	-316.4	537.8	-202.7	-268.6	293.0	-273.9	300.4	1000.2
	36	91.1	-151.7										
33	0	-113.9	391.5	-1290.6	-1633.9	-1.8	657.2	-226.4	-1512.1	-2437.8	4607.5	52.7	-132.3
	12	-404.4	-703.3	1476.7	-536.7	406.2	-971.1	-325.7	132.3	-77.4	-190.3	-396.2	-78.8
	24	543.0	-666.3	-384.1	-13.8	-248.7	70.3	39.4	-455.9	-164.3	-151.3	38.5	-60.1
	36	518.0	-238.1										
34	0	2203.0	-1110.3	-484.2	-925.9	-4177.6	617.9	-1636.7	-869.3	-35.1	1710.0	-1190.2	870.6
	12	-1177.0	1397.3	705.0	661.1	-542.8	-83.0	531.8	395.9	-146.6	401.7	-0.3	857.4
	24	-822.2	147.7	-529.6	-97.1	13.2	-176.2	-265.5	-543.6	-103.9	-104.4	547.9	280.2
	36	-445.2	-268.9										
35	0	1961.1	1652.6	2765.3	-887.7	1633.9	-2130.1	1782.2	-1657.5	-522.5	378.0	-1509.3	779.1
	12	747.7	345.3	-350.7	-1417.9	-163.8	1827.9	1084.2	513.3	-1025.9	-425.6	597.8	-158.5
	24	-55.7	516.8	-496.4	57.1	-331.9	38.2	335.4	-232.2	438.2	127.4	-264.8	164.9
	36	259.7	-1377.7										
36	0	426.3	-127.8	585.9	-595.2	60.5	-823.9	-1867.2	-5381.5	-2652.5	-509.7	839.4	640.1
	12	1589.8	-222.5	523.1	864.5	103.2	-180.1	617.3	-302.0	144.8	103.4	-328.1	-538.3
	24	-321.6	263.1	30.7	94.7	-112.1	-88.7	-209.7	-266.2	-223.2	369.2	154.1	86.2
	36	498.0	157.0										
37	0	-1143.2	-1010.5	573.8	-676.5	400.8	-460.3	-317.6	-933.8	275.4	208.9	-209.5	670.9
	12	48.5	-184.5	-213.7	135.3	-379.4	-400.1	1039.3	-327.4	-670.6	-483.6	-298.1	5202.0
	24	1237.5	-355.4	97.7	-898.6	-333.9	-260.4	137.8	213.4	168.1	-52.1	-674.5	-353.6
	36	404.1	860.2										
38	0	-868.6	1004.9	-516.2	1586.8	529.0	190.8	355.8	-264.6	-2847.1	-1558.5	657.3	124.5
	12	-95.3	-3495.4	566.5	1731.4	-2899.1	-1180.4	347.9	322.7	-463.3	108.7	-204.5	13.5
	24	6.3	315.6	-19.0	188.1	239.2	-65.8	-265.8	-257.4	122.2	158.9	508.5	229.0
	36	4.5	188.1										
39	0	-262.2	-1262.4	167.5	945.5	-123.8	1490.2	-1064.2	1223.0	4596.2	1631.8	1544.8	64.9
	12	-2283.7	-154.6	757.5	367.9	291.0	-1303.0	1004.6	-675.7	46.5	132.1	48.3	484.3
	24	-120.4	-411.1	-358.8	434.0	478.8	649.2	25.0	127.8	-634.5	-306.0	-226.2	-528.2
	36	-111.4	373.9										
40	0	1233.3	-1491.5	1355.3	2517.7	-1435.7	977.8	1136.6	2281.5	-1509.5	707.5	183.3	2415.1
	12	-1460.0	-2256.5	-21.8	1671.6	138.8	-236.4	-852.4	628.5	357.6	853.3	-82.7	732.0
	24	207.1	47.7	-99.3	-374.3	-246.7	689.1	-341.8	563.8	-158.0	234.7	-210.9	-116.4
	36	154.6	381.6										
41	0	-7509.2	459.9	-34.1	-784.0	122.4	-720.8	-230.7	-408.2	978.5	-232.3	-693.6	-301.5
	12	364.5	-222.6	883.3	458.8	613.1	166.4	574.8	14.6	-312.0	-75.4	-276.4	-389.6
	24	-669.9	101.3	-4.1	549.0	-116.2	221.0	-488.7	-605.2	-147.8	-363.4	47.8	237.8
	36	96.2	935.4										
42	0	-1.2	302.2	433.0	-301.7	-1011.6	-1137.1	-697.9	6074.6	-1765.9	595.9	8.2	293.1
	12	760.5	-435.6	513.0	291.5	-410.4	570.4	546.3	299.6	135.0	-125.5	-195.4	-199.9
	24	-187.6	789.2	-108.5	29.2	76.3	-508.0	-157.2	-351.1	316.8	628.6	115.2	236.8
	36	-80.5	325.9										
43	0	71.7	256.8	-540.1	464.6	113.0	-400.4	-361.8	-358.5	624.3	-1812.1	-1398.7	-3355.7
	12	-3904.4	1705.4	-173.3	847.3	2489.8	603.5	1.9	-1119.1	-35.3	154.6	-50.6	-35.0
	24	-268.6	-336.2	-252.5	-133.6	-93.1	-362.3	299.0	85.0	-477.7	-41.0	169.7	-153.5
	36	141.4	-404.9										
44	0	51.2	-976.8	-518.6	8.8	829.2	44.6	-78.3	3.9	197.8	-14.8	-8.2	-275.2
	12	-718.8	124.8	-585.9	-59.0	292.9	-614.4	257.0	-322.4	657.4	-183.3	-560.6	442.0
	24	-457.2	-9.7	103.1	5274.1	-118.3	375.1	306.7	746.4	-189.1	-3177.1	-330.4	183.5
	36	228.3	-565.8										
45	0	-470.4	-93.3	568.5	360.5	140.9	830.8	-622.9	1496.2	681.4	96.0	306.5	-500.8
	12	505.7	-637.0	151.4	-392.3	706.6	448.9	-1000.9	-818.9	-2404.2	3414.4	1995.4	-77.6
	24	789.0	477.7	668.5	399.0	466.9	-962.2	-282.0	315.1	453.3	748.1	399.2	533.6
	36	1848.7	4104.7										
46	0	350.0	-5.3	645.3	313.5	-566.6	-224.5	655.5	224.2	-448.6	-108.4	-75.3	-340.2
	12	421.5	-640.2	-253.9	571.1	538.9	-561.6	-583.9	171.1	-152.3	199.7	-853.6	263.1
	24	-741.3	-5340.2	893.5	233.1	-600.0	-511.0	474.8	-711.5	292.3	-147.3	-125.3	511.1
	36	-104.7	151.9										
47	0	108.4	-494.3	-1587.2	-291.7	-37.1	409.4	-1617.4	-1098.2	-709.6	379.1	798.7	-1046.5
	12	1566.6	-525.8	-871.7	-481.3	-233.1	-437.4	-227.2	-1510.2	-442.9	-152.2	74.9	-26.2
	24	-318.8	-347.3	212.3	244.3	-68.9	295.4	-3787.9	1153.8	-63.8	-832.7	-1062.9	1289.0
	36	-2125.6	-1585.5										
48	0	-2766.0	1432.0	-980.3	-1259.6	2287.7	-915.3	-3819.7	1708.8	-929.4	427.8	471.6	1156.8
	12	-513.2	-68.0	714.3	695.2	275.3	-1134.5	-327.8	966.3	-404.6	-343.8	-415.1	-242.1
	24	702.3	260.3	-148.6	40.7	34.9	-358.2	-150.5	-40.7	156.8	30.3	33.9	-952.9
	36	-509.4	-46.4										
49	0	1028.1	951.3	180.7	745.2	-699.0	-461.2	357.7	-453.7	298.2	164.3	-11.8	265.8
	12	-204.1	1686.6	241.7	-224.1	-239.3	-271.8	-6116.3	-238.7	153.4	120.7	58.6	-53.8
	24	-49.0	145.1	-22.1	271.0	121.8	-142.1	28.1	-164.0	299.5	104.9	-175.2	-181.7
	36	-13.5	-543.0										
50	0	303.9	-361.0	2015.4	-1568.3	1258.6	535.2	423.1	-1308.6	22.5	931.2	-119.5	-415.9
	12	541.3	1626.2	-3002.1	2672.9	-1944.7	434.1	-62.8	397.9	480.1	-1094.1	295.2	15.8
	24	757.9	201.9	-251.1	337.4	473.6	269.3	-438.4	-265.9	-74.6	63.4	-161.1	86.0
	36	583.2	761.1										
51	0	-11.2	804.6	-22.2	-126.7	1865.0	-805.4	1098.7	-1069.3	-658.3	-656.7	60.0	91.6
	12	-647.7	-853.2	-309.8	-222.6	159.4	-355.3	1783.9	-3439.6	512.5	-79.9	805.2	-159.6
	24	-471.8	144.7	-1819.0	360.5	-219.8	523.4	485.4	-1771.9	-1468.3	705.4	973.0	-478.9
	36	-159.2	1910.0										
52	0	332.1	-292.6	-391.0	-761.2	623.7	-1536.8	-749.8	-1008.0	1353.1	-81.9	2715.6	437.9
	12	551.2	921.7	849.8	1066.2	-1600.3	365.3	-1554.0	-483.1	-169.1	2045.1	785.6	-829.6

	24	-330.7	-564.0	-1366.3	-1021.6	-854.8	506.2	411.3	400.4	-286.9	-474.9	-780.5	717.0
	36	961.4	-3591.8										
53	0	-381.5	968.8	122.9	-434.4	-2153.3	-1215.8	538.5	661.9	141.5	1532.4	-1461.6	1777.9
	12	594.6	1814.3	-1636.9	-1063.4	313.3	-2224.8	4.7	327.3	-1698.1	277.5	-1507.8	709.8
	24	612.4	342.4	529.5	719.0	45.2	694.1	665.0	375.4	-383.7	432.1	-470.6	240.1
	36	-1155.0	-1105.3										
54	0	2258.0	-740.3	45.8	1770.5	89.7	106.3	-722.4	-336.3	163.9	392.4	734.3	-425.3
	12	502.5	-1456.2	-1300.9	517.3	597.4	-1471.5	-279.1	-897.7	116.1	-1757.8	1335.1	-513.9
	24	2090.6	421.7	-279.9	-6.6	-87.4	-46.7	-573.1	-258.5	2897.4	-635.4	-170.3	241.2
	36	-293.2	277.6										
55	0	-3022.5	112.5	267.4	-2126.7	1440.1	4089.7	830.1	-1296.7	574.3	449.1	358.3	1337.4
	12	-30.9	7.6	747.3	-512.2	178.8	23.9	-443.5	-902.4	125.7	100.3	546.6	-1095.2
	24	341.1	250.3	-242.3	681.5	272.4	456.3	-458.8	273.3	250.0	52.3	73.6	283.5
	36	-636.0	-226.1										
56	0	2763.4	159.3	-441.2	18.2	3177.3	-650.7	1004.5	365.2	-31.8	426.7	166.9	-822.3
	12	-251.7	186.9	-1079.6	-982.3	-413.2	101.8	190.8	111.3	758.3	3198.2	-310.3	-351.2
	24	-51.5	246.9	-139.1	-705.9	-187.9	-31.0	-374.9	197.3	167.1	-339.8	-215.0	29.8
	36	-450.9	-80.0										
57	0	334.2	-386.1	266.5	3178.5	-7156.6	1314.5	-82.2	448.3	77.3	61.9	-433.5	-957.1
	12	-82.9	-326.3	359.0	554.4	-468.9	-253.6	83.2	-335.9	175.3	340.9	-31.6	491.7
	24	419.4	26.2	-276.0	387.5	-590.5	-411.7	-225.6	433.0	112.1	359.5	569.2	50.3
	36	104.2	-430.1										
58	0	106.0	579.0	650.9	530.4	-518.5	820.6	29.0	685.2	723.1	839.4	139.9	306.8
	12	-117.4	-491.2	-715.9	410.6	-259.8	507.1	-335.2	-192.9	5802.4	-246.9	-953.8	-224.2
	24	-400.8	-168.7	-60.0	-204.0	-140.7	-115.0	64.1	82.8	239.9	211.7	0.5	-99.9
	36	-223.0	-227.9										
59	0	-572.1	885.3	-482.8	1724.7	115.0	2675.1	658.9	3054.1	-255.8	759.8	-105.0	1015.7
	12	1445.8	1398.8	-690.5	96.6	46.8	354.7	210.4	-993.7	-227.2	-196.3	7.2	102.6
	24	105.2	264.5	-578.7	-621.0	-860.7	-206.3	-375.3	353.3	-631.5	1260.6	-2883.2	58.3
	36	1957.2	-62.1										
60	0	-869.9	-176.0	851.6	-2679.5	3215.9	1892.2	-1023.7	1043.8	1329.8	-2458.6	-2328.1	-1561.9
	12	-350.9	-179.7	-15.8	-776.7	452.4	-1456.5	-366.3	285.7	-223.4	532.0	-206.7	107.4
	24	-31.4	-74.6	106.7	-32.4	342.0	-294.9	428.7	-140.9	-533.4	-143.0	-91.8	63.5
	36	75.3	-5.2										
61	0	-644.6	-1163.1	590.7	-2801.0	-3478.8	807.2	1746.7	555.6	-526.5	-2010.8	-1161.5	-657.7
	12	48.7	200.1	-153.2	742.3	-315.0	1195.8	498.1	274.0	1941.1	327.7	106.5	0.6
	24	259.5	-192.2	-340.2	468.8	-522.0	399.9	162.0	278.6	346.9	230.8	-298.6	-26.2
	36	-171.6	53.4										
62	0	-756.9	3368.4	-953.2	-327.8	1459.1	-602.6	82.3	972.0	-822.2	247.4	-4164.3	501.7
	12	455.2	449.7	-35.5	1196.0	1661.6	313.4	-85.9	-489.0	32.8	-417.9	65.8	-262.9
	24	-410.4	186.9	-58.7	463.3	-249.1	16.8	-64.5	-333.0	141.8	-87.2	-196.4	108.0
	36	84.7	-882.4										
63	0	-103.3	-185.8	60.1	-23.7	-454.5	-540.5	-561.5	-3491.8	3881.0	3696.6	-158.7	1014.0
	12	1080.3	-50.5	-658.2	512.1	505.2	130.8	273.1	272.8	186.8	-200.7	6.4	99.1
	24	-116.5	551.4	219.6	365.6	-91.9	-271.2	-352.3	-447.5	-211.8	205.9	-337.7	236.1
	36	-589.2	242.2										

Table C. 27 Shape codebook 0 for MODE_VQ == SCL_1, BLEN_TYPE == MEDIUM

Table C. 28 Shape codebook 0 for MODE_VQ == SCL_2, BLEN_TYPE == MEDIUM

Table C. 29 Shape codebook 1 for MODE_VQ == SCL_1, BLEN_TYPE == MEDIUM

Table C. 30 Shape codebook 1 for MODE_VQ == SCL_2, BLEN_TYPE == MEDIUM

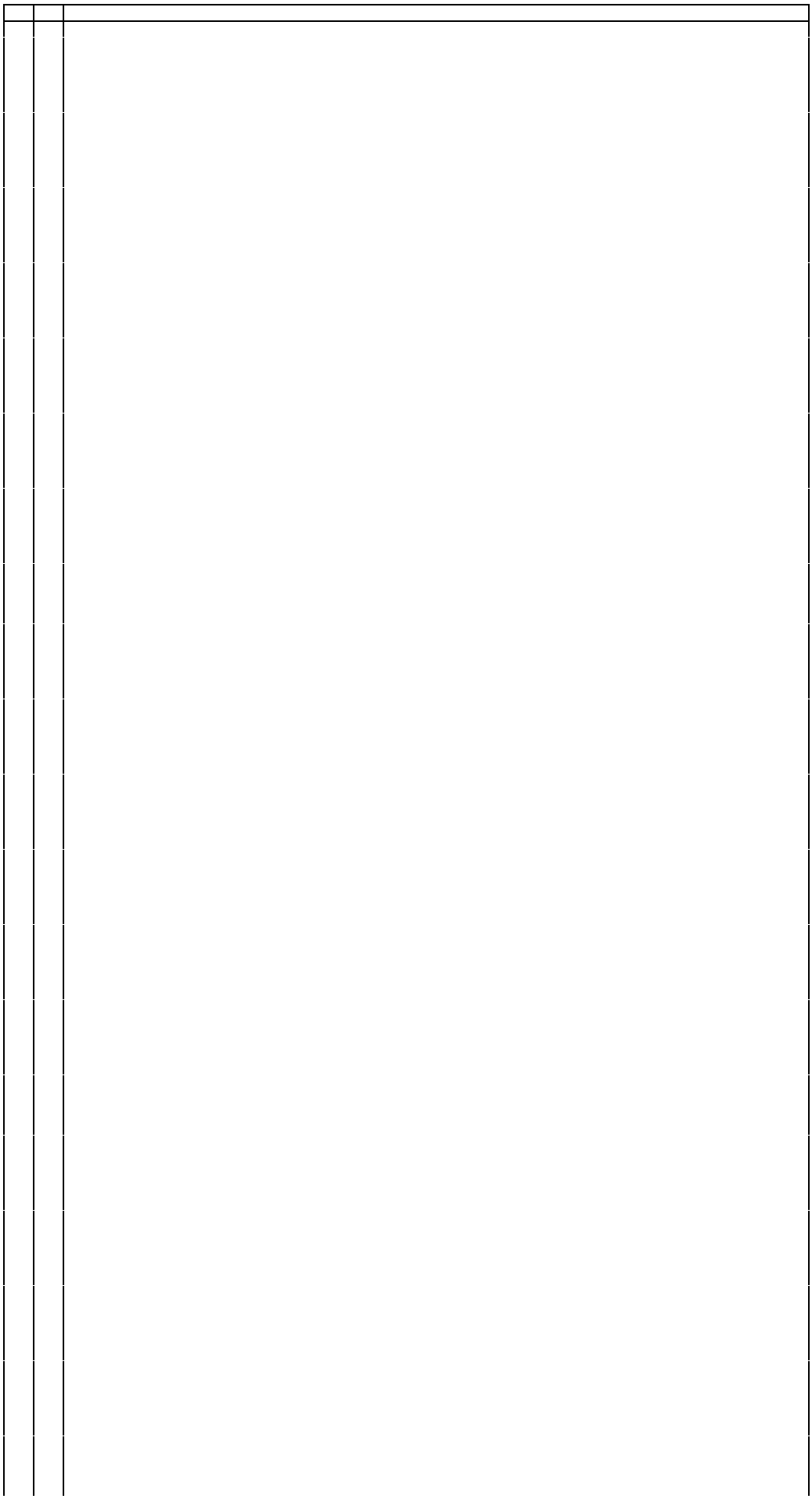


Table C. 31 Bark codebook for MODE_VQ == 16_16, BLEN_TYPE == LONG

VN	0	1	2	3	4	5	6
0	-0.43231	-0.39214	-0.37133	-0.18066	-0.15706	-0.36897	-0.57702
1	-0.58724	-0.56533	0.16131	-0.44199	0.56264	0.25027	0.51298
2	0.85492	-0.48008	-0.46088	0.28242	0.00519	0.14086	0.22066
3	-0.69482	-0.62906	-0.47856	-0.21593	-0.03997	-0.36647	7.18764
4	-0.53376	-0.08242	0.5373	-0.14057	-0.32657	-0.13935	0.25969
5	-0.60529	-0.62587	-0.61671	-0.29053	0.02674	0.23451	1.44358
6	0.9162	-0.26778	0.10622	-0.35448	-0.38034	0.05357	0.33657
7	-0.62391	-0.52491	-0.56602	-0.53809	0.3944	0.94184	0.50347
8	-0.58437	-0.53307	-0.55707	0.23914	-0.12434	0.20074	0.48747
9	-0.59719	-0.58423	-0.54615	0.79767	0.60613	0.3976	0.40115
10	0.26819	0.09919	-0.59141	-0.39232	0.2251	0.47702	0.60308
11	3.17158	-0.40643	-0.42192	-0.45658	-0.26799	0.08946	-0.31408
12	-0.54602	-0.55865	-0.53949	-0.52958	-0.48183	-0.11345	0.67107
13	-0.32607	0.1385	-0.61266	-0.14789	1.04014	-0.13136	0.94123
14	-0.62391	-0.58978	-0.54873	-0.02335	0.48127	0.42702	0.34805
15	-0.61115	-0.56908	-0.22591	-0.51193	-0.05606	0.46968	0.57918
16	-0.54785	-0.49301	-0.50797	-0.42665	0.53223	-0.46383	0.38475
17	0.49874	-0.35246	-0.48492	-0.46517	-0.32768	-0.26714	0.51968
18	10.26073	-0.56483	-0.5565	-0.52946	-0.39971	-0.57278	-0.72029
19	-0.60825	-0.63806	-0.57991	-0.2042	1.38183	0.4243	0.52997
20	0.59165	-0.39176	-0.58381	-0.03867	-0.37588	-0.35525	1.83752
21	-0.42043	-0.33384	-0.32534	1.51496	-0.23055	0.05344	0.36783
22	0.14581	-0.3612	0.5165	-0.01334	-0.13269	-0.05144	0.29691
23	1.96188	-0.4342	-0.43237	-0.38104	-0.4165	-0.26605	0.67067
24	0.22204	-0.61549	-0.63823	-0.42224	-0.5307	0.55124	0.86639
25	-0.52672	-0.55103	-0.58402	0.36025	0.08423	1.14429	0.63417
26	-0.50633	-0.33728	0.39	0.63329	-0.11317	0.35685	0.45356
27	-0.50855	0.1461	-0.45831	0.35249	0.11592	0.55004	0.37539
28	-0.40556	0.26609	-0.59518	-0.41623	-0.33616	0.41829	1.17118
29	-0.57534	2.22492	-0.52259	-0.31341	0.15449	0.24491	0.42272
30	-0.56593	-0.61026	-0.51872	-0.50739	-0.36322	2.60332	0.7144
31	-0.49376	-0.21894	-0.20904	0.49758	-0.21371	-0.41429	0.55119
32	-0.46182	0.65957	-0.36831	-0.43364	-0.18239	1.16538	0.42759
33	-0.13924	-0.4726	-0.50015	-0.13143	0.29094	0.39173	0.31037
34	-0.47953	0.34941	-0.12263	-0.11068	-0.33185	0.13473	0.18276
35	-0.44732	-0.28358	-0.18224	0.11231	0.7992	0.10606	0.22311
36	-0.12523	-0.35336	-0.1757	-0.35804	-0.26558	0.02485	0.21148
37	-0.49371	1.23568	-0.32171	-0.24921	-0.23034	0.023	0.36843
38	0.41277	0.12901	-0.16441	0.23991	-0.11031	-0.09042	-0.17742
39	-0.02173	0.34159	-0.3429	-0.06921	0.22281	-0.07819	0.19302
40	-0.48977	-0.34898	-0.46965	-0.34419	-0.39005	0.6577	0.21881
41	-0.02418	-0.22985	-0.31312	0.28567	-0.30881	0.42568	0.26742
42	1.18911	-0.42965	-0.54039	-0.44923	-0.29794	0.82152	0.12966
43	0.44277	0.62582	-0.33214	-0.40822	-0.30272	0.1046	0.55718
44	1.30304	0.45789	-0.30661	-0.44706	-0.23421	0.06173	-0.30169
45	-0.44553	0.75625	-0.17313	0.61036	-0.17917	0.04263	0.45259
46	-0.42021	1.04044	1.19288	-0.44572	0.02881	0.28087	0.51597
47	0.07319	-0.51369	-0.51263	0.54409	0.21598	0.0671	0.32422
48	-0.56279	-0.31502	0.05037	-0.01371	0.20297	0.93992	0.35616
49	-0.35415	-0.3424	1.82916	-0.39943	-0.2364	0.2533	0.38421
50	-0.59675	-0.11642	-0.63	-0.37934	0.71552	0.9046	0.57453
51	0.23443	-0.05418	-0.2452	-0.44529	-0.29091	1.4389	0.54011
52	-0.64425	-0.64742	-0.59373	-0.5926	-0.47045	1.19414	0.99384
53	-0.55485	-0.49038	-0.06218	0.29027	0.16781	0.17886	0.19593
54	-0.48536	-0.2786	0.04534	-0.45933	-0.53073	-0.33521	1.30464
55	0.34328	-0.45097	-0.07918	-0.04394	0.22559	0.15395	0.18158
56	-0.57549	-0.00088	-0.41325	-0.36835	0.16146	0.11296	0.29523
57	-0.52428	-0.51067	0.68306	-0.36499	-0.12048	0.67998	0.45323
58	-0.45251	0.4444	0.32699	-0.2829	0.45174	0.30766	0.40806
59	-0.29075	-0.13123	0.08303	-0.03219	0.14427	0.16467	0.13125
60	1.57703	-0.49443	-0.3944	-0.05263	0.37818	-0.01234	0.0323
61	-0.56762	0.71463	-0.59478	-0.42897	0.53002	0.39741	0.43878
62	-0.60911	-0.415	-0.32248	-0.33826	-0.37865	-0.15125	3.8516
63	0.40233	-0.41002	-0.55742	-0.32885	0.8796	0.26069	0.41012

Table C. 32 Bark codebook for MODE_VQ == 16_16, BLEN_TYPE == MEDIUM

VN	0	1	2	3	4	5	6	7	8	9
0	-0.65633	-0.58427	-0.46259	-0.40985	0.09297	0.1453	0.25094	0.35713	0.26324	0.53063
1	0.84498	-0.1281	0.56286	-0.19093	-0.01979	0.04667	-0.10229	-0.00389	0.01663	-0.13793
2	0.04682	-0.22334	-0.30508	-0.30208	-0.2798	0.06158	0.00899	0.3529	0.08712	0.18229
3	0.19783	0.17905	-0.32094	-0.05227	-0.2194	0.44921	-0.0842	-0.04617	0.0131	0.01114
4	0.91152	-0.35612	-0.2974	-0.25133	-0.12918	0.05086	0.04682	0.11292	0.08019	0.14439
5	5.62637	-0.11788	-0.47308	-0.4662	-0.39859	-0.44335	-0.41053	-0.44914	-0.6691	-0.27031
6	-0.42587	0.59485	-0.36976	-0.29791	0.1364	-0.01436	0.10385	0.00275	0.04751	0.09661
7	-0.2015	0.1193	1.2496	-0.25774	-0.07422	-0.00296	-0.08146	-0.12347	0.09869	-0.03131
8	-0.46039	-0.50191	0.62462	0.50424	0.22988	0.041	0.00264	0.06506	0.07019	0.05808
9	0.28384	0.62291	-0.15215	-0.05407	0.10329	-0.12654	-0.1473	-0.01339	-0.07146	-0.05701
10	0.4974	-0.24113	-0.32651	-0.2057	0.97711	-0.04274	-0.13879	-0.03846	0.08489	0.1067
11	-0.08812	-0.18619	-0.19133	-0.22517	-0.15531	1.33269	-0.13227	-0.01014	0.12661	0.03513
12	-0.44983	0.07524	-0.22939	-0.02266	-0.20014	0.13359	0.02401	0.0226	0.09292	0.14958
13	0.47367	-0.27117	-0.15103	0.16105	-0.08958	-0.10911	-0.02888	0.06293	0.03571	0.05161
14	-0.06482	0.10236	-0.17405	-0.04329	0.13077	-0.10937	0.00459	0.03903	0.02092	0.03649
15	0.00602	-0.03368	0.30541	0.35192	-0.13283	-0.07066	-0.197	-0.19021	-0.03502	-0.09905
16	-0.24985	0.42774	0.24318	-0.24044	-0.23406	0.02453	0.12392	0.038	0.04058	0.00303
17	0.11038	-0.35911	0.54225	-0.1843	-0.185	0.11304	0.04623	0.01522	0.06867	0.0574
18	0.19943	-0.45098	-0.23426	-0.14589	0.21541	0.19624	0.10158	0.00873	0.02889	0.04734
19	-0.10669	-0.33358	-0.06158	0.18264	-0.08082	-0.16137	0.20329	0.03399	0.01439	0.03887
20	1.93992	-0.26184	-0.22727	-0.00135	-0.10834	-0.0748	-0.08455	-0.0928	-0.05429	-0.08684
21	-0.37363	1.37781	-0.22992	-0.19731	-0.16318	0.05829	-0.02265	-0.00527	-0.03529	0.00901
22	0.17866	-0.34112	-0.32714	1.11494	-0.06458	-0.02111	-0.0127	-0.00251	0.09991	0.07575
23	-0.32559	0.35654	-0.29473	0.67281	-0.18443	-0.01199	-0.05298	0.04827	0.00251	0.03749
24	-0.52563	0.02948	-0.37935	-0.58826	-0.44134	-0.28436	0.04742	0.97967	0.10241	0.12339
25	-0.41813	-0.21819	0.24415	-0.30268	-0.06098	0.02846	-0.00957	0.01384	0.07111	0.07643
26	-0.28814	-0.13392	-0.11584	-0.17619	0.75542	-0.06941	0.01142	0.08032	0.0326	0.03451
27	-0.44123	-0.39571	-0.27102	0.35133	-0.07125	0.22665	0.02231	0.10938	0.0854	0.05928
28	-0.1918	-0.47181	-0.36035	-0.33536	-0.18342	-0.09316	-0.12421	0.04315	0.94217	0.31813
29	0.47845	0.32908	-0.45908	-0.48147	-0.50902	-0.3352	0.05815	-0.00462	0.06631	0.08606
30	-0.1947	-0.31193	-0.22235	-0.33628	-0.28743	-0.07146	1.35561	-0.07581	0.11227	0.11591
31	0.25175	-0.06033	0.18574	-0.28091	0.14143	-0.08635	-0.01938	0.02455	0.03297	0.04312

Table C. 33 Bark codebook for MODE_VQ == 16_16, BLEN_TYPE == SHORT

VN	0	1	2	3	4	5	6	7	8	9
0	-0.79673	-0.86666	-0.8557	-0.80763	-0.71574	-0.56341	-0.31198	0.22426	2.1781	2.13975
1	-0.69681	-0.7811	-0.78475	-0.66121	-0.48024	-0.55836	0.57957	1.53573	1.03428	0.55032
2	0.00888	0.09265	0.3562	-0.10061	-0.22199	0.68243	-0.12937	-0.1021	-0.2566	-0.4994
3	3.34309	0.21426	-0.38333	-0.47773	-0.49795	-0.50456	-0.51193	-0.51848	-0.49015	-0.42489
4	0.75319	1.82131	0.39478	-0.19971	-0.39367	-0.46868	-0.49689	-0.52713	-0.56443	-0.6017
5	2.16909	-0.32057	-0.63806	-0.6382	-0.67183	-0.63128	-0.58935	-0.42349	0.26657	1.10431
6	-0.63918	-0.55557	-0.52541	-0.58612	-0.39876	0.1116	0.88218	1.03311	0.62599	-0.31485
7	0.65199	0.07572	-0.3946	-0.30414	-0.36049	-0.29578	-0.00081	0.16333	0.21258	0.2306
8	0.97144	-0.15022	0.25323	-0.23275	-0.23063	-0.13977	0.3532	-0.16914	-0.28023	-0.34759
9	-0.35682	0.78611	0.06827	-0.27638	-0.21767	-0.173	0.0966	0.10617	-0.01377	-0.06438
10	-0.12432	0.60909	1.12759	-0.00634	-0.35062	-0.26923	-0.29621	-0.30427	-0.19919	-0.34806
11	-0.26492	-0.28814	-0.28326	0.048	0.79907	-0.13509	-0.15889	0.20816	0.01817	-0.07321
12	-0.24668	-0.34127	-0.41921	-0.45473	-0.35545	-0.42696	0.19463	3.09504	-0.45511	-0.37229
13	0.38338	-0.44794	-0.55575	-0.58726	-0.6001	-0.58793	-0.47679	-0.22022	1.1305	1.63811
14	0.41264	0.00261	0.11482	0.80344	0.07455	-0.19187	-0.09388	-0.33713	-0.47865	-0.48021
15	5.14649	0.20586	-0.58572	-0.65551	-0.7274	-0.75906	-0.76571	-0.74437	-0.77385	-0.69471
16	-0.16022	-0.26842	0.73824	-0.14801	-0.18018	-0.16929	-0.04393	0.16006	-0.00455	0.02877
17	-0.34997	-0.13285	-0.19849	0.24082	0.06316	0.0662	0.15586	-0.02599	0.10279	0.2625
18	0.02052	0.04985	0.02123	-0.00506	0.01232	-0.04066	-0.02229	-0.03727	0.01617	-0.01002
19	1.18337	-0.10429	-0.42223	-0.43019	-0.45645	-0.36608	-0.2998	-0.2637	0.27183	0.68043
20	-0.01833	-0.099	-0.03879	-0.3019	-0.30526	-0.28148	-0.15746	-0.09792	0.15371	1.01942
21	-0.33197	-0.33511	-0.0963	-0.19338	-0.21606	-0.13505	0.00565	0.64704	0.47872	0.2247
22	-0.52779	-0.51375	-0.38363	-0.3852	-0.21106	0.10151	0.65614	0.13736	0.40331	0.6218
23	0.04232	-0.117	-0.35779	-0.26209	-0.28332	0.43139	0.10528	0.05302	0.15853	0.22114
24	-0.26237	-0.58637	-0.7059	-0.7097	-0.71038	-0.66721	-0.53499	-0.36097	0.47437	3.93928
25	-0.48365	-0.58238	-0.57454	-0.57321	-0.45787	-0.26422	-0.09673	0.25737	0.85705	1.75366
26	-0.17291	-0.18068	0.07726	-0.23475	-0.0966	0.02764	0.96015	0.00135	-0.17454	-0.27849
27	1.82751	0.31842	-0.00417	-0.27243	-0.26025	-0.25565	-0.24979	-0.33001	-0.35621	-0.34853
28	0.32577	0.98843	-0.19465	-0.37047	-0.33564	-0.30227	-0.28323	-0.08041	0.06564	0.23567
29	-0.2183	-0.38881	-0.42971	-0.48935	-0.34245	-0.30764	-0.0904	0.21298	1.31502	0.51985
30	-0.39417	-0.41843	-0.30417	-0.04832	0.19775	0.77778	0.068	0.05766	-0.05497	0.06411
31	0.55228	-0.27188	-0.24463	-0.08926	0.21794	0.00091	0.0605	-0.11298	-0.17315	0.0377

Table C. 34 Bark codebook for MODE_VQ == 24_06 / 24_06_960, BLEN_TYPE == LONG

VN	0	1	2
0	5.57803	-0.04511	0.14882
1	-0.14642	-0.52823	0.15552
2	1.46255	0.62953	0.0494
3	-0.58318	0.01845	0.44599
4	1.94569	0.16055	0.16818
5	0.38212	1.09395	0.34245
6	0.19341	-0.5285	0.54847
7	0.72589	0.1179	0.43367
8	-0.09521	0.9354	0.07766
9	1.04508	0.17831	0.14994
10	-0.55363	-0.5876	0.4068
11	-0.33911	-0.00601	0.11204
12	-0.09404	0.24388	0.30549

13	0.80652	-0.48609	0.98259
14	-0.31829	0.66775	0.21812
15	-0.55757	1.07386	0.10685
16	0.52656	-0.46301	0.20881
17	0.20414	0.40772	0.1272
18	-0.23242	0.14858	-0.81754
19	1.98239	0.70013	0.7957
20	0.00656	0.43113	-0.71401
21	-0.28605	2.93923	0.17068
22	-0.54387	-0.18871	0.22269
23	0.67231	0.75537	0.13019
24	2.92919	0.60565	0.04658
25	-0.55578	0.18691	0.10124
26	0.3276	0.25346	-0.86729
27	1.41016	0.12933	0.36326
28	-0.60288	0.59051	0.09739
29	0.25933	0.69872	0.06769
30	0.24094	0.04998	0.38604
31	-0.42492	0.43497	-0.1672
32	-0.53061	1.3707	0.49726
33	-0.19264	4.29195	0.15791
34	-0.56514	-0.38389	-0.04548
35	-0.03075	1.26534	1.03056
36	0.46567	0.09426	0.10218
37	-0.36526	-0.42236	0.84785
38	1.11214	-0.13201	2.75933
39	-0.48238	1.74259	0.09196
40	2.02169	-0.51073	0.73245
41	-0.21122	0.40243	0.07799
42	1.53373	-0.45024	0.05635
43	-0.59811	0.37008	0.34332
44	0.54613	0.3234	0.20755
45	-0.00247	0.03068	0.02409
46	-0.37993	0.26549	0.78397
47	-0.14357	-0.30933	1.63986
48	-0.17086	-0.17005	0.41668
49	-0.0291	0.64778	0.42964
50	0.76432	0.39577	-0.04321
51	0.45225	-0.13631	-0.14334
52	1.91888	1.77166	-0.11202
53	1.03102	0.54233	0.35026
54	0.10395	0.21665	-0.91012
55	0.43476	0.51159	0.5237
56	1.09371	1.08619	0.48118
57	0.15706	-0.20391	0.15777
58	3.07149	-0.48515	0.35324
59	-0.58184	0.81648	0.50161
60	0.81027	-0.13531	0.1387
61	1.10831	-0.49851	0.42631
62	0.61754	1.69691	0.24082
63	-0.32568	2.18242	0.73073

Table C. 35 Bark codebook for MODE_VQ == 24_06 / 24_06_960, BLEN_TYPE == MEDIUM

VN	0	1	2
0	0	0	0
1	0	0	0

Table C. 36 Bark codebook for MODE_VQ == 24_06 / 24_06_960, BLEN_TYPE == SHORT

VN	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0

Table C. 37 Bark codebook for MODE_VQ == 08_06, BLEN_TYPE == LONG

VN	0	1	2	3	4	5	6	7	8	9
0	-0.61993	-0.79962	-0.65882	-0.81655	-0.69414	1.34306	0.10404	1.73991	-0.03634	0.75959
1	-0.73034	-0.56074	-0.57261	0.65971	-0.2366	-0.23097	0.69256	0.27201	0.43937	0.31025
2	2.9904	-0.55705	-0.56383	0.38243	-0.70521	-0.7373	-0.50862	0.87552	2.17747	0.64719
3	1.00369	-0.4498	-0.38769	-0.34861	0.79363	-0.02489	0.03815	-0.07812	0.17636	0.41776
4	5.0207	-0.74895	-0.53973	-0.7725	-0.83248	-0.82911	-0.70352	-0.48884	0.42229	0.99546
5	-0.65816	-0.41358	-0.693	0.71976	-0.66861	-0.58104	-0.54975	-0.32012	-0.06558	0.46393
6	-0.68265	-0.10841	-0.73145	-0.61647	1.26817	0.35608	1.23105	-0.26588	0.22323	1.005
7	1.86619	-0.76592	-0.77053	0.98334	-0.83867	-0.79105	-0.63102	1.35311	1.24969	2.41341
8	-0.73123	-0.76986	-0.84196	-0.75717	-0.86695	-0.84695	-0.53606	-0.31023	0.94469	1.3215
9	-0.68353	-0.48573	0.82294	-0.55814	-0.24961	0.77891	-0.13437	0.42058	0.23636	0.46779
10	-0.67104	-0.37352	-0.53989	-0.72926	-0.8195	-0.79631	-0.2545	2.509	0.02611	0.65
11	2.39724	-0.67209	1.0422	-0.80007	0.76296	-0.76448	-0.72606	-0.79359	1.57864	2.31549
12	-0.49661	-0.6608	-0.5919	-0.774	-0.63941	-0.55617	-0.03487	-0.4007	2.89463	1.45868
13	-0.31307	-0.64179	-0.58506	-0.57719	-0.49113	-0.26493	-0.62891	-0.59119	-0.53864	3.3524
14	-0.55195	0.67166	-0.43176	0.60603	-0.26008	-0.08468	-0.0334	0.00768	0.15104	0.09388
15	-0.55103	-0.54076	-0.41463	-0.65583	1.09078	-0.62576	-0.26152	-0.33168	1.89377	0.91103
16	-0.6972	2.20165	-0.49952	-0.36484	-0.54553	-0.53922	-0.01628	0.32712	0.59402	0.72392
17	-0.66443	-0.52513	-0.72415	-0.67879	-0.66706	-0.36845	-0.50915	-0.37413	1.69298	-0.30472
18	-0.47266	-0.53791	-0.19213	-0.43477	-0.21855	-0.48593	1.14579	-0.4439	-0.38311	0.47716
19	-0.55749	-0.58676	2.56678	-0.75796	-0.2621	-0.69294	-0.00026	0.29113	0.27171	0.8079

20	-0.54907	-0.59849	-0.40532	0.61891	-0.35144	0.87459	-0.26687	-0.14511	0.47765	0.18502
21	1.3059	-0.6161	-0.56635	-0.46459	-0.50171	1.23897	-0.22964	-0.17596	-0.13096	0.53633
22	-0.56373	-0.73905	-0.35638	-0.71069	2.76952	-0.60402	-0.39084	-0.42708	0.79587	0.62186
23	0.93226	-0.30452	-0.35384	0.69212	-0.27301	-0.15919	-0.12586	-0.09245	0.12976	0.10754
24	-0.139	-0.63619	-0.6214	-0.59748	-0.41136	2.11855	-0.45617	-0.48902	-0.24215	1.43759
25	2.09745	-0.45126	-0.49061	-0.53602	-0.50026	-0.16062	0.84784	0.05219	0.22104	0.07728
26	-0.50041	-0.26448	-0.63317	2.13725	-0.54552	-0.59796	-0.41206	0.53574	0.54778	0.61091
27	-0.40498	-0.67091	1.11916	-0.7352	0.90374	-0.56331	-0.46065	-0.47849	-0.26077	0.7054
28	-0.73768	-0.72976	-0.56033	-0.68217	0.74982	-0.38773	-0.01041	0.64142	0.31192	0.49593
29	-0.52373	-0.62281	-0.56968	-0.78249	-0.79025	-0.67373	2.21733	-0.11079	1.05942	0.43362
30	-0.66854	1.16827	-0.53114	-0.46298	0.52453	0.32947	0.1876	0.0609	0.2443	0.36599
31	0.19468	-0.53282	-0.35223	-0.05596	-0.00782	0.2019	0.09786	0.06598	0.14191	0.0927
32	-0.57873	-0.58019	-0.65396	-0.75659	-0.42532	-0.28044	1.13921	-0.16915	0.27084	1.8425
33	-0.60932	-0.74779	-0.68791	-0.63999	-0.49214	1.39695	1.20415	0.29714	0.52765	0.59786
34	-0.53881	-0.72119	0.92025	-0.68954	-0.4417	-0.71932	-0.41333	-0.09199	0.38212	1.00095
35	-0.71938	0.74779	-0.74746	-0.71267	-0.75519	-0.61527	-0.41802	0.58173	0.31862	0.45583
36	2.47691	-0.71099	-0.80737	1.83561	-0.85496	-0.2596	-0.67923	2.09832	-0.18314	-0.36688
37	-0.7669	-0.60691	0.7137	-0.62831	-0.44586	-0.30366	0.86354	0.60882	0.63054	0.38083
38	-0.66452	0.80173	0.54805	-0.30719	-0.16061	-0.0962	0.11798	0.17559	0.27371	0.26191
39	-0.71998	-0.2472	-0.74412	0.14581	-0.75762	-0.61274	-0.48097	1.75956	2.10806	0.46804
40	-0.43255	-0.11784	-0.56602	-0.45416	-0.61028	0.72776	-0.40307	0.13424	0.16983	0.15021
41	-0.51771	-0.50171	-0.39515	-0.51905	0.75886	0.83434	-0.11834	-0.01136	0.20709	0.39257
42	-0.64345	-0.41671	0.76718	0.92721	0.11702	-0.27145	0.14563	-0.07875	0.37955	0.37379
43	-0.71527	0.51415	-0.58544	-0.56493	-0.44799	0	0.71293	0.21864	0.57522	0.30727
44	-0.71079	-0.7773	0.1448	-0.87156	-0.82751	-0.79658	1.79195	1.86521	1.67829	1.59883
45	-0.46681	-0.32558	-0.4271	-0.13871	-0.41941	-0.09897	-0.33568	0.84384	-0.2501	-0.46465
46	2.29112	-0.66436	-0.60678	-0.58041	-0.7388	-0.21945	-0.76479	-0.74021	-0.59869	1.88366
47	0.60848	-0.59815	-0.42019	-0.52193	-0.36556	-0.01343	0.18555	0.5815	0.28893	0.26971
48	-0.67918	0.09496	-0.37295	-0.22637	0.16201	-0.12973	0.34299	0.08854	0.14202	-0.07117
49	-0.40963	-0.66359	-0.67454	-0.76659	-0.68333	-0.53171	-0.6628	-0.63278	-0.23742	-0.29924
50	0.80021	-0.28629	0.71203	-0.378	-0.10887	-0.30298	-0.05112	0.09251	0.16832	0.13143
51	0.75937	0.69757	-0.42354	-0.30367	-0.14375	-0.10087	0.01585	0.11495	0.12748	0.07878
52	-0.49874	-0.0517	0.31968	-0.11493	-0.13773	-0.00394	0.09233	-0.07802	-0.1067	0.05563
53	-0.53553	-0.64379	-0.61362	-0.65196	-0.45492	-0.23284	-0.50836	-0.58549	-0.55415	1.35817
54	-0.72235	-0.59674	-0.6822	-0.42373	-0.39811	-0.32321	0.36174	-0.11754	0.56561	0.35885
55	-0.70101	-0.49009	0.17139	-0.48344	-0.11996	-0.03562	-0.1697	-0.03542	0.68782	0.30958
56	-0.82888	-0.84365	-0.74924	-0.83808	-0.70158	-0.05011	0.19674	0.74164	0.92916	0.64034
57	-0.72129	-0.45432	-0.17385	0.24833	-0.07953	0.01065	-0.02272	0.22207	0.1893	0.08449
58	-0.64082	-0.6274	-0.72602	-0.57237	-0.66215	-0.47118	-0.54027	0.85966	-0.32733	1.05648
59	-0.03446	0.2038	-0.2732	-0.32609	-0.26652	-0.46156	-0.20991	-0.1776	0.07686	0.62542
60	1.0055	-0.64415	-0.63081	-0.47634	-0.66155	-0.61005	-0.38118	-0.23247	0.68882	0.61149
61	-0.58297	-0.45281	-0.71426	0.42535	-0.62583	-0.64829	-0.43983	0.75235	0.7477	0.59957
62	-0.80661	-0.65178	-0.52426	-0.3673	-0.07539	0.34693	0.40553	0.37749	0.13674	0.15012
63	-0.63229	-0.38903	-0.37192	0.49447	0.77312	-0.2053	-0.08692	-0.15815	0.07545	0.28431

Table C. 38 Bark codebook for MODE_VQ == 08_06, BLEN_TYPE == MEDIUM

VN	0	1	2	3	4	5	6	7	8	9
0	-0.72322	-0.52244	-0.52869	-0.35493	-0.39994	-0.26864	-0.20383	0.16746	0.23886	0.13425
1	-0.39785	-0.24926	-0.3477	-0.45579	-0.27292	-0.11566	0.01615	1.49022	0.22074	0.14731
2	-0.20229	-0.11587	-0.33388	-0.29265	1.77134	-0.21726	-0.11357	0.02775	0.02875	0.05465
3	-0.59883	-0.31221	2.00008	-0.31472	-0.11046	-0.0114	0.04402	0.07922	0.15303	0.05094
4	-0.67627	2.73782	-0.20914	-0.42004	-0.21839	-0.12959	-0.07099	-0.06445	0.05655	-0.09807
5	-0.65883	-0.62523	-0.59402	-0.51108	-0.32109	0.02101	0.65091	0.40592	0.32988	0.57345
6	-0.68075	1.11721	0.89851	-0.3332	-0.24301	-0.12516	0.01118	0.01076	0.12747	0.0346
7	-0.66873	-0.40379	-0.47607	1.11635	-0.16081	-0.1308	0.0695	0.01768	0.21366	0.15297
8	0.767	0.02566	-0.22983	0.304	-0.11937	-0.12093	-0.05592	0.01075	-0.05764	-0.26434
9	-0.57031	-0.29132	-0.39542	-0.45396	-0.32694	-0.15998	0.05014	0.25197	1.36248	0.29176
10	-0.64334	-0.42489	0.64834	0.48236	-0.01264	0.19135	-0.02348	-0.04026	0.10227	-0.11863
11	2.16056	-0.15227	-0.19348	-0.37369	-0.18671	-0.16445	-0.1783	-0.20249	-0.03653	-0.00657
12	0.5506	-0.28739	0.66576	-0.34901	0.17998	-0.22018	-0.19054	-0.17792	0.04131	0.06794
13	0.91051	0.9188	-0.00782	-0.38589	-0.13742	-0.13997	0.01878	-0.09088	-0.01553	-0.11649
14	-0.61032	0.12839	-0.41065	0.35738	-0.20255	-0.03023	-0.13385	0.13706	0.04929	0.27216
15	-0.41075	-0.37893	-0.49177	-0.51924	-0.33554	-0.18296	-0.17584	0.01554	0.51197	1.69525
16	-0.67661	1.2033	-0.41018	-0.54003	-0.34909	-0.19554	0.02774	0.21741	0.28716	0.20875
17	0.87186	-0.53389	-0.38239	-0.41356	0.02053	-0.01128	0.22752	0.19184	0.18687	0.20907
18	-0.25341	-0.12157	-0.02847	-0.38627	-0.33679	-0.28381	1.53641	-0.08953	0.14503	0.03231
19	-0.60718	0.51738	-0.52569	-0.4269	0.20335	0.26592	0.12384	0.03243	0.08231	0.10328
20	-0.67803	0.612	0.06765	-0.35534	-0.24888	0.44208	0.15601	-0.14518	0.03315	-0.25641
21	-0.71804	1.09237	-0.04291	0.26745	0.27183	-0.11695	0.04434	-0.07191	-0.05579	-0.14774
22	-0.4969	-0.40247	-0.2858	0.01238	-0.00012	0.03048	0.69444	-0.07576	-0.02008	-0.15636
23	-0.54955	-0.52205	-0.55413	-0.38257	0.55164	0.02237	0.08979	0.02328	0.34982	0.32861
24	0.2001	-0.571	-0.4448	-0.21842	-0.01684	0.03861	0.04646	0.22236	0.24613	0.18652
25	0.16702	-0.42861	-0.24725	-0.39682	-0.39317	1.48	-0.12184	-0.25298	-0.03385	0.2757
26	0.11909	0.12784	-0.16232	-0.21229	-0.0846	-0.01851	0.02998	0.02212	-0.00282	0.00336
27	-0.69998	-0.49304	0.69766	-0.46457	-0.28441	0.07549	0.07001	0.30521	0.18733	0.18945
28	-0.6798	-0.59705	-0.28239	0.11222	0.63208	0.10663	-0.0696	0.17348	-0.07298	0.09826
29	-0.65515	-0.55434	-0.41854	-0.37482	-0.13363	1.00542	0.12461	0.2827	0.17099	0.15394
30	-0.50871	-0.30173	0.12435	-0.24643	-0.12462	-0.04717	-0.26253	-0.19365	0.17761	0.52492
31	-0.66456	0.23637	0.2629	-0.3854	-0.34418	-0.21835	0.09797	0.25458	0.10429	0.04442

Table C. 39 Bark codebook for MODE_VQ == 08_06, BLEN_TYPE == SHORT

VN	0	1	2	3	4	5	6	7	8	9
0	-0.57802	-0.57141	-0.42355	-0.46304	-0.36455	0.24035	0.7476	0.68991	0.25601	0.04009
1	0.28845	-0.48584	-0.44753	-0.41993	-0.3321	0.00587	0.36256	0.46959	0.11241	0.12476
2	0.01683	0.01898	-0.08626	-0.00077	0.91816	-0.11728	-0.37554	-0.25956	0.13183	-0.0157
3	-0.3774	-0.24131	0.36966	-0.28485	-0.33066	-0.13748	0.3504	0.21495	0.27418	0.04453
4	-0.5475	-0.51496	-0.5806	-0.53744	-0.36412	-0.20409	-0.03024	0.1476	1.50359	0.88762

5	-0.67616	-0.72231	-0.72156	-0.73695	-0.52591	-0.15811	0.44067	1.14688	0.71501	0.50725
6	-0.23811	-0.08785	-0.21856	-0.152	-0.14474	-0.21451	-0.08413	1.05778	0.12008	-0.01374
7	-0.51311	-0.19062	-0.11628	0.30955	0.14795	-0.16366	0.41165	-0.0257	0.09042	0.11259
8	-0.05406	-0.21182	-0.33704	0.07261	0.03113	-0.14119	-0.1974	-0.22906	0.9512	0.11972
9	0.98439	0.65319	-0.1669	-0.27058	-0.30152	-0.22348	-0.2133	-0.2243	-0.19574	-0.25396
10	-0.32319	-0.00384	0.52639	0.4744	-0.1592	-0.16248	-0.01153	-0.36599	0.041	-0.05117
11	1.02868	-0.46254	-0.42344	-0.43129	-0.34465	-0.21156	-0.01726	0.08034	0.20873	0.31582
12	-0.45135	1.07255	-0.32036	-0.38978	-0.13768	0.106	-0.11095	0.06569	0.2178	-0.0075
13	-0.28556	0.01632	0.09036	-0.27922	-0.07802	0.90473	-0.19791	-0.04628	-0.03	-0.04055
14	-0.161	-0.25208	-0.19251	-0.22019	-0.08461	-0.06844	-0.02646	-0.07638	0.11045	0.90366
15	-0.36747	-0.63712	-0.67329	-0.62261	-0.49666	-0.33233	-0.16496	0.17398	0.55248	2.02301
16	-0.54715	-0.61226	-0.6201	-0.42023	0.0444	0.47545	0.07471	0.43297	0.29773	0.71023
17	-0.01456	0.01783	-0.05734	-0.39819	-0.31796	-0.169	1.01819	-0.20263	0.01185	-0.04581
18	-0.11511	-0.53903	-0.55302	-0.37049	-0.04789	0.73109	0.62771	-0.1414	-0.01667	0.19648
19	-0.31582	0.31106	1.04836	-0.33101	-0.17131	-0.12991	-0.11347	-0.02992	-0.09755	-0.1756
20	0.08195	0.75518	0.39884	-0.22119	-0.01732	-0.12258	-0.2289	-0.23981	-0.18117	-0.2898
21	-0.42906	0.70563	-0.38837	0.31819	0.05282	-0.07603	-0.09861	-0.00718	0.12227	-0.13255
22	-0.35793	0.38757	0.07555	-0.14113	-0.04218	0.00831	0.27244	0.31388	-0.28908	-0.28666
23	-0.04261	-0.17861	-0.15118	0.85638	-0.01961	0.04669	-0.25858	-0.04251	-0.10475	-0.11468
24	2.44129	-0.22773	-0.36897	-0.3908	-0.38351	-0.35267	-0.25991	-0.34349	-0.25698	-0.2275
25	0.28891	1.92271	-0.11223	-0.39286	-0.39498	-0.34129	-0.28163	-0.29956	-0.24706	-0.35407
26	0.00485	0.00679	-0.05727	-0.02688	0.04948	0.02555	0.06141	-0.0375	-0.01243	-0.01405
27	0.71773	-0.11975	0.00411	-0.01243	0.03191	-0.02596	-0.12844	-0.13814	-0.20069	-0.28737
28	-0.32601	0.67116	0.14848	-0.38446	-0.32286	-0.32995	0.02701	0.15641	0.35178	-0.00229
29	0.17895	-0.05065	-0.06667	-0.16227	-0.15379	-0.14349	-0.01903	0.06206	0.07415	0.18594
30	-0.64963	-0.65367	-0.30217	-0.02	0.62708	0.73268	0.22757	-0.25185	0.05146	-0.07901
31	0.00969	0.34064	-0.20402	-0.27327	-0.15497	-0.34782	-0.36956	-0.27702	0.36842	0.7233

Table C. 40 Bark codebook for MODE_VQ == SCL_1 / SCL_1_960, BLEN_TYPE == LONG

VN	0	1	2	3
0	15.97279	-0.32776	-0.4364	-0.35022
1	-0.33181	-0.39169	-0.40789	-0.37624
2	-0.31781	-0.25459	-0.16784	-0.30259
3	1.84493	-0.26835	-0.26142	0.15021
4	0.83697	-0.10218	-0.1666	-0.2539
5	0.02246	1.04304	-0.10738	0.05972
6	0.34463	-0.41719	-0.38229	3.57916
7	-0.27803	-0.25496	-0.3183	0.46518
8	-0.38289	-0.42048	-0.43488	-0.38168
9	-0.43025	-0.29074	1.09387	0.65533
10	-0.34937	1.11869	-0.32	-0.24793
11	-0.27591	-0.18472	0.87442	0.22148
12	0.56275	-0.22857	-0.21303	-0.20072
13	0.78892	-0.01353	-0.06869	0.05319
14	-0.39413	2.83598	-0.37421	-0.28316
15	-0.23743	-0.08994	-0.02388	0.17371
16	2.93656	-0.45117	-0.37346	-0.17016
17	-0.0022	-0.10266	0.00719	0.10319
18	6.03639	-0.39238	-0.34872	0.15975
19	-0.2118	-0.37262	-0.19912	-0.17291
20	-0.35911	1.51391	-0.2782	-0.23148
21	-0.19447	0.65257	0.04907	0.12937
22	1.8438	-0.22398	0.00658	-0.15067
23	-0.26703	-0.2801	-0.27831	-0.40732
24	-0.28692	-0.39461	-0.11711	-0.49627
25	3.49577	0.7085	-0.11709	0.323
26	-0.29669	-0.31341	2.08688	-0.28253
27	-0.2327	-0.11923	0.06734	-0.02423
28	-0.37443	-0.49589	4.81885	-0.5731
29	-0.37998	-0.27748	1.65689	-0.32076
30	-0.20133	-0.24443	1.30387	-0.32903
31	-0.30134	-0.16581	1.02766	0.19394
32	-0.25962	-0.24615	-0.37925	1.49641
33	1.36649	-0.30531	-0.19045	0.14048
34	-0.03229	-0.29641	-0.19416	-0.11819
35	-0.2314	-0.21867	-0.24705	1.59059
36	-0.29774	-0.12934	-0.13179	-0.29359
37	0.19577	0.51416	0.58029	0.1218
38	-0.36195	-0.36501	-0.32174	-0.32606
39	-0.09093	2.90911	-0.2835	0.2785
40	-0.20483	-0.22821	-0.29691	-0.15994
41	-0.27341	1.58812	1.24856	0.35948
42	0.03447	-0.39027	-0.33077	-0.20393
43	-0.29772	0.06856	2.70277	0.31162
44	0.48511	-0.39078	3.45247	-0.18981
45	-0.50634	-0.2721	-0.06816	0.89272
46	-0.30209	-0.06799	-0.29361	1.92904
47	-0.12171	-0.05918	0.57976	0.16952
48	-0.41903	-0.28341	0.29189	-0.27254
49	-0.30491	-0.14076	0.26644	0.11869
50	1.45685	-0.38068	-0.35485	-0.15544
51	0.1312	0.05522	0.26416	0.14805
52	0.71867	0.74498	-0.29525	-0.17204
53	-0.0688	-0.02908	0.02271	0.07263
54	1.65541	-0.3218	-0.18897	1.4702
55	-0.17208	-0.19179	-0.13102	0.14706
56	0.52083	-0.33951	-0.35165	-0.29863
57	-0.30294	0.32001	0.26815	0.13233
58	0.37049	-0.14782	-0.12007	-0.05449
59	0.06111	-0.2167	0.44791	0.00922
60	0.68447	-0.24481	0.73585	-0.11946
61	-0.1763	-0.0708	0.03114	0.06267
62	0.7997	-0.24214	-0.27998	-0.00908
63	-0.20398	-0.20574	-0.24416	0.25111

64	-0.1334	-0.22675	-0.30144	1.50904
65	-0.22594	-0.23179	-0.3046	-0.32451
66	-0.23552	-0.14151	-0.17528	-0.36286
67	1.14974	1.01421	-0.20947	0.1487
68	-0.28145	-0.15189	-0.29247	0.7695
69	-0.2679	1.06711	-0.00875	0.034
70	-0.36698	-0.28874	-0.34733	1.22156
71	-0.24687	-0.13018	-0.22216	0.4599
72	-0.53304	-0.50032	-0.25632	0.19348
73	0.19462	0.27696	0.18618	0.21315
74	-0.30692	-0.24446	-0.25346	0.60127
75	-0.20201	-0.15062	0.66315	0.08568
76	-0.29238	-0.24736	1.17585	-0.39713
77	0.65234	0.56276	0.01684	0.17701
78	-0.33018	-0.15793	0.87343	0.80012
79	-0.25691	-0.17128	0.09531	0.10283
80	0.35261	-0.2624	-0.22732	0.43887
81	0.02255	0.169	0.04252	0.05178
82	-0.42457	-0.32169	-0.35461	-0.18956
83	0.91711	-0.1966	0.58452	0.19995
84	-0.42911	0.25652	-0.20261	0.22473
85	-0.15614	0.16506	0.00676	0.05013
86	-0.22949	-0.32918	-0.21892	0.35888
87	-0.31265	-0.22287	-0.1315	0.06572
88	-0.3942	-0.42844	-0.41406	-0.30458
89	-0.16078	0.09564	0.13988	0.11743
90	-0.36494	-0.18409	0.32497	-0.02761
91	0.28924	-0.107	0.03887	0.03924
92	-0.24668	-0.27135	0.3662	-0.0839
93	-0.20678	0.43261	-0.11528	-0.03285
94	-0.36566	-0.20387	-0.32418	0.51753
95	0.44663	-0.18779	-0.12915	0.06873
96	-0.38369	-0.28077	-0.29431	-0.27078
97	-0.35421	-0.28922	-0.34196	-0.35336
98	-0.37896	-0.3483	-0.31407	-0.33842
99	0.48573	-0.12207	-0.22058	0.3604
100	-0.39092	-0.2499	-0.37095	-0.37151
101	-0.35177	0.99926	-0.18107	0.2259
102	-0.29084	1.20672	-0.22076	1.01738
103	-0.24265	-0.14349	-0.06127	0.14304
104	-0.43439	-0.37912	-0.3701	-0.37472
105	-0.4216	-0.3916	-0.31015	0.52597
106	-0.2893	-0.16274	-0.31592	-0.32973
107	-0.30848	-0.2879	0.80085	-0.23039
108	-0.29886	-0.31763	0.85991	-0.35811
109	-0.31918	-0.30435	-0.24334	0.21866
110	-0.37915	0.83504	-0.3043	-0.27288
111	-0.29253	-0.30857	-0.25001	0.15133
112	0.54299	-0.2973	-0.09463	-0.28935
113	-0.27628	-0.29538	-0.30319	-0.34282
114	-0.328	0.08835	-0.24318	-0.26408
115	0.47216	-0.13328	-0.15595	-0.21843
116	-0.37598	0.46964	-0.32806	-0.3567
117	-0.13534	0.222	0.14179	0.11079
118	0.00342	0.37881	0.04373	0.24485
119	-0.1266	-0.20878	-0.10992	-0.0838
120	0.11804	-0.13337	-0.34088	-0.37302
121	-0.29132	-0.32585	0.06994	0.14641
122	-0.30884	0.75518	0.74108	-0.21229
123	-0.19054	-0.08338	0.0554	0.05856
124	0.11366	0.01398	-0.07234	-0.23844
125	0.02724	0.15487	-0.12283	0.04624
126	-0.2962	1.02816	-0.25922	-0.17448
127	0.86822	-0.01822	0.0674	0.12285

Table C. 41 Bark codebook for MODE_VQ == SCL_2 / SCL_2_960, BLEN_TYPE == LONG

VN	0	1	2	3
0	7.82026	-0.09855	-0.27831	-0.28969
1	-0.29082	-0.33979	0.09328	-0.34642
2	-0.34936	-0.30753	-0.39481	-0.33076
3	1.57377	-0.37606	1.15562	0.09083
4	0.75344	-0.28297	-0.1024	0.47231
5	0.337	-0.05784	0.13409	0.04353
6	0.42652	0.09657	-0.12461	0.66641
7	-0.18211	-0.03114	-0.12152	0.01977
8	0.5209	-0.07041	-0.01017	-0.08313
9	-0.14936	-0.15698	-0.22743	-0.24789
10	-0.15303	0.82596	-0.24694	-0.27674
11	-0.20136	-0.20255	0.67185	0.21457
12	0.77381	0.79063	-0.02142	-0.11591
13	-0.22495	-0.23786	-0.29064	-0.30825
14	-0.19017	2.99777	-0.24765	-0.25454
15	-0.16715	-0.20534	-0.2091	0.01599
16	2.8186	0.05788	0.03022	-0.1667
17	-0.06623	-0.26103	-0.22494	-0.18336
18	1.35073	-0.33869	-0.20381	1.1546
19	-0.25015	-0.06645	-0.0627	0.24812
20	-0.31306	0.16528	0.28845	3.19581
21	-0.2221	-0.37958	-0.38785	-0.37917
22	1.10927	-0.10405	1.08575	-0.088
23	-0.02776	-0.0181	-0.09571	-0.00637
24	0.17235	-0.23938	-0.1826	0.19418
25	-0.0998	-0.08567	0.21472	-0.0487
26	0.3184	0.34058	-0.08586	-0.02159

27	-0.01504	-0.02433	-0.00988	-0.05773
28	-0.32521	-0.34004	2.09279	-0.19419
29	-0.31701	1.83272	1.00672	-0.43539
30	1.68842	-0.35189	-0.33057	-0.32088
31	-0.30653	-0.24073	1.25024	-0.15309
32	-0.26287	-0.20919	-0.2744	1.24218
33	-0.15039	1.03773	-0.26753	0.0625
34	-0.15263	-0.08687	-0.21116	-0.11222
35	0.61479	0.51076	0.00571	-0.02303
36	-0.11126	-0.03827	-0.12143	-0.16476
37	-0.16757	3.23845	0.08869	-0.21005
38	-0.12304	-0.1505	-0.14543	-0.03863
39	1.45061	1.3946	-0.31369	-0.04372
40	0.78281	-0.30524	-0.24318	-0.2618
41	-0.23665	-0.18436	0.17046	0.25858
42	1.07965	1.07502	0.01285	-0.25209
43	0.03419	0.32923	-0.043	0.27464
44	0.02374	1.30771	1.306	0.08107
45	-0.33626	0.04064	-0.34456	-0.10609
46	-0.12084	-0.09769	-0.05358	-0.09417
47	-0.19029	-0.18141	2.24686	-0.13668
48	-0.22694	-0.24403	-0.22728	-0.2328
49	1.23427	-0.32218	-0.28562	-0.26944
50	-0.36348	-0.34231	-0.34988	-0.30848
51	-0.31595	-0.21227	0.10173	0.26788
52	-0.0961	0.22997	-0.28009	0.04729
53	2.78891	-0.44767	-0.33163	0.30129
54	1.47172	-0.13064	-0.16753	-0.15435
55	-0.1209	-0.17731	-0.1787	-0.14546
56	-0.21383	-0.21107	-0.22793	-0.22558
57	-0.19056	-0.34752	-0.4738	1.42957
58	2.01648	-0.27406	-0.20693	-0.32107
59	-0.20953	1.18961	-0.35863	0.48818
60	-0.28547	-0.38528	-0.35467	-0.35872
61	-0.32152	-0.40827	-0.42783	-0.34712
62	-0.18644	1.50551	-0.00128	-0.04026
63	-0.2413	1.25067	-0.25489	0.37218
64	-0.25177	-0.12074	-0.18312	-0.16635
65	0.1208	-0.02133	0.27754	-0.04442
66	-0.25254	-0.22376	0.8664	-0.09045
67	-0.13986	0.74541	-0.02497	0.01115
68	-0.21975	-0.11241	1.10874	-0.01
69	1.0363	-0.2029	0.3822	0.18308
70	-0.16775	-0.26374	-0.28558	1.61926
71	-0.08932	-0.29707	0.39726	0.5884
72	-0.29517	-0.36174	-0.28405	-0.29632
73	-0.19592	0.38215	-0.1508	-0.07825
74	-0.21794	-0.31383	-0.2618	-0.23362
75	-0.17631	-0.30377	0.85805	-0.16009
76	-0.18097	-0.27186	2.06863	-0.24964
77	-0.19476	-0.2036	-0.28525	-0.15282
78	-0.37642	1.60511	-0.16277	1.64982
79	0.14668	-0.18943	-0.23796	-0.13438
80	-0.27729	-0.27796	-0.1872	0.78207
81	0.69748	-0.10812	0.04203	0.06043
82	-0.14911	-0.22484	-0.17434	0.25086
83	-0.05761	0.36149	-0.03948	0.11238
84	-0.32944	-0.29932	-0.23676	1.03797
85	-0.28074	-0.16227	-0.30855	-0.16019
86	-0.21215	0.4894	0.51926	-0.0824
87	-0.0733	-0.06125	-0.041	-0.00725
88	-0.17463	-0.12522	1.17981	1.09364
89	-0.19506	-0.10407	-0.10251	0.03096
90	-0.37205	-0.20249	0.18284	0.10874
91	-0.09186	-0.01869	0.10371	0.16176
92	-0.19995	-0.24593	0.38678	-0.09003
93	0.3775	-0.03193	-0.05753	-0.01756
94	-0.22705	0.38846	-0.16116	-0.08961
95	0.43322	-0.15988	-0.01832	0.05074
96	0.13357	-0.15862	-0.18846	-0.17688
97	-0.08902	-0.10017	-0.10239	0.18298
98	-0.28637	-0.30485	-0.31164	-0.27465
99	-0.17921	1.39292	-0.26067	0.3451
100	-0.34898	-0.36809	-0.34053	-0.27584
101	0.43658	-0.14374	-0.10547	0.20012
102	-0.21058	0.26284	-0.17039	-0.18048
103	-0.21576	-0.19886	-0.1304	-0.12544
104	-0.31967	-0.31609	-0.24746	0.21252
105	-0.06734	-0.19585	-0.16473	-0.02529
106	-0.0875	-0.3007	-0.20391	-0.22529
107	-0.18219	-0.14403	0.91016	0.94564
108	-0.22991	-0.31329	0.68057	-0.24077
109	-0.2288	-0.18396	-0.11372	0.02835
110	-0.21804	1.27174	-0.23625	-0.11318
111	-0.14616	-0.1324	-0.16081	-0.03736
112	-0.01402	-0.02661	-0.00695	-0.03338
113	0.05809	0.01513	-0.10805	-0.11373
114	0.56867	-0.22329	-0.13328	-0.12751
115	-0.08958	0.47008	0.03598	-0.01203
116	-0.23463	0.29486	-0.11569	0.35258
117	-0.06227	-0.02218	0.06439	-0.01557
118	0.1237	-0.06893	0.23141	-0.14079
119	-0.11436	0.01965	0.18422	0.00669
120	0.10492	-0.06786	0.30257	0.24144
121	-0.07127	-0.08448	-0.10433	-0.09589
122	-0.19604	-0.27188	-0.21543	-0.22056
123	-0.22289	0.90521	0.82469	0.02195
124	0.5434	-0.22816	-0.14396	-0.20118
125	0.57274	-0.12677	-0.0409	0.04178

126	-0.24622	0.38134	-0.14405	-0.1783
127	-0.13557	0.38515	-0.03826	0.09435

Table C. 42 Pitch codebook 0 for MODE_VQ == 16_16

VN	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.19328	0.63076	-2.39515	-0.19719	-0.15419	-3.50788	-0.13934	-0.35477	-0.64298	0.25741	0.00492	-0.92114	0.4776	-0.03272	0.19609
1	-0.31847	-0.24935	-1.72029	0.05609	-0.26187	0.10948	3.90635	-0.18041	-1.05426	-0.05995	-1.56363	-0.09787	-0.98082	-0.16781	-0.37276
2	0.27956	0.76702	-0.4263	0.01604	4.4533	-0.11603	0.0477	-0.51653	0.27022	0.1621	0.24092	0.28479	0.03884	0.18099	-0.24144
3	-4.30919	-0.59045	-0.37124	-0.27298	0.12449	0.37756	-0.617	0.37382	0.00229	0.03938	-0.12039	1.20714	0.70307	-0.14603	-0.23747
4	0.09143	0.45102	0.32431	-0.1234	-3.45192	-0.93693	0.431	-0.16866	-2.34798	0.73866	-0.11262	0.26579	0.76072	0.51277	0.32433
5	-1.74736	1.43894	0.08357	-0.08003	0.04447	-0.06374	0.03423	-0.10651	0.2973	-0.13968	-0.00732	0.09057	0.19224	0.06526	0.18518
6	0.07132	-0.22083	0.15273	-0.08839	-0.20716	-0.42641	0.04459	-0.13139	-5.54646	-0.18048	-0.02909	-0.34685	-0.21681	0.19078	-0.19677
7	3.59032	-5.35703	0.6338	-0.30303	-0.37248	-0.90726	-0.03432	-0.65963	0.06877	-2.59492	0.78681	0.18609	-0.57918	-0.12459	0.01355
8	0.18053	0.807	0.26539	-0.2154	-0.18244	-3.04928	-0.08112	0.26139	-0.66761	0.06734	0.00568	5.0787	-0.06501	0.09493	-0.70829
9	0.21261	1.74027	-2.02451	-0.07915	1.17089	-0.43837	-1.93802	-0.01762	1.37248	0.09083	-1.26393	0.1312	0.64181	0.21197	-0.03875
10	0.08578	-2.82061	0.67063	0.55369	-0.1269	3.4135	-0.0959	0.04107	-0.23385	-0.35902	-0.23194	1.34858	0.41662	-0.2663	0.00029
11	0.45299	0.22878	-0.6949	4.92403	0.42772	-4.30945	-0.13728	-0.69494	-0.03521	-0.98567	0.15119	1.10344	-0.42978	-0.02187	-0.36498
12	-0.30521	-0.60511	0.19447	6.91896	-0.01653	0.7995	-0.1433	0.08465	0.73208	-0.66829	-0.14943	2.76103	0.4934	-0.14122	0.44281
13	0.06775	-0.04736	-1.8209	-0.04024	-0.06274	-0.21391	0.00937	0.06263	0.57544	0.10642	-0.02	0.32092	-0.19464	-0.02587	0.0926
14	-0.12508	-1.75321	-2.28109	0.40829	1.25846	-0.10706	0.76401	0.07275	-2.18765	-0.01034	0.62369	-0.07537	3.98226	0.19619	0.33638
15	0.0386	1.03836	0.84558	-0.42575	0.02304	4.2599	0.03884	-0.01723	-0.54087	0.43271	-0.01313	-0.20576	-0.01852	0.05708	-0.6663
16	0.03528	2.11504	-0.36406	0.23267	1.41398	0.38402	0.076	-0.00624	-2.2675	0.22027	-0.06605	-0.12372	1.00826	0.02231	0.44988
17	0.12062	1.05116	2.37438	2.76367	-0.15861	0.51273	-0.31601	-0.15593	0.36905	-0.03214	-0.31562	0.39523	0.57059	0.61633	0.03489
18	0.60112	-0.13	-4.05609	0.2048	-0.30503	-0.05617	-0.94045	0.16984	1.32945	-0.01534	0.35859	-0.33224	-0.17627	-0.21753	0.03853
19	-1.41211	0.52443	1.02003	-0.00205	0.11986	-5.55761	-0.18906	-0.0822	0.55028	0.13583	-0.08802	-0.33587	-0.42665	0.24249	-0.42536
20	0.05403	-0.03349	-5.44747	0.1699	-0.12774	-1.75164	0.05994	0.02586	-0.73632	0.16941	-0.04575	-1.62237	-0.0996	-0.12004	0.10628
21	-0.16586	-0.0894	-0.47818	0.08874	0.16376	-0.13952	-0.14467	0.02788	0.07048	-0.0499	-0.03696	0.02439	-0.05629	0.12623	-5.09264
22	0.31397	0.41717	-0.85895	0.17025	0.04281	1.95639	-0.05249	-0.03588	-0.23186	0.11308	0.07846	-1.2473	0.14392	0.05166	-0.33494
23	0.0669	-0.1333	0.09693	2.68124	0.02215	0.66469	0.05526	-0.14608	0.53935	-4.00034	-0.10386	0.07001	-0.18661	-0.96232	-0.10579
24	0.25851	-0.48122	2.02057	-0.40686	-0.18644	0.69195	-1.52869	0.05482	-4.5997	0.12978	0.35459	0.09825	-0.06425	0.18335	0.22679
25	-0.22356	0.98268	0.43954	1.74014	-0.48477	-1.07498	0.03724	-0.71923	-0.23774	0.14055	0.26115	1.27014	-0.28403	0.01158	0.10996
26	0.04285	0.04234	0.20252	-0.15636	-0.07807	0.13183	5.80269	0.18192	0.63272	-0.09649	0.46587	0.16054	0.58397	0.26981	-0.00709
27	1.95347	-0.65443	2.34786	1.69285	1.39367	-1.31105	0.20222	0.33427	-1.23452	-0.92986	1.31054	0.84093	0.16699	-0.68894	-0.26675
28	-0.0514	-0.08536	-2.03244	0.00834	-0.2137	1.62213	-0.02068	0.04662	-4.95244	-0.2553	0.14937	-0.07133	-0.04742	-0.10693	-0.47335
29	0.28644	0.88753	-1.30252	-1.50342	-0.03434	0.36534	-4.23076	-0.08707	-0.28396	0.27714	-0.09119	0.38605	0.18491	0.05477	-0.49448
30	-2.06825	-0.48719	0.14409	0.20415	-2.23871	-0.13658	-0.05634	0.03737	0.12536	0.14239	-0.11636	0.17617	-0.10478	-0.17528	0.02358
31	4.00995	0.01231	-2.1865	0.36164	0.66551	-1.56522	0.27211	0.14022	-0.01986	-0.57895	0.08864	0.99164	0.10797	-0.05366	0.47407
32	-0.13031	-0.14444	-0.25549	0.18371	-0.93642	-2.18998	0.93958	0.17447	-1.49612	0.80371	2.72302	-0.35802	0.9164	0.29042	0.19062
33	-0.02838	-0.19005	0.02391	0.14143	-0.06521	-0.0854	-0.03045	-0.10144	0.16138	-0.00987	0.23123	-0.06121	-0.26553	-6.69689	-0.07929
34	0.00149	-1.0028	0.6455	-1.69246	2.24431	-0.29611	-0.06586	-0.4672	0.49285	0.19914	-0.23461	1.15665	0.41475	0.04127	-0.12255
35	-0.06601	0.23458	-5.94945	-0.05206	0.00086	0.60685	0.00758	0.132	0.16271	-0.23176	0.05519	0.34078	-0.1687	0.18863	-0.4262
36	0.22438	-0.04567	3.87807	0.1331	-0.10651	-0.23441	0.04706	0.03362	2.87689	-0.02631	-0.02368	0.33493	0.00725	-0.04345	0.5568
37	0.67316	-0.31263	-0.28589	2.54744	-0.0594	-0.01623	-0.49113	-0.32604	0.09962	1.68422	-0.07536	-0.23092	0.08426	-0.47939	0.10046
38	-0.16242	-4.7413	0.37165	-1.71527	0.20405	0.35533	0.05634	0.42186	0.27793	-0.76027	-0.13528	0.50684	0.20522	0.29206	-0.09133
39	4.5537	-0.96357	1.07333	-0.44063	0.36868	-0.04645	-0.40786	0.4031	-0.38383	0.23053	-0.00624	0.05388	0.0171	0.00216	0.44643
40	0.38952	-0.14522	-1.96549	0.17116	0.06073	-0.94762	-0.26181	-0.04173	-2.97264	-0.19284	-0.03978	2.30709	0.17567	-0.05922	1.67079
41	0.35292	-0.40741	0.38872	0.05138	-0.26975	0.36995	-0.01203	0.79502	0.14043	-2.76156	-0.12178	0.85416	-1.0529	0.4222	-0.05645
42	0.095	-0.32167	0.21124	0.19431	-0.06022	-0.28655	0.03077	-0.09998	-0.1371	-0.07503	-0.01332	7.23531	-0.25864	0.03741	-0.55107
43	0.83664	0.87569	-0.88981	-4.92305	0.05608	0.48593	0.22875	1.11318	0.32545	-0.34766	-0.46921	0.37748	0.26432	0.43936	0.09744
44	0.06881	3.37157	-0.40964	0.20861	-0.10359	2.39914	-0.08913	-0.80077	0.18676	-1.12896	-0.08623	1.47259	0.4514	0.05261	-0.1974
45	0.12191	0.16781	-2.8078	0.22123	-1.76801	-0.26266	0.14265	0.46197	-0.78793	1.40609	2.44921	0.53457	-0.13333	-0.2854	-0.05226
46	0.03637	0.23022	0.01993	-0.00064	0.13282	0.04369	-0.21685	-0.04703	-0.57193	0.00269	0.0059	-0.20988	-4.24139	-0.06579	0.16622
47	3.28075	-0.53508	-0.17384	0.13363	0.60907	-0.29082	1.62332	-0.14206	0.98081	0.33115	0.44549	0.21244	1.6038	-0.32424	0.55713
48	0.07678	0.38895	0.7952	-0.03533	-0.17462	6.36191	0.01957	0.3863	0.95455	-0.03012	-0.0842	2.42881	-0.4858	0.25918	0.2231
49	0.36762	-0.54484	0.44636	-0.06369	-0.12712	0.07897	0.1581	-0.01787	-0.22679	3.6498	-0.11451	0.24965	-0.01277	-0.12133	0.50191
50	0.24249	0.52133	-0.04211	-0.89585	-0.0837	1.52254	-0.0739	-4.94675	-0.18551	0.54686	0.0491	-0.743	-0.25099	0.03519	0.67337
51	-0.12288	-0.88372	-0.53566	4.07632	0.06154	0.6609	0.25066	-0.10944	0.94476	-0.47834	0.18238	-0.64789	0.09654	0.36583	0.13283
52	-0.93728	0.72253	-2.75527	2.91379	-0.36904	1.76762	-0.14261	0.58892	0.02342	0.5911	-0.39395	1.91781	0.00794	0.19281	-0.93547
53	0.14444	-0.63784	0.75174	-0.15818	-0.95164	0.23605	-0.24734	-1.68917	0.63454	-0.78179	1.64121	1.70586	0.45394	-0.57057	1.18119
54	-0.52553	0.6249	-0.28481	0.03199	-0.11834	1.52627	0.03641	0.04835	-0.13736	0.23763	0.07853	4.11254	-0.06401	0.05985	0.11504
55	-0.30823	0.67996	-0.35883	6.11738	0.02847	-0.22139	0.26616	0.09156	0.39339	1.65313	0.19829	0.13234	0.35709	0.7255	-0.03478
56	-0.11298	4.25384	-0.62785	-0.0618	1.97871	-0.41236	-0.2918	1.4505	0.29744	-0.67929	0.12657	0.04748	-0.574	0.76238	-0.13558
57	-0.01134	0.09589	-0.054	-1.50614	-0.05511	1.56107	-0.02547	3.58955	-0.22347	0.00546	0.1426	2.13878	-0.25526	0.27519	-0.04553
58	0.11779	3.20061	1.93368	0.0073	1.24764	-0.18523	-0.03805	-0.34109	-0.79529	-0.60709	0.68029	0.48886	0.1175	-0.33375	0.10542
59	1.66634	-0.15916	-0.45437	-5.18042	0.5886	-1.10096	0.11613	-0.05648	-1.7815	0.12044	0.02567	-3.78553	-0.20664	0.38618	0.03101
60	0.16354	0.02263	2.06556	0.19618	-3.75731	0.02402	0.11904	0.02399	5.64481	-0.44151	-0.328	0.01573	3.33388	0.18641	0.16414
61	0.1044	0.03027	-1.31243	0.01726	-0.80363	0.15638	-0.31454	-0.19575	-0.23883	0.0567	-4.43339	0.18846	0.40781	-0.12261	0.57492
62	-0.09966	3.55004	0.57085	-0.60032	-1.47415	-0.21754	0.31958	1.28335	-0.31255	-0.03081	-1.38184	0.11194	0.64373	0.50807	0.26244
63	-0.19683	-1.24127	1.64675	-0.1435	0.38836	0.10596	-2.7921	0.19931	0.11438	0.23029	-0.28036	-0.27051	0.11563	0.0703	-0.22076

Table C. 43 Pitch codebook 1 for MODE_VQ == 16_16

VN	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0.07515	0.11319	2.87417	-0.11457	0.32492	0.02079	0.79764	0.09507	0.45619	0.12386	-3.9498	-0.07387	0.07765	-0.07803	-0.13562
1	-0.10019	0.15728	-0.37445	0.1511	-0.63454	0.17363	-0.09245	-0.02848	0.418	0.18159	0.23766	-0.1423	-6.03055	0.20838	0.08756
2	-0.1278	3.20844	-0.15284	0.54794	-0.12477	-1.64954	0.08371	-1.21332	0.55174	-0.64343	-0.17452	0.03015	-0.21817	-0.24898	0.11361
3	-0.56784	0.42346	1.65367	0.64357	-2.07095	-0.84671	0.24436	0.13442	1.69729	0.46445	-0.45041	-0.01128	0.26029	0.3005	0.43173
4	-0.08298	5.30368	0.29362	-0.51596	0.24003	-0.77085	0.02521	0.22085	0.47681	-0.43049	0.23838	-0.57123	-0.05014	-0.12939	-0.06497
5	-0.04266	1.05639	2.59774	-0.06493	1.79827	-0.06381	-2.02454	-0.3792	0.90795	0.27397	0.83212	-0.41501	0.14266	0.20286	-0.19623
6	0.03178	-0.37679	0.02746	0.37625	6.00655	-0.0193	0.02079	-0.15695	-0.77699	0.60726	0.34067	0.22152	0.26194	0.29529	0.29967

7	0.01079	-0.36875	-1.65656	1.17907	-0.54736	-1.08506	-2.10622	0.42401	0.26086	0.58962	-1.29313	0.42841	-1.79375	0.19377	-0.43614
8	-0.0188	0.19015	-3.27296	0.08467	-0.40018	0.555	-0.209	-0.03955	2.53356	-0.11905	-0.33724	0.02738	0.66159	-0.12144	-1.7749
9	0.09532	-0.085	1.14151	-0.25035	0.91641	0.05583	-1.4177	0.03103	-0.65827	-0.0263	-3.31028	0.10631	-0.10641	-0.4025	-0.2503
10	1.26912	0.8839	-0.15738	0.05943	-0.09039	-0.02943	-0.01789	0.02791	0.18606	0.04413	-0.09735	-0.01198	-0.01083	0.03222	0.04394
11	0.31117	6.10706	0.31941	0.50109	-0.55423	0.80737	-0.02004	0.55129	-0.4738	0.16517	-0.02274	-0.8364	-0.31866	-0.07144	0.03331
12	-0.15605	3.20883	0.16155	0.01786	-3.30379	0.58514	0.08836	0.50725	0.71288	-0.40361	0.77817	-0.94546	-0.74525	-0.22772	0.38853
13	-0.05929	0.31212	-0.83849	-0.24632	0.30512	-0.04628	0.25872	0.10743	0.14679	0.21735	-4.20358	0.0388	0.32932	0.45485	0.73028
14	0.21708	-2.2151	0.21992	-0.55131	0.16488	1.41702	-0.00667	-4.40726	-0.39135	-0.03378	0.05478	-0.90141	0.16035	0.33298	-0.06462
15	-0.63073	-1.95098	-1.0992	0.34565	0.36513	3.43031	-1.38613	-1.64222	0.28411	-0.53907	0.2078	-1.7329	1.05701	-0.93012	0.36393
16	-0.09145	2.45255	-0.969	0.95829	-0.48819	2.49449	-0.12214	4.31199	-0.45828	0.41193	-0.03244	-1.2024	0.23649	0.61671	0.05257
17	-0.26269	0.43557	-2.58971	0.0036	0.29162	-0.24509	-0.55772	4.28692	-0.29634	-0.40119	0.34324	-0.47128	-1.49751	0.44551	0.2794
18	0.21037	-0.66513	-0.55396	-0.34471	0.01329	2.78551	0.08309	-0.49472	1.27566	-0.13718	-0.10344	4.80049	0.07622	0.17698	-1.04192
19	-0.16335	-0.2567	-2.92788	0.056	1.85222	0.17486	-1.93346	0.1266	-0.08244	0.08778	-1.96446	0.09765	2.0252	0.091	-0.59395
20	0.05138	-0.95443	-1.06426	1.25172	-4.63197	0.68934	0.07875	-2.77991	0.72607	0.16599	0.67421	-0.59917	0.08411	-0.41338	-0.76748
21	2.08614	0.36709	0.23863	-2.14486	-0.06667	-1.44115	0.07655	4.61807	0.12558	-0.12122	0.64217	-1.06553	0.49312	0.73706	0.61071
22	0.08484	3.74803	-0.14061	-2.75632	-0.14545	0.84032	-0.02391	-2.45464	-0.12663	-0.77803	-0.10146	0.34523	0.30698	-0.24459	-0.27874
23	-3.61401	-0.88497	-1.10788	-0.96734	-0.0319	-0.70807	0.09569	-3.18688	0.06336	-0.10726	0.17976	-0.62226	0.28424	-0.1225	0.1696
24	0.52069	-0.50577	0.3254	0.24099	-2.71418	0.21832	-3.20538	0.35748	-0.46042	-0.1358	0.24659	-0.09054	-0.90614	-0.23293	-0.1655
25	0.03343	-0.10787	0.43131	0.08671	-0.12629	-0.14816	0.17029	3.48604	-0.13138	0.0035	0.01601	-0.23413	-0.19355	-0.58049	-0.00837
26	-0.20129	-1.05338	0.40613	-0.17898	-0.04211	0.26854	0.07261	7.07607	0.15114	0.32967	-0.04265	0.25076	-0.10388	-0.13242	0.03479
27	-0.32993	-1.75243	-0.30902	-0.07839	3.05973	-0.15863	0.10678	1.31491	0.53562	0.75999	0.47277	0.02018	-0.77205	-0.22869	0.68702
28	0.13107	0.33473	3.69887	-0.0638	-0.11212	-4.07779	-0.15491	-0.04156	0.10178	-0.01428	-0.0018	-0.59913	-0.07067	-0.11297	-0.85461
29	0.15899	0.18146	-0.59241	0.72869	0.26592	-0.15948	-0.1276	-0.26342	-0.33389	-3.7573	0.10063	-0.26468	-0.03279	-0.11354	-0.20031
30	-0.23375	-3.26177	0.43024	0.75713	-0.03022	-0.98605	0.048	-2.19311	0.01969	0.39346	-0.06943	-0.82796	-0.05645	0.18579	0.30661
31	0.64425	-0.83823	-0.6747	0.34102	1.23711	-0.03891	0.34725	-0.10214	0.10903	4.00269	0.04143	0.09944	-0.14743	-1.00558	-0.03673
32	-0.4281	3.36244	0.96544	0.22008	0.57708	3.85838	-0.01716	-3.10788	-0.41782	1.00571	0.30738	-1.87352	-0.31206	0.11859	0.05191
33	-0.29492	-0.59046	2.83516	0.05005	-0.15607	0.66301	-0.18879	0.09857	-0.19937	-0.01183	0.01713	4.09231	-0.14434	-0.0775	-0.33893
34	0.22129	-0.04304	-4.68472	0.02141	0.00743	-0.3306	-0.10465	0.16798	3.33919	0.00259	-0.0642	-0.29747	-0.24076	-0.06076	1.58012
35	0.31081	-0.29281	0.50769	-0.16762	-0.20484	-0.19791	-0.03745	0.09081	4.09634	0.31827	-0.06943	-0.29975	0.32709	0.12044	0.11904
36	0.20953	-0.03818	-4.61873	0.14973	0.0037	0.33899	-0.03873	-0.01864	-0.66411	-0.17317	-0.30592	0.20641	0.1664	0.0287	-2.59223
37	-0.12816	-0.49175	-0.32243	0.32293	-0.0773	-0.10922	-3.84464	-0.086	0.55664	-0.57374	-0.01671	-0.37753	2.73268	0.0129	0.2055
38	-0.04304	0.38747	-0.48565	6.6804	-0.17147	2.04072	-0.14165	2.11734	1.18648	-0.3213	0.04176	-2.10021	0.1409	0.04939	-0.09682
39	-0.23431	-0.39817	-2.50515	2.73176	0.44566	0.12883	0.07136	-0.26544	-0.05658	-0.3932	0.26538	-0.32006	-0.84653	-0.53141	0.35137
40	0.24926	-1.4596	-1.53361	-0.00392	0.18712	0.45374	-0.37054	0.14934	3.35257	-0.57695	-0.02447	0.35588	0.72366	-0.08834	2.76379
41	-2.18475	0.59446	1.69281	-1.71898	-0.1597	-0.34504	0.21206	-2.31172	-0.06838	-0.53697	0.31962	-3.119	-0.52432	-0.29093	0.02138
42	0.02535	-0.00638	0.40833	0.49843	-0.03493	2.87832	-0.01969	-0.17181	-1.57212	-0.17886	-0.07614	-4.02344	-0.20691	-0.04756	-0.28897
43	0.84721	-0.97297	1.48798	2.03806	-0.41921	0.47076	-0.01131	0.87391	0.24244	-0.72018	0.11951	0.01512	0.15349	-0.39194	-0.35441
44	-0.05379	0.13581	-1.61996	0.01502	1.48105	-0.70295	0.12641	-0.01563	2.80001	0.08331	-0.02101	0.15741	-3.32673	0.09812	-0.02176
45	0.10918	0.00534	0.02197	-0.04859	-0.00922	-0.16951	-0.18902	0.08229	0.1683	0.15148	0.00211	0.11017	0.09813	-0.06703	-5.55034
46	-0.23617	0.13452	-0.62484	-0.22892	0.18087	0.99586	-0.00479	0.04154	-0.09448	0.19724	-0.03772	2.97732	-0.01384	-0.15837	0.53828
47	5.52906	0.02762	0.20244	-0.91659	0.18641	1.1037	0.4827	-1.34446	-0.52231	1.02205	-0.71495	0.60298	0.07695	0.16855	-0.24343
48	0.04969	0.04166	3.90498	0.12914	0.10583	1.60887	0.28849	0.23026	0.23761	0.17417	0.04373	-4.38149	-0.12667	0.01673	-0.38967
49	-0.32595	0.91283	0.19813	-3.53027	0.15008	0.32736	-0.10057	0.63967	1.70367	-0.75419	0.41418	0.91815	0.3866	0.20243	0.45315
50	0.12787	0.89428	0.49125	1.93351	-0.2943	1.33034	-0.15342	-2.33434	-0.51309	-0.48272	-0.01139	4.08678	0.225	0.03052	-0.6822
51	0.2249	-1.13364	-0.15203	0.27431	-0.00937	0.0422	0.1656	-0.13127	-0.32336	0.04394	0.06511	-0.09086	-0.18273	4.28353	-0.12916
52	-0.17045	-0.12573	-0.0861	-0.26732	-0.1267	-4.47492	-0.04788	-0.14898	-0.03276	0.22927	-0.01643	0.5113	0.05231	-0.02346	-0.11365
53	-0.18961	-0.85965	0.183	-0.76875	-0.08415	-0.30928	0.11793	-0.13249	0.06233	-5.8757	-0.06402	-0.29197	0.32138	-0.72959	0.08391
54	0.01163	0.35835	-0.35645	0.05301	-0.06936	0.115	-0.85637	-0.02587	0.34813	0.03458	0.10604	-0.42839	3.83733	0.09922	0.13178
55	2.23825	-1.1369	-0.4796	-0.21205	-0.42737	0.25157	0.83012	-0.72178	-0.20955	-0.03695	-0.22433	0.03494	-0.24387	-0.13344	-0.00317
56	-0.09374	-0.22337	0.94356	0.16001	4.11983	-0.09552	-0.00314	-0.0841	0.20084	-0.28334	-0.31738	0.101	0.61795	-0.85455	-0.66474
57	0.15478	-0.1689	-0.46269	-0.16716	0.15786	-0.32771	3.27557	0.0867	0.12052	0.15281	-0.14379	-0.09824	2.53738	-0.31277	0.21432
58	-0.02706	-1.31959	0.182	-2.87913	-0.81584	-0.33674	-0.27367	-1.37687	-0.29986	-0.71203	-0.1303	0.04717	-0.47065	0.82016	0.11175
59	-3.67304	0.19346	0.08016	-0.0663	3.40086	0.42772	-0.2013	0.35015	-0.01138	-0.4813	-0.34484	0.05374	0.12099	0.19907	-0.108
60	0.15044	0.16044	0.80289	-0.08943	-0.08978	-1.9377	-0.06831	0.25188	-2.75692	-0.07356	0.03098	1.66849	0.0751	-0.11883	0.78521
61	6.61346	-0.07342	0.49361	0.39715	-0.46374	0.70829	-0.18139	0.26013	0.37814	0.11156	-0.17548	-0.33876	0.58922	0.01438	-0.51686
62	2.92212	2.58846	-1.07459	-0.14242	0.51651	-0.2142	0.0479	0.51835	0.04167	0.1955	-0.12342	-0.87249	0.16811	-0.13763	-0.08772
63	-3.37307	-0.02352	-0.66619	0.55595	0.12855	0.31605	1.37403	-0.16533	-0.28366	0.54989	-0.14359	0.29074	-0.43325	-0.34333	-0.15383

Table C. 44 Pitch codebook 0 for MODE_VQ == 08_06

VN	0	1	2	3	4	5	6	7	8	9
0	1.28888	0.26874	3.16758	1.37315	0.49535	0.94071	1.44487	0.37004	-0.70932	-0.05462
1	-0.02234	-0.20001	0.19604	0.26626	0.75466	-0.81913	0.03708	-0.29647	-1.78632	-0.18008
2	-2.19075	0.13333	2.93822	-0.22869	0.54111	0.51598	0.29629	-1.60571	0.48978	-0.40555
3	-0.16435	-1.43562	0.12102	0.1109	0.83013	0.16508	-0.07873	2.78031	0.15479	-0.28546
4	3.06554	-1.41708	-1.88582	0.14353	-1.66046	0.26375	0.08646	-0.41044	-0.23005	0.47026
5	-0.11899	-0.14149	0.03797	0.01927	3.72378	0.16278	0.28207	0.15989	-0.17563	0.48802
6	-0.19166	0.05232	0.68887	0.24894	-0.48014	0.01794	-0.09456	-0.03353	-3.71716	-0.37879
7	-0.08722	0.69099	0.52234	0.74945	-0.62316	0.58827	0.30028	1.0085	0.24395	0.06055
8	0.12958	-0.22199	0.96387	3.13907	-1.96687	0.53834	-0.69258	-0.04179	-0.39869	-0.03142
9	-0.02278	0.45226	-0.46875	0.03839	2.35515	-0.02039	-0.03662	-0.33315	-0.3716	-3.22201
10	1.50588	-0.78015	1.43802	-0.42078	-1.56427	-0.32711	-1.81516	-0.1517	-1.39942	-0.70676
11	0.03549	0.97029	0.17778	2.40233	-0.1207	1.79612	-0.09612	-3.32378	-0.06238	-0.47122
12	0.11414	-0.06499	1.95345	-0.03737	-0.08911	1.86812	0.03298	0.3419	-0.9445	0.26631
13	-1.20479	-0.70412	0.30736	1.66473	0.25032	-1.62769	0.26336	-2.39109	1.66663	-1.01706
14	0.27057	0.28062	4.60481	0.14478	-0.04332	-0.26043	-0.15716	-0.52842	1.15402	0.02068
15	-0.02468	-2.63793	-0.06035	2.17716	0.55198	1.33668	0.18469	0.73687	-1.59535	0.47395
16	0.08763	-1.03259	0.29432	0.23395	-0.21428	0.13532	-4.58495	0.5589	1.19783	-1.8095
17	-1.87625	1.93644	0.2355	0.60058	1.07984	-0.13744	-0.89547	-0.8501	0.80741	-0.42413
18	0.23718	-1.82628	2.62266	0.48302	0.5773	0.00497	-0.88499	0.11386	0.50195	-1.61181
19	2.29405	0.76792	0.77194	0.45336	-0.15374	-1.68579	0.40538	-0.73821	-0.52721	0.27069
20	0.31311	0.67546	0.05131	-0.53788	0.75649	-1.92602	0.44063	-3.20197	0.24121	-1.52231
21	0.38234	-0.34155	-2.34677	0.44454	-0.12667	-0.01386	0.83684	-3.42385	0.27861	-0.62057
22	-2.294	-0.00411	0.10751	-2.0974	0.55445	-1.90704	0.46176	-0.22817	-1.05712	-0.31803

23	0.08838	0.104	0.07782	0.28709	0.77533	0.5377	-3.49315	-0.15122	-0.71822	-0.55433
24	-0.12964	-0.15924	0.08426	4.31671	0.35474	-0.3812	0.08329	-0.5072	0.22773	0.10575
25	0.37323	0.11307	-0.26726	1.19933	-0.2052	3.97109	0.23754	0.42287	-0.17526	1.6363
26	-0.8932	0.98569	-0.17186	-0.66099	0.34743	0.31512	0.56153	0.03641	-0.2279	-0.32669
27	0.49296	1.50705	0.11747	3.64671	-0.08886	0.95113	-0.10827	0.24228	-0.2188	0.09239
28	-0.05516	-0.36735	0.06542	-0.10454	-0.64008	-0.07052	-0.50515	2.55652	-0.94717	-1.30125
29	0.09405	0.74533	-1.13331	0.37229	-0.38024	0.17472	0.61134	0.20712	-0.10988	-0.49257
30	-0.11187	-0.17222	-1.73824	-0.13261	-0.52195	0.62213	-0.08559	0.17988	3.15089	-0.24704
31	0.01548	3.74936	0.08098	-0.20425	-0.05489	-0.31726	0.69418	0.32569	-0.1634	0.50865
32	-0.17257	0.28592	2.00649	0.53473	2.57991	0.33445	-0.33589	-0.71256	0.51466	2.46521
33	-0.09483	1.08439	0.03998	0.11961	1.85047	0.00631	-0.45111	-0.71269	0.15621	1.16161
34	-0.02632	2.00056	-0.32356	-0.56487	0.10923	1.26427	-0.2674	2.41749	-0.16922	-0.8848
35	-0.0544	0.16837	2.56225	0.1894	-0.22447	-0.39572	0.49943	0.11457	0.35493	0.89878
36	-0.26616	-0.43364	0.3558	0.05177	0.50149	-0.24278	0.17047	-0.12116	5.45574	-0.15218
37	0.06203	0.23433	-0.99429	0.12958	-0.628	-3.51385	-0.24958	0.06293	-0.89521	0.2426
38	3.72107	1.23981	-0.88335	0.46989	-0.3001	-0.42813	0.41867	1.0639	0.31391	-0.57627
39	-1.12063	0.44526	1.91996	0.34752	-0.524	-0.6746	-2.59639	0.11581	-0.20243	0.28307
40	-0.03832	-0.01316	-0.03846	-0.007	0.04931	-0.09357	0.06964	-0.00049	0.21041	-0.08892
41	0.10125	-0.03012	-0.03539	-2.37631	0.28491	2.4899	-0.34378	-0.73052	0.52019	-0.10068
42	1.15802	0.7305	0.86786	-0.15069	2.3452	-0.69111	0.57534	1.29397	0.15267	-0.16541
43	-0.12218	-0.03125	-0.01363	-1.5446	-0.27404	-0.62664	0.46486	-0.44052	0.17302	0.77978
44	-0.03625	-0.33463	-1.59515	-0.42669	-0.3637	3.45674	0.35785	0.34057	0.26874	-0.66518
45	0.12284	0.01668	-0.36273	0.00005	0.05147	-0.29728	-0.00085	5.10734	0.41718	-0.05075
46	-0.20474	1.11663	-1.4853	-1.6009	-0.02503	-0.24016	-0.09167	2.05302	0.41858	-0.52989
47	-1.22133	-0.80464	3.2915	0.29576	-1.01545	0.85362	0.25387	1.69078	0.05201	-0.3414
48	0.10775	-0.30053	-0.19873	0.20553	0.38424	0.2795	0.33175	0.10667	-0.43378	5.0211
49	0.06324	-1.1164	-0.34474	-0.03852	2.47664	0.30992	0.34303	-3.0838	-0.26944	-0.15003
50	-3.87998	0.08673	-0.92651	-0.28277	0.7912	0.06152	0.05675	-0.04378	-0.15055	-0.49744
51	0.06965	-0.24747	-0.30479	0.44796	-0.38039	0.30899	-2.49186	0.15352	-0.07609	2.77728
52	0.03771	-0.07037	1.44405	-0.08627	0.01465	0.65797	-0.20831	-0.02516	3.82295	0.58679
53	-0.28911	0.00076	2.25478	-2.67418	1.08246	0.20249	0.62566	-0.04001	-0.20703	-0.15767
54	1.76926	0.21839	0.4214	-0.18353	-0.23904	0.25714	-0.23697	0.07767	0.53228	0.22643
55	0.19622	-0.064	0.10897	0.79608	-1.20257	-1.10434	0.30441	0.49781	3.11769	-0.29082
56	-0.31973	-1.36087	1.1628	0.16043	-1.01046	2.64181	-0.19644	-0.27073	-0.77847	-0.42009
57	0.04065	-2.4468	-0.45823	0.51819	0.08004	-1.72072	-0.52	0.59359	0.37168	0.78352
58	-2.06255	0.4418	-1.32138	2.39634	-0.89775	1.60514	-1.52266	-0.22682	-0.10606	-0.5798
59	-3.31681	0.85384	-0.27833	0.29965	-1.55139	-0.12387	0.08035	-0.41261	0.26349	-0.21486
60	-0.91388	-0.98165	-1.13409	-1.47777	0.23383	1.1217	-0.27391	-2.83883	0.37536	-0.83449
61	-0.30317	0.03241	-2.27905	0.39328	-0.10539	2.62031	-0.21607	-0.11414	-3.2875	0.16852
62	0.05213	2.18937	0.84015	1.33077	-0.05358	1.3409	-0.09204	0.09985	0.93062	0.59825
63	0.12392	1.59708	0.9337	-0.90451	-0.12543	-2.01707	0.37513	1.94555	0.23099	-0.70055

Table C. 45 Pitch codebook 1 for MODE_VQ == 08_06

VN	0	1	2	3	4	5	6	7	8	9
0	2.13905	0.10191	1.24373	0.51358	1.63431	-0.08351	0.30739	-1.16675	-0.35279	-0.37096
1	0.0812	1.07058	0.15907	-0.09428	-0.14151	-0.4618	-0.21325	0.45456	0.66027	3.37816
2	-1.32206	-0.4821	1.43573	1.74563	2.01948	-1.56545	0.17182	0.45271	0.0628	-0.91456
3	-2.3898	-2.35197	-0.22554	0.33998	-0.79031	0.06321	-0.01377	1.07176	1.19673	0.2861
4	1.24864	-0.63868	-1.0763	-0.28812	1.14537	1.80661	-0.15262	2.16631	1.67133	-0.29882
5	-0.05193	-1.29291	-0.85556	-0.03597	4.35258	-0.32884	0.79375	0.41554	0.35156	-0.57719
6	-0.1198	-0.05545	0.43083	-0.10895	-2.37948	-0.49303	2.75568	-0.04768	-2.33369	-0.07613
7	0.21686	1.41341	-0.42804	-2.70242	0.61944	-0.06463	-0.43017	-0.1903	-2.0755	-0.07512
8	-0.02184	0.4818	0.88869	4.23426	0.32367	-0.16813	-0.0971	-0.93254	0.52898	0.05603
9	-0.15532	0.52294	-1.60824	-0.18328	3.23327	0.1135	-1.45172	-0.0493	-0.23632	-0.25873
10	-0.05867	-2.59416	0.8312	0.31796	-0.30826	0.2373	2.77116	0.02446	0.55313	-0.22707
11	-0.51455	1.30626	-0.77779	2.71737	0.5061	-1.37105	0.15878	0.48821	0.25587	1.37404
12	-0.12972	0.16455	2.0636	0.25143	0.28128	2.95291	0.08582	0.09976	-0.25123	-0.30165
13	-0.04632	-1.3975	-0.14369	0.86661	0.47791	-3.43773	-0.08283	0.40818	-0.40507	1.10291
14	-0.0182	-0.15582	1.34311	0.06751	-0.14121	-0.53351	0.03831	0.5043	2.10788	-0.58678
15	0.39477	-3.04864	-0.45931	2.01978	0.04376	-0.52536	-0.31599	-0.12492	-0.10121	0.11121
16	-0.03452	-0.03057	-0.09894	0.03226	0.01192	-0.48283	-4.65939	-0.11003	-0.3613	0.079
17	-1.31754	0.0574	-0.2971	0.08453	-0.30721	0.86434	2.45968	2.51238	1.23547	-0.50231
18	1.50294	-0.24216	2.28121	-0.90775	1.04895	0.21859	-2.06629	0.08329	0.34822	0.1581
19	0.36466	0.70677	-0.918	2.47546	3.79165	-0.24031	-0.37853	-0.32197	-0.37636	-0.02901
20	-0.12627	-0.04525	-0.82218	-2.26543	-0.84347	-1.03939	0.26653	-2.47079	1.23837	0.75649
21	0.24064	-0.62543	-2.94637	0.47164	1.58428	0.23203	-0.59003	0.3325	-0.13853	0.14422
22	-0.17463	-0.12583	0.85598	-0.17722	1.54264	0.05216	-0.08885	0.88151	0.46255	-0.48946
23	-0.06676	0.36942	0.58086	0.11589	-1.18536	-0.12857	-2.80603	0.00658	1.06034	-0.58457
24	-0.1561	-0.18171	-0.63499	2.6653	0.46641	0.59787	0.30223	-0.53096	1.29943	-0.98973
25	-0.00976	1.04356	0.16697	1.99034	0.23572	2.65679	-0.00325	1.03831	-0.0701	-0.77931
26	0.13381	0.49307	0.04619	-1.68299	0.19455	0.52731	0.14125	0.99717	0.43132	-0.97925
27	-0.1563	1.8791	0.19313	0.23078	0.30029	1.5847	0.36806	-0.91846	-0.10561	2.76858
28	-0.03141	-0.27603	-1.25022	0.76867	0.37696	-1.86725	0.45545	2.35843	0.57733	-0.51505
29	0.19081	-0.02002	0.89507	-0.17124	0.07392	0.07283	-0.47206	-0.03043	-0.34827	-4.29424
30	-0.29	0.04272	-1.72027	-0.11905	0.27361	-1.60983	0.1195	-0.37867	3.50315	0.36605
31	-0.1328	4.18273	0.04841	0.61437	0.60526	-0.23885	-0.05236	-0.82974	-0.29851	0.51121
32	0.03186	-0.24892	0.25614	-0.17325	3.12521	-0.12148	0.07243	-0.3332	-0.04557	-0.77374
33	-0.02997	2.20189	0.2339	0.05403	3.47503	-0.14415	0.23471	0.75477	0.00188	0.06228
34	-1.11961	0.43678	0.61581	-3.07064	-0.02222	-0.20063	-0.19664	-0.1743	1.29354	0.31722
35	0.80714	-0.31437	3.21672	0.05728	0.42811	-0.01624	0.4669	-0.14758	0.79094	-0.49991
36	-0.04883	0.18854	-0.0422	-0.24325	0.33396	0.58096	0.07173	0.18625	4.80098	-0.04774
37	0.12902	0.04455	-0.58451	0.22529	0.18288	-3.31484	0.20874	0.1439	-2.33226	-0.00004
38	1.4006	2.53879	2.23308	-0.40397	-0.91191	0.30424	-0.59309	-0.28013	-0.53681	-0.72096
39	0.14981	-0.34273	-0.1096	-2.08103	-0.55452	1.22105	-2.32346	0.71767	-0.04973	0.88489

40	-0.06296	0.00779	0.37141	-0.07311	-0.43599	-0.69232	0.37183	0.59265	-0.20391	0.76627
41	-0.04811	0.81816	-0.06811	-3.60488	-0.30926	1.2619	-0.25378	-2.67757	0.01421	0.15661
42	-0.18735	-0.29123	-0.1443	-0.17479	-0.2036	1.97977	-0.22341	-0.17369	-2.54577	-0.41017
43	-0.15575	0.64521	-0.20741	-0.06303	2.55173	0.08849	-1.3168	-0.0852	-3.59554	-0.16002
44	0.2538	0.42032	-2.50621	0.13451	-0.06898	2.41402	0.09969	-0.09772	1.45985	0.45849
45	0.19057	-0.22542	-0.16274	-0.19255	-0.085	0.1524	-0.06344	3.54373	-0.19634	0.42005
46	-0.00046	1.3075	0.44001	0.29739	0.12153	0.45681	-0.24122	3.97823	1.01631	-0.5718
47	-1.0474	0.76204	0.79817	0.02734	-0.85256	0.27437	-0.36512	0.45476	-0.34809	-0.26607
48	0.39011	-0.04813	0.28325	-0.40521	-0.26705	0.11646	-1.00205	-0.09658	0.39184	6.38045
49	0.18221	-3.1436	0.56642	-0.41265	2.10605	0.00959	0.32737	-0.30618	-0.46143	0.62372
50	-3.41695	-0.29748	0.93423	0.2968	-0.22186	-0.04393	0.4199	-0.02121	-0.0349	0.24145
51	0.08292	-0.34821	-0.98374	0.76693	-2.30082	0.11379	-2.00866	0.10299	-1.80156	0.49759
52	0.09801	0.02398	0.87932	-0.29853	-0.23085	0.60271	0.283	0.17743	-0.0655	0.11407
53	0.0495	-2.7049	0.93988	-0.84697	-0.47967	-0.45333	-0.34552	-0.60481	0.77467	1.79495
54	2.31525	-1.35283	1.51663	0.46045	-0.94423	0.66598	0.14087	-1.23472	0.07825	0.23723
55	-0.01891	0.01146	0.05543	-0.00939	-0.25309	-0.05408	-0.08464	-0.08331	0.28796	-0.23134
56	0.14456	-1.82728	0.54162	-1.45394	0.65876	2.88044	0.04801	1.52839	0.4177	0.24366
57	0.00192	-1.51934	-0.28914	0.27953	-0.43474	0.3359	0.34038	0.09476	-0.16812	0.55648
58	1.58963	0.62054	-1.19708	0.95618	-0.19293	0.13055	-0.46467	-1.08598	1.0294	-0.00537
59	-1.63945	1.27057	1.17362	0.77949	-0.25979	1.31919	1.09939	-0.59425	-1.04889	0.33754
60	-0.36973	-1.0016	0.18249	0.95907	0.72926	2.32241	0.60848	-2.12186	-0.24255	-0.65101
61	-0.00988	-0.04139	-0.76083	0.14922	0.57784	3.97886	-0.30668	-0.34279	-0.84522	0.11366
62	-0.12785	1.334	0.11998	0.17243	-1.82888	0.49492	0.09101	-0.99747	0.39561	-1.90151
63	0.1718	1.22115	0.41649	2.03242	-0.20785	-2.41295	-0.75	1.09243	0.65831	-2.08726

Table C. 46 LSP codebook 1 for MODE_VQ == 24_06 / 24_06_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12	
0	0	0.1650	0.3215	0.4376	0.5822	0.7322	0.8908	1.0505	1.2243	1.3871	1.5196	1.6560	1.8015	
	12	1.9661	2.0974	2.2348	2.4024	2.5544	2.6831	2.8283	2.9839					
1	0	0.1440	0.2613	0.3929	0.5390	0.6673	0.8243	0.9358	1.0599	1.1925	1.3569	1.5274	1.7049	
	12	1.8502	1.9903	2.1580	2.3390	2.4727	2.6148	2.7870	2.9539					
2	0	0.1145	0.3071	0.5101	0.6939	0.8315	0.9808	1.1159	1.2326	1.3347	1.4523	1.5814	1.7163	
	12	1.8697	2.0459	2.2344	2.4024	2.5466	2.6765	2.8185	2.9763					
3	0	0.1550	0.2946	0.4537	0.6124	0.7058	0.8083	1.0013	1.2107	1.3308	1.4104	1.5393	1.7495	
	12	1.9266	2.0508	2.1710	2.3505	2.5233	2.6695	2.7830	2.9203					
4	0	0.1554	0.2519	0.3708	0.5171	0.6480	0.7764	0.8840	0.9777	1.1273	1.2853	1.4674	1.6178	
	12	1.7646	1.9532	2.1267	2.3057	2.4733	2.6127	2.7803	2.9591					
5	0	0.1307	0.2410	0.3598	0.5154	0.7008	0.8852	1.0244	1.1526	1.2929	1.4664	1.6246	1.7354	
	12	1.8342	1.9609	2.1143	2.2644	2.4112	2.6023	2.7868	2.9548					
6	0	0.0991	0.1927	0.3853	0.6047	0.7963	0.9651	1.1749	1.3861	1.5342	1.6281	1.7041	1.7797	
	12	1.8652	1.9880	2.1160	2.2619	2.4396	2.6205	2.7883	2.9602					
7	0	0.1149	0.2251	0.3812	0.5659	0.7384	0.9209	1.0912	1.2207	1.3313	1.4514	1.5865	1.7312	
	12	1.8540	2.0139	2.1852	2.3544	2.5212	2.6615	2.7829	2.9413					
8	0	0.1157	0.2013	0.3177	0.4915	0.6670	0.8286	1.0002	1.1807	1.3418	1.4895	1.6245	1.7676	
	12	1.9092	2.0667	2.2347	2.3928	2.5539	2.6961	2.8290	2.9747					
9	0	0.1471	0.2302	0.3288	0.4481	0.5557	0.6855	0.8324	0.9898	1.1471	1.3052	1.4789	1.6482	
	12	1.8010	1.9741	2.1495	2.3035	2.4569	2.6260	2.7938	2.9671					
10	0	0.1856	0.3075	0.4611	0.6171	0.7198	0.8065	0.9310	1.1047	1.2923	1.4358	1.5578	1.7385	
	12	1.9511	2.1395	2.2489	2.3533	2.5115	2.7057	2.8630	2.9820					
11	0	0.1493	0.2412	0.3491	0.4860	0.5901	0.6966	0.8535	1.0470	1.2244	1.3835	1.5220	1.6805	
	12	1.8409	2.0140	2.1654	2.3165	2.4677	2.6116	2.7763	2.9609					
12	0	0.1406	0.2575	0.3981	0.5580	0.7094	0.8630	0.9983	1.1340	1.2674	1.4195	1.5839	1.7432	
	12	1.8955	2.0472	2.1974	2.3621	2.5120	2.6637	2.8178	2.9876					
13	0	0.1365	0.2496	0.3770	0.5201	0.6692	0.8355	0.9888	1.1227	1.2352	1.3598	1.5339	1.7242	
	12	1.8703	2.0052	2.1487	2.3199	2.4933	2.6682	2.8231	2.9675					
14	0	0.1329	0.2228	0.3753	0.5557	0.6998	0.7960	1.0155	1.1321	1.2338	1.3872	1.5716	1.7452	
	12	1.8385	1.9879	2.1989	2.3173	2.4558	2.6479	2.8505	2.9847					
15	0	0.1405	0.2522	0.3871	0.5539	0.6679	0.8085	0.9782	1.1256	1.2834	1.4237	1.5334	1.6838	
	12	1.8411	2.0095	2.1587	2.3210	2.4584	2.6029	2.7845	2.9541					
16	0	0.1730	0.3086	0.4441	0.5673	0.6766	0.8589	1.0437	1.1439	1.2665	1.4710	1.6407	1.7470	
	12	1.9276	2.1170	2.2291	2.3440	2.5186	2.6869	2.7890	2.9248					
17	0	0.1472	0.2522	0.3764	0.5331	0.6932	0.8436	0.9735	1.1143	1.2887	1.4567	1.6016	1.7378	
	12	1.8599	1.9913	2.1328	2.2723	2.3861	2.5333	2.7033	2.9075					
18	0	0.1601	0.2807	0.4260	0.5847	0.7252	0.8528	0.9535	1.0889	1.2720	1.4736	1.6478	1.7808	
	12	1.8838	2.0119	2.1692	2.3224	2.4384	2.5817	2.7478	2.9359					
19	0	0.1548	0.2476	0.3685	0.5377	0.7176	0.8714	0.9762	1.0849	1.2381	1.4341	1.6144	1.7844	
	12	1.9428	2.0689	2.1772	2.3227	2.4730	2.6300	2.7904	2.9511					
20	0	0.1856	0.3009	0.4106	0.5318	0.6380	0.7781	0.9339	1.1090	1.2743	1.4462	1.6301	1.8036	
	12	1.9718	2.1250	2.2112	2.3323	2.4857	2.6273	2.7547	2.9224					
21	0	0.0924	0.2289	0.3726	0.4893	0.6845	0.8278	0.9408	1.1536	1.2597	1.3593	1.5596	1.6953	
	12	1.8063	2.0220	2.1281	2.2973	2.4607	2.5804	2.7794	2.9459					
22	0	0.1452	0.2368	0.3545	0.5098	0.6426	0.8124	0.9939	1.1475	1.2978	1.4435	1.5855	1.7402	
	12	1.8867	2.0420	2.1718	2.3189	2.4471	2.5761	2.7399	2.9360					
23	0	0.1065	0.2011	0.3299	0.4837	0.6513	0.8406	0.9912	1.1124	1.2517	1.4317	1.5716	1.6827	
	12	1.8496	2.0381	2.1709	2.3073	2.4583	2.6242	2.7921	2.9713					
24	0	0.1090	0.2070	0.3122	0.4386	0.5664	0.7411	0.9291	1.0970	1.2622	1.4268	1.5825	1.7414	
	12	1.8898	2.0509	2.2116	2.3605	2.5114	2.6606	2.8107	2.9758					
25	0	0.1057	0.1911	0.2715	0.4003	0.5739	0.7804	0.9531	1.1072	1.2610	1.4203	1.5808	1.7437	
	12	1.8813	2.0321	2.1395	2.2829	2.4626	2.6548	2.8321	2.9887					
26	0	0.1830	0.3617	0.5045	0.6262	0.7522	0.9129	1.0331	1.1184	1.2078	1.4009	1.5685	1.6773	
	12	1.8109	2.0355	2.2018	2.3385	2.4899	2.6304	2.7552	2.9157					
27	0	0.1254	0.2479	0.3900	0.5091	0.6113	0.7797	0.9822	1.1037	1.1929	1.3823	1.5694	1.6692	
	12	1.8131	2.0411	2.1874	2.3204	2.4878	2.6575	2.7965	2.9444					
28	0	0.1275	0.2093	0.3074	0.4554	0.6171	0.7976	0.9875	1.1676	1.3003	1.4178	1.5570	1.7202	
	12	1.8979	2.0705	2.1996	2.3359	2.4796	2.6300	2.7887	2.9664					
29	0	0.1190	0.2072	0.3204	0.4689	0.6384	0.8210	1.0006	1.1535	1.2770	1.3989	1.5495	1.7091	
	12	1.8526	2.0032	2.1626	2.3366	2.5165	2.6702	2.8120	2.9721					
30	0	0.1371	0.2360	0.3399	0.4679	0.6157	0.7920	0.9383	1.0866	1.2290	1.3652	1.4984	1.6578	
	12	1.8188	1.9921	2.1324	2.2829	2.4557	2.6149	2.7786	2.9585					
31	0	0.1218	0.2134	0.3474	0.5132	0.6643	0.8554	1.0141	1.1064	1.1878	1.3548	1.5500	1.7169	
	12	1.8913	2.0685	2.1878	2.3308	2.4972	2.6570	2.7942	2.9541					
32	0	0.1661	0.2850	0.4149	0.5673	0.7145	0.8571	1.0023	1.1592	1.3169	1.4649	1.6131	1.7638	

	12	1.9044	2.0636	2.2217	2.3688	2.5124	2.6614	2.8064	2.9691					
33	0	0.1304	0.2235	0.3391	0.4833	0.6222	0.7598	0.9114	1.0656	1.2080	1.3671	1.5333	1.6974	
	12	1.8455	1.9925	2.1577	2.3263	2.4899	2.6479	2.7998	2.9651					
34	0	0.1928	0.3515	0.4757	0.6190	0.7858	0.9251	1.0253	1.1506	1.3035	1.4485	1.5951	1.7644	
	12	1.8962	2.0343	2.1909	2.3490	2.4989	2.6589	2.8191	2.9553					
35	0	0.1675	0.3365	0.4406	0.5687	0.7007	0.8082	0.9515	1.0763	1.1657	1.2923	1.4438	1.6230	
	12	1.7529	1.8994	2.1188	2.2915	2.4531	2.6294	2.7689	2.9412					
36	0	0.1875	0.3014	0.3940	0.5283	0.6704	0.8090	0.9534	1.1350	1.2630	1.4189	1.5922	1.7339	
	12	1.9189	2.0515	2.1611	2.3616	2.4704	2.5960	2.7679	2.9148					
37	0	0.1507	0.2621	0.3713	0.5117	0.6522	0.7800	0.8949	1.0581	1.2478	1.4490	1.6148	1.7476	
	12	1.8627	2.0005	2.1583	2.3211	2.4668	2.6153	2.7978	2.9675					
38	0	0.1729	0.3346	0.4652	0.6141	0.7600	0.8599	0.9577	1.1293	1.2960	1.4600	1.6410	1.7734	
	12	1.9018	2.0789	2.2302	2.3788	2.5110	2.6194	2.7601	2.9430					
39	0	0.1718	0.2992	0.4238	0.5733	0.7067	0.8555	0.9982	1.1020	1.2380	1.4443	1.5589	1.6832	
	12	1.8717	2.0280	2.1556	2.3187	2.4827	2.6129	2.7894	3.0025					
40	0	0.1180	0.2176	0.3538	0.5249	0.6729	0.7978	0.9500	1.1292	1.3053	1.4670	1.5994	1.7385	
	12	1.8984	2.0630	2.2234	2.3712	2.5155	2.6604	2.8203	2.9818					
41	0	0.1591	0.2591	0.3804	0.5211	0.6672	0.7915	0.8986	1.0498	1.2221	1.4029	1.5877	1.7710	
	12	1.9252	2.0812	2.2222	2.3774	2.5386	2.6825	2.8150	2.9627					
42	0	0.1651	0.2605	0.3647	0.5287	0.6621	0.8098	0.9738	1.1235	1.2295	1.3731	1.5279	1.6917	
	12	1.8350	1.9689	2.1130	2.2689	2.4304	2.5898	2.7457	2.9305					
43	0	0.1717	0.2745	0.3848	0.5395	0.6824	0.8203	0.9737	1.0844	1.1948	1.3838	1.5354	1.7176	
	12	1.8608	2.0426	2.1941	2.3380	2.5062	2.6313	2.7992	2.9431					
44	0	0.1908	0.3130	0.4312	0.5649	0.6814	0.8200	1.0236	1.1644	1.2658	1.4135	1.5735	1.7401	
	12	1.8795	2.0834	2.2396	2.3512	2.4725	2.6417	2.8558	2.9986					
45	0	0.1733	0.2814	0.3929	0.5409	0.6689	0.7865	0.9318	1.0943	1.2617	1.4002	1.5019	1.6649	
	12	1.8668	2.0656	2.2015	2.3247	2.4791	2.6583	2.8195	2.9537					
46	0	0.1752	0.3025	0.4111	0.5443	0.7135	0.8616	0.9526	1.0912	1.2701	1.4009	1.5454	1.7544	
	12	1.8730	2.0315	2.2127	2.3509	2.4808	2.6454	2.7844	2.9310					
47	0	0.1912	0.3310	0.4269	0.5807	0.7354	0.8432	0.9666	1.1541	1.2599	1.3986	1.5779	1.6742	
	12	1.8027	1.9895	2.1425	2.3302	2.4704	2.5681	2.7400	2.9385					
48	0	0.1261	0.2406	0.3670	0.5245	0.6555	0.7954	0.9349	1.0964	1.2441	1.4053	1.5507	1.6706	
	12	1.7878	1.9201	2.0929	2.2690	2.4355	2.6145	2.7877	2.9603					
49	0	0.1205	0.2852	0.4869	0.6858	0.8046	0.9002	0.9779	1.0755	1.2698	1.4900	1.7041	1.8181	
	12	1.8753	1.9550	2.1433	2.3739	2.5952	2.7283	2.8063	2.9251					
50	0	0.1037	0.2419	0.4597	0.6671	0.7926	0.9451	1.0862	1.1873	1.2983	1.4215	1.4869	1.5673	
	12	1.6495	1.8015	2.0399	2.2786	2.4522	2.6105	2.7820	2.9602					
51	0	0.1520	0.3347	0.5154	0.6704	0.7580	0.8280	0.9593	1.1763	1.3816	1.5150	1.5736	1.6706	
	12	1.8566	2.0910	2.2533	2.3322	2.4453	2.6367	2.8520	3.0084					
52	0	0.1630	0.2933	0.4185	0.5585	0.7427	0.8995	1.0005	1.0986	1.2545	1.4131	1.5140	1.6647	
	12	1.8443	1.9790	2.1361	2.3475	2.4734	2.6270	2.8246	2.9559					
53	0	0.1086	0.2317	0.4168	0.6238	0.8316	1.0177	1.1119	1.1726	1.2452	1.3184	1.4726	1.6984	
	12	1.8906	2.0806	2.2634	2.3981	2.4917	2.5927	2.7415	2.9370					
54	0	0.1758	0.3211	0.4676	0.5874	0.6848	0.8439	1.0554	1.2151	1.3024	1.4437	1.6444	1.8092	
	12	1.8877	2.0104	2.2105	2.3896	2.4999	2.6150	2.7839	2.9784					
55	0	0.1839	0.3133	0.4329	0.5478	0.6500	0.8277	1.0157	1.1291	1.2653	1.4869	1.6207	1.6972	
	12	1.8037	2.0343	2.1838	2.3036	2.4900	2.6616	2.7886	2.9283					
56	0	0.1106	0.1847	0.3450	0.5474	0.7128	0.8748	1.0542	1.2233	1.3879	1.5482	1.7076	1.8544	
	12	2.0377	2.2502	2.3317	2.3735	2.4706	2.5602	2.6696	2.8945					
57	0	0.1182	0.3140	0.4013	0.5658	0.6776	0.7660	0.9113	1.1627	1.3485	1.4638	1.6441	1.7621	
	12	1.9031	1.9937	2.0962	2.3087	2.4932	2.6422	2.8137	2.9251					
58	0	0.1504	0.2577	0.3827	0.5498	0.7037	0.8504	0.9746	1.1143	1.2696	1.4323	1.5864	1.7223	
	12	1.8201	1.9567	2.1489	2.3439	2.5120	2.6597	2.8077	2.9844					
59	0	0.1402	0.2443	0.3721	0.5198	0.6672	0.8198	0.9867	1.1748	1.3604	1.5358	1.6846	1.8132	
	12	1.9076	2.0349	2.1803	2.3279	2.4866	2.6422	2.7981	2.9606					
60	0	0.1545	0.2698	0.3886	0.5487	0.6789	0.8024	0.9705	1.1495	1.3102	1.4388	1.6051	1.7683	
	12	1.9162	2.0527	2.1842	2.3546	2.5312	2.6967	2.8139	2.9500					
61	0	0.1397	0.2567	0.4080	0.5710	0.6650	0.8080	0.9969	1.1766	1.2650	1.3892	1.5716	1.7588	
	12	1.8683	1.9966	2.1723	2.3647	2.4803	2.6017	2.7791	2.9788					
62	0	0.1661	0.2873	0.3961	0.5420	0.7142	0.8725	1.0358	1.1822	1.2892	1.4605	1.6068	1.7258	
	12	1.8781	2.0351	2.1801	2.3572	2.4756	2.6088	2.7750	2.9345					
63	0	0.1254	0.2432	0.3884	0.5464	0.6874	0.8191	0.9562	1.1370	1.3118	1.4631	1.5807	1.7012	
	12	1.8502	2.0163	2.1657	2.3289	2.4943	2.6480	2.8008	2.9550					

Table C. 47 LSP codebook 2 for MODE_VQ == 24_06 / 24_06_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.0555	0.0657	0.0137	-0.0113	0.0276	0.0414	0.0063	-0.0115	0.0353	0.0170	-0.0427	-0.0449
	12	-0.0003	-0.0028	-0.0147	-0.0055	-0.0005	0.0054	0.0133	-0.0025				
1	0	0.0106	0.0166	-0.0226	-0.0652	-0.0188	0.0520	0.0922	0.0794	0.0331	-0.0215	-0.0474	-0.0210
	12	0.0311	0.0424	0.0794	0.0917	0.0689	0.0298	0.0038	-0.0003				
2	0	-0.0420	-0.0696	-0.0794	-0.0581	-0.0116	-0.0021	-0.0052	-0.0105	-0.0114	-0.0082	0.0164	0.0469
	12	0.0731	0.0555	-0.0624	-0.0606	-0.0156	0.0009	0.0059	0.0096				
3	0	0.0269	-0.0008	-0.0387	-0.0122	0.0320	0.0148	0.0108	0.0331	0.0658	0.0322	0.0126	0.0394
	12	0.0418	0.0092	-0.0185	-0.0305	0.0095	0.0569	0.0712	0.0308				
4	0	-0.0036	0.0332	0.0427	-0.0141	-0.0332	0.0312	0.0418	0.0339	0.0355	0.0512	0.0650	0.0548
	12	0.0546	0.0462	0.0199	-0.0280	-0.0352	0.0226	0.0049	-0.0190				
5	0	-0.0006	-0.0007	0.0097	0.0028	0.0034	0.0214	0.0550	0.0744	0.0655	0.0337	0.0035	-0.0174
	12	-0.0333	-0.0253	-0.0217	-0.0777	-0.0890	-0.0346	0.0041	0.0076				
6	0	-0.0169	-0.0258	-0.0268	-0.0535	-0.0560	-0.0276	-0.0292	-0.0117	0.0021	0.0221	0.0072	-0.0263
	12	-0.0214	-0.0071	0.0058	-0.0066	-0.0377	-0.0377	-0.0004	0.0264				
7	0	0.0120	0.0306	0.0358	0.0320	0.0412	-0.0057	0.0103	0.0210	0.0266	0.0319	0.0165	-0.0040
	12	0.0175	0.0281	0.0157	0.0574	0.0391	-0.0193	-0.0211	0.0109				
8	0	-0.0427	-0.0213	-0.0091	-0.0248	0.0022	-0.0001	-0.0109	-0.0211	-0.0204	-0.0389	-0.0491	-0.0537
	12	-0.0513	-0.0262	-0.0191	-0.0127	0.0128	-0.0130	-0.0466	-0.0337				
9	0	-0.0360	0.0082	0.0618	0.0878	0.0621	0.0476	0.0478	0.0121	-0.0108	-0.0081	0.0173	0.0295
	12	0.0228	0.0067	0.0351	0.0450	0.0522	0.0614	0.0727	0.0409				
10	0	-0.0138	0.0547	0.0929	0.0496	0.0039	0.0103	-0.0042	-0.0241	-0.0379	-0.0411	-0.0252	-0.0081
	12	0.0146	0.0242	0.0481	0.0320	-0.0091	-0.0191	-0.0285	-0.0275				
11	0	-0.0493	-0.0256	0.0331	0.0288	-0.0031	-0.0306	-0.0335	-0.0013	0.0327	0.0472	0.0588	0.0539
	12	0.0139	-0.0244	-0.0286	0.0079	0.0133	-0.0063	0.0439	0.0656				
12	0	-0.0472	-0.0597	-0.0217	0.0227	0.0442	0.0224	-0.0061	-0.0350	-0.0125	0.0278	0.0335	0.0118
	12	0.0033	0.0017	0.0123	0.0161	0.0223	0.0217	0.0284	0.0182				
13	0	0.0573	0.0524	-0.0027	-0.0500	-0.0421	-0.0752	-0.0604	-0.0089	-0.0050	-0.0223	-0.0107	0.0122
	12	0.0306	0.0111	-0.0119	0.0215	0.0965	0.0896	0.0214	-0.0176				
14	0	0.0514	0.0969	0.0715	0.0091	-0.0059	-0.0350	-0.0781	-0.0906	-0.0578	-0.0100	0.0171	0.0206
	12	0.0202	0.0065	0.0558	0.0380	-0.0094	-0.0015	0.0404	0.0432				
15	0	0.0461	0.0285	0.0221	0.0388	-0.0081	0.0062	0.0174	0.0179	-0.0063	-0.0272	-0.0147	0.0069

	12	-0.0115	-0.0274	0.0408	0.0171	0.0301	0.0560	0.0120	-0.0129

Table C. 48 alpha codebook for MODE_VQ == 24_06 / 24_06_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.4950	0.4980	0.4999	0.4986	0.4984	0.4956	0.4950	0.4969	0.4986	0.4975	0.4966	0.4982
	12	0.4997	0.4968	0.4915	0.4906	0.4971	0.4924	0.4899	0.4818				
1	0	0.3372	0.3153	0.2946	0.2719	0.2599	0.2815	0.2590	0.2584	0.2626	0.2712	0.2549	0.2787
	12	0.2794	0.2483	0.2306	0.2545	0.2706	0.2875	0.2721	0.2851				

Table C. 49 LSP codebook1 for MODE_VQ == 16_16

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.2024	0.396	0.6143	0.8121	1.0059	1.1984	1.4115	1.6594	1.8991	2.1385	2.324	2.52
	12	2.5893	2.6641	2.7266	2.8705								
1	0	0.1212	0.348	0.4761	0.5657	0.663	0.7677	1.0193	1.2893	1.5195	1.7263	1.9459	2.2269
	12	2.3686	2.4891	2.6979	2.9222								
2	0	0.1626	0.4571	0.6193	0.6668	0.764	0.9905	1.1926	1.3141	1.53	1.8033	1.9137	2.0682
	12	2.3271	2.5159	2.6846	2.9427								
3	0	0.1512	0.2486	0.4308	0.5845	0.7969	0.972	1.1656	1.3707	1.5152	1.672	1.886	2.1606
	12	2.4176	2.6106	2.7875	2.9485								
4	0	0.1193	0.2803	0.5008	0.7023	0.8865	1.0891	1.2902	1.4961	1.6851	1.8637	2.0439	2.2346
	12	2.4157	2.5926	2.7798	2.9623								
5	0	0.171	0.2989	0.4738	0.5952	0.7348	0.9814	1.1563	1.3303	1.5023	1.7013	1.9038	2.1059
	12	2.306	2.4846	2.6654	2.8823								
6	0	0.1566	0.3175	0.4919	0.6606	0.8258	1.0248	1.2067	1.3828	1.5687	1.7864	1.9128	2.1215
	12	2.3346	2.52	2.7334	2.8966								
7	0	0.1647	0.2851	0.4544	0.6327	0.808	0.9724	1.1228	1.2803	1.4352	1.644	1.9086	2.1459
	12	2.3574	2.572	2.7532	2.9429								
8	0	0.1495	0.4451	0.6143	0.7094	0.7429	0.8719	1.1576	1.477	1.7006	1.8222	1.9765	2.182
	12	2.3653	2.4792	2.6123	2.85								
9	0	0.2155	0.3123	0.3664	0.44	0.4919	0.681	0.9765	1.2378	1.4755	1.7124	1.915	2.1085
	12	2.3404	2.5421	2.7442	2.9278								
10	0	0.1196	0.376	0.6183	0.7667	0.8178	0.9037	0.9776	1.1481	1.4456	1.7424	1.9579	2.1297
	12	2.3842	2.5898	2.7755	2.9621								
11	0	0.1302	0.2517	0.3809	0.5469	0.7365	0.9989	1.1801	1.3643	1.5365	1.7032	1.884	2.0944
	12	2.3305	2.5615	2.7723	2.942								
12	0	0.1312	0.2346	0.4368	0.6612	0.8457	0.9983	1.1686	1.3873	1.5649	1.696	1.8605	2.1128
	12	2.372	2.5757	2.7683	2.9445								
13	0	0.149	0.2612	0.4018	0.5665	0.7266	0.9078	1.1085	1.3314	1.5217	1.717	1.8788	2.0749
	12	2.2891	2.4843	2.69	2.9011								
14	0	0.1605	0.3049	0.4626	0.6324	0.8107	0.9779	1.1668	1.3787	1.5756	1.7409	1.9322	2.1343
	12	2.3247	2.5309	2.7316	2.9356								
15	0	0.1787	0.3138	0.4624	0.652	0.8182	1.0073	1.1939	1.3897	1.5331	1.7088	1.8857	2.0695
	12	2.2738	2.4807	2.6905	2.8979								
16	0	0.1841	0.3572	0.5329	0.724	0.9192	1.0985	1.2653	1.453	1.6366	1.8257	2.0194	2.1943
	12	2.3838	2.5738	2.7663	2.9477								
17	0	0.0708	0.1412	0.209	0.3534	0.6226	0.8815	1.0907	1.295	1.5268	1.7435	1.9355	2.1451
	12	2.3464	2.5496	2.7445	2.9351								
18	0	0.1494	0.4305	0.6498	0.7814	0.8468	0.9871	1.2032	1.4752	1.5951	1.7546	2.0064	2.2073
	12	2.3437	2.4978	2.7151	2.993								
19	0	0.0821	0.1234	0.2587	0.5075	0.741	0.9497	1.1337	1.3517	1.5708	1.7848	1.9765	2.1592
	12	2.3616	2.5534	2.7481	2.9326								
20	0	0.1346	0.2983	0.4986	0.6616	0.8388	1.0468	1.2311	1.4163	1.617	1.8309	2.0004	2.1902
	12	2.3935	2.5778	2.7658	2.946								
21	0	0.1906	0.2822	0.3641	0.3972	0.6127	0.9111	1.1789	1.3735	1.5772	1.7768	1.949	2.1644
	12	2.3519	2.5585	2.7491	2.9395								
22	0	0.1763	0.2963	0.4623	0.6705	0.8456	0.9992	1.1627	1.3308	1.4868	1.6437	1.8392	2.0813
	12	2.3158	2.5333	2.7455	2.939								
23	0	0.1964	0.3242	0.4644	0.5899	0.7542	1.0054	1.1817	1.3426	1.5056	1.6569	1.8243	2.0606
	12	2.2904	2.5082	2.7435	2.9402								
24	0	0.2529	0.4293	0.5734	0.7116	0.8684	1.1272	1.3451	1.5162	1.6333	1.7749	1.9745	2.1709
	12	2.3626	2.5513	2.7484	2.9121								
25	0	0.1862	0.2705	0.3478	0.3868	0.5165	0.7744	1.0569	1.3168	1.5158	1.7277	1.9246	2.1195
	12	2.3398	2.5323	2.7472	2.935								
26	0	0.2043	0.3573	0.5218	0.6807	0.8452	0.9777	1.1317	1.3193	1.4711	1.6853	1.8658	2.0654
	12	2.269	2.4698	2.6989	2.91								
27	0	0.1047	0.2143	0.4206	0.5868	0.7526	0.945	1.1344	1.3864	1.5977	1.7316	1.9078	2.1371
	12	2.367	2.5546	2.7448	2.9313								
28	0	0.0894	0.1825	0.3998	0.6591	0.8772	1.061	1.2473	1.4346	1.6295	1.8153	2.0179	2.1901
	12	2.3823	2.5739	2.7696	2.9508								
29	0	0.1469	0.265	0.4148	0.5709	0.7229	0.8836	1.0439	1.2254	1.4362	1.6677	1.891	2.1055
	12	2.3165	2.5299	2.7308	2.9283								
30	0	0.1466	0.3029	0.4734	0.6836	0.8659	1.0319	1.2126	1.4074	1.607	1.7848	1.9621	2.1615
	12	2.3571	2.5507	2.7377	2.9191								
31	0	0.2148	0.3644	0.5083	0.6722	0.8179	0.9982	1.2185	1.4305	1.6264	1.7124	1.8419	2.0895
	12	2.3567	2.5298	2.6668	2.8973								
32	0	0.1408	0.3465	0.5742	0.8145	1.0393	1.3051	1.5966	1.7797	1.868	1.9144	1.9772	2.0923
	12	2.1728	2.3451	2.6125	2.8943								
33	0	0.181	0.2719	0.3561	0.4793	0.5787	0.7693	1.0214	1.2967	1.5164	1.7555	1.9571	2.1246
	12	2.363	2.5547	2.7601	2.9338								
34	0	0.1934	0.4994	0.6707	0.7566	0.8364	0.9533	1.1562	1.3921	1.6029	1.8355	1.9614	2.0709
	12	2.2645	2.5047	2.737	2.9729								
35	0	0.1701	0.2909	0.4148	0.5672	0.814	1.0325	1.1829	1.3393	1.5111	1.7538	1.997	2.1951
	12	2.3752	2.5657	2.7662	2.9518								
36	0	0.0762	0.2387	0.5443	0.8263	1.0448	1.1382	1.2244	1.3371	1.4426	1.6842	1.944	2.1711

	12	2.3625	2.5499	2.736	2.9373								
37	0	0.1568	0.2748	0.3818	0.4837	0.6276	0.9163	1.114	1.312	1.5518	1.7845	1.9567	2.1421
	12	2.3632	2.5388	2.6956	2.8994								
38	0	0.2368	0.3748	0.4711	0.6639	0.866	1.0152	1.2318	1.4049	1.5219	1.7477	1.8985	2.0913
	12	2.3143	2.4996	2.717	2.9078								
39	0	0.1558	0.2846	0.4547	0.6045	0.7716	0.9491	1.1161	1.277	1.4479	1.6356	1.8184	2.0424
	12	2.2753	2.4823	2.6916	2.912								
40	0	0.2333	0.4319	0.5665	0.6616	0.8296	1.0427	1.2381	1.4532	1.6815	1.8473	1.9592	2.1432
	12	2.3665	2.5537	2.7621	2.9566								
41	0	0.3024	0.4882	0.5286	0.6027	0.6643	0.7168	0.8595	1.1256	1.4266	1.6768	1.8656	2.0911
	12	2.3152	2.5153	2.7375	2.9285								
42	0	0.1124	0.3568	0.5916	0.7111	0.8177	1.0378	1.2788	1.401	1.5086	1.7396	1.9989	2.1385
	12	2.3223	2.589	2.766	2.9026								
43	0	0.1268	0.2542	0.4072	0.5711	0.7454	0.9438	1.1316	1.3198	1.5381	1.7487	1.9393	2.1387
	12	2.3405	2.5456	2.7452	2.9378								
44	0	0.11	0.2438	0.4324	0.647	0.8202	0.9826	1.1743	1.3892	1.5927	1.7909	1.9812	2.1656
	12	2.3659	2.5611	2.7527	2.9407								
45	0	0.1424	0.2404	0.3476	0.442	0.6963	0.9066	1.082	1.3696	1.5773	1.7541	2.0053	2.1199
	12	2.2312	2.4567	2.7191	2.938								
46	0	0.1416	0.278	0.4405	0.6478	0.8161	0.971	1.1513	1.3192	1.514	1.7369	1.9457	2.1516
	12	2.3565	2.5588	2.7475	2.9327								
47	0	0.1563	0.3029	0.4735	0.6312	0.8047	0.9956	1.1632	1.324	1.507	1.6923	1.8995	2.1076
	12	2.2936	2.4959	2.6876	2.8966								
48	0	0.1667	0.3419	0.534	0.7884	0.9781	1.1135	1.3073	1.4418	1.5799	1.8039	1.9411	2.1291
	12	2.3535	2.523	2.7358	2.9354								
49	0	0.0406	0.1297	0.1974	0.3783	0.6488	0.9306	1.1257	1.3288	1.5534	1.7644	1.9642	2.1505
	12	2.3606	2.549	2.7482	2.934								
50	0	0.2318	0.4764	0.7455	1.012	1.1952	1.2853	1.3417	1.3922	1.4379	1.5405	1.7211	1.9709
	12	2.2399	2.4753	2.6961	2.9108								
51	0	0.1099	0.2357	0.3757	0.5511	0.7745	1.0008	1.1969	1.3998	1.6013	1.7981	1.9879	2.1902
	12	2.3836	2.5738	2.7635	2.9427								
52	0	0.1317	0.2802	0.4615	0.6367	0.8247	1.0309	1.2123	1.4016	1.5823	1.7786	1.9774	2.1687
	12	2.3717	2.5634	2.7545	2.9441								
53	0	0.1319	0.2571	0.3324	0.4622	0.7034	0.9574	1.1804	1.3854	1.5874	1.7835	1.9875	2.1747
	12	2.3712	2.5658	2.7622	2.9461								
54	0	0.2047	0.3408	0.4783	0.6721	0.8394	1.0386	1.1728	1.3676	1.5526	1.7178	1.9315	2.1545
	12	2.3484	2.531	2.7553	2.9537								
55	0	0.1383	0.2681	0.4284	0.6171	0.7786	0.947	1.1172	1.3177	1.5215	1.7018	1.8978	2.1059
	12	2.3036	2.5127	2.7219	2.9218								
56	0	0.1727	0.3907	0.6136	0.784	0.9781	1.184	1.3825	1.5826	1.7566	1.9413	2.1346	2.4395
	12	2.6602	2.8392	2.9475	3.0414								
57	0	0.1231	0.1999	0.2629	0.3337	0.5261	0.7983	1.0584	1.2847	1.5111	1.7366	1.9349	2.1423
	12	2.3388	2.5473	2.7408	2.9372								
58	0	0.318	0.6142	0.7857	0.8609	0.8906	0.9883	1.0574	1.1386	1.3217	1.6086	1.8574	2.0763
	12	2.301	2.5149	2.7282	2.9326								
59	0	0.0946	0.2095	0.39	0.5696	0.7139	0.8862	1.0974	1.3435	1.566	1.7622	1.9627	2.1729
	12	2.3718	2.5601	2.7528	2.938								
60	0	0.0896	0.1494	0.3091	0.5818	0.8294	1.0411	1.2174	1.4263	1.6073	1.8109	2.0017	2.1852
	12	2.3706	2.569	2.7599	2.9449								
61	0	0.1276	0.2342	0.3777	0.5208	0.6738	0.8724	1.0983	1.3236	1.5073	1.7062	1.9114	2.1258
	12	2.3373	2.5424	2.7419	2.9366								
62	0	0.1693	0.3278	0.5139	0.6904	0.8502	1.0241	1.2172	1.3789	1.5578	1.7395	1.9515	2.1422
	12	2.312	2.5109	2.7131	2.9321								
63	0	0.2349	0.3702	0.4659	0.6081	0.7895	0.9383	1.1532	1.3203	1.4794	1.7098	1.9221	2.1252
	12	2.3208	2.53	2.7428	2.9286								

Table C. 50 LSP codebook2 for MODE_VQ == 16_16

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.0092	0.0206	-0.0054	0.0039	-0.0107	-0.0078	-0.0203	-0.0119	-0.0068	0.0037	0.028	-0.0483
	12	-0.0104	-0.0382	-0.0663	-0.0019								
1	0	0.0515	0.0164	-0.0519	-0.0212	0.0284	-0.0179	-0.0725	-0.0673	-0.0105	0.0218	0.0181	0.0028
	12	0.0062	-0.0077	-0.017	-0.0129								
2	0	0.0163	-0.0066	0.0432	0.0056	0.0229	-0.0104	-0.0157	0.0035	-0.0183	-0.04	-0.021	-0.0099
	12	0.0355	0.0572	0.014	-0.0023								
3	0	-0.0107	0.025	0.0365	0.0354	0.0436	0.0325	-0.0088	-0.0491	-0.0718	-0.0995	-0.0841	-0.1445
	12	-0.1794	-0.1939	-0.1517	-0.0689								
4	0	0.0604	0.0688	0.0352	-0.0015	0.0641	0.1119	0.1171	0.125	0.1512	0.0893	-0.0053	-0.0904
	12	-0.0578	0.0234	0.0374	0.0114								
5	0	0.0236	0.0541	0.0885	0.0657	-0.0355	-0.006	0.0367	0.0358	0.0149	0.0043	0.0153	0.0329
	12	0.0151	0.0255	0.0372	0.0063								
6	0	-0.0407	-0.0372	0.0442	0.0696	-0.0268	-0.0234	-0.0009	-0.0039	0.0011	-0.0023	-0.01	0.0571
	12	0.0572	0.0296	-0.0235	-0.0407								
7	0	-0.0107	-0.05	-0.0655	-0.0323	-0.0449	-0.0704	-0.0607	-0.037	-0.0278	-0.016	0.0016	0.0001
	12	-0.0388	-0.0407	-0.0236	-0.0238								
8	0	-0.0024	0.0513	0.0143	-0.0318	-0.0393	-0.0229	-0.0056	-0.0154	-0.0339	-0.0482	-0.0347	0.1148
	12	0.1338	0.1208	0.0913	0.0407								
9	0	0.0476	0.0243	0.0031	0.0438	0.0138	0.0492	0.0492	0.0307	0.0293	0.0066	-0.0082	0.0482
	12	0.004	-0.0456	-0.0318	-0.0072								
10	0	-0.0135	-0.0241	-0.0099	-0.0218	0.0254	0.0018	-0.0059	-0.0029	0.0267	0.0159	-0.0175	-0.0333
	12	-0.0245	0.0025	-0.0305	-0.0425								
11	0	0.0004	-0.0216	-0.0348	0.0191	0.0297	0.0297	0.0318	0.0604	0.0927	0.101	0.1102	-0.0137
	12	-0.0026	0.0157	0.0495	0.0535								
12	0	0.024	0.0097	0.0008	-0.0445	0.0281	0.013	0.0223	0.037	0.0178	0.0156	0.0247	0.009
	12	0.0043	0.0042	0.0035	0.0058								
13	0	-0.0474	-0.0004	0.0133	-0.0003	0.0036	0.0201	0.0204	-0.0242	-0.0437	-0.0299	-0.0134	-0.0351
	12	-0.07	-0.0494	-0.0085	0.0096								

14	0	-0.0107	-0.0038	-0.0516	-0.0683	0.072	0.0717	0.0494	0.0085	-0.0308	-0.0324	-0.014	-0.019
	12	-0.0181	-0.0113	-0.0035	0.011								
15	0	-0.0541	-0.0733	-0.031	-0.0011	0.038	0.0331	0.013	-0.0262	-0.0202	0.0241	0.0376	0.0247
	12	0.0401	-0.0003	0.0004	0.0309								

Table C. 51 Alpha codebook for MODE_VQ == 16_16

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.4981	0.496	0.4951	0.4927	0.4956	0.4979	0.4999	0.4959	0.495	0.4929	0.4996	0.489
	12	0.4883	0.4896	0.4872	0.4794								
1	0	0.2915	0.2308	0.2038	0.2136	0.195	0.2094	0.1872	0.1871	0.1733	0.1785	0.1636	0.162
	12	0.1835	0.1842	0.2033	0.2416								

Table C. 52 LSP codebook 1 for MODE_VQ == 08_06

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.2702	0.5096	0.6437	0.7672	0.9639	1.0696	1.2625	1.5789	1.9285	2.2383	2.5129	2.847
1	0	0.174	0.3677	0.6082	0.8387	1.1084	1.3721	1.6362	1.8733	2.064	2.3442	2.6087	2.8548
2	0	0.1536	0.3279	0.5143	0.6859	0.9763	1.2744	1.5605	1.8566	2.1007	2.345	2.6075	2.885
3	0	0.2075	0.4533	0.7709	1.0377	1.2953	1.5132	1.7826	2.0351	2.259	2.4996	2.6795	2.8748
4	0	0.1393	0.2453	0.3754	0.5453	0.8148	1.1289	1.4389	1.7592	2.0353	2.3215	2.5934	2.8588
5	0	0.125	0.3627	0.7613	1.138	1.4163	1.5565	1.692	1.813	1.8678	2.0427	2.4318	2.8544
6	0	0.2256	0.4223	0.6452	0.8599	1.0673	1.3118	1.5486	1.8366	2.0759	2.3026	2.5284	2.803
7	0	0.2304	0.4404	0.6891	0.8964	1.151	1.4202	1.6483	1.858	2.1181	2.3686	2.6078	2.9128
8	0	0.223	0.3816	0.552	0.6062	0.7909	1.0988	1.433	1.7846	2.0713	2.3457	2.6048	2.8708
9	0	0.2447	0.58	0.8249	0.9905	1.1721	1.399	1.6694	1.9064	2.1307	2.4255	2.6815	2.9117
10	0	0.1974	0.3812	0.5802	0.7759	0.928	1.1547	1.417	1.6369	1.889	2.2587	2.5626	2.8239
11	0	0.1209	0.251	0.4841	0.8048	1.1197	1.3563	1.6073	1.8926	2.135	2.3669	2.6291	2.8985
12	0	0.2352	0.4347	0.6582	0.8178	0.9548	1.1654	1.4942	1.8812	2.1703	2.3779	2.6412	2.8871
13	0	0.2091	0.4084	0.673	0.9151	1.1259	1.3262	1.5937	1.8129	2.0237	2.3317	2.5778	2.862
14	0	0.1167	0.2406	0.452	0.7298	0.9848	1.2448	1.5137	1.7874	2.028	2.302	2.5914	2.8794
15	0	0.3003	0.4966	0.652	0.8505	1.16	1.3981	1.5805	1.8346	2.0757	2.3102	2.576	2.8499
16	0	0.2451	0.4163	0.596	0.7805	0.9507	1.2438	1.5587	1.8581	2.0735	2.3198	2.5704	2.822
17	0	0.3112	0.5517	0.7032	0.8528	1.1489	1.4257	1.6848	1.9388	2.1577	2.4265	2.6678	2.9051
18	0	0.2249	0.3897	0.5559	0.7473	1.0158	1.3581	1.6914	1.993	2.1843	2.3534	2.5512	2.8065
19	0	0.26	0.4574	0.7349	0.9691	1.1696	1.3848	1.6335	1.9021	2.1174	2.3481	2.5902	2.839
20	0	0.2246	0.3372	0.456	0.5249	0.7056	1.0273	1.381	1.7132	1.9819	2.2574	2.541	2.8491
21	0	0.1419	0.4834	0.8835	1.1453	1.2839	1.4224	1.5593	1.7877	2.1285	2.407	2.6043	2.8511
22	0	0.1886	0.3677	0.5617	0.8099	1.1277	1.3841	1.5804	1.8136	2.0307	2.2805	2.5399	2.8322
23	0	0.2351	0.4151	0.6675	0.8713	1.0464	1.3292	1.6586	1.9281	2.1355	2.3495	2.6222	2.8782
24	0	0.27	0.4489	0.6206	0.7121	0.7737	0.9848	1.3658	1.7433	2.0139	2.2243	2.4806	2.8175
25	0	0.2479	0.4425	0.649	0.8745	1.1161	1.3849	1.6773	1.9566	2.1491	2.3624	2.5685	2.8114
26	0	0.2035	0.3701	0.5567	0.7953	1.0082	1.2758	1.5373	1.7822	2.0175	2.2601	2.4759	2.7771
27	0	0.1856	0.3461	0.5998	0.9041	1.2383	1.4612	1.6667	1.9305	2.1617	2.4107	2.6477	2.8656
28	0	0.2107	0.3715	0.5289	0.6651	0.842	1.1168	1.4401	1.723	1.9901	2.2687	2.5452	2.8655
29	0	0.1218	0.2999	0.6348	0.9482	1.2745	1.5876	1.9129	2.2348	2.402	2.4922	2.6351	2.8357
30	0	0.1617	0.3483	0.5869	0.8163	1.0366	1.2344	1.4609	1.7029	1.9476	2.2337	2.5258	2.8442
31	0	0.2505	0.4894	0.751	0.9152	1.0845	1.3657	1.6528	1.8346	2.016	2.2811	2.5338	2.8136

Table C. 53 LSP codebook 2 for MODE_VQ == 08_06

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.0947	0.1158	0.0578	-0.0337	-0.0066	0.0104	-0.0447	-0.0505	-0.0778	-0.0293	0.0251	-0.0143
1	0	0.0349	-0.0227	-0.0909	0.0523	0.0325	-0.041	-0.1045	-0.0899	-0.0009	0.0075	-0.0575	-0.0855
2	0	-0.0129	0.0575	0.0597	0.0391	0.0371	-0.0184	-0.0083	0.0287	0.0143	0.0167	0.012	-0.0168
3	0	0.0452	0.0223	-0.0352	0.0119	-0.0496	-0.0965	-0.0661	-0.0072	0.1099	0.0843	-0.0087	-0.0478
4	0	-0.0128	-0.012	-0.0004	0.0731	0.1047	0.063	0.0196	-0.0103	-0.0399	-0.0986	-0.0912	-0.039
5	0	-0.0247	-0.0694	-0.0749	-0.0066	0.0223	0.0634	0.0343	-0.0134	0.0727	0.0241	0.0066	0.0437
6	0	0.061	0.0364	0.0248	-0.0358	-0.0686	-0.0104	0.0426	0.0088	-0.0137	-0.0165	0.0671	0.0815
7	0	-0.0863	-0.0644	-0.0088	0.0023	0.0482	0.1174	0.127	0.0594	0.0165	0.0949	0.1098	0.0137

Table C. 54 Alpha codebook for MODE_VQ == 08_06

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.4951	0.4999	0.4958	0.4907	0.4984	0.4965	0.4958	0.4996	0.4987	0.4958	0.4986	0.4977
1	0	0.2841	0.2186	0.1474	0.1687	0.2217	0.2632	0.2706	0.2624	0.2162	0.2453	0.246	0.2531

Table C. 55 LSP codebook 1 for MODE_VQ == SCL_1 / SCL_1_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.1600	0.2853	0.4529	0.6153	0.7685	0.9310	1.0801	1.2310	1.3901	1.5421	1.6828	1.8171
	12	1.9570	2.1078	2.2626	2.3974	2.5386	2.6880	2.8331	2.9771				
1	0	0.1311	0.2313	0.3753	0.5145	0.6446	0.7878	0.9300	1.0715	1.2347	1.4014	1.5635	1.7193
	12	1.8735	2.0350	2.1924	2.3463	2.4953	2.6491	2.7988	2.9617				
2	0	0.1626	0.2725	0.4151	0.5657	0.7191	0.8651	0.9996	1.1474	1.3066	1.4515	1.6174	1.7569
	12	1.8868	2.0495	2.2036	2.3519	2.4974	2.6522	2.7914	2.9528				
3	0	0.1493	0.2437	0.3873	0.5246	0.6803	0.8290	0.9227	1.0336	1.1451	1.3064	1.4951	1.6670
	12	1.8527	2.0466	2.1812	2.3573	2.5251	2.6618	2.8217	2.9682				
4	0	0.1474	0.2331	0.3627	0.4979	0.6173	0.7616	0.9015	1.0406	1.1980	1.3411	1.4930	1.6506
	12	1.8182	1.9984	2.1605	2.3267	2.4869	2.6485	2.8122	2.9719				

5	0	0.1293	0.2159	0.3510	0.4891	0.6321	0.7879	0.9037	1.0165	1.1660	1.3537	1.5252	1.6870
	12	1.8612	2.0293	2.1702	2.3172	2.4534	2.6068	2.7722	2.9476				
6	0	0.1595	0.2420	0.3701	0.5175	0.6480	0.7753	0.9248	1.0811	1.2361	1.3940	1.5230	1.6845
	12	1.8368	1.9967	2.1523	2.3006	2.4624	2.6073	2.7672	2.9495				
7	0	0.1431	0.2213	0.3574	0.4958	0.6235	0.7598	0.9103	1.0668	1.2251	1.3716	1.5392	1.6785
	12	1.8139	1.9663	2.1000	2.2732	2.4142	2.5779	2.7616	2.9389				
8	0	0.1476	0.2535	0.4041	0.5622	0.7207	0.8716	1.0196	1.1649	1.3208	1.4781	1.6343	1.7760
	12	1.9296	2.0866	2.2360	2.3776	2.5274	2.6806	2.8289	2.9808				
9	0	0.1553	0.3264	0.4215	0.5176	0.6459	0.7666	0.8294	0.9434	1.1682	1.4033	1.6004	1.7466
	12	1.8554	1.9827	2.1050	2.2835	2.4912	2.6575	2.7789	2.9436				
10	0	0.1590	0.2516	0.3819	0.5374	0.6929	0.8483	0.9927	1.1194	1.2616	1.4042	1.5488	1.7108
	12	1.8878	2.0739	2.2170	2.3494	2.5084	2.6597	2.8131	2.9688				
11	0	0.1232	0.2690	0.4656	0.5710	0.6440	0.7305	0.8820	1.0987	1.2817	1.4303	1.6053	1.7472
	12	1.8949	2.0730	2.2324	2.3426	2.4366	2.5816	2.7708	2.9533				
12	0	0.1396	0.2216	0.3652	0.5427	0.7096	0.8520	0.9746	1.1074	1.2769	1.4324	1.5691	1.6796
	12	1.8170	1.9940	2.1530	2.3286	2.4969	2.6578	2.8173	2.9737				
13	0	0.1238	0.2013	0.3417	0.4949	0.6180	0.7600	0.9152	1.0629	1.2395	1.4204	1.5741	1.6985
	12	1.8379	2.0131	2.1866	2.3735	2.5329	2.6734	2.8173	2.9673				
14	0	0.1398	0.2348	0.3796	0.5327	0.6642	0.8037	0.9488	1.0919	1.2664	1.4577	1.6060	1.7025
	12	1.8263	2.0102	2.1890	2.3588	2.5024	2.6419	2.8016	2.9665				
15	0	0.1068	0.1599	0.2538	0.4310	0.6308	0.7909	0.9421	1.1009	1.2628	1.4204	1.5789	1.7391
	12	1.8875	2.0504	2.1960	2.3406	2.4725	2.6070	2.7604	2.9455				
16	0	0.1832	0.2809	0.4140	0.5381	0.6481	0.7974	0.9713	1.1298	1.2729	1.4040	1.5464	1.6682
	12	1.7972	1.9957	2.1475	2.2841	2.4484	2.6201	2.7829	2.9472				
17	0	0.1810	0.2707	0.3869	0.5296	0.6750	0.8367	1.0141	1.1445	1.2703	1.4018	1.5140	1.6487
	12	1.7897	1.9403	2.1071	2.2768	2.4501	2.6387	2.8158	2.9892				
18	0	0.1349	0.3091	0.4717	0.5754	0.6622	0.8166	1.0225	1.1868	1.2898	1.4361	1.6508	1.8040
	12	1.8736	1.9793	2.1671	2.3733	2.4894	2.6191	2.7889	2.9919				
19	0	0.0940	0.2511	0.5032	0.7181	0.8163	0.9050	1.0048	1.0973	1.2800	1.4909	1.6043	1.6698
	12	1.7282	1.8536	2.0763	2.2837	2.4765	2.6550	2.8131	2.9517				
20	0	0.1622	0.2602	0.4020	0.5546	0.7130	0.8525	0.9949	1.1448	1.3070	1.4984	1.6371	1.7403
	12	1.8930	2.0402	2.1580	2.3223	2.4553	2.5999	2.7887	2.9841				
21	0	0.0748	0.1370	0.3140	0.5122	0.6833	0.8639	1.0101	1.1586	1.3200	1.4710	1.6296	1.7797
	12	1.9203	2.0737	2.1792	2.3169	2.4801	2.6487	2.8106	2.9704				
22	0	0.1878	0.2971	0.4205	0.5812	0.7555	0.9221	1.0350	1.2174	1.3957	1.5127	1.6567	1.7948
	12	1.8805	1.9784	2.1331	2.3418	2.4764	2.6143	2.7714	2.9596				
23	0	0.1673	0.2820	0.4025	0.5801	0.7340	0.8740	1.0063	1.1576	1.3701	1.5042	1.5957	1.6749
	12	1.7916	1.9757	2.1711	2.3563	2.4948	2.6065	2.7519	2.9425				
24	0	0.1299	0.2292	0.3623	0.5079	0.6517	0.8095	0.9621	1.1129	1.2782	1.4348	1.5901	1.7377
	12	1.8883	2.0485	2.2082	2.3834	2.5491	2.6992	2.8426	2.9826				
25	0	0.1409	0.2290	0.3641	0.5141	0.6406	0.7884	0.9505	1.1187	1.2630	1.3942	1.5467	1.6988
	12	1.8528	2.0081	2.1567	2.3205	2.4757	2.6295	2.7938	2.9636				
26	0	0.1290	0.2809	0.4604	0.5947	0.6930	0.8242	1.0036	1.2087	1.3567	1.4494	1.5850	1.7761
	12	1.9654	2.0743	2.1942	2.3577	2.5464	2.7084	2.8092	2.9460				
27	0	0.1324	0.2931	0.4733	0.6148	0.7241	0.8416	0.9660	1.1105	1.2606	1.4090	1.5714	1.7497
	12	1.8817	1.9819	2.1045	2.2801	2.4604	2.6309	2.8066	2.9774				
28	0	0.1372	0.2374	0.3716	0.5269	0.6837	0.8374	0.9879	1.1367	1.2994	1.4363	1.5710	1.7258
	12	1.8822	2.0353	2.1818	2.3424	2.4991	2.6525	2.8119	2.9726				
29	0	0.1493	0.2477	0.3864	0.5376	0.6568	0.8233	0.9883	1.1533	1.2792	1.4457	1.5940	1.7277
	12	1.8652	2.0275	2.1826	2.3426	2.4720	2.6107	2.7588	2.9146				
30	0	0.1557	0.2475	0.3795	0.5406	0.6780	0.8190	0.9897	1.1145	1.2848	1.4237	1.5511	1.7247
	12	1.8458	2.0159	2.1581	2.3078	2.4575	2.5995	2.7628	2.9312				
31	0	0.1461	0.2350	0.3937	0.5305	0.6675	0.8247	0.9801	1.1110	1.2424	1.3959	1.5665	1.6830
	12	1.8453	2.0165	2.1511	2.2960	2.4291	2.5612	2.7340	2.9465				
32	0	0.1676	0.2601	0.3986	0.5436	0.6963	0.8410	0.9628	1.0907	1.2356	1.3651	1.5072	1.6567
	12	1.8113	1.9890	2.1460	2.3000	2.4597	2.6307	2.8009	2.9669				
33	0	0.1336	0.2066	0.3178	0.4555	0.5766	0.7106	0.8431	0.9893	1.1591	1.3238	1.4904	1.6551
	12	1.8251	2.0020	2.1608	2.3237	2.4802	2.6408	2.8066	2.9669				
34	0	0.1316	0.2341	0.3653	0.5018	0.6286	0.7661	0.8983	1.0412	1.1983	1.3450	1.4957	1.6408
	12	1.7871	1.9488	2.1066	2.2778	2.4408	2.6090	2.7806	2.9519				
35	0	0.1151	0.2698	0.4836	0.6499	0.7337	0.7939	0.8521	0.9975	1.2260	1.4172	1.5632	1.7139
	12	1.8953	2.0646	2.2385	2.4008	2.5542	2.6774	2.7802	2.9354				
36	0	0.1645	0.2519	0.3709	0.5048	0.6839	0.8561	0.9715	1.0814	1.2060	1.3247	1.4541	1.5673
	12	1.7343	1.9571	2.1594	2.3409	2.4953	2.6532	2.8204	2.9866				
37	0	0.1608	0.2421	0.3839	0.5212	0.6201	0.7537	0.9398	1.0906	1.2263	1.3596	1.4982	1.6404
	12	1.7971	1.9706	2.1290	2.3020	2.4622	2.6157	2.7824	2.9623				
38	0	0.1597	0.2538	0.3639	0.4765	0.6109	0.7849	0.9660	1.1278	1.2818	1.4161	1.5431	1.6688
	12	1.7998	1.9342	2.0862	2.2909	2.4921	2.6727	2.8274	2.9784				
39	0	0.1449	0.2343	0.3646	0.5062	0.6556	0.7955	0.9063	1.0166	1.1550	1.3188	1.4840	1.6261
	12	1.7892	1.9719	2.1359	2.3064	2.4693	2.6340	2.7992	2.9640				
40	0	0.1242	0.2372	0.3941	0.5418	0.6858	0.8382	0.9859	1.1367	1.2999	1.4569	1.6135	1.7601
	12	1.9074	2.0659	2.2153	2.3653	2.5163	2.6710	2.8252	2.9783				
41	0	0.1623	0.3048	0.4217	0.5375	0.7062	0.8681	0.9709	1.1149	1.3106	1.4118	1.5577	1.7586
	12	1.8617	1.9704	2.1548	2.3371	2.4537	2.6376	2.7937	2.9351				
42	0	0.1345	0.2221	0.3493	0.4831	0.5933	0.7273	0.9027	1.0707	1.2309	1.3868	1.5404	1.7006
	12	1.8646	2.0339	2.1908	2.3427	2.4957	2.6564	2.8094	2.9678				
43	0	0.1612	0.2705	0.3986	0.5224	0.6693	0.8146	0.9306	1.0929	1.2422	1.3767	1.5461	1.6972
	12	1.8477	2.0211	2.1667	2.3302	2.4884	2.6428	2.8144	2.9661				
44	0	0.1526	0.2467	0.3861	0.5132	0.6529	0.8272	0.9693	1.1085	1.2831	1.4246	1.5400	1.6488
	12	1.8318	2.0680	2.2510	2.3853	2.5226	2.6607	2.7924	2.9447				
45	0	0.1537	0.2320	0.3469	0.4620	0.5734	0.7114	0.8469	1.0315	1.2316	1.3742	1.5322	1.6921
	12	1.8488	2.0166	2.1767	2.3374	2.4915	2.6473	2.7995	2.9671				
46	0	0.1623	0.2450	0.3823	0.5231	0.6524	0.8039	0.9610	1.0981	1.2552	1.3900	1.5090	1.6286
	12	1.7729	1.9704	2.1583	2.3362	2.4960	2.6544	2.8174	2.9718				
47	0	0.1563	0.2336	0.3827	0.5439	0.6891	0.8404	0.9753	1.0994	1.2370	1.3519	1.4879	1.6292
	12	1.8057	2.0218	2.2016	2.3665	2.5317	2.6632	2.8038	2.9691				
48	0	0.1143	0.2132	0.3858	0.5585	0.7518	0.9598	1.1207	1.1929	1.2733	1.3541	1.5055	1.7167
	12	1.8987	2.0617	2.1982	2.3564	2.4912	2.6271	2.8013	2.9594				
49	0	0.1390	0.2200	0.3450	0.4919	0.6389	0.8030	0.9710	1.0921	1.2266	1.3882	1.5420	1.6846
	12	1.8269	1.9835	2.1468	2.3166	2.4840	2.6592	2.8188	2.9684				
50	0	0.1607	0.2613	0.4012	0.5418	0.6631	0.8204	0.9635	1.1147	1.2491	1.3917	1.5350	1.6828
	12	1.8192	1.9534	2.1161	2.2836	2.4354	2.5966	2.7644	2.9430				
51	0	0.1646	0.2447	0.3965	0.5576	0.6979	0.8247	0.9533	1.0823	1.2231	1.3502	1.4885	1.6285
	12	1.7634	1.9331	2.0920	2.2692	2.4408	2.6108	2.7899	2.9606				
52	0	0.1534	0.2564	0.3963	0.5530	0.7137	0.8622	0.9979	1.1262	1.2669	1.4196	1.5645	1.7051
	12	1.8569	2.0181	2.1680	2.3280	2.4857	2.6389	2.7975	2.9615				
53	0	0.1803	0.3022	0.4398	0.5698	0.7035	0.8586	1.0070	1.1659	1.3331	1.4976	1.6681	1.7957
	12	1.8906	2.0198	2.1861	2.3577	2.5032	2.6583	2.8095	2.9673				
54	0	0.1693	0.3147	0.4721	0.6048	0.7297	0.8774	1.0264	1.1928	1.3699	1.5193	1.6181	1.7357
	12	1.9164	2.0822	2.2169	2.3358	2.4707	2.6512	2.8224	2.9731				

55	0	0.1833	0.2771	0.4024	0.5568	0.6948	0.8435	0.9974	1.1540	1.2893	1.4209	1.5671	1.6961
	12	1.8325	1.9939	2.1416	2.3022	2.4723	2.6306	2.7952	2.9630				
56	0	0.1145	0.1965	0.3075	0.4455	0.5985	0.7713	0.9445	1.1128	1.2836	1.4372	1.5892	1.7373
	12	1.8926	2.0605	2.2184	2.3669	2.5191	2.6707	2.8225	2.9813				
57	0	0.1279	0.1990	0.2855	0.3970	0.5413	0.7220	0.8854	1.0340	1.2057	1.3782	1.5509	1.7123
	12	1.8676	2.0283	2.1786	2.3358	2.4927	2.6523	2.8072	2.9642				
58	0	0.0913	0.2172	0.4557	0.6852	0.8239	0.8850	0.9634	1.0408	1.1908	1.4135	1.6060	1.7364
	12	1.8508	2.0287	2.2013	2.3546	2.5112	2.6860	2.8659	3.0248				
59	0	0.1719	0.2537	0.3798	0.5416	0.6940	0.8447	0.9855	1.1166	1.2709	1.4059	1.5353	1.6581
	12	1.7847	1.9348	2.1059	2.3268	2.5201	2.6761	2.8346	2.9773				
60	0	0.1639	0.2527	0.3919	0.5351	0.6801	0.8325	0.9935	1.1284	1.2756	1.4464	1.6179	1.7568
	12	1.8913	2.0237	2.1412	2.2742	2.4274	2.6057	2.7607	2.9415				
61	0	0.1548	0.2268	0.3458	0.4743	0.6349	0.8062	0.9537	1.0838	1.2643	1.4077	1.5858	1.7483
	12	1.8580	1.9528	2.1046	2.2874	2.4341	2.6129	2.7767	2.9507				
62	0	0.1578	0.2562	0.3954	0.5694	0.7586	0.9133	1.0197	1.1212	1.2796	1.4613	1.6514	1.7974
	12	1.9376	2.0514	2.1639	2.3285	2.4962	2.6754	2.8451	2.9942				
63	0	0.1518	0.2452	0.3583	0.4776	0.6280	0.7965	0.9540	1.1027	1.2562	1.4434	1.5565	1.7118
	12	1.8782	2.0373	2.1546	2.2556	2.4371	2.6103	2.7877	2.9749				

Table C. 56 LSP codebook 2 for MODE_VQ == SCL_1 / SCL_1_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.0402	0.0621	0.0216	0.0119	0.0219	0.0036	-0.0009	0.0004	0.0061	0.0062	-0.0177	-0.0227
	12	0.0007	0.0104	-0.0099	0.0036	0.0049	0.0073	0.0144	0.0008				
1	0	-0.0096	0.0005	-0.0312	-0.0435	-0.0161	0.0233	0.0387	0.0409	0.0112	-0.0170	-0.0293	-0.0125
	12	0.0123	0.0099	0.0537	0.0659	0.0550	0.0379	0.0274	0.0176				
2	0	-0.0413	-0.0458	-0.0523	-0.0292	0.0019	0.0080	0.0119	0.0081	-0.0010	-0.0042	-0.0001	0.0320
	12	0.0611	0.0397	-0.0391	-0.0401	-0.0136	-0.0032	-0.0106	-0.0115				
3	0	-0.0023	-0.0006	-0.0209	0.0011	0.0315	0.0379	0.0439	0.0428	0.0335	0.0296	0.0233	0.0256
	12	0.0273	0.0157	-0.0206	-0.0279	0.0045	0.0412	0.0527	0.0240				
4	0	0.0038	0.0122	0.0092	0.0118	-0.0088	0.0229	0.0413	0.0513	0.0708	0.0933	0.0892	0.0849
	12	0.0833	0.0668	0.0181	-0.0216	-0.0230	0.0108	0.0146	0.0117				
5	0	-0.0215	0.0276	0.0163	-0.0001	0.0256	0.0201	0.0356	0.0432	0.0401	0.0302	0.0056	-0.0240
	12	-0.0319	-0.0183	0.0008	-0.0203	-0.0359	-0.0375	-0.0168	-0.0012				
6	0	-0.0264	0.0121	0.0159	-0.0250	-0.0342	-0.0196	-0.0044	0.0028	-0.0120	-0.0079	0.0050	0.0173
	12	0.0130	0.0024	-0.0218	-0.0177	-0.0064	-0.0103	0.0190	0.0414				
7	0	0.0078	0.0235	0.0229	0.0391	0.0464	0.0007	0.0114	0.0246	0.0221	0.0242	0.0137	0.0109
	12	0.0213	0.0197	0.0363	0.0627	0.0547	0.0010	-0.0405	-0.0256				
8	0	-0.0435	-0.0158	-0.0058	-0.0084	-0.0014	0.0132	0.0120	0.0039	-0.0143	-0.0293	-0.0314	-0.0374
	12	-0.0374	-0.0208	-0.0075	0.0122	0.0155	-0.0245	-0.0203	0.0065				
9	0	-0.0307	0.0083	0.0493	0.0738	0.0477	0.0368	0.0332	0.0080	-0.0121	-0.0037	0.0135	0.0199
	12	0.0133	-0.0018	0.0213	0.0352	0.0562	0.0750	0.0748	0.0404				
10	0	-0.0070	0.0334	0.0528	0.0379	0.0101	0.0122	0.0115	-0.0052	-0.0362	-0.0516	-0.0418	-0.0005
	12	0.0271	0.0128	0.0315	0.0166	-0.0093	-0.0087	-0.0223	-0.0270				
11	0	-0.0390	0.0005	0.0356	0.0259	0.0009	0.0047	-0.0038	0.0069	0.0283	0.0408	0.0452	0.0397
	12	0.0093	-0.0124	0.0073	0.0184	0.0284	0.0258	0.0405	0.0352				
12	0	-0.0465	-0.0337	-0.0030	0.0326	0.0367	0.0179	-0.0146	-0.0341	-0.0195	0.0198	0.0243	0.0023
	12	0.0006	-0.0020	0.0214	0.0226	0.0213	0.0110	0.0091	0.0112				
13	0	0.0235	0.0468	-0.0014	-0.0375	-0.0166	-0.0230	-0.0275	-0.0189	-0.0228	-0.0263	-0.0302	-0.0197
	12	-0.0086	-0.0083	-0.0178	0.0124	0.0543	0.0519	0.0104	-0.0128				
14	0	0.0230	0.0732	0.0728	0.0564	0.0329	-0.0282	-0.0427	-0.0315	-0.0164	-0.0032	-0.0001	0.0184
	12	0.0348	0.0268	0.0497	0.0314	-0.0126	-0.0117	0.0272	0.0482				
15	0	0.0111	0.0490	0.0525	0.0055	-0.0235	0.0023	0.0134	0.0126	0.0005	0.0015	-0.0094	0.0109
	12	-0.0129	-0.0383	0.0344	0.0146	0.0154	0.0531	0.0250	-0.0049				

Table C. 57 Alpha codebook for MODE_VQ == SCL_1 / SCL_1_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.4978	0.4938	0.4978	0.4961	0.4999	0.4985	0.4970	0.4990	0.4977	0.4971	0.4977	0.4995
	12	0.4969	0.4998	0.4996	0.4996	0.4975	0.4991	0.4973	0.4897				
1	0	0.3628	0.3161	0.3320	0.3266	0.2807	0.2562	0.2555	0.2356	0.2270	0.2703	0.2707	0.2485
	12	0.2382	0.2114	0.2519	0.2628	0.2706	0.2995	0.3124	0.3111				

Table C. 58 LSP codebook1 for MODE_VQ == SCL_2 / SCL_2_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.1523	0.2837	0.4357	0.5877	0.7613	0.9090	1.0559	1.2101	1.3694	1.5101	1.6485	1.7853
	12	1.9376	2.1028	2.2282	2.3687	2.5205	2.6605	2.8111	2.9757				
1	0	0.1431	0.2454	0.3740	0.5255	0.7074	0.8569	0.9726	1.1341	1.3156	1.4682	1.5857	1.7082
	12	1.8785	2.0722	2.2101	2.3599	2.5281	2.6621	2.8002	2.9561				
2	0	0.1533	0.2722	0.4154	0.5598	0.7140	0.8461	0.9803	1.1427	1.2862	1.4354	1.5826	1.7251
	12	1.8904	2.0607	2.1890	2.3348	2.4953	2.6316	2.7950	2.9703				
3	0	0.1416	0.2569	0.4138	0.5756	0.7367	0.8693	0.9889	1.1350	1.2952	1.4582	1.6227	1.7291
	12	1.8388	2.0191	2.1979	2.3908	2.5512	2.6601	2.7801	2.9354				
4	0	0.1271	0.2010	0.3444	0.5007	0.6130	0.7469	0.8938	1.0651	1.2291	1.3815	1.5476	1.7130
	12	1.8773	2.0473	2.1765	2.3324	2.4981	2.6501	2.8033	2.9639				
5	0	0.1332	0.2302	0.3415	0.4761	0.6371	0.7870	0.9585	1.1347	1.3023	1.4334	1.5640	1.7105
	12	1.8886	2.0816	2.2445	2.3934	2.5351	2.6707	2.8191	2.9692				
6	0	0.1508	0.2772	0.4245	0.5458	0.6838	0.8156	0.9430	1.1005	1.2725	1.4522	1.6331	1.7748
	12	1.8861	2.0052	2.1425	2.3100	2.4852	2.6417	2.8094	2.9768				
7	0	0.1533	0.2645	0.4000	0.5440	0.6789	0.8126	0.9453	1.0922	1.2571	1.4050	1.5702	1.7327
	12	1.8946	2.0645	2.2010	2.3558	2.5239	2.6736	2.8163	2.9609				
8	0	0.1346	0.2449	0.3899	0.5419	0.7196	0.8763	1.0208	1.1736	1.3270	1.4725	1.6264	1.7682
	12	1.9205	2.0872	2.2281	2.3726	2.5310	2.6774	2.8267	2.9819				
9	0	0.1232	0.1971	0.3045	0.4780	0.6851	0.8427	0.9450	1.0737	1.2353	1.3939	1.5635	1.7152
	12	1.8754	2.0398	2.1927	2.3502	2.5028	2.6420	2.7944	2.9685				
10	0	0.1213	0.2307	0.3709	0.5099	0.6692	0.8150	0.9588	1.1094	1.2627	1.4101	1.5744	1.7275
	12	1.8810	2.0422	2.1895	2.3415	2.5012	2.6514	2.8084	2.9786				
11	0	0.1276	0.2173	0.3588	0.5155	0.6968	0.8561	0.9864	1.1173	1.2717	1.4363	1.6088	1.7682
	12	1.8908	2.0165	2.1761	2.3482	2.5118	2.6584	2.8015	2.9633				
12	0	0.1095	0.2144	0.3840	0.5247	0.7013	0.8436	0.9573	1.1291	1.3046	1.4735	1.6414	1.7288
	12	1.8443	2.0454	2.1724	2.3355	2.5175	2.6378	2.8054	2.9853				
13	0	0.0828	0.2282	0.3366	0.4174	0.5818	0.7642	0.9332	1.0731	1.2841	1.4610	1.5720	1.6531
	12	1.8525	2.0113	2.1434	2.2819	2.4383	2.6070	2.7616	2.9604				
14	0	0.1491	0.2523	0.3750	0.5109	0.6352	0.7790	0.9168	1.0522	1.2248	1.4226	1.6303	1.7518
	12	1.8567	1.9915	2.1338	2.3102	2.4762	2.6155	2.7858	2.9638				
15	0	0.1055	0.2159	0.3503	0.4775	0.6381	0.7788	0.9258	1.0928	1.2685	1.4445	1.6061	1.7152
	12	1.8511	2.0129	2.1667	2.3275	2.4862	2.6241	2.7920	2.9748				
16	0	0.1664	0.2878	0.3915	0.5076	0.6702	0.8286	0.9963	1.1645	1.3445	1.4848	1.5798	1.6633
	12	1.7819	1.9672	2.1477	2.3049	2.4614	2.6188	2.7882	2.9675				
17	0	0.1156	0.2156	0.3693	0.5001	0.6693	0.8323	0.9909	1.1562	1.2946	1.4228	1.5849	1.7388
	12	1.8949	2.0570	2.2007	2.3626	2.4688	2.5622	2.7345	2.9432				
18	0	0.1561	0.2831	0.4384	0.5660	0.7065	0.8485	1.0222	1.1725	1.2907	1.4407	1.6061	1.7674
	12	1.9072	2.0538	2.1951	2.3533	2.5179	2.6456	2.8004	2.9787				
19	0	0.1672	0.2400	0.3577	0.5502	0.6555	0.8288	0.9801	1.1000	1.2991	1.4353	1.6265	1.7390
	12	1.8759	2.0554	2.1403	2.3091	2.4650	2.6113	2.7827	2.9258				
20	0	0.1367	0.2419	0.4001	0.5526	0.7334	0.8861	1.0310	1.1929	1.3571	1.5020	1.6319	1.7356
	12	1.8562	2.0164	2.1624	2.3179	2.4890	2.6422	2.7968	2.9698				
21	0	0.1165	0.2199	0.3671	0.5277	0.7169	0.8821	1.0316	1.1915	1.3495	1.4974	1.6649	1.8175
	12	1.9511	2.0838	2.1973	2.3382	2.4897	2.6338	2.7848	2.9548				
22	0	0.1615	0.2983	0.4487	0.5783	0.7184	0.8538	1.0048	1.1780	1.3461	1.4766	1.5902	1.7276
	12	1.9010	2.0805	2.2277	2.3579	2.5046	2.6462	2.8119	2.9831				
23	0	0.1296	0.2472	0.3818	0.5315	0.7078	0.8625	1.0178	1.1769	1.3337	1.4648	1.5985	1.7410
	12	1.8938	2.0472	2.1941	2.3400	2.4816	2.6192	2.7771	2.9545				
24	0	0.1489	0.2663	0.4075	0.5436	0.6847	0.8269	0.9876	1.1579	1.3283	1.4827	1.6432	1.7873
	12	1.9365	2.0968	2.2403	2.3826	2.5315	2.6651	2.8101	2.9683				

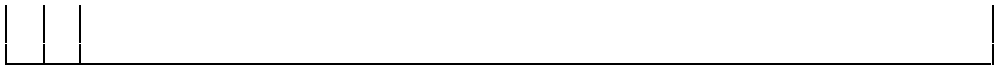
25	0	0.1199	0.2019	0.3015	0.4399	0.6471	0.8518	1.0326	1.1813	1.3230	1.4560	1.6025	1.7501
	12	1.9000	2.0594	2.1961	2.3592	2.5233	2.6709	2.8207	2.9692				
26	0	0.1408	0.2344	0.3638	0.5010	0.6262	0.7741	0.9564	1.1276	1.2824	1.4283	1.5867	1.7259
	12	1.8799	2.0456	2.1843	2.3374	2.4989	2.6476	2.8047	2.9728				
27	0	0.1349	0.2446	0.3966	0.5444	0.7464	0.9175	1.0350	1.1376	1.2645	1.4263	1.5937	1.7867
	12	1.9910	2.1240	2.2064	2.3154	2.4477	2.6108	2.8122	2.9837				
28	0	0.1212	0.2168	0.3548	0.5201	0.6840	0.8343	1.0017	1.1600	1.3177	1.4400	1.5544	1.7005
	12	1.8757	2.0408	2.1715	2.3242	2.4896	2.6379	2.8025	2.9646				
29	0	0.1274	0.2551	0.4168	0.5895	0.7597	0.9068	1.0467	1.1717	1.3105	1.4718	1.6275	1.7497
	12	1.9218	2.1261	2.2609	2.3663	2.4904	2.6011	2.7352	2.9314				
30	0	0.1806	0.2689	0.3756	0.5610	0.7532	0.8565	0.9895	1.1557	1.2956	1.4804	1.5686	1.7154
	12	1.9344	2.0742	2.1388	2.2875	2.5078	2.6325	2.7978	2.9981				
31	0	0.1555	0.2728	0.4055	0.5360	0.6931	0.8577	1.0247	1.1761	1.3073	1.4101	1.5140	1.6635
	12	1.8620	2.0568	2.2261	2.3884	2.5104	2.6312	2.7968	2.9627				
32	0	0.1787	0.2579	0.3639	0.5222	0.7634	0.8460	0.9073	1.0834	1.2470	1.4324	1.5580	1.6513
	12	1.8661	2.0119	2.0971	2.2621	2.4856	2.6214	2.7682	2.9668				
33	0	0.1399	0.2480	0.3634	0.4563	0.6072	0.7452	0.8844	1.0356	1.1950	1.3522	1.5228	1.6488
	12	1.8079	1.9986	2.1568	2.3215	2.4692	2.6286	2.7817	2.9555				
34	0	0.1255	0.2830	0.4070	0.5534	0.6971	0.7987	0.8829	1.0622	1.2092	1.3263	1.5020	1.6894
	12	1.8564	2.0160	2.1714	2.3279	2.4925	2.6367	2.8145	2.9730				
35	0	0.1262	0.2665	0.4443	0.5986	0.7730	0.9104	1.0049	1.1013	1.2371	1.3996	1.5571	1.7033
	12	1.8627	2.0353	2.1978	2.3653	2.5304	2.6841	2.8347	2.9802				
36	0	0.1518	0.2994	0.4241	0.5769	0.7453	0.8559	1.0390	1.1965	1.3388	1.5038	1.6048	1.6716
	12	1.8185	2.0474	2.2268	2.4127	2.5792	2.6499	2.7199	2.9130				
37	0	0.1325	0.2337	0.3708	0.5076	0.6568	0.8035	0.9305	1.0812	1.2502	1.4068	1.5522	1.6935
	12	1.8214	1.9806	2.1779	2.3511	2.4893	2.6122	2.7643	2.9502				
38	0	0.1327	0.2307	0.3812	0.4790	0.7169	0.8727	0.9721	1.0720	1.2225	1.3758	1.5115	1.5792
	12	1.7485	1.9383	2.1041	2.3080	2.4878	2.6324	2.8006	2.9943				
39	0	0.1389	0.2465	0.3758	0.5238	0.6709	0.8239	0.9236	1.0892	1.2345	1.3872	1.5453	1.6811
	12	1.8148	2.0105	2.1435	2.2738	2.4751	2.6162	2.7797	2.9668				
40	0	0.1304	0.2430	0.3916	0.5411	0.6881	0.8255	0.9833	1.1468	1.3079	1.4467	1.5930	1.7427
	12	1.9010	2.0620	2.2045	2.3562	2.5188	2.6676	2.8236	2.9798				
41	0	0.1720	0.2652	0.3683	0.5308	0.7088	0.8140	0.9659	1.1484	1.2526	1.4115	1.6162	1.7243
	12	1.8797	2.0840	2.1899	2.3374	2.5396	2.6449	2.7953	2.9999				
42	0	0.1529	0.2678	0.4005	0.5329	0.6787	0.8046	0.9404	1.1075	1.2690	1.4106	1.5488	1.6899
	12	1.8563	2.0229	2.1720	2.3397	2.4940	2.6361	2.8035	2.9793				
43	0	0.1601	0.2904	0.4424	0.5397	0.6637	0.7944	0.9407	1.0770	1.2499	1.4151	1.5433	1.6928
	12	1.8769	2.0540	2.2138	2.3695	2.5168	2.6419	2.8069	2.9914				
44	0	0.1392	0.2465	0.3812	0.5220	0.6662	0.7993	0.9549	1.1306	1.2939	1.4441	1.5618	1.6611
	12	1.8186	2.0169	2.1854	2.3551	2.5188	2.6536	2.8007	2.9683				
45	0	0.1476	0.2413	0.3589	0.5185	0.6425	0.7617	0.9361	1.1129	1.2932	1.3957	1.5101	1.6746
	12	1.8123	2.0014	2.1534	2.3326	2.5034	2.6397	2.7982	2.9420				
46	0	0.1721	0.2882	0.3889	0.5031	0.6602	0.7977	0.9414	1.0986	1.2551	1.3785	1.5000	1.6111
	12	1.7403	1.9419	2.1197	2.2686	2.4399	2.6039	2.7725	2.9609				
47	0	0.1259	0.2486	0.3917	0.5110	0.6512	0.7823	0.9303	1.0832	1.2359	1.3720	1.5263	1.6819
	12	1.8403	2.0014	2.1578	2.3176	2.4679	2.6223	2.7844	2.9602				
48	0	0.1301	0.2339	0.3712	0.5209	0.6757	0.8179	0.9938	1.1448	1.2521	1.3689	1.5364	1.7024
	12	1.8480	2.0188	2.1564	2.3212	2.4615	2.5885	2.7723	2.9664				
49	0	0.1565	0.2178	0.3969	0.5370	0.6568	0.8377	0.9875	1.0841	1.3003	1.4273	1.5222	1.7344
	12	1.8531	1.9769	2.1774	2.2828	2.4358	2.6472	2.7530	2.9445				
50	0	0.1501	0.2669	0.4310	0.5590	0.7191	0.8660	0.9689	1.0861	1.1966	1.3208	1.4866	1.6466
	12	1.8192	1.9919	2.1072	2.2996	2.4930	2.6108	2.7391	2.9295				
51	0	0.1349	0.2580	0.4036	0.5575	0.7196	0.8609	0.9949	1.1307	1.2833	1.4672	1.5986	1.6793
	12	1.7542	1.8858	2.0899	2.2949	2.4773	2.6183	2.7698	2.9450				
52	0	0.1368	0.2554	0.4120	0.5659	0.7328	0.8819	1.0135	1.1505	1.3040	1.4446	1.5915	1.7386
	12	1.8849	2.0404	2.1863	2.3395	2.5018	2.6542	2.8159	2.9785				
53	0	0.1296	0.2331	0.3711	0.5116	0.6679	0.8223	0.9894	1.1736	1.3558	1.5168	1.6607	1.7807
	12	1.9073	2.0482	2.1792	2.3251	2.4902	2.6426	2.8088	2.9715				
54	0	0.1623	0.3074	0.4617	0.6001	0.7584	0.8940	1.0487	1.1999	1.3359	1.4741	1.6203	1.7634
	12	1.9077	2.0572	2.1764	2.3388	2.5178	2.6740	2.8185	2.9660				
55	0	0.1476	0.2658	0.4086	0.5443	0.6923	0.8383	0.9767	1.1238	1.2727	1.4137	1.5623	1.7054
	12	1.8477	2.0083	2.1545	2.3063	2.4647	2.6186	2.7835	2.9601				
56	0	0.1439	0.2520	0.3874	0.5335	0.7037	0.8415	0.9877	1.1403	1.2737	1.4244	1.5414	1.6614
	12	1.8383	2.0083	2.1459	2.3100	2.4826	2.6362	2.8009	2.9692				
57	0	0.1438	0.2239	0.3343	0.4757	0.6725	0.8664	1.0279	1.1630	1.3017	1.4161	1.5500	1.7055
	12	1.8320	1.9701	2.1225	2.3134	2.4971	2.6667	2.8169	2.9591				
58	0	0.1506	0.2996	0.4575	0.5980	0.7504	0.9000	1.0290	1.1749	1.3083	1.4294	1.5626	1.6851
	12	1.8225	1.9879	2.1414	2.3147	2.4992	2.6567	2.8160	2.9763				
59	0	0.1231	0.2489	0.3965	0.5319	0.6688	0.8162	0.9734	1.1061	1.2338	1.3925	1.5457	1.6730
	12	1.7885	1.9433	2.1301	2.2962	2.4871	2.6767	2.8617	2.9909				
60	0	0.1208	0.2495	0.3985	0.5319	0.7076	0.8447	0.9838	1.1646	1.3153	1.4274	1.5779	1.7244
	12	1.8672	1.9994	2.1341	2.3014	2.4607	2.6150	2.7891	2.9638				
61	0	0.1511	0.3248	0.3785	0.4768	0.7136	0.8969	0.9928	1.0700	1.3184	1.5435	1.6492	1.6919
	12	1.8652	2.0819	2.1728	2.2753	2.4286	2.6350	2.7922	2.9329				
62	0	0.1487	0.2761	0.4188	0.5617	0.7444	0.8940	1.0423	1.1923	1.3545	1.4961	1.6338	1.7939
	12	1.9039	1.9968	2.1231	2.3049	2.4889	2.6303	2.7839	2.9667				
63	0	0.1498	0.2533	0.3880	0.5244	0.6823	0.8304	0.9882	1.1430	1.2991	1.4740	1.6173	1.7078
	12	1.8289	2.0012	2.1388	2.2837	2.4630	2.6277	2.7979	2.9736				

Table C. 59 LSP codebook2 for MODE_VQ == SCL_2 / SCL_2_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.0430	0.0479	0.0095	0.0184	0.0130	0.0220	0.0057	-0.0102	-0.0041	0.0018	-0.0053	-0.0033
	12	-0.0001	-0.0056	-0.0047	0.0081	0.0063	0.0062	0.0141	0.0133				
1	0	0.0076	0.0076	-0.0293	-0.0292	-0.0147	0.0378	0.0530	0.0349	-0.0009	-0.0182	-0.0217	-0.0069
	12	0.0071	0.0045	0.0521	0.0580	0.0399	0.0375	0.0237	0.0044				
2	0	-0.0368	-0.0471	-0.0489	-0.0227	-0.0118	0.0094	0.0143	0.0065	-0.0081	-0.0055	0.0050	0.0380
	12	0.0604	0.0358	-0.0339	-0.0370	-0.0144	-0.0048	0.0039	0.0137				
3	0	-0.0005	-0.0180	-0.0356	0.0032	0.0240	0.0240	0.0474	0.0630	0.0573	0.0406	0.0219	0.0186
	12	0.0237	0.0190	-0.0285	-0.0254	0.0039	0.0421	0.0401	0.0152				
4	0	-0.0019	0.0149	0.0131	0.0107	-0.0142	0.0151	0.0237	0.0287	0.0354	0.0495	0.0538	0.0593
	12	0.0585	0.0397	0.0068	-0.0159	-0.0198	0.0230	0.0233	-0.0081				
5	0	-0.0053	0.0158	0.0060	0.0178	0.0222	0.0255	0.0331	0.0246	0.0248	0.0295	0.0084	-0.0117
	12	-0.0276	-0.0252	0.0012	-0.0376	-0.0647	-0.0346	-0.0076	0.0069				
6	0	-0.0173	-0.0045	0.0043	-0.0181	-0.0431	-0.0117	-0.0086	-0.0129	-0.0226	-0.0138	-0.0021	0.0104
	12	0.0076	-0.0077	0.0064	-0.0202	-0.0367	0.0036	0.0456	0.0429				
7	0	0.0124	0.0346	0.0301	0.0440	0.0317	-0.0045	0.0034	0.0123	0.0158	0.0223	0.0120	0.0101
	12	0.0232	0.0214	0.0222	0.0554	0.0412	-0.0030	-0.0213	-0.0025				
8	0	-0.0354	-0.0083	-0.0099	-0.0066	-0.0035	0.0120	0.0136	-0.0052	-0.0311	-0.0289	-0.0276	-0.0275
	12	-0.0252	-0.0197	-0.0113	-0.0032	-0.0036	-0.0091	-0.0156	-0.0158				
9	0	-0.0232	0.0062	0.0406	0.0714	0.0442	0.0333	0.0351	0.0166	0.0035	0.0112	0.0205	0.0317
	12	0.0211	0.0009	0.0384	0.0476	0.0492	0.0629	0.0604	0.0325				
10	0	-0.0038	0.0311	0.0493	0.0396	0.0005	0.0064	0.0052	-0.0088	-0.0335	-0.0330	-0.0285	-0.0026
	12	0.0259	0.0197	0.0347	0.0166	-0.0217	-0.0232	-0.0152	-0.0052				
11	0	-0.0393	-0.0056	0.0307	0.0286	-0.0013	-0.0043	-0.0034	0.0064	0.0243	0.0434	0.0413	0.0284
	12	0.0038	-0.0152	-0.0006	0.0088	0.0207	0.0363	0.0605	0.0464				
12	0	-0.0425	-0.0428	-0.0135	0.0378	0.0395	0.0210	-0.0004	-0.0287	-0.0243	0.0102	0.0356	0.0374
	12	0.0163	-0.0041	0.0243	0.0224	0.0117	0.0206	0.0059	-0.0103				
13	0	0.0228	0.0499	0.0076	-0.0259	-0.0333	-0.0243	-0.0301	-0.0246	-0.0253	-0.0161	-0.0167	-0.0100
	12	-0.0105	-0.0223	-0.0075	0.0096	0.0440	0.0619	0.0249	-0.0124				
14	0	0.0189	0.0601	0.0646	0.0588	0.0284	-0.0211	-0.0389	-0.0314	-0.0114	0.0087	0.0152	0.0306
	12	0.0350	0.0157	0.0440	0.0312	-0.0015	0.0053	0.0246	0.0291				
15	0	0.0059	0.0515	0.0418	0.0053	-0.0154	-0.0057	0.0182	0.0228	0.0037	-0.0067	-0.0022	0.0166
	12	-0.0012	-0.0253	0.0160	0.0245	0.0228	0.0329	0.0321	0.0165				

Table C. 60 Alpha codebook for MODE_VQ == SCL_2 / SCL_2_960

VN	EN	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0.4911	0.4959	0.4939	0.4988	0.4995	0.4993	0.4960	0.4892	0.4890	0.4893	0.4989	0.4973
	12	0.4988	0.4990	0.4998	0.4901	0.4965	0.4906	0.4887	0.4559				
1	0	0.3410	0.3217	0.2982	0.2780	0.2704	0.2488	0.2454	0.2387	0.2139	0.1878	0.2154	0.2162
	12	0.2105	0.1887	0.2221	0.2338	0.2280	0.2502	0.2554	0.2984				



//ISO/JTC 1/SC 29 **N2203TF**

Date: 1998-05-15

ISO/IEC CD 14496-3 Subpart 4

ISO/JTC 1/SC 29/WG11

Secretariat:

Information Technology - Coding of Audiovisual Objects

Part 3: Audio

Subpart 4: Time/Frequency Coding

Document type: International standard
Document:sub-type if applicable
Document:stage (20) Préparation
Document:language E

C:\ISOST18\BASICEN.DOT ISOSTD Basic Version 1.8 1996-10-30

Subpart 4 of CD 14496-3 is split into several files:

w2203tfs	This file, syntax, semantics and decoder description
w2203tft	T/F tool descriptions and normative Annex
w2203tfa	Informative annex (Transport streams, Encoder tools)
w2203tvq	Twin-VQ vector quantizer tables

0 Introduction

The MPEG-4 Audio T/F based coding is mainly intended to be used for generic audio coding at all but the lowest bitrates. Typically, T/F based encoding is used for complex music material from 6 kbit/s per channel and for stereo signals from 16 kbit/s per stereo signal up to broadcast quality audio at 64 kbit/s per channel and more. MPEG-2 Advanced Audio Coding (AAC) syntax (including support for multi-channel audio) is fully supported by MPEG-4 Audio T/F based coding. The tools derived from MPEG-2 AAC are available together with other MPEG-4 T/F based coding tools to code audio objects using MPEG-4 functionality (including scalability). MPEG-4 T/F based coding is not restricted to some fixed bitrates but supports a wide range of bitrates and variable rate coding.

The block diagrams of the T/F based encoder and decoder reflect the structure of MPEG-4 T/F coding. In general, there are the MPEG-2 AAC related tools with MPEG-4 add-ons for some of them and the tools related to the Twin-VQ Quantization and Coding. The Twin-VQ is an alternative module for the AAC-type quantization and it is based on an interleave vector quantization and LPC (Linear Predictive Coding) spectral estimation. It covers from 6 kbit/s to over 40 kbit/s with constant bit rate. This feature provides merits in terms of error robustness, scaleability and random accecss.

While efficient mono, stereo and multi-channel coding is possible using the MPEG-2 AAC tools, the document also provides extensions to this toolset further enhancing compression performance, scaleability based on a core coder, mono/stereo scaleability etc..

In this context, the Bit-sliced Arithmetic Coding (BSAC) scheme provides a possibility for noiseless transcoding of an AAC stream into a fine granule scalable stream between 16 kbit/s to 64 kbit/s per channel.With BSAC, the bit rate control can be done with a stepsize of 1 kbit/s, which enables the decoder to stop anywhere between 16 kbit/s and the encoded bit rate with a 1 kbit/s stepsize.

All the features and possibilities of the MPEG-2 AAC standard also apply to MPEG-4. AAC has been tested to allow for ITU-R ‘indistinguishable’ quality according to [4] at data rates of 320 kb/s for five full-bandwidth channel audio signals.

0.1 Overview of tools

The basic structure of the MPEG-4 T/F system is shown in Figures 1 and 2. The data flow in this diagram is from left to right, top to bottom. The functions of the decoder are to find the description of the quantized audio spectra in the bitstream, decode the quantized values and other reconstruction information, reconstruct the quantized spectra, process the reconstructed spectra through whatever tools are active in the bitstream in order to arrive at the actual signal spectra as described by the input bitstream, and finally convert the frequency domain spectra to the time domain, with or without an optional gain control tool. Following the initial reconstruction and scaling of the spectrum reconstruction, there are many optional tools that modify one or more of the spectra in order to provide more efficient coding. For each of the optional tools that operate in the spectral domain, the option to “pass through” is retained, and in all cases where a spectral operation is omitted, the spectra at its input are passed directly through the tool without modification.

The input to the bitstream demultiplexer tool is the MPEG-4 T/F bitstream. The demultiplexer separates the bitstream into the parts for each tool, and provides each of the tools with the bitstream information related to that tool.
The outputs from the bitstream demultiplexer tool are:

- The quantized (and optionally noiselessly coded) spectra represented by either
  - the sectioning information and the noiselessly coded spectra (AAC) or
  - the BSAC information or
  - a set of indices of code vectors (TwinVQ)
- The M/S decision information (optional)
- The predictor state information (optional)
- The perceptual noise substitution (PNS) information (optional)
- The intensity stereo control information and coupling channel control information (both optional)
- The temporal noise shaping (TNS) information (optional)
- The filterbank control information
- The gain control information (optional)

The AAC noiseless decoding tool takes information from the bitstream demultiplexer, parses that information, decodes the Huffman coded data, and reconstructs the quantized spectra and the Huffman and DPCM coded scalefactors.
The inputs to the noiseless decoding tool are:

- The sectioning information for the noiselessly coded spectra
- The noiselessly coded spectra

The outputs of the noiseless decoding tool are:

- The decoded integer representation of the scalefactors:
- The quantized values for the spectra

The BSAC tool provides an alternative to the AAC noiseless coding tool, which provides fine granule scalability. This tool takes information from bitstream demultiplexer, parses that information, decodes the Arithmetic coded bit-sliced data, and reconstructs the quantized spectra and the scalefactors.
The inputs to the BSAC decoding tool are:

- The noiselessly coded bit-sliced data
- The target layer information to be decoded

The outputs from the BSAC decoding tool are:

- The decoded integer representation of the scalefactors
- The quantized value for the spectra

The inverse quantizer tool takes the quantized values for the spectra, and converts the integer values to the non-scaled, reconstructed spectra. This quantizer is a non-uniform quantizer.
The input to the Inverse Quantizer tool is:

- The quantized values for the spectra

The output of the inverse quantizer tool is:

- The un-scaled, inversely quantized spectra

The scalefactor tool converts the integer representation of the scalefactors to the actual values, and multiplies the un-scaled inversely quantized spectra by the relevant scalefactors.
The inputs to the scalefactors tool are:

- The decoded integer representation of the scalefactors
- The un-scaled, inversely quantized spectra

The output from the scalefactors tool is:

- The scaled, inversely quantized spectra

The M/S tool converts spectra pairs from Mid/Side to Left/Right under control of the M/S decision information, improving stereo imaging quality and sometimes providing coding efficiency.
The inputs to the M/S tool are:

- The M/S decision information
- The scaled, inversely quantized spectra related to pairs of channels

The output from the M/S tool is:

- The scaled, inversely quantized spectra related to pairs of channels, after M/S decoding

Note: The scaled, inversely quantized spectra of individually coded channels are not processed by the M/S block, rather they are passed directly through the block without modification. If the M/S block is not active, all spectra are passed through this block unmodified.

The prediction tool reverses the prediction process carried out at the encoder. This prediction process re-inserts the redundancy that was extracted by the prediction tool at the encoder, under the control of the predictor state information. This tool is implemented as a second order backward adaptive predictor. The inputs to the prediction tool are:

- The predictor state information
- The scaled, inversely quantized spectra

The output from the prediction tool is:

- The scaled, inversely quantized spectra, after prediction is applied.

Note: If the prediction is disabled, the scaled, inversely quantized spectra are passed directly through the block without modification.

Alternatively, there is a low complexity prediction mode and a long term predictor provided.

The perceptual noise substitution (PNS) tool implements noise substitution decoding on channel spectra by providing an efficient representation for noise-like signal components.
The inputs to the perceptual noise substitution tool are:

- The inversely quantized spectra
- The perceptual noise substitution control information

The output from the perceptual noise substitution tool is:

- The inversely quantized spectra

Note: If either part of this block is disabled, the scaled, inversely quantized spectra are passed directly through this part without modification. If the perceptual noise substitution block is not active, all spectra are passed through this block unmodified.

The intensity stereo / coupling tool implements intensity stereo decoding on pairs of spectra. In addition, it adds the relevant data from a dependently switched coupling channel to the spectra at this point, as directed by the coupling control information.
The inputs to the intensity stereo / coupling tool are:

- The inversely quantized spectra
- The intensity stereo control information and coupling control information

The output from the intensity stereo / coupling tool is:

- The inversely quantized spectra after intensity and coupling channel decoding.

Note: If either part of this block is disabled, the scaled, inversely quantized spectra are passed directly through this part without modification. The intensity stereo tool and M/S tools are arranged so that the operation of M/S and Intensity stereo are mutually exclusive on any given scalefactor band and group of one pair of spectra.

The temporal noise shaping (TNS) tool implements a control of the fine time structure of the coding noise. In the encoder, the TNS process has flattened the temporal envelope of the signal to which it has been applied. In the decoder, the inverse process is used to restore the actual temporal envelope(s), under control of the TNS information. This is done by applying a filtering process to parts of the spectral data.
The inputs to the TNS tool are:

- The inversely quantized spectra
- The TNS information

The output from the TNS block is:

- The inversely quantized spectra

Note: If this block is disabled, the inversely quantized spectra are passed through without modification.

The filterbank tool applies the inverse of the frequency mapping that was carried out in the encoder, as indicated by the filterbank control information and the presence or absence of gain control information. An inverse modified discrete cosine transform (IMDCT) is used for the filterbank tool. If the gain control tool is not used, the IMDCT in the standard AAC mode input consists of either 1024 or 128 spectral coefficients, depending of the value of window_sequence (see 1.3, Table 6.11). If the gain control tool is used, the filterbank tool is configured to use four sets of either 256 or 32 coefficients, depending of the value of window_sequence.
The inputs to the filterbank tool are:

- The inversely quantized spectra

- The filterbank control information

The output(s) from the filterbank tool is (are):

- The time domain reconstructed audio signal(s).

Currently five alternative, but very similar versions of this tool are part of the VM.

a) 1024 or 128 shift length type with the option to select two window shapes (AAC)
b) 4 x switchable 256 or 32 shift length type with the option to select two window shapes (AAC)
c) 2048 or 512 or 128 shift length type with a sine window as defined for TwinVQ
d) 960 or 120 shift length type with the option to select two window shapes (AAC-derived)

When present, the gain control tool applies a separate time domain gain control to each of 4 frequency bands that have been created by the gain control PQF filterbank in the encoder. Then, it assembles the 4 frequency bands and reconstructs the time waveform through the gain control tool's filterbank.
The inputs to the gain control tool are:

- The time domain reconstructed audio signal(s)
- The gain control information

The output(s) from the gain control tool is (are):

- The time domain reconstructed audio signal(s)

If the gain control tool is not active, the time domain reconstructed audio signal(s) are passed directly from the filterbank tool to the output of the decoder. This tool is used for the scaleable sampling rate (SSR) profile only.

The spectral normalisation tool converts the reconstructed flat spectra to the actual values at the decoder. The spectral envelope is specified by LPC coefficients, a Bark scale envelope, periodic pulse components, and gain.
The input to the spectral normalization tool is

- The reconstructed flat spectra

The output from the spectral normalization tool is

- The reconstructed actual spectra

The TwinVQ tool converts the vector index to a flattened spectra at the decoder
by means of table look-up of the codebook and inverse interleaving. Quantization noise is minimized by a weighted distortion measure at the encoder instead of an adaptive bit allocation. This is an alternative to the AAC quantization tool.
The input to the TwinVQ tool is:

- A set of indices of the code vector.

The output from the TwinVQ tool is:

- The reconstructed actual spectra

Besides the above mentioned tools, there are a number of building blocks provided to facilitate scaleable coder configurations, like the scalable controller, frequency selective switch and upsampling filter.

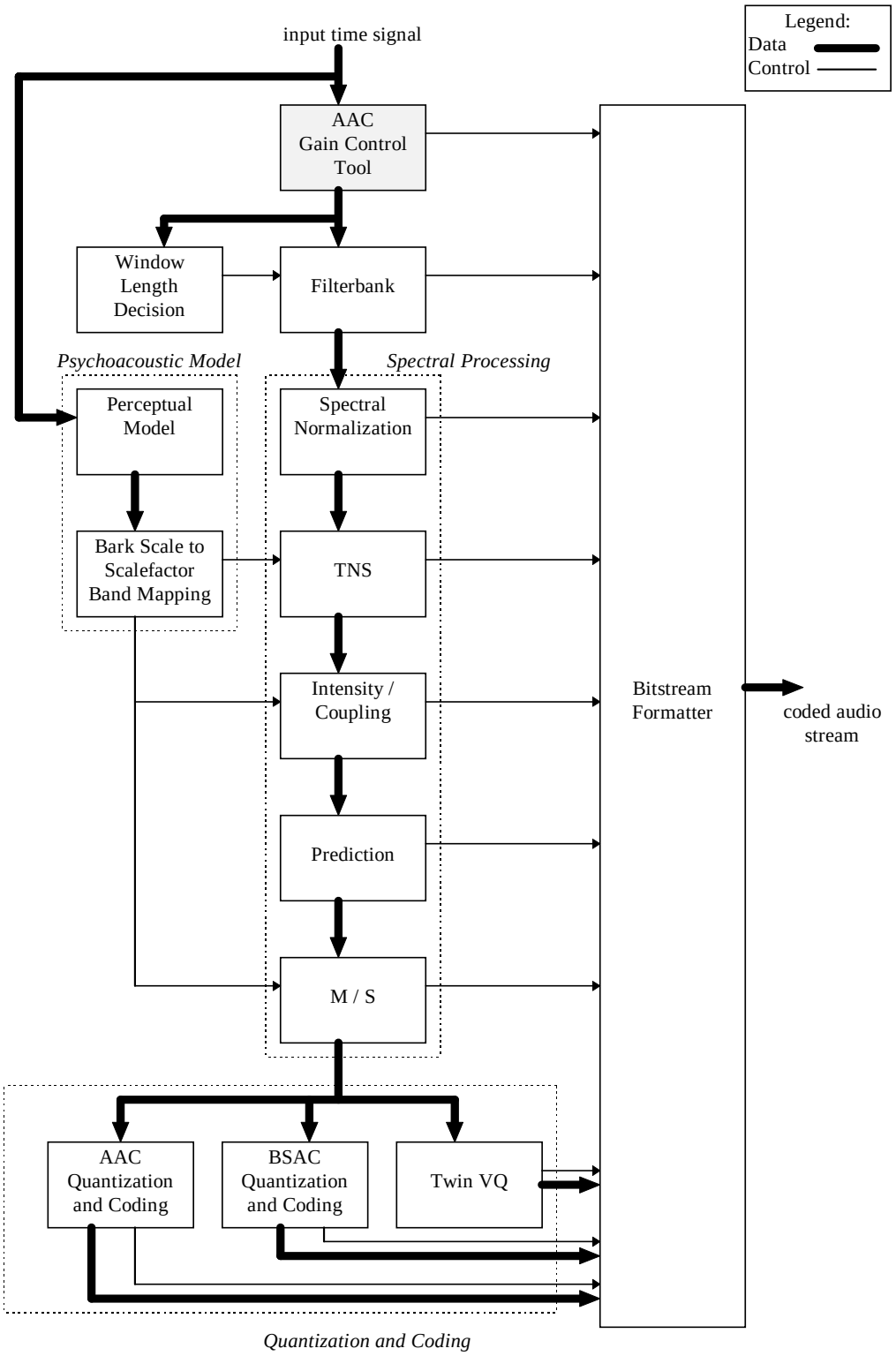


Fig. 1: Blockdiagram TF-based encoder

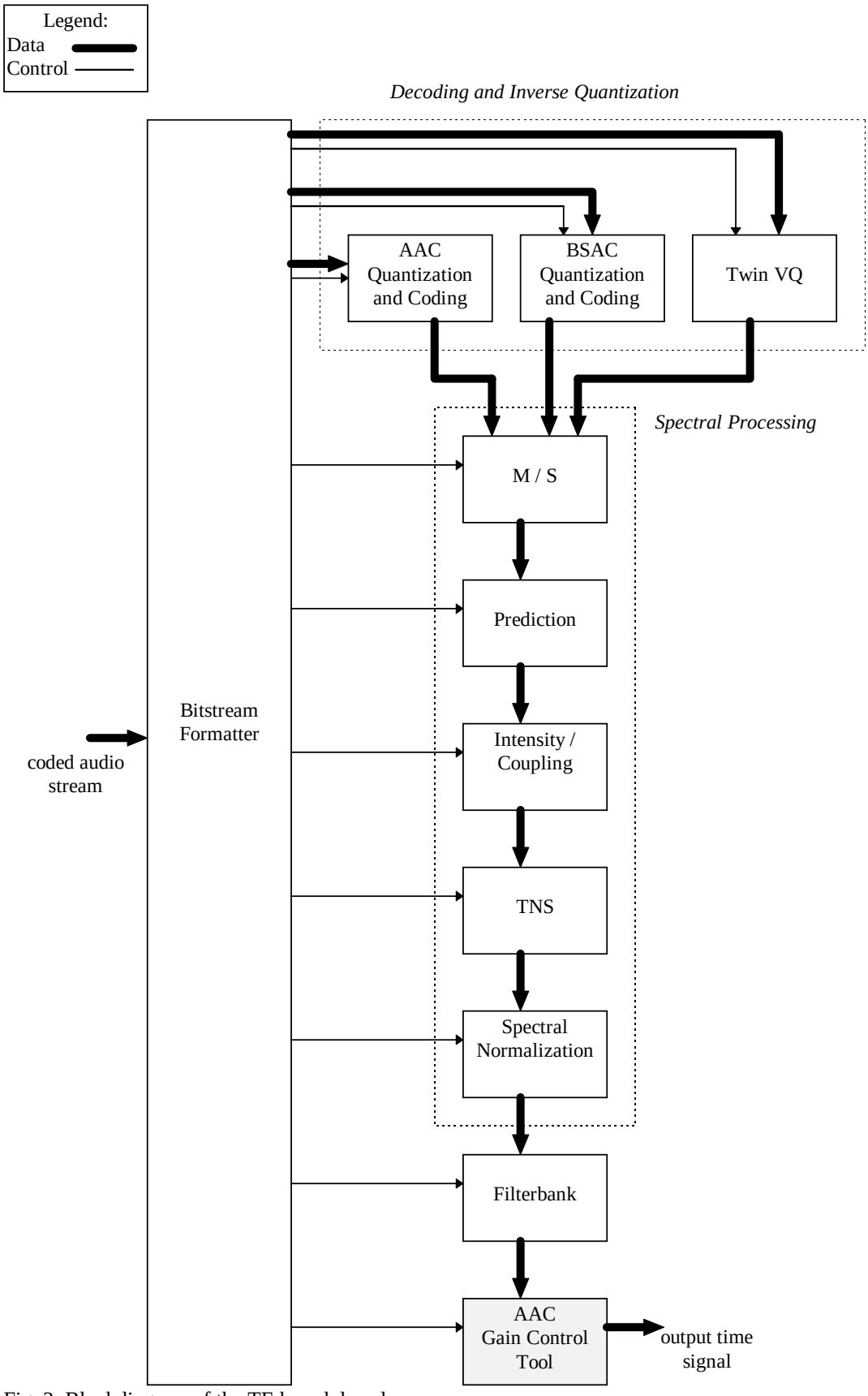


Fig. 2: Blockdiagram of the TF-based decoder

0.2 T/F-specific glossary

0.3 Normative References

[1] ISO/IEC 11172-3:1993, *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s, Part 3: Audio.*

[2] ISO/IEC 13818-3:1997, *Information technology - Generic coding of moving pictures and associated audio, Part 3: Audio.*

[3] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson, Y. Oikawa, "ISO/IEC MPEG-2 Advanced Audio Coding", Presented at the 101st AES Convention, Los Angeles, November 1996.

[4] ITU-R Document TG10-2/3- E only, *Basic Audio Quality Requirements for Digital Audio Bit-Rate Reduction Systems for Broadcast Emission and Primary Distribution*, 28 October 1991.

[5] F. J. Harris, *On the Use of Windows For Harmonic Analysis of the Discrete Fourier Transform*, Proc. of the IEEE, Vol. 66, pp. 51- 83, January 1975.

[6] ISO/IEC 13818-1:1996, *Information technology - Generic coding of moving pictures and associated audio: Systems*

1 Syntax

Three types of streams are part of the MPEG-4 T/F coder syntax. These are

1. Interchange format streams
2. Header streams
3. Raw data streams

Raw data streams are intended to be transported via the MPEG-4 Systems layer. These streams contain all information variing on a frame to frame basis and therefore carry the actual audio information.

The header streams are also transported via MPEG-4 systems. These streams contain configuration information, which is necessary for the decoding process and parsing of the raw data streams. However, an update is only necessary if there are changes in the configuration.

The header and the raw data streams are abstract elements, which define all information for the decoding and parsing of the bitstream. However, for real applications these streams need a transport layer, which cares for the delivery of these streams. Normally this transport mechanism will be handled by MPEG-4 Systems. However, the interface format streams defined in the following section define a simple way of multiplexing the header and the raw data streams.

1.1 Interchange format streams

1.1.1 Audio_Data_Interchange_Format, ADIF

Tabelle 1-1

Syntax	No. of bits	Mnemonic

adif_sequence()
{
  adif_header()
  raw_data_stream()		

```
}
```

Tabelle 1-2 Syntax of adif_header()

Syntax	No. of bits	Mnemonic
adif_header()		
{		
adif_id	**32**	**bslbf**
copyright_id_present	**1**	**bslbf**
if(copyright_id_present)		
copyright_id	**72**	**bslbf**
original_copy	**1**	**bslbf**
home	**1**	**bslbf**
bitstream_type	**1**	**bslbf**
bitrate	**23**	**uimsbf**
num_program_config_elements	**4**	**bslbf**
for (i = 0; i < num_program_config_elements + 1; i++) {		
if(bitstream_type == '0')		
adif_buffer_fullness	**20**	**uimsbf**
program_config_element()		
}		
}		

Tabelle 1-3:Syntax of raw_data_stream()

Syntax	No. of bits	Mnemonic
raw_data_stream()		
{		
while (data_available()) {		
raw_data_block()		
byte_alignment()		
}		
}		

1.1.2 Audio_Data_Transport_Stream frame, ADTS

Tabelle 1-4: Syntax of adts_sequence()

Syntax	No. of bits	Mnemonic
adts_sequence()		
{		
while (nextbits()==syncword) {		
adts_frame()		
}		
}		

Tabelle 1-5: Syntax of adts_frame()

Syntax	No. of bits	Mnemonic
adts_frame()		
{		
byte_alignment()		
adts_fixed_header()		
adts_variable_header()		
adts_error_check()		

```
    for( i=0; i<number_of_raw_data_blocks_in_frame+1; i++) {
        raw_data_block()
    }
}
```

1.1.2.1 Fixed Header of ADTS

Tabelle 1-6: Syntax of adts_fixed_header()

Syntax	No. of bits	Mnemonic
adts_fixed_header()		
{		
syncword	**12**	**bslbf**
ID	**1**	**bslbf**
layer	**2**	**uimsbf**
protection_absent	**1**	**bslbf**
profile	**2**	**uimsbf**
sampling_frequency_index	**4**	**uimsbf**
private_bit	**1**	**bslbf**
channel_configuration	**3**	**uimsbf**
original/copy	**1**	**bslbf**
home	**1**	**bslbf**
emphasis	**2**	**bslbf**
}		

1.1.2.2 Variable Header of ADTS

Tabelle 1-7: Syntax of adts_variable_header()

Syntax	No. of bits	Mnemonic
adts_variable_header()		
{		
copyright_identification_bit	**1**	**bslbf**
copyright_identification_start	**1**	**bslbf**
frame_length	**13**	**bslbf**
adts_buffer_fullness	**11**	**bslbf**
number_of_raw_data_blocks_in_frame	**2**	**uimsfb**
}		

1.1.2.3 Error detection

Tabelle 1-8: Syntax of adts_error_check()

Syntax	No. of bits	Mnemonic
adts_error_check()		
{		
if (protection_absent == ‘0’)		
crc_check	**16**	**rpchof**
}		

1.1.3 Twin-VQ audio sequence

Table 1-9 Syntax of vq_audio_sequence()

Syntax	No. of bits	Mnemonic

vq_audio_sequence()
{
vq_header()
while (nextbit() != NULL){
vq_single_element()
}
}		

Table 1-10 Syntax of vq_scaleable_sequence()

Syntax	No. of bits	Mnemonic

vq_scaleable_sequence()
{
vq_header()
while (nextbit() != NULL){
vq_scaleable_element()
}
}		

1.1.4 AAC-scalable core stream

Tabelle 1-11 Scalable coder stream

Syntax	No. of bits	Mnemonic

core_aac_scalable_stream()
{
while (1) {
syncword
if(**syncword** != 0x37) {
break;
}
scal_header()
bitrate_index
padding_bit
protection_bit
main_data_begin
for(ch=0; ch<no_core_ch; ch++) {
core_coder_stream()
}
aac_ scalable_ main_stream()
for(lay=0; lay<intermediate_layers; lay++) {
aac_ scalable_ extension_stream()
}
for(ch=0; ch<no_core_ch; ch++) {
core_coder_stream()
}
aac_ scalable_ main_stream()
for(lay=0; lay<intermediate_layers; lay++) {
aac_ scalable_ extension_stream()
}
if(third_core_stream()) {
for(ch=0; ch<no_core_ch; ch++) {
core_coder_stream()
}
}
aac_ scalable_ main_stream()
for(lay=0; lay<intermediate_layers; lay++) {
	7
	
	4
	1
	1
	10
		bslbf
		
		bslbf
		bslbf
		bslbf
		bslbf

```
            aac_ scalable_ extension_stream()
        }
    }
}
```

1.1.5 core BSAC stream

Tabelle 1-12 BSAC Scalable coder stream

Syntax	No. of bits	Mnemonic
core_bsac_stream()		
{		
while (1) {		
syncword	**7**	**bslbf**
if(**syncword** != 0x37) {		
break;		
}		
scal_header()		
bitrate_index	**4**	**bslbf**
padding_bit	**1**	**bslbf**
protection_bit	**1**	**bslbf**
main_data_begin	**10**	**bslbf**
for(ch=0; ch<no_core_ch; ch++) {		
core_coder_stream()		
}		
bsac_raw_data_block()		
for(ch=0; ch<no_core_ch; ch++) {		
core_coder_stream()		
}		
bsac_raw_data_block()		
if(third_core_stream()) {		
for(ch=0; ch<no_core_ch; ch++) {		
core_coder_stream()		
}		
}		
bsac_raw_data_block()		
}		
}		

1.1.6 BSAC stream

Tabelle 1-13:Syntax of bsac_data_stream()

Syntax	No. of bits	Mnemonic
bsac_data_stream()		
{		
nch	**3**	**uimbf**
sampling_frequency_index	**4**	**uimbf**
frame_length_flag	**1**	**uimbf**

```
    while  (data_available()) {
        bsac_raw_data_block()
    }
}
```

1.1.7 Scalable Header

Tabelle 1-14: Syntax of program_config_element()

Syntax	No. of bits	Mnemonic
scal_header()		
{		
op_mode	**4**	**bslbf**
if(low_rate_channel_present)		
ccbrflsre	**4**	**bslbf**
if(7<= op_mode<=13)		
intermediate_layers	**2**	**bslbf**
sampling_frequency_index	**4**	**bslbf**
}		

1.2 T/F Audio Specific Configuration

```
class TFSpecificConfig( uint(4) samplingFrequencyIndex, uint(4) channelConfiguration ) {
        uint(2) TFCodingType;
        uint(1) frameLength;
        uint(1) dependsOnCoreCoder;
        if (dependsOnCoreCoder == 1){
                uint(14)coreCoderDelay
        }
        if (TFCodingType==BSAC) {
                uint(11) lslayer_length
        }
        uint (1) extensionFlag;
        if (channelConfiguration == 0 ){
                program_config_element();
        }
        if (extensionFlag==1){
                <to be defined in mpeg4 phase 2>
        }
}
```

1.2.1 Program config element

Tabelle 1-15: Syntax of program_config_element()

Syntax	No. of bits	Mnemonic
program_config_element()		
{		
element_instance_tag	**4**	**uimsbf**
profile	**2**	**uimsbf**
sampling_frequency_index	**4**	**uimsbf**
num_front_channel_elements	**4**	**uimsbf**
num_side_channel_elements	**4**	**uimsbf**
num_back_channel_elements	**4**	**uimsbf**

num_lfe_channel_elements	**2**	**uimsbf**
num_assoc_data_elements	**3**	**uimsbf**
num_valid_cc_elements	**4**	**uimsbf**
mono_mixdown_present	**1**	**uimsbf**
if (mono_mixdown_present == 1)		
mono_mixdown_element_number	**4**	**uimsbf**
stereo_mixdown_present	**1**	**uimsbf**
if (stereo_mixdown_present == 1)		
stereo_mixdown_element_number	**4**	**uimsbf**
matrix_mixdown_idx_present	**1**	**uimsbf**
if (matrix_mixdown_idx_present == 1) {		
matrix_mixdown_idx	**2**	**uimsbf**
pseudo_surround_enable	**1**	**uimsbf**
}		
for (i = 0; i < num_front_channel_elements; i++) {		
front_element_is_cpe[i];	**1**	**bslbf**
front_element_tag_select[i];	**4**	**uimsbf**
}		
for (i = 0; i < num_side_channel_elements; i++) {		
side_element_is_cpe[i];	**1**	**bslbf**
side_element_tag_select[i];	**4**	**uimsbf**
}		
for (i = 0; i < num_back_channel_elements; i++) {		
back_element_is_cpe[i];	**1**	**bslbf**
back_element_tag_select[i];	**4**	**uimsbf**
}		
for (i = 0; i < num_lfe_channel_elements; i++)		
lfe_element_tag_select[i];	**4**	**uimsbf**
for (i = 0; i < num_assoc_data_elements; i++)		
assoc_data_element_tag_select[i];	**4**	**uimsbf**
for (i = 0; i < num_valid_cc_elements; i++) {		
cc_element_is_ind_sw[i];	**1**	**uimsbf**
valid_cc_element_tag_select[i];	**4**	**uimsbf**
}		
byte_alignment()		
comment_field_bytes	**8**	**uimsbf**
for (i = 0; i < comment_field_bytes; i++)		
comment_field_data[i];	**8**	**uimsbf**
}		

1.3 T/F Bitstream Payload

1.3.1 Top Level Payloads of the AAC-only Profiles

Tabelle 1-16: Syntax of raw_data_block()

Syntax	No. of bits	Mnemonic
raw_data_block()		
{		
while((id = **id_syn_ele**) != ID_END){	**3**	**uimsbf**
switch (id) {		
case ID_SCE: single_channel_element()		
break;		
case ID_CPE: channel_pair_element()		
break;		

```
                case ID_CCE:    coupling_channel_element()
                    break;
                case ID_LFE:    lfe_channel_element()
                    break;
                case ID_DSE:    data_stream_element()
                    break;
                case ID_PCE:    program_config_element()
                    break;
                case ID_FIL:    fill_element()
                    break;
            }
        }
}
```

Tabelle 1-17: Syntax of single_channel_element()

Syntax	No. of bits	Mnemonic
single_channel_element() {		
element_instance_tag	**4**	**uimsbf**
individual_channel_stream(0,0) }		

Tabelle 1-18: Syntax of channel_pair_element()

Syntax	No. of bits	Mnemonic
channel_pair_element() {		
element_instance_tag	**4**	**uimsbf**
common_window	**1**	**uimsbf**
if(common_window) {		
ics_info()		
ms_mask_present	**2**	**uimsbf**
if(ms_mask_present == 1) {		
for(g=0; g < num_window_groups; g++) {		
for(sfb=0; sfb < max_sfb; sfb++) {		
ms_used[g][sfb**]**	**1**	**uimsbf**
}		
}		
}		
}		
individual_channel_stream(common_window,0)		
individual_channel_stream(common_window,0) }		

1.3.2 Top Level Payloads of the scalable Profiles

Tabelle 1-19 Syntax of tf_scalable_main_element()

Syntax	**No. of bits**	**Mnemonic**

```
tf_scalable_main_element()
{
    if (TFCodingType != BSAC){
        tf_scalable_main_header(stereo_flag, mono_lay)
    }
    else {
        /* tf_scalable_main_header is included
           in bsac_lstep_stream(0) in case of BSAC */
    }
    if (TFCodingType == AAC_scaleable){
        for (ch=0; ch<(!mono_lay ? 2:1); ch++){
            individual_channel_stream(1, 1)
        }
    } else if (TFCodingType == BSAC){
        bsac_lstep_stream(0)
    } else if (TFCodingType == TwinVQ){
        vq_single_element(0)
    }
}
```

Tabelle 1-20: Syntax of tf_scalable_extension_element()

```
tf_scalable_extension_element()
{
    tf_scalable_extension_header(stereo_flag, mono_lay)
    if (TFCodingType == AAC_scaleable){
        for (ch=0; ch<(!mono_lay ? 2:1); ch++){
            individual_channel_stream(1, 1)
        }
    } else if (TFCodingType == BSAC){
        bsac_lstep_stream(lay)
    } else if (TFCodingType == TwinVQ){
        vq_single_element(lay)
    }
}
```

Tabelle 1-21: Syntax of tf_scalable_main_header()

Syntax	No. of bits	Mnemonic
tf_scalable_main_header(stereo_flag, mono_lay)		
{		
if (TFCodingType != TwinVQ) {		
ics_info()		
}else {		
window_sequence	**2**	**bslbf**
window_shape	**1**	**bslbf**
}		
if(stereo_flag && !mono_lay) {		
ms_mask_present	**2**	**bslbf**
if(ms_mask_present == 1 && TFCodingType != BSAC) {		
if (TFCodingType == TwinVQ){		
if (window_sequence == ONLY_SHORT_WINDOW){		
max_sfb	**4**	**bslbf**
}else{		
max_sfb	**6**	**bslbf**
}		

```
                }
                for( g=0; g<num_window_groups; g++ ) {
                    for( sfb=0; sfb<max_sfb; sfb++ ) {
                        ms_used[g][sfb];
                    }
                }
            }
        }

        for( ch=0 ch< (stereo_flag ? 2:1); ch++ ) {
            ltp_data_present                                    1    bslbf
            if (ltp_data_present){
                ltp_data (last_max_sfb, max_sfb,)


            }
            tns_data_present                                    1    bslbf
            if( tns_data_present )
                tns_data()



            if( core_flag && ( (ch==0) | !mono_lay )
                diff_control_data()
        }
}
```

Tabelle 1-22: Syntax of tf_scalable_extension_header()

Syntax	No. of bits	Mnemonic
tf_scalable_extension_header()		
{		
if (TFCodingType == AAC_scaleable){		
aac_scalable_extension_header()		
}else if (TFCodingType == TwinVQ){		
tvq_scalable_extension_header()		
}		
}		

Tabelle 1-23: Syntax of aac_scalable_extension_header()

Syntax	No. of bits	Mnemonic
aac_scalable_extension_header()		
{		
if(window_sequence == ONLY_SHORT_WINDOW) {		
max_sfb[lay]	**4**	**bslbf**
} else {		
max_sfb[lay]	**6**	**bslbf**
}		
if(!mono_lay) {		
if(stereo_flag)		
ms_mask_present	**2**	**bslbf**
if(ms_mask_present == 1) {		
for(g=0; g<num_window_groups; g++) {		
for(sfb=last_max_sfb_ms; sfb<max_sfb; sfb++) {		
ms_used[g][sfb];	**1**	**bslbf**
}		

```
            }
        }
    }
    for( ch=0; ch<(!mono_flag ? 2:1); ch++ ) {
        ltp_data_present                                    1       bslbf
        if (ltp_data_present)
            ltp_data(last_max_sfb, max_sfb)
        if( mono_stereo_flag ) {
            tns_channel_mono_layer                          1       bslbf
            tns_data_present                                1
            if( tns_data_present
               tns_data()
            if( block_type != SHORT_WINDOW )
                for( sfb=0; sfb<max_sfb; sfb++ )
                    if( !ms_used[0][sfb])
                        diff_control_lr[0][sfb]             1       bslbf
            else
                for( win=0; win<8; win++ )
                    diff_control_lr[win][0]                 1       bslbf
        }
     }
}
```

Tabelle 1-24: Syntax of tvq_scalable_extension_header()

```
Syntax                                                  No. of bits   Mnemonic
tvq_scalable_extension_header()
{
    ltp_data_present                                        1         bslbf
    if (ltp_data_present)
        ltp_data(last_max_sfb, max_sfb)
    if( !mono_lay ) {




        if( mono_stereo_flag )
            ms_mask_present                                 2         bslbf
        if( ms_mask_present == 1 ) {
            if (window_sequence == ONLY_SHORT_WINDOW){
                max_sfb[lay]                                4         bslbf
            } else {
                max_sfb[lay]                                6         bslbf
            }
            for( g=0; g<num_window_groups; g++ ) {
                for( sfb=last_max_sfb_ms; sfb<max_sfb; sfb++ ) {
                    ms_used[g][sfb];                        1         bslbf
                }
            }
        }
    }
}
```

1.3.3 Subsidiary Payloads

Tabelle 1-25: Syntax of ics_info()

```
ics_info()
{
```

	ics_reserved_bit	**1**	**bslbf**
	window_sequence	**2**	**uimsbf**
	window_shape	**1**	**uimsbf**
	if(window_sequence == EIGHT_SHORT_SEQUENCE) {		
	max_sfb	**4**	**uimsbf**
	scale_factor_grouping	**7**	**uimsbf**
	}		
	else {		
	max_sfb	**6**	**uimsbf**
	ltp_data_present	**1**	**uimsbf**
	if (ltp_data_present)		
	ltp_data**()**		
	if (common_window) {		
	ltp_data_present	**1**	**uimsbf**
	if (ltp_data_present)		
	ltp_data**()**		
	}		
	}		
	}		
}			

Tabelle 1-26: Syntax of individual_channel_stream()

Syntax	No. of bits	Mnemonic
individual_channel_stream(common_window,scale_flag)		
{		
global_gain	**8**	**uimsbf**
if(!common_window && !scale_flag)		
ics_info()		
section_data()		
scale_factor_data()		
pulse_data_present	**1**	**uismbf**
if(pulse_data_present) {		
pulse_data()		
}		
if(!scale_flag) {		
tns_data_present	**1**	**uimsbf**
if(tns_data_present)		
tns_data()		
gain_control_data_present	**1**	**uimsbf**
if(gain_control_data_present)		
gain_control_data()		
}		
spectral_data()		

```
}
```

Tabelle 1-27: Syntax of section_data()

Syntax	No. of bits	Mnemonic
section_data()		
{		
if(window_sequence == EIGHT_SHORT_SEQUENCE)		
sect_esc_val = (1<<3) - 1		
else		
sect_esc_val = (1<<5) - 1		
for(g=0; g < num_window_groups; g++) {		
k=0		
i=0		
while (k<max_sfb) {		
sect_cb[g][i]	**4**	**uimsbf**
sect_len=0		
while (**sect_len_incr** == sect_esc_val)	**3/5**	**uimsbf**
sect_len += sect_esc_val		
sect_len += sect_len_incr		
sect_start[g][i] = k		
sect_end[g][i] = k+sect_len		
for (sfb=k; sfb<k+sect_len; k++)		
sfb_cb[g][sfb] = sect_cb[g][i];		
k += sect_len		
i++		
}		
num_sec[g] = i		
}		
}		

Tabelle 1-28: Syntax of scale_factor_data()

Syntax	No. of bits	Mnemonic
scale_factor_data()		
{		
noise_pcm_flag = 1		
for (g=0; g<num_window_groups; g++) {		
for (sfb=0; sfb<max_sfb; sfb++) {		
if (sect_cb[g][sfb] != ZERO_HCB) {		
if (is_intensity(g,sfb))		
hcod_sf[dpcm_is_position[g][sfb]]	**1..19**	**bslbf**
else if (is_noise(g,sfb)) {		
if (noise_pcm_flag) {		
noise_pcm_flag = 0		
dpcm_noise_energy[g][sfb]	**9**	**uimsbf**
} else		
hcod_sf[dpcm_noise_energy[g][sfb]]	**1..19**	**bslbf**
else		
hcod_sf[dpcm_sf[g][sfb]]	**1..19**	**bslbf**
}		
}		
}		
}		

Tabelle 1-29: Syntax of tns_data()

Syntax	No. of bits	Mnemonic
tns_data()		
{		
for (w=0; w<num_windows; w++) {		
n_filt[w]	**1..2**	**uimsbf**
if (n_filt[w])		
coef_res[w]	**1**	**uimsbf**
for (filt=0; filt<n_filt[w]; filt++) {		
length[w][filt]	**4/6**	**uimsbf**
order[w][filt]	**3/5**	**uimsbf**
if (order[w][filt]) {		
direction[w][filt]	**1**	**uimsbf**
coef_compress[w][filt]	**1**	**uimsbf**
for (i=0; i<order[w][filt]; i++)		
coef[w][filt][i]	**2..4**	**uimsbf**
}		
}		
}		
}		

Tabelle 1-30: Syntax of ltp_data()

Syntax	No. of bits	Mnemonic
ltp_data(start_sfb, stop_sfb)		
{		
ltp_lag	**11**	**uimsbf**
ltp_coef	**3**	**uimsbf**
if (TFcodingType != TwinVQ && TFCodingType !=		
AAC_non_scaleable {		
if(window_sequence==EIGHT_SHORT_SEQUENCE) {		
for (w=0; w<num_windows; w++) {		
ltp_short_used[w]	**1**	**uimsbf**
if (ltp_short_used [w]) {		
ltp_short_lag_present[w]	**1**	
}		
if (ltp_short_lag_present[w]) {		
ltp_short_lag[w]	**4**	**uimsbf**
}		
}		
}		
else {		
for (sfb=start_sfb; sfb< stop_sfb); sfb++) {		
ltp_long_used[sfb]	**1**	**uimsbf**
}		
}		
}		
else {		
for (sfb=start_sfb0; sfb<min(max_sfb, stop_sfb); sfb++) {		
ltp_long_used[sfb]	**1**	**uimsbf**
}		
}		

```
}
```

Tabelle 1-31: Syntax of spectral_data()

Syntax	No. of bits	Mnemonic
spectral_data()		
{		
for(g=0; g<num_window_groups; g++) {		
for (i=0; i<num_sec[g]; i++) {		
if (sect_cb[g][i] != ZERO_HCB &&		
sect_cb[g][i] != NOISE_HCB &&		
sect_cb[g][i] != INTENSITY_HCB &&		
sect_cb[g][i] != INTENSITY_HCB2) {		
for (k=sect_sfb_offset[g][sect_start[g][i]];		
k< sect_sfb_offset[g][sect_end[g][i]];) {		
if (sect_cb[g][i]<FIRST_PAIR_HCB) {		
hcod[sect_cb[g][i]][w][x][y][z]	**1..31**	**bslbf**
if(unsigned_cb[sect_cb[g][i]])		
quad_sign_bits	**0..4**	**bslbf**
k += QUAD_LEN		
}		
else {		
hcod[sect_cb[g][i]][y][z]	**1..31**	**bslbf**
if(unsigned_cb[sect_cb[g][i]])		
pair_sign_bits	**0..2**	**bslbf**
k += PAIR_LEN		
if (sect_cb[g][i]==ESC_HCB) {		
if (y==ESC_FLAG)		
hcod_esc_y	**1..31**	**bslbf**
if (z==ESC_FLAG)		
hcod_esc_z	**1..31**	**bslbf**
}		
}		
}		
}		
}		
}		

Tabelle 1-32: Syntax of pulse_data()

Syntax	No. of bits	Mnemonic
pulse_data() {		
number_pulse	**2**	**uimsbf**
pulse_start_sfb	**6**	**uimsbf**
for (i=0; i<number_pulse+1; i++) {		
pulse_offset[i]	**5**	**uimsbf**
pulse_amp[i]	**4**	**uimsbf**
}		
}		

Tabelle 1-33: Syntax of coupling_channel_element()

Syntax	No. of bits	Mnemonic
coupling_channel_element()		
{		

element_instance_tag	**4**	**uimsbf**
ind_sw_cce_flag	**1**	**uimsbf**
num_coupled_elements	**3**	**uimsbf**
num_gain_element_lists = 0		
for (c=0; c<num_coupled_elements+1; c++) {		
num_gain_element_lists++		
cc_target_is_cpe[c]	**1**	**uimsbf**
cc_target_tag_select[c]	**4**	**uimsbf**
if (cc_target_is_cpe[c]) {		
cc_l[c]	**1**	**uimsbf**
cc_r[c]	**1**	**uimsbf**
if (cc_l[c] && cc_r[c])		
num_gain_element_lists++		
}		
}		
cc_domain	**1**	**uimsbf**
gain_element_sign	**1**	**uimsbf**
gain_element_scale	**2**	**uimsbf**
individual_channel_stream(0,0)		
for (c=1; c<num_gain_element_lists; c++) {		
if (ind_sw_cce_flag) {		
cge = 1		
} else {		
common_gain_element_present[c]	**1**	**uimsbf**
cge = common_gain_element_present[c]		
}		
if (cge)		
hcod_sf[common_gain_element[c]]	**1..19**	**bslbf**
else {		
for (g=0; g<num_window_groups) {		
for (sfb=0; sfb<max_sfb; sfb++) {		
if (sfb_cb[g][sfb] != ZERO_HCB)		
hcod_sf[dpcm_gain_element[c][g][sfb]]	**1..19**	**bslbf**
}		
}		
}		
}		
}		

Tabelle 1-34: Syntax of lfe_channel_element()

Syntax	No. of bits	Mnemonic
lfe_channel_element()		
{		
element_instance_tag	**4**	**uimsbf**
individual_channel_stream(0,0)		
}		

Tabelle 1-35: Syntax of data_stream_element()

Syntax	No. of bits	Mnemonic
data_stream_element()		
{		
element_instance_tag	**4**	**uimsbf**
data_byte_align_flag	**1**	**uimsbf**
cnt = **count**	**8**	**uimsbf**

if (cnt == 255)		
cnt += **esc_count**;	**8**	**uimsbf**
if (data_byte_align_flag)		
byte_alignment()		
for (i=0; i<cnt; i++)		
data_stream_byte[element_instance_tag][i];	**8**	**uimsbf**
}		

Tabelle 1-36: Syntax of fill_element()

Syntax	No. of bits	Mnemonic
fill_element()		
{		
cnt = **count**	**4**	**uimsbf**
if (cnt == 15) {		
cnt += **esc_count** - 1;	**8**	**uimsbf**
}		
for (i=0; i<cnt; i++)		
fill_byte[i];	**8**	**uimsbf**
}		

Tabelle 1-37: Syntax of gain_control_data()

Syntax	No. of bits	Mnemonic
gain_control_data()		
{		
max_band	**2**	**uimsbf**
if (window_sequence == ONLY_LONG_SEQUENCE) {		
for (bd=1; bd<=max_band; bd++) {		
for (wd=0; wd<1; wd++) {		
adjust_num[bd][wd]	**3**	**uimsbf**
for (ad=0; ad<adjust_num[bd][wd]; ad++) {		
alevcode[bd**][**wd**][**ad**]**	**4**	**uimsbf**
aloccode[bd**][**wd**][**ad**]**	**5**	**uimsbf**
}		
}		
}		
}		
else if (window_sequence == LONG_START_SEQUENCE) {		
for (bd=1; bd<=max_band; bd++) {		
for (wd=0; wd<2; wd++) {		
adjust_num[bd**][**wd**]**	**3**	**uimsbf**
for (ad=0; ad<adjust_num[bd][wd]; ad++) {		
alevcode[bd**][**wd**][**ad**]**	**4**	**uimsbf**
if (wd == 0)		
aloccode[bd**][**wd**][**ad**]**	**4**	**uimsbf**
else		
aloccode[bd**][**wd**][**ad**]**	**2**	**uimsbf**
}		
}		
}		
}		
else if (window_sequence == EIGHT_SHORT_SEQUENCE) {		
for (bd=1; bd<=max_band; bd++) {		
for(wd=0; wd<8; wd++) {		
adjust_num[bd**][**wd**]**	**3**	**uimsbf**
for (ad=0; ad<adjust_num[bd][wd]; ad++) {		

	alevcode[bd**][**wd**][**ad**]**	**4**	**uimsbf**
	aloccode[bd**][**wd**][**ad**]**	**2**	**uimsbf**
	}		
	}		
	}		
}			
else if (window_sequence == LONG_STOP_SEQUENCE) {			
for (bd=1; bd<=max_band; bd++) {			
for(wd=0; wd<2; wd++) {			
adjust_num[bd**][**wd**]**		**3**	**uimsbf**
for (ad=0; ad<adjust_num[bd][wd]; ad++) {			
alevcode[bd**][**wd**][**ad**]**		**4**	**uimsbf**
if (wd == 0)			
aloccode[bd**][**wd**][**ad**]**		**4**	**uimsbf**
else			
aloccode[bd**][**wd**][**ad**]**		**5**	**uimsbf**
}			
}			
}			
}			
}			

Tabelle 1-38: Syntax of diff_control_data()

Syntax	No. of bits	Mnemonic
diff_control_data()		
{		
if(block_type == SHORT_WINDOW)		
for(w=0; w<8; w++)		
diff_control[w][0]	**1**	**bslbf**
else		
for(dc_group=0; dc_group<no_of_dc_groups; dc_group++)		
diff_control[0][dc_group]	**2..5**	**bslbf**
}		

Tabelle 1-39: Syntax of Twin-VQ scaleable element

Syntax	No. of bits	Mnemonic
vq_scaleable_element()		
{		
for (lyr=0; lyr<=num_enhancement_lyr; lyr++){		
vq_single_element(lyr)		
}		
}		

Tabelle 1-40 : Syntax of Twin-VQ single element

Syntax	No. of bits	Mnemonic
vq_single_element(lyr)		
{		
window_sequence	**4**	**uimsbf**
switch (window_sequence){		
case ONLY_LONG_WINDOW:		
case LONG_MEDIUM_WINDOW:		
case MEDIUM_LONG_WINDOW:		
case LONG_SHORT_WINDOW:		

```
    case SHORT_LONG_WINDOW:
        vq_data(LONG, lyr)
    case ONLY_MEDIUM_WINDOW:
    case MEDIUM_SHORT_WINDOW:
    case SHORT_MEDIUM_WINDOW:
        vq_data(MEDIUM, lyr)
    case ONLY_SHORT_WINDOW:
        vq_data(SHORT, lyr)
    }
}
```

Tabelle 1-41 : Syntax of Twin-VQ data element

Syntax	No. of bits	Mnemonic
vq_data(b_type, lyr)		
{		
if (b_type == LONG && LTP_ACTIVE){		
ltp_data_present	**1**	**uimsbf**
if (ltp_data_present)		
ltp_data()		
}		
if (b_type != SHORT && lyr == 0)		
optional_info	**2**	**uimsbf**
if (lyr > 0)		
for (i_ch=0; i_ch<n_ch; i_ch++){		
fb_shift[i_ch]	**2**	**uimsbf**
}		
if (lyr == 0){		
for (i_ch=0; i_ch<n_ch; i_ch++){		
index_blim_h[i_ch]	**0,2,3**	**uimsbf**
index_blim_l[i_ch]	**0/1**	**uimsbf**
}		
}		
for (idiv=0; idiv<N_DIV; idiv++){		
index_shape0[idiv]	**6/7**	**uimsbf**
index_shape1[idiv]	**6/7**	**uimsbf**
}		
for (i_ch=0; i_ch<n_ch; i_ch++){		
for (isb=0; isb<N_SF; isb++){		
for (ifdiv=0; ifdiv<FW_N_DIV; ifdiv++){		
index_env[i_ch][isb][ifdiv]	**0,3,5,6**	**uimsbf**
}		
}		
}		
for (i_ch=0; i_ch<n_ch; i_ch++){		
for (isbm=0; isbm<N_SF; isbm++){		
index_fw_alf[i_ch][isbm]	**1**	**uimsbf**
}		
}		
for (i_ch=0; i_ch<n_ch; i_ch++){		
index_gain[i_ch]	**7..10**	**uimsbf**
if (N_SF[b_type]>1){		
for (isbm=0; isbm<N_SF[b_type]; isbm++){		
index_gain_sb[i_ch][isbm]	**4..6**	**uimsbf**
}		
}		
}		
for (i_ch=0; i_ch<n_ch; i_ch++){		

index_lsp0[i_ch]	**1**	**uimsbf**
index_lsp1[i_ch]	**5/6**	**uimsbf**
for (isplt=0; isplt<LSP_SPLIT; isplit++){		
index_lsp2[i_ch][isplt]	**3/4**	**uimsbf**
}		
}		
if (ppc_present){		
for (idiv=0; idiv<N_DIV_P; idiv++){		
index_shape0_p[idiv]	**7**	**uimsbf**
index_shape1_p[idiv]	**7**	**uimsbf**
}		
for (i_ch=0; i_ch<n_ch; i_ch++){		
index_pit[i_ch]	**8/9**	**uimsbf**
index_pgain[i_ch]	**6/7**	**uimsbf**
}		
}		
}		

Tabelle 1-42 : Syntax of bsac_lstep_stream()

Syntax	No. of bits	Mnemonic
bsac_lstep_stream(lslayer)		
{		
for(i=Lstep_offset[lslayer];i<Lstep_offset[lslayer+1];i++)		
BSAC_stream_buf[i]	**8**	**uimsbf**
/* Large step stream is saved in BSAC_stream_buf[]. BSAC_stream_buf[] is mapped to small step stream, bsac_raw_data_block(), f or the actual decoding. see the decoding process of BSAC large step scalability for more detailed description. */		
}		

Tabelle 1-43 : Syntax of bsac_raw_data_block()

Syntax	No. of bits	Mnemonic
bsac_raw_data_block()		
{		
bsac_main_stream()		
layer=1;		
while(data_available() && layer<=encoded_layer) {		
bsac_layer_stream(nch, layer)		
layer++;		
}		
byte_alignment()		
}		

Tabelle 1-44 : Syntax of bsac_main_stream()

Syntax	No. of bits	Mnemonic
bsac_main_stream()		
{		
switch(nch) {		
case 1 : tf_scalable_main_header(0, 0)		
break		
case 2 : tf_scalable_main_header(1, 0)		

```
                break
        }
        bsac_general_info(nch)
        bsac_layer_stream(nch, 0)
}
```

Tabelle 1-45 : Syntax of bsac_layer_stream()

Syntax	No. of bits	Mnemonic

bsac_layer_stream(nch, layer)
{
bsac_side_info(nch, layer)
bsac_spectral_data(nch, layer)
}		

Tabelle 1-46 : Syntax of bsac_general_info()

Syntax	No. of bits	Mnemonic
bsac_general_info(nch) {		
frame_length	**10/11**	**uimbf**
encoded_layer	**6**	**uimbf**
for(ch=0;ch<nch;ch++) {		
max_scalefactor[ch]	**8**	**uimbf**
scalefactor_model[ch]	**2**	**uimbf**
min_ArModel[ch]	**5**	**uimbf**
ArModel_model[ch]	**2**	**uimbf**
scf_coding[ch]	**1**	**uimbf**
}		
pns_data_present	**1**	**uimbf**
if (pns_data_present**)**		
pns_start_sfb	**6**	**uimbf**
}		

Tabelle 1-47 : Syntax of bsac_side_info ()

Syntax	No. of bits	Mnemonic
bsac_side_info (nch, layer) {		
if (nch==1) {		
if(pns_data_present) {		
for(g = 0; g < num_window_groups; g++) {		
for(sfb=layer_sfb[layer];sfb<layer_sfb[layer+1];sfb++){		
if(sfb>=pns_start_sfb)		
acode_noise_flag	**1**	**bslbf**
}		
}		
}		
}		
else if (ms_mask_present !=2) {		
for(g = 0; g < num_window_groups; g++) {		
for(sfb=layer_sfb[layer];sfb<layer_sfb[layer+1];sfb++){		
if (ms_mask_present==1) {		
acode_ms_used	**1**	**bslbf**
pns_data_present = 0		
}		
else if (ms_mask_present==3) {		

```
                acode_stereo_info                                   0..4        bslbf
            }
            if(pns_data_present && sfb>=pns_start_sfb) {
                acode_noise_flag_l                                  1           bslbf
                acode_noise_flag_r                                  1           bslbf
                if(ms_mask_present==3 && stereo_info==3)  {
                    if(noise_flag_l[g][sfb] && noise_flag_r[g][sfb])
                        acode_noise_mode                            2           bslbf
                }
            }
        }
    }
}

noise_pcm_flag = 1;
for(ch=0;ch<nch;ch++) {
    for(g = 0; g < num_window_group; g++) {
        for(sfb=layer_sfb[layer]; sfb<layer_sfb[layer+1]; sfb++) {
            if (scf_coding[ch]) {
                if (noise_flag[ch][g][sfb]) {
                    if (!noise_pcm_flag)
                        acode_dpcm_noise_energy                     0..13       bslbf
                }
                else if (stereo_info[ch][g][sfb]>=2)
                    acode_is_position                               0..13       bslbf
                else
                    acode_scf                                       0..13       bslbf
            }
            else {
                if (noise_flag[ch][g][sfb]) {
                    if (!noise_pcm_flag) {
                        acode_dpcm_noise_energy_index               0..14       bslbf
                        if(dpcm_noise_energy_index==ESC_ INDEX)
                            acode_esc_dpcm_noise_energy_index       3..8        bslbf
                    }
                }
                else if (stereo_info[ch][g][sfb]>=2) {
                    acode_is_position_index                         0..14       bslbf
                    if (is_position_index==ESC_ INDEX)
                        acode_esc_is_position_index                 3..8        bslbf
                }
                else {
                    acode_scf_index                                 1..14       bslbf
                    if (scf_index==ESC_SCF_INDEX)
                        acode_esc_scf_index                         3..8        bslbf
                }
            }
            if (noise_flag[ch][g][sfb] && noise_pcm_flag) {
                acode_pcm_noise_energy                              9           bslbf
                noise_pcm_flag = 0
            }
        }
    }
}

for(ch=0;ch<nch;ch++) {
    for(sfb=layer_sfb[layer]; sfb<layer_sfb[layer+1]; sfb++) {
        for(g = 0; g < num_window_group; g++) {
            band = ( sfb * num_window_group ) + g
```

```
                for(i=swb_offset[band]; i<swb_offset[band+1]; i+=4) {
                    cband = index2cb(ch, i);
                    if(!decode_cband[ch][cband]) {
                        acode_ArModel                    1..14        bslbf
                        decode_cband[ch][cband] = 1
                    }
                }
            }
        }
    }
}
```

Tabelle 1-48 :Syntax of bsac_spectral_data ()

Syntax	No. of bits	Mnemonic
bsac_spectral_data(nch, layer)		
{		
for (snf=maxsnf; snf>0; snf--) {		
for (i =0; i <last_index; i +=4) {		
for(ch=0;ch<nch;ch++) {		
if(i >= layer_index[ch]) continue;		
if (cur_snf[ch][i]<snf) continue;		
dim0 = dim1 = 0		
for(k = 0; k < 4; k++)		
if(prestate[ch][i +k]) dim1++		
else dim0++		
if(dim0)		
acode_vec0	**0..14**	**bslbf**
if(dim1)		
acode_vec1	**0..14**	**bslbf**
for(k = 0; k < 4; k++) {		
if(sample[ch][i +k] &&!prestate[ch][i +k]) {		
acode_sign	**1**	**bslbf**
prestate[ch][i +k] = 1		
}		
}		
cur_snf[ch][i]--		
if (total_estimated_bits >= available_bits[layer]) return		
}		
}		
if (total_estimated_bits >= available_bits[layer]) return		
}		
}		

2 General information

2.1 Decoding of interface formats

2.1.1 Audio_Data_Interchange_Format (ADIF), Audio_Data_Transport_Stream (ADTS) and raw_data_block

2.1.1.1 Definitions

Bit stream elements:

adif_sequence()	a sequence according to the Audio_Data_Interchange_Format (Table 6.1)
adif_header()	header of the Audio_Data_Interchange_Format located at the beginning of an adif_sequence (Table 6.2)
raw_data_block()	see subclause **Fehler! Verweisquelle konnte nicht gefunden werden.** and Table 6.8
adif_id	ID that indicates the Audio_Data_Interchange_Format. Its value is 0x41444946 (most significant bit first), the ASCII representation of the string „ADIF“ (Table 6.2)
copyright_id_present	indicates whether **copyright_id** is present or not (Table 6.2)
copyright_id	The field consists of an 8-bit copyright_identifier, followed by a 64-bit copyright_number (Table 6.2). The copyright identifier is given by a Registration Authority as designated by SC 29. The copyright_number is a value which identifies uniquely the copyrighted material. See ISO/IEC 13818-3, subclause 2.5.2.13.
original_copy	see ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.2)
home	see ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.2)
bitstream_type	a flag indicating the type of a bitstream (Table 6.2): ‘0’ constant rate bitstream. This bitstream may be transmitted via a channel with constant rate ‘1’ variable rate bitstream. This bitstream is not designed for transmission via constant rate channels
bitrate	a 23 bit unsigned integer indicating either the bitrate of the bitstream in bits/sec in case of constant rate bitstream or the maximum peak bitrate (measured per frame) in case of variable rate bitstreams. A value of 0 indicates that the bitrate is not known (Table 6.2)
num_program_config_element	number of program config elements specified for this adif_sequence() (Table 6.2)
adif_buffer_fullness	number of bits remaining in the encoder buffer after encoding the first raw_data_block in the adif_sequence (Table 6.2)
program_config_element()	contains information about the configuration for one program (Table 6.2). See subclause 2.2.1.
adts_sequence()	a sequence according to Audio_Data_Transport_Stream ADTS (Table 6.3)
adts_frame()	a ADTS frame, consisting of a fixed header, a variable header, an optional error check and a specified number of raw_data_blocks() (Table 6.4)
adts_fixed_header()	fixed header of ADTS. The information in this header does not change from frame to frame. It is repeated every frame to allow random access into a bitstream bitstream (Table 6.5)
adts_variable_header()	variable header of ADTS. This header is transmitted every frame as well as the fixed header, but contains data that changes from frame to frame (Table 6.6)
adts_error_check()	CRC error detection data generated as described in ISO/IEC 11172-3, subclause 2.4.3.1 (Table 6.7) The following bits are protected and fed into the CRC algorithm in order of their appearance: all bits of the headers first 192 bits of any single_channel_element (SCE) channel_pair_element (CPE) coupling_channel_element (CCE) low frequency enhancement channel (LFE) In addition, the first 128 bits of the second individual_channel_stream in the channel_pair_element must be protected. All information in any program configuration element or data element must be protected. For any element where the specified protection length of 128 or 192 bits exceeds its actual length, the element is zero padded to the specified protection length for CRC calculation.
byte_alignment()	if called from within a raw_data_block then align with respect to the first bit of the raw_data_block, else align with respect to the first bit of the header.

syncword The bit string ‘1111 1111 1111’. See ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.5)

ID MPEG identifier, set to ‘1’. See ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.5)

layer Indicates which layer is used. Set to ‘00’. See ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.5)

protection_absent Indicates whether error_check() data is present or not. Same as syntax element ‘protection_bit’ in ISO/IEC 11172-3, subclause 2.4.1 and 2.4.2 (Table 6.5)

profile profile used. See clause 2. (Table 6.5)

sampling_frequency_index indicates the sampling frequency used according to the following table: (Table 6.5)

sampling_frequency_index	sampling frequeny
0x0	96000
0x1	88200
0x2	64000
0x3	48000
0x4	44100
0x5	32000
0x6	24000
0x7	22050
0x8	16000
0x9	12000
0xa	11025
0xb	8000
0xc	reserved
0xd	reserved
0xe	reserved
0xf	reserved

private_bit see ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.5)

channel_configuration indicates the channel configuration used. If channel_configuration is greater than 0, the channel configuration is given by the ‘Default bitstream index number’ in **Fehler! Verweisquelle konnte nicht gefunden werden.**, see subclause 2.2.1. If channel_configuration equals 0, the channel configuration is not specified in the header and must be given by a program_config_element following as first bitstream element in the first raw_data_block after the header, or by the implicit configuration (see subclause 8.5) or must be known in the application. (Table 6.5)

emphasis see ISO/IEC 11172-3, subclause 2.4.2.3 (Table 6.5)

copyright_identification_bit One the bits of the 72-bit copyright identification field (see copyright_id above). The bits of this field are transmitted frame by frame; the first bit is indicated by the copyright_identification_start bit set to ‘1’. The field consists of an 8-bit copyright_identifier, followed by a 64-bit copyright_number. The copyright identifier is given by a Registration Authority as designated by SC29. The copyright_number is a value which identifies uniquely the copyrighted material. See ISO/IEC 13818-3, subclause 2.5.2.13 (Table 6.6)

copyright_identification_start One bit to indicate that the copyright_identification_bit in this audio frame is the first bit of the 72-bit copyright identification. If no copyright identification is transmitted, this bit should be kept '0'.

'0' no start of copyright identification in this audio frame

'1' start of copyright identification in this audio frame

See ISO/IEC 13818-3, subclause 2.5.2.13 (Table 6.6)

frame_length length of the frame including headers and error_check (Table 6.5) in bytes

adts_buffer_fullness number of 32 bit words remaining in the encoder buffer after encoding the first raw_data_block in the ADTS frame (Table 6.6). A value of hexadecimal 7FF signals that the bitstream is a variable rate bitstream. In this case, buffer fullness is not applicable.

number_of_raw_data_blocks_in_frame number of raw_data_blocks in the ADTS frame (Table 6.6)

Help elements:

data_available()	function that returns '1' as long as data is available, otherwise '0'

2.1.1.2 Overview

The raw_data_block() contains all data which belongs to the audio (including ancillary data). Beyond that, additional information like sampling_frequency is needed to fully describe an audio sequence. The Audio_Data_Interchange_Format (ADIF) contains all elements that are necessary to describe a bitstream according to this standard.
For specific applications some or all of the syntax elements like those specified in the header of the ADIF, e.g. sampling_rate, may be known to the decoder by other means and hence do not appear in the bitstream. Furthermore, additional information that varies from block to block (e.g. to enhance the parsability or error resilience) may be required. Therefore transport streams may be designed for a specific application and are not specified in this standard. However, one possible transport stream called Audio_Data_Transport_Stream (ADTS) is given, which may be used for applications.

2.1.1.3 Audio_Data_Interchange_Format ADIF

The Audio_Data_Interchange_Format (ADIF) contains one header at the start of the sequence followed by a raw_data_stream (). The raw_data_stream() may not contain any further channel_configuration_elements.
As such, the ADIF is useful only for systems with a defined start and no need to start decoding from within the audio data stream, such as decoding from disk file. It can be used as an interchange format in that it contains all information necessary to decode and play the audio data.

2.1.1.4 Audio_Data_Transport_Stream ADTS

The Audio_Data_Transport_Stream (ADTS) is similar to syntax used in ISO/IEC 11172-3 and 13818-3. This will be recognized by ISO/IEC 11172-3 decoders as a "Layer 4" bit-stream.
The fixed header of the ADTS contains the syncword plus all parts of the header which are necessary for decoding and which do not change from frame to frame. The variable header of the ADTS contains header data which changes from frame to frame.
The ADTS only supports a raw_data_stream() with only one program. The program may have up to 7 channels plus an independently switched coupling channel.

2.1.2 AAC scalable core stream

The format of the transport stream, which is used in the VM.

2.1.2.1 Definitions

syncword	The sequence 0110111.
protection_bit	See ISO/IEC 11172-3.
padding_bit	See ISO/IEC 11172-3.
main_data_begin	See ISO/IEC 11172-3.
bitrate_index	See ISO/IEC 11172-3. However, the **bitrate_index** table has been redifined in the VM (see table below).
granule	See ISO/IEC 11172-3.
third_core_stream()	A helper function which returns 1, if a third core stream is present in a transport frame, which comprises three granules, or 0 if not.

2.1.2.2 Decoding process

Padding_bit, bitrate_index , and main_data_begin are used in the VM transport multiplexer to implement a stream similar to Layer III of ISO/IEC 11172-3. If used, a core coder is inserted at the start of a granule. This combination allows to send out the core coder stream and to decode the core coder with the optimum delay, without having to wait for the delayed AAC stream.

third_core_stream()

The function third_core_stream() is defined as follows:

Return 1, if two times the core coder frame length is less than three times of half of the AAC frame length. Return 0 else.

2.2 Decoding of the T/F Audio Specific Configuration

2.2.1 General configuration

TFCodingType specifies the TF coding type

	TFCodingType
0x0	AAC scaleable
0x1	BSAC
0x2	TwinVQ
0x3	AAC non scaleable (i.e. multichannel)

`frameLength` specifies the window length of the IMDCT: if set to 0 a 1024 lines IMDCT is used if set to 1 a 960 line IMDCT is used.

`dependsOnCoreCoder` shall be set to 1 if a non MPEG-4 TF coder or a MPEG-4 TF coder at an different sampling rate is used as a core coder in a scaleable bitstream.

`coreCoderDelay` is the delay that has to be applied to the core decoder output befor the MDCT if the core coder and the 1st scaleable TF layer should be decoded.

`lslayer_length` is the length of the BSAC large step scalability layer which is represented in the unit of byte. A BSAC large step scalability layer is conveyed over an elementary stream.

`extensionFlag` is used for use in MPEG-4 phase 2.

`program_config_element()` shall be used only if audio data are MPEG2 AAC data.

2.2.2 Program Config Element (PCE)

profile The two-bit profile index from Table 7.1 (Table 6.21)

sampling_frequency_index Indicates the sampling rate of the program (and all other programs in this bitstream). See definition in 8.1.1 (Table 6.21)

num_front_channel_elements The number of audio syntactic elements in the front channels, front center to back center, symmetrically by left and right, or alternating by left and right in the case of single channel elements (Table 6.21)

num_side_channel_elements Number of elements to the side as above (Table 6.21)

num_back_channel_elements As number of side and front channel elements, for back channels (Table 6.21)

num_lfe_channel_elements number of LFE channel elements associated with this program (Table 6.21)

num_assoc_data_elements The number of associated data elements for this program (Table 6.21)

num_valid_cce_elements The number of cce's that can add to the audio data for this program (Table 6.21)

mono_mixdown_present One bit, indicating the presence of the mono mixdown element (Table 6.21)

mono_mixdown_element_number The number of a specified SCE that is the mono mixdown (Table 6.21)

stereo_mixdown_present One bit, indicating that there is a stereo mixdown present (Table 6.21)

stereo_mixdown_element_number The number of a specified CPE that is the stereo mixdown element (Table 6.21)

matrix_mixdown_idx_present One bit, indicating the presence of matrix mixdown information (Table 6.21)

matrix_mixdown_idx Two bit field, specifying the index of the surround mixdown coefficient (Table 6.21)

pseudo_surround_enable One bit, indicating the possibility of mixdown for pseudo surround reproduction (Table 6.21)

front_element_is_cpe indicates whether a SCE or a CPE is addressed as a front element (Table 6.21)
'0' selects an SCE
'1' selects an CPE
The instance of the SCE or CPE addressed is given by **front_element_tag_select**

front_element_tag_select the instance_tag of the SCE/CPE addressed as a front element (Table 6.21)

side_element_is_cpe	see **front_element_is_cpe**, but for side elements (Table 6.21)
side_element_tag_select	see **front_element_tag_select**, but for side elements (Table 6.21)
back_element_is_cpe	see **front_element_is_cpe**, but for back elements (Table 6.21)
back_element_tag_select	see **front_element_tag_select**, but for back elements (Table 6.21)
lfe_element_tag_select	instance_tag of the LFE addressed (Table 6.21)
assoc_data_element_tag_select	instance_tag of the DSE addressed (Table 6.21)
valid_cce_element_tag_select	instance_tag of the CCE addressed (Table 6.21)
cc_element_is_ind_sw	One bit, indicating that the corresponding CCE is an independently switched coupling channel (Table 6.21)
comment_field_bytes	The length, in bytes, of the following comment field (Table 6.21)
comment_field_data	The data in the comment field (Table 6.21)

SCE or CPE elements within the PCE are addressed with two syntax elements. First, an is_cpe syntax element selects whether a SCE or CPE is addressed. Second, a tag_select syntax element selects the instance_tag of a SCE/CPE. LFE, CCE and DSE elements are directly addressed with their instance_tag.

2.2.2.1 Implicit and defined channel configurations

The MPEG-2 AAC audio syntax provides two ways to convey the mapping of channels within a set of syntactic elements to physical locations of speakers. The first way is a default mapping based on the specific set of syntactic elements received and the order in which they are received. The most common mappings are further defined in**Fehler! Verweisquelle konnte nicht gefunden werden.**. If a mapping shown in **Fehler! Verweisquelle konnte nicht gefunden werden.** is not used, the following methods describe the default determination of channel mapping:

1) Any number of SCE's may appear (as long as permitted by other constraints, for example profile). If this number of SCE's is odd, then the first SCE represents the front center channel, and the other SCE's represent L/R pairs of channels, proceeding from center front outwards and back to center rear.

If the number of SCE's is even, then the SCE's are assigned as pairs as center-front L/R, in pairs proceeding out and back from center front toward center back.

2) Any number of CPE's or *PAIRS* of SCE's may appear. Each CPE or pair of SCE's represents one L/R pair, proceeding from where the first sets of SCE's left off, pairwise until reaching either center back pair.

3) Any number of SCE's. If this number is even, allocating pairs of SCE's Left/Right, from 2), back to center back. If this number is odd, allocated as L/R pairs, except for the final SCE, which is assigned to center back.

In case of this default (or implicit) mapping the number and order of SCE's and CPE's and the resulting configuration may not change within the bitstream without sending a program_configuration_element, i.e. an implicit reconfiguration is not allowed.

Other audio syntactic elements that do not imply additional output speakers, such as coupling channel_element, may follow the listed set of syntactic elements. Obviously non-audio syntactic elements may be received in addition and in any order relative to the listed syntactic elements.

If reliable mapping of channel set to speaker geometry is a concern, then it is recommended that an implicit mapping from **Fehler! Verweisquelle konnte nicht gefunden werden.** or a program configuration element be used.

For more complicated configurations a **Program Configuration Element** (PCE) is defined. There are 16 available PCE's, and each one can specify a distinct program that is present in the raw data stream. All available PCE's within a raw_data_block must come before all other syntactic elements. Programs may or may not share audio syntactic elements, for example, programs could share a channel_pair_element and use distinct coupling channels for voice over in different languages. A given program configuration element contains information pertaining to only one program out of many that may be included in the raw data stream. Included in the PCE are "list of front channels", again using the rule center outwards, left before right. In this list, a center channel SCE, if any, must come first, and any other SCE's must appear in pairs, constituting an LR pair. If only two SCE's are specified, this signifies one LR stereophonic pair.

After the list of front channels, there is a list of "side channels" consisting of CPE's, or of pairs of SCE's. These are listed in the order of front to back. Again, in the case of a pair of SCE's, the first is a left channel, the second a right channel.

After the list of side channels, a list of back channels is available, listed from outside in. Any SCE's except the last SCE must be paired, and the presence of exactly two SCE's (alone or preceeded by a CPE) indicates that the two SCE's are Left and Right Rear center, respectively.

The configuration indicated by the PCE takes effect at the raw_data_block containing the PCE. The number of front, side and back channels as specified in the PCE must be present in that block and all subsequent raw_data_blocks until a raw_data_block containing a new PCE is transmitted.

Other elements are also specified. A list of one or more LFE's is specified for application to this program. A list of one or more CCE's (profile-dependent) is also provided, in order to allow for dialog management as well as different intensity coupling streams for different channels using the same main channels. A list of data streams associated with the program can also associate one or more data streams with a program. The program configuration element also allows for the specification of one monophonic and one stereophonic simulcast mixdown channel for a program.

Note that the MPEG-2 Systems standard [6] supports alternate methods of simulcast.

The PCE element is not intended to allow for rapid program changes. At any time when a given PCE, as selected by its element_instance_tag, defines a new (as opposed to repeated) program, the decoder is not obliged to provide audio signal continuity.

2.2.3 AAC/BSAC scalable core header

Configuration data for the scalable decoder. This is part of the MPEG-4 audio element identifier definition and will be covered there, when the final definition of this element is available. The following tables and definitions reflect the status of the current VM.

2.2.3.1 Definitions

op_mode	definition
0	core layer only (mono)
1	core layer only (stereo)
2	high layer only (mono)
3	high layer only (stereo)
4	core and high (mono, mono)
5	core and high (mono, stereo)
6	core and high (stereo, stereo)
7	core and intermediate and high (mono, mono, mono)
8	core and intermediate and high (mono, mono, stereo)
9	core and intermediate and high (mono, stereo, stereo)
10	core and intermediate and high (stereo, stereo, stereo)
11	intermediate and high (mono, mono)
12	intermediate and high (mono, stereo)
13	intermediate and high (stereo, stereo)
14	reserved
15	reserved

ccbrflsre	core coder bitrate (kbit/s)	core frame length (ms)	number of bytes per frame for the core coder	window length (samples)	example core coder
0	8	10	10	1920	G7.29
1	6.3	30	24	1920	G7.23

2	5.3	30	20	1920	G7.23
3	3.8	30	14	1920	MPEG-4 CELP core
4	4.9	20	12	1920	MPEG-4 CELP core
5	6	20	15	1920	MPEG-4 CELP core
6	7.7	20	19	1920	MPEG-4 CELP core
7	9.9	20	25	1920	MPEG-4 CELP core
8	11	10	14	1920	MPEG-4 CELP core
9	12.2	10	15	1920	MPEG-4 CELP core
10	4.8	30	18	1920	FS1016
11	6				MPEG-4 parametric IL core
12					
13					
14					
15					

intermediate_layers The number of enhancement layers -1. Only transmitted if more than one AAC layer is used

2.2.4 Twin-VQ header

2.2.4.1 Definitions

function indicates flags of supported functionalities.
samplingFrequencyIndex descriptor elment which indicates sampling rate.
bitrate indicates bitrate.

2.2.4.2 Decoder configuration

Configuration mode parameter MODE_VQ, bitrate parameter BITRATE, and sampling frequency parameter SAMPLING_FREQUENCY are set by syntax elements **bitrate, frameLength** and **samplingFrequencyIndex**.

```
if (lyr == 0){
   switch (samplingFrequencyIndex){
      case 24000:
         SAMPLING_FREQUENCY = 24000
         if (bitrate >= 6) {
            if (frameLength == 0) {
               MODE_VQ = 24_06;
            }
            else MODE_VQ = 24_06_960;
         break;
         }
      case 16000:
         SAMPLING_FREQUENCY = 16000.;
         if (bitrate >= 16) MODE_VQ = 16_16;
         break;
      case 8000:
         SAMPING_FREQUENCY = 8000.;
         if (bitrate >= 6) MODE_VQ = 08_06;
         }
         break;
      default:
         break;
   }
}
else{
   if (lyr < 2){
      if (frameLength == 0) {
         MODE_VQ = SCL_1;
      }
```

```
      else{
         MODE_VQ = SCL_1_960;
      }
   }
   else{
      if (frameLength == 0) {
         MODE_VQ = SCL_2;
      }
      else{
         MODE_VQ = SCL_2_960;
      }
   }
}
```

Lower limit of syntax elements **bitrate** is 6 for 48 kHz sampling, 16 kbit/s for 16 kbit/s, and 6 for 8 kHz sampling in the non-scaleable case. In the scaleable case, the lower limit of the **bitrate** is 8.

2.3 Decoding of the T/F Bitstream payload

2.3.1 Definitions

raw_data_stream() sequence of raw_data_block()'s

raw_data_block() block of raw data that contains audio data for a time period of 1024 samples, related information and other data. There are 8 bitstream elements, identified as bitstream element id_syn_ele. The audio elements in one raw data stream and one raw data block must have one and only one sampling rate. In the raw data block, several instances of the same id_syn_ele may occur, but each such instance of an id_syn_ele except for a data_stream_element must have a different 4-bit element_instance_tag. Therefore, in one raw data block, there can be from 0 to at most 16 of any id_syn_ele. The exceptions to this are the data_stream_element, the fill_element and the terminator element. If multiple data stream elements occur which have unique element_instance_tags then they are part of distinct data streams. If multiple data stream elements occur which have the same element_instance_tag then they are part of the same data stream. The fill_element has no element_instance_tag (since the content does not require subsequent reference) and can occur any number of times. The terminator element has no element_instance_tag and must occur exactly once , as it marks the end of the raw_data_block (Table 6.8).

id_syn_ele a bitstream element that identifies one of the following syntactic elements: (Table 6.8)

Syntactic Element	ID name	encoding	Abbreviation
single_channel_element	ID_SCE	0x0	SCE
channel_pair_element	ID_CPE	0x1	CPE
coupling_channel_element	ID_CCE	0x2	CCE
lfe_channel_element	ID_LFE	0x3	LFE
data_stream_element	ID_DSE	0x4	DSE
program_config_element	ID_PCE	0x5	PCE
fill_element	ID_FIL	0x6	FIL
terminator	ID_END	0x7	TERM

single_channel_element() abbreviaton SCE. Syntactic element of the bitstream containing coded data for a single audio channel. A single_channel_element() basically consists of an individual_channel_stream(). There may be up to 16 such elements per raw data block, each one must have a unique element_instance_tag (Table 6.9)

channel_pair_element() abbreviation CPE. Syntactic element of the bitstream containing data for a pair of channels. A channel_pair_element consists of two individual_channel_streams and additional joint channel coding information. The two channels may share common side information. The channel_pair_element has the same restrictions as

	the single channel element as far as element_instance_tag, and number of occurrances (Table 6.10).
coupling_channel_element()	Abbreviation CCE. Syntactic element that contains audio data for a coupling channel. A coupling channel represents the information for multi-channel intensity for one block, or alternately for dialogue for multilingual programming. The rules for number of coupling_channel_elements and instance tags are as for single_channel_elements (Table 6.18). See clause 12.3
lfe_channel_element()	Abbreviation LFE. Syntactic element that contains a low sampling frequency enhancement channel. The rules for the number of lfe_channel_elements and instance tags are as for single_channel_elements (Table 6.19). See clause 8.4
program_config_element()	Abbreviation PCE. Syntactic element that contains program configuration data. The rules for the number of program_config_elements and element instance tags are the same as for single_channel_elements (Table 6.21). PCE's must come before all other syntactic elements in a raw_data_block(). See clause 8.5
fill_element()	Abbreviation FIL. Syntactic element that contains fill data. There may be any number of fill elements, that can come in any order in the raw data block (Table 6.22). See clause 8.7
data_stream_element()	Abbreviation DSE. Syntactic element that contains data. Again, there are 16 element_instance_tags. There is, however, no restriction on the number of data_stream_elements with any one instance tag, as a single data stream may continue across multiple data_stream_elements with the same instance tag (Table 6.20). See clause 8.6
terminator (ID_END)	The terminator id_syn_ele ID_END indicates the end of a raw data block. There must be one and only one terminator per raw data block. (Table 6.8)
element_instance_tag	unique instance tag for syntactic elements other than terminator element and fill element. All syntactic elements containing instance tags may occur more than once, but, except for data_stream_elements, must have an unique element_instance_tag in each raw_data_block. This tag is also used to reference audio syntactic elements in a coupling_channel_element, and single_channel_elements, channel_pair_elements, lfe_channel_elements, data_channel_elements, and coupling_channel_elements inside a program_config_element, and provides the possibility of up to 16 independent program_config_elements (tables 6.9, 6.10, 6.18, 6.19, 6.20, 6.21, 6.22)
audio_channel_element	generic term for single_channel_element, channel_pair_element, coupling_channel_element and lfe_channel_element.

2.3.2 Buffer requirements

Minimum decoder input buffer:

The following rules are used to calculate the maximum number of bits in the input buffer both for the bitstream as a whole, for any given program, or for any given SCE/CPE/CCE:

The input buffer size is 6144 bits per SCE or independently switched CCE, plus 12288 bits per CPE. Both the total buffer and the individual buffer sizes are limited, so that the buffering limit can be calculated for either the entire bitstream, any entire program, or the individual audio elements permitting the decoder to break a multichannel bitstream into separate mono and stereo bitstreams which are decoded by separate mono and stereo decoders, respectively. Although the 6144 bit/CCE must be obeyed for dependent CCE's as well, any bits for dependent CCE's must be supplied from the total buffer requirements based on the independent CCE's, SCE's, and CPE's.

Bit reservoir:

The bit reservoir is controlled at the encoder. The maximum bit reservoir in the encoder depends on the number of channels and the mean bitrate. The maximum bit reservoir size for constant rate channels can be calculated by subtracting the mean number of bits per block from the minimum decoder input buffer size. For example, at 96 kbit/s for a stereo signal at 48 kHz sampling frequency the maximum bit reservoir size is 12288 bit- (96000 kbit/s / 48000 1/s * 1024) = 10240 bit. For variable bitrate channels the encoder must operate in a way that the input buffer requirements do not exceed the minimum decoder input buffer.

The state of the bit reservoir is transmitted in the buffer_fullness field as the number of available bits in the bit reservoir divided by the number of audio channels divided by 32 and truncated to an integer value.

Maximum bit rate:

The maximum bit rate depends on the audio sampling rate. The maximum bit rate per channel can be calculated based on the minimum input buffer according to the formula:

$$\frac{6144\frac{bit}{block}}{1024\frac{samples}{block}} \cdot sampling\_frequency$$

So, for example, this leads to the following maximum bit rates:

sampling_frequency	maximum bit rate / channel
48 kHz	288 kbit/sec
44.1 kHz	264.6 kbit/sec
32 kHz	192 kbit/sec

2.3.3 Decoding process

Assuming that the start of a raw_data_block is known, it can be decoded without any additional "transport-level" information and produces 1024 audio samples per output channel. The sampling rate of the audio signal may be specified in a program_config_element or it may be implied in the specific application domain. Assuming that the start of the first raw_data_block in a raw_data_stream is known, the sequence can be decoded without any additional "transport-level" information and produces 1024 audio samples per raw_data_block per output channel.

The raw_data_stream supports encoding for both constant rate and variable rate channels. In each case the structure of the bitstream and the operation of the decoder are identical except for some minor qualifications. For constant rate channels, the encoder may have to insert a FIL element to adjust the rate upwards to exactly the desired rate. A decoder reading from a constant rate channel must accumulate a minimum number of bits in its input buffer prior to the start of decoding so that output buffer underrun does not occur. In the case of variable rate, demand read channels, each raw_data_block can have the minimum length (rate) such that the desired audio quality is achieved, and in the decoder there is no minimum input data requirement prior to the start of decoding.

Examples of the simplest possible bitstreams are

bitstream segment	**output signal**
<SCE><TERM><SCE><TERM>…	mono signal
<CPE><TERM><CPE><TERM>…	stereo signal
<SCE><CPE><CPE><LFE><TERM><SCE><CPE><CPE><LFE><TERM>…	5.1 channel signal

where angle brackets (< >) are used to delimit syntactic elements. For the mono signal each SCE must have the same value in its **element_instance_tag**, and similarly, for the stereo signal each CPE must have the same value in its **element_instance_tag**. For the 5.1 channel signal each SCE must have the same value in its **element_instance_tag**, each CPE associated with the front channel pair must have the same value in its **element_instance_tag**, and each CPE associated with the back channel pair must have the same value in its **element_instance_tag**.

If these bitstreams are to be transmitted over a constant rate channel then they might include a fill_element to adjust the instantaneous bit rate. In this case an example of a coded stereo signal is

<CPE><FIL><TERM><CPE><FIL><TERM>…

If the bitstreams are to carry ancillary data and run over a constant rate channel then an example of a coded stereo signal is

<CPE><DSE><FIL><TERM><CPE><DSE><FIL><TERM>…

All data_stream_elements have the same element_instance_tag if they are part of the same data stream.

2.3.4 Decoding of a single_channel_element (SCE), channel_pair_element (CPE) and individual_channel_stream (ICS)

2.3.4.1 Definitions

Bit stream elements:

individual_channel_stream()	contains data necessary to decode one channel (Table 6.12)
ics_info()	contains side information necessary to decode an individual_channel_stream. The individual_channel_streams of a channel_pair_element may share one common ics_info (Table 6.11)
common_window	a flag indicating whether the two individual_channel_streams share a common ics_info or not. In case of sharing, the ics_info is part of the channel_pair_element and must be used for both channels. Otherwise, the ics_info is part of each individual_channel_stream (Table 6.10)
ics_reserved_bit	bit reserved for future use
window_sequence	indicates the sequence of windows as defined in Table 2.2 (Table 6.11)
window_shape	A 1 bit field that determines what window is used for the trailing part of this analysis window (Table 6.11)
max_sfb	number of scalefactor bands transmitted per group (Table 6.11)
scale_factor_grouping	A bit field that contains information about grouping of short spectral data (Table 6.11)

Help elements:

scalefactor window band	term for scalefactor bands within a window, given in table Table 2.3 to Table 2.5
scalefactor band	term for scalefactor band within a group. In case of EIGHT_SHORT_SEQUENCE and grouping a scalefactor band may contain several scalefactor window bands of corresponding frequency. For all other window_sequences scalefactor bands and scalefactor window bands are identical.
g	group index
win	window index within group
sfb	scalefactor band index within group
swb	scalefactor window band index within window
bin	coefficient index
num_window_groups	number of groups of windows which share one set of scalefactors
window_group_length[g]	number of windows in each group.
bit_set(bit_field,bit_num)	function that returns the value of bit number bit_num of a bit_field (most right bit is bit 0)
num_windows	number of windows of the actual window sequence
num_swb_long_window	number of scalefactor bands for long windows. This number has to be selected depending on the sampling frequency. See clause 2.3.8.
num_swb_short_window	number of scalefactor window bands for short windows. This number has to be selected depending on the sampling frequency. See clause 2.3.8.
num_swb	number of scalefactor window bands for shortwindows in case of EIGHT_SHORT_SEQUENCE, number of scalefactor window bands for long windows otherwise
swb_offset_long_window[swb]	table containing the index of the lowest spectral coefficient of scalefactor band sfb for long windows. This table has to be selected depending on the sampling frequency. See clause 2.3.8.
swb_offset_short_window[swb]	table containing the index of the lowest spectral coefficient of scalefactor band sfb for short windows. This table has to be selected depending on the sampling frequency. See clause 2.3.8.
swb_offset[swb]	table containing the index of the lowest spectral coefficient of scalefactor band sfb for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows
sect_sfb_offset[g][section]	table that gives the number of the start coefficient for the section_data() within a group. This offset depends on the window_sequence and scale_factor_grouping.
sampling_frequency_index	see clause 2.1.1.1

2.3.4.2 Decoding process

Single_channel_element and channel_pair_element

A single_channel_element is composed of an element_instance_tag and an individual_channel_stream. In this case ics_info is always located in the individual_channel_stream.

A channel_pair_element begins with an element_instance_tag and common_window flag. If the common_window equals '1', then ics_info is shared amongst the two individual_channel_stream elements and the MS information is transmitted. If common_window equals '0', then there is an ics_info within each individual_channel_stream and there is no MS information.

Decoding an individual_channel_stream (ICS)

In the individual_channel_stream, the order of decoding is:

Get global_gain
Get ics_info (parse bitstream if common information is not present)
Get Section Data
Get Scalefactor Data, if present
Get pulse data if present
Get TNS data, if present
Get gain control data, if present
Get Spectral Data, if present.

The process of recovering pulse_data is described in section 9, tns_data in section 14, and gain_control data in section 16. An overview of how to decode ics_info (clause 8.3), section data (clause 9), scalefactor data (clause 9 and 11), and spectral data (clause 9) will be given here.

Recovering ics_info

For single_channel_elements ics_info is always located immediately after the global_gain in the inidividual_channel_stream. For a channel pair element there are two possible locations for the ics_info. If each individual channel in the pair window switch together then the ics_info is located immediately after common_window in the channel_pair_element() and common_window is set to 1. Otherwise there is an ics_info immediately after global_gain in each of the two individual_channel_stream() in the channel_pair_element and common_window is set to 0.

ics_info carries window information associated with an ICS and thus permits channels in a channel_pair to switch separately if desired. In addition it carries the max_sfb which places an upper limit on the number of ms_used[] and predictor_used[] bits that must be transmitted. If the window_sequence is EIGHT_SHORT_SEQUENCE then scale_factor_grouping is transmitted. If a set of short windows form a group then they share scalefactors as well as intensity stereo positions and have their spectral coefficients interleaved. The first short window is always a new group so no grouping bit is transmitted. Subsequent short windows are in the same group if the associated grouping bit is 1. A new group is started if the associated grouping bit is 0. It is assumed that grouped short windows have similar signal statistics. Hence their spectra are interleaved so as to place correlated coefficients next to each other. The manner of interleaving is indicated in figure 8.3. ics_info also carries the prediction data for the individual channel or channel pair (see clause 13).

Recovering Sectioning Data

In the ICS, the information about one long window, or eight short windows, is recovered. The sectioning data is the first field to be decoded, and describes the Huffman codes that apply to the scalefactor bands in the ICS (see clause 9 and 11). The form of the section data is:

sect_cb The codebook for the section

and

sect_len The length of the section. This length is recovered by reading the bitstream sequentially for a section length, adding the escape value to the total length of the section until a non-escape value is found, which is added to establish the total length of the section. This process is clearly explained in the C-like syntax description. Note that within each group the sections must delineate the scalefactor bands from zero to **max_sfb** so that the first section within each group starts at bands zero and the last section within each group ends at **max_sfb**.

The sectioning data describes the codebook, and then the length of the section using that codebook, starting from the first scalefactor band and continuing until the total number of scalefactor bands is reached.

After this description is provided, all scalefactors and spectral data corresponding to codebook zero are zeroed, and no values corresponding to these scalefactors or spectral data will be transmitted. When scanning for scale-factor data it is important to note that scalefactors for any scalefactor bands whose Huffman codebook is zero

will be omitted. Similarly, all spectral data associated with Huffman codebook zero are omitted (see clause 9 and 11)
In addition spectral data associated with the scalefactor bands that have an intensity codebook will not be transmitted, but intensity steering coefficients will be transmitted in place of the scalefactors, as described in section 12.2

Scalefactor Data Parsing and Decoding
For each scalefactor band that is not in a section coded with the zero codebook (ZERO_HCB), a scalefactor is transmitted. These will be denoted as 'active' scalefactor bands and the associated scalefactors as active scalefactors. Global gain, the first bitstream element in an ICS, is typically the value of the first active scalefactor. All scalefactors (and steering coefficients) are transmitted using Huffman coded DPCM relative to the previous active scalefactor (see clause 9 and 11). The first active scalefactor is differentially coded relative to the global gain. Note that it is not illegal, merely inefficient, to provide a global_gain that is different from the first active scalefactor and then a non-zero DPCM value for the first scalefactor DPCM value. If any intensity steering coefficients are received interspersed with the DPCM scalefactor elements, they are sent to the intensity stereo module, and are not involved in the DPCM coding of scalefactor values (see clause 12.2). The value of the first active scalefactor is usually transmitted as the global_gain with the first DPCM scalefactor having a zero value. Once the scalefactors are decoded to their integer values, the actual values are found via a power function (see clause 11).

Spectral Data Parsing and Decoding
The spectral data is recovered as the last part of the parsing of an ICS. It consists of all the non-zeroed coefficients remaining in the spectrum or spectra, ordered as described in the ICS_info. For each non-zero, non-intensity codebook, the data are recovered via Huffman decoding in quads or pairs, as indicated in the noiseless coding tool (see clause 9). If the spectral data is associated with an unsigned Huffman codebook, the necessary sign bits follow the Huffman codeword (see section 9.3). In the case of the ESCAPE codebook, if any escape value is received, a corresponding escape sequence will appear after that Huffman code. There may be zero, one or two escape sequences for each codeword in the ESCAPE codebook, as indicated by the presence of escape values in that decoded codeword. For each section the Huffman decoding continues until all the spectral values in that section have been decoded. Once all sections have been decoded, the data is multiplied by the decoded scalefactors and deinterleaved if necessary.

2.3.4.3 Windows and window sequences

Quantization and coding is done in the frequency domain. For this purpose, the time signal is mapped into the frequency domain in the encoder. The decoder performs the inverse mapping as described in clause 15. Depending on the signal, the coder may change the time/frequency resolution by using two different windows: LONG_WINDOW and SHORT_WINDOW. To switch between windows, the transition windows LONG_START_WINDOW and LONG_STOP_WINDOW are used. Table 2.1 lists the windows, specifies the corresponding transform length and shows the shape of the windows schematically. Two transform lengths are used: 1024 (referred to as long transform) and 128 coefficients (refered to as short transform).
Window sequences are composed of windows in a way that a raw_data_block always contains data representing 1024 output samples. The bitstream element **window_sequence** indicates the window sequence that is actually used. Table 2.2 lists how the window sequences are composed of individual windows. Refer to clause 15 for more detailed information about the transform and the windows.

2.3.4.4 Scalefactor bands and grouping

Many tools of the decoder perform operations on groups of consecutive spectral values called scalefactor bands (abbreviation 'sfb'). The width of the scalefactor bands is built in imitation of the critical bands of the human auditory system. For that reason the number of scalefactor bands in a spectrum and their width depend on the transform length and the sampling frequency. Table 2.3 to Table 2.5 list the offset to the beginning of each scalefactor band for the transform lengths 1024 and 128 and the different sampling frequencies, respectively.
To reduce the amount of side information in case of sequences which contain SHORT_WINDOWS, consecutive SHORT_WINDOWs may be grouped (see figure 8.1). The information about the grouping is contained in the **scale_factor_grouping** bitstream element. Grouping means that only one set of scalefactors is transmitted for all grouped windows as if there was only one window. The scalefactors are then applied to the corresponding spectral data in all grouped windows. To increase the efficiency of the noiseless coding (see clause 9), the spectral data of a group is transmitted in an interleaved order given in clause 8.3.5. The interleaving is done on a scalefactor band by scalefactor band basis, so that the spectral data can be grouped to form a virtual scalefactor band to which the common scalefactor can be applied. Within this document the expression 'scalefactor band'

(abbreviation ‘sfb’) denotes these virtual scalefactor bands. If the scalefactor bands of the single windows are referred to, the expression ‘scalefactor window band’ (abbreviation ‘swb’) is used. Due to its influence on the scalefactor bands, grouping affects the meaning of section_data (see clause 9), the order of spectral data (see clause 8.3.5), and the total number of scalefactor bands. For a LONG_WINDOW scalefactor bands and scalefactor window bands are identical since there is only one group with only one window.
To reduce the amount of information needed for the transmission of side information specific to each scalefactor band, the bitstream element **max_sfb** is transmitted. Its value is one greater than the highest active scalefactor band in all groups. **max_sfb** has influence on the interpretation of section data (see clause 9), the transmission of scalefactors (see clause 9 and 11), the transmission of predictor data (see clause 13) and the transmission of the ms_mask (see clause 12.1).

Since scalefactor bands are a basic element of the coding algorithm, some help variables and arrays are needed to describe the decoding process in all tools using scalefactor bands. These help variables depend on sampling_frequency, **window_sequence, scalefactor_grouping** and **max_sfb** and must be built up for each raw_data_block. The pseudo code shown below describes

- how to determine the number of windows in a window_sequence *num_windows*
- how to determine the number of window_groups *num_window_groups*
- how to determine the number of windows in each group *window_group_length[g]*
- how to determine the total number of scalefactor window bands *num_swb* for the actual window type
- how to determine *swb_offset[swb]*, the offset of the first coefficient in scalefactor window band *swb* of the window actually used
- how to determine *sect_sfb_offset[g][section]*, the offset of the first coefficient in section *section*. This offset depends on **window_sequence** and **scale_factor_grouping** and is needed to decode the spectral_data().

A long transform window is always described as a window_group containing a single window. Since the number of scalefactor bands and their width depend on the sampling frequency, the affected variables are indexed with sampling_frequency_index to select the appropriate table.

```
fs_index = sampling_frequency_index;
switch( window_sequence ) {
   case ONLY_LONG_SEQUENCE:
   case LONG_START_SEQUENCE:
   case LONG_STOP_SEQUENCE:
      num_windows = 1;
      num_window_groups = 1;
      window_group_length[num_window_groups-1] = 1;
      num_swb = num_swb_long_window[fs_index];
      /* preparation of sect_sfb_offset for long blocks */
      /* also copy the last value! */
      for( i=0; i< max_sfb + 1; i++ ) {
         sect_sfb_offset[0][i] = swb_offset_long_window[fs_index][i];
         swb_offset[i] = swb_offset_long_window[fs_index][i];
      }
      break;
   case EIGHT_SHORT_SEQUENCE:
      num_windows = 8;
      num_window_groups = 1;
      window_group_length[num_window_groups-1] = 1;
      num_swb = num_swb_short_window[fs_index];
      for( i=0; i< num_swb_short_window[fs_index] + 1; i++ )
         swb_offset[i] = swb_offset_short_window[fs_index][i];
      for( i=0; i< num_windows-1; i++) {
         if( bit_set(scale_factor_grouping,6-i)) == 0 ) {
            num_window_groups += 1;
            window_group_length[num_window_groups-1] = 1;
         }
         else {
            window_group_length[num_window_groups-1] += 1;
         }
      }
      /* preparation of sect_sfb_offset for short blocks */
      for( g=0; g<num_window_groups; g++ ) {
         sect_sfb = 0;
         offset = 0;
         for( i=0; i< max_sfb; i++ ) {
            width = swb_offset_short_window[fs_index][i+1] -
                    swb_offset_short_window[fs_index][i];
            width *= window_group_length[g];
```

```
            sect_sfb_offset[g][sect_sfb++] = offset;
            offset += width;
          }
          sect_sfb_offset[g][sect_sfb] = offset;
        }
        break;
    default:
        break;
}
```

2.3.4.5 Order of spectral coefficients in spectral_data

For ONLY_LONG_SEQUENCE windows (num_window_groups = 1, window_group_length[0] = 1) the spectral data is in ascending spectral order, as shown in figure 8.2.

For the EIGHT_SHORT_SEQUENCE window, the spectral order depends on the grouping in the following manner:

- Groups are ordered sequentially
- Within a group, a scalefactor band consists of the spectral data of all grouped SHORT_WINDOWs for the associated scalefactor window band. To clarify via example, the length of a group is in the range of one to eight SHORT_WINDOWs.
    - If there are eight groups each with length one (num_window_groups = 8, window_group_length[0] = 1), the result is a sequence of eight spectrums, each in ascending spectral order.
    - If there is only one group with length eight (num_window_groups = 1, window_group_length[0] = 1), the results is that spectral data of all eight SHORT_WINDOWs is interleaved by scalefactor window bands.
    - Figure 8.3 shows the spectral ordering for an EIGHT_SHORT_SEQUENCE with grouping of SHORT_WINDOWs according to figure 8.1 (num_window_groups = 4).
- Within a scalefactor window band, the coefficients are in ascending spectral order.

2.3.4.6 Output word length

The global gain for each audio channel is scaled such that the integer part of the output of the IMDCT can be used directly as a 16-bit PCM audio output to a digital-to-analog (D/A) converter. This is the default mode of operation and will result in correct audio levels. If the decoder has a D/A converter that has greater than 16-bit resolution then the output of the IMDCT can be scaled up such that the appropriate number of fractional bits are included to form the desired D/A word size. In this case the level of the converter output would be matched to that of a 16-bit D/A, but would have the advantage of greater signal dynamic range and lower converter noise floor. Similarly, shorter D/A word lengths can be accommodated.

2.3.4.7 Use of emphasis

This standard does not support pre-emphasis and de-emphasis and no signalling bits are provided to transport such information in the bitstream.

2.3.4.8 Matrix-mixdown Method

2.3.4.8.1 Description

The matrix-mixdown method applies only for mixing a 3-front/2-back speaker configuration, 5-channel program, down to a stereo or a mono program. It is not applicable to any program with other than the 3/2 configuration and only decoders capable of decoding a 3/2 configuration must be able to decode this mode.

2.3.4.8.2 Definitions

matrix_mixdown_idx_present One bit indicating that a stereo matrix coefficient index is present (see Table 6.21). For all configurations other than the 3/2 format this bit must be zero.

matrix_mixdown_idx A two bit field that indicates that the coefficient to be used in the 5-channel to 2-channel matrix-mixdown. Possible matrix coefficients are listed in section 8.3.8.5.

pseudo_surround_enable One bit indicating that pseudo surround decoding is possible.

2.3.4.8.3 Matrix-mixdown process

A derived stereo signal can be generated within a matrix-mixdown decoder by use of one of the two following sets of equations.

Set 1:

$$L' = \frac{1}{1+1/\sqrt{2}+A} \cdot [L + C/\sqrt{2} + A \cdot L_S]$$

$$R' = \frac{1}{1+1/\sqrt{2}+A} \cdot [R + C/\sqrt{2} + A \cdot R_S]$$

Set 2:

$$L' = \frac{1}{1+1/\sqrt{2}+2 \cdot A} \cdot [L + C/\sqrt{2} - A \cdot (L_S + R_S)]$$

$$R' = \frac{1}{1+1/\sqrt{2}+2 \cdot A} \cdot [R + C/\sqrt{2} + A \cdot (L_S + R_S)]$$

Where L, C, R, LS and RS are the source signals, L' and R' are the derived stereo signals and A is the matrix coefficient indicated by matrix_mixdown_idx. LFE channels are omitted from the mixdown.

If pseudo_surround_enable is not set, then only set 1 should be used. If pseudo_surround_enable is set, then either set 1 or set 2 equations can be used, depending on whether the receiver has facilities to invoke some form of surround synthesis.

As further information it should be noted that one can derive a mono signal using the following equation:

$$M = \frac{1}{3+2 \cdot A} \cdot [L + C + R + A \cdot (L_S + R_S)]$$

2.3.4.8.4 Advisory

The matrix-mixdown provision enables a mode of operation which may be beneficial to some operators in some circumstances. However, it is advised that this method should not be used. The psychoacoustic principles on which the audio coding are based are violated by this form of post-processing, and a perceptually faithful reconstruction of the signal cannot be guaranteed. The preferred method is to use the stereo or mono mixdown channels in the AAC syntax to provide stereo or mono programming which is specifically created by conventional studio mixing prior to bitrate reduction.

The stereo and mono mixdown channels additionally enable the content provider to separately optimize the stereo and multichannel program mixes - this is not possible by using the matrix-mixdown method.

It is additionally relevant to note that, due to the algorithms used for the multichannel and stereo mixdown coding, a better combination of quality and bitrate is usually provided by use of the stereo mixdown channels than can be provided by the matrix-mixdown process.

2.3.4.8.5 Tables

Matrix-mixdown coefficients

matrix_mixdown_idx	A
0	$1/\sqrt{2}$
1	$1/2$
2	$1/(2\sqrt{2})$
3	0

2.3.5 Low Frequency Enhancement Channel (LFE)

2.3.5.1 General

In order to maintain a regular structure of the decoder, the lfe_channel_element is defined as a standard individual_channel_stream(0) element, i.e. equal to a single_channel_element. Thus, decoding can be done using the standard procedure for decoding a single_channel_element.

In order to accomodate a more bitrate and hardware efficient implementation of the LFE decoder, however, several restrictions apply to the options used for the encoding of this element:

- The window_shape field is always set to 0, i.e. sine window (see 6.3, Table 6.11)
- The window_sequence field is always set to 0 (ONLY_LONG_SEQUENCE) (see 6.3, Table 6.11)
- The index of the highest non-zero spectral coefficient present in the element is 12
- No Temporal Noise Shaping is used, i.e. tns_data_present is set to 0 (see 6.3, Table 6.12)
- No prediction is used, i.e. predictor_data_present is set to 0 (see 6.3, Table 6.11)

The presence of LFE channels depends on the profile used. Refer to clause 7 for detailed information.

2.3.6 Data stream element (DSE)

Bitstream elements:

data_byte_align_flag	One bit indicating that a byte aligment is performed within the data stream element (Table 6.20)
count	Initial value for length of data stream (Table 6.20)
esc_count	Incremental value of length of data or padding element (Table 6.20)
data_stream_byte	A data stream byte extracted from bitstream (Table 6.20)

A data element contains any additional data, e.g. auxiliary information, that is not part of the audio information itself. Any number of data elements with the same element_instance_tag or up to 16 data elements with different element_instance_tags are possible. The decoding process of the data element is described in this clause.

Decoding process:

The first syntactic element to be read is the the 1 bit **data_byte_align_flag**. Next is the 8 bit value **count**. It contains the initial byte-length of the data stream. If **count** equals 255, its value is incremented by a second 8 bit value, **esc_count**, this final value represents the number of bytes in the data stream element. If **data_byte_align_flag** is set, a byte alignment is performed. The bytes of the data stream follow.

2.3.7 Fill element (FIL)

Bitstream elements:

count	Initial value for length of fill data (Table 6.22)
esc_count	Incremental value of length of fill data (Table 6.22)
fill_byte	byte to be discarded by the decoder (Table 6.22)

Fill elements have to be added to the bitstream if the bitsum of all audio data together with all additional data is lower than the minimum allowed number of bits in this frame necessary to reach the target bitrate. Under normal conditions fill bits are avoided and free bits are used to fill up the bit reservoir. Only if the bitreservoir is full, fill bits are written. Any number of fill elements are allowed.

Decoding process:

The syntactic element **count** gives the initial value of the length of the fill data. In the same way as for the data element this value is incremented with the value of **esc_count** if **count** equals 15. The resulting number gives the number of **fill_byte**s to be read.

2.3.8 Scalable core + AAC/BSAC elements

The scalable core plus AAC elements provides one way of achieving bit rate scalability. It is based on the calculation of a difference signal between the outüut signal of a core coder and the original input signal. At least one enhancement layer based on the AAC/BSAC based VM modules is used. Joint stereo coding and mixed mono/stereo configurations are possible. All AAC joint stereo modes are available in the combined coder.

2.3.8.1 Definitions

diff_control_lr	Element used in a mono T/F / stereo T/F configuration, to control the interaction of the M-channel with the L and R channel.
stereo_flag	Set, if there is a stereo enhancement stage.
mono_layer	Signals a mono T/F layer.
mono_stereo_flag	Set, if it is the first stereo layer.
last_max_sfb	max_sfb of the previous scalability layer.
last_max_sfb_ms	max_sfb of the previous stereo scalability layer.
core_flag	Set, if a core coder is present

2.3.8.2 Decoding / Specifications

Core coder integration

There is no principal limitation on to which core coder can be used. However, in general the core coder should encode the waveform of the input signal, to allow a useful difference signal to be calculated. For all CELP-type speech coders the post-filter has to be switched off for the use of it's output signal in the scalable coder. If a continuous bitstream is desired (in opposition to a packetized transmission of core stream and enhancement layer streams), common bitstream frames are desirable. In order to allow for these common frames, the T/F modules provide -in addition to the standard 2048 length - also a 1920 samples per frame option. This allows for AAC frame lengths of a multiple of 5 and 10 ms. The following table gives an overview:

Sampling rate (Hz):	96	64	48	32	24	16	8
Frame length (1920, ms)	10	15	20	30	40	60	120

These frame lengths allow for an easy integration of the MPEG-4 VM CELP core, and forto the construction of bitstream frames which integrate speech coders standardized elsewhere, which usually have a freame length of a multiple of 10 ms (information on the integration of some standard CELP coders are given in the informative annex). Some combinations of core coders and T/F coder sampling rates require different numbers of core coder frames and T/F frames to build common integrated frames. An example is given in the transport layer used in the reference software, which is described in section 1.1.

Furthermore also common bitstream frames at sampling rates of 44.1 and 22.05 can be achieved, by adjusting the sampling rate of the core coder to match either AAC frames with either 2048 or 1920 samples window length.

Sampling rates of 12 and 11.025 kHz are possible, if a core coder operating at these sampling rates is used. The available combinations are/will be defined in the element identifier elements. The current status, which is used in the reference software, uses the preliminary table given in the header section 1.2.

The ratio of the sampling rate of the core coder and the sampling rate of the enhancement coder must be an integer, to allow for the upsampling tool defined in section 3 to be used.

The example of a transport multiplex given in section 1.1 allows for a delay optimized multiplex of core and enhancement layer streams. This stream allows the original delay of the core coder to be utilized, if only the core decoder is decoded. The bitstream of the core layer then has to be delayed by the equivivalent amount of the maximum bit buffer allowed.

Enhancement layers

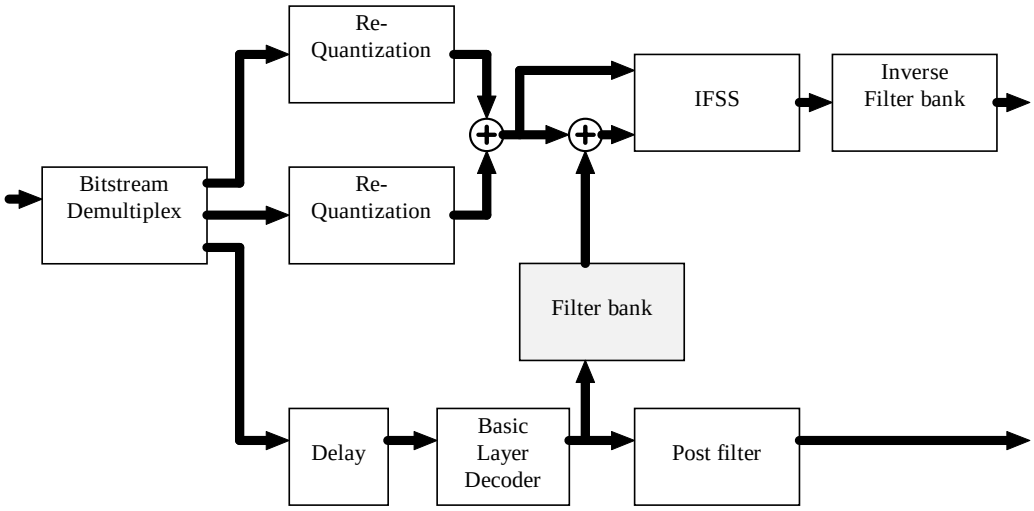
Both, AAC-like enhancement layers, and BSAC scalable coding can be used to enhance the quality of the core stream. In the case of AAC multiple enhancement Quantization&Coding (Q&C) stages are possible by difference encoding in the spectrum.

Up to five enhancement layers are defined in the scalable header syntax (To be changed depending on profiles/levels). The bitrate of the additional layers can be any bit rate possible for the AAC/BSAC enhancement stages.

Different basic configurations are possible:

1.	mono core	mono scal. T/F Q&C	
2.	stereo core	stereo scal. T/F Q&C	
3.	mono core	stereo scal. T/F Q&C	
4.	mono core	mono scal. T/F Q&C	stereo scal. T/F Q&C
5.		mono scal. T/F Q&C	stereo scal. T/F Q&C
6.		mono scal. T/F Q&C	

Figure 1 below shows the basic configuration of the mono / mono or stereo/stereo configuration. In the case of stereo the appropriate inverse joint stereo operations have to be performed before the inverse filter bank is applied.



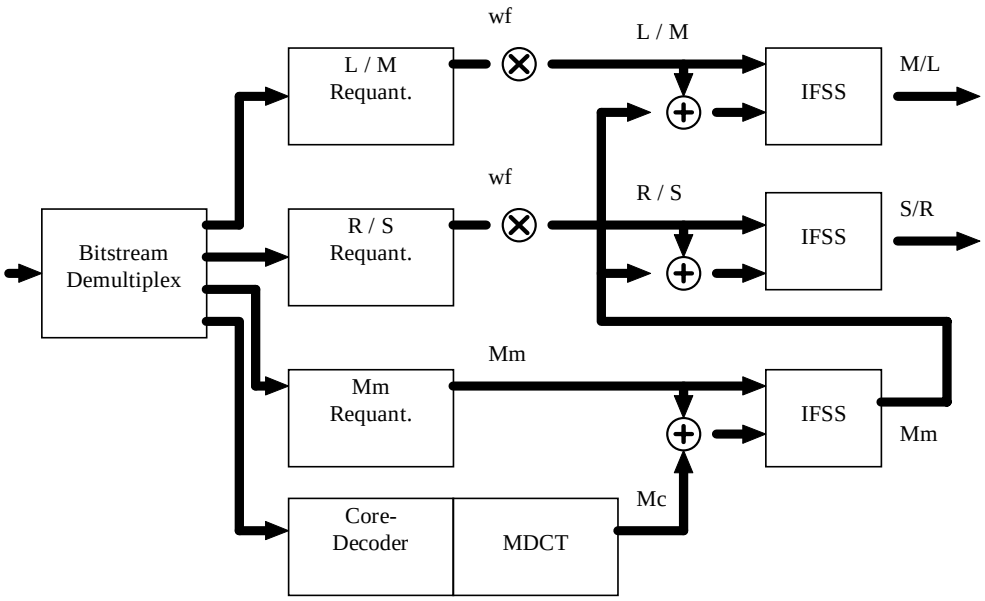
Decoding of the mono/mono and stereo/stereo modes 1-2:

Reconstruct all T/F coded spectra of in all scale factor bands as specified in the „Noiseless Coding“ and „Scalefactors“ or BSAC sections, respectively. Add all these contributions to form the combined T/F spectrum 'S_{TF}'.

If a core channel is present, decode the core and up-sample and transform the decoded core signal into the frequency domain using the upsampling (filterbank) tool to get the core spectrum '$S_{C'}$. Apply the Inverse Frequency Selective Switch (IFSS) tool to get the complete output spectrum '$S_{O'}$. If TNS is used the encoder TNS filter is applied to the core spectrum S_C. The normal decoder TNS-filter is applied to S_O before the calculation of the inverse Filterbank.

In the stereo/stereo case (mode 2) instead of S_{TF} two signals $S_{TF\text{-}L/M}$ and $S_{TF\text{-}R/S}$ are available The usual inverse joint stereo procedures are applied to calculate $S_{TF\text{-}L}$ and $S_{TF\text{-}R}$. These signals are then combined in IFFS(left channel) and IFSS(right channel) with the two core spectra $S_{C\text{-}L}$ and $S_{C\text{-}R}$.to get the output spectra $S_{O\text{-}L}$ and $S_{O\text{-}R}$.

Figure 2 below shows the block diagram for the combined mono/stereo modes 3-5.



Decoding of the mixed mono / stereo modes 3-5:

Reconstruct the spectra M'/S, or L'/R', in all scale factor bands as specified in the „Noiseless Coding" and „Scalefactors" sections or BSAC sections, respectively. If a separate T/F coded M-channel is present in mode 4 and 5 also decode this channel as described above to get the signal Mm.

If (in mode 3 and 4) a core channel is present, decode the core, up-sample and transform into the frequency domain as described above to get the signal Mc.

In mode 4 further apply the Inverse Frequency Selective Switch (IFSS) as described for the „Core + T/F scalability" tool to get the signal Mm'. In this case diff_control_m(win,sb) is used to control the switching. In mode 4 Mm' is identical to Mc. In mode 5 Mm' is identical to Mm.

For all bands where M/S coding is selected only the IFSS information for the M-switch is transmitted in diff_control[0][win][sfb]. Diff_control[1][win][sfb] is not transmitted, but set to 1, in order to avoid processing of the S signal.

Generate the spectra M/S or L/R for each scalefactor band using diff_control(ch,win,sb) to control the switching. If ms_used[sfb] shows L/R encoding, *wf* = 2.0 (for reversing the MS-Matrix factor 0.5), and if it shows M/S coding *wf* = 1.0 is used to modify the spectra before feeding them to the IFFS units.

Finally, on the L/R, or M/S output signals the inverse M/S matrix is applied as described in the „Joint coding" tool to get the final L and R spectra.

2.3.9 VQ single element and VQ scaleable element

2.3.9.1 Definitions

vq_single_element() data element to decode TwinVQ data.
vq_scaleable_element() data element to decode scaleable TwinVQ data.
window_sequence indicates type of window sequence.

num	**window_sequence**
0	ONLY_LONG_SEQUENCE

1	LONG_SHORT_SEQUENCE
2	ONLY_SHORT_SEQUENCE
3	SHORT_LONG_SEQUENCE
4	SHORT_MEDIUM_SEQUENCE
5	MEDIUM_LONG_SEQUENCE
6	LONG_MEDIUM_SEQUENCE
7	MEDIUM_SHORT_SEQUENCE
8	ONLY_MEDIUM_SEQUENCE

After receiving the window sequence information, a parameter BLEN_TYPE is set as listed as follows:

window_sequence	BLEN_TYPE
ONLY_LONG_SEQUENCE LONG_SHORT_SEQUENCE SHORT_LONG_SEQUENCE LONG_MEDIUM_SEQUENCE MEDIUM_LONG_SEQUENCE	LONG
ONLY_MEDIUM_SEQUENCE MEDIUM_SHORT_SEQUENCE SHORT_MEDIUM_SEQUENCE	MEDIUM
ONLY_SHORT_SEQUENCE	SHORT

vq_data() A data block containing one frame of syntax elements for VQ base layer element and VQ enhancement layer element.

optional_info indicates postfilter switch.

fb_shift indicates location of active frequency band in enhancement layer

index_blim_h indicates the upper boundary for the bandwidth control process of the spectrum normalization tool.

index_blim_l indicates the lower boundary for the bandwidth control process of the spectrum normalization tool

index_shape0 indicates the codevector number of the shape codebook 1 and polarity of the codevector for the interleaved vector quantization tool.

index_shape1 indicates the codevector number of the shape codebook 1 of the interleaved vector quantization tool.

index_env indicates the codevector number of the Bark-scale envelope codebook of the spectrum normalization tool.

index_fw_alf indicates the prediction switch of the Bark-scale envelope coding of the spectrum normalization tool.

index_gain indicates the gain factor of the spectrum normalization tool.

index_gain_sb indicates the sub-block gain factor of the spectrum normalization tool.

index_lsp0 indicates LSP MA prediction switch of the spectrum normalization tool.

index_lsp1 indicates codevector number of the first-stage LSP VQ in the spectrum normlization tool.

index_lsp2 indicates codevector number of the second-stage LSP VQ in the spectrum normalization tool.

index_shape0_p indicates the codevector number of codebook 0 of the periodic peak component coding in the spectrum normalization tool.

index_shape1_p indicates the codevector number of codebook 1 of the periodic peak component coding in the spectrum normalization tool.

index_pit indicates the base frequency of the periodic peak component in the spectrum normalization tool.

index_pgain indicates the gain factor of the periodic peak component in the spectrum normalization tool.

2.3.9.2 Parameter setting

Following parameters are used for decoding of VQ base layer element and the VQ enhancement layer element:

N_CH	number of channels defined by the system layer
N_DIV	number of sub-vector division for interleaved vector quantization
N_SF	number of filterbank subblocks in a frame
FW_N_DIV	number of codebook division for the Bark-scale envelope quantization
LSP_SPLIT	number of subvectors for LSP VQ.
ppc_present	switch for the periodic peak component coding
N_DIV_P	number of sub-vector division for periodic peak component coding

These parameters are also referenced from the interleave vector quantization tool and the spectrum normalization tool.

2.3.9.2.1 N_DIV

Number of sub-vector division for interleaved vector quantization, N_DIV is calculated according to the decoder status. This calculation is defined in section 2.3.9.3.2.

2.3.9.2.2 N_SF

Number of filterbank subblocks, N_SF, is set according to the parameters MODE_VQ and BLEN_TYPE as listed below:

MODE_VQ	BLEN_TYPE	N_SF
	LONG	1
24_06 24_06_960 SCL_1 SCL_1_960 SCL_2 SCL_2_960	SHORT	8
16_16 08_06	MEDIUM	2
	SHORT	8

2.3.9.2.3 FW_N_DIV

Number of sub-vector division for the Bark-scale envelope coding, FW_N_DIV, is set according to the parameters MODE_VQ and BLEN_TYPE as listed below:

MODE_VQ	BLEN_TYPE	FW_N_DIV
24_06/ 24_06_960	LONG SHORT	7 1
16_16	LONG MEDIUM SHORT	3 2 1
08_06	LONG MEDIUM SHORT	3 2 1
SCL_1/ SCL_1_960	LONG SHORT	8 1
SCL_2/ SCL_2_960	LONG SHORT	8 1

2.3.9.2.4 LSP_SPLIT

This parameter defines the number of sub-vectors for 2nd-stage LSP vector quantization in spectrum normalization tool. Values are asigned according to the parameter MODE_VQ:

MODE_VQ	LSP_SPLIT
24_06	3
24_06_960	3
16_16	3
08_06	3
SCL_1	3
SCL_1_960	3
SCL_2	3
SCL_2_960	3

2.3.9.2.5 ppc_present

This parameter activates the periodic peak component coding process of the spectrum normalization tool. The value is set as follows:

```
if (BLEN_TYPE == LONG && (MODE_VQ == 16_16 || MODE_VQ == 08_06))
   ppc_present = TRUE;
else
   ppc_present = FALSE;
```

2.3.9.2.6 N_DIV_P

This parameter indicates of number of sub-vector division of the periodic peak component coding in spectrum normalization tool. The value is always set to 2.

2.3.9.3 Bit allocation

2.3.9.3.1 Spectrum normalization tool

For syntax elements listed below, the number of bits is set according to parameters MODE_VQ and BLEN_TYPE:

index_blim_h
index_blim_l
index_env
index_gain
index_gain_sb
index_lsp1
index_lsp2
index_pit
index_pgain

Number of bits are set as follows:

MODE_VQ	BLEN_TYPE	**env**	**blim_h**	**blim_l**	**gain**	**gain_sb**	**lsp1**	**lsp2**
24_06	LONG	6	0	0	9	4	6	4
24_06_960	SHORT	0						
16_16	LONG	6	2	1	8	5	6	4
	MEDIUM	5						
	SHORT	6						

08_06	LONG	6	0	0	8	4	5	3
	MEDIUM	6						
	SHORT	3						
SCL_1	LONG	6	0	0	8	4	6	4
SCL_1_960	SHORT	0						
SCL_2	LONG	6	0	0	7	4	6	4
SCL_2_960	SHORT	0						

The number of bits for index_pit is 9 for MODE_VQ == 16_16, 8 for MODE_VQ == 08_06.
The number of bits for index_pgain is 7 for MODE_VQ == 16_16, 6 for MODE_VQ == 08_06.

2.3.9.3.2 Interleaved vector quantization tool

Parameter N_DIV, Number of bits of shape code index 0, bits0, and number of bits of shape code index 1, bits1, are calculated as following procedure:

First, number of bits for side information, bits_for_side_information is calculated as follows:

```
bits_for_side_information =
   4 + OPT_TBIT + LSP_TBIT + GAIN_TBIT + FW_TBIT + PIT_TBIT+ used_bits
```

where used_bit is number of bits used by tools other than spectrum normalization tool. OPT_TBIT, LSP_TBIT, GAIN_TBIT, FW_TBIT, and PIT_TBIT is number of bits for optional information, lsp coding, gain coding, Bark-scale envelope coding, and periodic peak components coding respectively. They are set as follows:

```
LSP_TBIT = (LSP_BIT0+LSP_BIT1+(LSP_BIT2*LSP_SPLIT)) * N_CH;
if (FW_N_BIT>0){
   FW_TBIT = ((FW_N_BIT * FW_N_DIV + 1) * N_SF) * N_CH;
}
else{
   FW_TBIT = 0;
}

switch(BLEN_TYPE){
   case SHORT:
      OPT_TBIT = 0;
      GAIN_TBIT = (GAIN_BIT + SUB_GAIN_BIT * N_SF) * N_CH;
      PIT_TBIT = 0;
   break;
   case MEDIUM:
      OPT_TBIT = 2;
      GAIN_TBIT = (GAIN_BIT + SUB_GAIN_BIT * N_SF) * N_CH;
      PIT_TBIT = 0;
      break;
   default:
      OPT_TBIT = 2;
      GAIN_TBIT = GAIN_BIT * N_CH;
      if (ppc_present == TRUE}
         PIT_TBIT = PIT_N_BIT + (BASF_BIT + PGAIN_BIT) * N_CH;
      else
         PIT_TBIT = 0;
   break;
}
```

Parameters LSP_BIT0, LSP_BIT1, LSP_BIT2, LSP_SPLIT, FW_N_BIT, FW_N_DIV, GAIN_BIT, SUB_GAIN_BIT, SUB_GAIN_BIT, PIT_N_BIT, BASF_BIT, and PGAIN_BIT is set according to the parameters BLEN_TYPE and MODE_VQ as listed in tables from 3.9.1 to 3.9.7.

Number of available bits, bits_available_vq is calculated as follows:

```
available_vq =
   (int)(FRAME_SIZE * BITRATE/SAMPLING_FREQUENCY) -
   bits_for_side_information
```

Finally, number of shape sub-vector, `N_DIV` and number of bits for shape code indexes, bits0 and bits1, are calculated as follows:

```
N_DIV = ((int)((bits_available_vq + MAXBIT*2-1)/(MAXBIT*2)));
bits = (bits_available_vq + N_DIV - 1 - idiv) / N_DIV;
bits0 = (int)(bits+1) / 2;
bits1 = (int)bits/2;
```

where MAXBIT is maximum number of shape code bit. The MAXBIT is always set to 7.

2.3.10 Decoding Process for BSAC large step scalability

2.3.10.1 Definitions

Bit stream elements:

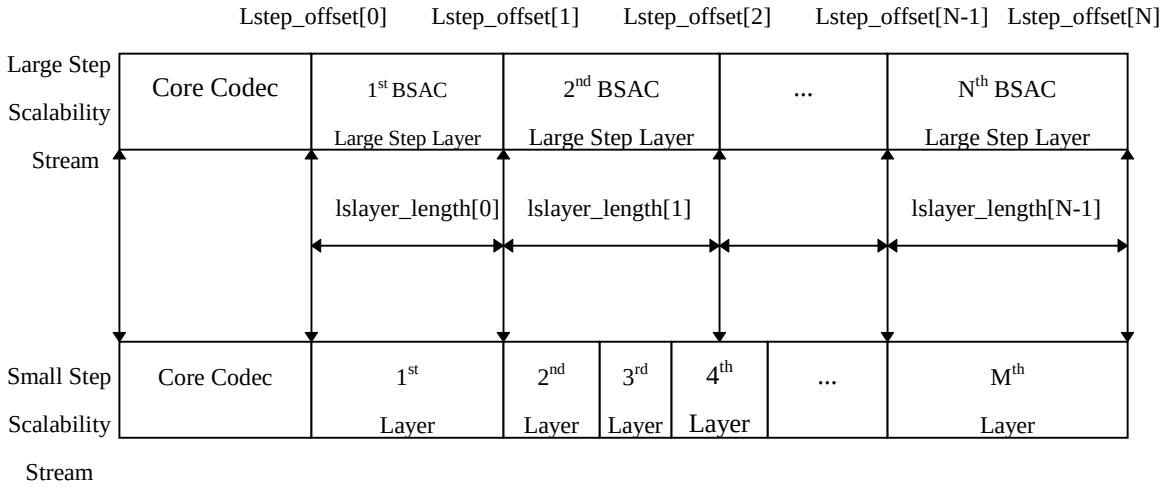
bsac_lstep_data_block()	block of raw data that contains audio data for a time period of 1024(960) samples, related information and other data. A bsac_lstep_data_block() basically consists of several bsac_lstep_stream().
bsac_lstep_stream()	abbreviaton BLSS. Syntactic element of the large step layer bitstream containing coded audio data for a time period of 1024(960) samples, related information and other data.
BSAC_stream_buf[]	a bitstream buffer for large step scalability bitstream. This buffer is mapped to small step scalability bitstream for the actual decoding

Help elements:

data_available()	function that returns ‘1’ as long as data is available, otherwise ‘0
Lstep_offset[]	array containing the offset of the bits to be used in the large step scalability layer. See clause 2.3.10.2
lslayer	large step scalability layer index

2.3.10.2 Decoding process

- In BSAC decoder, large step scalability bitstream can be obtained from each FLEXmuxPDU payload. Large step scalability bitstream should be mapped to small step scalability bitstream, since we decode small step scalability bitstream to make the reconstructed signal.
- In BSAC encoder, small step BSAC bitstream is generated. BSAC would create large overhead if one would try to transmit small step scalability over multiple elementary streams. So, some small step enhancement layers can grouped into a large step enhancement layer in order to behave like large step scalability. The relationship between small step BSAC bitstream and large step bitstream is shown in the following figure.



Mapping of **BSAC_stream_buf[]** from **Large step scalability stream** to **small step scalability stream**

In Encoder, the length of the large step enhancement layer depends upon the number of the large step enhancement layer which the user is going to make. The length of the large step layer is conveyed in the TFSpecificConig : _lslayer_length_ for each large layer.
Since the large step layer bitstream is byte-aligned, its length is represented in the unit of byte.
Lstep_offset[n] is the offset value of nth layer when each large step scalable bitstream is concatenated together. If the large step layer is not the first BSAC layer, Lstep_offset[n] is calculated using the length of the large step layer and the length of the previous layer, Lstep_offset[n-1] as follows :

Lstep_offset[n] (byte) = Lstep_offset[n-1] (byte) + lslayer_length[n]. (byte)

If the large step layer is the first BSAC layer, Lstep_offset[n] depends on the core codec. If the core codec is present, Lstep_offset[0] is initialized to the size of the core codec. Otherwise Lstep_offset[0] is set to 0.

2.3.11 Decoding Process for BSAC small step scalability

2.3.11.1 Definitions

Bit stream elements:

bsac_data_stream()	nch, sampling_frequency_index, frame_length_flag and sequence of syncword and bsac_raw_data_block()'s
nch	a bitstream element that identifies the number of the channel.
sampling_frequency_index	indicates the sampling frequency. See the sampling frequency table in clause 2.1.1.1
frame_length_flag	1 bit flag which specifies the window length of the IMDCT: if set to 0 a 1024 lines IMDCT is used if set to 1 a 960 line IMDCT is used.
bsac_raw_data_block()	block of raw data that contains audio data for a time period of 1024(960) samples, related information and other data. A bsac_raw_data_block() basically consists of bsac_main_stream() and several bsac_layer_stream().
bsac_main_stream()	abbreviaton BMS. Syntactic element of the base layer bitstream containing coded audio data for a time period of 1024(960) samples, related information and other data. There are 2 bitstream elements, identified as bitstream element nch.
bsac_layer_stream()	abbreviaton BLS. Syntactic element of the enhancement layer bitstream containing coded audio data for a time period of 1024(960) samples, related information and other data.
tf_scalable_main_header()	contains header data used for all t/f scalable coding
ms_mask_present	this two bit field (see) indicates that the stereo mask is 00 Independent 01 1 bit mask of max_sfb bands of ms_used is located in the layer side information part. 10 All ms_used are ones 11 2 bit mask of max_sfb bands of stereo_info is located in the layer side information part.

ics_info()	contains side information necessary to decode an bsac_channel_stream. The bsac_channel_stream of a bsac_pair_main_stream may share one common ics_info.
ics_reserved_bit	bit reserved for future use
window_sequence	indicates the sequence of windows as defined in Table 2.2
window_shape	A 1 bit field that determines what window is used for the trailing part of this analysis window
max_sfb	number of scalefactor bands transmitted per group
scale_factor_grouping	A bit field that contains information about grouping of short spectral data

Help elements:

data_available()	function that returns '1' as long as each layer's bitstream is available, otherwise '0'
nch	a bitstream element that identifies the number of the channel.
encoded_layer	top scalability layer index to be encoded. If top layer index, encoded_layer is smaller than 63, the number of the layer is smaller than 64 and is (encoded+1). Otherwise, since the number of the layer is larger than or equal to 64, encoded_layer value is reset to 1000.
scalefactor window band	term for scalefactor bands within a window, given in Table 2.3 to 2.15.
scalefactor band	term for scalefactor band within a group. In case of EIGHT_SHORT_SEQUENCE and grouping a scalefactor band may contain several scalefactor window bands of corresponding frequency. For all other window_sequences scalefactor bands and scalefactor window bands are identical.
g	group index
win	window index within group
sfb	scalefactor band index within group
swb	scalefactor window band index within window
num_window_groups	number of groups of windows which share one set of scalefactors. See clause 2.3.11.4
window_group_length[g]	number of windows in each group. See clause 2.3.11.4
bit_set(bit_field,bit_num)	function that returns the value of bit number bit_num of a bit_field (most right bit is bit 0)
num_windows	number of windows of the actual window sequence. See clause 2.3.11.4
num_swb_long_window	number of scalefactor bands for long windows. This number has to be selected depending on the sampling frequency. See clause 2.4
num_swb_short_window	number of scalefactor window bands for short windows. This number has to be selected depending on the sampling frequency. See clause 2.4
num_swb	number of scalefactor window bands for shortwindows in case of EIGHT_SHORT_SEQUENCE, number of scalefactor window bands for long windows otherwise. See clause 2.3.11.4
swb_offset_long_window[swb]	table containing the index of the lowest spectral coefficient of scalefactor band sfb for long windows. This table has to be selected depending on the sampling frequency. See clause 2.4
swb_offset_short_window[swb]	table containing the index of the lowest spectral coefficient of scalefactor band sfb for short windows. This table has to be selected depending on the sampling frequency. See clause 2.4
swb_offset[swb]	table containing the index of the lowest spectral coefficient of scalefactor band sfb for short windows in case of EIGHT_SHORT_SEQUENCE, otherwise for long windows. See clause 2.3.11.4

2.3.11.2 Decoding process

bsac_raw_data_block

A total BSAC stream, bsac_raw_data_block has the layered structure. First, bsac_main_stream is parsed and decoded which is the bitstream for 1st BSAC scalability layer. Then, bsac_layer_stream for the next enhancement

layer is parsed and decoded. bsac_layer_stream decoding routine is repeated until the decoded bitstream data is available and layer is smaller than or equal to the top layer, **encoded_layer.**

bsac_main_stream

A bsac_main_stream is made up of tf_scalable_main_header, bsac_general_info and bsac_layer_stream().If nch(the number of the channel) is 1, tf_scalable_main_header is parsed whose input parameters, stereo_flag and mono_lay, are 0 and 0, respectively. If nch is 2, tf_scalable_main_header is parsed whose input parameters, stereo_flag and mono_lay, are 1 and 0, respectively. bsac_general_info and bsac_layer_stream are parsed sequentially. And, the decoded samples is reconstructed with the decoded bit-sliced data.
An overview of how to decode bsac_general_info and bsac_layer_stream and reconstruct the decoded samples will be given here.

bsac_layer_stream

A bsac_layer_stream is an enhancement layer bitstream and composed of a bsac_side_info() and bsac_spectral_data(). Decoding process of bsac_layer_stream is as follows :

Decode bsac_side_info
Decode bsac_spectral_data
Reconstruct the decoded samples from the decoded bit-sliced data.

An overview of how to decode bsac_side_info and bsac_spectral_data will be given here. bsac_side_info is made up of as follows :

Decoding of stereo_info, ms_used or noise_flag.
Decoding of scalefactors
Decoding of arithmetic model index

An overview of how to decode stereo_info, scalefactor and arithmetic model index will be given in clause 3.13.

Decoding a tf_scalable_main_header

In the tf_scalable_main_header, the order of decoding is :

Get ics_info()
Get ms_mask_present, if present
Get ltp_data_present
Get ltp data, if present
Get tns_data_present
Get TNS data, if present
Get gain_control_data_present
Get gain control data, if present

If the number of the channel is not 1, the decoding of another channel is done as follows :
Get ltp data, if present
Get tns_data_present
Get TNS data, if present
Get gain_control_data_present
Get gain control data, if present

The process of recovering gain_control_data and tns_data is described in clause 3.12 and 3.8, respectively. An overview of how to decode ics_info will be given in clause 2.3.4.2.

Recovering bsac_general_info

BSAC provides a 1-kits/sec/ch fine granule scalability whose bitstream has the layered structure, one BSAC base layer and several enhancement layers. BSAC base layer contains the common side information for all small step layers, the specific side information for only the base layer and the audio data. The common side information is transmitted in the syntax of bsac_general_info().
bsac_general_info consists of frame_length, encoded_layer, max_scalefactor, scalefactor_model and scf_coding necessary for decoding the scalefactor, min_ArModel and Armodel_model necessary for decoding the arithmetic model and pns_data_present and pns_start_sfb for Perceptual Noise Substitution(pns). All the bitstream elements are included in the form of the unsigned integer.
First, frame length is parsed from syntax. It represents the length of the frame including headers in bytes If the number of the channel is 1, frame length has 10 bits. Otherwise it has 11 bits.

Next, encoded_layer is parsed which preresents the top scalability layer index to be encoded. If top layer index, encoded_layer is smaller than 63, the number of the layer is smaller than 64 and is (encoded+1). Otherwise, since the number of the layer is larger than or equal to 64, encoded_layer value is reset to 1000.
max_scalefactor, scalefactor_model, min_ArModel, ArModel_model and scf_coding are parsed sequentially whose length are 8, 6, 2, 5, 2 and 1bits, respectively. If the number of the channel is not 1, all elements are parsed one more.
And, pns_data_present is parsed from syntax. If the value of the parsed pns_data_present is '1', pns_start_sfb is parsed.

Decoding of stereo_info, noise_flag or ms_used
Decoding process of stereo_info, noise_flag or ms_used is depended on pns_data_present, number of channel, ms_mask_present.

If pns data is not present, decoding process is as follows :
If ms_mask_present is 0, arithmetic decoding of stereo_info or ms_used is not needed.
If ms_mask_present is 2, all ms_used values are ones in this case. So, M/S stereo processing of AAC is done at all scalefactor band.
If ms_mask_present is 1, 1 bit mask of max_sfb bands of ms_used is conveyed in this case. So, ms_used is arithmetic decoded. M/S stereo processing of AAC is done according to the decoded ms_used.
If ms_mask_present is 3, stereo_info is arithmetic decoded. stereo_info is two-bit flag per scalefactor band indicating the M/S coding or Intensity coding mode. If stereo_info is not 0, M/S stereo or intensity stereo of AAC is done with these decoded data.

If pns data is present and the number of channel is 1, decoding process is as follows :
If the number of channel is 1 and pns data is present, noise flag of the scalefactor bands between **pns_start_sfb** to **max_sfb** is arithmetic decoded. Perceptual noise substitution is done according to the decoded noise flag.

If pns data is present and the number of channel is 2, decoding process is as follows :
If ms_mask_present is 0, noise flag for pns is arithmetic decoded. Perceptual noise substitution of independent mode is done according to the decoded noise flag.
If ms_mask_present is 2, all ms_used values are ones in this case. So, M/S stereo processing of AAC is done at all scalefactor band. However, there is no pns processing regardless of pns_data_present flag
If ms_mask_present is 1, 1 bit mask of max_sfb bands of ms_used is conveyed in this case. So, ms_used is arithmetic decoded. M/S stereo processing of AAC is done according to the decoded ms_used. However, there is no pns processing regardless of pns_data_present flag
If ms_mask_present is 3, stereo_info is arithmetic decoded. If stereo_info is 1 or 2, M/S stereo or intensity stereo processing of AAC is done with these decoded data and there is no pns processing. If stereo_info is 3 and scalefactor band is smaller than pns_start_sfb, out_of_phase intensity stereo processing is done. If stereo_info is 3 and scalefactor band is larger than or equal to pns_start_sfb, noise flag for pns is arithmetic decoded. And then if the both noise flags of two channel are 1, noise substitution mode is arithmetic decoded. The perceptual noise is substituted or out_of_phase intensity stereo processing is done according to the substitution mode. Otherwise, the perceptual noise is substituted only if noise flag is 1.

Decoding of scalefactors
The spectral coefficients are divided into scalefactor bands that contain a multiple of 4 quantized spectral coefficients. Each scalefactor band has a scalefactor. The noiseless coding has two ways to represent the scalefactors.

One way is to use coding scheme similar to AAC. For all scalefactors the difference to the preceding value is mapped into new value using Table B.1. If the newly mapped value is smaller than 54, it is arithmetic-coded using the arithmetic model given in Table B.3. Otherwise, the escape value 54 is arithmetic coded using the scalefactor arithmetic model given in Table B.3 and the difference to escape value 54 is arithmetic coded using the arithmetic model given in Table B.4. The initial preceding value is given explicitly as a 8 bit PCM in the bitstream element **max_scalefactor**.

Another way is BSAC scalefactor coding method. For all scalefactors the difference to the offset value is arithmetic-coded using the arithmetic model. The arithmetic model used for coding differential scalefactors is given as a 2-bit unsigned integer in the bitstream element, **scalefactor_model**.

Decoding of arithmetic model index
The spectral coefficients are divided into coding bands which contain 32 quantized spectral coefficients for the noiseless coding. Coding bands are the basic units used for the noiseless coding.
arithmetic model index is the model information used for encoding/decoding the bit-sliced data of each coding band.
For all arithmetic model indexes the difference to the offset value is arithmetic-decoded using the arithmetic model.

Bit-Sliced Spectral Data Parsing and Decoding
A quantized sequence is mapped into a bit-sliced sequence. Four-dimension vectors are formed from the bit-sliced sequence of the quantized spectrum and are divided into two subvectors depending upon the previous states. Noiseless coding of the subvectors relies on the arithmetic model of the coding band, the dimension , the significance of the sub-vector and the previous states.

One- to four-dimensional subvector of bit-sliced sequence are arithmetic coded and transmitted from MSB to LSB, starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient. For the case of multiple windows per block, the concatenated and possibly grouped and interleaved set of spectral coefficients is treated as a single set of coefficients that progress from low to high. This set of spectral coefficients may need to be de-interleaved after they are decoded. The set of bit-sliced sequence is divided into coding bands. The arithmetic model index for encoding the bit-sliced data within each coding band is transmitted starting from the lowest frequency coding band and progressing to the highest frequency coding band. The spectral information for all scalefactor bands equal to or greater than **max_sfb** is set to zero.

Reconstruction of the decoded sample from bit-sliced data
The result of arithmetic decoding each bit-sliced sequence is the codeword index. This index is translated to the bit values as specified in the following pseudo C code:

```
pre_state[] = State that indicates whether the current decoded value is 0 or not.
snf  = the significance of the vector to be decoded.
idx0 = codeword index whose previous states are 0
idx1 = codeword index whose previous states are 1
sample[] = data to be decoded
start_i = start frequence line of the decoded vectors.

for (i=start_i; i < (start_i+4); i++) {
    if (pre_state[i]) {
        if (idx1 & 0x01)
            sample[i] |= (1<<(snf-1))
        idx1 >>= 1;
    }
    else {
        if (idx0 & 0x01)
            sample[i] |= (1<<(snf-1))
        idx0 >>= 1;
    }
}
```

And if the sign bit of the decoded sample is 1, the decoded sample y has the negative value as follows :

```
if (y != 0)
    if (sign_bit == 1)
        y = -y
```

2.3.11.3 Windows and window sequences for BSAC

Quantization and coding is done in the frequency domain. For this purpose, the time signal is mapped into the frequency domain in the encoder. Depending on the signal, the coder may change the time/frequency resolution by using two different windows: LONG_WINDOW and SHORT_WINDOW. To switch between windows, the transition windows LONG_START_WINDOW and LONG_STOP_WINDOW are used. Refer to clause 2.3.4.3 for more detailed information about the transform and the windows as BSAC has the same transform and windows with AAC.

2.3.11.4 Scalefactor bands, grouping and coding bands for BSAC

Many tools of the AAC/BSAC decoder perform operations on groups of consecutive spectral values called scalefactor bands (abbreviation _sfb_). The width of the scalefactor bands is built in imitation of the critical bands of the human auditory system. For that reason the number of scalefactor bands in a spectrum and their width depend on the transform length and the sampling frequency. Refer to clause 2.3.4.4 for more detailed information about the scalefactor bands and grouping as BSAC has the same process with AAC.
BSAC decoding tool performs operations on groups of consecutive spectral values called coding bands (abbreviation _cband_). To increase the efficiency of the noiseless coding, the width of the coding bands is fixed as 32 irrespective of the transform length and the sampling frequency. In case of sequences which contain LONG_WINDOW, 32 spectral data are simply grouped into a coding band. Since the spectral data are transmitted in an interleaved order in case of sequences which contain SHORT_WINDOWs, the interleaved spectral data are grouped into a coding band. Each spectral index is mapped into a coding band with a mapping function, index2cb(ch, i), which returns the coding band using the mapping table index2cband[][] in case of EIGHT_SHORT_SEQUENCE, otherwise i/32. The mapping table depends on **window_sequence** and **scalefactor_grouping.**

Since scalefactor bands and coding bands are a basic element of the BSAC coding algorithm, some help variables and arrays are needed to describe the decoding process in all tools using scalefactor bands and coding bands. These help variables must be defined for BSAC decoding. These help variables depend on sampling_frequency, **window_sequence, scalefactor_grouping** and **max_sfb** and must be built up for each bsac_raw_data_block. The pseudo code shown below describes

- how to determine the number of windows in a window_sequence *num_windows*
- how to determine the number of window_groups *num_window_groups*
- how to determine the number of windows in each group *window_group_length[g]*
- how to determine the total number of scalefactor window bands *num_swb* for the actual window type
- how to determine *swb_offset[swb]*, the offset of the first coefficient in scalefactor window band *swb* of the window actually used
- how to determine *index2cband[i],* the mapping table from the spectral index to the coding band . This mapping table depends on **window_sequence** and **scale_factor_grouping** and is needed to decode the bsac_spectral_data().

A long transform window is always described as a window_group containing a single window. Since the number of scalefactor bands and their width depend on the sampling frequency, the affected variables are indexed with sampling_frequency_index to select the appropriate table.

```
fs_index = sampling_frequency_index;
switch( window_sequence ) {
   case ONLY_LONG_SEQUENCE:
   case LONG_START_SEQUENCE:
   case LONG_STOP_SEQUENCE:
      num_windows = 1;
      num_window_groups = 1;
      window_group_length[num_window_groups-1] = 1;
      num_swb = num_swb_long_window[fs_index];
      for( sfb=0; sfb< max_sfb+1; sfb++ ) {
           swb_offset[sfb] = swb_offset_long_window[fs_index][sfb];
      }

      /* preparation of index2cband for long blocks */
      for( sfb=0; sfb< max_sfb; sfb++ ) {
         for (i= swb_offset[sfb]; i< swb_offset[sfb+1]; i+=4){
            index2cband[i] = i / 32;
      }

      break;
   case EIGHT_SHORT_SEQUENCE:
```

```
      num_windows = 8;
      num_window_groups = 1;
      window_group_length[num_window_groups-1] = 1;
      num_swb = num_swb_short_window[fs_index];
      for( i=0; i< num_windows-1; i++) {
         if( bit_set(scale_factor_grouping,6-i)) == 0 ) {
            num_window_groups += 1;
            window_group_length[num_window_groups-1] = 1;
         }
         else {
            window_group_length[num_window_groups-1] += 1;
         }
      }

      startRegion[0] = 0;
      endRegion[num_window_groups-1] = 8;
      for( i=0; i< num_window_groups-1; i++) {
         endRegion[i] = startRegion[i] + window_group_length[i];
         startRegion[i+1] = endRegion[i];
      }

      swb_offset[0] = 0;
      b = 1
      for(i = 0; i < max_sfb; i++) {
         for(w = 0; w < num_window_groups; w++, b++) {
            width = swb_offset_short_window[fs_index][i+1]
            width -= swb_offset_short_window[fs_index][i]
            width *= window_group_length[w];
            swb_offset[b] = swb_offset[b-1] + width;
         }
      }

      /* preparation of index2cband for short blocks */
      for(qband=0; qband<max_sfb; qband++) {
         for (i=swb_offset[qband]; i<swb_offset[qband+1]; i+=4){
            for(w=0; w<num_window_groups; w++)  {
               cband = i*(endRegion[w]-startRegion[w])/32;
               for (b=startRegion[w]; b<endRegion[w]; b++) {
                  for (k=0; k<4; k++) {
                     tempband0[128*b+i+k] = 24*w+cband;
                  }
               }
            }
            j = 0;
            for(i = 0; i < swb_offset_short[maxSfb]; i+=4) {
               for(b = 0; b < 8; b++) {
                  for(k = 0; k < 4; k++, j++)
                     index2cband[j] = tempband0[128*b+i+k];
               }
            }
         }
      }
      break;

   default:
      break;
}
```

2.3.11.5 BSAC small step scalability layer

BSAC provides a 1-kits/sec/ch fine granule scalability whose bitstream has the layered structure, one BSAC base layer and various enhancement layers. BSAC base layer is made up of the common side information for all small step layers, the specific side information for only the base layer and the audio data. BSAC enhancement layers contain the layer side information and the audio data.
In order to provide the small step scalability, BSAC has the fixed band-limit according to the small step layer. Table 2.18 and Table 2.19 list the scalefactor band offset to the band-limit of each layer for the transform lengths 1024(960) and 128(120) and the different sampling frequencies, respectively. Table 2.16 and Table 2.17 list the spectral component offset to the band-limit of each layer for the transform lengths 1024(960) and 128(120) and the different sampling frequencies, respectively.

Some help variables are needed to describe the BSAC decoding process. These help variables depend on sampling_frequency, **nch** and **frame_length** and must be built up for each bsac_raw_data_block. The pseudo code shown below describes

- how to determine *layer_length*, the length of each small step enhancement layer :
  `layer_length = BLOCK_SIZE_SAMPLES_IN_FRAME *1000 * nch / SAMPLING_FREQUENCY`
- how to determine *layer_index[]*, the spectral component offset to the band-limit of each layer
- how to determine *available_bits[0]*, the maximum available bits to be used in the BSAC base layer

```
available_bits[0] = base_layer_bitrate * layer_length / 1000
if (available_bits[0] > frame_length*8)
    available_bits[0] = frame_length*8
```

  where, base_layer_bitrate is 16000 bits/s.
- initialization of *layer_index[ch][0]*, the spectral component offset to the band-limit of the base layer
  `layer_index[ch][0] = 0`
- initialization of *layer_sfb[ch][0]*, the scalefactor band offset to the band-limit of the base layer
  `layer_sfb[ch][0] = 0`

And, some help variables and arrays are needed to describe the bit-sliced decoding process of the side information and spectral data in each BSAC small step layer. These help variables depend on sampling_frequency, layer, **nch, frame_length, encoded_layer, window_sequence** and **max_sfb** and must be built up for each bsac_layer_stream. The pseudo code shown below describes

- how to determine *available_bits[i]*, the available maximum size of the bitstream from the BSAC base layer to the *i*-th layer.
- how to determine *layer_sfb[][]*, the scalefactor band offset to the band-limit of each layer
- how to determine *layer_index[]*, the spectral component offset to the band-limit of each layer
- how to determine *last_index*, the highest spectral index of **nch** channel band-limits

```
layer = BSAC_small_step_layer_index
available_bits[layer+1] = available_bits[layer] + layer_length
if (available_bits[layer+1] > frame_length*8)
    available_bits[layer+1] = frame_length*8

fs_index = sampling_frequency_index
last_index = 0

for(ch = 0; ch < nch; ch++) {
    switch( window_sequence ) {
        case ONLY_LONG_SEQUENCE:
        case LONG_START_SEQUENCE:
        case LONG_STOP_SEQUENCE:
            if (layer_sfb_offset_long [fs_index][layer] < max_sfb[ch] && layer<encoded_layer)
                layer_sfb[ch][layer+1] = layer_sfb_offset_long[fs_index][layer];
            else
                layer_sfb[ch][layer+1] = max_sfb[ch];

            sfb = layer_sfb[ch][layer+1];
            layer_index[ch][layer+1] = layer_index_offset_long[fs_index][layer];
            if (swb_offset[ch][sfb] < layer_index[ch][layer+1])
                    layer_index[ch][layer+1] = swb_offset[ch][sfb];
        break;


        case EIGHT_SHORT_SEQUENCE:
            if (layer_sfb_offset_short[fs_index][layer] < max_sfb[ch] && layer<encoded_layer)
                layer_sfb[ch][layer+1] = layer_sfb_offset_short[fs_index][layer];
            else
                layer_sfb[ch][layer+1] = max_sfb[ch];

            sfb = layer_sfb[ch][layer+1] * num_window_groups[ch];
            layer_index[ch][layer+1] = layer_index_offset_short[fs_index][layer];
            if (swb_offset[ch][sfb] < layer_index[ch][layer+1])
```

```
                layer_index[ch][layer+1] = swb_offset[ch][sfb];
            break;

        default:
            break;
    }

    /*  find last index  */
    qband = layer_sfb[ch][layer+1] * num_window_groups[ch]
    if(last_index < swb_offset[ch][qband])
        last_index = swb_offset[ch][qband]
}
```

BSAC scalable coding scheme has the fixed band-limit according to the small step layer. The spectral band is extended more and more as the number of the enhancement layer is increased. So, the new spectral componets are added to be decoded in each layer. some help variables and arrays are needed to describe the bit-sliced decoding process of the spectral values in each BSAC small step layer. cur_snf[ch][*i*] is initialized as the allocated bit to the coding band *cband*, Abit[ch][cband] as shown below, where we can get Abit[][] from **ArModel[ch][cband]** and map *i* into *cband* using the function *index2cb(ch, i)*. And, we need the offset significance to start the decoding of the bit-sliced data in each layer. The maximum significance, *maxsnf*, is used as the offset.
These help variables and arrays must be built up for each bsac_spectral_data(). The pseudo code shown below describes

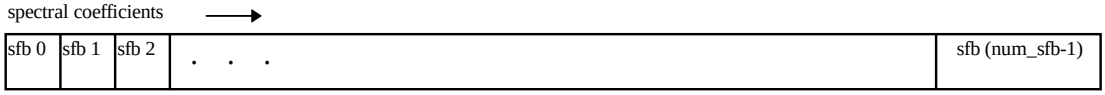
- how to initialize *cur_snf[][]*, the current significance of the 4-dimensional vectors to be added newly. due to the spectral band extension in each enhancement scalability layer.
- how to determine *maxsnf*, the maximum significance of all vectors to be decoded.

```
maxsnf = 0;
for(ch = 0; ch < nch; ch++) {
    /* set current snf */
    for(sfb=layer_sfb[ch][layer]; sfb<layer_sfb[ch][layer+1]; sfb++) {
        for(w = 0; w < num_window_groups[ch]; w++) {
            qband = (sfb * num_window_groups[ch]) + w
            for (i=swb_offset [ch][qband]; i<swb_offset[ch][qband+1]; i+=4) {
                cband = index2cb(ch, i);
                cur_snf[ch][i] = Abit[ch][cband]
            }
        }
    }

    /*  find maximum snf  */
    qband = layer_sfb[ch][layer+1] * num_window_groups[ch]
    for(i = 0; i< swb_offset[ch][qband]; i+=4)
        if (maxsnf < cur_snf[ch][i]) maxsnf = cur_snf[ch][i]
}
```

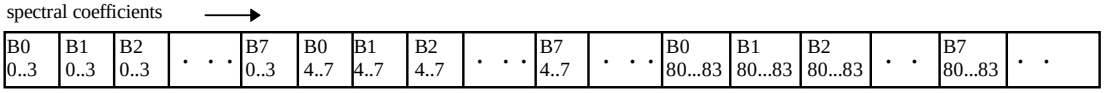
2.3.11.6 Order of spectral coefficients in spectral_data

For ONLY_LONG_SEQUENCE windows (num_window_groups = 1, window_group_length[0] = 1) the spectral data is in ascending spectral order, as shown in the following :



Order of scalefactor bands for ONLY_LONG_SEQUENCE

For the EIGHT_SHORT_SEQUENCE window, each 4 spectral data in each block is interleaving in ascending spectral order, as shown in the following :



Order of spectral data for EIGHT_SHORT_SEQUENCE

2.4 Tables

Table 2.1 – Transform windows (for 48 kHz)

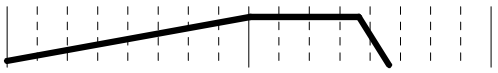
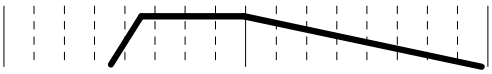
window	num_swb	#coeffs	looks like
LONG_WINDOW	49	1024	
SHORT_WINDOW	14	128	
LONG_START_WINDOW	49	1024	
LONG_STOP_WINDOW	49	1024	

Table 2.2 – Window Sequences

value	window_sequence	num_ windows	looks like
0	ONLY_LONG_SEQUENCE = LONG_WINDOW	1	
1	LONG_START_SEQUENCE = LONG_START_WINDOW	1	
2	EIGHT_SHORT_SEQUENCE = 8 * SHORT_WINDOW	8	
3	LONG_STOP_SEQUENCE = LONG_STOP_WINDOW	1	

Table 2.3 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 44.1 and 48 kHz

fs [kHz]	44.1,48
num_swb_long_ window	49
swb	swb_offset_long _window
0	0
1	4
2	8
3	12
4	16
5	20

swb	swb_offset_long_ window
25	216
26	240
27	264
28	292
29	320
30	352

6	24
7	28
8	32
9	36
10	40
11	48
12	56
13	64
14	72
15	80
16	88
17	96
18	108
19	120
20	132
21	144
22	160
23	176
24	196

31	384
32	416
33	448
34	480
35	512
36	544
37	576
38	608
39	640
40	672
41	704
42	736
43	768
44	800
45	832
46	864
47	896
48	928
	1024 (960)

Table 2.4 – scalefactor bands for a window length of 256 and 240 (values for 240 in brackets) for SHORT_WINDOW at 32, 44.1 and 48 kHz

fs [kHz]	32,44.1,48
num_swb_short_window	14
swb	swb_offset_short_window
0	0
1	4
2	8
3	12
4	16
5	20
6	28
7	36

swb	swb_offset_short_window
8	44
9	56
10	68
11	80
12	96
13	112
	128 (120)

Table 2.5 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 32 kHz

fs [kHz]	32
num_swb_long_window	51
swb	swb_offset_long_window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32
9	36
10	40
11	48
12	56
13	64
14	72
15	80
16	88
17	96
18	108
19	120
20	132

swb	swb_offset_long_window
26	240
27	264
28	292
29	320
30	352
31	384
32	416
33	448
34	480
35	512
36	544
37	576
38	608
39	640
40	672
41	704
42	736
43	768
44	800
45	832
46	864

21	144
22	160
23	176
24	196
25	216

47	896
48	928
49	960
50	992 (960)
	1024 (960)

Table 8.6 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 8 kHz

fs [kHz]	8
num_swb_long_ window	40
swb	swb_offset_long _window
0	0
1	12
2	24
3	36
4	48
5	60
6	72
7	84
8	96
9	108
10	120
11	132
12	144
13	156
14	172
15	188
16	204
17	220
18	236
19	252
20	268

swb	swb_offset_long_ window
21	288
22	308
23	328
24	348
25	372
26	396
27	420
28	448
29	476
30	508
31	544
32	580
33	620
34	664
35	712
36	764
37	820
38	880
39	944
	1024 (960)

Table 8.7 – scalefactor bands for a window length of 256 and 240 (values for 240 in brackets) for SHORT_WINDOW at 8 kHz

fs [kHz]	8
num_swb_short_ window	15
swb	swb_offset_short _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28

swb	swb_offset_short_ window
8	36
9	44
10	52
11	60
12	72
13	88
14	108
	128 (120)

Table 2.8 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 11.025, 12 and 16 kHz

fs [kHz]	11.025, 12, 16
num_swb_long_ window	43
swb	swb_offset_long

swb	swb_offset_long_

	_window
0	0
1	8
2	16
3	24
4	32
5	40
6	48
7	56
8	64
9	72
10	80
11	88
12	100
13	112
14	124
15	136
16	148
17	160
18	172
19	184
20	196
21	212

	window
22	228
23	244
24	260
25	280
26	300
27	320
28	344
29	368
30	396
31	424
32	456
33	492
34	532
35	572
36	616
37	664
38	716
39	772
40	832
41	896
42	960
	1024 (960)

Table 2.9 – scalefactor bands for a window length of 256 and 240 (values for 240 in brackets) for SHORT_WINDOW at 11.025, 12 and 16 kHz

fs [kHz]	11.025, 12, 16
num_swb_short_ window	15
swb	swb_offset_short _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28

swb	swb_offset_short_ window
8	32
9	40
10	48
11	60
12	72
13	88
14	108
	128 (120)

Table 2.10 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 22.05 and 24 kHz

fs [kHz]	22.05 and 24
num_swb_long_ window	47
swb	swb_offset_long _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32

swb	swb_offset_long_ window
24	160
25	172
26	188
27	204
28	220
29	240
30	260
31	284
32	308

9	36
10	40
11	44
12	52
13	60
14	68
15	76
16	84
17	92
18	100
19	108
20	116
21	124
22	136
23	148

33	336
34	364
35	396
36	432
37	468
38	508
39	552
40	600
41	652
42	704
43	768
44	832
45	896
46	960
	1024 (960)

Table 2.11 – scalefactor bands for a window length of 256 and 240 (values for 240 in brackets) for SHORT_WINDOW at 22.05 and 24 kHz

fs [kHz]	22.05 and 24
num_swb_short_ window	15
swb	swb_offset_short _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28

swb	swb_offset_short_ window
8	36
9	44
10	52
11	64
12	76
13	92
14	108
	128 (120)

Table 2.12 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 64 kHz

fs [kHz]	64
num_swb_long_ window	47
swb	swb_offset_long _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32
9	36
10	40
11	44
12	48
13	52
14	56
15	64
16	72

swb	swb_offset_long_ window
24	172
25	192
26	216
27	240
28	268
29	304
30	344
31	384
32	424
33	464
34	504
35	544
36	584
37	624
38	664
39	704
40	744

17	80
18	88
19	100
20	112
21	124
22	140
23	156

41	784
42	824
43	864
44	904
45	944
46	984 (960)
	1024 (960)

Table 2.13 – scalefactor bands for a window length of 256 and 240 (values for 240 in brackets) for SHORT_WINDOW at 64 kHz

fs [kHz]	64
num_swb_short_ window	12
swb	swb_offset_short _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24

swb	swb_offset_short_ window
7	32
8	40
9	48
10	64
11	92
	128 (120)

Table 2.14 – scalefactor bands for a window length of 2048 and 1920 (values for 1920 in brackets) for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 88.2 and 96 kHz

fs [kHz]	88.2 and 96
num_swb_long_ window	41
swb	swb_offset_long _window
0	0
1	4
2	8
3	12
4	16
5	20
6	24
7	28
8	32
9	36
10	40
11	44
12	48
13	52
14	56
15	64
16	72
17	80
18	88
19	96
20	108

swb	swb_offset_long_ window
21	120
22	132
23	144
24	156
25	172
26	188
27	212
28	240
29	276
30	320
31	384
32	448
33	512
34	576
35	640
36	704
37	768
38	832
39	896
40	960
	1024 (960)

Table 2.15 – scalefactor bands for a window length of 256 and 240 (values for 240 in brackets) for SHORT_WINDOW at 88.2 and 96 kHz

fs [kHz]	88.2 and 96
num_swb_short_	12

window	
swb	swb_offset_short_window
0	0
1	4
2	8
3	12
4	16
5	20
6	24

swb	swb_offset_short_window
7	32
8	40
9	48
10	64
11	92
	128 (120)

Table 2.16 BSAC layer index for a window length of 2048 and 1920 for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 48, 44.1, 32, 24, 22.05, 16, 12, 11.025, 8 kHz

layer	48 kHz	44.1 kHz	32 kHz	24 kHz	22.05 kHz	16 kHz	12 kHz	11.025 kHz	8 kHz
0	160	176	240	336	364	492	664	716	1024(960)
1	168	184	264	348	396	532	716	772	1024(960)
2	180	192	292	364	432	572	772	832	1024(960)
3	192	208	320	380	468	616	832	896	1024(960)
4	200	216	336	396	488	664	896	960	1024(960)
5	212	232	352	432	508	716	960	960	1024(960)
6	224	240	368	468	540	772	960	1024(960)	1024(960)
7	232	256	384	508	572	832	1024(960)	1024(960)	1024(960)
8	244	264	416	544	600	896	1024(960)	1024(960)	1024(960)
9	256	280	440	576	632	960	1024(960)	1024(960)	1024(960)
10	264	288	464	608	664	960	1024(960)	1024(960)	1024(960)
11	276	304	488	652	696	960	1024(960)	1024(960)	1024(960)
12	288	312	512	704	728	1024(960)	1024(960)	1024(960)	1024(960)
13	296	320	536	736	768	1024(960)	1024(960)	1024(960)	1024(960)
14	308	336	560	768	808	1024(960)	1024(960)	1024(960)	1024(960)
15	320	352	584	800	848	1024(960)	1024(960)	1024(960)	1024(960)
16	328	360	608	832	896	1024(960)	1024(960)	1024(960)	1024(960)
17	340	368	624	848	912	1024(960)	1024(960)	1024(960)	1024(960)
18	352	384	640	864	928	1024(960)	1024(960)	1024(960)	1024(960)
19	360	392	656	880	944	1024(960)	1024(960)	1024(960)	1024(960)
20	372	408	672	896	960	1024(960)	1024(960)	1024(960)	1024(960)
21	384	416	696	912	960	1024(960)	1024(960)	1024(960)	1024(960)
22	392	424	720	928	992 (960)	1024(960)	1024(960)	1024(960)	1024(960)
23	404	440	744	944	992 (960)	1024(960)	1024(960)	1024(960)	1024(960)
24	416	456	768	960	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
25	424	464	776	960	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
26	436	472	788	992 (960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
27	448	488	800	992 (960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
28	456	496	808	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
29	468	512	820	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
30	480	520	832	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
31	488	528	848	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
32	500	544	864	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
33	512	560	896	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
34	520	568	928	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
35	532	576	960	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
36	544	592	992 (960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
37	552	600	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
38	564	616	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
39	576	624	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
40	584	632	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
41	596	648	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
42	608	664	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
43	616	672	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
44	628	680	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)

45	640	696	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
46	648	704	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
47	660	720	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
48	672	728	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)
> 48	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)	1024(960)

Table 2.17 BSAC layer index for a window length of 256 and 240 for SHORT_WINDOW at 48, 44.1, 32, 24, 22.05, 16, 12, 11.025, 8 kHz

layer	48 kHz	44.1 kHz	32 kHz	24 kHz	22.05 kHz	16 kHz	12 kHz	11.025 kHz	8 kHz
0	20	20	28	36	44	60	80	88	128(120)
1	20	20	32	40	48	64	88	96	128(120)
2	20	24	36	44	52	68	96	104	128(120)
3	24	24	40	44	56	76	104	112	128(120)
4	24	24	40	48	60	80	112	120	128(120)
5	24	28	44	52	60	88	120	120	128(120)
6	28	28	44	56	64	96	120	128(120)	128(120)
7	28	32	48	60	68	104	128(120)	128(120)	128(120)
8	28	32	52	64	68	112	128(120)	128(120)	128(120)
9	32	36	52	72	72	120	128(120)	128(120)	128(120)
10	32	36	56	72	76	120	128(120)	128(120)	128(120)
11	32	36	60	80	80	120	128(120)	128(120)	128(120)
12	36	40	64	88	88	128(120)	128(120)	128(120)	128(120)
13	36	40	64	92	88	128(120)	128(120)	128(120)	128(120)
14	36	40	68	96	92	128(120)	128(120)	128(120)	128(120)
15	40	44	72	100	96	128(120)	128(120)	128(120)	128(120)
16	40	44	76	100	96	128(120)	128(120)	128(120)	128(120)
17	40	44	76	104	100	128(120)	128(120)	128(120)	128(120)
18	44	48	80	108	104	128(120)	128(120)	128(120)	128(120)
19	44	48	80	108	108	128(120)	128(120)	128(120)	128(120)
20	44	52	84	112	112	128(120)	128(120)	128(120)	128(120)
21	48	52	84	112	120	128(120)	128(120)	128(120)	128(120)
22	48	52	88	116	120	128(120)	128(120)	128(120)	128(120)
23	48	56	92	116	120	128(120)	128(120)	128(120)	128(120)
24	52	56	96	120	120	128(120)	128(120)	128(120)	128(120)
25	52	56	96	120	128(120)	128(120)	128(120)	128(120)	128(120)
26	52	60	96	120	128(120)	128(120)	128(120)	128(120)	128(120)
27	56	60	100	120	128(120)	128(120)	128(120)	128(120)	128(120)
28	56	64	100	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
29	56	64	100	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
30	60	64	104	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
31	60	68	104	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
32	60	68	108	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
33	64	68	112	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
34	64	72	116	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
35	64	72	120	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
36	68	76	124	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
37	68	76	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
38	68	76	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
39	72	80	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
40	72	80	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
41	72	80	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
42	76	84	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
43	76	84	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
44	76	84	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
45	80	88	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
46	80	88	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
47	80	88	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
48	84	92	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)
> 48	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)	128(120)

Table 2.18 BSAC layer scalefactor band for a window length of 2048 and 1920 for LONG_WINDOW, LONG_START_WINDOW, LONG_STOP_WINDOW at 48, 44.1, 32, 24, 22.05, 16, 12, 11.025, 8 kHz

layer	48 kHz	44.1 kHz	32 kHz	24 kHz	22.05 kHz	16 kHz	12 kHz	11.025 kHz	8 kHz
0	22	23	26	33	34	33	37	38	40
1	23	24	27	34	35	34	38	39	40
2	24	24	28	34	36	35	39	40	40
3	24	25	29	35	37	36	40	41	40
4	25	25	30	35	38	37	41	42	40
5	25	26	30	36	38	38	42	42	40
6	26	26	31	37	39	39	42	43	40
7	26	27	31	38	40	40	43	43	40
8	27	27	32	39	40	41	43	43	40
9	27	28	33	40	41	42	43	43	40
10	27	28	34	41	42	42	43	43	40
11	28	29	35	41	42	42	43	43	40
12	28	29	35	42	43	43	43	43	40
13	29	29	36	43	43	43	43	43	40
14	29	30	37	43	44	43	43	43	40
15	29	30	38	44	45	43	43	43	40
16	30	31	38	44	45	43	43	43	40
17	30	31	39	45	46	43	43	43	40
18	30	31	39	45	46	43	43	43	40
19	31	32	40	45	46	43	43	43	40
20	31	32	40	45	46	43	43	43	40
21	31	32	41	46	46	43	43	43	40
22	32	33	42	46	47	43	43	43	40
23	32	33	43	46	47	43	43	43	40
24	32	34	43	46	47	43	43	43	40
25	33	34	44	46	47	43	43	43	40
26	33	34	44	47	47	43	43	43	40
27	33	35	44	47	47	43	43	43	40
28	34	35	45	47	47	43	43	43	40
29	34	35	45	47	47	43	43	43	40
30	34	36	45	47	47	43	43	43	40
31	35	36	46	47	47	43	43	43	40
32	35	36	46	47	47	43	43	43	40
33	35	37	47	47	47	43	43	43	40
34	36	37	48	47	47	43	43	43	40
35	36	37	49	47	47	43	43	43	40
36	36	38	50	47	47	43	43	43	40
37	37	38	51	47	47	43	43	43	40
38	37	39	51	47	47	43	43	43	40
39	37	39	51	47	47	43	43	43	40
40	38	39	51	47	47	43	43	43	40
41	38	40	51	47	47	43	43	43	40
42	38	40	51	47	47	43	43	43	40
43	39	40	51	47	47	43	43	43	40
44	39	41	51	47	47	43	43	43	40
45	39	41	51	47	47	43	43	43	40
46	40	41	51	47	47	43	43	43	40
47	40	42	51	47	47	43	43	43	40
48	40	42	51	47	47	43	43	43	40
> 48	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb

Table 2.19 BSAC layer scalefactor band for a window length of 256 and 240 for SHORT_WINDOW at 48, 44.1, 32, 24, 22.05, 16, 12, 11.025, 8 kHz

layer	48 kHz	44.1 kHz	32 kHz	24 kHz	22.05 kHz	16 kHz	12 kHz	11.025 kHz	8 kHz
0	5	5	6	8	9	11	13	13	15
1	5	5	7	9	10	12	13	14	15

2	5	6	7	9	10	12	14	14	15
3	6	6	8	9	11	13	14	15	15
4	6	6	8	10	11	13	15	15	15
5	6	6	8	10	11	13	15	15	15
6	6	6	8	11	11	14	15	15	15
7	6	7	9	11	12	14	15	15	15
8	6	7	9	11	12	15	15	15	15
9	7	7	9	12	12	15	15	15	15
10	7	7	9	12	12	15	15	15	15
11	7	7	10	13	13	15	15	15	15
12	7	8	10	13	13	15	15	15	15
13	7	8	10	13	13	15	15	15	15
14	7	8	10	14	13	15	15	15	15
15	8	8	11	14	14	15	15	15	15
16	8	8	11	14	14	15	15	15	15
17	8	8	11	14	14	15	15	15	15
18	8	9	11	14	14	15	15	15	15
19	8	9	11	14	14	15	15	15	15
20	8	9	12	15	15	15	15	15	15
21	9	9	12	15	15	15	15	15	15
22	9	9	12	15	15	15	15	15	15
23	9	9	12	15	15	15	15	15	15
24	9	9	12	15	15	15	15	15	15
25	9	9	12	15	15	15	15	15	15
26	9	10	12	15	15	15	15	15	15
27	9	10	13	15	15	15	15	15	15
28	9	10	13	15	15	15	15	15	15
29	9	10	13	15	15	15	15	15	15
30	10	10	13	15	15	15	15	15	15
31	10	10	13	15	15	15	15	15	15
32	10	10	13	15	15	15	15	15	15
33	10	10	13	15	15	15	15	15	15
34	10	11	14	15	15	15	15	15	15
35	10	11	14	15	15	15	15	15	15
36	10	11	14	15	15	15	15	15	15
37	10	11	14	15	15	15	15	15	15
38	10	11	14	15	15	15	15	15	15
39	11	11	14	15	15	15	15	15	15
40	11	11	14	15	15	15	15	15	15
41	11	11	14	15	15	15	15	15	15
42	11	12	14	15	15	15	15	15	15
43	11	12	14	15	15	15	15	15	15
44	11	12	14	15	15	15	15	15	15
45	11	12	14	15	15	15	15	15	15
46	12	12	14	15	15	15	15	15	15
47	12	12	14	15	15	15	15	15	15
48	12	12	14	15	15	15	15	15	15
> 48	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb	max_sfb

2.5 Figures

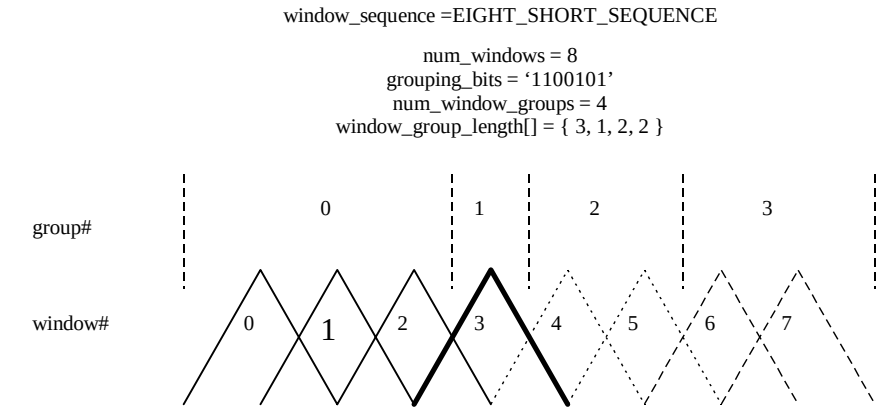


Figure 2.1 – Example for short window grouping

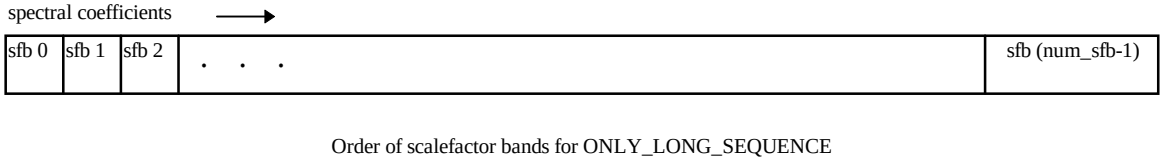


Figure 2.2 – Spectral order of scalefactor bands in case of ONLY_LONG_SEQUENCE

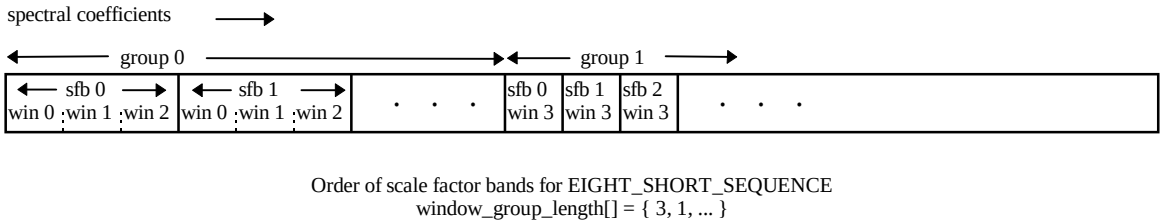


Figure 2.3 – Spectral order of scalefactor bands in case of EIGHT_SHORT_SEQUE

Subpart 4 of CD 14496-3 is split into several files:

w2203tfs	Syntax, semantics and decoder description
w2203tft	T/F tool descriptions and normative Annex
w2203tfa	This file. Informative annex (Transport streams, Encoder tools)
w2203tvq	Twin-VQ vector quantizer tables

ANNEX A: Encoder

Weighted interleave vector quantization

In the weighted interleave VQ section, the flattened MDCT coefficients vector and the weight vector is divided into subvector at the first stage. Then, the subvectors are applied to weighted vector quantization.

Definition

bits_available_vqNumber of bits available for VQ section.

Initialization

Number of sub-blocks in a frame, N_SF, is set as follows:

```
N_SF = FRAME_SIZE / N_FR
```

Value of N_FR is changed according to bitrate mode and sub-block type (see Tables from TSPN1 to TSPN?). FRAME_SIZE equals to N_FR of long block.

Value of bits_available is calculated as follows:

```
available_vq =
   (int)(FRAME_SIZE*number_of_channel*bitrate/sampling_frequency) -
   bits_for_side_information
```

Number of vector division, N_DIV, and length for each sub-vector length[] are set as follows.

Number of shape sub-vector, `N_DIV`. Length of the codevector, `lengh[idiv]`, are calculated by the following formula:

```
   N_DIV = ((int)((bits_available_vq + MAXBIT*2-1)/(MAXBIT*2)))

   for(idiv=0; idiv<ntt_N_DIV; idiv++){
      length[idiv] = (FRAME_SIZE * n_ch + N_DIV - 1 - idiv) / N_DIV
      bits = (bits_available_vq + N_DIV - 1 - idiv) / N_DIV
      bits0[idiv] = (int)(bits+1) / 2
      bits1[idiv] = (int)bits/2
   }
```

Division of the vectors

At the first stage of encoding process, input vector is divided into subvectors as follows:

```
for (idiv=0; idiv<N_DIV; idiv++){
   for (icv=0; icv<length[idiv]; icv++){
      itmp = idiv + icv * N_DIV
      isf  = itmp % N_SF
      ismp = itmp / N_SF
```

```
      Usub[idiv][icv] = U[isf][ismp]
      wtsub[idiv][icv] = lpcspectrum[ismp]*qenv[isf][ismp]
    }
}
```

where Usub[] and wtsub[] is element of input-coefficients subvector and weight subvector respectively.

Vector quantization

Basic structure

This vector quantizer has a two-channel conjugate structure, where two sets of codebooks are prepared. The search procedure consists of three steps; pre-selection, main selection and index packing.

Pre-selection

At the first step of the vector quantization procedure, candidates for the nearest codevector are selected from each codebook. To select the candidates, weighted distortion measures are calculated.

$$dist[idiv][icb] = \sum_{ismp=0}^{vec\_len[idiv]-1} wt_{sub}{}^{2}(sp\_cv0[icb][ismp] - U_{sub}[idiv][ismp])^{2},$$

$$\text{for } idiv = 0 \text{ to } N\_DIV - 1,\ icb = 0 \text{ to } 2^{bits[idiv]} - 1$$

If the *bits*0[*idiv*] is grater than MAXBIT_SHAPE, the following distortion are also calculated.

$$dist_{negative}[idiv][icb] = \sum_{ismp=0}^{vec\_len[idiv]-1} wt_{sub}{}^{2}(sp\_cv0[icb][ismp] + U_{sub}[idiv][ismp])^{2},$$

$$\text{for } idiv = 0 \text{ to } N\_DIV - 1,\ icb = 0 \text{ to } 2^{bits[idiv]} - 1$$

The N_CAN candidates of dist and $dist_{negative}$ are selected with minimum distortion measure

Main selection

The distortion measures are calculated as below.

$$dist_{cross}[idiv][icb] = \sum_{ismp=0}^{vec\_len[idiv]-1} wt_{sub}[ismp]\left(\begin{array}{l} pol0[idiv] \\ \cdot sp\_cv0[can\_ind0[ican0]][ismp] \\ + pol1[idiv] \\ \cdot sp\_cv1[can\_ind1[ican1]][ismp] \\ \hline 2 \end{array} - U_{sub}[idiv][ismp] \right)^{2}$$ which

$$\text{for } idiv = 0 \text{ to } N\_DIV - 1,\ ican0 = 0 \text{ to } N\_CAN - 1,\ ican1 = 0 \text{ to } N\_CAN - 1$$

$$pol0[idiv] = 1 \quad \text{if } bits0[idiv] = MAXBIT\_SHAPE$$

$$pol0[idiv] = \begin{cases} 1 \\ -1 \end{cases} \quad \text{if } bits0[idiv] > MAXBIT\_SHAPE$$

$$pol1[idiv] = 1 \quad \text{if } bits1[idiv] = MAXBIT\_SHAPE$$

$$pol1[idiv] = \begin{cases} 1 \\ -1 \end{cases} \quad \text{if } bits1[idiv] > MAXBIT\_SHAPE$$

Indexes *can_ind*0[*ican*0] and *can_ind*1[*ican*1] which give the least distortion measure is the quantization indexes *index*0[*idiv*] and *index*1[*idiv*].

Index packing

For each VQ index, the polarity information is added as follows.

If $pol0[idiv] = -1$ then $index0[idiv] = index0[idiv] + 2^{MAXBIT\_SHAPE}$, for $idiv = 0$ to N_DIV

If $pol1[idiv] = -1$ then $index1[idiv] = index1[idiv] + 2^{MAXBIT\_SHAPE}$, for $idiv = 0$ to N_DIV

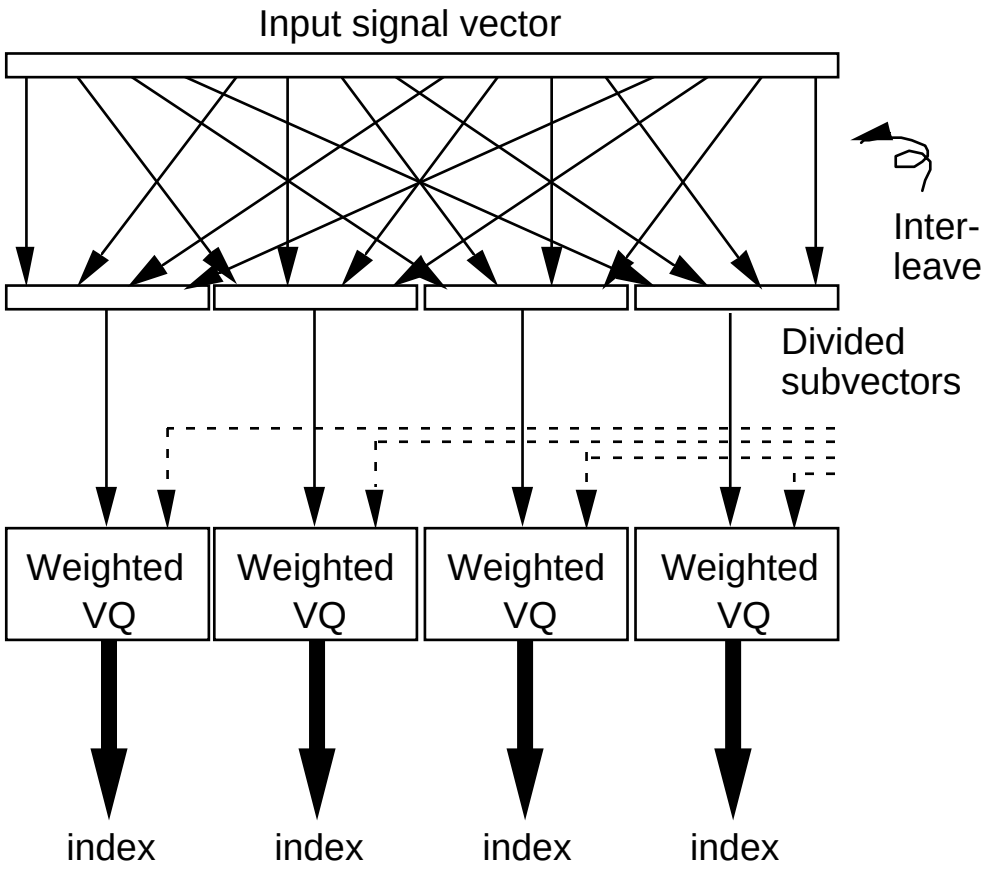


Figure 4. Weighted interleave vector quantization.

Spectrum normaliztion

Definition

Encoding process

Bandwidth control

This process limits the bandwidth of MDCT spectrum.

LPC spectrum coding

At the first stage of the spectrum normalization process, input samples are divided into two paths. In one path they are transformed into frequency domain coefficients by MDCT. In the other path, LPC coefficients are calculated by LPC analysis; then, the LPC modelled spectrum envelope is calculated; and finally, each MDCT coefficient is divided by its corresponding spectrum envelope and first-stage flattened coefficients are produced. The main advantage of this flattening method is that fewer bits are required to normalize the frequency characteristics of input signals as a result of making use of an efficient quantization scheme of LSP (Line Spectral Pair) parameters. In order to avoid square root calculation in the de-flattening procedure at the decoder, LPC coefficients for the square root envelope are quantized; they are derived from the half-value of the LPC cepstrum of the normal LPC coefficients. The procedure for obtaining spectral envelope is listed as follows.

- Autocorrelation function is calculated from the windowed input signal.
- LPC coefficients are calculated by the Levinson-Durbin-Shur method.
- LPC cepstrum coefficients are generated from LPC coefficients.
- LPC cepstrum coefficients are multiplied by 0.5.
- LPC coefficients (corresponding to square root spectrum) are converted from the LPC cepstrum coefficients by solving a normal equation.
- Stability of the obtained coefficients is checked.
- LPC coefficients are converted to LSP parameters.

LSP parameters are quantized by 2-stage split vector quantizer with moving average inter-frame prediction. Inverse LPC spectral envelope values are calculated by the same method in 2.3.1.5.

First-stage flattening

MDCT coefficients $X[isf][j]$ are flattened using LPC spectrum as follows.

$$X_{flat1}[isf][ismp] = X[isf][ismp] / lpc\_spectrum[j],$$
$$\text{for } 0 \le isf < N\_SF, \quad 0 \le ismp < N\_FR$$

Periodic peak components coding

Fundamental frequency estimation

For the low rate coder mode and only for a long frame mode, the periodic peak components of the flattened MDCT coefficients due to the pitch frequency of speech, vocal and some musical instruments.

Vector quantization of the periodic peak components

Extracted pitch components are quantized by the interleaved weighted vector quantization similar to those used for flattened MDCT coefficients.

Subtraction of periodic peak components

Bark-scale envelope coding

Calculation of the Bark-scale envelope

The inputs of the Bark-scale envelope calculation procedure are the MDCT coefficients xflat[] flattened by the LPC spectrum. First the square-rooted power of the input coefficients corresponding to each Bark-scale subband is calculated. The upper frequency boundary of each subband is listed in tables from ? to ?.

$$env\_tmp[isf][ib] = \sqrt{\frac{\sum_{ismp=iblow}^{ibhigh-1} X_{flat1}[ismp]^2}{ibhigh - iblow}},$$
$$\text{for } 0 \le isf < N\_SF,\ 0 \le ib < N\_CRB$$

Next, the average of all square-rooted powers is calculated.

$$env\_avr[isf] = \frac{\sum_{ib=0}^{N\_CRB-1} env\_tmp[isf][ib]}{N\_CRB}, \quad \text{for } 0 \le isf < N\_SF$$

Then, square-rooted powers are normalized by the average to create the MDCT envelope.

$$env[isf][ib] = env\_tmp[isf][ib] / env\_avr[isf],$$
$$\text{for } 0 \le isf < N\_SF,\ 0 \le ib < N\_CRB$$

Weight calculation

The weight is used for the vector quantization of the MDCT envelope.

$$env\_fwt[ib] = \frac{\sum_{ismp=iblow}^{ibhigh-1} (lpc\_spectrum[ismp])}{ibhigh - iblow}$$

Quantization

Interframe prediction

Before quantizing the MDCT envelope, the prediction switch is set. The values of envelope elements are subtracted by their average (is 1).

$$env[is][ib] = env[isf][ib] - 1, \quad \text{for } 0 \le isf < N\_SF,\ 0 \le ib < N\_CRB$$

Correlation between current-frame env[][] and previous-frame env[][] is calculated.

$$correlation = \frac{\sum_{ib=0}^{N\_CRB-1} env_{current}[isf][ib] \cdot env_{previous}[isf][ib]}{\sum_{ib=0}^{N\_CRB-1} (env_{current}[isf][ib])^2}, \quad \text{for } 0 \le isf < N\_SF$$

if N_SF is not equal to 1, the envelope of the previous frame is set by a following equation:

$$env_{previous}[0][ib] = 0,$$
$$env_{previous}[isf][ib] = env_{current}[isf-1][ib], \quad \text{for } 0 \le isf < N\_SF,\ 0 \le ib < N\_CRB$$

If the correlation measure is greater than 0.5, prediction is on, otherwise, prediction is off.

$$\text{if } (correlation[isf] > 0.5) \text{ then } index\_fw\_alf[isf] = 1$$
$$\text{else} \qquad index\_fw\_alf[isf] = 0,$$
$$\text{for } 0 \le isf < N\_SF$$

Vector quantization of the MDCT envelope

At the first step of quantization, the envelope and the weight vector are divided into FW_N_DIV subvectors.

$$denv[isf][ifdiv][icv] = env[isf][FW\_N\_DIV * icv + ifdiv],$$
$$\text{for } 0 \le isf < N\_SF,\ 0 \le ifdiv < N\_CRB,\ 0 \le icv < FW\_CB\_LEN$$
$$dfwt[ifdiv][icv] = env\_fwt[FW\_N\_DIV * icv + ifdiv],$$
$$\text{for } ifdiv = 0 \text{ to } N\_CRB - 1,\ icv = 0 \text{ to } FW\_CB\_LEN - 1$$

Distortion measures $fwdist[itmp]$ are calculated as follows.

$$alfq[isf] = index\_fw\_alf[isf] \cdot FW\_ALF\_STEP$$

$$fwdist[isf][itmp] =$$
$$\sum_{icv=0}^{FW\_CB\_LEN-1} dfwt[ifdiv][icv]^2 \cdot$$
$$(denv[isf][ifdiv][icv] - alfq \cdot p\_cv\_env[isf][icv] - cb\_env[itmp][icv])^2$$
$$\text{for } 0 \le isf < N\_SF,\ 0 \le itmp < FW\_CB\_SIZE,\ 0 \le ifdiv < FW\_N\_DIV$$

The cv_env[][] are elements of the envelope codebook. The p_cv_env[] represents the MDCT envelope of the previous frame. It is calculated by the procedure mensioned in section 2.4.4.3.

Finally, $itmp$ for the minimum distortion measure $fwdist[itmp]$ is the quantization index $index\_env[ifdiv]$.

Local decoding

After the vector quantization, the MDCT envelope is reproduced as follows.

```
for (isf=0; isf<N_SF; isf++){
  alfq[isf] = index_fw_alf[isf] * FW_ALF_STEP
  for (ifdiv=0; ifdiv<FW_N_DIV; ifdiv++){
    for (icv=0; icv<FW_CB_LEN; icv++){
      ienv = FW_N_DIV * icv + ifdiv
      dtmp = cv_env[index_env[isf][ifdiv]][icv]
      qenv_b[isf][ienv] = dtmp + alfq * p_cv_env[isf][icv] + 1
      if (N_SF==1) p_cv_env[isf][icv] = dtmp
      else         p_cv_env[isf-1][icv] = dtmp
    }
  }
}
```

The cv_env[][] are the envelope codebook.

Then, Bark-scale envelope *qenv_b*[*isf*][*ienv*] are projected into linear-scale envelope *qenv*[*isf*][*ismp*].

```
for (isf=0; isf<N_SF; ist++){
  for (ib=0; ib<N_CRB; ib++){
    for (ismp=iblow[ib]; ismp<ibhigh[ib]; ismp++){
      qenv[isf][ismp]=qenv_b[isf][ib];
    }
  }
}
```

Second-stage flattening

$X_{flat2}[][]$ are flattened as follows:

$$X_{flat2}[isf][j] = X_{flat1}[isf][j] / qenv[j] \qquad \text{for } 0 \le isf < N\_SF,\ 0 \le j < N\_FR$$

Gain quantization and amplitude normalization

Before quantizing the flattened MDCT coefficients, their amplitudes are normalized by the gain factor.

The gain factor is calculated as follow:

$$gain[isf] = \frac{\sqrt{\sum_{ismp=0}^{N\_FR-1} X_{flat2}[isf]^2}}{AMP\_NM}, \quad \text{for } 0 \le isf < N\_SF$$

If N_SF == 1, the gain factor is quantized using the following equation:

$$index\_gain[0] = (\text{int}) \frac{\log10(1 + MU \cdot gain[0] / AMP\_MAX)}{\log10(1 + MU)}$$

Then, gain factor is locally decoded.

```
g_temp = index_gain * STEP + STEP / 2
qgain = AMP_MAX * (exp10(g_temp * log10(1.+MU) / AMP_MAX) -1) / MU
qgain /= AMP_NM
```

If N_SF > 1, the global gain and subframe gains are quantized:

$$index\_gain[0] = (\text{int}) \frac{\log10(1 + MU \cdot gain[0] / AMP\_MAX)}{\log10(1 + MU)}$$

Finally, the flattened MDCT coefficients are normalized by the decoded gain factor *qgain*.

$$U[ismp] = X_{flat2}[ismp] / qgain \quad \text{for } ismp = 0 \text{ to } N\_FR - 1$$

Psychoacoustic Model

General

This annex presents the general Psychoacoustic Model for the AAC encoder . The psychoacoustic model calculates the maximum distortion energy which is masked by the signal energy. This energy is called *threshold.* The threshold generation process has three inputs. They are:

1. The shift length for the threshold calculation process is called *iblen.* This *iblen* must remain constant over any particular application of the threshold calculation process. Since it is necessary to calculate thresholds for two different shift lengths, two processes, each running with a fixed shift length, are necessary. For long FFT *iblen* = 1024, for short FFT *iblen* = 128.

2. For each FFT type the newest *iblen* samples of the signal, with the samples delayed (either in the filterbank or psychoacoustic calculation) such that the window of the psychoacoustic calculation is centered in the time-window of the codec time/frequency transform .

3. The sampling rate. There are sets of tables provided for the standard sampling rates. Sampling rate, just as *iblen*, must necessarily remain constant over one implementation of the threshold calculation process.

The output from the psychoacoustic model is :

1. a set of Signal-to-Mask Ratios and thresholds, which are adapted to the encoder as described below,

2. the delayed time domain data (PCM samples) , which are used by the MDCT,

3. the block type for the MDCT (long, start, stop or short type)

4. an estimation of how many bits should be used for encoding in addition to the average available bits.

The delay of the PCM samples is necessary , because if the switch decision algorithm detects an attack, so that *short blocks* have to be used for the actual frame, the *long block* before the *short blocks* has to be 'patched' to a *start block type* in this case.. Before running the model initially, the array used to hold the preceding FFT source data window and the arrays used to hold *r(w)* and *f(w)* should be zeroed to provide a known starting point.

Comments on notation

Throughout this threshold calculation process, three indices for data values are used. These are:

w- indicates that the calculation is indexed by frequency in the FFT spectral line domain. An index of 0 corresponds to the DC term and an index of 1023 corresponds to the spectral line at the Nyquist frequency.

b - indicates that the calculation is indexed in the threshold calculation partition domain. In the case where the calculation includes a convolution or sum in the threshold calculation partition domain, *bb* will be used as the summation variable. Partition numbering starts at 0.

n - indicates that the calculation is indexed in the coder scalefactor band domain. An index of 0 corresponds to the lowest scalefactor band.

The "spreading function"

Several points in the following description refer to the "spreading function". It is calculated by the following method:

if j>=i
tmpx =3,0 (j-i)
else
tmpx = 1.5(j-i)

Where *i* is the Bark value of the signal being spread, *j* is the Bark value of the band being spread into, and *tmpx* is a temporary variable.

$tmpz = 8 * minimum\ ((tmpx\text{-}0{,}5)^2\text{-}2(tmpx\text{-}0{,}5),0)$

Where *tmpz* is a temporary variable, and minimum (a , b) is a function returning the more negative of a or b.

$tmpy = 15.811389 + 7.5(tmpx + 0.474)\text{-}17{,}5(1{,}0+(tmpx + 0.474)^2)^{0{,}5}$

where *tmpy* is another temporary variable.

if (*tmpy*<-100) then {*sprdngf (i , j)*=0} else {*sprdngf (i , j)*=10^((*tmpz* + *tmpy*)/10)}

Steps in threshold calculation

The following are the necessary steps for the calculation of *SMR(n)* and *xmin(n)* used in the coder for long and short FFT.

1. Reconstruct *2 * iblen* samples of the input signal.
iblen new samples are made available at every call to the threshold generator. The threshold generator must store *2 * iblen*

- *iblen* samples, and concatenate those samples to accurately reconstruct *2 * iblen* consecutive samples of the input signal, *s(i)*, where *i* represents the index,
$0 <= i < 2 * iblen$, of the current input stream.

2. Calculate the complex spectrum of the input signal.
First, *s(i)* is windowed by a Hann window, i.e.

*sw(i) =s(i) * (0,5-0,5 * cos((pi *(i+0.5))/ iblen).*

Second, a standard forward FFT of *sw(i)*calculated.

Third, the polar representation of the transform is calculated. *r(w)* and *f(w)* represent the magnitude and phase components of the transformed *sw(i),* respectively.

3. Calculate a predicted *r(w)*and *f(w).*
A predicted magnitude, r_pred(w) and phase, f_pred(w) are calculated from the preceding two threshold calculation blocks *r(w)*and *f(w)*:

*r_pred(w) = 2.0 * r (t-1)-r(t-2)*

*f_pred(w) =2.0 * f(t-1)-f (t-2)*

where *t* represents the current block number, *t-1* indexes the previous block's data, and *t-2* indexes the data from the threshold calculation block before that.

4. Calculate the unpredictability measure *c(w)*:

```
c(w)= (((r(w) * cos(f(w)) - r_pred(w) * cos(f_pred(w)))^2
      + (r(w) * sin(f(w)) - r_pred(w) * sin(f_pred(w)))^2)^0.5 ) /
      (r(w) + abs(r_pred(w))
```

This formula is used for each of the short blocks with the short FFT, for long blocks for the first 6 lines the unpredictability measure is calculated from the long FFT, for the remaining lines the minimum of the unpredictability of all short FFT's is used. If calculation power should be saved, the unpredictability of the upper part of the spectrum can be set to 0.4.

5. Calculate the energy and unpredictability in the threshold calculation partitions.

The energy in each partition, *e(b),* is:

```
do for each partition b:
        e(b) = 0
        do from lower index to upper index w of partition b
                e(b) = e(b) + r(w)^2
        end do
```

(e(b) is used in the M/S-module (see section 2.6.1 of annex B :„Joint Coding“) : e(b) is equal to Xengy with 'X' = [R,L,M,S])
and the weighted unpredictability, *c(b),* is:

```
do for each partition b:
        c(b) = 0
        do from lower index to upper index w of partition b
                c(b) = c(b) + r(w)^2 * c(w)
        end do
end do
```

The threshold calculation partitions provide a resolution of approximately either one FFT line or 1/3 critical band, whichever is wider. At low frequencies, a single line of the FFT will constitute a calculation partition. At high frequencies, many lines will be combined into one calculation partition. A set of partition values is provided for each of the three sampling rates in tables B.2.1.1 to B.2.1.12. These table elements will be used in the threshold calculation process. There are several elements in each table entry:

1. The index of the calculation partition, *b*.

2. The lowest frequency line in the partition, *w_low(b).*

3. The highest frequency line in the partition, *w_high(b)*

4. The median bark value of the partition, *bval(b)*

5. The threshold in quiet qsthr(b)

A largest value of *b*, *bmax*, equal to the largest index, exists for each sampling rate.

6. Convolve the partitioned energy and unpredictability with the spreading function.

```
for each partition b:
        ecb(b) = 0
        do for each partition bb:
                ecb(b) = ecb(b) +e(bb)* sprdngf(bval(bb),bval(b))
        end do
end do


do for each partition b:
        ct(b) = 0
        do for each partition bb:
                ct(b) = ct(b) +c(bb)* sprdngf(bval(bb),bval(b))
        end do
end do
```

Because *ct(b)* is weighted by the signal energy, it must be renormalized to *cb(b)*.

$$cb(b) = ct(b) / ecb(b)$$

Just as this, due to the non-normalized nature of the spreading function, ecb_b should be renormalized and the normalized energy en_b, calculated.

$$en(b) = ecb(b) * rnorm(b)$$

The normalization coefficient, *rnorm(b).* is:

```
do for each partition b
        tmp(b) = 0
        do for each partition bb
                tmp(b) = tmp(b) + sprdngf(bval(bb),bval(b))
        end do
        rnorm(b) = 1/ tmp(b)
end do
```

7. Convert *cb(b)* to *tb(b)* , the tonality index.

$$tb(b) = -0{,}299 - 0{,}43 \log_e (cb(b))$$

Each *tb(b)* is limited to the range of $0<tb(b)<1$.

8. Calculate the required SNR in each partition.

NMT(b) = 6 dB for all *b*. *NMT(b)* is the value for noise masking tone (in dB) for the partition. *TMN(b) = 18 dB* for all *b*. *TMN(b)* is the value for tone masking noise (in dB) .The required signal to noise ratio, *SNR(b)* , is:

*SNR(b) = tb(b) * TMN(b) + (1-tb(b)) * NMT(b)*

9. Calculate the power ratio.

   The power ratio, *bc(b)* , is:

   bc(b) =10^(-SNR(b) /10)

10. Calculation of actual energy threshold, *nb(b)* .

    *nb(b)= en(b) * bc(b)*

    nb(b) is also used in the M/S-module (see chapter 7 „Joint coding“): nb(b) is equal to Xthr with ‘X’=[R,L,M,S]

11. Pre-echo control and threshold in quiet.

    To avoid pre-echoes the pre-echo control is calculated for short and long FFT, the threshold in quiet is also considered here:

    nb_l(b) is the threshold of partition b for the last block , *qsthr(b)* is the threshold in quiet . The dB values of *qsthr(b)* shown in tables B.2.1.1 to B.2.1.12 are relative to the level that a sine wave of + or - ½ lsb has in the FFT used for threshold calculation. The dB values must be converted into the energy domain after considering the FFT normalization actually used.

    *nb(b) = max (qsthr(b), min (nb(b), nb_l(b)*rpelev))*

    rpelev is set to ‘1’ for short blocks and ‘2’ for long blocks

12. The PE is calculated for each block type from the ratio e(b) / nb (b) , where nb(b) is the threshold and e(b) is the energy for each threshold partition :

    ```
    PE = 0
    do for threshold partition b
            PE = PE - ( w_high(b)-w_low(b)) * log10 ( nb(b) / ( e(b) +1) )
    end do
    ```

13. The decision , whether long or short block type is used for encoding is made according to this pseudo code:

    ```
    if PE for long block is greater than switch_pe then
            coding_block_type = short_block_type
    else
            coding_block_type = long_block_type
    end if
    if ( coding_block_type == short_block_type) and ( last_coding_block_type == long_type ) then
            last coding block type = start_type
    else
            last_coding_block_type = short_type
    ```

    The last four lines are necessary since there is no combined stop/start block type in AAC. *switch_pe* is a implementation dependend constant

14. Calculate the signal-to-mask ratios, *SMR(n)* and the codec threshold *xmin(n)*.

    Tables 3.4 to 3.16 (normative part) shows:

    1. The index, *swb*, of the coder partition called scalefactor band.

    2. The offset of mdct line for the scalefactor band *swb_offset_long/short_window*.

    we define the following variable :
    n = swb

w_low(n) = swb_offset_long/short_window(n)
w_high(n) = swb_offset_long/short_window(n+1) - 1

The FFT energy in the scalefactor band, *epart(n)* , is:

do for each scalefactor band ***n***
epart(n) = 0
do for ***w*** *= lower index w_low(n) to n = upper index w_high(n)*
epart(n) = epart (n) + r(w)^2
end do
end do

the threshold for one line of the spectrum is calculated according to:

do for each threshold partition ***b***
thr(all line_indices in this partition ***b*** *)=*
thr (w_low(b),...,w_high(b)) = nb(b) / (w_high(b)+1-w_low(b))
end do

the noise level in the scalefactor band on FFT level , *npart(n)* is calculated as:

do for each scalefactor band ***n***
npart(n) = minimum(thr(w_low(n)),..., thr(w_high(n)))
** (w_high(n)+1-w_low(n))*
end do

Where, in this case, minimum (a,...,z) is a function returning the smallest positive argument of the arguments a...z.

The ratios to be sent to the quantization module, *SMR(n)*, are calculated as:

SMR(n) = epart(n) / npart(n)

For the calculation of coder thresholds xmin(n) the MDCT energy for each scalefactor band is calculated:

do for all scalefactor bands n
codec_e(n) = 0
do for lower index ***i*** *to higher index* ***i*** *of this scalefactor band*
codec_e(n) = codec_e(n) +(mdct_line(i))^2
end do
end do

Then *xmin(n)*, the maximum allowed error energy on MDCT level, can be calculated according to this formula :

*xmin (n) = npart(n) * codec_e(n) / epart (n)*

15. Calculate the bit allocation out of the psychoacoustic entropy (PE).
bit_allocation = pew1 PE + pew2*sqrt(PE);*

for long blocks the constants are defined as:
pew1 = 0.3 , pew2 = 6.0

for short blocks the PE of the eight short blocks is summed up and the constants are :
pew1 = 0.6 , pew2 = 24

then bit_allocation is limited to 0<bit_allocation < 3000 and
more_bits is calculated :

more_bits = bit_allocation - (average_bits - side_info_bits)

Figure B.2.1.1 - block diagram psychoacoustic model

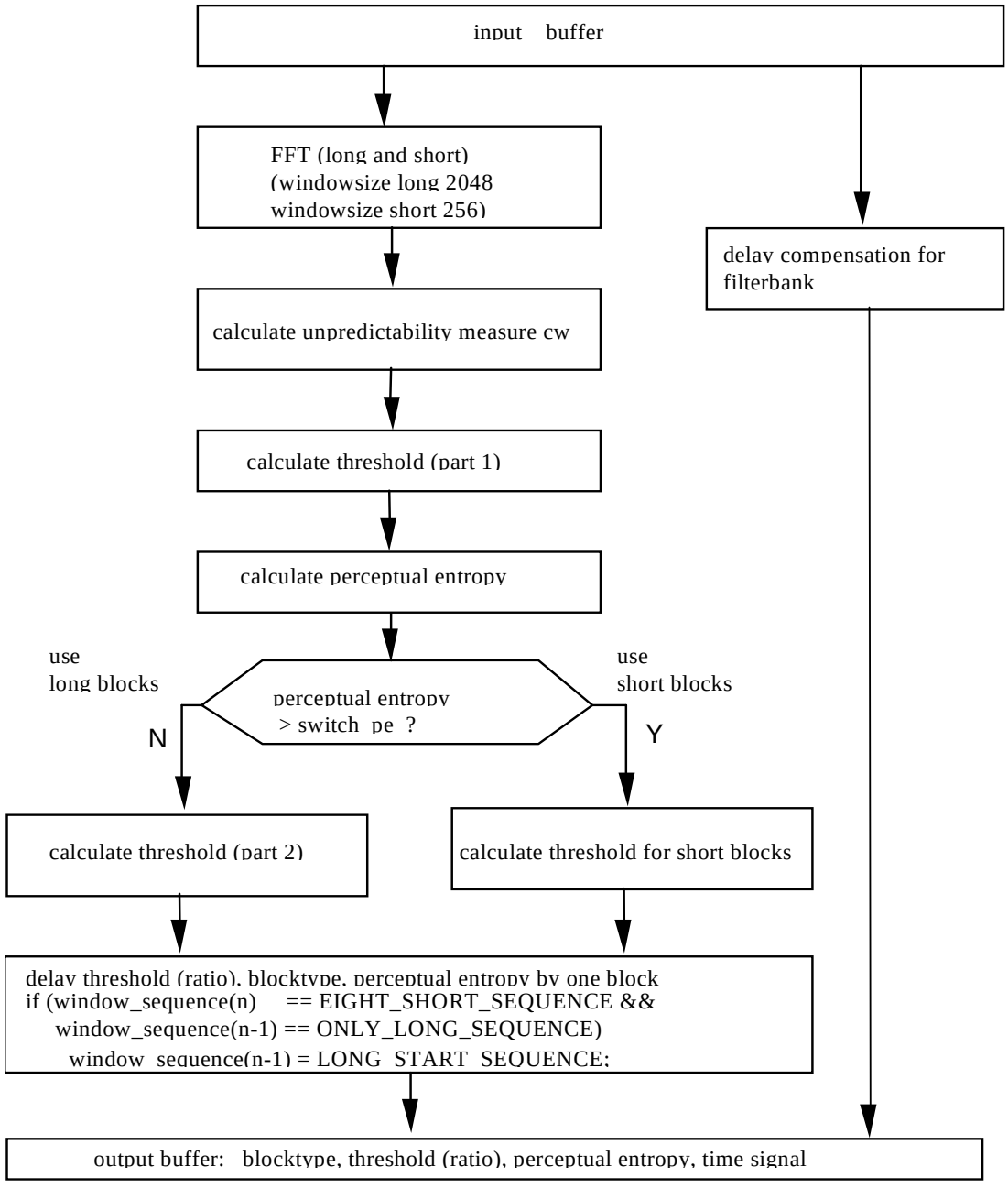


Table B.2.1.1.a -- Psychoacoustic parameters for 8 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	8	9	0,18	46,82
1	9	17	9	0,53	46,82
2	18	26	9	0,89	46,82
3	27	35	9	1,24	41,82
4	36	44	9	1,59	41,82
5	45	53	9	1,94	41,82
6	54	62	9	2,29	38,82
7	63	71	9	2,63	38,82
8	72	80	9	2,98	38,82
9	81	89	9	3,31	33,82
10	90	98	9	3,65	33,82
11	99	108	10	3,99	34,28
12	109	118	10	4,35	32,28
13	119	128	10	4,71	32,28
14	129	138	10	5,05	32,28
15	139	148	10	5,39	32,28
16	149	159	11	5,74	32,69
17	160	170	11	6,10	32,69
18	171	181	11	6,45	32,69
19	182	192	11	6,79	32,69
20	193	204	12	7,13	33,07
21	205	216	12	7,48	33,07
22	217	228	12	7,82	33,07
23	229	241	13	8,17	33,42
24	242	254	13	8,51	33,42
25	255	268	14	8,85	33,74
26	269	282	14	9,20	33,74
27	283	297	15	9,54	34,04
28	298	312	15	9,88	34,04
29	313	328	16	10,22	34,32
30	329	345	17	10,56	34,58
31	346	363	18	10,91	34,83
32	364	381	18	11,25	34,83
33	382	400	19	11,58	35,06
34	401	420	20	11,91	35,29
35	421	441	21	12,24	35,50
36	442	464	23	12,58	35,89
37	465	488	24	12,92	36,08
38	489	514	26	13,26	36,43
39	515	541	27	13,59	36,59
40	542	570	29	13,93	36,90
41	571	601	31	14,26	37,19
42	602	634	33	14,60	37,46
43	635	670	36	14,93	37,84
44	671	708	38	15,27	38,07
45	709	749	41	15,60	38,40
46	750	793	44	15,93	38,71
47	794	841	48	16,26	39,09
48	842	893	52	16,60	39,44
49	894	949	56	16,93	39,76
50	950	1009	60	17,26	40,06
51	1010	1023	14	17,47	33,74

Table B.2.1.1.b -- Psychoacoustic parameters for 8 KHz short FFT

index	w_low	w_high	width	bval	
0	0	1	2	0,32	30,29
1	2	3	2	0,95	30,29
2	4	5	2	1,57	25,29
3	6	7	2	2,19	22,29
4	8	9	2	2,80	22,29
5	10	11	2	3,40	17,29
6	12	13	2	3,99	17,29
7	14	15	2	4,56	15,29
8	16	17	2	5,12	15,29
9	18	19	2	5,66	15,29
10	20	21	2	6,18	15,29
11	22	23	2	6,68	15,29
12	24	25	2	7,16	15,29
13	26	27	2	7,63	15,29
14	28	29	2	8,07	15,29
15	30	31	2	8,50	15,29
16	32	33	2	8,90	15,29
17	34	35	2	9,29	15,29
18	36	37	2	9,67	15,29
19	38	39	2	10,03	15,29
20	40	41	2	10,37	15,29
21	42	44	3	10,77	17,05
22	45	47	3	11,23	17,05
23	48	50	3	11,66	17,05
24	51	53	3	12,06	17,05
25	54	56	3	12,44	17,05
26	57	59	3	12,79	17,05
27	60	63	4	13,18	18,30
28	64	67	4	13,59	18,30
29	68	71	4	13,97	18,30
30	72	75	4	14,32	18,30
31	76	80	5	14,69	19,27
32	81	85	5	15,07	19,27
33	86	90	5	15,42	19,27
34	91	96	6	15,77	20,06
35	97	102	6	16,13	20,06
36	103	109	7	16,49	20,73
37	110	116	7	16,85	20,73
38	117	124	8	17,20	21,31
39	125	127	3	17,44	17,05

Table B.2.1.2.a -- Psychoacoustic parameters for 11,025 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	6	7	0,19	45,73
1	7	13	7	0,57	45,73
2	14	20	7	0,95	45,73
3	21	27	7	1,33	40,73
4	28	34	7	1,71	40,73
5	35	41	7	2,08	37,73
6	42	48	7	2,45	37,73
7	49	55	7	2,82	37,73
8	56	62	7	3,18	32,73
9	63	69	7	3,54	32,73
10	70	76	7	3,89	32,73
11	77	83	7	4,24	30,73
12	84	90	7	4,59	30,73

13	91	97	7	4,92	30,73
14	98	105	8	5,28	31,31
15	106	113	8	5,65	31,31
16	114	121	8	6,01	31,31
17	122	129	8	6,36	31,31
18	130	137	8	6,70	31,31
19	138	146	9	7,06	31,82
20	147	155	9	7,42	31,82
21	156	164	9	7,77	31,82
22	165	173	9	8,11	31,82
23	174	183	10	8,46	32,28
24	184	193	10	8,82	32,28
25	194	203	10	9,16	32,28
26	204	214	11	9,50	32,69
27	215	225	11	9,85	32,69
28	226	237	12	10,19	33,07
29	238	249	12	10,54	33,07
30	250	262	13	10,88	33,42
31	263	275	13	11,22	33,42
32	276	289	14	11,56	33,74
33	290	304	15	11,90	34,04
34	305	320	16	12,24	34,32
35	321	337	17	12,59	34,58
36	338	355	18	12,94	34,83
37	356	374	19	13,28	35,06
38	375	394	20	13,62	35,29
39	395	415	21	13,96	35,50
40	416	438	23	14,29	35,89
41	439	462	24	14,63	36,08
42	463	488	26	14,96	36,43
43	489	516	28	15,29	36,75
44	517	546	30	15,63	37,05
45	547	579	33	15,96	37,46
46	580	614	35	16,30	37,72
47	615	652	38	16,63	38,07
48	653	693	41	16,97	38,40
49	694	737	44	17,30	38,71
50	738	785	48	17,64	39,09
51	786	836	51	17,97	39,35
52	837	891	55	18,30	39,68
53	892	950	59	18,64	39,98
54	951	1014	64	18,97	40,34
55	1015	1023	9	19,16	31,82

Table B.2.1.2.b -- Psychoacoustic parameters for 11,025 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	0,44	27,28
2	2	2	1	0,87	27,28
3	3	3	1	1,30	22,28
4	4	4	1	1,73	22,28
5	5	5	1	2,16	19,28
6	6	6	1	2,58	19,28
7	7	7	1	3,00	14,28
8	8	8	1	3,41	14,28
9	9	9	1	3,82	14,28
10	10	10	1	4,22	12,28

11	11	11	1	4,61	12,28
12	12	12	1	4,99	12,28
13	13	13	1	5,37	12,28
14	14	14	1	5,74	12,28
15	15	15	1	6,10	12,28
16	16	16	1	6,45	12,28
17	17	17	1	6,79	12,28
18	18	19	2	7,44	15,29
19	20	21	2	8,05	15,29
20	22	23	2	8,64	15,29
21	24	25	2	9,19	15,29
22	26	27	2	9,70	15,29
23	28	29	2	10,19	15,29
24	30	31	2	10,65	15,29
25	32	33	2	11,08	15,29
26	34	35	2	11,48	15,29
27	36	37	2	11,86	15,29
28	38	39	2	12,22	15,29
29	40	42	3	12,64	17,05
30	43	45	3	13,10	17,05
31	46	48	3	13,53	17,05
32	49	51	3	13,93	17,05
33	52	54	3	14,30	17,05
34	55	58	4	14,69	18,30
35	59	62	4	15,11	18,30
36	63	66	4	15,49	18,30
37	67	70	4	15,84	18,30
38	71	75	5	16,21	19,27
39	76	80	5	16,58	19,27
40	81	85	5	16,92	19,27
41	86	91	6	17,27	20,06
42	92	97	6	17,62	20,06
43	98	104	7	17,97	20,73
44	105	111	7	18,32	20,73
45	112	119	8	18,67	21,31
46	120	127	8	19,02	21,31

Table B.2.1.3.a -- Psychoacoustic parameters for 12 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	5	6	0,18	45,06
1	6	11	6	0,53	45,06
2	12	17	6	0,89	45,06
3	18	23	6	1,24	40,06
4	24	29	6	1,59	40,06
5	30	35	6	1,94	40,06
6	36	41	6	2,29	37,06
7	42	47	6	2,63	37,06
8	48	53	6	2,98	37,06
9	54	59	6	3,31	32,06
10	60	65	6	3,65	32,06
11	66	72	7	4,00	30,73
12	73	79	7	4,38	30,73
13	80	86	7	4,75	30,73
14	87	93	7	5,11	30,73
15	94	100	7	5,47	30,73
16	101	107	7	5,82	30,73
17	108	114	7	6,15	30,73

18	115	122	8	6,51	31,31
19	123	130	8	6,88	31,31
20	131	138	8	7,24	31,31
21	139	146	8	7,58	31,31
22	147	154	8	7,92	31,31
23	155	163	9	8,27	31,82
24	164	172	9	8,62	31,82
25	173	181	9	8,96	31,82
26	182	191	10	9,31	32,28
27	192	201	10	9,66	32,28
28	202	212	11	10,01	32,69
29	213	223	11	10,36	32,69
30	224	235	12	10,71	33,07
31	236	247	12	11,06	33,07
32	248	260	13	11,41	33,42
33	261	273	13	11,75	33,42
34	274	287	14	12,09	33,74
35	288	302	15	12,43	34,04
36	303	318	16	12,77	34,32
37	319	335	17	13,11	34,58
38	336	353	18	13,46	34,83
39	354	372	19	13,80	35,06
40	373	392	20	14,13	35,29
41	393	414	22	14,47	35,70
42	415	437	23	14,81	35,89
43	438	462	25	15,14	36,26
44	463	489	27	15,48	36,59
45	490	518	29	15,81	36,90
46	519	549	31	16,15	37,19
47	550	583	34	16,48	37,59
48	584	619	36	16,82	37,84
49	620	658	39	17,15	38,19
50	659	700	42	17,48	38,51
51	701	745	45	17,81	38,81
52	746	794	49	18,14	39,18
53	795	847	53	18,48	39,52
54	848	904	57	18,81	39,83
55	905	965	61	19,15	40,13
56	966	1023	58	19,47	39,91

Table B.2.1.3.b -- Psychoacoustic parameters for 12 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	0,47	27,28
2	2	2	1	0,95	27,28
3	3	3	1	1,42	22,28
4	4	4	1	1,88	22,28
5	5	5	1	2,35	19,28
6	6	6	1	2,81	19,28
7	7	7	1	3,26	14,28
8	8	8	1	3,70	14,28
9	9	9	1	4,14	12,28
10	10	10	1	4,57	12,28
11	11	11	1	4,98	12,28
12	12	12	1	5,39	12,28
13	13	13	1	5,79	12,28
14	14	14	1	6,18	12,28

15	15	15	1	6,56	12,28
16	16	16	1	6,93	12,28
17	17	17	1	7,28	12,28
18	18	18	1	7,63	12,28
19	19	20	2	8,28	15,29
20	21	22	2	8,90	15,29
21	23	24	2	9,48	15,29
22	25	26	2	10,02	15,29
23	27	28	2	10,53	15,29
24	29	30	2	11,00	15,29
25	31	32	2	11,45	15,29
26	33	34	2	11,86	15,29
27	35	36	2	12,25	15,29
28	37	38	2	12,62	15,29
29	39	40	2	12,96	15,29
30	41	43	3	13,36	17,05
31	44	46	3	13,80	17,05
32	47	49	3	14,21	17,05
33	50	52	3	14,59	17,05
34	53	55	3	14,94	17,05
35	56	59	4	15,32	18,30
36	60	63	4	15,71	18,30
37	64	67	4	16,08	18,30
38	68	72	5	16,45	19,27
39	73	77	5	16,83	19,27
40	78	82	5	17,19	19,27
41	83	88	6	17,54	20,06
42	89	94	6	17,90	20,06
43	95	101	7	18,26	20,73
44	102	108	7	18,62	20,73
45	109	116	8	18,97	21,31
46	117	124	8	19,32	21,31
47	125	127	3	19,55	17,05

Table B.2.1.4.a -- Psychoacoustic parameters for 16 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	4	5	0,20	0,20
1	5	9	5	0,59	0,59
2	10	14	5	0,99	0,99
3	15	19	5	1,38	1,38
4	20	24	5	1,77	1,77
5	25	29	5	2,16	2,16
6	30	34	5	2,54	2,54
7	35	39	5	2,92	2,92
8	40	44	5	3,29	3,29
9	45	49	5	3,66	3,66
10	50	54	5	4,03	4,03
11	55	59	5	4,39	4,39
12	60	64	5	4,74	4,74
13	65	69	5	5,09	5,09
14	70	74	5	5,43	5,43
15	75	80	6	5,79	5,79
16	81	86	6	6,18	6,18
17	87	92	6	6,56	6,56
18	93	98	6	6,92	6,92
19	99	104	6	7,28	7,28
20	105	110	6	7,63	7,63

21	111	116	6	7,96	7,96
22	117	123	7	8,31	8,31
23	124	130	7	8,68	8,68
24	131	137	7	9,03	9,03
25	138	144	7	9,37	9,37
26	145	152	8	9,71	9,71
27	153	160	8	10,07	10,07
28	161	168	8	10,41	10,41
29	169	177	9	10,75	10,75
30	178	186	9	11,10	11,10
31	187	196	10	11,45	11,45
32	197	206	10	11,80	11,80
33	207	217	11	12,14	12,14
34	218	228	11	12,48	12,48
35	229	240	12	12,82	12,82
36	241	253	13	13,16	13,16
37	254	267	14	13,51	13,51
38	268	282	15	13,86	13,86
39	283	298	16	14,21	14,21
40	299	315	17	14,56	14,56
41	316	333	18	14,90	14,90
42	334	352	19	15,24	15,24
43	353	373	21	15,58	15,58
44	374	395	22	15,91	15,91
45	396	419	24	16,25	16,25
46	420	445	26	16,58	16,58
47	446	473	28	16,92	16,92
48	474	503	30	17,25	17,25
49	504	536	33	17,59	17,59
50	537	571	35	17,93	17,93
51	572	609	38	18,26	18,26
52	610	650	41	18,60	18,60
53	651	694	44	18,94	18,94
54	695	741	47	19,27	19,27
55	742	791	50	19,60	19,60
56	792	845	54	19,94	19,94
57	846	903	58	20,27	20,27
58	904	965	62	20,61	20,61
59	966	1023	58	20,92	20,92

Table B.2.1.4.b -- Psychoacoustic parameters for 16 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	0,63	27,28
2	2	2	1	1,26	22,28
3	3	3	1	1,88	22,28
4	4	4	1	2,50	19,28
5	5	5	1	3,11	14,28
6	6	6	1	3,70	14,28
7	7	7	1	4,28	12,28
8	8	8	1	4,85	12,28
9	9	9	1	5,39	12,28
10	10	10	1	5,92	12,28
11	11	11	1	6,43	12,28
12	12	12	1	6,93	12,28
13	13	13	1	7,40	12,28
14	14	14	1	7,85	12,28

15	15	15	1	8,29	12,28
16	16	16	1	8,70	12,28
17	17	17	1	9,10	12,28
18	18	18	1	9,49	12,28
19	19	19	1	9,85	12,28
20	20	20	1	10,20	12,28
21	21	22	2	10,85	15,29
22	23	24	2	11,44	15,29
23	25	26	2	11,99	15,29
24	27	28	2	12,50	15,29
25	29	30	2	12,96	15,29
26	31	32	2	13,39	15,29
27	33	34	2	13,78	15,29
28	35	36	2	14,15	15,29
29	37	39	3	14,57	17,05
30	40	42	3	15,03	17,05
31	43	45	3	15,45	17,05
32	46	48	3	15,84	17,05
33	49	51	3	16,19	17,05
34	52	55	4	16,57	18,30
35	56	59	4	16,97	18,30
36	60	63	4	17,33	18,30
37	64	68	5	17,71	19,27
38	69	73	5	18,09	19,27
39	74	78	5	18,44	19,27
40	79	84	6	18,80	20,06
41	85	90	6	19,17	20,06
42	91	97	7	19,53	20,73
43	98	104	7	19,89	20,73
44	105	112	8	20,25	24,31
45	113	120	8	20,61	24,31
46	121	127	7	20,92	23,73

Table B.2.1.5.a -- Psychoacoustic parameters for 22,05 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	3	4	0,22	43,30
1	4	7	4	0,65	43,30
2	8	11	4	1,09	38,30
3	12	15	4	1,52	38,30
4	16	19	4	1,95	38,30
5	20	23	4	2,37	35,30
6	24	27	4	2,79	35,30
7	28	31	4	3,21	30,30
8	32	35	4	3,62	30,30
9	36	39	4	4,02	28,30
10	40	43	4	4,41	28,30
11	44	47	4	4,80	28,30
12	48	51	4	5,18	28,30
13	52	55	4	5,55	28,30
14	56	59	4	5,92	28,30
15	60	63	4	6,27	28,30
16	64	67	4	6,62	28,30
17	68	71	4	6,95	28,30
18	72	76	5	7,32	29,27
19	77	81	5	7,71	29,27
20	82	86	5	8,10	29,27
21	87	91	5	8,46	29,27

22	92	96	5	8,82	29,27
23	97	101	5	9,16	29,27
24	102	107	6	9,52	30,06
25	108	113	6	9,89	30,06
26	114	119	6	10,25	30,06
27	120	125	6	10,59	30,06
28	126	132	7	10,95	30,73
29	133	139	7	11,31	30,73
30	140	146	7	11,65	30,73
31	147	154	8	12,00	31,31
32	155	162	8	12,35	31,31
33	163	171	9	12,70	31,82
34	172	180	9	13,05	31,82
35	181	190	10	13,40	32,28
36	191	200	10	13,74	32,28
37	201	211	11	14,07	32,69
38	212	223	12	14,41	33,07
39	224	236	13	14,76	33,42
40	237	250	14	15,11	33,74
41	251	265	15	15,46	34,04
42	266	281	16	15,80	34,32
43	282	298	17	16,14	34,58
44	299	317	19	16,48	35,06
45	318	337	20	16,82	35,29
46	338	359	22	17,16	35,70
47	360	382	23	17,50	35,89
48	383	407	25	17,84	36,26
49	408	434	27	18,17	36,59
50	435	463	29	18,51	36,90
51	464	494	31	18,84	37,19
52	495	527	33	19,17	37,46
53	528	563	36	19,51	37,84
54	564	601	38	19,84	38,07
55	602	642	41	20,17	41,40
56	643	686	44	20,50	41,71
57	687	733	47	20,84	42,00
58	734	784	51	21,17	44,35
59	785	839	55	21,50	44,68
60	840	898	59	21,84	44,98
61	899	962	64	22,17	50,34
62	963	1023	61	22,48	50,13

Table B.2.1.5.b -- Psychoacoustic parameters for 22,05 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	0,87	27,28
2	2	2	1	1,73	22,28
3	3	3	1	2,58	19,28
4	4	4	1	3,41	14,28
5	5	5	1	4,22	12,28
6	6	6	1	4,99	12,28
7	7	7	1	5,74	12,28
8	8	8	1	6,45	12,28
9	9	9	1	7,12	12,28
10	10	10	1	7,75	12,28
11	11	11	1	8,36	12,28
12	12	12	1	8,92	12,28

13	13	13	1	9,45	12,28
14	14	14	1	9,96	12,28
15	15	15	1	10,43	12,28
16	16	16	1	10,87	12,28
17	17	17	1	11,29	12,28
18	18	18	1	11,68	12,28
19	19	19	1	12,05	12,28
20	20	21	2	12,71	15,29
21	22	23	2	13,32	15,29
22	24	25	2	13,86	15,29
23	26	27	2	14,35	15,29
24	28	29	2	14,80	15,29
25	30	31	2	15,21	15,29
26	32	33	2	15,58	15,29
27	34	35	2	15,93	15,29
28	36	38	3	16,32	17,05
29	39	41	3	16,75	17,05
30	42	44	3	17,15	17,05
31	45	47	3	17,51	17,05
32	48	51	4	17,89	18,30
33	52	55	4	18,30	18,30
34	56	59	4	18,67	18,30
35	60	63	4	19,02	18,30
36	64	68	5	19,37	19,27
37	69	73	5	19,74	19,27
38	74	78	5	20,09	22,27
39	79	84	6	20,44	23,06
40	85	90	6	20,79	23,06
41	91	97	7	21,15	25,73
42	98	104	7	21,50	25,73
43	105	112	8	21,85	26,31
44	113	120	8	22,20	31,31
45	121	127	7	22,49	30,73

Table B.2.1.6.a -- Psychoacoustic parameters for 24 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	2	3	0,18	42,05
1	3	5	3	0,53	42,05
2	6	8	3	0,89	42,05
3	9	11	3	1,24	37,05
4	12	14	3	1,59	37,05
5	15	17	3	1,94	37,05
6	18	20	3	2,29	34,05
7	21	23	3	2,63	34,05
8	24	26	3	2,98	34,05
9	27	29	3	3,31	29,05
10	30	32	3	3,65	29,05
11	33	36	4	4,03	28,30
12	37	40	4	4,46	28,30
13	41	44	4	4,88	28,30
14	45	48	4	5,29	28,30
15	49	52	4	5,69	28,30
16	53	56	4	6,08	28,30
17	57	60	4	6,46	28,30
18	61	64	4	6,83	28,30
19	65	68	4	7,19	28,30
20	69	72	4	7,54	28,30

21	73	76	4	7,88	28,30
22	77	81	5	8,25	29,27
23	82	86	5	8,64	29,27
24	87	91	5	9,02	29,27
25	92	96	5	9,38	29,27
26	97	101	5	9,73	29,27
27	102	107	6	10,09	30,06
28	108	113	6	10,47	30,06
29	114	119	6	10,83	30,06
30	120	125	6	11,18	30,06
31	126	132	7	11,53	30,73
32	133	139	7	11,89	30,73
33	140	146	7	12,23	30,73
34	147	154	8	12,57	31,31
35	155	162	8	12,92	31,31
36	163	171	9	13,26	31,82
37	172	180	9	13,61	31,82
38	181	190	10	13,95	32,28
39	191	201	11	14,29	32,69
40	202	213	12	14,65	33,07
41	214	225	12	15,00	33,07
42	226	238	13	15,33	33,42
43	239	252	14	15,66	33,74
44	253	267	15	16,00	34,04
45	268	284	17	16,34	34,58
46	285	302	18	16,69	34,83
47	303	321	19	17,02	35,06
48	322	342	21	17,36	35,50
49	343	364	22	17,70	35,70
50	365	388	24	18,03	36,08
51	389	414	26	18,37	36,43
52	415	442	28	18,70	36,75
53	443	472	30	19,04	37,05
54	473	504	32	19,38	37,33
55	505	538	34	19,71	37,59
56	539	575	37	20,04	40,96
57	576	614	39	20,38	41,19
58	615	656	42	20,71	41,51
59	657	701	45	21,04	43,81
60	702	750	49	21,37	44,18
61	751	803	53	21,70	44,52
62	804	860	57	22,04	49,83
63	861	922	62	22,37	50,20
64	923	989	67	22,70	50,54
65	990	1023	34	22,95	47,59

Table B.2.1.6.b -- Psychoacoustic parameters for 24 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	0,95	27,28
2	2	2	1	1,88	22,28
3	3	3	1	2,81	19,28
4	4	4	1	3,70	14,28
5	5	5	1	4,57	12,28
6	6	6	1	5,39	12,28
7	7	7	1	6,18	12,28
8	8	8	1	6,93	12,28

9	9	9	1	7,63	12,28
10	10	10	1	8,29	12,28
11	11	11	1	8,91	12,28
12	12	12	1	9,49	12,28
13	13	13	1	10,03	12,28
14	14	14	1	10,53	12,28
15	15	15	1	11,01	12,28
16	16	16	1	11,45	12,28
17	17	17	1	11,87	12,28
18	18	18	1	12,26	12,28
19	19	19	1	12,62	12,28
20	20	21	2	13,28	15,29
21	22	23	2	13,87	15,29
22	24	25	2	14,40	15,29
23	26	27	2	14,88	15,29
24	28	29	2	15,32	15,29
25	30	31	2	15,71	15,29
26	32	33	2	16,08	15,29
27	34	36	3	16,49	17,05
28	37	39	3	16,94	17,05
29	40	42	3	17,35	17,05
30	43	45	3	17,73	17,05
31	46	48	3	18,07	17,05
32	49	52	4	18,44	18,30
33	53	56	4	18,83	18,30
34	57	60	4	19,20	18,30
35	61	65	5	19,57	19,27
36	66	70	5	19,96	19,27
37	71	75	5	20,31	22,27
38	76	81	6	20,67	23,06
39	82	87	6	21,04	25,06
40	88	94	7	21,41	25,73
41	95	101	7	21,77	25,73
42	102	109	8	22,13	31,31
43	110	117	8	22,48	31,31
44	118	126	9	22,82	31,82
45	127	127	1	23,01	32,28

Table B.2.1.7.a -- Psychoacoustic parameters for 32 KHz long FFT

Index	w_low	w_high	width	bval	qsthr
0	0	2	3	0.24	42.05
1	3	5	3	0.71	42.05
2	6	8	3	1.18	37.05
3	9	11	3	1.65	37.05
4	12	14	3	2.12	34.05
5	15	17	3	2.58	34.05
6	18	20	3	3.03	29.05
7	21	23	3	3.48	29.05
8	24	26	3	3.92	29.05
9	27	29	3	4.35	27.05
10	30	32	3	4.77	27.05
11	33	35	3	5.19	27.05
12	36	38	3	5.59	27.05
13	39	41	3	5.99	27.05
14	42	44	3	6.37	27.05
15	45	47	3	6.74	27.05

16	48	50	3	7.10	27.05
17	51	53	3	7.45	27.05
18	54	56	3	7.80	27.05
19	57	60	4	8.18	28.30
20	61	64	4	8.60	28.30
21	65	68	4	9.00	28.30
22	69	72	4	9.39	28.30
23	73	76	4	9.76	28.30
24	77	80	4	10.11	28.30
25	81	84	4	10.45	28.30
26	85	89	5	10.81	29.27
27	90	94	5	11.19	29.27
28	95	99	5	11.55	29.27
29	100	104	5	11.90	29.27
30	105	110	6	12.25	30.06
31	111	116	6	12.62	30.06
32	117	122	6	12.96	30.06
33	123	129	7	13.31	30.73
34	130	136	7	13.66	30.73
35	137	144	8	14.01	31.31
36	145	152	8	14.36	31.31
37	153	161	9	14.71	31.82
38	162	171	10	15.07	32.28
39	172	181	10	15.42	32.28
40	182	192	11	15.76	32.69
41	193	204	12	16.10	33.07
42	205	217	13	16.45	33.42
43	218	231	14	16.80	33.74
44	232	246	15	17.14	34.04
45	247	262	16	17.48	34.32
46	263	279	17	17.82	34.58
47	280	298	19	18.15	35.06
48	299	318	20	18.49	35.29
49	319	340	22	18.84	35.70
50	341	363	23	19.17	35.89
51	364	388	25	19.51	36.26
52	389	415	27	19.85	36.59
53	416	444	29	20.19	39.90
54	445	475	31	20.53	40.19
55	476	508	33	20.87	40.46
56	509	543	35	21.20	42.72
57	544	581	38	21.53	43.07
58	582	622	41	21.86	43.40
59	623	667	45	22.20	48.81
60	668	715	48	22.53	49.09
61	716	768	53	22.86	49.52
62	769	826	58	23.20	59.91
63	827	890	64	23.53	60.34
64	891	961	71	23.86	60.79
65	962	1023	62	24.00	65.89

Table B.2.1.7.b -- Psychoacoustic parameters for 32 KHz short FFT

Index	w_low	w_high	width	bval	qsthr
0	0	0	1	0.00	27.28
1	1	1	1	1.26	22.28
2	2	2	1	2.50	19.28

3	3	3	1	3.70	14.28
4	4	4	1	4.85	12.28
5	5	5	1	5.92	12.28
6	6	6	1	6.93	12.28
7	7	7	1	7.85	12.28
8	8	8	1	8.70	12.28
9	9	9	1	9.49	12.28
10	10	10	1	10.20	12.28
11	11	11	1	10.85	12.28
12	12	12	1	11.45	12.28
13	13	13	1	12.00	12.28
14	14	14	1	12.50	12.28
15	15	15	1	12.96	12.28
16	16	16	1	13.39	12.28
17	17	17	1	13.78	12.28
18	18	18	1	14.15	12.28
19	19	20	2	14.80	15.29
20	21	22	2	15.38	15.29
21	23	24	2	15.89	15.29
22	25	26	2	16.36	15.29
23	27	28	2	16.77	15.29
24	29	30	2	17.15	15.29
25	31	32	2	17.50	15.29
26	33	35	3	17.90	17.05
27	36	38	3	18.34	17.05
28	39	41	3	18.74	17.05
29	42	44	3	19.11	17.05
30	45	48	4	19.50	18.30
31	49	52	4	19.92	18.30
32	53	56	4	20.30	21.30
33	57	60	4	20.65	21.30
34	61	65	5	21.02	24.27
35	66	70	5	21.40	24.27
36	71	75	5	21.75	24.27
37	76	81	6	22.10	30.06
38	82	87	6	22.45	30.06
39	88	94	7	22.80	30.73
40	95	102	8	23.16	41.31
41	103	110	8	23.51	41.31
42	111	119	9	23.85	41.82
43	120	127	8	24.00	60.47

Table B.2.1.8.a -- Psychoacoustic parameters for 44.1 KHz long FFT

Index	w_low	w_high	width	bval	qsthr
0	0	1	2	0.22	40.29
1	2	3	2	0.65	40.29
2	4	5	2	1.09	35.29
3	6	7	2	1.52	35.29
4	8	9	2	1.95	35.29
5	10	11	2	2.37	32.29
6	12	13	2	2.79	32.29
7	14	15	2	3.21	27.29
8	16	17	2	3.62	27.29
9	18	19	2	4.02	25.29
10	20	21	2	4.41	25.29
11	22	23	2	4.80	25.29

12	24	25	2	5.18	25.29
13	26	27	2	5.55	25.29
14	28	29	2	5.92	25.29
15	30	31	2	6.27	25.29
16	32	33	2	6.62	25.29
17	34	35	2	6.95	25.29
18	36	38	3	7.36	27.05
19	39	41	3	7.83	27.05
20	42	44	3	8.28	27.05
21	45	47	3	8.71	27.05
22	48	50	3	9.12	27.05
23	51	53	3	9.52	27.05
24	54	56	3	9.89	27.05
25	57	59	3	10.25	27.05
26	60	62	3	10.59	27.05
27	63	66	4	10.97	28.30
28	67	70	4	11.38	28.30
29	71	74	4	11.77	28.30
30	75	78	4	12.13	28.30
31	79	82	4	12.48	28.30
32	83	87	5	12.84	29.27
33	88	92	5	13.22	29.27
34	93	97	5	13.57	29.27
35	98	103	6	13.93	30.06
36	104	109	6	14.30	30.06
37	110	116	7	14.67	30.73
38	117	123	7	15.03	30.73
39	124	131	8	15.40	31.31
40	132	139	8	15.76	31.31
41	140	148	9	16.11	31.82
42	149	157	9	16.45	31.82
43	158	167	10	16.79	32.28
44	168	178	11	17.13	32.69
45	179	190	12	17.48	33.07
46	191	203	13	17.83	33.42
47	204	217	14	18.18	33.74
48	218	232	15	18.52	34.04
49	233	248	16	18.87	34.32
50	249	265	17	19.21	34.58
51	266	283	18	19.54	34.83
52	284	303	20	19.88	35.29
53	304	324	21	20.22	38.50
54	325	347	23	20.56	38.89
55	348	371	24	20.90	39.08
56	372	397	26	21.24	41.43
57	398	425	28	21.57	41.75
58	426	455	30	21.91	42.05
59	456	488	33	22.24	47.46
60	489	524	36	22.58	47.84
61	525	563	39	22.91	48.19
62	564	606	43	23.25	58.61
63	607	653	47	23.58	59.00
64	654	706	53	23.91	59.52
65	707	765	59	24.00	69.98
66	766	832	67	24.00	70.54
67	833	908	76	24.00	71.08
68	909	996	88	24.00	71.72
69	997	1023	27	24.00	72.09

Table B.2.1.8.b -- Psychoacoustic parameters for 44.1 KHz short FFT

Index	w_low	w_high	width	bval	qsthr
0	0	0	1	0.00	27.28
1	1	1	1	1.73	22.28
2	2	2	1	3.41	14.28
3	3	3	1	4.99	12.28
4	4	4	1	6.45	12.28
5	5	5	1	7.75	12.28
6	6	6	1	8.92	12.28
7	7	7	1	9.96	12.28
8	8	8	1	10.87	12.28
9	9	9	1	11.68	12.28
10	10	10	1	12.39	12.28
11	11	11	1	13.03	12.28
12	12	12	1	13.61	12.28
13	13	13	1	14.12	12.28
14	14	14	1	14.59	12.28
15	15	15	1	15.01	12.28
16	16	16	1	15.40	12.28
17	17	17	1	15.76	12.28
18	18	19	2	16.39	15.29
19	20	21	2	16.95	15.29
20	22	23	2	17.45	15.29
21	24	25	2	17.89	15.29
22	26	27	2	18.30	15.29
23	28	29	2	18.67	15.29
24	30	31	2	19.02	15.29
25	32	34	3	19.41	17.05
26	35	37	3	19.85	17.05
27	38	40	3	20.25	20.05
28	41	43	3	20.62	20.05
29	44	47	4	21.01	23.30
30	48	51	4	21.43	23.30
31	52	55	4	21.81	23.30
32	56	59	4	22.15	28.30
33	60	64	5	22.51	29.27
34	65	69	5	22.87	29.27
35	70	75	6	23.23	40.06
36	76	81	6	23.59	40.06
37	82	88	7	23.93	40.73
38	89	96	8	24.00	51.31
39	97	105	9	24.00	51.82
40	106	115	10	24.00	52.28
41	116	127	12	24.00	53.07

Table B.2.1.9.a -- Psychoacoustic parameters for 48 KHz long FFT

Index	w_low	w_high	width	bval	qsthr
0	0	1	2	0.24	40.29
1	2	3	2	0.71	40.29
2	4	5	2	1.18	35.29
3	6	7	2	1.65	35.29
4	8	9	2	2.12	32.29
5	10	11	2	2.58	32.29

6	12	13	2	3.03	27.29
7	14	15	2	3.48	27.29
8	16	17	2	3.92	27.29
9	18	19	2	4.35	25.29
10	20	21	2	4.77	25.29
11	22	23	2	5.19	25.29
12	24	25	2	5.59	25.29
13	26	27	2	5.99	25.29
14	28	29	2	6.37	25.29
15	30	31	2	6.74	25.29
16	32	33	2	7.10	25.29
17	34	35	2	7.45	25.29
18	36	37	2	7.80	25.29
19	38	40	3	8.20	27.05
20	41	43	3	8.68	27.05
21	44	46	3	9.13	27.05
22	47	49	3	9.55	27.05
23	50	52	3	9.96	27.05
24	53	55	3	10.35	27.05
25	56	58	3	10.71	27.05
26	59	61	3	11.06	27.05
27	62	65	4	11.45	28.30
28	66	69	4	11.86	28.30
29	70	73	4	12.25	28.30
30	74	77	4	12.62	28.30
31	78	81	4	12.96	28.30
32	82	86	5	13.32	29.27
33	87	91	5	13.70	29.27
34	92	96	5	14.05	29.27
35	97	102	6	14.41	30.06
36	103	108	6	14.77	30.06
37	109	115	7	15.13	30.73
38	116	122	7	15.49	30.73
39	123	130	8	15.85	31.31
40	131	138	8	16.20	31.31
41	139	147	9	16.55	31.82
42	148	157	10	16.91	32.28
43	158	167	10	17.25	32.28
44	168	178	11	17.59	32.69
45	179	190	12	17.93	33.07
46	191	203	13	18.28	33.42
47	204	217	14	18.62	33.74
48	218	232	15	18.96	34.04
49	233	248	16	19.30	34.32
50	249	265	17	19.64	34.58
51	266	283	18	19.97	34.83
52	284	303	20	20.31	38.29
53	304	324	21	20.65	38.50
54	325	347	23	20.99	38.89
55	348	371	24	21.33	41.08
56	372	397	26	21.66	41.43
57	398	425	28	21.99	41.75
58	426	456	31	22.32	47.19
59	457	490	34	22.66	47.59
60	491	527	37	23.00	47.96
61	528	567	40	23.33	58.30
62	568	612	45	23.67	58.81
63	613	662	50	24.00	69.27

64	663	718	56	24.00	69.76
65	719	781	63	24.00	70.27
66	782	853	72	24.00	70.85
67	854	937	84	24.00	71.52
68	938	1023	86	24.00	70.20

Table B.2.1.9.b -- Psychoacoustic parameters for 48 KHz short FFT

Index	w_low	w_high	width	bval	qsthr
0	0	0	1	0.00	27.28
1	1	1	1	1.88	22.28
2	2	2	1	3.70	14.28
3	3	3	1	5.39	12.28
4	4	4	1	6.93	12.28
5	5	5	1	8.29	12.28
6	6	6	1	9.49	12.28
7	7	7	1	10.53	12.28
8	8	8	1	11.45	12.28
9	9	9	1	12.26	12.28
10	10	10	1	12.96	12.28
11	11	11	1	13.59	12.28
12	12	12	1	14.15	12.28
13	13	13	1	14.65	12.28
14	14	14	1	15.11	12.28
15	15	15	1	15.52	12.28
16	16	16	1	15.90	12.28
17	17	18	2	16.56	15.29
18	19	20	2	17.15	15.29
19	21	22	2	17.66	15.29
20	23	24	2	18.13	15.29
21	25	26	2	18.54	15.29
22	27	28	2	18.93	15.29
23	29	30	2	19.28	15.29
24	31	33	3	19.69	17.05
25	34	36	3	20.14	20.05
26	37	39	3	20.54	20.05
27	40	42	3	20.92	20.05
28	43	45	3	21.27	22.05
29	46	49	4	21.64	23.30
30	50	53	4	22.03	28.30
31	54	57	4	22.39	28.30
32	58	62	5	22.76	29.27
33	63	67	5	23.13	39.27
34	68	73	6	23.49	40.06
35	74	79	6	23.85	40.06
36	80	86	7	24.00	50.73
37	87	94	8	24.00	51.31
38	95	103	9	24.00	51.82
39	104	113	10	24.00	52.28
40	114	125	12	24.00	53.07
41	126	127	1	24.00	53.07

Table B.2.1.10.a -- Psychoacoustic parameters for 64 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	1	2	0,32	40,29
1	2	3	2	0,95	40,29

2	4	5	2	1,57	35,29
3	6	7	2	2,19	32,29
4	8	9	2	2,80	32,29
5	10	11	2	3,40	27,29
6	12	13	2	3,99	27,29
7	14	15	2	4,56	25,29
8	16	17	2	5,12	25,29
9	18	19	2	5,66	25,29
10	20	21	2	6,18	25,29
11	22	23	2	6,68	25,29
12	24	25	2	7,16	25,29
13	26	27	2	7,63	25,29
14	28	29	2	8,07	25,29
15	30	31	2	8,50	25,29
16	32	33	2	8,90	25,29
17	34	35	2	9,29	25,29
18	36	37	2	9,67	25,29
19	38	39	2	10,03	25,29
20	40	41	2	10,37	25,29
21	42	44	3	10,77	27,05
22	45	47	3	11,23	27,05
23	48	50	3	11,66	27,05
24	51	53	3	12,06	27,05
25	54	56	3	12,44	27,05
26	57	59	3	12,79	27,05
27	60	63	4	13,18	28,30
28	64	67	4	13,59	28,30
29	68	71	4	13,97	28,30
30	72	75	4	14,32	28,30
31	76	80	5	14,69	29,27
32	81	85	5	15,07	29,27
33	86	90	5	15,42	29,27
34	91	96	6	15,77	30,06
35	97	102	6	16,13	30,06
36	103	109	7	16,49	30,73
37	110	116	7	16,85	30,73
38	117	124	8	17,20	31,31
39	125	132	8	17,54	31,31
40	133	141	9	17,88	31,82
41	142	151	10	18,23	32,28
42	152	161	10	18,58	32,28
43	162	172	11	18,91	32,69
44	173	184	12	19,25	33,07
45	185	197	13	19,60	33,42
46	198	211	14	19,94	33,74
47	212	226	15	20,29	37,04
48	227	242	16	20,63	37,32
49	243	259	17	20,97	37,58
50	260	277	18	21,31	39,83
51	278	297	20	21,64	40,29
52	298	318	21	21,98	40,50
53	319	341	23	22,31	45,89
54	342	366	25	22,65	46,26
55	367	394	28	22,98	46,75
56	395	424	30	23,32	57,05
57	425	458	34	23,66	57,59
58	459	495	37	23,99	57,96
59	496	537	42	24,00	68,51

60	538	584	47	24,00	69,00
61	585	638	54	24,00	69,60
62	639	701	63	24,00	70,27
63	702	774	73	24,00	70,91
64	775	861	87	24,00	71,67
65	862	966	105	24,00	72,49
66	967	1023	57	24,00	69,83

Table B.2.1.10.b -- Psychoacoustic parameters for 64 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	2,50	19,28
2	2	2	1	4,85	12,28
3	3	3	1	6,93	12,28
4	4	4	1	8,70	12,28
5	5	5	1	10,20	12,28
6	6	6	1	11,45	12,28
7	7	7	1	12,50	12,28
8	8	8	1	13,39	12,28
9	9	9	1	14,15	12,28
10	10	10	1	14,81	12,28
11	11	11	1	15,39	12,28
12	12	12	1	15,90	12,28
13	13	13	1	16,36	12,28
14	14	14	1	16,78	12,28
15	15	15	1	17,16	12,28
16	16	17	2	17,82	15,29
17	18	19	2	18,40	15,29
18	20	21	2	18,92	15,29
19	22	23	2	19,39	15,29
20	24	25	2	19,82	15,29
21	26	27	2	20,21	18,29
22	28	29	2	20,57	18,29
23	30	32	3	20,98	20,05
24	33	35	3	21,43	22,05
25	36	38	3	21,84	22,05
26	39	41	3	22,22	27,05
27	42	45	4	22,61	28,30
28	46	49	4	23,02	38,30
29	50	53	4	23,39	38,30
30	54	58	5	23,75	39,27
31	59	63	5	24,00	49,27
32	64	69	6	24,00	50,06
33	70	76	7	24,00	50,73
34	77	84	8	24,00	51,31
35	85	93	9	24,00	51,82
36	94	104	11	24,00	52,69
37	105	117	13	24,00	53,42
38	118	127	10	24,00	52,28

Table B.2.1.11.a -- Psychoacoustic parameters for 88,2 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	37,28
1	1	1	1	0,44	37,28
2	2	2	1	0,87	37,28
3	3	3	1	1,30	32,28

4	4	4	1	1,73	32,28
5	5	5	1	2,16	29,28
6	6	6	1	2,58	29,28
7	7	7	1	3,00	24,28
8	8	8	1	3,41	24,28
9	9	9	1	3,82	24,28
10	10	10	1	4,22	22,28
11	11	11	1	4,61	22,28
12	12	12	1	4,99	22,28
13	13	13	1	5,37	22,28
14	14	14	1	5,74	22,28
15	15	15	1	6,10	22,28
16	16	16	1	6,45	22,28
17	17	17	1	6,79	22,28
18	18	19	2	7,44	25,29
19	20	21	2	8,05	25,29
20	22	23	2	8,64	25,29
21	24	25	2	9,19	25,29
22	26	27	2	9,70	25,29
23	28	29	2	10,19	25,29
24	30	31	2	10,65	25,29
25	32	33	2	11,08	25,29
26	34	35	2	11,48	25,29
27	36	37	2	11,86	25,29
28	38	39	2	12,22	25,29
29	40	42	3	12,64	27,05
30	43	45	3	13,10	27,05
31	46	48	3	13,53	27,05
32	49	51	3	13,93	27,05
33	52	54	3	14,30	27,05
34	55	58	4	14,69	28,30
35	59	62	4	15,11	28,30
36	63	66	4	15,49	28,30
37	67	70	4	15,84	28,30
38	71	75	5	16,21	29,27
39	76	80	5	16,58	29,27
40	81	85	5	16,92	29,27
41	86	91	6	17,27	30,06
42	92	97	6	17,62	30,06
43	98	104	7	17,97	30,73
44	105	111	7	18,32	30,73
45	112	119	8	18,67	31,31
46	120	127	8	19,02	31,31
47	128	136	9	19,35	31,82
48	137	146	10	19,71	32,28
49	147	156	10	20,05	35,28
50	157	167	11	20,39	35,69
51	168	179	12	20,73	36,07
52	180	192	13	21,08	38,42
53	193	206	14	21,43	38,74
54	207	221	15	21,77	39,04
55	222	237	16	22,11	44,32
56	238	255	18	22,45	44,83
57	256	274	19	22,80	45,06
58	275	295	21	23,13	55,50
59	296	318	23	23,47	55,89
60	319	344	26	23,81	56,43
61	345	373	29	24,00	66,90

62	374	405	32	24,00	67,33
63	406	442	37	24,00	67,96
64	443	484	42	24,00	68,51
65	485	533	49	24,00	69,18
66	534	591	58	24,00	69,91
67	592	660	69	24,00	70,66
68	661	745	85	24,00	71,57
69	746	851	106	24,00	72,53
70	852	988	137	24,00	73,64
71	989	1023	35	24,00	67,72

Table B.2.1.11.b -- Psychoacoustic parameters for 88,2 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	3,41	14,28
2	2	2	1	6,45	12,28
3	3	3	1	8,92	12,28
4	4	4	1	10,87	12,28
5	5	5	1	12,39	12,28
6	6	6	1	13,61	12,28
7	7	7	1	14,59	12,28
8	8	8	1	15,40	12,28
9	9	9	1	16,09	12,28
10	10	10	1	16,69	12,28
11	11	11	1	17,21	12,28
12	12	12	1	17,68	12,28
13	13	13	1	18,11	12,28
14	14	14	1	18,49	12,28
15	15	15	1	18,85	12,28
16	16	17	2	19,48	15,29
17	18	19	2	20,05	18,29
18	20	21	2	20,55	18,29
19	22	23	2	21,01	20,29
20	24	25	2	21,43	20,29
21	26	27	2	21,81	20,29
22	28	29	2	22,15	25,29
23	30	32	3	22,55	27,05
24	33	35	3	22,98	27,05
25	36	38	3	23,36	37,05
26	39	42	4	23,75	38,30
27	43	46	4	24,00	48,30
28	47	51	5	24,00	49,27
29	52	56	5	24,00	49,27
30	57	62	6	24,00	50,06
31	63	69	7	24,00	50,73
32	70	77	8	24,00	51,31
33	78	87	10	24,00	52,28
34	88	99	12	24,00	53,07
35	100	115	16	24,00	54,32
36	116	127	12	24,00	53,07

Table B.2.1.12.a -- Psychoacoustic parameters for 96 KHz long FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	37,28
1	1	1	1	0,47	37,28
2	2	2	1	0,95	37,28

3	3	3	1	1,42	32,28
4	4	4	1	1,88	32,28
5	5	5	1	2,35	29,28
6	6	6	1	2,81	29,28
7	7	7	1	3,26	24,28
8	8	8	1	3,70	24,28
9	9	9	1	4,14	22,28
10	10	10	1	4,57	22,28
11	11	11	1	4,98	22,28
12	12	12	1	5,39	22,28
13	13	13	1	5,79	22,28
14	14	14	1	6,18	22,28
15	15	15	1	6,56	22,28
16	16	16	1	6,93	22,28
17	17	17	1	7,28	22,28
18	18	18	1	7,63	22,28
19	19	20	2	8,28	25,29
20	21	22	2	8,90	25,29
21	23	24	2	9,48	25,29
22	25	26	2	10,02	25,29
23	27	28	2	10,53	25,29
24	29	30	2	11,00	25,29
25	31	32	2	11,45	25,29
26	33	34	2	11,86	25,29
27	35	36	2	12,25	25,29
28	37	38	2	12,62	25,29
29	39	40	2	12,96	25,29
30	41	43	3	13,36	27,05
31	44	46	3	13,80	27,05
32	47	49	3	14,21	27,05
33	50	52	3	14,59	27,05
34	53	55	3	14,94	27,05
35	56	59	4	15,32	28,30
36	60	63	4	15,71	28,30
37	64	67	4	16,08	28,30
38	68	72	5	16,45	29,27
39	73	77	5	16,83	29,27
40	78	82	5	17,19	29,27
41	83	88	6	17,54	30,06
42	89	94	6	17,90	30,06
43	95	101	7	18,26	30,73
44	102	108	7	18,62	30,73
45	109	116	8	18,97	31,31
46	117	124	8	19,32	31,31
47	125	133	9	19,67	31,82
48	134	143	10	20,03	35,28
49	144	153	10	20,38	35,28
50	154	164	11	20,72	35,69
51	165	176	12	21,07	38,07
52	177	189	13	21,42	38,42
53	190	203	14	21,77	38,74
54	204	218	15	22,12	44,04
55	219	234	16	22,46	44,32
56	235	252	18	22,80	44,83
57	253	271	19	23,14	55,06
58	272	292	21	23,47	55,50
59	293	316	24	23,81	56,08
60	317	342	26	24,00	66,43

61	343	372	30	24,00	67,05
62	373	406	34	24,00	67,59
63	407	445	39	24,00	68,19
64	446	490	45	24,00	68,81
65	491	543	53	24,00	69,52
66	544	607	64	24,00	70,34
67	608	685	78	24,00	71,20
68	686	783	98	24,00	72,19
69	784	910	127	24,00	73,31
70	911	1023	113	24,00	72,81

Table B.2.1.12.b -- Psychoacoustic parameters for 96 KHz short FFT

index	w_low	w_high	width	bval	qsthr
0	0	0	1	0,00	27,28
1	1	1	1	3,70	14,28
2	2	2	1	6,93	12,28
3	3	3	1	9,49	12,28
4	4	4	1	11,45	12,28
5	5	5	1	12,96	12,28
6	6	6	1	14,15	12,28
7	7	7	1	15,11	12,28
8	8	8	1	15,90	12,28
9	9	9	1	16,57	12,28
10	10	10	1	17,16	12,28
11	11	11	1	17,67	12,28
12	12	12	1	18,13	12,28
13	13	13	1	18,55	12,28
14	14	14	1	18,93	12,28
15	15	16	2	19,60	15,29
16	17	18	2	20,20	18,29
17	19	20	2	20,73	18,29
18	21	22	2	21,21	20,29
19	23	24	2	21,64	20,29
20	25	26	2	22,03	25,29
21	27	28	2	22,39	25,29
22	29	31	3	22,79	27,05
23	32	34	3	23,23	37,05
24	35	37	3	23,62	37,05
25	38	41	4	24,00	48,30
26	42	45	4	24,00	48,30
27	46	50	5	24,00	49,27
28	51	55	5	24,00	49,27
29	56	61	6	24,00	50,06
30	62	68	7	24,00	50,73
31	69	77	9	24,00	51,82
32	78	88	11	24,00	52,69
33	89	102	14	24,00	53,74
34	103	120	18	24,00	54,83
35	121	127	7	24,00	50,73

Gain Control

Encoding process

The gain control tool consists of a PQF (Polyphase Quadrature Filter), gain detectors and gain modifiers. This tool receives the input time-domain signals and **window_sequence**, and then outputs gain_control_data and a gain controlled signal whose length is equal to the length of the MDCT window. The block diagram for the gain control tool is shown in Figure B.2.2.11.

Due to the characteristics of the PQF filterbank, the order of the MDCT coefficients in each even PQF band needs to be reversed. This is done by reversing the spectral order of the MDCT coefficients, i.e. exchanging the higher frequency MDCT coefficients with the lower frequency MDCT coefficients.

If the gain control tool is used, the configuration of the filterbank tool is changed as follows. In the case of an EIGHT_SHORT_SEQUENCE window_sequence, the number of coefficients for the MDCT is 32 instead of 128 and eight MDCTs are carried out. In the case of other window_sequence values, the number of coefficients for the MDCT is 256 instead of 1024 and one MDCT is carried out. In all cases, the filter bank tool receives a total of 2048 gain controlled signal values per frame, because the input samples have been overlapped.

PQF

The input signal is divided by a PQF into four equal width frequency bands. The coefficients of each band PQF are given as follows.

$$h_i(n) = \frac{1}{4}\cos\left(\frac{(2i+1)(2n+5)\pi}{16}\right)Q(n), \quad 0 \le n \le 95, 0 \le i \le 3$$

where

$$Q(n) = Q(95-n), \quad 48 \le n \le 95$$

and the values of *Q(n)* are the same values as those of the decoder.

Gain detector

The gain detectors produce gain control data which satisfies the bitstream syntax. This information consists of the number of gain changes, the index of gain change positions and the index of gain change level. Note that the output gain control data applies to the previous input time signal. This means that the gain detector has a one frame delay.

The detection of the gain change point is done in the second half of the MDCT window region and in the non-overlapped region (of LONG_START_SEQUENCE and LONG_STOP_SEQUENCE). Thus the number of regions are one for ONLY_LONG_SEQUENCE, two for LONG_START_SEQUENCE and LONG_STOP_SEQUNCE, and eight for EIGHT_SHORT_SEQUENCE.

The samples in each region are divided into subregions, each having eight-tuple samples. Then one value (e.g. peak value of samples) is selected in these subregions. The ratios between the values of subregions and the value of the last subregion are calculated. If the ratio is greater or less than the value of 2 ^ n where n is an integer between -4 to 11, those subregions can be detected as the gain change points of the signals. The subregion number which is detected as the gain change point is set to be the position data. The exponent of the ratio is set to be the gain data. The time resolution of the gain control is approximately 0.7 ms at 48 kHz sampling rate.

Gain modifier

The gain modifier for each PQF band controls the gain of each signal band. The complementary gain control process in the decoder decreases the pre-echo and reconstructs the original signal. A window function for gain control, the Gain Modification Function (GMF), which is defined in the decoding process, is derived from the gains and the gain-changed positions. The gain controlled signals are derived by applying the GMF to the corresponding band signals.

Diagrams

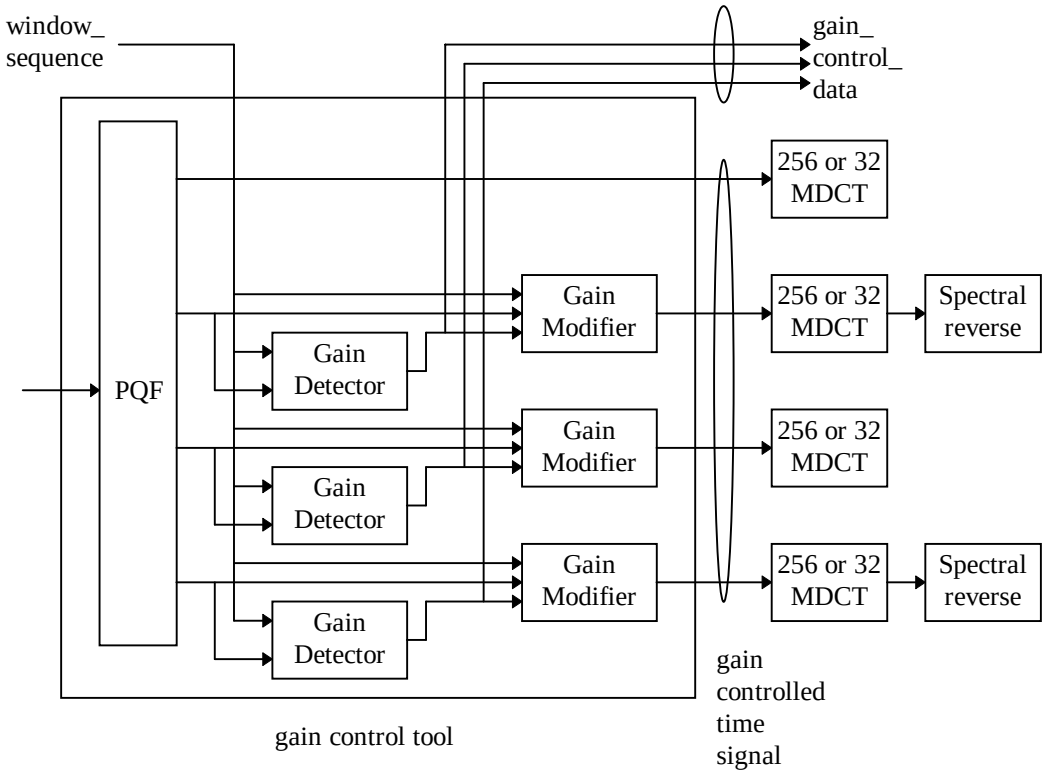


Figure B.2.2.1 -- Block diagram of gain control tool for encoder

Filterbank and Block Switching

A fundamental component in the audio coding process is the conversion of the time domain signals into a time-frequency representation. This conversion is done by a forward modified discrete cosine transform (MDCT).

Encoding process

In the encoder the filterbank takes the appropriate block of time samples, modulates them by an appropriate window function, and performs the MDCT. Each block of input samples is overlapped by 50% with the immediately preceding block and the following block. The transform input block length N can be set to either 2048 or 256 samples. Since the window function has a significant effect on the filterbank frequency response, the filterbank has been designed to allow a change in window shape to best adapt to input signal conditions. The shape of the window is varied simultaneously in the encoder and decoder to allow the filterbank to efficiently separate spectral components of the input for a wider variety of input signals.

Windowing and block switching

The adaptation of the time-frequency resolution of the filterbank to the characteristics of the input signal is done by shifting between transforms whose input lengths are either 2048 or 256 samples. By enabling the block switching tool, the following transitions are meaningful:

from ONLY_LONG_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from LONG_START_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE
from LONG_STOP_SEQUENCE to	{ONLY_LONG_SEQUENCE LONG_START_SEQUENCE
from EIGHT_SHORT_SEQUENCE to	{EIGHT_SHORT_SEQUENCE LONG_STOP_SEQUENCE

Window shape decisions are made by the encoder on a frame-by-frame-basis. The window selected is applicable to the second half of the window function only, since the first half is constrained to use the appropriate window shape from the preceding frame. Figure B.2.3.1 shows the sequence of blocks for the transition (D-E-F) to and from a frame employing the sine function window. The window shape selector generally produces window shape run-lengths greater than that shown in the figure.

The 2048 time-domain values $x'_{i,n}$ to be windowed are the last 1024 values of the previous window_sequence concatenated with 1024 values of the current block. The formula below shows this fact:

$$x'_{i,n} = \begin{cases} x_{(i-1),(n+1024)}, & \text{for } 0 \le n < 1024 \\ x_{i,n}, & \text{for } 1024 \le n < 2048 \end{cases}$$

Where i is the block index and n is the sample index within a block. Once the window shape is selected, the **window_shape** syntax element is initialized. Together with the chosen **window_sequence** all information needed for windowing exist.

With the window halves described below all **window_sequences** can be assembled.

For **window_shape** == 1, the window coefficients are given by the Kaiser - Bessel derived (KBD) window as follows:

$$W_{KBD\_LEFT,N}(n) = \sqrt{\frac{\sum_{p=0}^{n}\left[W'(n,\alpha)\right]}{\sum_{p=0}^{N/2}\left[W'(p,\alpha)\right]}} \qquad \text{for} \quad 0 \le n < \frac{N}{2}$$

$$W_{KBD\_RIGHT,N}(n) = \sqrt{\frac{\sum_{p=0}^{N-n}\left[W'(n,\alpha)\right]}{\sum_{p=0}^{N/2}\left[W'(p,\alpha)\right]}} \qquad \text{for} \quad \frac{N}{2} \le n < N$$

where:

W', Kaiser - Bessel kernel window function, see also [5], is defined as follows:

$$W'(n) = \frac{I_0\left[\pi\alpha\sqrt{1.0-\left(\frac{n-N/4}{N/4}\right)^2}\right]}{I_0\left[\pi\alpha\right]} \qquad \text{for } 0 \le n \le \frac{N}{2}$$

$$I_0\left[x\right] = \sum_{k=0}^{\infty}\left[\frac{\left(\frac{x}{2}\right)^k}{k!}\right]^2$$

α = kernel window alpha factor, $\alpha = \begin{cases} 4 \text{ for } N = 2048 \\ 6 \text{ for } N = 256 \end{cases}$

Otherwise, for **window_shape** == 0, a sine window is employed as follows:

$$W_{SIN\_LEFT,N}(n) = \sin(\frac{\pi}{N}(n+\frac{1}{2})) \qquad \text{for} \quad 0 \le n < \frac{N}{2}$$

$$W_{SIN\_RIGHT,N}(n) = \sin(\frac{\pi}{N}(n+\frac{1}{2})) \qquad \text{for} \quad \frac{N}{2} \le n < N$$

The window length N can be 2048 or 256 for the KBD and the sine window. How to obtain the possible window sequences is explained in the parts a)-d) of this clause. All four window_sequences explained below have a total length of 2048 samples.

For all kinds of window_sequences the window_shape of the left half of the first transform window is determined by the window shape of the previous block. The following formula expresses this fact:

$$W_{LEFT,N}(n) = \begin{cases} W_{KBD\_LEFT,N}(n), \text{ if } & window\_shape\_previous\_block == 1 \\ W_{SIN\_LEFT,N}(n), \text{ if } & window\_shape\_previous\_block == 0 \end{cases}$$

where:

window_shape_previous_block: **window_shape** of the block (i-1).

For the first block to be encoded the **window_shape** of the left and right half of the window are identical.

a) ONLY_LONG_SEQUENCE:

The **window_sequence** == ONLY_LONG_SEQUENCE is equal to one LONG_WINDOW (see Table 8.3) with a total window length of 2048.

For **window_shape**==1 the ONLY_LONG_SEQUENCE is given as follows:

$$w(n) = \begin{cases} W_{LEFT,2048}(n), & \text{for } 0 \le n < 1024 \\ W_{KBD\_RIGHT,2048}(n), & \text{for } 1024 \le n < 2048 \end{cases}$$

If **window_shape**==0 the ONLY_LONG_SEQUENCE can be described as follows:

$$w(n) = \begin{cases} W_{LEFT,2048}(n), & \text{for } 0 \le n < 1024 \\ W_{SIN\_RIGHT,2048}(n), & \text{for } 1024 \le n < 2048 \end{cases}$$

The time domain values after windowing ($z_{i,n}$)can be expressed as:

$$z_{i,n} = w(n) \cdot x'_{i,n};$$

b) LONG_START_SEQUENCE:

The LONG_START_SEQUENCE is needed to obtain a correct overlap and add for a block transition from a ONLY_LONG_SEQUENCE to a EIGHT_SHORT_SEQUENCE.

If **window_shape**==1 the LONG_START_SEQUENCE is given as follows:

$$w(n) = \begin{cases} W_{LEFT,2048}(n), & \text{for } 0 \le n < 1024 \\ 1.0, & \text{for } 1024 \le n < 1472 \\ W_{KBD\_RIGHT,256}(n+128-1472), & \text{for } 1472 \le n < 1600 \\ 0.0, & \text{for } 1600 \le n < 2048 \end{cases}$$

If **window_shape**==0 the LONG_START_SEQUENCE looks like:

$$w(n) = \begin{cases} W_{LEFT,2048}(n), & \text{for } 0 \le n < 1024 \\ 1.0, & \text{for } 1024 \le n < 1472 \\ W_{SIN\_RIGHT,256}(n + 128 - 1472), & \text{for } 1472 \le n < 1600 \\ 0.0, & \text{for } 1600 \le n < 2048 \end{cases}$$

The windowed domain values can be calculated with the formula explained in a).

c) EIGHT_SHORT

The **window_sequence** == EIGHT_SHORT comprises eight overlapped SHORT_WINDOWs (see Table 8.3) with a window length of 256 samples each (2048 in total). Each of the eight short windows are windowed separately first. The short window number is indexed with the variable j = 0, …, 7.
The **window_shape** of the previous block influences the first of the eight short windows ($W_0(n)$) only.
If **window_shape**==1 the window functions can be given as follows:

$$W_0(n) = \begin{cases} W_{LEFT,256}(n), & \text{for } 0 \le n < 128 \\ W_{KBD\_RIGHT,256}(n), & \text{for } 128 \le n < 256 \end{cases}$$

$$W_{1-7}(n) = \begin{cases} W_{KBD\_LEFT,256}(n), & \text{for } 0 \le n < 128 \\ W_{KBD\_RIGHT,256}(n), & \text{for } 128 \le n < 256 \end{cases}$$

Otherwise, if **window_shape**==0, the window functions can be described as:

$$W_0(n) = \begin{cases} W_{LEFT,256}(n), & \text{for } 0 \le n < 128 \\ W_{SIN\_RIGHT,256}(n), & \text{for } 128 \le n < 256 \end{cases}$$

$$W_{1-7}(n) = \begin{cases} W_{SIN\_LEFT,256}(n), & \text{for } 0 \le n < 128 \\ W_{SIN\_RIGHT,256}(n), & \text{for } 128 \le n < 256 \end{cases}$$

An EIGHT_SHORT sequence $x'_{i,n}$ is subdivided into eight overlapping segments first and then multiplied with the appropriate window function:

$$z_{i,n} = \begin{cases} x'_{i,n+448} \cdot W_0(n), & \text{for } 0 \le n < 256 \\ x'_{i,n+576} \cdot W_1(n-256), & \text{for } 256 \le n < 512 \\ x'_{i,n+704} \cdot W_2(n-512), & \text{for } 512 \le n < 768 \\ x'_{i,n+832} \cdot W_3(n-768), & \text{for } 768 \le n < 1024 \\ x'_{i,n+960} \cdot W_4(n-1024), & \text{for } 1024 \le n < 1280 \\ x'_{i,n+1088} \cdot W_5(n-1280), & \text{for } 1280 \le n < 1536 \\ x'_{i,n+1216} \cdot W_6(n-1536), & \text{for } 1536 \le n < 1792 \\ x'_{i,n+1344} \cdot W_7(n-1792), & \text{for } 1792 \le n < 2048 \end{cases}$$

d) LONG_STOP_SEQUENCE

This window_sequence is needed to switch from a EIGHT_SHORT_SEQUENCE back to a ONLY_LONG_SEQUENCE.

If **window_shape**==1 the LONG_STOP_SEQUENCE is given as follows:

$$w(n) = \begin{cases} 0.0, & \text{for } 0 \le n < 448 \\ W_{LEFT,256}(n-448), & \text{for } 448 \le n < 576 \\ 1.0, & \text{for } 576 \le n < 1024 \\ W_{KBD\_RIGHT,2048}(n), & \text{for } 1024 \le n < 2048 \end{cases}$$

If **window_shape**==0 the LONG_START_SEQUENCE is determined by:

$$w(n) = \begin{cases} 0.0, & \text{for } 0 \le n < 448 \\ W_{LEFT,256}(n-448), & \text{for } 448 \le n < 576 \\ 1.0, & \text{for } 576 \le n < 1024 \\ W_{SIN\_RIGHT,2048}(n), & \text{for } 1024 \le n < 2048 \end{cases}$$

The windowed time domain values can be calculated with the formula explained in a).

MDCT

The spectral coefficient, $X_{i,k}$, are defined as follows:

$$X_{i,k} = 2 \cdot \sum_{n=0}^{N-1} z_{i,n} \cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right) \text{ for } 0 \le k < N/2 .$$

where:

z_{in} = windowed input sequence

n = sample index

k = spectral coefficient index

i = block index

N = window length of the one transform window based on the window_ sequence value

n_0 = (N / 2 + 1) / 2

The analysis window length N of one transform window of the mdct is a function of the syntax element **window_sequence** and is defined as follows:

$$N = \begin{cases} 2048, & \text{if ONLY\_LONG\_SEQUENCE (0x0)} \\ 2048, & \text{if LONG\_START\_SEQUENCE (0x1)} \\ 256, & \text{if EIGHT\_SHORT\_SEQUENCE (0x2) (8 times)} \\ 2048, & \text{if LONG\_STOP\_SEQUENCE (0x3)} \end{cases}$$

Diagrams

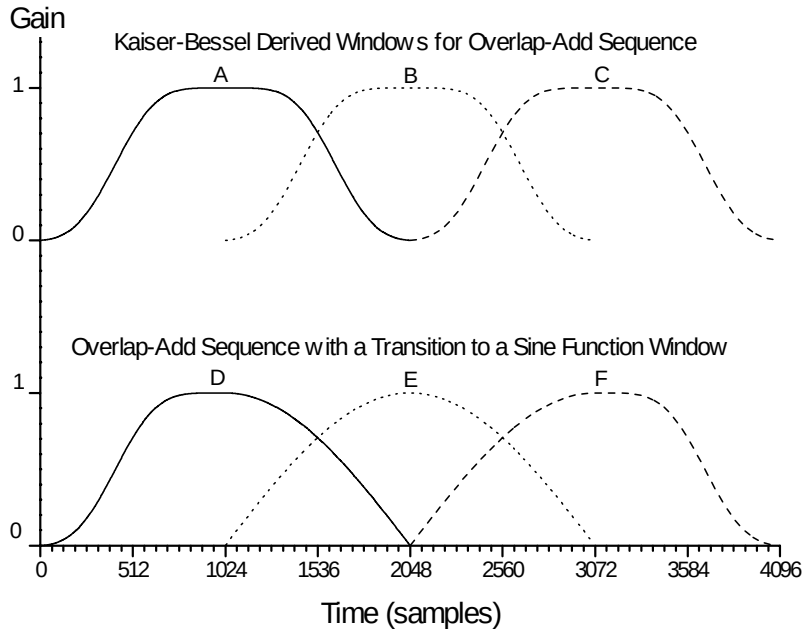


Figure B.2.3.1 -- Example of the Window Shape Adaptation Process.

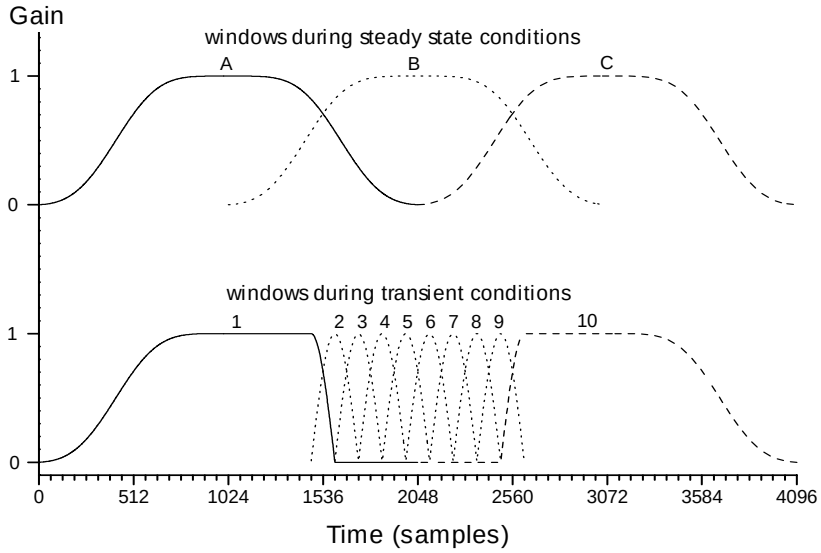


Figure B.2.3.2 – Example of Block Switching During Transient Signal Conditions

Prediction

Tool description

Since each predictor itself is identical on both, the encoder and decoder side, all descriptions and definitions as specified for the decoder in section 13 of the normative part are also valid here.

Prediction is used for an improved redundancy reduction and is especially effective in case of more or less stationary parts of a signal which belong to the most demanding parts in terms of required bitrate. Prediction can be applied to every channel using an intra channel (or mono) predictor which exploits the auto-correlation between the spectral components of consecutive frames. Because a window_sequence of type EIGHT_SHORT_SEQUENCE indicates signal changes, i.e. non-stationary signal characteristic, prediction is only used if window_sequence is of type ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE or LONG_STOP_SEQUENCE.

For each channel prediction is applied to the spectral components resulting from the spectral decomposition of the filterbank. For each spectral component up to limit specified by PRED_SFB_MAX, there is one corresponding predictor resulting in a bank of predictors, where each predictor exploits the auto-correlation between the spectral component values of consecutive frames.

The overall coding structure using a filterbank with high spectral resolution implies the use of backward adaptive predictors to achieve high coding efficiency. In this case, the predictor coefficients are calculated from preceding quantized spectral components in the encoder as well as in the decoder and no additional side information is needed for the transmission of predictor coefficients - as would be required for forward adaptive predictors. A second order backward-adaptive lattice structure predictor is used for each spectral component, so that each predictor is working on the spectral component values of the two preceding frames. The predictor parameters are adapted to the current signal statistics on a frame by frame base, using an LMS based adaptation algorithm. If prediction is activated, the quantizer is fed with a prediction error instead of the original spectral component, resulting in a coding gain.

Encoding process

For each spectral component up to the limit specified by PRED_SFB_MAX of each channel there is one predictor. The following description is valid for one single predictor and has to be applied to each predictor. As said above, each predictor is identical on both, the encoder and decoder side. Therefore, the predictor structure is the same as shown in figure 8.1 and the calculations of the estimate $x_{est}(n)$ of the current spectral component $x(n)$ as well as the calculation and adaptation of the predictor coefficients are exactly the same as those described for the decoder in clause 8.3.2 of the normative part.

The only difference on the encoder side is that the prediction error has to be calculated according to

$$e(n) = x(n) - x_{est}(n)$$

to be fed to the quantizer. In this case the quantized prediction error is transmitted instead of the quantized spectral component.

Predictor control

In order to guarantee that prediction is only used if this results in a coding gain, an appropriate predictor control is required and a small amount of predictor control information has to be transmitted to the decoder. For the predictor control, the predictors are grouped into scalefactor bands.

The following description is valid for either one single_channel_element or one channel_pair_element and has to be applied to each such element. Since prediction is only used if window_sequence is of type ONLY_LONG_SEQUENCE, LONG_START_SEQUENCE or LONG_STOP_SEQUENCE for the channel associated with the single_channel_element or for both channels associated with the channel_pair_element, the following applies only in these cases.

The predictor control information for each frame, which has to be transmitted as side information, is determined in two steps. First, it is determined for each scalefactor band whether or not prediction leads to a coding gain and if yes, the **prediction_used** bit for that scalefactor band is set to one. After this has been done for all scalefactor bands up to PRED_SFB_MAX, it is determined whether the overall coding gain by prediction in this frame compensates at least the additional bit need for the predictor side information. If yes, the **predictor_data_present** bit is set to 1, the complete side information including that needed for predictor reset (see below) has to be transmitted and the prediction error value is fed to the quantizer. Otherwise, the **predictor_data_present** bit is set to 0, the **prediction_used** bits are all reset to zero and are not transmitted. In this case, the spectral component value is fed to the quantizer. Figure B 2.4.1 shows a block diagram of the prediction unit for one scalefactor band. As described above, the predictor control first operates on all predictors of one scalefactor band and is then followed by a second step over all scalefactor bands.

In case of a single_channel_element or a channel_pair_element with **common_window** = 0 the control information is calculated and valid for the predictor bank(s) of the channel(s) associated with that element. In case of a channel_pair_element with **common_window** = 1 the control information is calculated considering both channels associated with that element together. In this case the control information is valid for both predictor banks of the two channels in common.

Predictor reset

When the encoding process is started, all predictors are initialized. This means that the predictor state variables for each predictor are set as follows: $r_0 = r_1 = 0$, $COR_1 = COR_2 = 0$, $VAR_1 = VAR_2 = 1$.

A cyclic reset mechanism is applied by the encoder, where all predictors are initialized again in a certain time interval in an interleaved way. On one hand this increases the stability by re-synchronizing the predictors of the encoder and the decoder and on the other hand it allows defined entry points in the bitstream.

The whole set of predictors is subdivided into 30 so-called reset groups according to the following table:

Reset group number	Predictors of reset group
1	P_0, P_{30}, P_{60}, P_{90},...
2	P_1, P_{31}, P_{61}, P_{91},...
3	P_2, P_{32}, P_{62}, P_{92},...
...	
30	P_{29}, P_{59}, P_{89}, P_{119},...

where P_i is the predictor which corresponds to the spectral coefficient indexed by i.

An encoder is required to reset at least one group every 8 frames and to reset every group within the maximum reset interval of $8 \times 30 = 240$ frames. It is recommended that groups be reset in ascending order, although that is not mandatory. Shorter reset intervals are supported by the bitstream syntax, for example if one group is reset every fourth frame, then all predictors can be reset within an interval of 4 x 30 = 120 frames.

A typical reset cycle starts with reset group number 1 and the reset group number is then incremented by 1 until it reaches 30, and then it starts with 1 again. Nevertheless, it may happen, e.g. due to switching between programs (bitstreams) or cutting and pasting, that there will be a discontinuity in the reset group numbering. If this is the case, there are the following three possibilities for the decoder to operate:

- Ignore the discontinuity and carry on the normal processing. This may result in a short audible distortion due to a mismatch (drift) between the predictors in the encoder and decoder. After one complete reset cycle (reset group n, n+1, ..., 30, 1, 2, ..., n-1) the predictors are re-synchronized again. Furthermore, a possible distortion is faded out because of the attenuation factors a and b.
- Detect the discontinuity, carry on the normal processing but mute the output until one complete reset cycle is performed and the predictors are re-synchronized again.
- Reset all predictors.

The reset mechanism is controlled by the **pred_reset** bit, which always has to be transmitted as soon as the **predictor_data_present** bit is set to 1, and the **pred_reset_group_number**, which specifies the group of predictors to be reset and has only to be transmitted if the **pred_reset** bit is set to 1. If a group reset is applied in the current frame, the **pred_reset** bit has to be set to 1 and the number of the predictor group to be reset has to be coded and transmitted as 5 bit unsigned binary number in **pred_reset_group_number**. After the quantized value has been reconstructed (see below), the reset of the specified group is then carried out by initializing all predictors belonging to that group as described above.

If no group reset is applied in the current frame, the **pred_reset** bit has to be set to 0.

In case of a single_channel_element or a channel_pair_element with **common_window** = 0 the reset has to be applied to the predictor bank(s) of the channel(s) associated with that element and the control information relates to each individual channel. In case of a channel_pair_element with **common_window** = 1 the reset has to be applied to the two predictor banks of the two channels associated with that element and the control information relates to both channels together.

In case of short blocks, i.e. window_sequence is of type EIGHT_SHORT_SEQUENCE, prediction is always disabled and a reset is carried out for all predictors in all scalefactor bands, which is equivalent to a complete reinitialization, see above.

Reconstruction of the quantized spectral component

Since the reconstructed value of the quantized spectral component is needed as predictor input signal, it has to be calculated in the encoder, see also figure 13.2 in the normative part and figure B.2.4.1. Depending on the value of the **prediction_used** bit, the reconstructed value is either the quantized spectral component or the quantized prediction error. Therefore, the following steps are necessary:

- If the bit is set (1), then the quantized prediction error, reconstructed from data to be transmitted, is added to the estimate $x_{est}(n)$, calculated by the predictor, resulting in the reconstructed value of the quantized spectral component, i.e.
  $$x_{rec}(n) = x_{est}(n) + e_q(n)$$
- If the bit is not set (0), then the quantized value of the spectral component is identical to the value reconstructed directly from the data to be transmitted.

Diagrams

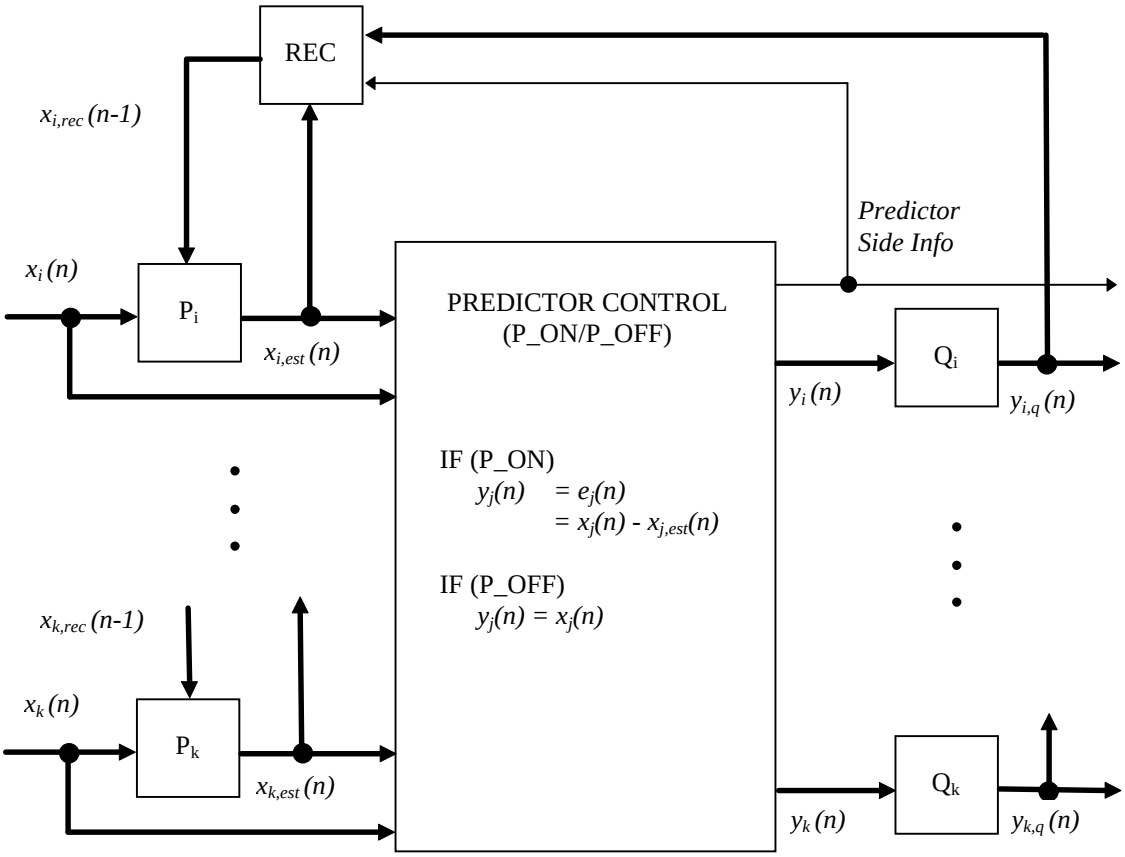


Figure B.2.4.1 -- Block diagram of prediction unit for one scalefactor band. The complete processing is only shown for predictor P_i (Q - quantizer, REC - reconstruction of last quantized value). Note that the predictor control operates on all predictors P_i ... P_j ... P_k of a scalefactor band and is followed by a second control over all scalefactor bands.

Long Term Prediction

The general structure of the perceptual audio coder with LTP is shown in Figure 1. First, the optimal LTP is estimated. Then, the predicted signals based quantized data are obtained and corresponding error signals are calculated. Finally, the error signals are quantized and transmitted with the side information, such as predictor control and predictor parameters.

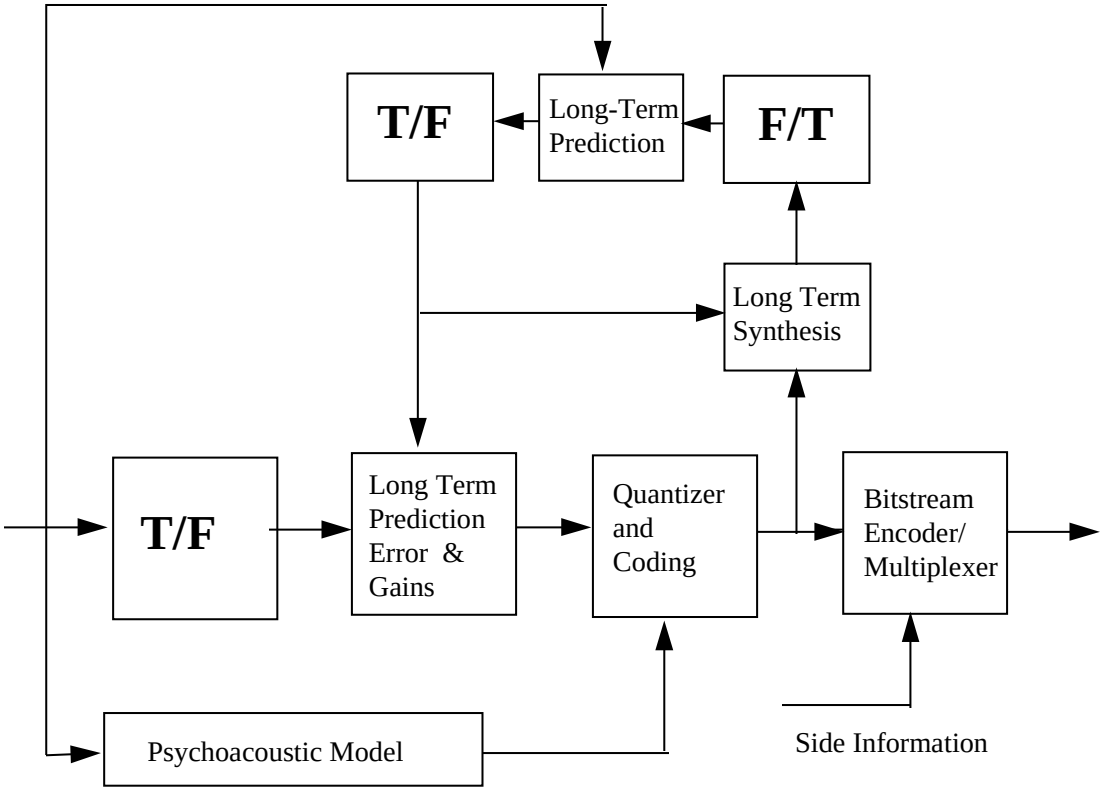


Figure 1. Block diagram of the encoding process of Long Term Prediction

As the perceptual audio coder processes the signal frame by frame, it is obvious that when we want to quantize $\mathbf{X}_{m+1}$, the quantized data $\tilde{\mathbf{X}}_k, k = m, m-1, m-2, \ldots,$ are available both in encoder and decoder. In time domain, it means that the quantised time domain samples $\tilde{x}(i), i = mN, mN-1, \ldots,$ are available.

To remove the redundancy of the signal in present frame $m+1$ based on the previously quantized data, an LTP is used

$$P(z) = \sum_{k=-m_1}^{m_2} b_k z^{-(\alpha+k)}$$

where α represents a long delay in the range 1 to 1024. For $m_1 = m_2 = 0$, we have a one-tap predictor.

The parameters α and b_k are determined by minimizing the mean squared error over the whole window. The delay α is quantized with 10 bits with 1024 possible values in the range 1 to 1024. For prediction coefficients b_k are non-uniformly quantized to 3 bits. Due to their nonuniform distribution, nonuniform quantization has to be used. The table for coding the coefficient values is presented in the decoding process description.

After the LTP is determined, the predicted signal can be obtained for the $(m+1)$th frame

$$\hat{x}(i) = \sum_{k=-m_1}^{m_2} b_k \tilde{x}(i - 2N + 1 - k - \alpha),$$

$$i = mN + 1, mN + 2, \ldots, (m+1)N$$

Using the forward MDCT, we have the predicted spectral coefficient $\tilde{\mathbf{X}}_{m+1}$ for the $(m+1)$th frame. In order to guarantee that prediction is only used if it results in a coding gain, an appropriate predictor control is required and a small amount of predictor control information has to be transmitted to the decoder. The predictor control scheme is the same as the backward predictor control scheme which has been used in MPEG-2 Advanced Audio Coding (AAC), i.e., one bit per scalefactor band is sent as side information to flag whether prediction is used for that band.

In the encoding process the prediction gain resulting from LTP is estimated for each scalefactor band by comparing the predicted energy to the amount of side information needed. LTP is switched on only for those scalefactor bands where there is

positive net coding gain. The corresponding bits of side information are set accordingly. Finally, the net coding gain for the whole frame (all scalefactor bands) is checked, and the switch for using LTP at all for the frame is set accordingly.

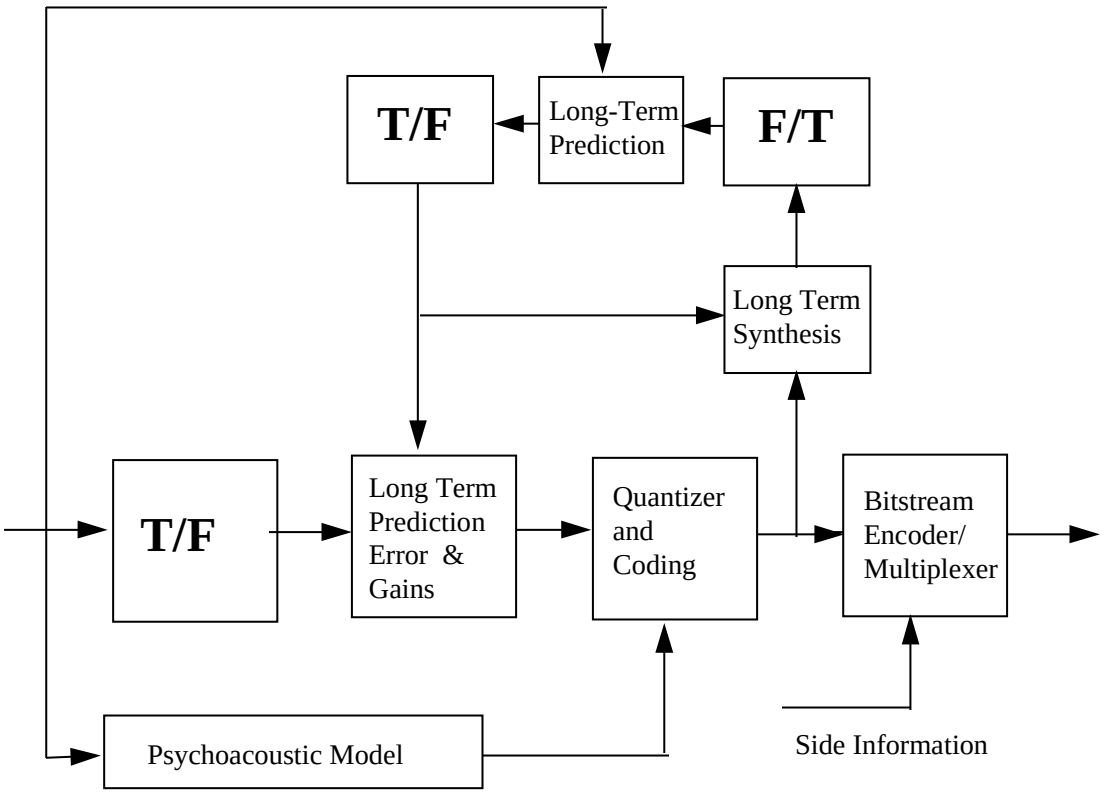


Figure 1. Block diagram of the encoding process of Long Term Prediction

Temporal Noise Shaping (TNS)

Temporal Noise Shaping is used to control the temporal shape of the quantization noise within each window of the transform. This is done by applying a filtering process to parts of the spectral data of each channel.
Encoding is done on a window basis. The following steps are carried out to apply the Temporal Noise Shaping tool to one window of spectral data:

- A target frequency range for the TNS tool is chosen. A suitable choice is to cover a frequency range from 1.5 kHz to the uppermost possible scalefactor band with one filter. Please note that this parameter (TNS_MAX_BANDS) depends on profile and sampling rate as indicated in the normative part.

- Next, a linear predictive coding (LPC) calculation is carried out on the spectral MDCT coefficients corresponding to the chosen target frequency range. For better stability, coefficients corresponding to frequencies below 2.5 kHz may be excluded from this process. Standard LPC procedures as known from speech processing can be used for the LPC calculation, e.g. the well-known Levinson-Durbin algorithm. The calculation is carried out for the maximum permitted order of the noise shaping filter (TNS_MAX_ORDER). Please note that this value depends on the profile as indicated in the normative part.

- As a result of the LPC calculation, the expected prediction gain g_p is known as well as the TNS_MAX_ORDER reflection coefficients r[] (so-called PARCOR coefficients).

- If the prediction gain g_p does not exceed a certain threshold t, no temporal noise shaping is used. In this case, the tns_data_present bit is set to zero and TNS processing is finished. A suitable threshold value is t = 1.4.

- If the prediction gain g_p exceeds the threshold t, temporal noise shaping is used.

- In a next step the reflection coefficients are quantized using coef_res bits. An appropriate choice for coef_res is 4 bits. The following pseudo code describes the conversion of the reflection coefficients r[] to index values index[] and back to quantized reflection coefficients rq[].

```
iqfac   = ((1 << (coef_res-1)) - 0.5) / (π/2.0);
iqfac_m = ((1 << (coef_res-1)) + 0.5) / (π/2.0);

/* Reflection coefficient quantization */
for (i=0; i<TNS_MAX_ORDER; i++)  {
    index[i] = NINT( arcsin( r[i] ) * ((r[i] >= 0) ? iqfac : iqfac_m) );
}
/* Inverse quantization */
for (i=0; i<TNS_MAX_ORDER; i++)  {
    rq[i] = sin( index[i] / ((index[i] >= 0) ? iqfac : iqfac_m) );
}
```

where arcsin() denotes the inverse sin() function.

- The order of the used noise shaping filter is determined by subsequently removing all reflection coefficients with an absolute value smaller than a threshold p from the "tail" of the reflection coefficient array. The number of the remaining reflection coefficients is the order of the noise shaping filter. A suitable threshold for truncation is p = 0.1.

- The remaining reflection coefficients rq[] are converted into order+1 linear prediction coefficients a[] (known as "step-up procedure"). A description of this procedure is provided in the normative part as a part of the tool description (see "/* Conversion to LPC coefficients */").

- The computed LPC coefficients a[] are used as the encoder noise shaping filter coefficients. This FIR filter is slid across the specified target frequency range exactly the way it is described in the normative part for the decoding process (tool description). The difference between the decoding and encoding filtering is that the all-pole (auto-regressive) filter used for decoding is replaced by its inverse all-zero (moving-average) filter, i.e. replacing the decoder filter equation

- ```
  y[n]  =  x[n] - a[1]*y[n-1] - ... - a[order]*y[n-order]
  ```

  by the inverse (encoder) filter equation

  ```
  y[n]  =  x[n] + a[1]*x[n-1] + ... + a[order]*x[n-order]
  ```

  By default, an upward direction of the filtering is appropriate.

- Finally, the side information for Temporal Noise Shaping is transmitted:

Bitstream Element	Algorithmic Variable or Value
n_filt	1
coef_res	coef_res-3
coef_compress	0
length	Number of processed scalefactor bands
direction	0 (upwards)
order	Order of noise shaping filter
coef[]	index[]

Joint Coding

M/S Stereo

The decison to code left and right coefficients as either left + right (L/R) or mid/side (M/S) is made on a noiseless coding band by noiseless coding band basis for all spectral coefficients in the current block. For each noiseless coding band the following decison process is used:

1. For each noiseless coding band, not only L and R raw thresholds, but also M=(L+R)/2 and S=(L-R)/2 raw thresholds are calculated. For the raw M and S thresholds, rather than using the tonality for the M or S threshold, one uses the more tonal value from the L or R calculation in each threshold calculation band, and proceed with the psychoacoustic model for M and S from the M and S energies and the minimum of the L or R values for C(ω) in each threshold calculation band. The
  ```

values that are provided to the imaging control process are identified in the psychoacoustic model information section as en(b) (the spread normalized energy) and nb(b), the raw threshold.

2. The raw thresholds for M, S, L and R, and the spread energy for M, S, L and R, are all brought into an "imaging control process". The resulting adjusted thresholds are inserted as the values for cb(b) into step 11 of the psychoacoustic model for further processing.
3. The final, protected and adapted to coder-band thresholds for all of M,S,L and R are directly applied to the appropriate spectrum by quantizing the actual L, R, M and S spectral values with the appropriate calculated and quantized threshold.
4. The number of bits actually required to code M/S, and the number of bits required to code L/R are calculated.
5. The method that uses the least bits is used in each given noiseless coding band, and the stereo mask is set accordingly.

With these definitions

Mthr,Sthr,Rthr, Lthr	raw thresholds. (the nb(b) from step 10 of the psychoacoustic model)
Mengy,Sengy,Rengy,Sengy	spread energy.(en(b) from step 6 of the psychoacoustic model)
Mfthr, Sfthr, Rfthr, Lfthr	final (output) thresholds. (returned as nb(b) in step 11 of the psychoacoustic model)
bmax(b)	BMLD protection ratio, as can be calculated from

$$bmax(b) = 10^{-3\cdot\left[0.5+0.5\cdot\cos\left(\pi\cdot\frac{\min(bval(b),15.5)}{15.5}\right)\right]}$$

the imaging control process for each noiseless coding band is as follows:

```
t=Mthr/Sthr
if (t>1)
        t=1/t
Rfthr= max(Rthr*t, min (Rthr,  bmax*Rengy)
Lfthr= max(Lthr*t, min (Lthr, bmax) *Lengy)
t=min(Lthr, Rthr)
Mfthr=min(t, max(Mthr, min(Sengy*bmax,Sthr) )
Sfthr=min(t, max(Sthr, min(Mengy*bmax,Mthr) )
```

Intensity Stereo Coding

Intensity stereo coding is used to exploit irrelevance in the between both channels of a channel pair in the high frequency region. The following procedure describes one possible implementation while several different implementations are possible within the framework of the defined bitstream syntax.

Encoding is done on a window group basis. The following steps are carried out to apply the intensity stereo coding tool to one window group of spectral data:

- A suitable approach is to code a consecutive region of scalefactor bands in intensity stereo technique starting above a lower border frequency f_0. An average value of f_0 = 6 kHz is appropriate for most types of signals.

- For each scalefactor band, the energy of the left, right and the sum channel is calculated by summing the squared spectral coefficients, resulting in values $E_l[sfb]$, $E_r[sfb]$, $E_s[sfb]$. If the window group comprises several windows, the energies of the included windows are added.

- For each scalefactor band, the corresponding intensity position value is computed as

$$is\_position[sfb] = NINT\left(2\cdot\log_2\left(\frac{E_l[sfb]}{E_r[sfb]}\right)\right)$$

- Next, the intensity signal spectral coefficients $spec_i[i]$ are calculated for each scalefactor bands by adding spectral samples from the left and right channel ($spec_l[i]$ and $spec_r[i]$) and rescaling the resulting values like

$$spec_i[i] = (spec_l[i] + spec_r[i])\cdot\sqrt{\frac{E_l[sfb]}{E_s[sfb]}}$$

- The intensity signal spectral components are used to replace the corresponding left channel spectral coefficients. The corresponding spectral coefficients of the right channel are set to zero.

Then, the standard process for quantization and encoding is performed on the spectral data of both channels. However, the prediction status of the right channel predictors is forced to "off" for the scalefactor bands coded in intensity stereo. These predictors are updated by using an intensity decoded version of the quantized spectral coefficients. The procedure for this is described in the tool description for the intensity stereo decoding process in the normative part.

Finally, before transmission the Huffman codebook INTENSITY_HCB (15) is set in the sectioning information for all scalefactor bands that are coded in intensity stereo.

Quantization

Introduction

The description of the AAC quantization module is subdivided into three levels. The top level is called "loops frame program". The loops frame program calls a subroutine named "outer iteration loop" which calls the subroutine "inner iteration loop". For each level a corresponding flow diagram is shown.

The loops module quantizes an input vector of spectral data in an iterative process according to several demands. The inner loop quantizes the input vector and increases the quantizer step size until the output vector can be coded with the available number of bits. After completion of the inner loop an outer loop checks the distortion of each scalefactor band and, if the allowed distortion is exceeded, amplifies the scalefactor band and calls the inner loop again.

AAC loops module input:
1. vector of the magnitudes of the spectral values mdct_line(0..1023).
2. xmin(sb) (see B 2.1.4. „Steps in threshold calculation“, Step 12)
3. mean_bits (average number of bits available for encoding the bitstream).
4. more_bits, the number of bits in addition to the average number of bits, calculated by the psychoacoustic module out of the perceptual entropy (PE).
5. the number and width of the scalefactor bands (see table 3.5 normative part)
6. for short block grouping the spectral values have to be interleaved so that spectral lines that belong to the same scalefactor band but to different block types which shall be quantized with the same scalefactors are put together in one (bigger) scalefactor band (for a full description of grouping see clause 3.3.4 normative part)

AAC loops module output:
1. vector of quantized values x_quant(0..1023).
2. a scalefactor for each scalefactor band (sb)
3. common_scalefac (quantizer step size information for all scalefactor bands)
4. number of unused bits available for later use.

Preparatory steps

Reset of all iteration variables

1. The start value of gloabal_gain for the quantizer is calculated so that all quantized MDCT values can be encoded in the bitstream :.

$start\_common\_scalefac = 16/3*(\log_2((max\_mdct\_line \wedge (3/4))/MAX\_QUANT)$

max_mdct_line is the maximum MDCT coefficient, MAX_QUANT is the maximum quantized value witch can be encoded in the bitstream, it is defined to 8191. During the iteration process, the common_scalefac must not become less than start_common_scalefac.

2. All scalefactors are set to zero.

Bit reservoir control

Bits are saved to the reservoir when fewer than the mean_bits are used to code one frame.

*mean_bits = bit_rate * 1024 / sampling_rate.*

The number of bits which can be saved in the bit reservoir at maximum is called 'maximum_bitreservoir_size' which is calculated using the procedure outlined in normative clause 3.2.2. If the reservoir is full, unused bits have to be encoded in the bitstream as fillbits.

The maximum amount of bits available for a frame is the sum of mean_bits and bits saved in the bit reservoir.

The number of bits that should be used for encoding a frame depends on the more_bits value which is calculated by the psychoacoustic model and the maximum available bits. The simplest way to control bit reservoir is :

if more_bits > 0 :
available_bits = average_bits + min (more_bits, bitres_bits)
if more_bits < 0 :
available_bits = average_bits + max (more_bits, bitres_bits - maximum_bitreservoir_size)

Quantization of MDCT coefficients

The formula for the quantization in the encoder is the inverse of the decoder dequatization formula (see also the decoder description) :

*x_quant = int ((abs(mdct_line) * (2^(¼ * (sf_decoder - SF_OFFSET))))^(3/4) + MAGIC_NUMBER)*

MAGIC_NUMBER is defined to 0.4054, SF_OFFSET is defined as 100 and mdct_line is one of spectral values, which is calculated from the MDCT. These values are also called 'coefficients'

For use in the iteration loops, the scalefactor 'sf_decoder' is split in two variables:

sf_decoder = scalefactor - common_scalefac + SF_OFFSET

It follows from this, that the formula used in the distortion control loop is:

*x_quant = int (abs(mdct_line) * (2^(¼ * (scalefactor - common_scalefac))))^(3/4) + MAGIC_NUMBER)*

The sign of the *mdct_line* is saved separately and added again only for counting the bits and encoding the bitstream

Outer iteration loop (distortion control loop)

The outer iteration loop controls the quantization noise which is produced by the quantization of the frequency domain lines within the inner iteration loop. The coloring of the noise is done by multiplication of the lines within scalefactor bands with the actual scalefactors before doing the quantization. The following pseudo-code illustrates the multiplication.

*do for each scalefactor band **sb**:*
*do from lower index to upper index **i** of scalefactor band*
*mdct_scaled(i) = abs(mdct_line(i))^(3/4) * 2^(3/16 * (scalefactor(sb))*
end do
end do

Call of inner iteration loop

For each outer iteration loop (distortion control loop) the inner iteration loop (rate control loop) is called. The parameters are the frequency domain values with the scalefactors applied to the values within the scalefactor bands (mdct_scaled(0..1023)), a start value for common_scalefac, and the number of bits which are available to the rate control loop. The result is the number of bits actually used and the quantized frequency lines x_quant(i), and a new common_scalefac.
The Formula to calculate the quantized MDCT coefficients is:

*x_quant(i) = int ((mdct_scaled (i) * 2^(-3/16 * common_scalefac)) + MAGIC_NUMBER)*

The bits, that would be needed to encode the quantizes values and the side information (scalefactors etc.) are counted according to the bitstream syntax, described in [A 2.8 Noisless Coding].

Amplification of scalefactor bands which violate the masking threshold

The calculation of the distortion (*error_energy(sb)*)of the scalefactor band is done as follows:

```
do for each scalefactor band sb:
    error_energy(sb)=0
    do from lower index to upper index i of scalefactor band
        error_energy(sb) = error_energy(sb) + (abs( mdct_line(i))
                    - (x_quant(i) ^(4/3) * 2^( -¼ * (scalefactor(sb) -common_scalefac ))))^2
    end do
end do
```

All spectral values of the scalefactor bands which have a distortion that exceeds the allowed distortion (*xmin(sb)*)are amplified according to formula in B 2.7.4.1 (“Outer Iteration Loop“), the new scalefactors can calculated according to this pseudocode:

```
do for each scalefactor band sb
        if ( error_energy(sb) > xmin(sb) ) then
                scalefactor(sb) = scalefactor(sb) + 1
        end if
end do
```

Conditions for the termination of the loops processing

Normally the loops processing terminates, if there is no scalefactor band with more than the allowed distortion. However this is not always possible to obtain. In this case there are other conditions to terminate the outer loop. If

- All scalefactor bands are already amplified, or
- The difference between two consecutive scalefactors is greater than 64

The loop processing stops, and by restoring the saved scalefactors(sb) a useful output is available. For real-time implementation, there might be a third condition added which terminates the loops in case of a lack of computing time.

Inner iteration loop (rate control loop)

The inner iteration loop calculates the actual quantization of the frequency domain data (*mdct_scaled*) with the following function, which uses the formula from B.2.7.4.2: „call of Inner iteration loop“ :

```
quantize_spectrum(x_quant[] , mdct_scaled[] , common_scalefac):
        do for all MDCT coefficients i :
                x_quant(i) = int (( mdct_scaled (i) * 2^(-3/16 * common_scalefac)) + MAGIC_NUMBER)
        end do
```

and then calls a function *bit_count()*. This function counts the number of bits that would be neccessary to encode a bitstream according to clause 1 “Syntax“ of the normative part.

The inner iteration loop can be implemented using successive approximation.:

```
inner_loop():
        if (outer_loop_count == 0 )
```

```
            common_scalefac = start_common_scalefac
            quantizer_change = 64
    else
            quantizer_change = 2
    end if
    do
            quantize_spectrum()
            counted_bits = bit_count()
            quantizer change = quantizer_change / 2
            if (counted_bits > available_bits) then
                    common_scalefac = common_scalefac + quantizer_change
            else
                    common_scalefac = common_scalefac - quantizer_change
            end if
            if (quantizer_change == 0) && (counted_bits > available_bits)
                    quantizer_change = 1
            end if
    while ( quantizer_change != 0 )
```

Due to the choice of *start_common_scalefac* calculated form 2.7.2.1, after the first run through the inner loop the number of needed bits is always greater than the available bits , and therefor *common_scalefac* is always increased by the *quantizer_change.*

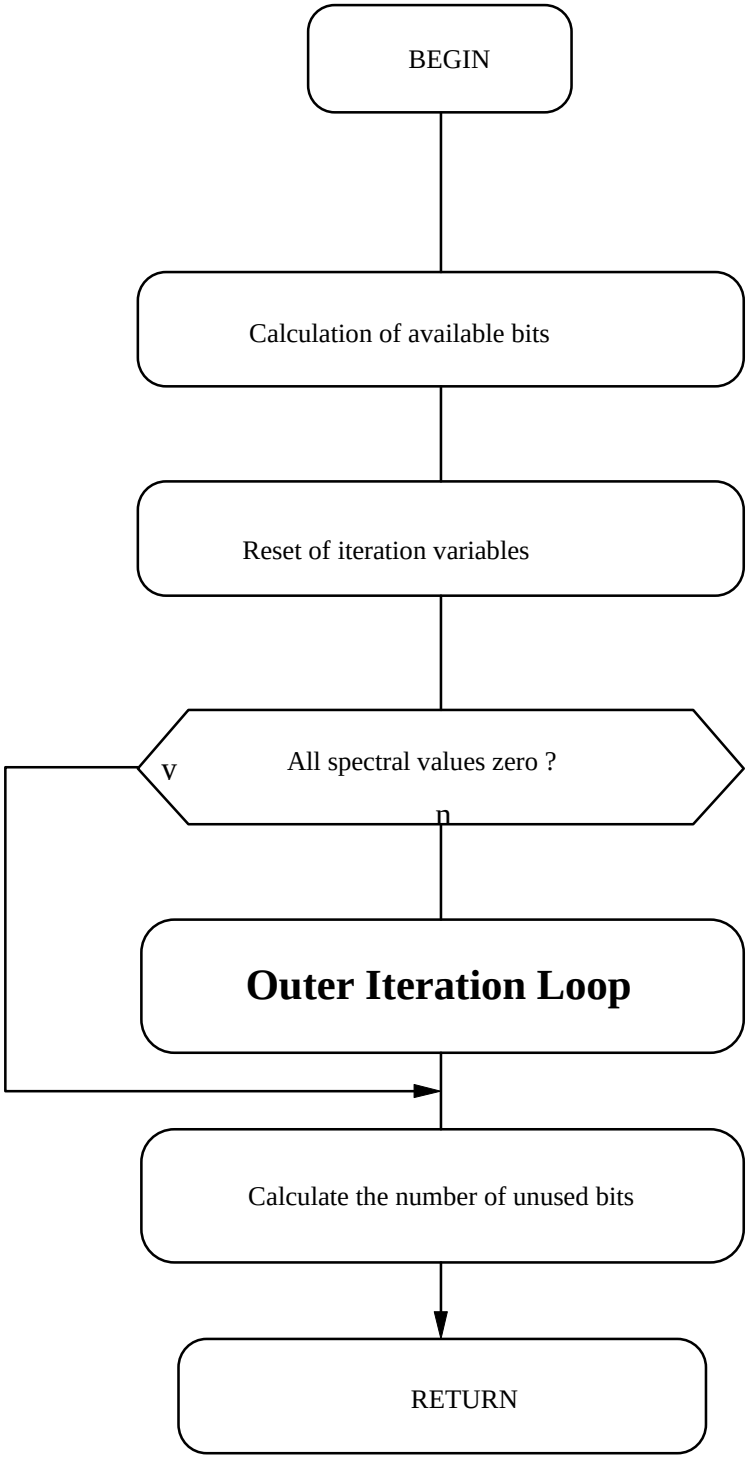


Figure B.2.7.1 -- AAC iteration loop

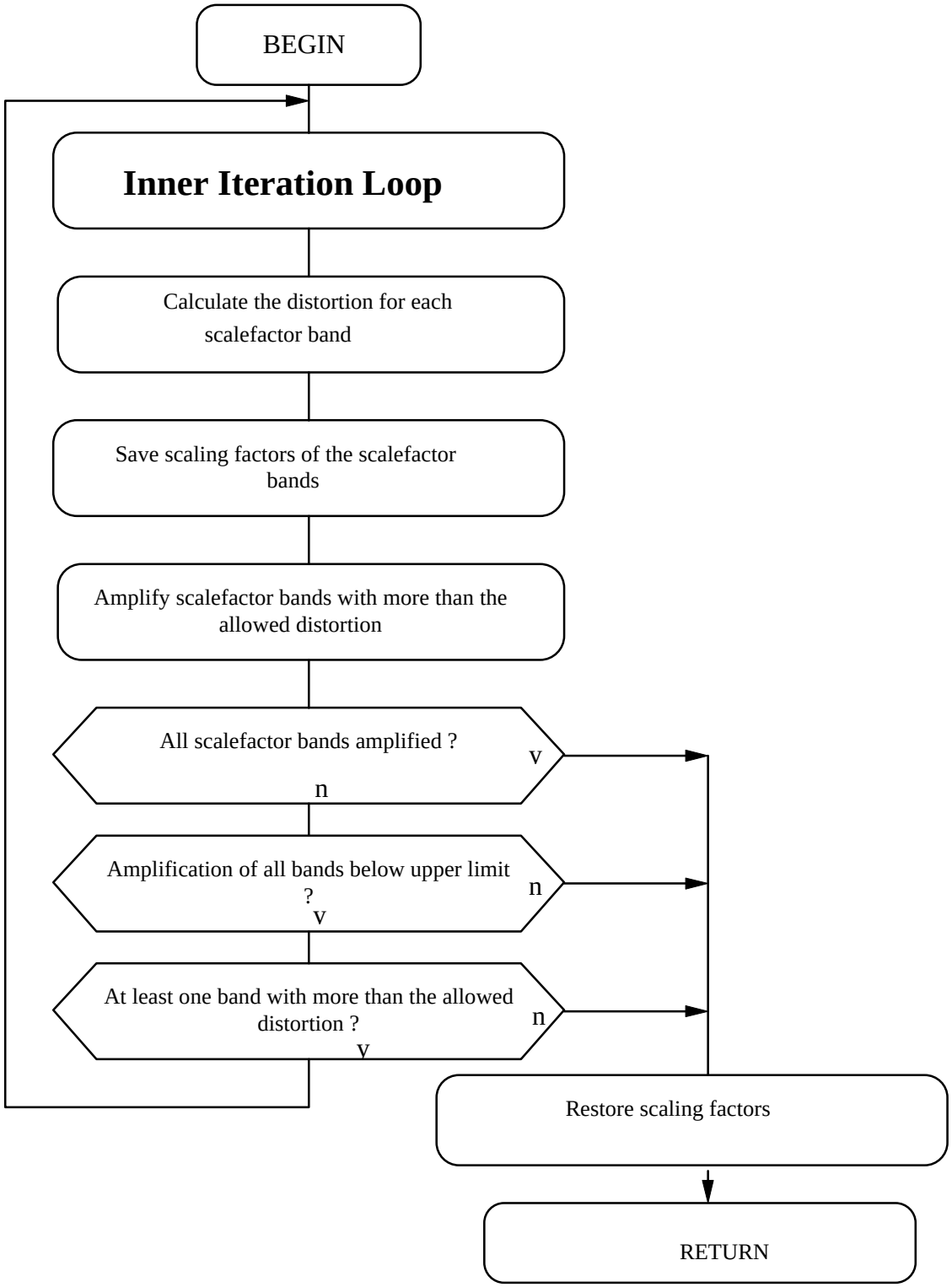


Figure B.2.7.2 -- AAC outer iteration loop

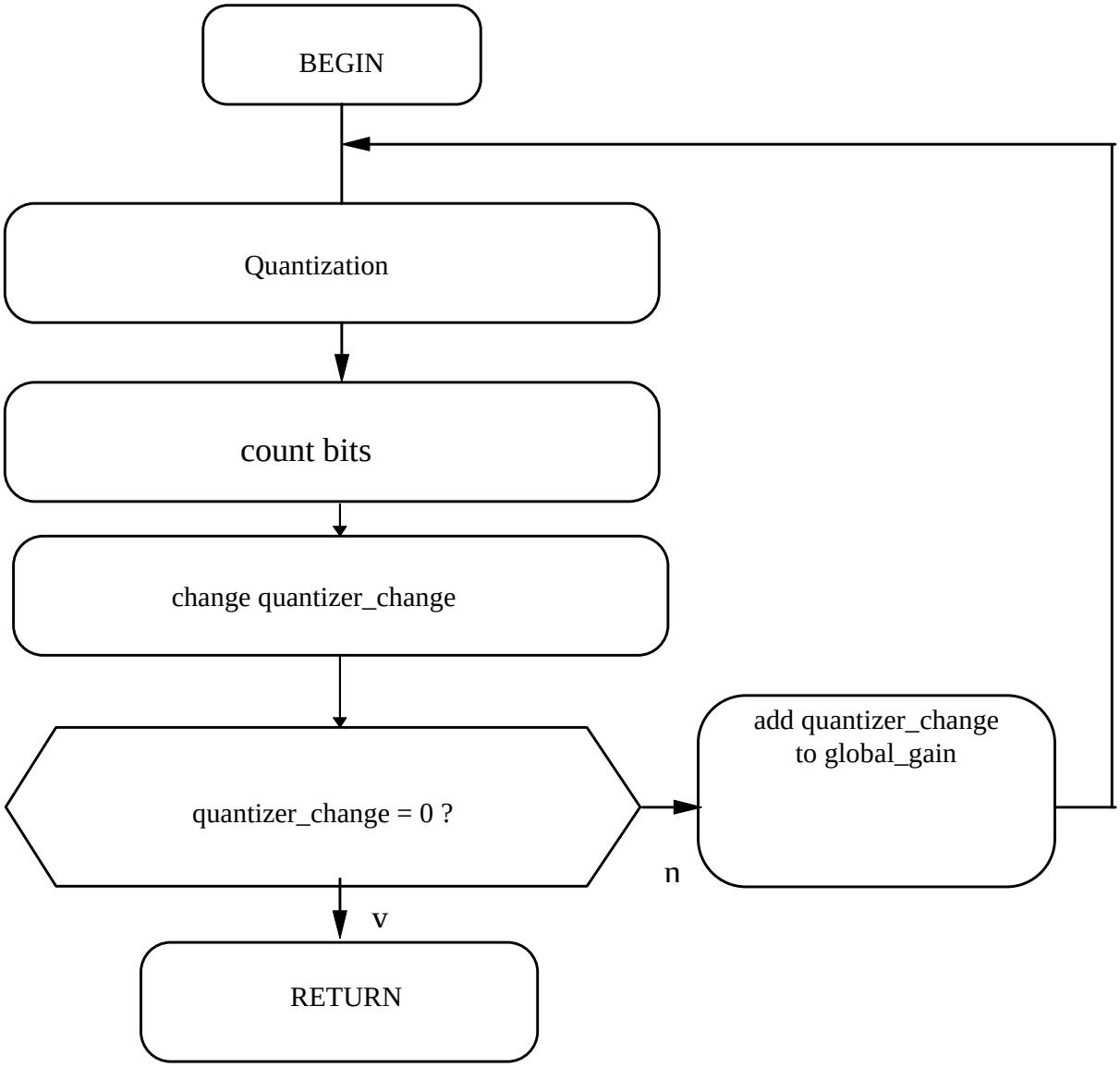


Figure B.2.7.3 -- AAC inner iteration loop

Noiseless Coding

Introduction

In the AAC encoder the input to the noiseless coding module is the set of 1024 quantized spectral coefficients. Since the noiseless coding is done inside the quantizer inner loop, it is part of an iterative process that converges when the total bit count (of which the noiseless coding is the vast majority) is within some interval surrounding the allocated bit count. This section will describe the encoding process for a single call to the noiseless coding module.

Noiseless coding is done via the following steps:

- Spectrum clipping
- Preliminary Huffman coding using maximum number of sections
- Section merging to achieve lowest bit count

Spectrum clipping

As a first step a method of noiseless dynamic range limiting may be applied to the spectrum. Up to four coefficients can be coded separately as magnitudes in excess of one, with a value of +-1 left in the quantized coefficient array to carry the sign. The index of the scalefactor band containing the lowest-frequency “clipped” coefficients is sent in the bitstream. Each of the “clipped” coefficients is coded as a magnitude (in excess of 1) and an offset from the base of the previously indicated scalefactor band. For this the long block scalefactor bands and coefficient ordering within those bands are used regardless of the window sequence. One strategy for applying spectrum clipping is to clip high-frequency coefficients whose absolute

amplitudes are larger than one. Since the side information for carrying the clipped coefficients costs some bits, this noiseless compression is applied only if it results in a net savings of bits.

Sectioning

The noiseless coding segments the set of 1024 quantized spectral coefficients into *sections*, such that a single Huffman codebook is used to code each section (the method of Huffman coding is explained in a later section). For reasons of coding efficiency, section boundaries can only be at scalefactor band boundaries so that for each section of the spectrum one must transmit the length of the section, in scalefactor bands, and the Huffman codebook number used for the section.

Sectioning is dynamic and typically varies from block to block, such that the number of bits needed to represent the full set of quantized spectral coefficients is minimized. This is done using a greedy merge algorithm starting with the maximum possible number of sections each of which uses the Huffman codebook with the smallest possible index. Sections are merged if the resulting merged section results in a lower total bit count, with merges that yield the greatest bit count reduction done first. If the sections to be merged do not use the same Huffman codebook then the codebook with the higher index must be used.

Sections often contain only coefficients whose value is zero. For example, if the audio input is band limited to 20 kHz or lower, then the highest coefficients are zero. Such sections are coded with Huffman codebook zero, which is an escape mechanism that indicates that all coefficients are zero and it does not require that any Huffman codewords be sent for that section.

Grouping and interleaving

If the window sequence is eight short windows then the set of 1024 coefficients is actually a matrix of 8 by 128 frequency coefficients representing the time-frequency evolution of the signal over the duration of the eight short windows. Although the sectioning mechanism is flexible enough to efficiently represent the 8 zero sections, *grouping* and *interleaving* provide for greater coding efficiency. As explained earlier, the coefficients associated with contiguous short windows can be grouped such that they share scalefactors amongst all scalefactor bands within the group. In addition, the coefficients within a group are interleaved by interchanging the order of scalefactor bands and windows. To be specific, assume that before interleaving the set of 1024 coefficients *c* are indexed as

c[g][w][b][k]

where

g is the index on groups
w is the index on windows within a group
b is the index on scalefactor bands within a window
k is the index on coefficients within a scalefactor band
and the right-most index varies most rapidly.

After interleaving the coefficients are indexed as

c[g][b][w][k]

This has the advantage of combining all zero sections due to band-limiting within each group.

Scalefactors

The coded spectrum uses one quantizer per scalefactor band. The step sizes of each of these quantizers is specified as a set of scalefactors and a global gain which normalizes these scalefactors. In order to increase compression, scalefactors associated with scalefactor bands that have only zero-valued coefficients are ignored in the coding process and therefore do not have to be transmitted. Both the global gain and scalefactors are quantized in 1.5 dB steps. The global gain is coded as an 8-bit unsigned integer and the scalefactors are differentially encoded relative to the previous scalefactor (or global gain for the first scalefactor) and then Huffman coded. The dynamic range of the global gain is sufficient to represent full-scale values from a 24-bit PCM audio source.

Huffman coding

Huffman coding is used to represent n-tuples of quantized coefficients, with the Huffman code drawn from one of 11 codebooks. The spectral coefficients within n-tuples are ordered (low to high) and the n-tuple size is two or four coefficients. The maximum absolute value of the quantized coefficients that can be represented by each Huffman codebook and the number of coefficients in each n-tuple for each codebook is shown in Table B.2.8.1. There are two codebooks for each maximum absolute value, with each representing a distinct probability distribution function. The best fit is always chosen. In order to

save on codebook storage (an important consideration in a mass-produced decoder), most codebooks represent unsigned values. For these codebooks the magnitude of the coefficients is Huffman coded and the sign bit of each non-zero coefficient is appended to the codeword.

Table B.2.8.1 -- Huffman Codebooks

Codebook index	n-Tuple size	Maximum absolute value	Signed values
0		0	
1	4	1	yes
2	4	1	yes
3	4	2	no
4	4	2	no
5	2	4	yes
6	2	4	yes
7	2	7	no
8	2	7	no
9	2	12	no
10	2	12	no
11	2	16 (ESC)	no

Two codebooks require special note: codebook 0 and codebook 11. As mentioned previously, codebook 0 indicates that all coefficients within a section are zero. Codebook 11 can represent quantized coefficients that have an absolute value greater than or equal to 16. If the magnitude of one or both coefficients is greater than or equal to 16, a special *escape coding* mechanism is used to represent those values. The magnitude of the coefficients is limited to no greater than 16 and the corresponding 2-tuple is Huffman coded. The sign bits, as needed, are appended to the codeword. For each coefficient magnitude greater or equal to 16, an *escape sequence* is also appended, as follows:

escape sequence = <escape_prefix><escape_separator><escape_word>

where
<escape_prefix> is a sequence of N binary “1’s”
<escape_separator> is a binary “0”
<escape_word> is an N+4 bit unsigned integer, msb first
and N is a count that is just large enough so that the magnitude of the quantized coefficient is equal to

2^(N+4) + <escape_word>

Perceptual Noise Substitution (PNS)

The encoding procedure for noise substitution is similar to the encoding procedure for intensity stereo and is performed as follows:

- For each scalefactor band containing spectral coefficients above a lower border frequency (e.g. 4 kHz) a noise detection is carried out. The scalefactor band is classified as noise-like if the corresponding signal is neither tonal nor contains strong changes in energy over time. The tonality of the signal can be estimated by using the tonality values calculated in the psychoacoustic model. Similarly, changes in signal energy can be evaluated using the FFT energies calculated in the psychoacoustic model.
- From the detection procedure, a map, noise_flag[sfb], is constructed such that noise-like scalefactor bands are flagged with a non-zero value.
- For each flagged scalefactor band the energy (sum of squares) of the corresponding spectral coefficients is calculated and mapped to a logarithmic representation with a resolution of 1.5 dB. An offset (NOISE_OFFSET=90) is added to the logarithmic noise energy values.
- For each flagged scalefactor band, the corresponding spectral coefficients are set to zero before quantization of the coefficients is carried out as usual.
- During the noiseless coding procedure, the pseudo-codebook NOISE_HCB is set for all flagged scalefactor bands. Apart from this, the regular section / noiseless coding procedure is carried out on the quantized coefficient data.

- The logarithmic noise energy values are coded analogous to the regular scalefactors, i.e. with a differential encoding scheme starting with the „global_gain value“. They are transmitted in place of the scalefactors belonging to the flagged scalefactor bands.

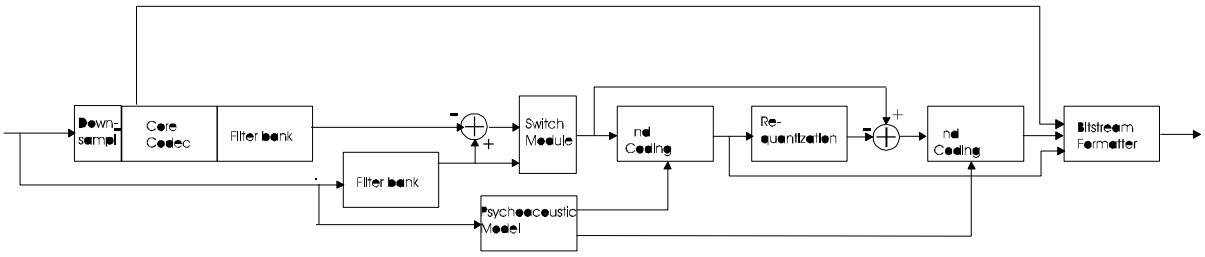
Scalable AAC with Core Coder

Mono or Stereo Encoder

The description provided below describes one possible way of achieving bit rate scalability. Bit-rate functionality can also be achieved using individual coding schemes as well. Here, a particular configuration is described where the T/F modules and a core coder is used. Since it is based on the calculation of a difference signal, the core coder must encode the waveform of the input signal. Currently the tool has been successfully tested based on the ITU-T G.729 codec. A slightly different implementation of the tool was used in the 1995 MPEG-4 test with the FS 1016 CELP coder and the „individual lines“ parametric coder of the MPEG-4 audio VM.

In addition to the core coder there is at least one enhancement layer based on the T/F based VM modules. In order to allow for integer frame lengths, depending on the core coder, alternative frame lengths for the AAC module may be used. The T/F modules provides, in addition to the standard 1024 length, also a 960 samples per frame implementation, which at 48 kHz sampling rate leads to 20 ms and at 32 kHz to a frame length of 30 ms. This allows for an easy integration of the MPEG-4 VM CELP core, and to construct bitstream frames which integrate standard speech coders, like G.729, which have a frame length of a multiple of 10 ms.

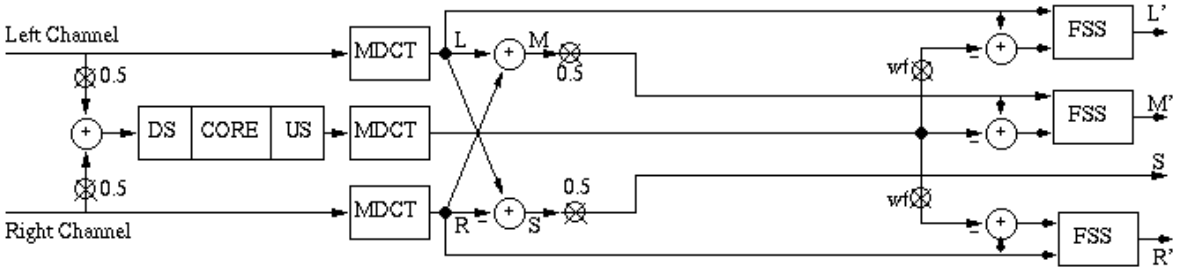
The bitrate of the additional layers can be any bit rate defined for the T/F based enhancement stages. The ratio of the sampling rate of the core coder and the sampling rate of the enhancement coder must be an integer.



Encoder structure

Mixed Mono/Stereo Coder

The block diagrams of the encoders of mode 1 and mode 2 are given in Figure 1 and 2. The block diagram of the third mode is identical to mode 2, with the core coder path



removed. Fig. 1: Core + stereo T/F

From the stereo input signal a mono signal is derived, which is then coded/decoded with a core coder, which - in general - operates at a lower sampling rate, up-sampled and transformed into the frequency-

domain. This part is identical to the corresponding part of the mono scalable coder. Next the standard M/S-Stereo matrix, as defined for the AAC coder is calculated on the spectra of the Left (L) and Right (R) signal, giving the Mid (M) and Side (S) signals. Three Frequency-Selective Switch modules (FSS) - identical to the switch modules of the mono scalable coder - then are used to generate the signals L', R' and M' from the L, R, and M signals. These signals and the original S signal then are used as the input signal to the standard AAC joint stereo coding modules. Both, M/S- and Intensity stereo coding is possible. The standard AAC ms_mask_present and ms_used[] flags are used to indicate M/S or L/R coding. However, since either M/S or L/R coding is used, it is not necessary to transmit all three FSS-related side-information. Instead, if M/S coding is selected for a band, only the FSS information for the M-switch needs to be transmitted in diff_control[0][win][sfb]. diff_control[1][win][sfb] is not transmitted in this case. If L/R coding is selected the FSS-side-information for the L and R channel needs to be transmitted in diff_control[0][win][sfb] for the left channel and in diff_control[1][win][sfb] for the right channel.

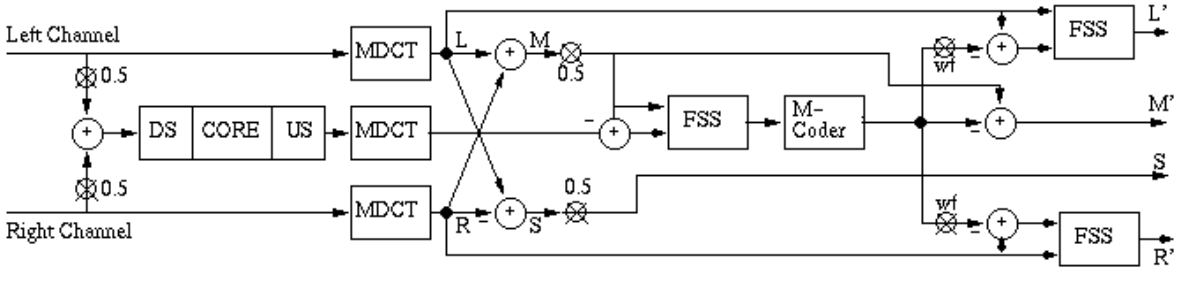


Fig. 2: Core + mono T/F + stereo T/F

This is an extended version of the Core + stereo T/F coder. An additional T/F - Coder is used to encode the output signal of the core-coder in the frequency domain. This makes the core plus M - T/F coder combination identical to the scalable mono core plus T/F coder. Two more FSS modules allow to select L'/R' to be identical to L, or R, respectively, or L - M*wf or R-M*wf. Again some of the switching conditions need not to be transmitted: If M/S coding is selected , only the FSS information for the M-switch needs to be transmitted in diff_control[0][win][sfb]. Diff_control[1][win][sfb] is not transmitted.. If L/R coding is selected the FSS-side-information for the L and R and the M channel needs to be transmitted in diff_control[0][win][sfb] for the left channel and in diff_control[1][win][sfb] for the right channel, and diff_control_m[win][0] for the M channel.

Bit-Sliced Arithmetic Coding (BSAC)

In BSAC, a quantized sequence is mapped into a bit-sliced sequence as shown in Figure bsac1.

MSB LSB

$B_{0,m}$	$B_{0,m-1}$	…	$B_{0,0}$	0^{th} quantized spectral data
$B_{1,m}$	$B_{1,m-1}$	…	$B_{1,0}$	
$B_{2,m}$	$B_{2,m-1}$	…	$B_{2,0}$	
…	…	…	…	
$B_{k,m}$	$B_{k,m-1}$	…	$B_{k,0}$	k^{th} quantized spectral data

where, allocated_bit is (m+1) bit

⇒

$vector_{0,m}$ … $vector_{k/4,m}$

$B_{0,m}$	$B_{1,m}$	$B_{2,m}$	$B_{3,m}$	…	$B_{k-3,m}$	$B_{k-2,m}$	$B_{k-1,m}$	$B_{k,m}$

$vector_{0,m-1}$ $vector_{k/4,m-1}$

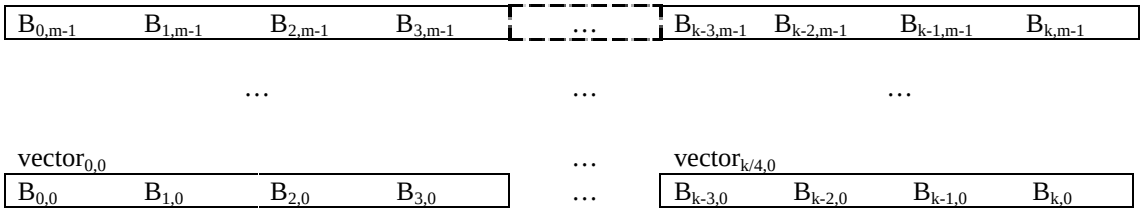


Figure bsac1. Bit-Sliced Representation

In order to maximize the match of the statistics of the bit-sliced sequences to that of the arithmetic model, 4-dimension vectors are formed from the bit-sliced sequence of the quantized spectrum and the 4-dimension vector is divided into two subvectors depending upon the previous state.
For example, consider a quantized sequence, x[n] as follows :

x[0] = 9, x[1]=0, x[2]=7 and x[3]=11 ...

If the allocated_bit is 5, 5 vectors are formed from a quantized sequence as shown in Figure bsac2.

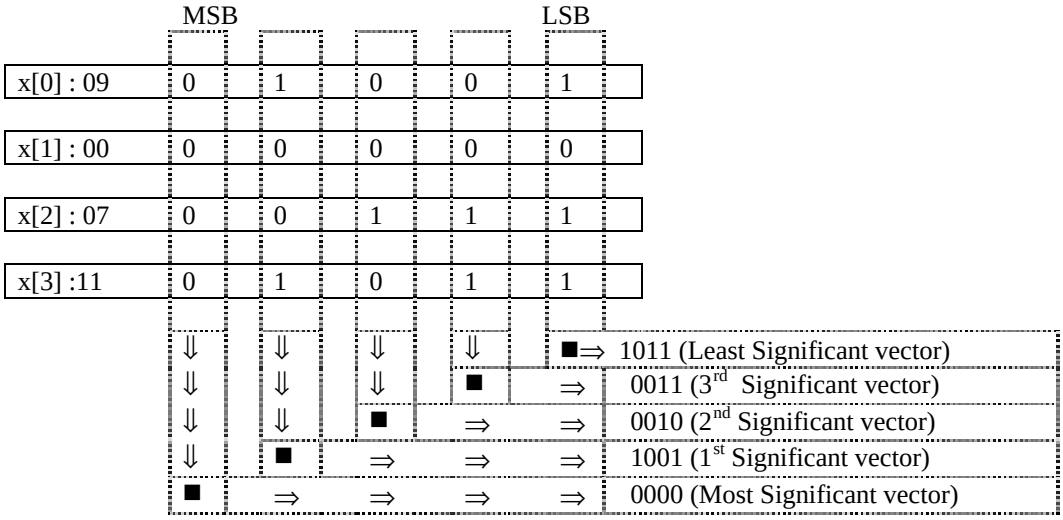
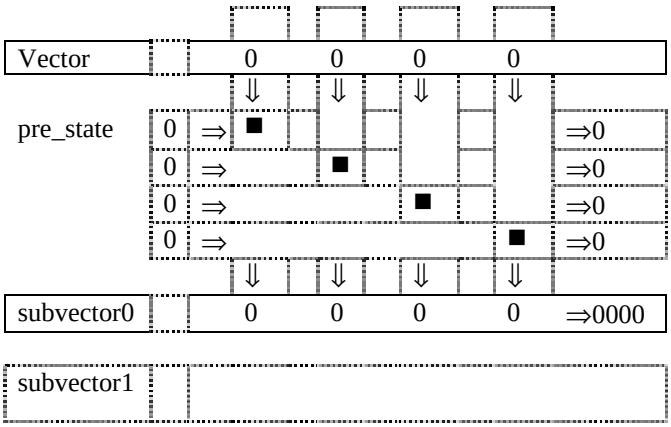
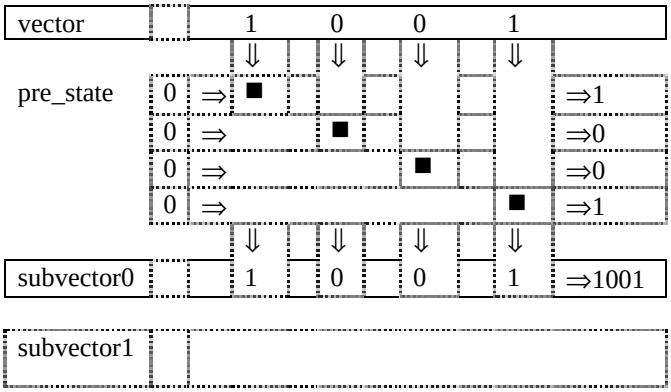


Figure bsac2 Example of Bit-Sliced Vector Representation

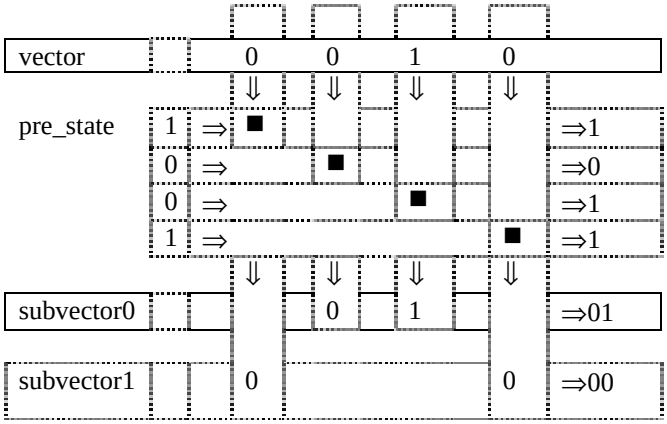
The previous states are updated along with coding the vectors from MSB to LSB. They are initialized to 0, are unchanged when bit value is zero and are set to 1 when bit value is non-zero as shown in Figure bsac3. The vectors are divided into two subvectors as shown in Figure bsac3. One is the subvector which is composed of bit-values whose previous state is 0, another is the subvector which is composed of bit-values whose previous state is 1.



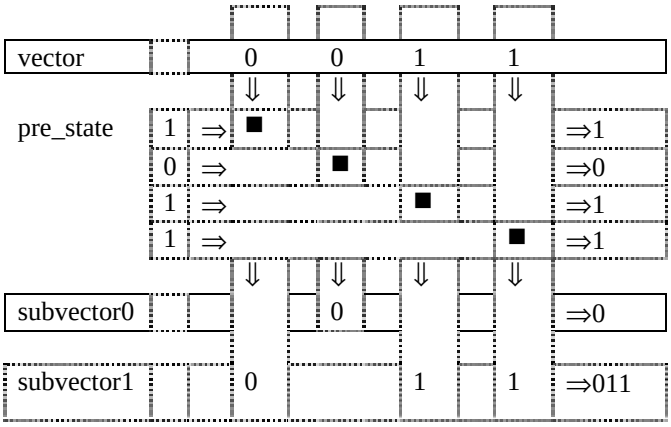
a) Most Significant vector and subvector



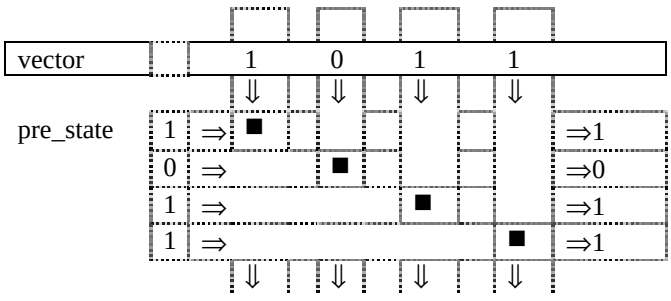
b) 1st Significant vector and subvector

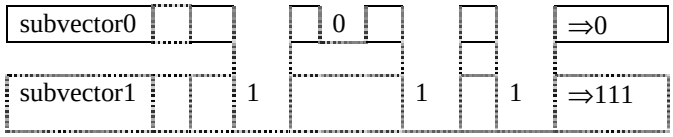


c) 2nd Significant vector and subvector



d) 3rd Significant vector and subvectors





e) Least Significant vector and subvectors

Figure bsac3 Examples of splitting vectors into subvectors

The arithmetic model relies on the allocated_bit of the coding band, the significance of the bit-sliced vector and the previous state.

Scaleable controller

This tool controlls the spectrum normalization and interleave vector quantization tools to construct a scalable coder. In each scaleable layer, spectrum normalization and interleaved vector quantization tool is cascaded. Each scalable layer has active frequency band which is shifted according to signal characteristics.

ANNEX B: Interface definitions for the VM software

Decoder functions

Quantization, Scalefactors, Noiseless Coding

```
void aac_decode_init();                          /* Initial AAC-decoder settings                          */

int aac_decode_frame(                            /* Decode bits to MDCT coefficients                      */
                BsBitStream *fixed_stream,       /*  Input --- bitstream                                  */
                double *spectral_line_vector[MAX_TIME_CHANNELS],  /*  Output spectrum*/
                int *block_type,                 /* Output --- block (window) type                        */
                Window_shape *window_shape);     /* Output --- window shape                               */
```

Interleaved Vector Quantization and Spectrum Normalization

```
void ntt_tf_decoder(                             /* time/frequency mapping decoder                        */
                ntt_INDEX  *indexp,              /* input --- index packet                                */
                double out[]);                   /* output --- time domain reconstructed signal           */

void ntt_tf_proc_spectrum_d(                     /* de-normalizer of flattened MDCT coefficients          */
                ntt_INDEX  *indexp,              /* input --- index packet                                */
                double flat_spectrum[],          /* input --- decoded flattened MDCT                      */
                double spectrum[]);              /* output --- denormalized MDCT                          */

void ntt_tf_freq2time(                           /*  frequency to time conversion IMDCT                   */
                double spectrum[],               /* input --- MDCT coefficient                            */
                int    w_type,                   /* input --- window type                                 */
                double audio_sample[]);          /* output --- time domain reconstructed signal           */

void ntt_dec_lpc_spectrum(                       /* LPC spectrum decoder                                  */
                int    index_lsp[ntt_N_SUP_MAX][ntt_LSP_NIDX_MAX],   /* in LSP index   */
                int    w_type,                   /* input --- window type                                 */
                double lpc_spec[]);              /* output --- decoded LPC spectrum (square root)         */

void ntt_dec_bark_env(                           /* Bark-scale envelope decoder                           */
                int    index_fw[],               /* input --- index of Bark-scale envelope VQ             */
                int    index_fw_alf[],           /* input --- index of prediction switch                  */
                int    w_type,                   /* input --- window type                                 */
                int    pf_switch,                /* input --- postfilter switch                           */
                double bark_env[]);              /* output --- reconstructed Bark-scale envelope          */

void ntt_dec_gain(                               /* global and subframegain decoder                       */
                int    index_pow[],              /* input --- index  of global and subframe power         */
                int    w_type,                   /* input --- window type                                 */
                double gain[]);                  /* output --- decoded power                              */
```

```
void ntt_denormalizer_spectrum(          /* denormalize MDCT coefficients                  */
                int    w_type,           /* Input --- window type                          */
                double flat_spectrum[],  /* Input --- reconstructed flattened MDCT         */
                double gain[],           /* Input --- reconstructed values of subframe gain*/
                double pit_seq[],        /* Input --- reconstructed pitch components       */
                double ggain[],          /* Input --- reconstructed value of global gain:  */
                double bark_env[],       /* Input --- reconstructed Bark-scale envelope    */
                double lpc_spec[],       /* Input --- reconstructed LPC envelope           */
                double spectrum[]);      /* Output --- denormalized MDCT coefficients      */

void ntt_post_process(                   /*  post process for MDCT coefficients            */
                int    index_pf,         /* Input --- switch for postfilter                */
                int    w_type,           /* Input --- window type                          */
                double spectrum[],       /* Input --- reconstructed MDCT coefficients      */
                double lpc_spectrum[],   /* Input --- reconstructed LPC spectrum           */
                double out_spectrum[]);  /* Output --- post processed MDCT coefficients    */


void ntt_dec_pitch(                      /*  decoder of pitch components                   */
                int    index_pit[],      /* Input --- index of pitch frequency             */
                int    index_pls[],      /* Input --- index of pitch components shape VQ   */
                int    index_pgain[],    /* Output --- index of pitch gain                 */
                int    w_type,           /* Output --- window type                         */
                double pit_seq[],        /* Output --- locally decoded pitch components    */
                double pgain[]);         /* Output --- locally decoded value of pitch gain */

void ntt_dec_pgain(                      /* decoder of pitch gain                          */
                int    index,            /* Input --- index of pitch gain                  */
                double *pgain);          /* Output --- reconstructed value of pitch gain   */

void ntt_dec_pit_seq(                    /* decoder of pitch component shape VQ            */
                int    index_pit[],      /* Input --- index of pitch frequency             */
                int    index_pls[],      /* Input --- index of pitch coponents shape       */
                double pit_seq[]);               /* Output --- reconstructed pitch components      */

void ntt_vec_lenp(                       /* Pitch vec. bits for each divided subvector     */
                int bits[],              /* output --- bits assigned for pitch subvector   */
                int length[]);           /* output --- dimension of  pitch subvector       */

void ntt_extend_pitch(                   /* extend pitch components onto MDCT coefficients */
                int    index_pit[],/* Input --- index of pitch frequency                  */
                double  pit_pack[],      /* Input --- paccked array of pitch components    */
                double  pit_seq[]);      /* Output --- extended array onto MDCT coefficients*/


void ntt_dec_lpc_spectrum_inv(           /* decoder of reciprocal LPC envelope             */
                int    index_lsp[ntt_N_SUP_MAX][ntt_LSP_NIDX_MAX], /* Input LSP index*/
                int    w_type,           /* Input --- window type                          */
                double inv_lpc_spec[]);  /* Output --- reconstructed reciprocal LPC env.   */

void ntt_TfInit(                         /* initializer T/F mapping coder                  */
                float  sampling_rate,    /* input --- sampling rate                        */
                float  bit_rate,         /* input --- bit rate                             */
                long   num_channel,      /* input --- number of channel                    */
                int    *frameNumSample,  /* input --- number of sample per frame           */
                int    *delayNumSample,  /* input --- number of delay sample          */
                int    *med_win_in_long, /* input --- number of medium subframe            */
                int    *short_win_in_long);/* input --- number of short subframe           */
```

```
int ntt_BitUnPack(                      /* unpack bitstream                                  */
                BsBitStream *stream,    /* input --- bitstream                               */
                int    available_bits,  /* input --- available bits                          */
                int    block_type,      /* input --- block (window) type                     */
                ntt_INDEX *index,       /* output ---- index packet                          */
                int    InitFlag);       /* input ---- init flag                              */

void ntt_win_sw_init(                   /* initializer for sin window                        */
                int    max_ch,          /* input ---maximan channel number                   */
                int    block_size_samples, /* input --- number of samples in block (window) */
                int    sampling_rate,   /* input --- sampling rate                           */
                int    bit_rate,        /* input --- bitrate                                 */
                int    short_win_in_long); /* input ---                                     */


void ntt_init();                        /* initial setting                                   */

void ntt_cp_mdtbl(                      /* read mode table                                   */
                ntt_MODE_TABLE ltbl);


void ntt_vec_len(                       /* get subvector lengtht (dimension) and bits        */
                int    bits[],          /* output --- assigned bits for two channel code     */
                int    length[]);       /* output --- dimension (length) of subvectors       */

void ntt_vec_lens(                      /* get subvector lengtht (dimension) and bits        */
                int    bits[],          /* output --- assigned bits for two channel code     */
                int    length[]);       /* output --- dimension (length) of subvectors       */

void ntt_vec_lenm(                      /* get subvector lengtht (dimension) and bits        */
                int    bits[],          /* output --- assigned bits for two channel code     */
                int    length[]);       /* output --- dimension (length) of subvectors       */


void ntt_vex_pns(                       /*  reconstructor from codebook (short frame)        */
                int    *index,          /* index --- index for MDCT coefficients             */
                double *sig);           /* index --- reconstructed flattened MDCT            */

void ntt_vex_pnm(                       /*  reconstructor from codebook (medium frame)       */
                int    *index,          /* index --- index for MDCT coefficients             */
                double *sig);           /* index --- reconstructed flattened MDCT            */

oid ntt_get_cdbk(                               /* read codebook file                        */
                char   *name,           /* input --- codebook name                           */
                double *codev_l,        /* output --- reconstruction code vector             */
                int    cb_size,         /* input --- codebook size (1<<numbits)              */
                int    cb_len,          /* input --- dimension of reconstruction vector      */
                int    cb_len_mx);      /* input --- maximam codebook dimension              */

void ntt_lsptowts(                      /* LSP to reciprocal LPC envelope (short frame)      */
                double lsp[],           /* input --- LSP parameters                          */
                double wt[]);           /*output --- reciprocal LPC envelope                 */

void ntt_fwdecl(                        /* decoder of Bark-scale envelope (long frame)       */
                int    index[],         /* input --- index for Bark-scale envelope           */
                int    ind_alf,         /* input --- index for prediction coeff in Bark-scale */
                int    pf_switch,       /* input --- switch for postfilter                   */
                int    i_sup,           /* input --- channel number                          */
                double pred[],          /* output --- reconstructed envelope                 */
                int    InitFlag);       /* input --- init flag                               */
```

```
void ntt_fwdecm(                        /* decoder of Bark-scale envelope (medium frame)   */
                int    index[],         /* input --- index for Bark-scale envelope         */
                int    ind_alf,         /* input --- index for prediction coeff in Bark-scale  */
                int    pf_switch,       /* input --- switch for postfilter                 */
                int    i_sup,           /* input --- channel number                        */
                double pred[],          /* output --- reconstructed envelope               */
                int    InitFlag);       /* input --- init flag                             */

void ntt_fwdecs(                        /* decoder of Bark-scale envelope (short frame)    */
                int    index[],         /* input --- index for Bark-scale envelope         */
                int    ind_alf,         /* input --- index for prediction coeff in Bark-scale  */
                int    i_sup,           /* input --- channel number                        */
                double pred[],          /* output --- reconstructed envelope               */
                int    InitFlag);       /* input --- init flag                             */

void ntt_fwexs(                         /* look up codebook        (short mode)            */
                int    index[],         /* input --- index for Bark-scale envelope         */
                double env[]);          /* outpur --- code vector output                   */

void ntt_fwexm(                         /* look up codebook        (medium frame )         */
                int    index[],         /* input --- index for Bark-scale envelope         */
                double env[]);          /* outpur --- code vector output                   */

void ntt_fwex(                          /* look up codebook        (long frame)            */
                int    index[],         /* input --- index for Bark-scale envelope         */
                double env[]);          /* outpur --- code vector output                   */

void ntt_cnst_chk(                      /* check constant if it is beyond ranges           */
                int    calcv,           /* input --- constant valure                  */
                int    defv,            /* input --- predetermined value                   */
                char   *defname);       /* input --- constant name                         */


void ntt_lsptowd(                       /* LSP to LPC envelope with interpolation (long)   */
                double lsp[],           /* input --- reconstruction LSP parameter          */
                double wt[]);           /* output --- LPC spectral envelope                */

void ntt_lsptowmd(                      /* LSP to LPC envelope with interpolation (medium)*/
                double lsp[],           /* input --- reconstruction LSP parameter          */
                double wt[]);           /* output --- LPC spectral envelope                */

void ntt_adjust_bits(                   /* assign bits                                     */
                int    available_bits,  /* input --- avalable bits                         */
                int    w_type);         /* input --- window type                           */

void ntt_set_interleave(                /* set interleave matrix                           */
                enum   ntt_INTERLEAVE_SET_MODE set_mode);

void ntt_dec_pit_seq(                   /* decoder of pitch components sequence            */
                int    index_pit[],     /* input --- index for pitch frequency             */
                int    index_pls[],     /* input --- index for pitch components shape vq   */
                double pit_seq[])       /* output --- reconstructed pitch components       */

void ntt_get_code(                      /* read LSP structured codebook                    */
                char *fname,            /* input --- codebook  name for LSP                */
                int nstage,             /* input --- number of stage                       */
                int csize[],            /* input --- codebook size for each stage          */
                int cdim[],             /* input --- dimension boundary of code vector     */
                double code[][ntt_N_PR_MAX],    /*output --- codebook array                */
```

```
                double fgcode[ntt_N_MODE_MAX][ntt_MA_NP][ntt_N_PR_MAX]) /*pred   */

void ntt_redec(                         /*  decoder of LSP                                  */
                int ifr_fi,             /* input --- frame number                           */
                int index[],            /* input --- index for LSP                          */
                int nstg1,              /* input --- number of stage                        */
                int csize1[],           /* input --- codebook size for LSP                  */
                int cdim1[],            /* input --- dimension boundary of code vector      */
                double code[][ntt_N_PR_MAX],    /* input --- codebook                       */
                double fgcode[ntt_N_MODE_MAX][ntt_MA_NP][ntt_N_PR_MAX], /*pred   */
                double fg_sum[ntt_N_PR_MAX],  /* input --- contribution from previous       */
                double pred_vec[ntt_N_PR_MAX],          /* input --- previous contribution  */
                double lspq[],          /* output --- reconstructed LSP                     */
                double out_vec[])               /* output --- partially reconstructed LSP for next fr.   */

void ntt_relspwed(                      /* intermediate LSP quantizer                       */
                double lsp[ntt_N_PR_MAX+1],     /* input --- original LSP                   */
                double wegt[ntt_N_PR_MAX+1],  /* input --- weight vector                    */
                double lspq[ntt_N_PR_MAX+1],   /* output --- reconstructed LSP vector       */
                int    ifr_fi,          /* input --- frame number                           */
                int    nstage,          /* input --- number of stage                        */
                int    csize[],         /* input --- codebook size                          */
                int    cdim[],          /* input --- code vector dimension                  */
                double code[][ntt_N_PR_MAX],    /* input --- codebook                       */
                double fg_sum[ntt_N_MODE_MAX][ntt_N_PR_MAX],       /* input pred            */
                double amp_cur,         /* input  --- current frame power                   */
                double cmpe[ntt_N_MODE_MAX][ntt_N_PR_MAX],         /* input buffer          */
                double target_buf[ntt_N_MODE_MAX][ntt_N_PR_MAX],   /* input buffer          */
                double out_vec[ntt_N_PR_MAX], /*output --- reconstructed LSP vector        */
                int    *vqword,         /* output --- packed index for LSP                  */
                int    index_lsp[])     /* output --- index for LSP                         */

void ntt_setd(                          /*  set value                                       */
                int n, double c,        /* input --- scalar value                           */
                double xx[])            /* output --- output data array                     */

void ntt_set_isp(                       /* set baundary for LSP code vector                 */
                int nsp)                /* input --- number of split for LSP code vector    */

void ntt_vq_coder(                      /* t/f encoder based on NTT_VQ mode                 */
                double in[],            /* input --- time domain input signal               */
                double     *spectral_line_vector[MAX_TIME_CHANNELS],          /* input: spectrum*/
                double     external_pw[],  /* input --- external perceptual model           */
                int        pw_select,   /* input --- perceptual model select                */
                int        block_type,  /* input --- block (window) type                    */
                ntt_INDEX   *index,     /* input --- index packet                           */
                ntt_PARAM   *param_ntt,/* input --- parameter packet                        */
                int        available_bits)  /* input --- available bits per frmae           */

void ntt_vq_decoder(                    /*  t/f decoder based on NTT_VQ                     */
                ntt_INDEX  *indexp,     /* input --- index packet                           */
                double *spectral_line_vector[MAX_TIME_CHANNELS])    /* out                  */

void ntt_zerod(                         /* clear array                                      */
                int n,                  /* input --- number of data                 */
                double xx[])            /* in/out --- array                                 */
```

Filterbank Tool

```
void freq2buffer(                       /* Frequency to time conversion IMDCT                */
                double p_in_data[],     /* Input ---- MDCT coefficients                      */
                double p_out_data[],    /* Output --- time domain reconstructed signal       */
                double p_overlap[],     /* Input/Output --- overlap-buffer                   */
                enum WINDOW_TYPE block_type,        /* Input --- window type             */
                int nlong,              /* Input --- shift length for long windows           */
                int nmed,               /* Input --- shift length for medium windows         */
                int nshort,             /* Input --- shift length for short windows          */
                Window_shape wfun_select,       /* Input --- window shape                */
                Imdct_out overlap_select);/* IMDCT output, non-/overlapped, for gain-control */

void imdct(                             /* IMDCT                                             */
                double in_data[],       /* Input --- MDCT Coefficients                       */
                double out_data[],      /* Output --- Time domain reconstructed signal       */
                int len);               /* Input --- Length of the MDCT coefficients-vector  */

void calc_window(                       /* Calculate the window shape                        */
                double window[],        /* Output --- window shape values                    */
                int len,                /* Input --- length of window: long, medium, short   */
                Window_shape wfun_select);      /* Input --- selected window shape       */
```

Noiseless Coding for BSAC

```
void bsac_decode_init();                /* Initial BSAC-decoder settings                     */

int bsac_decode_frame(                  /* Decode bits to MDCT coefficients                  */
                int target_layer                /*  Input – target layer to be decoded */
                BsBitStream *fixed_stream,      /*  Input --- bitstream                   */
                double *spectral_line_vector,   /*  Output spectrum*/
                int *block_type,                /* Output --- block (window) type         */
                Window_shape *window_shape);    /* Output --- window shape                */
```

Encoder functions

Interleaved Vector Quantization and Spectrum Normalization

```
void ntt_tf_coder(                      /* time/frequency mapping coder                      */
                double sig[],           /* input --- time domain signal                      */
                ntt_INDEX  *indexp,     /* output --- index packet                           */
                int    iframe);         /* input --- frame number                            */

void ntt_tf_requantize_spectrum(        /* quantizer for flattened MDCT coefficients         */
                ntt_INDEX *indexp,      /* output --- index packet                           */
                double flat_spectrum[]);/* input --- flattened MDCT coefficients             */

void ntt_enc_lpc_spectrum(              /* quantizer of LPC coefficients                     */
                double sig[],           /* Input --- input time domain signal                */
                int    w_type,          /* Input --- window type                             */
                int    index_lsp[ntt_N_SUP_MAX][ntt_LSP_NIDX_MAX], /* Out:   LSP index*/
                double lpc_spectrum[]); /* Output ---reconstructed reciprocal LPC envelope   */

void ntt_tf_pre_process(                /*  preprocess  before transform                     */
                double sig[],           /* Input --- input time domain signal                */
                ntt_INDEX     *indexp,  /* Output --- index packet                           */
                ntt_PARAM *param_ntt,   /* Output --- parameter packet                       */
```

```
                int   med_sw,           /* input  ---                                   */
                int    InitFlag);       /* input  --- init flag                         */

void ntt_tf_time2freq(                  /* windowing and MDCT                           */
                double sig[],           /* Input --- time domain signal                 */
                int    w_type,          /* Input --- window type                        */
                double spectrum[]);     /* Output --- MDCT coefficients                 */

void ntt_tf_proc_spectrum(              /* Spectral normalization                       */
                double sig[],           /* Input --- time domain signal:                */
                double spectrum[],      /* Input --- MDCT coefficients                  */
                ntt_INDEX  *indexp,     /* In/Out --- index packet                      */
                double lpc_spectrum[],  /* Output --- reconstructed reciprocal LPC spectrum */
                double bark_env[],      /* Output --- reconstructed Bark-scale envelope */
                double pitch_component[],       /* Output --- reconstructed pitch components*/
                double gain[],          /* Output --- reconstructed value of global gain */
                ntt_PARAM param_ntt);   /* In/Out: --- parameter packet                 */

void ntt_freq_domain_pre_process(       /*  preprocess on MDCT                          */
                double spectrum[],      /* Input --- MDCT coefficient                   */
                double lpc_spectrum[],  /* Input --- LPC spectral envelope              */
                ntt_PARAM param_ntt,    /* Input --- prameter packet                    */
                int    w_type,          /* Input --- window type                        */
                double spectrum_out[],  /* Output: --- process ed MDCT coefficients                 */
                double lpc_spectrum_out[]);     /* Outpu --- processed LPC envelope:    */


void ntt_tf_quantize_spectrum(          /* WVQ of normalized MDCT subvectors            */
                double spectrum[],      /* Input --- MDCT coefficient:                  */
                double lpc_spectrum[],  /* Input --- reciprocal LPC spectrum            */
                double bark_env[],      /* Input --- Bark-scale envelope                */
                double pitch_component[],       /* Input -- reconstructed pitch components  */
                double gain[],          /* Input ---  reconstructed value of global gain */
                double perceptual_weight[],     /* input --- weight for distortion measure  */
                ntt_INDEX  *indexp,     /* In/Out --- index packet                      */
                ntt_PARAM *param_ntt);/* In/Out --- parameter packet                    */

void ntt_tf_perceptual_model(           /*  perceptual weight generator                 */
                double lpc_spectrum[],  /* Input --- reciprocal LPC envelope            */
                double bark_env[],      /* Input --- Bark-scal envelope                 */
                double gain[],          /* Input --- reconstructed value of global gain */
                int      w_type,        /* Input --- window type                        */
                double spectrum[],      /* Input --- MDCT coefficients:                 */
                double pitch_sequence[],/* Input --- pitch components                   */
                double perceptual_weight[],     /* Output --- weight for distortion measure  */
                ntt_PARAM *param_ntt);/* In/Out --- parameter packet:                   */

int ntt_BitPack(                        /* pack bits for bitstream                      */
                ntt_INDEX *index,       /* input --- index packet                       */
                BsBitStream *stream,    /* output ---bitstream                          */
                int    InitFlag);       /* input --- init flag                          */
```

to be updated from Tampere document N1298

Filterbank Tool

ANNEX C: MSDL Bit Stream Description

```
class adif_header {
        bit(32) adif_id;
        bit(1) copyright_id_present;
        if ( copyright_id_present ) bit(72) copyright_id;
        bit(1) original_copy;
        uint(1) home;
        bit(1) bitstream_type;
        uint(23) bitrate;
        uint(4) num_program_config_elements;
        repeat(num_program_config_elements + 1) {
                if (bitstream_type == 0) uint(20) adif_buffer_fullness;
                program_config_element ProgramConfigElement;
        }
};

aligned(8) class adts0_sequence {
        while (SYNCWORD) adts0_frame Adts0Frame;
};

aligned(8) class adts0_frame {
        adts0_fixed_header Adts0FixedHeader;
        adts0_variable_header Adts0VariableHeader;
        if (protection_bit==0) uint(16) crc_check;
        raw_data_block RawDataBlock[Adts0Header.number_of_raw_data_blocks_in_frame +1];
};

aligned(8) class adts0_fixed_header : bit(12)=SYNCWORD {
        bit(1) ID;
        uint(2) layer;
        bit(1) protection_absent;
        vlc(sampling_frequency_table) sampling_frequency;
        bit(1) private_bit;
        vlc(channel_configuration_table) channel_configuration;
        bit(1) original_copy;
        uint(1) home;
        vlc(emphasis_table) emphasis;
};

aligned(8) class adts0_variable_header {
        bit(1) copyright_id;
        bit(1) copyright_id_start;
        uint(13) frame_length;
        bit(12) end_of_raw_data;
        bit(8) adts0_buffer_fullness;
        uint(2) number_of_raw_data_blocks_in_frame;
};

map program_config_elements_table () {};
```

```
class raw_data_block {
        repeat { syntactic_element SyntacticElement; } until ( [ID_END] );
};
class single_channel_element is syntactic_element :  bit(3) ID_SCE = 0  {
        bit(4) element_instance_tag;
        individual_channel_stream(false) IndividualChannelStream;
};
class  channel_pair_element  is syntactic_element:  bit(3) ID_CPE = 1  {
        int ws=-1;
        bit(4)element_instance_tag;
        bit(1) common_window;
        if( common_window ) {
                ics_info ICSInfo;
                bit(2) ms_mask_present;
                if ( ms_mask_present == 1 ) bit(1) ms_used[max_sfb];
                ws = ICSInfo.window_sequence;
        }
        individual_channel_stream(common_window, ws) IndividualChannelStream0;
        individual_channel_stream(common_window, ws) IndividualChannelStream1;
};
class ics_info {
        uint(3) window_sequence;
        bit(1) window_shape;
        if ( window_sequence == EIGHT_SHORT_SEQUENCE ) {
                uint(4) max_sfb  ;
                bit(7) ScalefactorGrouping;
        } else {
                uint(6) max_sfb;
                bit(1) predictor_data_present;
                if ( predictor_data_present ) {
                        bit(1) predictor_reset;
                        if ( predictor_reset ) uint(5) predictor_reset_index;
                }
                bit(1) prediction_used[ min(max_sfb,PRED_SFB_MAX) ];
        }
};
class  individual_channel_stream(  bit common_window,
                                        int common_window_sequence ) {
        int window_sequence = common_window_sequence;
        int(8) global_gain;
        if(  !common_window ) {
                ics_info ICSInfo;
                window_sequence = ICSInfo.window_sequence;
                int sfb_cb[number_of_coderbands[ICSInfo.window_sequence] - 1];
                        // this defines an array that is not taken from the
                        // bitstream
                rle(sfb_cb, number_of_coderbands[ICSInfo.window_sequence] - 1);
                        // this function fills the sfb_cb table
                        // from what is read in the bitstream
        } else {
                int sfb_cb[number_of_coderbands[common_window_sequence] - 1];
                rle(sfb_cb, number_of_coderbands[common_window_sequence] - 1);
        }
```

```
        bit (1) pulse_data_presentflag;
        if( pulse_data_presentflag ) pulse_data PulseData;
        bit (1) tns_data_presentflag;
        if( tns_data_presentflag ) tns_data(window_sequence) TNSData;
        bit (1) gain_control_data_presentflag;
        if( gain_control_data_presentflag ) gain_control_data GainControlData;
        scalefactor_data( sfb_cb, number_of_coderbands[window_sequence] - 1, global_gain )
ScaleFactorData;
        spectral_data SpectralData;
};
class scalefactor_data( int sfb_cb[], int sfb_cb_length, int global_gain ) {
        repeat (nonzero(sfb_cb, sfb_cb_length )) {
                vlc(hcod_sf_vlc) hcod_sf;
                                // same as before, reordering is not done in MSDL-S
                                // so we only pull the requested number of values from
                                // the bitstream instead of trying to put them in their
                                // definitive place.
        }
};
map nfilt_bits_for_window_type ( window ) {
        // defined in TNS tool description
}
map  tns_length_bits_for_window_type ( window ) {
        // defined in TNS tool description
}
map  tns_order_bits_for_window_type ( window ) {
        // defined in TNS tool description
}

class tns_data( int window_sequence ) {
        int w, filt, coef_bits;
        for ( w=0 ; w < num_windows[window_sequence]; w++ ) {
                uint(nfilt_bits_for_window_type[w]) n_filt;
                if ( n_filt != 0 )bit (1) coef_res;
                if ( coef_res[w] == 1) coef_bits = 4;
                else coef_bits = 3;
                repeat ( n_filt ) {
                        uint(tns_length_bits_for_window_type[w]) length;
                        uint(tns_order_bits_for_window_type[w]) order;
                        if ( order != 0 ) {
                                bit ( 1 ) direction;
                                bit ( 1 ) coef_compress;
                                uint ( coef_bits  ) coef[order];
                        }
                }
        }
};
class spectral_data {
        repeat (i: max_sd[window_sequence]) {
                repeat (sd_nelems[window_sequence][sfb_cb[i]]) {
                        vlc(hcod_vlc[sfb_cb[i]]) sval;
                                // same as before, no reordering
                }
        }
};

class pulse_data  {
        uint(2) number_pulse;
        uint(6) pulse_start_sfb;
        repeat( numberpulse+1 ){
```

```
                uint(5) pulse_offset;
                uint(4) pulse_amp;
        }
};
class coupling_channel_element is syntactic_element: bit(3) ID_CCE = 2 {
        uint(4) element_instance_tag;
        uint(3) number_of_coupled_elements;
        int num_gain_element_lists = number_of_coupled_elements+1;
        repeat ( number_of_coupled_elements+1 ) {
                uint(5) cc_target;
                if( bitfield(cc_target, 4)) {
                        bit(1) cc_l;
                        bit(1) cc_r;
                        if ( cc_l && cc_r ) num_gain_element_lists += 1;
                }
        }
        bit(1) cc_domain;
        bit(1) gain_element_sign;
        uint(2) gain_element_scale;
        individual_channel_stream(0) channelStream;
        repeat(num_gain_element_lists) {
                bit (1) common_gain_element_present;
                if ( common_gain_element_present ) vlc( hcod_sf_vlc ) hcod_sf;
                else {
                        repeat (nonzero(sfb_cb, sfb_cb_length )) {
                                vlc(hcod_sf_vlc) hcod_sf;
                                        // same as before, reordering is not done in MSDL-S
                                        // so we only pull the requested number of values from
                                        // the bitstream instead of trying to put them in their
                                        // definitive place.
                        }
                }
        }
};
class lfe_channel_element is syntactic_element:bit(3) ID_FXE= 3 {
        uint(4) element_instance_tag;
        individual_channel_stream(0) ChannelStream;
};
class data_stream_element is syntactic_element :bit(3) ID_DSE = 4 {
        bit(4) element_instance_tag;
        uint(4) cnt;
        if ( cnt == 15 ) {
                uint(12) esc_l_cnt;
                cnt += esc_l_cnt;
        }
        repeat ( cnt )uint(8) data_stream_byte[element_instance_tag];
};
class program_config_element : bit(3) ID_PCE = 5 {
        bit(4)element_instance_tag;
        bit(2) profile;
        uint(4) num_front_channel_elements;
        uint(4) num_side_channel_elements;
        uint(4) num_back_channel_elements;
        uint(2) num_lfe_channel_elements;
        uint(3) num_assoc_data_elements;
        uint(4) num_valid_cce_elements;
        bit mono_mixdown_present;
        if ( mono_mixdown_present ) uint(4) mono_mixdown_element_number;
        bit stereo_mixdown_present;
        if( stereo_mixdown_present ) uint(4) stereo_mixdown_element_number;
```

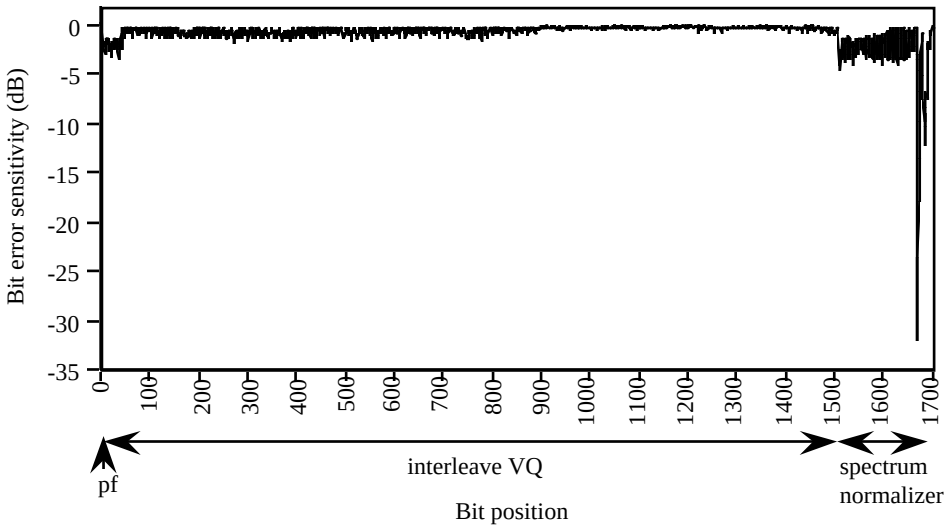
```
        uint(5) front_element_list[num_front_channel_elements];
        uint(5) side_element_list[num_side_channel_elements ];
        uint(5) back_element_list[num_back_channel_elements ];
        uint(4) lfe_element_list[num_lfe_channel_elements ];
        uint(4) assoc_data_element_list[num_assoc_data_elements ];
        uint(4) cce_element_list[num_valid_cce_elements ];
        uint(8) comment_field_bytes;
        uint(8) comment_field_data[comment_field_bytes ];
};
class fill_element is syntactic_element: bit(3) ID_FIL= 6 {
        uint(4) cnt;
        if ( cnt == 15 ) {
                uint(8) esc_cnt;
                cnt += esc_cnt;
        }
        uint(8) fill_byte[cnt];
};
class gain_control_data {
                // nested repeats achieve an arrangement of the data with the inner
                // repeat doing the same as the last index of a C table
                // so in this element, we describe three arrays, adjust_num, alevcode, and aloccode
                // all repeated access fill subsequent elements of the table
                // the inner loop being equivalent to the last index of a C table
        int wd;
        uint(2) maxBand;
        if( window_sequence == ONLY_LONG_SEQUENCE ) {
                repeat( maxBand ){
                        uint(3) adjust_num;
                        repeat( adjust_num ) {
                                uint(4) alevcode;
                                uint(5) aloccode;
                        }}
        } else if ( window_sequence == LONG_START_SEQUENCE ) {
                repeat( maxBand ){
                        for(wd=0;wd<2;wd++) {
                                uint(3) adjust_num;
                                repeat( adjust_num ) {
                                        uint(4) alevcode;
                                        if(wd==0) uint(4) aloccode;
                                        elseuint(2) aloccode;
                                }}}
        } else if ( window_sequence == EIGHT_SHORT_SEQUENCE ) {
                repeat( maxBand ){
                        repeat(8) {
                                uint(3) adjust_num;
                                repeat( adjust_num ) {
                                        uint(4) alevcode;
                                        uint(2) aloccode;
                                }}}
        } else if ( window_sequence == LONG_STOP_SEQUENCE ) {
                repeat( maxBand ){
                        for(wd=0;wd<2;wd++) {
                                uint(3) adjust_num;
                                repeat( adjust_num ) {
                                        uint(4) alevcode;
                                        if(wd==0) uint(4) aloccode;
                                        else uint(5) aloccode;
                                }}}};
```

ANNEX D: Error resilience

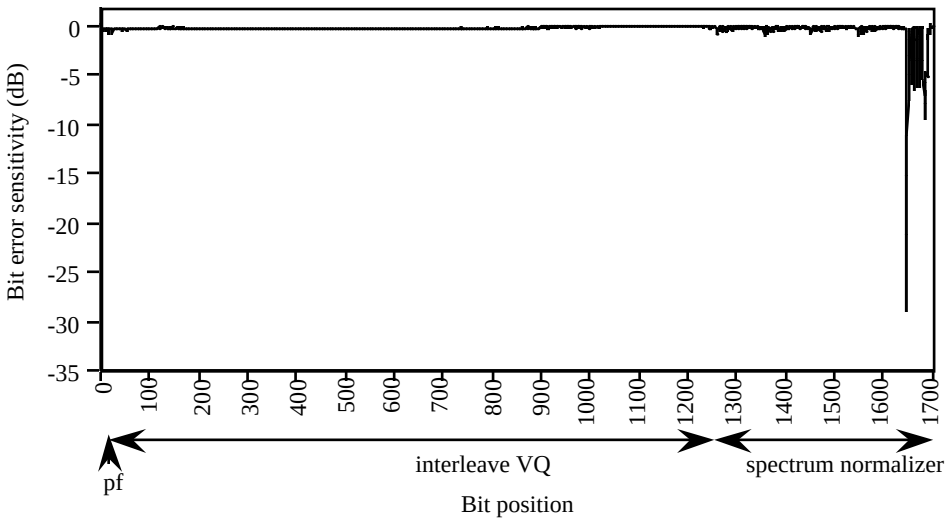
Error resilience for TwinVQ mode

Bit Error Sensitivity

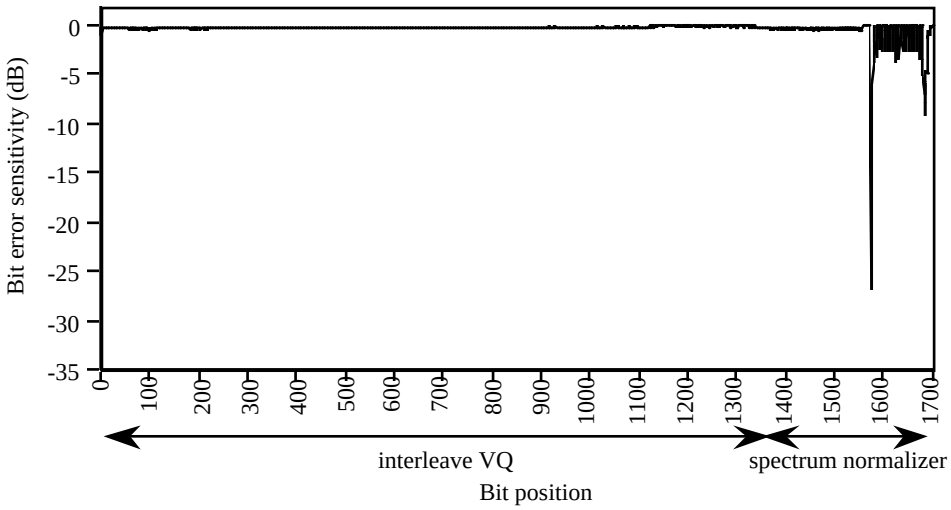
Bit error sensitivity of NTT's T/F coder bits (without window-type bits which are very sensitive to bit error) is shown in Fig.8.1. According to the analysis of error sensitivity, we have found that majority of bits are insensitive to bit error. It is, therefore, feasible to use unequal error protection scheme or Bit Selective Forward Error Control (BS-FEC) scheme to make robust against channel errors by adding small amount of redundancy bits.



(a) LONG



(b) MEDIUM



(c)SHORT

Fig. 8.1. Bit error sensitivity of NTT_VQ mode T/F coder (40 kbit/s).

Suggested Basic Structure of Error Protection

Suggested basic structure of error protection scheme is shown in Fig. 8.2. This is based on he following five major error protection tools to support unequal error protection scheme or BS-FEC (Bit Selective Forward Error Protection) for the bit stream of NTT_VQ mode according to the observation of bit error sensitivities. Tools(1)-(4) are error-detection and error-correction codec for T/F coder's output bitstream; Tool(5) is an error-concealment tool for T/F decoder. These tools can efficiently protect NTT_VQ mode T/F codec even for bad channel conditions.

(1) Error protection tool for window type
(Error correction by block code)
(2) Error protection tool for spectrum normalizer
(Error detection by CRC, correction by convolutional code)
(3) Error protection tool for interleave vector quantizer
(Error detection by parity)
(4) Bit interleave tool
(5) Error protection tool for T/F decoder
(Error concealment by interpolation etc.)

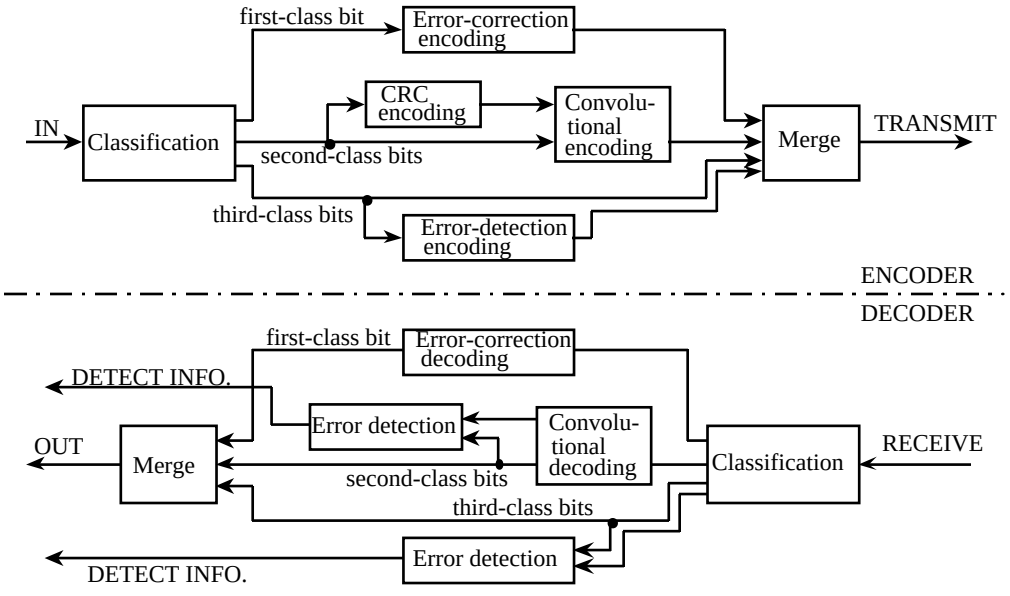


Fig. 8.2. Basic structure of error protection tool.

ISO/JTC 1/SC 29 WG 11 N2203SA

Date: 1998-05-15

ISO/IEC FCD 14496-3 Subpart 5

ISO/JTC 1/SC 29/WG11

Secretariat: Narumi Hirose

Information Technology - Coding of Audiovisual Objects – Low Bitrate Coding of Multimedia Objects

Part 3: Audio

Subpart 5: Structured Audio

Document type: Final Committee Draft
Document:sub-type if applicable
Document:stage (20) Préparation
Document:language E

C:\WINDOWS\Profiles\eds\Desktop\SA FCD 980507.doc ISOSTD Basic Version 1.8 1996-10-30

FCD 14496-3 Subpart 5

MPEG-4 Structured Audio

Editor:
Eric D. Scheirer, MIT Media Laboratory
eds@media.mit.edu
+1 617 253 0112
<http://sound.media.mit.edu/mpeg4>

5.0

5.0 Introduction

5.0.1 Overview of subpart

5.0.1.1 Purpose

The Structured Audio decoder allows for the transmission and decoding of synthetic sound effects and music using several techniques. Using Structured Audio, high-quality sound can be created at extremely low bandwidth. Typical synthetic music may be coded in this format at bit rates ranging from 0 kbps (no continuous cost) to 2 or 3 kbps for extremely subtle coding of expressive performance using multiple instruments.

MPEG-4 does not standardise a particular set of synthesis methods, but a method for describing synthesis methods. Any current or future sound-synthesis method may be described in the MPEG-4 Structured Audio format.

5.0.1.2 Introduction to major elements

There are five major elements to the Structured Audio toolset:

1. The Structured Audio Orchestra Language, or SAOL. SAOL is a digital-signal processing language which allows for the description of arbitrary synthesis and control algorithms as part of the content bitstream. The syntax and semantics of SAOL are standardised here in a normative fashion.

2. The Structured Audio Score Language, or SASL. SASL is a simple score and control language which is used in certain profiles (see Subclause 5.2 Profiles) to describe the manner in which sound-generation algorithms described in SAOL are used to produce sound.

3. The Structured Audio Sample Bank Format, or SASBF. The Sample Bank format allows for the transmission of banks of audio samples to be used in wavetable synthesis and the description of simple processing algorithms to use with them.

4. A normative scheduler description. The scheduler is the supervisory run-time element of the Structured Audio decoding process. It maps structural sound control, specified in SASL or MIDI, to real-time events dispatched using the normative sound-generation algorithms.

5. Normative reference to the MIDI standards, standardised externally by the MIDI Manufacturers Association. MIDI is an alternate means of structural control which can be used in conjunction with or instead of SASL. Although less powerful and flexible than SASL, MIDI support in this standard provides important backward-compatibility with existing content and authoring tools.

5.0.2 Normative References

[MIDI] *The Complete MIDI 1.0 Detailed Specification* v. 96.2, (c) 1996 MIDI Manufacturers Association

5.0.3 Glossary of Terms

Absolute time The time at which sound corresponding to a particular **event** is really created; time in the real-world. Contrast **score time**.

Actual parameter The **expression** which, upon evaluation, is passed to an **opcode** as a parameter value.

A-cycle See **audio cycle**.

A-rate See **audio rate**.

asig The lexical tag indicating an **a-rate variable**.

Audio cycle The sequence of processing which computes new values for all **a-rate expressions** in a particular code block.

Audio rate The **rate type** associated with a **variable**, **expression** or **statement** which may generate new values as often as the **sampling rate**.

Audio sample A short snippet or clip of digitally represented sound. Typically used in **wavetable synthesis**.

Authoring In **Structured Audio**, the combined processes of creatively composing music and sound **control** scripts, creating **instruments** which generate and alter sound, and **encoding** the instruments, control scripts, and **audio samples** in MPEG-4 Structured Audio format.

Backus-Naur Format (**BNF**) A format for describing the syntax of programming languages, used here to specify the **SAOL** and **SASL** syntax. See Subclause 5.0.4.2 SAOL syntax.

Bank A set of **samples** used together to define a particular sound or class of sounds with wavetable **synthesis**.

Beat The unit in which **score time** is measured.

BNF See **Backus-Naur Format.**

Bus An area in memory which is used to pass the output of one **instrument** into the input of another.

Context See **state space**.

Control An instruction used to describe how to use a particular **synthesis** method to produce sound.

EXAMPLES

"Using the piano instrument, play middle C at medium volume for 2 seconds."
"Glissando the violin instrument up to middle C."
"Turn off the reverberation for 8 seconds."

Control cycle The sequence of processing which computes new values for all **control-rate expressions** in a particular code block.

Control period	The length of time (typically measured in audio samples) corresponding to the **control rate**.
Control rate	1. The rate at which **instantiation** and **termination** of **instruments**, parametric **control** of running **instrument instances**, sharing of global **variables**, and other non-sample-by-sample computation occurs in a particular **orchestra**. 2. The **rate type** of **variables**, **expressions**, and **statements** which can generate new values as often as the control rate.
Decoding	The process of turning an MPEG-4 Structured Audio bitstream into sound.
Duration	The amount of time between **instantiation** and **termination** of an instrument **instance**.
Encoding	The process of creating a legal MPEG-4 bitstream, whether automatically, by hand, or using special **authoring** tools.
Envelope	A loudness-shaping function applied to a sound, or more generally, any function controlling a parametric aspect of a sound
Event	One **control** instruction.
Expression	A mathematical or functional combination of **variable** values, **symbolic constants**, and **opcode** calls.
Formal parameter	The syntactic element which gives a name to one of the parameters of an **opcode**.
Future wavetable	A **wavetable** which is declared but not defined in the **SAOL orchestra**; its definition must arrive in the bitstream before it is used.
Global block	The section of the **orchestra** which describes **global variables**, **route** and **send** statements, **sequence rules**, and **global parameters**.
Global context	The **state space** used to hold values of **global variables** and **wavetables**.
Global parameters	The **sampling rate**, **control rate**, and number of input and output channels of audio associated with a particular **orchestra**.
Global variable	A **variable** which can be accessed and/or changed by several different **instruments**.
Grammar	A set of rules which describes the set of allowable sequences of **lexical elements** comprising a particular language.
Guard expression	The expression standing at the front of an **if**, **while**, or **else statement** which determines whether or how many times a particular block of code is executed.
I-cycle	See **initialisation cycle.**
Identifier	A sequence of characters in a textual **SAOL** program which denotes a **symbol**.

Informative Aspects of a standards document which are provided to assist implementors, but are not required to be implemented in order for a particular system to be compliant to the standard.

I-pass See **initialisation pass.**

I-rate See **initialisation rate**.

Initialisation cycle See **initialisation pass.**

Initialisation rate The **rate type** of **variables**, **expressions**, and **statements** which are set once at **instrument instantation** and then do not change.

Initialisation pass The sequence of processing which computes new values for each **i-rate expression** in a particular code block.

Instance See **instrument instantiation**.

Instantiation The process of creating a new **instrument instantiation** based on an **event** in the **score** or **statement** in the **orchestra**.

Instrument An algorithm for parametric sound **synthesis**, described using **SAOL**. An instrument encapsulates all of the algorithms needed for one sound-generation element to be controlled with a **score**.

NOTE

An MPEG-4 Structured Audio instrument does not necessarily correspond to a real-world instrument. A single instrument might be used to represent an entire violin section, or an ambient sound such as the wind. On the other hand, a single real-world instrument which produces many different **timbres** over its performance range might be represented using several SAOL instruments.

Instrument instantiation The **state space** created as the result of executing a note-creation **event** with respect to a **SAOL orchestra**.

ivar The lexical tag indicating an **i-rate variable**.

K-cycle See **control cycle**.

K-rate See **control rate**.

ksig The lexical tag indicating a **k-rate variable**.

Lexical element See **token**.

Looping A typical method of **wavetable synthesis**. Loop points in an **audio sample** are located and the sound between those endpoints is played repeatedly while being simultaneously modified by **envelopes**, modulators**,** etc.

MIDI The Musical Instrument Digital Interface standards, see **[MIDI]** in Subclause 5.0.2 Normative References. MIDI is one method for specifying **control** of **synthesis** in MPEG-4 **Structured Audio**.

Natural Sound A sound created through recording from a real acoustic space. Contrasted with **synthetic sound**.

Normative Those aspects of a standard which must be implemented in order for a particular system to be compliant to the standard.

Opcode A parametric signal-processing function which encapsulates a certain functionality so that it may be used by several **instruments**.

Orchestra The set of sound-generation and sound-processing algorithms included in an MPEG-4 bitstream. Includes **instruments**, **opcodes**, **routing**, and **global parameters**.

Orchestra cycle A complete pass through the orchestra, during which new **instrument instantiations** are created, expired ones are terminated, each **instance** receives one **k-cycle** and one **control period** worth of **a-cycles**, and output is produced.

Parameter fields The names given to the parameters to an **instrument**.

P-fields See **parameter fields**.

Production rule In **Backus-Naur Form** grammars, a rule which describes how one syntactic element may be expressed in terms of other lexical and syntactic elements.

Rate-mismatch error The condition that results when the **rate semantics** rules are violated in a particular **SAOL** construction. A type of **syntax error**.

Rate semantics The set of rules describing how **rate types** are assigned to **variables**, **expressions**, **statements**, and **opcodes**, and the normative restrictions that apply to a bitstream regarding combining these elements based on their **rate types**.

Rate type The “speed of execution” associated with a particular **variable**, **expression**, **statement**, or **opcode**.

Route statement A statement in the **global block** which describes how to place the output of a certain set of **instruments** onto a **bus**.

Run-time error The condition that results from improper calculations or memory accesses during execution of a **SAOL** orchestra.

SASBF See **Sample Bank Format**

SAOL The Structured Audio Orchestra Language, pronounced like the English word “sail.” SAOL is a digital-signal processing language which allows for the description of arbitrary **synthesis** and **control** algorithms as part of the content bitstream.

SAOL orchestra See **orchestra**.

SASL The Structured Audio Score Language. SASL is a simple format which allows for powerful and flexible **control** of music and sound **synthesis**.

Sample See **Audio sample**.

Sample Bank Format A component format of MPEG-4 **Structured Audio** which allows the description of a set of samples for use in **wavetable synthesis** and processing methods to apply to them.

Scheduler The component of MPEG-4 **Structured Audio** which describes the mapping from **control** instructions to **sound synthesis** using the specified synthesis techniques. The scheduler description provides **normative** bounds on event-dispatch times and responses.

Scope The code within which access to a particular **variable** name is allowed.

Score A description in some format of the sequence of **control** parameters needed to generate a desired music composition or sound scene. In MPEG-4 Structured Audio, scores are described in **SASL** and/or **MIDI**.

Score time The time at which an **event** happens in the **score**, measured in **beats**. Score time is mapped to **absolute time** by the current **tempo**.

Send statement A statement in the **global block** which describes how to pass a **bus** on to an **effect instrument** for post-processing.

Semantics The rules describing what a particular instruction or bitstream element should do. Most aspects of bitstream and **SAOL** semantics are **normative** in MPEG-4.

Sequence rules The set of rules, both default and explicit, given in the **global block** which define in what order to execute **instrument instantiations** during an **orchestra cycle**.

Signal variable A unit of memory, labelled with a name, which holds intermediate processing results. Each signal variable in MPEG-4 Structured Audio is instantaneously representable by a 32-bit floating point value.

Spatialisation The process of creating special sounds which a listener perceives as emanating from a particular direction.

State space A set of variable-value associations which define the current computational state of an **instrument instantiation** or **opcode** call. All the “current values” of the variables in an instrument or opcode call.

Statement “One line” of a **SAOL orchestra**.

Structured audio Sound-description methods which make use of high-level models of sound generation and **control**. Typically involving **synthesis** description, structured audio techniques allow for ultra-low bitrate description of complex, high-quality sounds. See **[SAUD]** in Subclause 5.0.5 Bibliography.

Symbol A sequence of characters in a **SAOL** program, or a symbol **token** in a MPEG-4 Structured Audio bitstream, which represents a variable name, instrument name, opcode name, table name, bus name, etc.

Symbol table In an MPEG-4 Structured Audio bitstream, a sequence of data which allows the **tokenised** representation of **SAOL** and **SASL** code to be converted back to a readable textual representation. The symbol table is an optional component.

Symbolic constant A floating-point value explicitly represented as a sequence of characters in a textual **SAOL orchestra**, or as a **token** in a bitstream.

Syntax The rules describing what a particular instruction or bitstream element should look like. All aspects of bitstream and **SAOL** syntax are **normative** in MPEG-4.

Syntax error The condition that results when a bitstream element does not comply with its governing rules of **syntax**.

Synthesis The process of creating sound based on algorithmic descriptions.

Synthetic Sound Sound created through **synthesis**.

Tempo The scaling parameter which specifies the relationship between **score time** and **absolute time**. A tempo of 60 beats per minute means that the **score time** measured in **beats** is equivalent to the **absolute time** measured in seconds; higher numbers correspond to faster tempi, so that 120 beats per minute is twice as fast.

Terminal The “client side” of an MPEG transaction; whatever hardware and software are necessary in a particular implementation to allow the capabilities described in this document.

Termination The process of destroying an **instrument instantiation** when it is no longer needed.

Timbre The combined features of a sound which allow a listener to recognise such aspects as the type of instrument, manner of performance, manner of sound generation, etc. Those aspects of sound which distinguish sounds equivalent in pitch and loudness.

Token A lexical element of a **SAOL orchestra**: a keyword, punctuation mark, **symbol** name, or symbolic constant.

Tokenisation The process of converting a **orchestra** in textual **SAOL** format into a bitstream representation consisting of a stream of **tokens**.

Variable See **signal variable**.

Wavetable synthesis A **synthesis** method in which sound is created by simple manipulation of **audio samples**, such as **looping**, pitch-shifting, **enveloping**, etc.

Width The number of channels of data which an expression represents.

5.0.4 Description methods

5.0.4.1 Bitstream syntax

The Structured Audio bitstream syntax is described using MSDL, the MPEG-4 Syntactic Description Language. See 14496-1 Subclause XXX.

5.0.4.2 SAOL syntax

The textual SAOL syntax (in Subclause 5.4 SAOL syntax and semantics) is described using extended Backus-Naur format (BNF) notation [see **DRAG** in Subclause 5.0.5 Bibliography]. BNF is a description for context-free grammars of programming languages. Normative BNF rules will be described in the ARIEL font.

BNF grammars are composed of terminals, also called tokens, and production rules. Terminals represent syntactic elements of the language, such as keywords and punctuation; production rules describe the composition of these elements into structural groups.

Terminals will be represented in **boldface**; production rules will be represented in <angle brackets>.

The rewrite rules which map productions into sequences of other productions and terminals are represented with the -> symbol.

EXAMPLE

<letter> -> **a**
<letter> -> **b**
<sequence> -> <letter>
<sequence> -> <letter> <sequence>

This grammar (starting from the sequence token) describes, using a recursive rewrite rule and a two-symbol alphabet, all strings containing at least one letter which are made up of ‘a’ and ‘b’ characters.

In addition, rewrite rules using optional elements will be described using the [ ] symbols. Using this notation does not increase the power of the syntax description (in terms of the languages it can represent), but makes certain constructs simpler.

EXAMPLE

<head> -> **c**
<seqhead> -> [<head>] <sequence>

This grammar (starting from the seqhead token) describes, in addition to the set above, all strings beginning with a ‘c’ character and followed by a sequence of ‘a’s and ‘b’s.

The **NULL** token may be used to indicate that a sequence of no characters (the empty string) is a permissible rewrite for a particular production.

Normative aspects of the relationship between the BNF grammar, other grammar representation methods, the bitstream syntax, and the textual description format are described in Subclause 5.4.1 Relationship with bitstream syntax.

5.0.4.3 SASL Syntax

The SASL syntax is specified using extended BNF grammars, as described in Subclause 5.0.4.2 SAOL syntax.

5.0.5 Bibliography

[DRAG] Aho, Alfred V., and Ravi Sethi and Jeffrey Ullman, *Compilers: Principles, Techniques, and Tools*. Reading, Mass: Addison-Wesley, 1984.

[ICASSP] Scheirer, Eric, "The MPEG-4 Structured Audio standard", *Proc 1998 IEEE ICASSP*, Seattle, 1998.

[**NETSOUND]** Casey, Michael, and Paris Smaragdis, "Netsound", *Proc. 1996 ICMC*, Hong Kong, 1996.

[SAFX] Scheirer, Eric, "Structured audio and effects processing in the MPEG-4 multimedia standard", *ACM Multimedia Sys. J.*, in press.

[SAOL] Scheirer, Eric, "SAOL: The MPEG-4 Structured Audio Orchestra Language", *Proc 1998 ICMC*, Ann Arbor, MI, 1998.

[SAUD] Vercoe, Barry, and William G. Gardner and Eric D. Scheirer , "Structured Audio: Creation, Transmission, and Rendering of Parametric Sound Descriptions". *Proc. IEEE* **85**:5 (May 1998), pp.

[WAVE] Scheirer, Eric, and Lee Ray, "Algorithmic and wavetable synthesis in the MPEG-4 multimedia standard". *Proc 105th Conv AES*, San Francisco, 1998.

5.1 Bitstream syntax and semantics

5.1.1 Introduction to bitstream syntax

This Subclause describes the bitstream format defining an MPEG-4 Structured Audio bitstream.

Each group of classes is notated with normative semantics, which define the meaning of the data represented by those classes.

5.1.2 Bitstream syntax

```
/*********************************
      symbol table definitions
*********************************/

class symbol {
  unsigned int(16) sym;         // no more than 65536 symbols/orch + score
}

class sym_name {                // one name in a symbol table
  unsigned int(4) length;       // names up to 16 chars long
  unsigned int(8) name[length];
}

class symtable {                // a whole symbol table
  unsigned int(16) length;      // no more than 65536 symbols/orch+score
  sym_name name[length];
}
```

A bitstream may contain a symbol table, but this is not required. The symbol table allows textual SAOL and SASL code to be recovered from the tokenised bitstream representation. The inclusion or exclusion of a symbol table does not affect the decoding process.

If a symbol table is included, then all or some of the symbols in the orchestra and score shall be associated with a textual name in the following way: each symbol (a symbol is just an integer) shall be associated with the character string paired with that symbol in a `sym_name` object. There shall be no more than one name associated with a given symbol, otherwise the bitstream is invalid. It is permissible for the symbol table to be incomplete and contain names associated with some, but not all, symbols used in the orchestra and score.

SAOL and SASL implementations which require textual input, rather than tokenised input, are permissible in a compliant decoder, in which case the decoder must detokenise the bitstream before it can be processed. In such a case, any symbols without associated names are suggested to be associated with a default name of the form `_sym_x`, where x is the symbol value. Names of this form are reserved in SAOL for this purpose, and so following this suggestion guarantees that names will not clash with symbol-table-defined symbol names.

```
/*********************************
    orchestra file definitions
*********************************/

class orch_token {              // a token in an orchestra
  int done;

  unsigned int(8) token;         // see standard token table, Annex A
  switch (token) {
  case 0xF0 :                   // a symbol
    symbol sym;                 // the symbol name
    break;
```

```
  case 0xF1 :                   // a constant value
    float(32) val;              // the floating-point value
    break;
  case 0xF2 :                    // a constant int value
    unsigned int(32) val;       // the integer value
    break;
  case 0xF3 :                   // a string constant
    int(8) length;
    unsigned int(8) str[length]; // strings no more than 256 chars
    break;
  case 0xFF : // end of orch
    done = 1;
    break;
  }
}

class orc_file {                // a whole orch file
  unsigned int(16) length;
  orch_token data[length];
}
```

An orchestra file is a string of tokens. These tokens represent syntactic elements such as reserved words, core opcode names, and punctuation marks as given in the table in Annex A; in addition, there are five special tokens. Token 0xF0 is the symbol token; when it is encountered, the next 16 bits in the bitstream shall be a symbol number. Token 0xF1 is the value token; when it is encountered, the next 32 bits in the bitstream shall be a floating-point value. This token shall be used for all symbolic constants within the SAOL program except for those encountered in special integer contexts, as described in Subclause 5.8 SAOL/SASL . Token 0xF2 is the integer token; when it is encountered, the next 32 bits in the bitstream shall be an unsigned integer value. Token 0xF3 is the string token; when it is encountered, the next several bits in the bitstream shall represent a character string (this token is currently unused). Token 0xFF is the end-of-orchestra token; this token has no syntactic function in the SAOL orchestra, but signifies the end of the orchestra file section of the bitstream.

Not every sequence of tokens is permitted to occur as an orchestra file. Subclause 5.4 SAOL syntax and semantics contains extensive syntactic rules restricting the possible sequence of tokens, described according to the textual SAOL format. Normative rules for mapping back and forth between the tokenised format and the textual format are given in Subclause 5.8 SAOL/SASL . The overall sequence of orchestra tokens shall correspond to an <orchestra> production as given in Subclause 5.4.4 Orchestra.

```
/*********************************
      score file definitions
*********************************/

class instr_event {             // a note-on event
  bit(1) has_label;
  if (has_label)
    symbol label;
  symbol iname_sym;             // the instrument name
  float(32) dur;                // note duration
  unsigned int(8) num_pf;
  float(32) pf[num_pf];         // all the pfields (no more than 256)
}

class control_event {           // a control event
  bit(1) has_label;
  if (has_label)
    symbol label;
  symbol varsym;                // the controller name
  float(32) value;              // the new value
}

class table_event {
  symbol tname;       // the name of the table
```

```
  bit(1) destroy;     // a table destructor
  if (!destroy) {
    token tgen;         // a core wavetable generator
    bit(1) refers_to_sample;
    if (refers_to_sample)
      symbol table_sym;          // the name of the sample
    unsigned int(16) num_pf;     // the number of pfields
    float(32) pf[num_pf];        // all the pfields
  }
}

class end_event {
  // fixed at nothing
}

class tempo_event {  // a tempo event
  float(32) tempo;
}

class score_line {
  float(32) time;              // the event time
  bit(3) type;
  switch (type) {
    case 0b000 : instr_event inst; break;
    case 0b001 : control_event control; break;
    case 0b010 : table_event table; break;
    case 0b100 : end_event end; break;
    case 0b101 : tempo_event tempo; break;
  }
}

class score_file {
  unsigned int(20) num_lines;  // a whole score file
  score_line lines[num_lines];
}
```

A score file is a set of lines of score information provided in the stream information header. Thus, events which are known before the real-time bitstream transmission begins may be included in the header, so that they are available to the decoder immediately, which may aid efficient computation in certain implementations. Each line shall be one of five events. Each type of event has different implications in the decoding and scheduling process, see Subclause 5.3.3 Bitstream data and sound creation. An instrument event specifies the start time, instrument name symbol, duration, and any other parameters of a note played on a SAOL instrument. A control event specifies a control parameter which is passed to a instrument or instruments already generating sound. A table event dynamically creates or destroys a global wavetable in the orchestra. An end event signifies the end of orchestra processing. A tempo event dynamically changes the tempo of orchestra playback.

A score file need not be presented in increasing order of event times; the events shall be “sorted” by the scheduler as they are processed.

```
/*********************************
         MIDI definitions
*********************************/

/* NB that a midi_file (SMF format) is not just an array
   of MIDI events */

class midi_event {
  // not done yet
}

class midi_file {
  /* Right now it’s just an array of bytes; I’d rather do
```

```
     this that lay out the whole SMF format here */
  unsigned int(20) length;
  unsigned int(8) data[length];
}
```

The MIDI chunks allow the inclusion of MIDI score information in the bitstream header and bitstream. The MIDI event class contains a single MIDI instruction as specified in **[MIDI]**; the MIDI file class contains an array of bytes corresponding to a Standard MIDIFile as specified in **[MIDI]**. Note that not every sequence of data may occur in either case; the legal syntaxes of MIDI events and MIDIFiles as specified in **[MIDI]** place normative bounds on syntactically valid MPEG-4 Structured Audio bitstreams. The semantics of MIDI data are given in Subclause 5.9 Sample Bank syntax and semantics (for Profile 1 and 2 implementations) and Subclause 0 (for Profile 4 implementations).

```
/**********************************
         sample data
**********************************/

class sample {
  /* note that 'sample' can be used for any big chunk of data
     which needs to get into a wavetable */
  symbol sample_name_sym;
  unsigned int(24) length;  // length in samples
  bit(1) has_srate;
  if (has_srate)
    unsigned int(17) srate; // sampling rate (needs to go to 96 KHz)
  bit(1) has_loop;
  if (has_loop) {
    unsigned int(24) loopstart; // loop points in samples
    unsigned int(24) loopend;
  }
  bit(1) has_base;
  if (has_base)
    float(32) basecps;         // base freq in Hz
  bit(1) float_sample;
  if (float_sample) {
    float(32) float_sample_data[length];
  }
  else {
    int(16) sample_data[length]; // all the data
  }
}
```

A sample chunk includes a block data which will be included in a wavetable in a SAOL orchestra. Each sample consists of a name, a length, a block of data, and four optional parameters: the sampling rate, the loop start and loop end points, and the base frequency. Access to the data in the sample is provided through the **sample** core wavetable generator, see Subclause 5.6.2 Sample.

The sample data may be represented either as 32-bit floating point values, in which case it shall be scaled between –1 and 1, or may be represented as 16-bit integer values, in which case it shall be scaled between -32767 and 32768. In the case that the sample data is represented as integer values, upon inclusion in a wavetable, it shall be rescaled to floating-point as described in Subclause 5.6.2 Sample.

```
/**********************************
         sample bank data
**********************************/
```

The sample bank chunk describes a bank of wavetable data and associated processing parameters for use with the sample bank synthesis procedure in Subclause 5.9 Sample Bank syntax and semantics.

```
const int sbf_chunk_ID = 0x7366626b;                // 'sfbk'
const int INFO_list_ID       = 0x494e464f;          // 'INFO'
const int ifil_chunk_ID      = 0x6966696c;          // 'ifil'
```

```
const int isng_chunk_ID      = 0x69736e67;           // ‘isng’
const int INAM_chunk_ID      = 0x494e414d;           // ‘INAM’
const int irom_chunk_ID      = 0x69726f5d;           // ‘irom’
const int iver_chunk_ID      = 0x69766572;           // ‘iver’
const int ICRD_chunk_ID      = 0x49435244;           // ‘ICRD’
const int IENG_chunk_ID      = 0x49454e47;           // ‘IENG’
const int IPRD_chunk_ID      = 0x49505244;           // ‘IPRD’
const int ICOP_chunk_ID      = 0x49434f50;           // ‘ICOP’
const int ICMT_chunk_ID      = 0x49434d54;           // ‘ICMT’
const int ISFT_chunk_ID      = 0x49534654;           // ‘ISFT’
const int sdta_chunk_ID      = 0x73647461;           // ‘sdta’
const int smpl_chunk_ID      = 0x736d706c;           // ‘smpl’
const int pdta_chunk_ID      = 0x70647461;           // ‘pdta’
const int phdr_chunk_ID      = 0x70686472;           // ‘phdr’
const int pbag_chunk_ID      = 0x70626167;           // ‘pbag’
const int pmod_chunk_ID      = 0x706d6f64;           // ‘pmod’
const int pgen_chunk_ID      = 0x7067656e;           // ‘pgen’
const int inst_chunk_ID      = 0x696e7374;           // ‘inst’
const int ibag_chunk_ID      = 0x69626167;           // ‘ibag’
const int imod_chunk_ID      = 0x696d6f64;           // ‘imod’
const int igen_chunk_ID      = 0x6967656e;           // ‘igen’
const int shdr_chunk_ID      = 0x73686472;           // ‘shdr’


aligned(16) class chunk: bit(32) ckID = 0x00000000 {
  unsigned int(32) ckSize; // size of chunk data in bytes
}

class ifil_chunk extends chunk: bit(32) ckID = ifil_chunk_ID {
  unsigned int(16) wMajor;       // file format version number
  unsigned int(16) wMinor;
}

class isng_chunk extends chunk: bit(32) ckID = isng_chunk_ID {
  char(8) isng[ck_hdr.ckSize];          // sound engine identifier
}

class INAM_chunk extends chunk: bit(32) ckID = INAM_chunk_ID {
  char(8) INAM[ck_hdr.ckSize];          // bank name
}

class irom_chunk extends chunk: bit(32) ckID = irom_chunk_ID {
  char(8) irom[ck_hdr.ckSize];          // rom name
}

class iver_chunk extends chunk: bit(32) ckID = iver_chunk_ID {
  unsigned int(16) wMajor;       // rom version
  unsigned int(16) wMinor;
}

class ICRD_chunk extends chunk: bit(32) ckID = ICRD_chunk_ID {
  char(8) ICRD[ck_hdr.ckSize];          // creation date
}

class IENG_chunk extends chunk: bit(32) ckID = IENG_chunk_ID {
  char(8) IENG[ck_hdr.ckSize];          // sound designer name
}

class IPRD_chunk extends chunk: bit(32) ckID = IPRD_chunk_ID {
  char(8) IPRD[ck_hdr.ckSize];          // product name
}

class ICOP_chunk extends chunk: bit(32) ckID = ICOP_chunk_ID {
  char(8) ICOP[ck_hdr.ckSize];          // copyright string
}

class ICMT_chunk extends chunk: bit(32) ckID = ICMT_chunk_ID {
```

```
  char(8) ICMT[ck_hdr.ckSize];          // comment string
}

class ISFT_chunk extends chunk: bit(32) ckID = ISFT_chunk_ID {
  char(8) ISFT[ck_hdr.ckSize];          // tool name
}

class INFO_list extends chunk: bit(32) ckID = INFO_list_ID {
  ifil_chunk ifil_ck;
  isng_chunk isng_ck;
  INAM_chunk INAM_ck;
  aligned(16) bit(32)* test0;
  if (test0 == irom_chunk_ID) {
    irom_chunk irom_ck;
  }
  aligned(16) bit(32)* test1;
  if (test1 == iver_chunk_ID) {
    iver_chunk iver_ck;
  }
  aligned(16) bit(32)* test2;
  if (test2 == ICRD_chunk_ID) {
    ICRD_chunk ICRD_ck;
  }
  aligned(16) bit(32)* test3;
  if (test3 == IENG_chunk_ID) {
    IENG_chunk IENG_ck;
  }
  aligned(16) bit(32)* test4;
  if (test4 == IPRD_chunk_ID) {
    IPRD_chunk IPRD_ck;
  }
  aligned(16) bit(32)* test5;
  if (test5 == ICOP_chunk_ID) {
    ICOP_chunk ICOP_ck;
  }
  aligned(16) bit(32)* test6;
  if (test6 == ICMT_chunk_ID) {
    ICMT_chunk ICMT_ck;
  }
  aligned(16) bit(32)* test7;
  if (test7 == ISFT_chunk_ID) {
    ISFT_chunk ISFT_ck;
  }
}

class smpl_chunk extends chunk: bit(32) ckID = smpl_chunk_ID {
  int(16) smpl[ck_hdr.ckSize / 2];      // sample data
}

class sdta_list extends chunk: bit(32) ckID = sdta_chunk_ID {
  smpl_chunk smpl_ck;
}

class phdr_chunk extends chunk: bit(32) ckID = phdr_chunk_ID {
  unsigned int i;
  for (i = 0; i < ck_hdr.ckSize / 38; i++) {
    char(8) achPresetName[20];
    unsigned int(16) wPreset;
    unsigned int(16) wBank;
    unsigned int(16) wPresetBagNdx;
    unsigned int(32) dwLibrary;
    unsigned int(32) dwGenre;
    unsigned int(32) dwMorphology;
  }
}

class bag_chunk extends chunk {
```

```
  unsigned int i;
  for (i = 0; i < ck_hdr.ckSize / 4; i++) {
    unsigned int(16) wGenNdx;
    unsigned int(16) wModNdx
  }
}

class pbag_chunk extends bag_chunk: bit(32) ckID = pbag_chunk_ID {
}

class ibag_chunk extends bag_chunk: bit(32) ckID = ibag_chunk_ID {
}

class mod_chunk extends chunk {
  unsigned int i;
  for (i = 0; i < ck_hdr.ckSize / 10; i++) {
    unsigned int(16) sfModSrcOper;
    unsigned int(16) sfModDestOper;
    int(16) modAmount;
    unsigned int(16) sfModAmtSrcOper;
    unsigned int(16) sfModTransOper;
  }
}

class pmod_chunk extends mod_chunk: bit(32) ckID = pmod_chunk_ID {
}

class imod_chunk extends mod_chunk: bit(32) ckID = imod_chunk_ID {
}

class gen_chunk extends chunk {
  unsigned int i;
  for (i = 0; i < ck_hdr.ckSize / 4; i++) {
    unsigned int(16) sfGenOper;
    bit(16) genAmount;
  }
}

class pgen_chunk extends gen_chunk: bit(32) ckID = pgen_chunk_ID {
}

class igen_chunk extends gen_chunk: bit(32) ckID = igen_chunk_ID {
}

class inst_chunk extends chunk {
  unsigned int i;
  for (i = 0; i < ck_hdr.ckSize / 22; i++) {
    char(8) achInstName[20];
    unsigend int(16) wInstBagNdx;
  }
}

class shdr_chunk extends chunk {
  unsigned int i;
  for (i = 0; i < ck_hdr.ckSize / 46; i++) {
    char(8) achSampleName[20];
    unsigned int(32) dwStart;
    unsigned int(32) dwEnd;
    unsigned int(32) dwStartloop;
    unsigned int(32) dwEndloop;
    unsigned int(32) dwSampleRate;
    unsigned int(8) byOriginalPitch;
    int(8) chCorrection;
    unsigned int(16) wSampleLink;
    unsigned int(16) sfSampleType;
  }
}
```

```
class pdta_list extends chunk: bit(32) ckID = pdta_chunk_ID {
  phdr_chunk phdr_ck;                  // preset headers
  pbag_chunk pbag_ck;                  // preset index list
  pmod_chunk pmod_ck;                  // preset modulator list
  pgen_chunk pgen_ck;                  // preset generator list
  inst_chunk inst_ck;                  // instrument names and indices
  ibag_chunk ibag_ck;                  // instrument index list
  imod_chunk imod_ck;                  // instrument modulator list
  igen_chunk igen_ck;                  // instrument generator list
  shdr_chunk shdr_ck;                  // sample headers
}

class sbf extends chunk: bit(32) ckID = sbf_chunk_ID {
  INFO_list INFO_lt;
  sdta_list sdta_lt;
  pdta_list pdta_lt;
}

/*********************************
         bitstream formats
*********************************/

class SA_decoder_config {  // the bitstream header
  bit more_data = 1;

  while (more_data) {  // must have at least one chunk
    bit(3) chunk_type;
    switch (chunk_type) {
    case 0b000 : orc_file orc; break;
    case 0b001 : score_file score; break;
    case 0b010 : midi_file SMF; break;
    case 0b011 : sample samp; break;
    case 0b100 : sbf sample_bank; break;
    case 0b101 : symtable sym; break;
    }
    bit(1) more_data;
  }
}
```

The bitstream decoder configuration contains all the information required to configure and start up a structured audio decoder. It contains a sequence of one or more chunks, where each chunk is of one of the following types: orchestra file, score file, midi file, sample data, sample bank, or symbol table.

```
class SA_access_unit {      // the streaming data

 bit(2) event_type;
 switch (event_type) {
   case 0b00 : score_line score_ev; break;
   case 0b01 : midi_event midi_ev; break;
   case 0b10 : sample samp; break;
   }
}
```

The Structured Audio access unit contains real-time streaming control information to be provided to a running Structured Audio decoding process. It shall not contain new instrument definitions; the orchestra configuration is fixed at decoder startup. It may contain score lines, MIDI events, and new sample data.

5.2 Profiles

There are three profiles standardised for Structured Audio, called Profile1, Profile2, and Profile4. Each of these profiles corresponds to a particular set of application requirements. The default profile is Profile 4; when reference is made to MPEG-4 Structured Audio format without reference to a profile, it shall be understood that the reference is to Profile 4.

Terminals implementing MPEG-4 Systems Audio Composition Profile XXX (see ISO/IEC 14496-1, Subclause XXX) shall also implement Structured Audio Profile 4.

1. **MIDI only. In this profile, only the** midi_file **chunk shall occur in the stream information header, and only the** midi_event **event shall occur in the bitstream data. In this profile, the General MIDI patch mappings are used, and the decoding process is described in Subclause 5.9 Sample Bank syntax and semantics**

5.9.1 Introduction

This Subclause describes the operation of the Sample Bank synthesis method for both Profile 2 and Profile 4. In Profile 2, only Sample Bank and MIDI class types shall appear in the bitstream, and this Subclause describes the normative process of generating sound from a Sample Bank bitstream data element and a sequence of MIDI instructions. In Profile 4, Sample Banks are used in the context of a SAOL instrument as described in Subclause 5.4.6.6.8 Output, and this Subclause describes the normative process of generating sound and placing it on three busses, depending on the Sample Bank bitstream data element and the particular call to **sbsynth**.

5.9.2 Elements of bitstream

5.9.2.1 RIFF Structure

5.9.2.1.1 General RIFF File Structure

The RIFF (Resource Interchange File Format) is a tagged file structure developed for multimedia resource files, and is described in some detail in the Microsoft Windows 3.1 SDK Multimedia Programmer's Reference. The Tagged-file structure is useful because it helps prevent compatibility problems which can occur as the file definition changes over time. Because each piece of data in the file is identified by a standard header, an application that does not recognise a given data element can skip over the unknown information. The relevant information is provided in the bitstream description, Subclause 0.

A RIFF file is constructed from a basic building block called a "chunk." In 'C' syntax, a chunk is defined:

```
typedef DWORD FOURCC;      //  Four-character code
typedef struct {
  FOURCC ckID;             //  Chunk ID identifies type of data in the chunk.
  DWORD ckSize;            //  Size of chunk data in bytes, excluding pad byte.
  BYTE ckDATA[ckSize];     //  Actual data + a pad byte if req'd to word align.
};
```

Two types of chunks, the "RIFF" and "LIST" chunks, may contain nested chunks called subchunks as their data.

Within a given level of the hierarchy, the ordering of the chunks is arbitrary. Any chunk with an unknown chunk ID should be ignored.

5.9.2.1.2 The Sample Bank Chunks and Subchunks

A Structured Audio Sample Bank bitstream element comprises three chunks: an INFO-list chunk containing a number of required and optional subchunks describing the bitstream element, its history, and its intended

use, an sdta-list chunk comprising a single subchunk containing any referenced digital audio samples, and a pdta-list chunk containing nine subchunks which define the articulation of the digital audio data.

5.9.2.1.3 Redundancy and Error Handling in the RIFF structure

The RIFF bitstream element structure contains redundant information regarding the length of the bitstream element and the length of the chunks and subchunks. This fact enables any reader of a SASBF bitstream element to determine if the bitstream element has been damaged by loss of data.

If any such loss is detected, the SASBF bitstream element is termed “structurally unsound” and in general should be rejected.

5.9.2.2 The INFO-list Chunk

The INFO-list chunk in a SASBF bitstream element contains three mandatory and a variety of optional subchunks as defined below. The INFO-list chunk gives basic information about the SASBF bank contained in the bitstream element.

5.9.2.2.1 The ifil Subchunk

The ifil subchunk is a mandatory subchunk identifying the SASBF specification version level to which the bitstream element complies. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the SASBF specification version. The WORD wMinor contains the value to the right of the decimal point. For example, version 2.11 would be implied if wMajor=2 and wMinor=11.

These values can be used by applications which read SASBF bitstream elements to determine if the format of the bitstream element is usable by the program. Within a fixed wMajor, the only changes to the format will be the addition of Generator, Source and Transform enumerators, and additional info subchunks. These are all defined as being ignored if unknown to the program.

If the ifil subchunk is missing, or its size is not four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.2.2 The isng Subchunk

The isng subchunk is a mandatory subchunk identifying the wavetable sound engine for which the bitstream element was optimised. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even.

The ASCII should be treated as case-sensitive. In other words “engine” is not the same as “ENGINE.”

The isng string may be optionally used by chip drivers to vary their synthesis algorithms to emulate the target sound engine.

If the isng subchunk is missing not terminated in a zero valued byte, or its contents are an unknown sound engine, the field should be ignored.

5.9.2.2.3 The INAM Subchunk

The INAM subchunk is a mandatory subchunk providing the name of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical inam subchunk would be the fourteen bytes representing “General MIDI” as twelve ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “General MIDI” is not the same as “GENERAL MIDI.”

If the inam subchunk is missing, or not terminated in a zero valued byte, the field should be ignored and the user supplied with an appropriate error message if the name is queried. If the bitstream element is re-written, a valid name should be placed in the INAM field.

5.9.2.2.4 The irom Subchunk

The irom subchunk is an optional subchunk identifying a particular wavetable sound data ROM to which any ROM samples refer. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical irom field would be the six bytes representing “1MGM” as four ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “1mgm” is not the same as “1MGM.”

The irom string is used by drivers to verify that the ROM data referenced by the bitstream element is available to the sound engine.

If the irom subchunk is missing, not terminated in a zero valued byte, or its contents are an unknown ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. A bitstream element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.5 The iver Subchunk

The iver subchunk is an optional subchunk identifying the particular wavetable sound data ROM revision to which any ROM samples refer. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the ROM version. The WORD wMinor contains the value to the right of the decimal point. For example, version 1.36 would be implied if wMajor=1 and wMinor=36.

The iver subchunk is used by drivers to verify that the ROM data referenced by the bitstream element is located in the exact locations specified by the sound headers.

If the iver subchunk is missing, not four bytes in length, or its contents indicate an unknown or incorrect ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. Note that for ROM samples to function correctly, both iver and irom must be present and valid. A bitstream

element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.6 The ICRD Subchunk

The ICRD subchunk is an optional subchunk identifying the creation date of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICRD field would be the twelve bytes representing “May 1, 1995” as eleven ASCII characters followed by a zero byte.

Conventionally, the format of the string is “Month Day, Year” where Month is initially capitalised and is the conventional full English spelling of the month, Day is the date in decimal followed by a comma, and Year is the full decimal year. Thus the field should conventionally never be longer than 32 bytes.

The ICRD string is provided for library management purposes.

If the ICRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.7 The IENG Subchunk

The IENG subchunk is an optional subchunk identifying the names of any sound designers or engineers responsible for the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IENG field would be the twelve bytes representing “Tim Swartz” as ten ASCII characters followed by two zero bytes.

The IENG string is provided for library management purposes.

If the IENG subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.8 The IPRD Subchunk

The IPRD subchunk is an optional subchunk identifying any specific product for which the SASBF bank is intended. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IPRD field would be the twelve bytes representing “MPEG SASBF” as ten ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “mpeg sasbf” is not the same as “MPEG SASBF.”

The IPRD string is provided for library management purposes.

If the IPRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.9 The ICOP Subchunk

The ICOP subchunk is an optional subchunk containing any copyright assertion string associated with the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICOP field would be the 38 bytes representing "Copyright (c) 1998 Content Developer" as 36 ASCII characters followed by two zero bytes.

The ICOP string is provided for intellectual property protection and management purposes.

If the ICOP subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.10 The ICMT Subchunk

The ICMT subchunk is an optional subchunk containing any comments associated with the SASBF bank. It contains an ASCII string of 65,536 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICMT field would be the 40 bytes representing "This space unintentionally left blank." as 38 ASCII characters followed by two zero bytes.

The ICMT string is provided for including comments.

If the ICMT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.11 The ISFT Subchunk

The ISFT subchunk is an optional subchunk identifying the SASBF tools used to create and most recently modify the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ISFT field would be the twenty-six bytes representing "Editor 2.00a:Editor 2.00a" as twenty-five ASCII characters followed by a zero byte.

The ASCII should be treated as case-sensitive. In other words "Editor" is not the same as "EDITOR."

Conventionally, the tool name and revision control number are included first for the creating tool and then for the most recent modifying tool. The two strings are separated by a colon. The string should be produced by the creating program with a null modifying tool field (e.g. "Editor 2.00a:), and each time a tool modifies the bank, it should replace the modifying tool field with its own name and revision control number.

The ISFT string is provided primarily for error tracing purposes.

If the ISFT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.3 The sdta-list Chunk

The sdta-list chunk in a SASBF bitstream element contains a single optional smpl subchunk which contains all the sound data associated with the SASBF bank. The smpl subchunk is of arbitrary length, and contains an even number of bytes.

5.9.2.3.1 Sample Data Format in the smpl Subchunk

The smpl subchunk, contains one or more samples of digital audio information in the form of linearly coded sixteen bit, signed, little endian (least significant byte first) words. Each sample is followed by a minimum of forty-six zero valued sample data points. These zero valued data points are necessary to guarantee that any reasonable upward pitch shift using any reasonable interpolator can loop on zero data at the end of the sound.

5.9.2.3.2 Sample Data Looping Rules

Within each sample, one or more loop point pairs may exist. The locations of these points are defined within the pdta-list chunk, but the sample data points themselves must comply with certain practices in order for the loop to be compatible across multiple platforms.

The loops are defined by “equivalent points” in the sample. This means that there are two sample data points which are logically equivalent, and a loop occurs when these points are spliced atop one another. In concept, the loop end point is never actually played during looping; instead the loop start point follows the point just prior to the loop end point. Because of the bandlimited nature of digital audio sampling, an artefact free loop will exhibit virtually identical data surrounding the equivalent points.

In actuality, because of the various interpolation algorithms used by wavetable synthesisers, the data surrounding both the loop start and end points may affect the sound of the loop. Hence both the loop start and end points must be surrounded by continuous audio data. For example, even if the sound is programmed to continue to loop throughout the decay, sample data points must be provided beyond the loop end point. This data will typically be identical to the data at the start of the loop. A minimum of eight valid data points are required to be present before the loop start and after the loop end.

The eight data points (four on each side) surrounding the two equivalent loop points should also be forced to be identical. By forcing the data to be identical, all interpolation algorithms are guaranteed to properly reproduce an artefact-free loop.

5.9.2.4 The pdta-list Chunk

5.9.2.4.1 The PHDR Subchunk

The PHDR subchunk is a required subchunk listing all presets within the SASBF bitstream element. It shall be a multiple of thirty-eight bytes in length, and shall contain a minimum of two records, one record for each preset and one for a terminal record according to the structure:

```
struct sfPresetHeader
{
       CHAR achPresetName[20];
       WORD wPreset;
       WORD wBank;
       WORD wPresetBagNdx;
       DWORD dwLibrary;
       DWORD dwGenre;
       DWORD dwMorphology;
};
```

The ASCII character field achPresetName shall contain the name of the preset expressed in ASCII, with unused terminal characters filled with zero valued bytes. Preset names are case-sensitive. A unique name shall always be assigned to each preset in the SASBF bank.

The WORD wPreset shall contain the MIDI Preset Number and the WORD wBank the MIDI Bank Number which apply to this preset. Note that the presets are not ordered within the SASBF bank. Presets

shall have a unique set of wPreset and wBank numbers. The special case of a General MIDI percussion bank is handled conventionally by a wBank value of 128. If the value in either field is not a valid MIDI value of 0 through 127, or 128 for wBank, the preset cannot be played but shall be maintained in memory for future renumbering.

The WORD wPresetBagNdx is an index to the preset's zone list in the PBAG subchunk. Because the preset zone list is in the same order as the preset header list, the preset bag indices shall be monotonically increasing with increasing preset headers. The size of the PBAG subchunk in bytes shall be equal to four times the terminal preset's wPresetBagNdx plus four. If the preset bag indices are non-monotonic or if the terminal preset's wPresetBagNdx does not match the PBAG subchunk size, the SASBF chunk is structurally defective and shall be rejected at load time. All presets except the terminal preset shall have at least one zone.

The DWORDs dwLibrary, dwGenre and dwMorphology are reserved for future implementation in a preset library management function and should be preserved as read, and created as zero.

The terminal sfPresetHeader record should never be accessed, and exists only to provide a terminal wPresetBagNdx with which to determine the number of zones in the last preset. All other values shall be conventionally zero, with the exception of achPresetName, which may be optionally be "EOP" indicating end of presets.

If the PHDR subchunk is missing, contains fewer than two records, or its size is not a multiple of 38 bytes, the SASBF bitstream element shall be rejected as structurally unsound.

5.9.2.4.2 The PBAG Subchunk

The PBAG subchunk is a required subchunk listing all preset zones within the SASBF bitstream element. It shall be a multiple of four bytes in length, and shall contain one record for each preset zone plus one record for a terminal zone according to the structure:

```
struct sfPresetBag
{
       WORD wGenNdx;
       WORD wModNdx;
};
```

The first zone in a given preset shall be located at that preset's wPresetBagNdx. The number of zones in the preset shall be determined by the difference between the next preset's wPresetBagNdx and the current wPresetBagNdx.

The WORD wGenNdx shall be an index to the preset's zone list of generators in the PGEN subchunk, and the wModNdx shall be an index to its list of modulators in the PMOD subchunk. Because both the generator and modulator lists are in the same order as the preset header and zone lists, these indices will be monotonically increasing with increasing preset zones. The size of the PMOD subchunk in bytes shall be equal to ten times the terminal preset's wModNdx plus ten and the size of the PGEN subchunk in bytes shall be equal to four times the terminal preset's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective PGEN or PMOD subchunks, the bitstream element is structurally defective and shall be rejected at load time.

If a preset has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not an Instrument generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks an Instrument generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the PBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.3 The PMOD Subchunk

The PMOD subchunk is a required subchunk listing all preset zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The preset zone's wModNdx points to the first modulator for that preset zone, and the number of modulators present for a preset zone is determined by the difference between the next higher preset zone's wModNdx and the current preset's wModNdx. A difference of zero indicates there are no modulators in this preset zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator with the same zone, the first modulator will be ignored.

Modulators in the PMOD subchunk act as additively relative modulators with respect to those in the IMOD subchunk. In other words, a PMOD modulator can increase or decrease the amount of an IMOD modulator.

In SASBF, no modulators have yet been defined, and the PMOD subchunk will always consist of ten zero valued bytes.

If the PMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.4 The PGEN Subchunk

The PGEN chunk is a required chunk containing a list of preset zone generators for each preset zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each preset zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined:

```
typedef struct
{
       BYTE byLo;
       BYTE byHi;
} rangesType;

typedef union
{
       rangesType ranges;
       SHORT shAmount;
       WORD wAmount;
} genAmountType;
```

The sfGenOper is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. This value indicates the type of generator being indicated. Note that this enumeration is two bytes in length.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers or MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The preset zone's wGenNdx points to the first generator for that preset zone. Unless the zone is a global zone, the last generator in the list is an "Instrument" generator, whose value is a pointer to the instrument associated with that zone. If a "key range" generator exists for the preset zone, it is always the first generator in the list for that preset zone. If a "velocity range" generator exists for the preset zone, it will only be preceded by a key range generator. If any generators follow an Instrument generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator with the same zone, the first generator will be ignored.

Generators in the PGEN subchunk are applied relative to generators in the IGEN subchunk in an additive manner. In other words, PGEN generators increase or decrease the value of an IGEN generator.

If the PGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in an instrument generator, the zone should be ignored. If the instrument generator value is equal to or greater than the terminal instrument, the bitstream element should be rejected as structurally unsound.

5.9.2.4.5 The INST Subchunk

The inst subchunk is a required subchunk listing all instruments within the SASBF bitstream element. It is always a multiple of twenty-two bytes in length, and contains a minimum of two records, one record for each instrument and one for a terminal record according to the structure:

```
struct sfInst
{
       CHAR achInstName[20];
       WORD wInstBagNdx;
};
```

The ASCII character field achInstName contains the name of the instrument expressed in ASCII, with unused terminal characters filled with zero valued bytes. Instrument names are case-sensitive. A unique name should always be assigned to each instrument in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of instruments with identical names, the instruments should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The WORD wInstBagNdx is an index to the instrument's zone list in the IBAG subchunk. Because the instrument zone list is in the same order as the instrument list, the instrument bag indices will be monotonically increasing with increasing instruments. The size of the IBAG subchunk in bytes will be four greater than four times the terminal (EOI) instrument's wInstBagNdx. If the instrument bag indices are non-monotonic or if the terminal instrument's wInstBagNdx does not match the IBAG subchunk size, the bitstream element is structurally defective and should be rejected at load time. All instruments except the terminal instrument must have at least one zone; any preset with no zones should be ignored.

The terminal sfInst record should never be accessed, and exists only to provide a terminal wInstBagNdx with which to determine the number of zones in the last instrument. All other values are conventionally zero, with the exception of achInstName, which should be "EOI" indicating end of instruments.

If the INST subchunk is missing, contains fewer than two records, or its size is not a multiple of 22 bytes, the bitstream element should be rejected as structurally unsound. All instruments present in the inst subchunk are typically referenced by a preset zone. However, a bitstream element containing any "orphaned" instruments need not be rejected. SASBF applications can optionally ignore or filter out these orphaned instruments based on user preference.

5.9.2.4.6 The IBAG Subchunk

The IBAG subchunk is a required subchunk listing all instrument zones within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one record for each instrument zone plus one record for a terminal zone according to the structure:

```
struct sfInstBag
{
       WORD wInstGenNdx;
       WORD wInstModNdx;
```

```
};
```

The first zone in a given instrument is located at that instrument's wInstBagNdx. The number of zones in the instrument is determined by the difference between the next instrument's wInstBagNdx and the current wInstBagNdx.

The WORD wInstGenNdx is an index to the instrument zone's list of generators in the IGEN subchunk, and the wInstModNdx is an index to its list of modulators in the IMOD subchunk. Because both the generator and modulator lists are in the same order as the instrument and zone lists, these indices will be monotonically increasing with increasing zones. The size of the IMOD subchunk in bytes will be equal to ten times the terminal instrument's wModNdx plus ten and the size of the IGEN subchunk in bytes will be equal to four times the terminal instrument's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective IGEN or IMOD subchunks, the bitstream element is structurally defective and should be rejected at load time.

If an instrument has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not a sampleID generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks a sampleID generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the IBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.7 The IMOD Subchunk

The IMOD subchunk is a required subchunk listing all instrument zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The zone's wInstModNdx points to the first modulator for that zone, and the number of modulators present for a zone is determined by the difference between the next higher zone's wInstModNdx and the current zone's wModNdx. A difference of zero indicates there are no modulators in this zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator within the same zone, the first modulator will be ignored.

Modulators in the IMOD subchunk are absolute. This means that an IMOD modulator replaces, rather than adds to, a default modulator.

In SASBF, no modulators have yet been defined, and the IMOD subchunk will always consist of ten zero valued bytes.

If the IMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.8 The IGEN Subchunk

The IGEN chunk is a required chunk containing a list of zone generators for each instrument zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfInstGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined as in the PGEN zone above.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers of MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The zone's wInstGenNdx points to the first generator for that zone. Unless the zone is a global zone, the last generator in the list is a "sampleID" generator, whose value is a pointer to the sample associated with that zone. If a "key range" generator exists for the zone, it is always the first generator in the list for that zone. If a "velocity range" generator exists for the zone, it will only be preceded by a key range generator. If any generators follow a sampleID generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator within the same zone, the first generator will be ignored.

Generators in the IGEN subchunk are absolute in nature. This means that an IGEN generator replaces, rather than adds to, the default value for the generator.

If the IGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in a sampleID generator, the zone should be ignored. If the sampleID generator value is equal to or greater than the terminal sampleID, the bitstream element should be rejected as structurally unsound.

5.9.2.4.9 The SHDR Subchunk

The SHDR chunk is a required subchunk listing all samples within the smpl subchunk and any referenced ROM samples. It is always a multiple of forty-six bytes in length, and contains one record for each sample plus a terminal record according to the structure:

```
struct sfSample
{
       CHAR achSampleName[20];
       DWORD dwStart;
       DWORD dwEnd;
       DWORD dwStartloop;
       DWORD dwEndloop;
       DWORD dwSampleRate;
       BYTE byOriginalPitch;
       CHAR chPitchCorrection;
       WORD wSampleLink;
       SFSampleLink sfSampleType;
};
```

The ASCII character field achSampleName contains the name of the sample expressed in ASCII, with unused terminal characters filled with zero valued bytes. Sample names are case-sensitive. A unique name should always be assigned to each sample in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of samples with identical names, the samples should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The DWORD dwStart contains the index, in sample data points, from the beginning of the sample data field to the first data point of this sample.

The DWORD dwEnd contains the index, in sample data points, from the beginning of the sample data field to the first of the set of 46 zero valued data points following this sample.

The DWORD dwStartloop contains the index, in sample data points, from the beginning of the sample data field to the first data point in the loop of this sample.

The DWORD dwEndloop contains the index, in sample data points, from the beginning of the sample data field to the first data point following the loop of this sample. Note that this is the data point “equivalent to” the first loop data point, and that to produce portable artefact free loops, the eight proximal data points surrounding both the Startloop and Endloop points should be identical.

The values of dwStart, dwEnd, dwStartloop, and dwEndloop must all be within the range of the sample data field included in the SASBF bank or referenced in the sound ROM. Also, to allow a variety of hardware platforms to be able to reproduce the data, the samples have a minimum length of 48 data points, a

minimum loop size of 32 data points and a minimum of 8 valid points prior to dwStartloop and after dwEndloop. Thus dwStart must be less than dwStartloop-7, dwStartloop must be less than dwEndloop-31, and dwEndloop must be less than dwEnd-7. If these constraints are not met, the sound may optionally not be played if the hardware cannot support artefact-free playback for the parameters given.

The DWORD dwSampleRate contains the sample rate, in hertz, at which this sample was acquired or to which it was most recently converted. Values of greater than 50000 or less than 400 may not be reproducible by some hardware platforms and should be avoided. A value of zero is illegal. If an illegal or impractical value is encountered, the nearest practical value should be used.

The BYTE byOriginalPitch contains the MIDI key number of the recorded pitch of the sample. For example, a recording of an instrument playing middle C (261.62 Hz) should receive a value of 60. This value is used as the default “root key” for the sample, so that in the example, a MIDI key-on command for note number 60 would reproduce the sound at its original pitch. For unpitched sounds, a conventional value of 255 should be used. Values between 128 and 254 are illegal. Whenever an illegal value or a value of 255 is encountered, the value 60 should be used.

The CHAR chPitchCorrection contains a pitch correction in cents which should be applied to the sample on playback. The purpose of this field is to compensate for any pitch errors during the sample recording process. The correction value is that of the correction to be applied. For example, if the sound is 4 cents sharp, a correction bringing it 4 cents flat is required; thus the value should be -4.

The value in sfSampleType is an enumeration with eight defined values: monoSample = 1, rightSample = 2, leftSample = 4, linkedSample = 8, RomMonoSample = 32769, RomRightSample = 32770, RomLeftSample = 32772, and RomLinkedSample = 32776. It can be seen that this is encoded such that bit 15 of the 16 bit value is set if the sample is in ROM, and reset if it is included in the SASBF bank. The four LS bits of the word are then exclusively set indicating mono, left, right, or linked.

If the sound is flagged as a ROM sample and no valid “irom” subchunk is included; the bitstream element is structurally defective and should be rejected at load time.

If sfSampleType indicates a mono sample, then wSampleLink is undefined and its value should be conventionally zero, but will be ignored regardless of value. If sfSampleType indicates a left or right sample, then wSampleLink is the sample header index of the associated right or left stereo sample respectively. Both samples should be played entirely synchronously, with their pitch controlled by the right sample’s generators. All non-pitch generators should apply as normal; in particular the panning of the individual samples to left and right should be accomplished via the pan generator. Left-right pairs should always be found within the same instrument. Note also that no instrument should be designed in which it is possible to activate more than one instance of a particular stereo pair. The linked sample type is not currently fully defined in the SASBF specification, but will ultimately support a circularly linked list of samples using wSampleLink. Note that this enumeration is two bytes in length.

The terminal sample record is never referenced, and is conventionally entirely zero with the exception of achSampleName, which should be “EOS” indicating end of samples. All samples present in the smpl subchunk are typically referenced by an instrument, however a bitstream element containing any “orphaned” samples need not be rejected. SASBF applications can optionally ignore or filter out these orphaned samples according to user preference.

If the SHDR subchunk is missing, or its is size is not a multiple of 46 bytes the bitstream element should be rejected as structurally unsound.

5.9.3 Enumerators

5.9.3.1 Generator Enumerators

Subclause 7.1 defines the generator and generator kinds. Subclause 8.4 defines the generator operation model.

5.9.3.1.1 Kinds of Generator Enumerators

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

An Index Generator's amount is an index into another data structure. The only two Index Generators are Instrument and sampleID.

A Range Generator defines a range of note-on parameters outside of which the zone is undefined. Two Range Generators are currently defined, keyRange and velRange.

Substitution Generators are generators which substitute a value for a note-on parameter. Two Substitution Generators are currently defined, overridingKeyNumber and overridingVelocity.

Sample Generators are generators which directly affect a sample's properties. These generators are undefined at the preset level. The currently defined Sample Generators are the eight address offset generators, the sampleModes generator, the Overriding Root Key generator and the Exclusive Class generator.

Value Generators are generators whose value directly affects a signal processing parameter. Most generators are value generators.

5.9.3.1.2 Generator Enumerators Defined

The following is an exhaustive list of SASBF generators and their strict definitions:

0	startAddrsOffset	The offset, in sample data points, beyond the Start sample header parameter to the first sample data point to be played for this instrument. For example, if Start were 7 and startAddrsOffset were 2, the first sample data point played would be sample data point 9.
1	endAddrsOffset	The offset, in sample data points, beyond the End sample header parameter to the last sample data point to be played for this instrument. For example, if End were 17 and endAddrsOffset were -2, the last sample data point played would be sample data point 15.
2	startloopAddrsOffset	The offset, in sample data points, beyond the Startloop sample header parameter to the first sample data point to be repeated in the loop for this instrument. For example, if Startloop were 10 and startloopAddrsOffset were -1, the first repeated loop sample data point would be sample data point 9.
3	endloopAddrsOffset	The offset, in sample data points, beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. For example, if Endloop were 15 and endloopAddrsOffset were 2, sample data point 17 would be considered equivalent to the

		Startloop sample data point, and hence sample data point 16 would effectively precede Startloop during looping.
4	startAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Start sample header parameter and the first sample data point to be played in this instrument. This parameter is added to the startAddrsOffset parameter. For example, if Start were 5, startAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point played would be sample data point 65544.
5	modLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
6	vibLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Vibrato LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
7	modEnvToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence pitch. A positive value indicates an increase in pitch; a negative value indicates a decrease in pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will rise 1 semitone at the envelope peak.
8	initialFilterFc	This is the cutoff and resonant frequency of the lowpass filter in absolute cent units. The lowpass filter is defined as a second order resonant pole pair whose pole frequency in Hz is defined by the Initial Filter Cutoff parameter. When the cutoff frequency exceeds 20kHz and the Q (resonance) of the filter is zero, the filter does not affect the signal.
9	initialFilterQ	This is the height above DC gain in centibels which the filter resonance exhibits at the cutoff frequency. A value of zero or less indicates the filter is not resonant; the gain at the cutoff frequency (pole angle) may be less than zero when zero is specified. The filter gain at DC is also affected by this parameter such that the gain at DC is reduced by half the specified gain. For example, for a value of 100, the filter gain at DC would be 5 dB below unity gain, and the height of the resonant peak would be 10 dB above the DC gain, or 5 dB above unity gain. Note also that if initialFilterQ is set to zero or less and the cutoff frequency exceeds 20 kHz, then the filter response is flat and unity gain.

10	modLfoToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence filter cutoff frequency. A positive number indicates a positive LFO excursion increases cutoff frequency; a negative number indicates a positive excursion decreases cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1200 indicates that the cutoff frequency will first rise 1 octave, then fall one octave.
11	modEnvToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence filter cutoff frequency. A positive number indicates an increase in cutoff frequency; a negative number indicates a decrease in filter cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1000 indicates that the cutoff frequency will rise one octave at the envelope attack peak.
12	endAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the End sample header parameter and the last sample data point to be played in this instrument. This parameter is added to the endAddrsOffset parameter. For example, if End were 65536, startAddrsOffset were -3 and startAddrsCoarseOffset were -1, the last sample data point played would be sample data point 32765.
13	modLfoToVolume	This is the degree, in centibels, to which a full scale excursion of the Modulation LFO will influence volume. A positive number indicates a positive LFO excursion increases volume; a negative number indicates a positive excursion decreases volume. Volume is always modified logarithmically, that is the deviation is in decibels rather than in linear amplitude. For example, a value of 100 indicates that the volume will first rise ten dB, then fall ten dB.
14	unused1	Unused, reserved. Should be ignored if encountered.
15	chorusEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the chorus effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the chorus effects processor.
16	reverbEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the reverb effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the reverb effects processor.

17	pan	This is the degree, in 0.1% units, to which the “dry” audio output of the note is positioned to the left or right output. A value of -50% or less indicates the signal is sent entirely to the left output and not sent to the right output; a value of +50% or more indicates the signal is sent entirely to the right and not sent to the left. A value of zero sends the signal equally to left and right. For example, a value of -250 indicates that the signal is sent at 75% of full level to the left output and 25% of full level to the right output.
18	unused2	Unused, reserved. Should be ignored if encountered.
19	unused3	Unused, reserved. Should be ignored if encountered.
20	unused4	Unused, reserved. Should be ignored if encountered.
21	delayModLFO	This is the delay time, in absolute timecents, from key on until the Modulation LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second and a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
22	freqModLFO	This is the frequency, in absolute cents, of the Modulation LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 MHz would be 1200log2(.01/8.176) = -11610.
23	delayVibLFO	This is the delay time, in absolute timecents, from key on until the Vibrato LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
24	freqVibLFO	This is the frequency, in absolute cents, of the Vibrato LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 mHz would be 1200log2(.01/8.176) = -11610.
25	delayModEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Modulation envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
26	attackModEnv	This is the time, in absolute timecents, from the end of the Modulation Envelope Delay Time until the point at which the Modulation Envelope value reaches its peak. Note that the attack is

		"convex"; the curve is nominally such that when applied to a decibel or semitone parameter, the result is linear in amplitude or Hz respectively. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
27	holdModEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
28	decayModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during decay phase. For the Modulation Envelope, the decay phase linearly ramps toward the sustain level. If the sustain level were zero, the Modulation Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
29	sustainModEnv	This is the decrease in level, expressed in 0.1% units, to which the Modulation Envelope value ramps during the decay phase. For the Modulation Envelope, the sustain level is properly expressed in percent of full scale. Because the volume envelope sustain level is expressed as an attenuation from full scale, the sustain level is analogously expressed as a decrease from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; values above 1000 are to be interpreted as 1000. For example, a sustain level which corresponds to an absolute value 40% of peak would be 600.
30	releaseModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during release phase. For the Modulation Envelope, the release phase linearly ramps toward zero from the current level. If the current level were full scale, the Modulation Envelope Release Time would be the time spent in release phase until zero value were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
31	keynumToModEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave

		causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
32	keynumToModEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
33	delayVolEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Volume envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
34	attackVolEnv	This is the time, in absolute timecents, from the end of the Volume Envelope Delay Time until the point at which the Volume Envelope value reaches its peak. Note that the attack is “convex”; the curve is nominally such that when applied to the decibel volume parameter, the result is linear in amplitude. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
35	holdVolEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the Volume envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
36	decayVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during decay phase. For the Volume Envelope, the decay phase linearly ramps toward the sustain level, causing a constant dB change for each time unit. If the sustain level were -96dB, the Volume Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
37	sustainVolEnv	This is the decrease in level, expressed in centibels, to which the Volume Envelope value ramps during the decay phase. For the

		Volume Envelope, the sustain level is best expressed in centibels of attenuation from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; conventionally 1000 indicates full attenuation. For example, a sustain level which corresponds to an absolute value 12dB below of peak would be 120.
38	releaseVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during release phase. For the Volume Envelope, the release phase linearly ramps toward zero from the current level, causing a constant dB change for each time unit. If the current level were full scale, the Volume Envelope Release Time would be the time spent in release phase until 96dB attenuation were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
39	keynumToVolEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
40	keynumToVolEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
41	instrument	This is the index into the INST subchunk providing the instrument to be used for the current preset zone. A value of zero indicates the first instrument in the list. The value should never exceed two less than the size of the instrument list. The instrument enumerator is the terminal generator for PGEN zones. As such, it should only appear in the PGEN subchunk, and it must appear as the last generator enumerator in all but the global preset zone.
42	reserved1	Unused, reserved. Should be ignored if encountered.
43	keyRange	This is the minimum and maximum MIDI key number values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid key. The

		keyRange enumerator is optional, but when it does appear, it must be the first generator in the zone generator list.
44	velRange	This is the minimum and maximum MIDI velocity values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid velocity. The velRange enumerator is optional, but when it does appear, it must be preceded only by keyRange in the zone generator list.
45	startloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Startloop sample header parameter and the first sample data point to be repeated in this instrument's loop. This parameter is added to the startloopAddrsOffset parameter. For example, if Startloop were 5, startloopAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point in the loop would be sample data point 65544.
46	keynum	This enumerator forces the MIDI key number to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
47	velocity	This enumerator forces the MIDI velocity to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
48	initialAttenuation	This is the attenuation, in .4 centibel units, by which a note is attenuated below full scale. A value of zero indicates no attenuation; the note will be played at full scale. For example, a value of 60 indicates the note will be played at 2.4 dB below full scale for the note.
49	reserved2	Unused, reserved. Should be ignored if encountered.
50	endloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. This parameter is added to the endloopAddrsOffset parameter. For example, if Endloop were 5, endloopAddrsOffset were 3 and endAddrsCoarseOffset were 2, sample data point 65544 would be considered equivalent to the Startloop sample data point, and hence sample data point 65543 would effectively precede Startloop during looping.
51	coarseTune	This is a pitch offset, in semitones, which should be applied to the note. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Coarse Tune value of -4 would cause the sound to be reproduced four semitones flat.
52	fineTune	This is a pitch offset, in cents, which should be applied to the note. It is additive with coarseTune. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Fine Tuning value of -5 would cause the sound to be reproduced five cents flat.

53	sampleID	This is the index into the SHDR subchunk providing the sample to be used for the current instrument zone. A value of zero indicates the first sample in the list. The value should never exceed two less than the size of the sample list. The sampleID enumerator is the terminal generator for IGEN zones. As such, it should only appear in the IGEN subchunk, and it must appear as the last generator enumerator in all but the global zone.
54	sampleModes	This enumerator indicates a value which gives a variety of Boolean flags describing the sample for the current instrument zone. The sampleModes should only appear in the IGEN subchunk, and should not appear in the global zone. The two LS bits of the value indicate the type of loop in the sample: 0 indicates a sound reproduced with no loop, 1 indicates a sound which loops continuously, 2 is unused but should be interpreted as indicating no loop, and 3 indicates a sound which loops for the duration of key depression then proceeds to play the remainder of the sample.
55	reserved3	Unused, reserved. Should be ignored if encountered.
56	scaleTuning	This parameter represents the degree to which MIDI key number influences pitch. A value of zero indicates that MIDI key number has no effect on pitch; a value of 100 represents the usual tempered semitone scale.
57	exclusiveClass	This parameter provides the capability for a key depression in a given instrument to terminate the playback of other instruments. This is particularly useful for percussive instruments such as a hi-hat cymbal. An exclusive class value of zero indicates no exclusive class; no special action is taken. Any other value indicates that when this note is initiated, any other sounding note with the same exclusive class value should be rapidly terminated. The exclusive class generator can only appear at the instrument level. The scope of the exclusive class is the entire preset. In other words, any other instrument zone within the same preset holding a corresponding exclusive class will be terminated.
58	overridingRootKey	This parameter represents the MIDI key number at which the sample is to be played back at its original sample rate. If not present, or if present with a value of -1, then the sample header parameter Original Key is used in its place. If it is present in the range 0-127, then the indicated key number will cause the sample to be played back at its sample header Sample Rate. For example, if the sample were a recording of a piano middle C (Original Key = 60) at a sample rate of 22.050 kHz, and Root Key were set to 69, then playing MIDI key number 69 (A above middle C) would cause a piano note of pitch middle C to be heard.
59	unused5	Unused, reserved. Should be ignored if encountered.
60	endOper	Unused, reserved. Should be ignored if encountered. Unique name provides value to end of defined list.

5.9.3.1.3 Generator Summary

The following tables give the ranges and default values for all SASBF defined generators.

#	Name	Unit	Abs Zero	Min	Min Useful	Max	Max Useful	De-fault	Def Value
0	startAddrsOffset +	smpls	0	0	None	*	*	0	None
1	endAddrsOffset +	smpls	0	*	*	0	None	0	None
2	startloopAddrsOffset +	smpls	0	*	*	*	*	0	None
3	endloopAddrsOffset +	smpls	0	*	*	*	*	0	None
4	startAddrsCoarseOffset +	32k	0	0	None	*	*	0	None
5	modLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
6	vibLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
7	modEnvToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
8	initialFilterFc	cent	8.176 Hz	1500	20 Hz	13500	20 kHz	13500	Open
9	initialFilterQ	cB	0	0	None	960	96 dB	0	None
10	modLfoToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
11	modEnvToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
12	endAddrsCoarseOffset +	32k	0	*	*	0	None	0	None
13	modLfoToVolume	cB fs	0	-960	-96 dB	960	96 dB	0	None
15	chorusEffectsSend	0.1%	0	0	None	1000	100%	0	None
16	reverbEffectsSend	0.1%	0	0	None	1000	100%	0	None
17	pan	0.1%	Cntr	-500	Left	+500	Right	0	Centre
21	delayModLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
22	freqModLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
23	delayVibLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
24	freqVibLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
25	delayModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
26	attackModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
27	holdModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
28	decayModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
29	sustainModEnv	-0.1%	attk peak	0	100%	1000	0%	0	attk pk
30	releaseModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
31	keynumToModEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
32	keynumToModEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
33	delayVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
34	attackVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
35	holdVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
36	decayVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
37	sustainVolEnv	cB attn	attk peak	0	0 dB	1440	144dB	0	attk pk
38	releaseVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
39	keynumToVolEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
40	keynumToVolEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
43	keyRange	MIDI ky#	key# 0	0	lo key	127	hi key	0-127	full kbd
44	velRange	MIDI vel	0	0	min vel	127	max vel	0-127	all vels
45	startloopAddrsCoarseOffset +	smpl	0	*	*	*	*	0	None

46	keynum +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None
47	velocity +	MIDI vel	0	1	min vel	127	max vel	-1	None
48	initialAttenuation	.4 cB	0	0	0 dB	1440	144dB	0	None
50	endloopAddrsCoarseOffset	smpls	0	*	*	*	*	0	None
51	CoarseTune	semitone	0	-120	-10 oct	120	10 oct	0	None
52	fineTune	cent	0	-99	-99cnt	99	99cent	0	None
54	sampleModes +	Bit Flags	Flags	**	**	**	**	0	No Loop
56	scaleTuning	cent/key	0	0	none	1200	oct/ky	100	semi-tone
57	exclusiveClass +	arbitrary #	0	1	--	127	--	0	None
58	overridingRootKey +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None

* Range depends on values of start, loop, and end points in sample header.

** Range has discrete values based on bit flags

\+ This generator is only valid at the instrument level.

5.9.3.2 Default Modulators

The “default” modulators are described below.

5.9.3.2.1 MIDI Key Velocity to Initial Attenuation

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.2 MIDI Key Velocity to Filter Cutoff

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is -2400 Cents.

The product of these values is added to the Initial Filter Cutoff generator summing node.

5.9.3.2.3 MIDI Channel Pressure to Vibrato LFO Pitch Depth

The MIDI Channel Pressure data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 50 cents per max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.4 MIDI Continuous Controller 1 to Vibrato LFO Pitch Depth

The MIDI Continuous Controller 1 data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. The MIDI Continuous Controller 33 data value may be optionally used for increased resolution of the controller input.

There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 50 cents/max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.5 MIDI Continuous Controller 7 to Initial Attenuation

The MIDI Continuous Controller 7 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.6 MIDI Continuous Controller 10 to Pan Position

The MIDI Continuous Controller 10 data value is used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 127 is mapped to 63/64 and all other values are mapped between -1 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 1000 tenths of a percent panned-right.

The product of these values is added to the Pan generator summing node.

5.9.3.2.7 MIDI Continuous Controller 11 to Initial Attenuation

The MIDI Continuous Controller 11 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.8 MIDI Continuous Controller 91 to Reverb Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added reverb send.

The product of these values is added to the Reverb Send generator summing node.

5.9.3.2.9 MIDI Continuous Controller 93 to Chorus Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added chorus send.

The product of these values is added to the Chorus Send generator summing node.

5.9.3.2.10 MIDI Pitch Wheel to Initial Pitch Controlled by MIDI Pitch Wheel Sensitivity

The MIDI Pitch Wheel data values are used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 16383 is mapped to 8191/8192 and all other values are mapped between -1 and 8191/8192 in a linear fashion.

The MIDI Pitch Wheel Sensitivity data values are used as a secondary source. This source is Positive Unipolar, thus an input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion.

The amount of this modulator is 12700 Cents.

The product of these values is added to the Initial Pitch generator summing node.

5.9.3.3 Precedence and Absolute and Relative values.

Most SASBF generators are available at both the Instrument and Preset Levels, as well as having a default value. Generators at the Instrument Level are considered “absolute” and determine an actual physical value for the associated synthesis parameter, which is used instead of the default. For example, a value of 1200 for the attackVolEnv generator would produce an absolute time of 1200 timecents or 2 seconds of attack time for the volume envelope, instead of the default value of -12000 timecents or 1 msec.

Generators at the Preset Level are instead considered “relative” and additive to all the default or instrument level generators within the Preset Zone. For example, a value of 2400 timecents for the attackVolEnv generator in a preset zone containing an instrument with two zones, one with the default attackVolEnv and one with an absolute attackVolEnv generator value of 1200 timecents would cause the default zone to actually have a value of -9600 timecents or 4 msec, and the other to have a value of 3600 timecents or 8 seconds attack time.

There are some generators which are not available at the Preset Level. These are:

#	Name
0	startAddrsOffset
1	endAddrsOffset
2	startloopAddrsOffset
3	endloopAddrsOffset
4	startAddrsCoarseOffset
12	endAddrsCoarseOffset
45	startloopAddrsCoarseOffset
46	keynum
47	velocity
50	endloopAddrsCoarseOffset
54	sampleModes

57	exclusiveClass
58	overridingRootKey

If these generators are encountered in the Preset Level, they should be ignored.

The effect of modulators on a given destination is always relative to the generator value at the Instrument level. However modulators may supersede or add to other modulators depending on their position within the hierarchy. Please see Subclause 8.4 for details on the Modulator implementation and the hierarchical details.

5.9.4 Parameters and Synthesis Model

The SASBF standard has been established with the intent of providing support for an expanding base of wavetable based synthesis models. The model supported by the SASBF specification originates with the EMU8000 wavetable synthesiser chip. The description below of the underlying synthesis model and the associated parameters are provided to allow mapping of this synthesis model onto other hardware platforms.

5.9.4.1 Synthesis Model

The SASBF specification Synthesis Model comprises a wavetable oscillator, a dynamic lowpass filter, an enveloping amplifier, and programmable sends to pan, reverb, and chorus effects units. An underlying modulation engine comprises two low frequency oscillators (LFOs) and two envelope generators with appropriate routing amplifiers.

5.9.4.1.1 Wavetable Oscillator/Interpolator

The SASBF specification wavetable oscillator model is capable of playing back a sample at an arbitrary sampling rate with an arbitrary pitch shift. In practice, the upward pitch shift (downward sample rate conversion) will be limited to a maximum value, typically at least two octaves. The pitch is described in terms of an initial pitch shift which is based on the sample's sampling rate, the root key at which the sample should be unshifted on the keyboard, the coarse, fine, and correction tunings, the effective MIDI key number, and the keyboard scale factor. All modulations in pitch are in octaves, semitones, and cents.

In general, interpolators have a symmetric impulse response, and thus a linear phase characteristic. The interpolation filter's magnitude response can be characterised by three regions - the pass band, the transition band, and the stop band.

The interpolation filter's pass band is the portion of its response which corresponds to the frequencies of the signal stored in waveform memory from DC to that signal's Nyquist frequency

The interpolation filter's transition band is the portion of its response which corresponds to the first image of the signal stored in waveform memory, that is from that signal's Nyquist frequency back to DC.

The interpolation filter's stop band is the portion of its response which corresponds to the remaining images above the transition band to the Nyquist frequency of the upsampled signal.

The guardband will be assumed to begin at 5/6 of the Nyquist frequency, as is the case for a 20 kHz bandwidth in a 48 kHz sample rate system.

Due to poor aliasing rejection in the stopband, linear interpolation does not satisfy this specification.

The specification constrains the Fourier transform of the impulse response of the interpolator. The impulse response is taken with a downward pitch shift of eight octaves. This gives a response which has been upsampled by a factor of 2**8 or 256. In other words, a frequency of the impulse response's Nyquist

frequency divided by 256 gives the filter frequency corresponding to the waveform memory's Nyquist frequency. In the specification below, Fn represents this waveform memory Nyquist frequency.

(All responses measured with downward pitch shift of 8 octaves.)	
Passband Response:	
Ripple:	No more than +/- 0.5 dB
Roll-off:	Minimal, but not to exceed 6 dB at 83.3% Fn.
Transition Band Response:	
Roll-off:	Monotonic and as rapid as possible to at least -80 dB attenuation
Return Lobes:	None above -80 dB
Attenuation:	Greater than 80 dB for all frequencies DC to at least 2% Fn in the first lobe.
Stop Band Response:	
Attenuation:	Greater than 90 dB for all frequencies DC to at least 1% Fn
	Greater than 80 dB for all frequencies DC to at least 20%Fn
	Greater than 60 dB for all frequencies

5.9.4.1.2 Sample Looping

The wavetable oscillator is playing a digital sample which is described in terms of a start point, end point, and two points describing a loop. The sound can be flagged as unlooped, in which case the loop points are ignored. If the sound is looped, it can be played in two ways. If it is flagged as "loop during release", the sound is played from the start point through the loop, and loops until the note becomes inaudible. If not, the sound is played from the start point through the loop, and loops until the key is released. At this point, the next time the loop end point is reached, the sound continues through the loop end point and plays until the end point is reached, at which time audio is terminated.

5.9.4.1.3 Lowpass Filter

The synthesis model contains a resonant lowpass filter, which is characterised by a dynamic cutoff frequency and a fixed resonance (Q).

The filter is idealised at zero resonance as having a flat passband to the cutoff frequency, then a rolloff at 12 dB per octave above that frequency. The resonance, when non-zero, comprises a peak at the cutoff frequency, superimposed on the above response. The resonance is measured as a dB ratio of the resonant peak to the DC gain. The DC gain at any resonance is half of the resonance value below the DC gain at zero resonance; hence the peak height is half the resonance value above DC gain at zero resonance.

All modulations in cutoff frequency are in cents.

Topology: 2nd order AR lowpass filter or equivalent. Filter coefficients are updated in a smooth manner at the sample rate.

Noise and Distortion: Less than 0.003% of full scale on any sinusoidal input for any parameter setting.

Control Parameters: Cutoff Frequency and Resonance.

Resonance Parameter: Specifies the ratio in decibels of the gain of a resonant peak to the gain at DC, when the filter cutoff frequency is set to 1.5 kHz and the modulation is zero. The DC gain of a filter is reduced from the DC gain of a filter with zero resonance by half the resonance value. The resonance parameter will remain fixed throughout the sounding of a musical note.

For a specified resonance, the actual gain ratio should be accurate within +/- 1 dB, and the DC gain accurate within +/-0.5 dB. The ratio of the gain at the resonant peak to the DC gain for all cutoff frequency values shall not deviate by more than 2dB per octave deviation from 1.5 kHz. Nominal gain at DC for a minimum resonance parameter is unity gain.

Cutoff Frequency Parameter: Specifies the frequency at which the filter achieves exactly 3dB of attenuation. The Cutoff Frequency is computed by the combination of the Initial Cutoff Frequency, specified in Hz, and the modulation, specified in semitones and fractions thereof. The Initial Cutoff Frequency is fixed throughout the duration of the a musical note; the modulation can vary at the sample rate. Within the region from 200 Hz to 6.4 kHz, the cutoff frequency parameter accuracy will be +/- 2 semitones.

Frequency Magnitude Response: When the cutoff frequency is at its maximum value and the resonance is at zero, the filter must pass audio without alteration. The filter must support cutoff frequencies down to 200 Hz. There must be a minimum of 2048 distinct cutoff frequencies per octave within the region from 200 Hz to 6.4 kHz, with a worst case spacing between distinct frequencies of 0.01 semitones.

5.9.4.1.4 Final Gain Amplifier

The final gain amplifier is a multiplier on the filter output, which is controlled by an initial gain in dB. This is added to the volume envelope. Additional modulation can also be added. The gain is always specified in dB.

5.9.4.1.5 Effects Sends

The output of the final gain amplifier can be routed into the effects unit. This unit causes the sound to be located (panned) in the stereo field, and a degree of reverberation and chorus to be added. The pan is specified in terms of percentage left and right, which also could be considered as an azimuth angle. The reverb and chorus sends are specified as a percentage of the signal amplitude to be sent to these units, from 0% to 100%.

5.9.4.1.6 Low Frequency Oscillators

The synthesis model provides for two low frequency oscillators (LFOs) for modulating pitch, filter cutoff, and amplitude. The “vibrato” LFO is only capable of modulating pitch. The “modulation” LFO can modulate any of the three parameters.

An LFO is defined as having a delay period during which its value remains zero, followed by a triangular waveshape ramping linearly to positive one, then downward to negative 1, then upward again to positive one, etc.

Each parameter can be modulated to a varying degree, either positively or negatively, by the associated LFO. Modulations of pitch and cutoff are in octaves, semitones, and cents, while modulations of amplitude are in dB. The degree of modulation is specified in cents or dB for the full scale positive LFO excursion.

5.9.4.1.7 Envelope Generators

The synthesis model provides for two envelope generators. The volume envelope generator controls the final gain amplifier and hence determines the volume contour of the sound. The modulation envelope can control pitch and/or filter cutoff.

An envelope generates a control signal in six phases. When key-on occurs, a delay period begins during which the envelope value is zero. The envelope then rises in a convex curve to a value of one during the attack phase. When a value of one is reached, the envelope enters a hold phase during which it remains at one. When the hold phase ends, the envelope enters a decay phase during which its value decreases linearly to a sustain level. When the sustain level is reached, the envelope enters sustain phase, during which the envelope stays at the sustain level. Whenever a key-off occurs, the envelope immediately enters a release phase during which the value linearly ramps from the current value to zero. When zero is reached, the envelope value remains at zero.

Modulation of pitch and filter cutoff are in octaves, semitones, and cents. These parameters can be modulated to varying degree, either positively or negatively, by the modulation envelope. The degree of modulation is specified in cents for the full scale attack peak.

The volume envelope operates in dB, with the attack peak providing a full scale output, appropriately scaled by the initial volume. The zero value, however, is actually zero gain. When 96 dB of attenuation is reached in the final gain amplifier, an abrupt jump to zero gain (infinite dB of attenuation) occurs. In a 16-bit system, this jump is inaudible.

5.9.4.1.8 Modulation Interconnection Summary

The following diagram shows the interconnections expressed in the SASBF specification synthesis model:

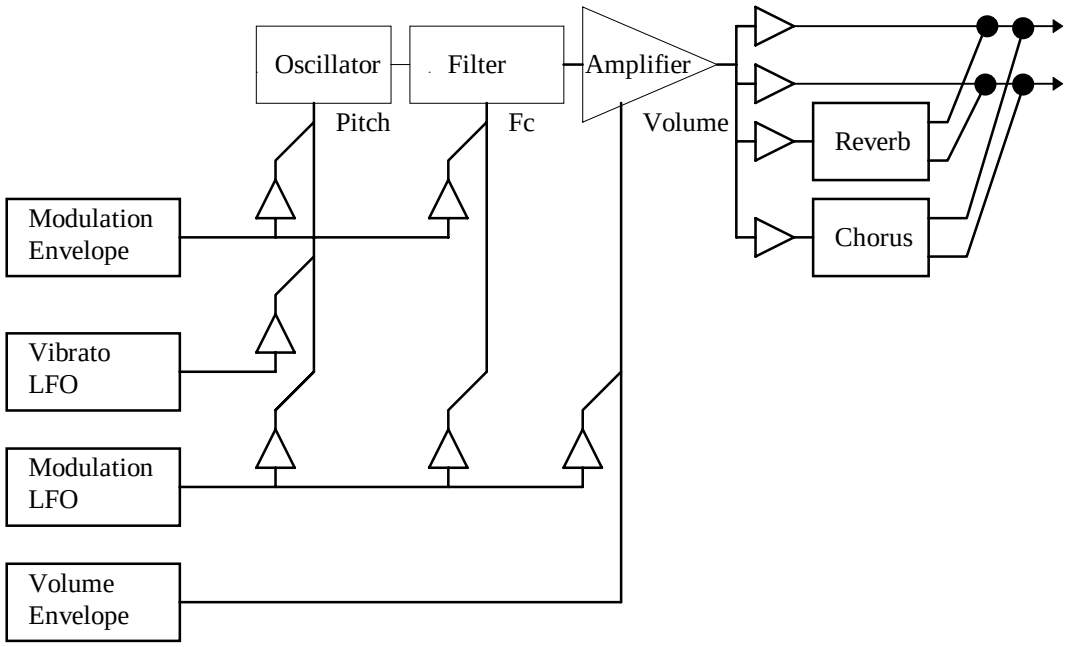


Figure 1: Generator Based Modulation Structure

5.9.4.2 MIDI Functions

The response to certain MIDI commands is defined within the MIDI specification, and therefore not considered to be part of the SASBF specification. These MIDI commands may not be used as sources for the Modulator implementation.

For completeness, the expected responses are given here.

MIDI CC0 Bank Select - When received, the following program change should select the MIDI program in this bank value instead of the default bank of 0.

MIDI CC6 - Data Entry MSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism depending on the Data Entry mode.

MIDI CC32 Bank Select LSB - When received, may behave in conjunction with CC0 Bank Select to provide a total of 16384 possible MIDI banks of programs.

MIDI CC38 Data Entry LSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism, depending on the Data Entry mode.

MIDI CC64 Sustain - ACTIVE when greater than or equal to 64. When the sustain function is active, all notes in the key-on state remain in the key-on state regardless of whether a key-off command for the note arrives. The key-off commands are stored, and when sustain becomes inactive, all stored key-off commands are executed.

MIDI CC66 Soft - ACTIVE when greater than or equal to 64. When active, all new key-ons are modulated in such a way to make the note sound “soft.” This typically affects initial attenuation and filter cutoff in a pre-defined manner.

MIDI CC67 Sostenuto - ACTIVE when greater than or equal to 64. When sostenuto becomes active, all notes currently in the key-on state remain in the key-on state until the sostenuto becomes inactive. All other notes behave normally. Notes maintained by sostenuto in key-on state remain in key-on state even if sustain is switched on and off.

MIDI CC98 NRPN LSB - When received, should be processed by the NRPN implementation mechanism.

MIDI CC99 NRPN MSB - When received, should put the synthesiser in NRPN Data Entry mode and then should be processed by the NRPN implementation mechanism.

MIDI CC100 RPN LSB - When received, should be processed by the RPN implementation mechanism.

MIDI CC101 RPN MSB - When received, should put the synthesiser in RPN Data Entry mode and then should be processed by the RPN implementation mechanism.

MIDI CC120 All Sound Off - When received with any data value, all notes playing in the key-on state bypass the release phase and are shut off, regardless of the sustain or sostenuto positions.

MIDI CC121 Reset All Controllers - Defined as Reset All Controllers as defined by the MIDI specification.

MIDI CC123 All Notes Off - When received with any data value, all notes playing in the key-on state immediately enter release phase, pending their status in SUSTAIN or SOSTENUTO state.

5.9.4.3 Parameter Units

The units with which SASBF generators are described are all well defined. The strict definitions appear below:

ABSOLUTE SAMPLE DATA POINTS - A numeric index of 16 bit sample data point words as stored in ROM or supplied in the smpl-ck, indexing the first sample data point word of memory or the chunk as zero.

RELATIVE SAMPLE DATA POINTS - A count of 16 bit sample data point words based on an absolute sample data point reference. A negative value implies a relative count toward the beginning of the data.

ABSOLUTE SEMITONES - An absolute logarithmic measure of frequency based on a reference of MIDI key numbers. A semitone is 1/12 of an octave, and value 69 is 440 Hz (A-440). Negative values and values above 127 are allowed.

RELATIVE SEMITONES - A relative logarithmic measure of frequency ratio based on units of 1/12 of an octave, which is the twelfth root of two, approximately 1.059463094.

ABSOLUTE CENTS - An absolute logarithmic measure of frequency based on a reference of MIDI key number scaled by 100. A cent is 1/1200 of an octave, and value 6900 is 440 Hz (A-440). Negative values and values above 12700 are allowed.

RELATIVE CENTS - A relative logarithmic measure of frequency ratio based on units of 1/1200 of an octave, which is the twelve hundredth root of two, approximately 1.000577790.

ABSOLUTE CENTIBELS - An absolute measure of the attenuation of a signal, based on a reference of zero being no attenuation. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

RELATIVE CENTIBELS - A relative measure of the attenuation of a signal. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

ABSOLUTE TIMECENTS - An absolute measure of time, based on a reference of zero being one second. A timecent represents a ratio in time of the twelve hundredth root of two, approximately 1.011579454.

RELATIVE TIMECENTS - A relative measure of time ratio, based on a unit size of the twelve hundredth root of two, approximately 1.011579454.

ABSOLUTE PERCENT - An absolute measure of gain, based on a reference of unity. In SASBF, absolute percent is measured in 0.1% units, so a value of zero is 0% and a value of 1000 is 100%.

RELATIVE PERCENT - A relative measure of gain difference. In SASBF, relative percent is measured in 0.1% units. When the gain goes below zero, zero is assumed; when the gain exceeds 100%, 100% is used.

5.9.4.4 The SASBF Generator Model

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

In case it is not clear in the general description of the SASBF hierarchy, the following is the precedence of SASBF generator in the hierarchy.

- A ‘generator’ sets or offsets the value of a destination or a synthesis parameter. In exception cases, it sets ranges (Range Generators), or sets values and never offsets values (Index Generators, Sample Generators, and Substitution Generators).
- A generator is defined as identical to another generator if its generator operator is the same in both generators.
- A generator in a global instrument zone which is identical to a default generator supersedes or replaces the default generator.
- A generator in a local instrument zone which is identical to a default generator or to a generator in a global instrument zone supersedes or replaces that generator.
- Points below (until noted) apply to Value Generators ONLY.
- A generator at the preset level adds to a generator at the instrument level if both generators are identical.
- A generator in a global preset zone which is identical to a default generator or to a generator in an instrument adds to that generator.
- A generator in a global preset zone which is not identical to a default generator and is not identical to a generator in an instrument has its effect added to the given synthesis parameter.
- A generator in a local preset zone which is identical to a generator in a global preset zone supersedes or replaces that generator in the global preset zone. That generator then has its effects added to the destination summing node of all zones in the given instrument.
- A generator in a local preset zone which is not identical to a default generator or a generator in a global preset zone has its effects added to the destination summing node of all zones in the given instrument.
  If the generator operator is a Range Generator, the generator values are NOT ADDED to those

in the instrument level, rather they serve as an intersection filter to those key number or velocity ranges in the instrument which is used in the preset zone.

- If the generator operator is a Substitution Generator or a Sample Generator, they are illegal at the preset level. The only Index Generator legal at the Preset Level is ‘instrumentID’, whereas the only Index Generator legal at the Instrument Level is ‘sampleID’

5.9.4.5 The SASBF Modulator Controller Model

SASBF Modulators are used to allow real-time control over the sound in sound designer programmable manner. Each instance of a SASBF modulator structure defines a real-time perceptually additive effect to be applied to a given destination or synthesiser parameter.

Modulators provide future extensibility to the SASBF standard. They are not used in this version.

5.9.5 Error Handling

5.9.5.1 Structural Errors

Structural Errors are errors which are determined from the implicit redundancy of the SASBF RIFF bitstream element structure, and indicate that the structure is not intact. Examples are incorrect lengths for the chunks or subchunks, pointers out of valid range, or missing required chunks or subchunks for which no error correction procedure exists.

In all cases, bitstream elements should be checked for structural errors at load time, and if any are found, the bitstream elements should be rejected. Separate tools or options can be used to “repair” structurally defective bitstream elements, but these tools should validate that the reconstructed bitstream element is not only a valid SASBF bank but also complies with the intended timbral results in all cases.

5.9.5.2 Unknown Chunks

In parsing the RIFF structure, unknown but well formed chunks or subchunks may be encountered. Unknown chunks within the INFO-list chunk should simply be ignored. Other unknown chunks or subchunks are illegal and should be treated as structural errors.

5.9.5.3 Unknown Enumerators

Unknown enumerators may be encountered in Generators, Modulator Sources, or Transforms. This is to be expected if the ifil field exceeds the specification to which the application was written. Even if unexpected, unknown enumerators should simply cause the associated Generator or Modulator to be ignored.

5.9.5.4 Illegal Parameter Values

Some SASBF parameters are defined for only a limited range of the possible values which can be expressed in their field. If the value of the field is not in the defined range, the parameter has an illegal value.

Illegal values for may be detected either at load or at run time. If detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the default value for the parameter should be used if the parameter is required, or the entire Generator or Modulator ignored if it is optional. Certain parameters may have more specific procedures for illegal values as expressed elsewhere in this specification.

5.9.5.5 Out-of-range Values

SASBF parameters have a specified minimum and useful range the span the perceptually relevant values for the associated sonic property. When the parameter value is exceeds this useful range, the parameter is said to have an out of range value.

Out of range values can result from two distinct causes. An out of range value can be actually present as a SASBF generator value, or the out of range value can be the result of the summation of instrument and preset values.

Out of range values should be handled by substituting the nearest perceptually relevant or realisable value. SASBF banks should not be created with out of range values in the instrument generators. While it is acceptable practice to create SASBF banks which produce out of range values as a result of summation, it is undesirable and should be avoided where practical.

5.9.5.6 Missing Required Parameter or Terminator

Certain parameters and terminators are required by the SASBF specification. If these are missing, the bitstream element is technically not within specification. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the instrument or zone for which the required parameter is missing should simply be ignored. If this causes no sound, the corresponding key-on event is ignored.

5.9.5.7 Illegal enumerator

Certain enumerators are illegal in certain contexts. For example, key and velocity ranges must be the first generators in a zone, instruments are not allowed in instrument zones, and sampleIDs are not allowed in preset zones. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the enumerator should simply be ignored.

5.9.6 Profile 2 (Sample Bank and MIDI decoding)

This Subclause describes the decoding process in which a bitstream conforming to Profile 2 is converted into sound. Profile 2 supports the Structured Audio Sample Bank Format and the General MIDI Format (Profile 1) for sound description. Profile 2 supports the MIDI Protocol and the Standard MIDI File Format for score specification.

5.9.6.1 Stream information header

The SASBF bitstream element is part of the bitstream header. Score elements in the form of MIDI files may be included in the bitstream header.

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class `SA_streaminfo` provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- MIDI file: The events in the MIDI file shall be time ordered, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed. The sample bank requires no processing for this preparation. Processing may be used to improve the computational

efficiency of a decoder. Banks shall be assigned consecutive increasing bank ID numbers in the order of the banks' position in the bitstream. The first bank in the bitstream shall be numbered 0 unless a bank resident in the terminal is being used. In that case, the resident bank shall be numbered 0, an other banks shall be numbered proceeding from there.

5.9.6.2 Bitstream data and sound creation

The MIDI Note On message is defined in the MIDI specification. When the SASBF decoder receives a Note On message, a voice shall be instantiated. Synthesis parameters for the voice shall be determined by the data in the sample bank containing the preset corresponding to the MIDI channel of the Note On message. The number of wavetable oscillators that are used by the voice is defined by the sample bank. Each required oscillator shall have the state variables required for synthesis allocated to it. The state variables shall be initialised according to the preset data from the sample bank.

The MIDI Program Change message is defined in the MIDI specification. When the SASBF decoder receives a Program Change message, an assignment shall be made in the scheduler between the specified MIDI channel and the specified preset. Until this assignment is changed by a subsequent Program Change message, subsequent voice instantiations using the specified MIDI channel shall use sample bank data corresponding to the assigned preset. In a MIDI program change message, if no preset exists in the specified bank with the specified preset number, a replacement preset is used. The replacement preset is the preset with the specified preset number in the bank with the highest bank ID number less than the specified bank ID number which contains a preset with the specified preset number.

The run time scheduler is used to issue MIDI messages to the SASBF decoder. MIDI messages from a MIDI standard file shall be issued by the scheduler when the scheduler clock equals or exceeds the time stamp associated with the message.

5.9.6.3 Conformance

Floating point computation is not required in Profile 2. Audio samples are stored in the bitstream as 16-bit integers. The resolution of the interpolator filter is constrained by the interpolator specification given in Subclause 0.

5.9.7 Profile 4 (Sample Bank decoding in SAOL instruments)

5.9.8 Sample Bank Format Glossary

absolute - Describes a parameter which gives a definitive real-world value. Contrast to relative.

additive - Describes a parameter which is to be numerically added to another parameter.

articulation - The process of modulation of amplitude, pitch, and timbre to produce an expressive musical note.

attack - That phase of an envelope or sound during which the amplitude increases from zero to a peak value.

attenuation - A decrease in volume or amplitude of a signal.

bag - A Sample Bank Format data structure element containing a list of zones.

balance - A form of stereo volume control in which both left and right channels are at maximum when the control is centred, and which attenuates only the opposite channel when taken to either extreme.

bank - A collection of presets. See also MIDI bank.

bipolar - In the SASBF standard, said of a modulator source whose minimum is -1 and whose maximum is 1. Contrast “unipolar”

big endian - Refers to the organisation in memory of bytes within a word such that the most significant byte occurs at the lowest address. Contrast “little endian.”

byte - A data structure element of eight bits without definition of meaning to those bits.

BYTE - A data structure element of eight bits which contains an unsigned value from 0 to 255.

case-insensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as identical. Contrast “case-sensitive.”

case-sensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as distinct. Contrast “case-insensitive.”

cent - A unit of pitch ratio corresponding to the twelve hundredth root of two, or one hundredth of a semitone, approximately 1.000577790.

centibel - A unit of amplitude ratio corresponding to the two hundredth root of ten, or one tenth of a decibel, approximately 1.011579454.

CHAR - A data structure of eight bits which contains a signed value from -128 to +127.

chorus - An effects processing algorithm which involves cyclically shifting the pitch of a signal and remixing it with itself to produce a time varying comb filter, giving a perception of motion and fullness to the resulting sound.

chunk - The top-level division of a RIFF file.

convex - A curve which is bowed in such a way that it is steeper on its lower portion.

concave - (1) A curve which is bowed in such a way that it is steeper on its upper portion. (2) In the SASBF standard, said of a modulator source whose shape is that of the amplitude squared characteristic. Contrast with “convex” and “linear.”

cutoff frequency - The frequency of a filter function at which the attenuation reaches a specified value.

data points - The individual values comprising a sample. Sometimes also called sample points. Contrast “sample.”

decay - The portion of an envelope or sound during which the amplitude declines from a peak to steady state value.

decibel - A unit of amplitude ratio corresponding to the twentieth root of ten, approximately 1.122018454.

delay - The portion of an envelope or LFO function which elapses from a key-on event until the amplitude becomes non-zero.

destination - The generator to which a modulator is applied.

DC gain - The degree of amplification or attenuation a system presents to a static or zero frequency signal.

digital audio - Audio represented as a sequence of quantised values spaced evenly over time. The values are called “sample data points.”

doubleword - A data structure element of 32 bits without definition of meaning to those bits.

downloadable - Said of samples which are loaded from a file into RAM, in contrast to samples which are maintained in ROM.

dry - Refers to audio which has not received any effects processing such as reverb or chorus.

DWORD - A data structure of 32 bits which contains an unsigned value from zero to 4,294,967,295.

envelope - A time varying signal which typically controls the pitch, volume, and/or filter cutoff frequency of a note, and comprises multiple phases including attack, decay, sustain, and release.

enumerated - Said of a data element whose symbols correspond to particular assigned functions.

flat - A. Said of a tone that is lower in pitch than another reference tone. B. Said of a frequency response that does not deviate significantly from a single fixed gain over the audio range.

generator - In the SASBF standard, a parameter which directly affects sound reproduction. Contrast with “modulator.”

global - Refers to parameters which affect all associated structures. See “global zone.”

global zone - A zone whose generators and modulators affect all other zones within the object.

header - A data structure element which describes several aspects of a SASBF element.

instrument - In the SASBF standard, a collection of zones which represents the sound of a single musical instrument or sound effect set.

instrument zone - A sample and associated articulation data defined to play over certain key numbers and velocities.

interpolator - A circuit or algorithm which computes intermediate points between existing sample data points. This is of particular use in the pitch shifting operation of a wavetable synthesiser, in which these intermediate points represent the output samples of the waveform at the desired pitch transposition.

key number - See MIDI key number.

LFO - Acronym for Low Frequency Oscillator. A slow periodic modulation source.

linear - In the SASBF standard, said of a modulator source whose shape is that of a straight line. Contrast with “concave.”

linear coding - The most common method of encoding amplitudes in digital audio in which each step is of equal size.

little endian - A method of ordering bytes within larger words in memory in which the least significant byte is at the lowest address. Contrast “big endian.”

loop - In wavetable synthesis, a portion of a sample which is repeated many times to increase the duration of the resulting sound.

loop points - The sample data points at which a loop begins and ends.

lowpass - Said of a filter which attenuates high frequencies but does not attenuate low frequencies.

modulator - In the SASBF standard, a parameter which routes an external controller to dynamically alter the setting of a “generator.” Contrast with “generator.”

monotonic - Continuously increasing or decreasing. Said of a sequence which never reverses direction.

MIDI - Acronym for Musical Instrument Digital Interface. The standard protocol for sending performance information to a musical synthesiser.

MIDI bank - A group of up to 128 presets selected by a MIDI “change bank” command.

MIDI continuous controller - A construct in the MIDI protocol.

MIDI key number - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the key of the musical instrument keyboard to which the command refers.

MIDI pitch bend - A special MIDI construct akin to the MIDI continuous controllers which controls the real-time value of the pitch of all notes played in a MIDI channel.

MIDI preset - A “preset” selected to be active in a particular MIDI channel by a MIDI “change preset” command.

MIDI velocity - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the speed with which the key was pressed or released.

modulator - In the SASBF standard, a set of parameters which affect a particular generator. Contrast with “generator.”

mono - Short for “monophonic.” Indicates a sound comprising only one channel or waveform. Contrast with “stereo.”

negative - In the SASBF standard, said of a modulator which has a negative sloping characteristic. Contrast with “positive.”

octave - A factor of two in ratio, typically applied to pitch or frequency.

orphan - Said of a data structure which under normal circumstances is referenced by a higher level, but in this particular instance is no longer linked. Specifically, it is an instrument which is not referenced by any preset zone, or a sample which is not referenced by any instrument zone.

oscillator - In wavetable synthesis, the wavetable interpolator is considered an oscillator.

pan - Short for “panorama.” This is the control of the apparent azimuth of a sound source over 180 degrees from left to right. It is generally implemented by varying the volume at the left and right speakers.

pitch - The perceived value of frequency. Generally can be used interchangeably with frequency.

pitch shift - A change in pitch. Wavetable synthesis relies on interpolators to cause pitch shift in a sample to produce the notes of the scale.

pole - A mathematical term used in filter transform analysis. Traditionally in synthesis, a pole is equated with a rolloff of 6dB per octave, and the rolloff of a filter is specified in “poles.”

positive - In the SASBF standard, said of a modulator source which has a positive sloping characteristic. Contrast “negative.”

preset - A keyboard full of sound. Typically the collection of samples and articulation data associated with a particular MIDI preset number.

preset zone - A subset of a preset containing generators, modulators, and an instrument.

proximal - Closest to. Proximal sample data points are the data points closest in either direction to the named point.

Q - A mathematical term used in filter transform analysis. Indicates the degree of resonance of the filter. In synthesis terminology, it is synonymous with resonance.

RAM - Random Access Memory. Conventionally, this term implies read-write memory. Contrast “ROM.”

record - A single instance of a data structure.

relative - Describes a parameter which merely indicates an offset from an otherwise established value. Contrast to absolute.

release - The portion of an envelope or sound during which the amplitude declines from a steady state to zero value or inaudibility.

resonance - Describes the aspect of a filter in which particular frequencies are given significantly more gain than others. The resonance can be measured in dB above the DC gain.

resonant frequency - The frequency at which resonance reaches its maximum.

reverb - Short for reverberation. In synthesis, a synthetic signal processor which adds artificial spaciousness and ambience to a sound.

RIFF - Acronym for Resource Interchange File Format.

ROM - Acronym for Read Only Memory. A memory whose contents are fixed at manufacture, and hence cannot be written by the user. Contrast with RAM.

sample - This term is often used both to indicate a “sample data point” and to indicate a collection of such points comprising a digital audio waveform. The latter meaning is exclusively used in this Subclause (the SASBF specification.)

sample rate - The frequency, in Hertz, at which sample data points are taken when recording a sample.

semitone - A unit of pitch ratio corresponding to the twelfth root of two, or one twelfth of an octave, approximately 1.059463094.

sharp - Said of a tone that is higher in pitch than another reference tone.

SHORT - A data structure element of sixteen bits which contains a signed value from -32,768 to +32,767.

soft - The pedal on a piano, so named because it causes the damper to be lowered in such a way as to soften the timbre and loudness of the notes. In MIDI, continuous controller #66, which behaves in a similar manner.

sostenuto - The pedal on a piano which causes the dampers on all keys depressed to be held until the pedal is released. In MIDI, continuous controller #67, which behaves in a similar manner.

sustain - The pedal on a piano which prevents all dampers on keys as they are depressed from being released. In MIDI, continuous controller #64, which behaves in a similar manner.

source - In a SASBF modulator, the enumerator indicating the particular real-time value which the modulator will transform, scale, and add to the destination generator.

stereo - Literally indicating three dimensions. In this specification, the term is used to mean two channel stereophonic, indicating that the sound is composed of two independent audio channels, dubbed left and right. Contrast monophonic.

subchunk - A division of a RIFF file below that of the chunk.

synthesis engine - The hardware and software associated with the signal processing and modulation path for a particular synthesiser.

synthesiser - A device ideally capable of producing arbitrary musical sound.

terminator - A data structure element indicating the final element in a sequence.

timecent - A unit of duration ratio corresponding to the twelve hundredth root of two, or one twelve hundredth of an octave, approximately 1.000577790.

transform - In a SASBF modulator, the enumerator indicating the particular transfer function through which the source will be passed prior to scaling and addition to the destination generator.

tremolo - A periodic change in amplitude of a sound, typically produced by applying a low frequency oscillator to the final volume amplifier.

triangular - A waveform which ramps upward to a positive limit, then downward at the opposite slope to the symmetrically negative limit periodically.

unipolar - In the SASBF standard, said of a modulator source whose minimum is 0 and whose maximum is 1. Contrast “bipolar.”

velocity - In synthesis, the speed with which a keyboard key is depressed, typically proportionally to the impact delivered by the musician. See also MIDI velocity.

vibrato - A periodic change in the pitch of a sound, typically produced by applying a low frequency oscillator to the oscillator pitch.

volume - The loudness or amplitude of a sound, or the control of this parameter.

wavetable - A music synthesis technique wherein musical sounds are recorded or computed mathematically and stored in a memory, then played back at a variable rate to produce the desired pitch. Additional timbral adjustments are often made to the sound thus produced using amplifiers, filters, and effects processing such as reverb and chorus.

WORD - A data structure of 16 bits which contains an unsigned value from zero to 65,535.

word - A data structure element of 16 bits without definition of meaning to those bits.

zone - An object and associated articulation data defined to play over certain key numbers and velocities.

5.10 MIDI semantics. This profile is used to enable backward-compatibility with existing MIDI content and rendering devices. Implementation-independent sound quality cannot be produced in this profile.

1. Wavetable synthesis. In this profile, only the **midi_file** and **sbf** chunks shall occur in the stream information header, and only the **midi_event** event shall occur in the bitstream data. This profile is used to describe music and sound-effects content in situations which the full flexibility and functionality of SAOL, including 3-D audio, is not required. In this case, the decoding process is described in Subclause 5.9.6 Profile 2 (Sample Bank and MIDI decoding).

4. Standard profile. All bitstream elements and stream information elements may occur.

The decoding process for Profile 4 is described in Subclause 5.3 Decoding process.

5.3 Decoding process

5.3.1 Introduction

This Subclause describes the decoding process, in which a bitstream conforming to Profile 4 is converted into sound. The decoding process for Profile 1 bitstreams is described in Subclause 5.9 Sample Bank syntax and semantics

5.9.1 Introduction

This Subclause describes the operation of the Sample Bank synthesis method for both Profile 2 and Profile 4. In Profile 2, only Sample Bank and MIDI class types shall appear in the bitstream, and this Subclause describes the normative process of generating sound from a Sample Bank bitstream data element and a sequence of MIDI instructions. In Profile 4, Sample Banks are used in the context of a SAOL instrument as described in Subclause 5.4.6.6.8 Output, and this Subclause describes the normative process of generating sound and placing it on three busses, depending on the Sample Bank bitstream data element and the particular call to **sbsynth**.

5.9.2 Elements of bitstream

5.9.2.1 RIFF Structure

5.9.2.1.1 General RIFF File Structure

The RIFF (Resource Interchange File Format) is a tagged file structure developed for multimedia resource files, and is described in some detail in the Microsoft Windows 3.1 SDK Multimedia Programmer's Reference. The Tagged-file structure is useful because it helps prevent compatibility problems which can occur as the file definition changes over time. Because each piece of data in the file is identified by a standard header, an application that does not recognise a given data element can skip over the unknown information. The relevant information is provided in the bitstream description, Subclause 0.

A RIFF file is constructed from a basic building block called a "chunk." In 'C' syntax, a chunk is defined:

```
typedef DWORD FOURCC;      //  Four-character code
typedef struct {
  FOURCC ckID;             //  Chunk ID identifies type of data in the chunk.
  DWORD ckSize;            //  Size of chunk data in bytes, excluding pad byte.
  BYTE ckDATA[ckSize];     //  Actual data + a pad byte if req'd to word align.
};
```

Two types of chunks, the "RIFF" and "LIST" chunks, may contain nested chunks called subchunks as their data.

Within a given level of the hierarchy, the ordering of the chunks is arbitrary. Any chunk with an unknown chunk ID should be ignored.

5.9.2.1.2 The Sample Bank Chunks and Subchunks

A Structured Audio Sample Bank bitstream element comprises three chunks: an INFO-list chunk containing a number of required and optional subchunks describing the bitstream element, its history, and its intended use, an sdta-list chunk comprising a single subchunk containing any referenced digital audio samples, and a pdta-list chunk containing nine subchunks which define the articulation of the digital audio data.

5.9.2.1.3 Redundancy and Error Handling in the RIFF structure

The RIFF bitstream element structure contains redundant information regarding the length of the bitstream element and the length of the chunks and subchunks. This fact enables any reader of a SASBF bitstream element to determine if the bitstream element has been damaged by loss of data.

If any such loss is detected, the SASBF bitstream element is termed “structurally unsound” and in general should be rejected.

5.9.2.2 The INFO-list Chunk

The INFO-list chunk in a SASBF bitstream element contains three mandatory and a variety of optional subchunks as defined below. The INFO-list chunk gives basic information about the SASBF bank contained in the bitstream element.

5.9.2.2.1 The ifil Subchunk

The ifil subchunk is a mandatory subchunk identifying the SASBF specification version level to which the bitstream element complies. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the SASBF specification version. The WORD wMinor contains the value to the right of the decimal point. For example, version 2.11 would be implied if wMajor=2 and wMinor=11.

These values can be used by applications which read SASBF bitstream elements to determine if the format of the bitstream element is usable by the program. Within a fixed wMajor, the only changes to the format will be the addition of Generator, Source and Transform enumerators, and additional info subchunks. These are all defined as being ignored if unknown to the program.

If the ifil subchunk is missing, or its size is not four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.2.2 The isng Subchunk

The isng subchunk is a mandatory subchunk identifying the wavetable sound engine for which the bitstream element was optimised. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even.

The ASCII should be treated as case-sensitive. In other words “engine” is not the same as “ENGINE.”

The isng string may be optionally used by chip drivers to vary their synthesis algorithms to emulate the target sound engine.

If the isng subchunk is missing not terminated in a zero valued byte, or its contents are an unknown sound engine, the field should be ignored.

5.9.2.2.3 The INAM Subchunk

The INAM subchunk is a mandatory subchunk providing the name of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical inam subchunk would be the fourteen bytes representing “General MIDI” as twelve ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “General MIDI” is not the same as “GENERAL MIDI.”

If the inam subchunk is missing, or not terminated in a zero valued byte, the field should be ignored and the user supplied with an appropriate error message if the name is queried. If the bitstream element is re-written, a valid name should be placed in the INAM field.

5.9.2.2.4 The irom Subchunk

The irom subchunk is an optional subchunk identifying a particular wavetable sound data ROM to which any ROM samples refer. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical irom field would be the six bytes representing “1MGM” as four ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “1mgm” is not the same as “1MGM.”

The irom string is used by drivers to verify that the ROM data referenced by the bitstream element is available to the sound engine.

If the irom subchunk is missing, not terminated in a zero valued byte, or its contents are an unknown ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. A bitstream element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.5 The iver Subchunk

The iver subchunk is an optional subchunk identifying the particular wavetable sound data ROM revision to which any ROM samples refer. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the ROM version. The WORD wMinor contains the value to the right of the decimal point. For example, version 1.36 would be implied if wMajor=1 and wMinor=36.

The iver subchunk is used by drivers to verify that the ROM data referenced by the bitstream element is located in the exact locations specified by the sound headers.

If the iver subchunk is missing, not four bytes in length, or its contents indicate an unknown or incorrect ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. Note that for ROM samples to function correctly, both iver and irom must be present and valid. A bitstream

element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.6 The ICRD Subchunk

The ICRD subchunk is an optional subchunk identifying the creation date of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICRD field would be the twelve bytes representing “May 1, 1995” as eleven ASCII characters followed by a zero byte.

Conventionally, the format of the string is “Month Day, Year” where Month is initially capitalised and is the conventional full English spelling of the month, Day is the date in decimal followed by a comma, and Year is the full decimal year. Thus the field should conventionally never be longer than 32 bytes.

The ICRD string is provided for library management purposes.

If the ICRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.7 The IENG Subchunk

The IENG subchunk is an optional subchunk identifying the names of any sound designers or engineers responsible for the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IENG field would be the twelve bytes representing “Tim Swartz” as ten ASCII characters followed by two zero bytes.

The IENG string is provided for library management purposes.

If the IENG subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.8 The IPRD Subchunk

The IPRD subchunk is an optional subchunk identifying any specific product for which the SASBF bank is intended. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IPRD field would be the twelve bytes representing “MPEG SASBF” as ten ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “mpeg sasbf” is not the same as “MPEG SASBF.”

The IPRD string is provided for library management purposes.

If the IPRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.9 The ICOP Subchunk

The ICOP subchunk is an optional subchunk containing any copyright assertion string associated with the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICOP field would be the 38 bytes representing "Copyright (c) 1998 Content Developer" as 36 ASCII characters followed by two zero bytes.

The ICOP string is provided for intellectual property protection and management purposes.

If the ICOP subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.10 The ICMT Subchunk

The ICMT subchunk is an optional subchunk containing any comments associated with the SASBF bank. It contains an ASCII string of 65,536 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICMT field would be the 40 bytes representing "This space unintentionally left blank." as 38 ASCII characters followed by two zero bytes.

The ICMT string is provided for including comments.

If the ICMT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.11 The ISFT Subchunk

The ISFT subchunk is an optional subchunk identifying the SASBF tools used to create and most recently modify the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ISFT field would be the twenty-six bytes representing "Editor 2.00a:Editor 2.00a" as twenty-five ASCII characters followed by a zero byte.

The ASCII should be treated as case-sensitive. In other words "Editor" is not the same as "EDITOR."

Conventionally, the tool name and revision control number are included first for the creating tool and then for the most recent modifying tool. The two strings are separated by a colon. The string should be produced by the creating program with a null modifying tool field (e.g. "Editor 2.00a:), and each time a tool modifies the bank, it should replace the modifying tool field with its own name and revision control number.

The ISFT string is provided primarily for error tracing purposes.

If the ISFT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.3 The sdta-list Chunk

The sdta-list chunk in a SASBF bitstream element contains a single optional smpl subchunk which contains all the sound data associated with the SASBF bank. The smpl subchunk is of arbitrary length, and contains an even number of bytes.

5.9.2.3.1 Sample Data Format in the smpl Subchunk

The smpl subchunk, contains one or more samples of digital audio information in the form of linearly coded sixteen bit, signed, little endian (least significant byte first) words. Each sample is followed by a minimum of forty-six zero valued sample data points. These zero valued data points are necessary to guarantee that any reasonable upward pitch shift using any reasonable interpolator can loop on zero data at the end of the sound.

5.9.2.3.2 Sample Data Looping Rules

Within each sample, one or more loop point pairs may exist. The locations of these points are defined within the pdta-list chunk, but the sample data points themselves must comply with certain practices in order for the loop to be compatible across multiple platforms.

The loops are defined by “equivalent points” in the sample. This means that there are two sample data points which are logically equivalent, and a loop occurs when these points are spliced atop one another. In concept, the loop end point is never actually played during looping; instead the loop start point follows the point just prior to the loop end point. Because of the bandlimited nature of digital audio sampling, an artefact free loop will exhibit virtually identical data surrounding the equivalent points.

In actuality, because of the various interpolation algorithms used by wavetable synthesisers, the data surrounding both the loop start and end points may affect the sound of the loop. Hence both the loop start and end points must be surrounded by continuous audio data. For example, even if the sound is programmed to continue to loop throughout the decay, sample data points must be provided beyond the loop end point. This data will typically be identical to the data at the start of the loop. A minimum of eight valid data points are required to be present before the loop start and after the loop end.

The eight data points (four on each side) surrounding the two equivalent loop points should also be forced to be identical. By forcing the data to be identical, all interpolation algorithms are guaranteed to properly reproduce an artefact-free loop.

5.9.2.4 The pdta-list Chunk

5.9.2.4.1 The PHDR Subchunk

The PHDR subchunk is a required subchunk listing all presets within the SASBF bitstream element. It shall be a multiple of thirty-eight bytes in length, and shall contain a minimum of two records, one record for each preset and one for a terminal record according to the structure:

```
struct sfPresetHeader
{
       CHAR achPresetName[20];
       WORD wPreset;
       WORD wBank;
       WORD wPresetBagNdx;
       DWORD dwLibrary;
       DWORD dwGenre;
       DWORD dwMorphology;
};
```

The ASCII character field achPresetName shall contain the name of the preset expressed in ASCII, with unused terminal characters filled with zero valued bytes. Preset names are case-sensitive. A unique name shall always be assigned to each preset in the SASBF bank.

The WORD wPreset shall contain the MIDI Preset Number and the WORD wBank the MIDI Bank Number which apply to this preset. Note that the presets are not ordered within the SASBF bank. Presets

shall have a unique set of wPreset and wBank numbers. The special case of a General MIDI percussion bank is handled conventionally by a wBank value of 128. If the value in either field is not a valid MIDI value of 0 through 127, or 128 for wBank, the preset cannot be played but shall be maintained in memory for future renumbering.

The WORD wPresetBagNdx is an index to the preset's zone list in the PBAG subchunk. Because the preset zone list is in the same order as the preset header list, the preset bag indices shall be monotonically increasing with increasing preset headers. The size of the PBAG subchunk in bytes shall be equal to four times the terminal preset's wPresetBagNdx plus four. If the preset bag indices are non-monotonic or if the terminal preset's wPresetBagNdx does not match the PBAG subchunk size, the SASBF chunk is structurally defective and shall be rejected at load time. All presets except the terminal preset shall have at least one zone.

The DWORDs dwLibrary, dwGenre and dwMorphology are reserved for future implementation in a preset library management function and should be preserved as read, and created as zero.

The terminal sfPresetHeader record should never be accessed, and exists only to provide a terminal wPresetBagNdx with which to determine the number of zones in the last preset. All other values shall be conventionally zero, with the exception of achPresetName, which may be optionally be "EOP" indicating end of presets.

If the PHDR subchunk is missing, contains fewer than two records, or its size is not a multiple of 38 bytes, the SASBF bitstream element shall be rejected as structurally unsound.

5.9.2.4.2 The PBAG Subchunk

The PBAG subchunk is a required subchunk listing all preset zones within the SASBF bitstream element. It shall be a multiple of four bytes in length, and shall contain one record for each preset zone plus one record for a terminal zone according to the structure:

```
struct sfPresetBag
{
       WORD wGenNdx;
       WORD wModNdx;
};
```

The first zone in a given preset shall be located at that preset's wPresetBagNdx. The number of zones in the preset shall be determined by the difference between the next preset's wPresetBagNdx and the current wPresetBagNdx.

The WORD wGenNdx shall be an index to the preset's zone list of generators in the PGEN subchunk, and the wModNdx shall be an index to its list of modulators in the PMOD subchunk. Because both the generator and modulator lists are in the same order as the preset header and zone lists, these indices will be monotonically increasing with increasing preset zones. The size of the PMOD subchunk in bytes shall be equal to ten times the terminal preset's wModNdx plus ten and the size of the PGEN subchunk in bytes shall be equal to four times the terminal preset's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective PGEN or PMOD subchunks, the bitstream element is structurally defective and shall be rejected at load time.

If a preset has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not an Instrument generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks an Instrument generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the PBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.3 The PMOD Subchunk

The PMOD subchunk is a required subchunk listing all preset zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The preset zone's wModNdx points to the first modulator for that preset zone, and the number of modulators present for a preset zone is determined by the difference between the next higher preset zone's wModNdx and the current preset's wModNdx. A difference of zero indicates there are no modulators in this preset zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator with the same zone, the first modulator will be ignored.

Modulators in the PMOD subchunk act as additively relative modulators with respect to those in the IMOD subchunk. In other words, a PMOD modulator can increase or decrease the amount of an IMOD modulator.

In SASBF, no modulators have yet been defined, and the PMOD subchunk will always consist of ten zero valued bytes.

If the PMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.4 The PGEN Subchunk

The PGEN chunk is a required chunk containing a list of preset zone generators for each preset zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each preset zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined:

```
typedef struct
{
       BYTE byLo;
       BYTE byHi;
} rangesType;

typedef union
{
       rangesType ranges;
       SHORT shAmount;
       WORD wAmount;
} genAmountType;
```

The sfGenOper is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. This value indicates the type of generator being indicated. Note that this enumeration is two bytes in length.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers or MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The preset zone's wGenNdx points to the first generator for that preset zone. Unless the zone is a global zone, the last generator in the list is an "Instrument" generator, whose value is a pointer to the instrument associated with that zone. If a "key range" generator exists for the preset zone, it is always the first generator in the list for that preset zone. If a "velocity range" generator exists for the preset zone, it will only be preceded by a key range generator. If any generators follow an Instrument generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator with the same zone, the first generator will be ignored.

Generators in the PGEN subchunk are applied relative to generators in the IGEN subchunk in an additive manner. In other words, PGEN generators increase or decrease the value of an IGEN generator.

If the PGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in an instrument generator, the zone should be ignored. If the instrument generator value is equal to or greater than the terminal instrument, the bitstream element should be rejected as structurally unsound.

5.9.2.4.5 The INST Subchunk

The inst subchunk is a required subchunk listing all instruments within the SASBF bitstream element. It is always a multiple of twenty-two bytes in length, and contains a minimum of two records, one record for each instrument and one for a terminal record according to the structure:

```
struct sfInst
{
       CHAR achInstName[20];
       WORD wInstBagNdx;
};
```

The ASCII character field achInstName contains the name of the instrument expressed in ASCII, with unused terminal characters filled with zero valued bytes. Instrument names are case-sensitive. A unique name should always be assigned to each instrument in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of instruments with identical names, the instruments should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The WORD wInstBagNdx is an index to the instrument's zone list in the IBAG subchunk. Because the instrument zone list is in the same order as the instrument list, the instrument bag indices will be monotonically increasing with increasing instruments. The size of the IBAG subchunk in bytes will be four greater than four times the terminal (EOI) instrument's wInstBagNdx. If the instrument bag indices are non-monotonic or if the terminal instrument's wInstBagNdx does not match the IBAG subchunk size, the bitstream element is structurally defective and should be rejected at load time. All instruments except the terminal instrument must have at least one zone; any preset with no zones should be ignored.

The terminal sfInst record should never be accessed, and exists only to provide a terminal wInstBagNdx with which to determine the number of zones in the last instrument. All other values are conventionally zero, with the exception of achInstName, which should be "EOI" indicating end of instruments.

If the INST subchunk is missing, contains fewer than two records, or its size is not a multiple of 22 bytes, the bitstream element should be rejected as structurally unsound. All instruments present in the inst subchunk are typically referenced by a preset zone. However, a bitstream element containing any "orphaned" instruments need not be rejected. SASBF applications can optionally ignore or filter out these orphaned instruments based on user preference.

5.9.2.4.6 The IBAG Subchunk

The IBAG subchunk is a required subchunk listing all instrument zones within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one record for each instrument zone plus one record for a terminal zone according to the structure:

```
struct sfInstBag
{
       WORD wInstGenNdx;
       WORD wInstModNdx;
```

```
};
```

The first zone in a given instrument is located at that instrument's wInstBagNdx. The number of zones in the instrument is determined by the difference between the next instrument's wInstBagNdx and the current wInstBagNdx.

The WORD wInstGenNdx is an index to the instrument zone's list of generators in the IGEN subchunk, and the wInstModNdx is an index to its list of modulators in the IMOD subchunk. Because both the generator and modulator lists are in the same order as the instrument and zone lists, these indices will be monotonically increasing with increasing zones. The size of the IMOD subchunk in bytes will be equal to ten times the terminal instrument's wModNdx plus ten and the size of the IGEN subchunk in bytes will be equal to four times the terminal instrument's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective IGEN or IMOD subchunks, the bitstream element is structurally defective and should be rejected at load time.

If an instrument has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not a sampleID generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks a sampleID generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the IBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.7 The IMOD Subchunk

The IMOD subchunk is a required subchunk listing all instrument zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The zone's wInstModNdx points to the first modulator for that zone, and the number of modulators present for a zone is determined by the difference between the next higher zone's wInstModNdx and the current zone's wModNdx. A difference of zero indicates there are no modulators in this zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator within the same zone, the first modulator will be ignored.

Modulators in the IMOD subchunk are absolute. This means that an IMOD modulator replaces, rather than adds to, a default modulator.

In SASBF, no modulators have yet been defined, and the IMOD subchunk will always consist of ten zero valued bytes.

If the IMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.8 The IGEN Subchunk

The IGEN chunk is a required chunk containing a list of zone generators for each instrument zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfInstGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined as in the PGEN zone above.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers of MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The zone's wInstGenNdx points to the first generator for that zone. Unless the zone is a global zone, the last generator in the list is a "sampleID" generator, whose value is a pointer to the sample associated with that zone. If a "key range" generator exists for the zone, it is always the first generator in the list for that zone. If a "velocity range" generator exists for the zone, it will only be preceded by a key range generator. If any generators follow a sampleID generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator within the same zone, the first generator will be ignored.

Generators in the IGEN subchunk are absolute in nature. This means that an IGEN generator replaces, rather than adds to, the default value for the generator.

If the IGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in a sampleID generator, the zone should be ignored. If the sampleID generator value is equal to or greater than the terminal sampleID, the bitstream element should be rejected as structurally unsound.

5.9.2.4.9 The SHDR Subchunk

The SHDR chunk is a required subchunk listing all samples within the smpl subchunk and any referenced ROM samples. It is always a multiple of forty-six bytes in length, and contains one record for each sample plus a terminal record according to the structure:

```
struct sfSample
{
       CHAR achSampleName[20];
       DWORD dwStart;
       DWORD dwEnd;
       DWORD dwStartloop;
       DWORD dwEndloop;
       DWORD dwSampleRate;
       BYTE byOriginalPitch;
       CHAR chPitchCorrection;
       WORD wSampleLink;
       SFSampleLink sfSampleType;
};
```

The ASCII character field achSampleName contains the name of the sample expressed in ASCII, with unused terminal characters filled with zero valued bytes. Sample names are case-sensitive. A unique name should always be assigned to each sample in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of samples with identical names, the samples should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The DWORD dwStart contains the index, in sample data points, from the beginning of the sample data field to the first data point of this sample.

The DWORD dwEnd contains the index, in sample data points, from the beginning of the sample data field to the first of the set of 46 zero valued data points following this sample.

The DWORD dwStartloop contains the index, in sample data points, from the beginning of the sample data field to the first data point in the loop of this sample.

The DWORD dwEndloop contains the index, in sample data points, from the beginning of the sample data field to the first data point following the loop of this sample. Note that this is the data point “equivalent to” the first loop data point, and that to produce portable artefact free loops, the eight proximal data points surrounding both the Startloop and Endloop points should be identical.

The values of dwStart, dwEnd, dwStartloop, and dwEndloop must all be within the range of the sample data field included in the SASBF bank or referenced in the sound ROM. Also, to allow a variety of hardware platforms to be able to reproduce the data, the samples have a minimum length of 48 data points, a

minimum loop size of 32 data points and a minimum of 8 valid points prior to dwStartloop and after dwEndloop. Thus dwStart must be less than dwStartloop-7, dwStartloop must be less than dwEndloop-31, and dwEndloop must be less than dwEnd-7. If these constraints are not met, the sound may optionally not be played if the hardware cannot support artefact-free playback for the parameters given.

The DWORD dwSampleRate contains the sample rate, in hertz, at which this sample was acquired or to which it was most recently converted. Values of greater than 50000 or less than 400 may not be reproducible by some hardware platforms and should be avoided. A value of zero is illegal. If an illegal or impractical value is encountered, the nearest practical value should be used.

The BYTE byOriginalPitch contains the MIDI key number of the recorded pitch of the sample. For example, a recording of an instrument playing middle C (261.62 Hz) should receive a value of 60. This value is used as the default “root key” for the sample, so that in the example, a MIDI key-on command for note number 60 would reproduce the sound at its original pitch. For unpitched sounds, a conventional value of 255 should be used. Values between 128 and 254 are illegal. Whenever an illegal value or a value of 255 is encountered, the value 60 should be used.

The CHAR chPitchCorrection contains a pitch correction in cents which should be applied to the sample on playback. The purpose of this field is to compensate for any pitch errors during the sample recording process. The correction value is that of the correction to be applied. For example, if the sound is 4 cents sharp, a correction bringing it 4 cents flat is required; thus the value should be -4.

The value in sfSampleType is an enumeration with eight defined values: monoSample = 1, rightSample = 2, leftSample = 4, linkedSample = 8, RomMonoSample = 32769, RomRightSample = 32770, RomLeftSample = 32772, and RomLinkedSample = 32776. It can be seen that this is encoded such that bit 15 of the 16 bit value is set if the sample is in ROM, and reset if it is included in the SASBF bank. The four LS bits of the word are then exclusively set indicating mono, left, right, or linked.

If the sound is flagged as a ROM sample and no valid “irom” subchunk is included; the bitstream element is structurally defective and should be rejected at load time.

If sfSampleType indicates a mono sample, then wSampleLink is undefined and its value should be conventionally zero, but will be ignored regardless of value. If sfSampleType indicates a left or right sample, then wSampleLink is the sample header index of the associated right or left stereo sample respectively. Both samples should be played entirely synchronously, with their pitch controlled by the right sample’s generators. All non-pitch generators should apply as normal; in particular the panning of the individual samples to left and right should be accomplished via the pan generator. Left-right pairs should always be found within the same instrument. Note also that no instrument should be designed in which it is possible to activate more than one instance of a particular stereo pair. The linked sample type is not currently fully defined in the SASBF specification, but will ultimately support a circularly linked list of samples using wSampleLink. Note that this enumeration is two bytes in length.

The terminal sample record is never referenced, and is conventionally entirely zero with the exception of achSampleName, which should be “EOS” indicating end of samples. All samples present in the smpl subchunk are typically referenced by an instrument, however a bitstream element containing any “orphaned” samples need not be rejected. SASBF applications can optionally ignore or filter out these orphaned samples according to user preference.

If the SHDR subchunk is missing, or its is size is not a multiple of 46 bytes the bitstream element should be rejected as structurally unsound.

5.9.3 Enumerators

5.9.3.1 Generator Enumerators

Subclause 7.1 defines the generator and generator kinds. Subclause 8.4 defines the generator operation model.

5.9.3.1.1 Kinds of Generator Enumerators

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

An Index Generator's amount is an index into another data structure. The only two Index Generators are Instrument and sampleID.

A Range Generator defines a range of note-on parameters outside of which the zone is undefined. Two Range Generators are currently defined, keyRange and velRange.

Substitution Generators are generators which substitute a value for a note-on parameter. Two Substitution Generators are currently defined, overridingKeyNumber and overridingVelocity.

Sample Generators are generators which directly affect a sample's properties. These generators are undefined at the preset level. The currently defined Sample Generators are the eight address offset generators, the sampleModes generator, the Overriding Root Key generator and the Exclusive Class generator.

Value Generators are generators whose value directly affects a signal processing parameter. Most generators are value generators.

5.9.3.1.2 Generator Enumerators Defined

The following is an exhaustive list of SASBF generators and their strict definitions:

0	startAddrsOffset	The offset, in sample data points, beyond the Start sample header parameter to the first sample data point to be played for this instrument. For example, if Start were 7 and startAddrsOffset were 2, the first sample data point played would be sample data point 9.
1	endAddrsOffset	The offset, in sample data points, beyond the End sample header parameter to the last sample data point to be played for this instrument. For example, if End were 17 and endAddrsOffset were -2, the last sample data point played would be sample data point 15.
2	startloopAddrsOffset	The offset, in sample data points, beyond the Startloop sample header parameter to the first sample data point to be repeated in the loop for this instrument. For example, if Startloop were 10 and startloopAddrsOffset were -1, the first repeated loop sample data point would be sample data point 9.
3	endloopAddrsOffset	The offset, in sample data points, beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. For example, if Endloop were 15 and endloopAddrsOffset were 2, sample data point 17 would be considered equivalent to the

		Startloop sample data point, and hence sample data point 16 would effectively precede Startloop during looping.
4	startAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Start sample header parameter and the first sample data point to be played in this instrument. This parameter is added to the startAddrsOffset parameter. For example, if Start were 5, startAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point played would be sample data point 65544.
5	modLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
6	vibLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Vibrato LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
7	modEnvToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence pitch. A positive value indicates an increase in pitch; a negative value indicates a decrease in pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will rise 1 semitone at the envelope peak.
8	initialFilterFc	This is the cutoff and resonant frequency of the lowpass filter in absolute cent units. The lowpass filter is defined as a second order resonant pole pair whose pole frequency in Hz is defined by the Initial Filter Cutoff parameter. When the cutoff frequency exceeds 20kHz and the Q (resonance) of the filter is zero, the filter does not affect the signal.
9	initialFilterQ	This is the height above DC gain in centibels which the filter resonance exhibits at the cutoff frequency. A value of zero or less indicates the filter is not resonant; the gain at the cutoff frequency (pole angle) may be less than zero when zero is specified. The filter gain at DC is also affected by this parameter such that the gain at DC is reduced by half the specified gain. For example, for a value of 100, the filter gain at DC would be 5 dB below unity gain, and the height of the resonant peak would be 10 dB above the DC gain, or 5 dB above unity gain. Note also that if initialFilterQ is set to zero or less and the cutoff frequency exceeds 20 kHz, then the filter response is flat and unity gain.

10	modLfoToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence filter cutoff frequency. A positive number indicates a positive LFO excursion increases cutoff frequency; a negative number indicates a positive excursion decreases cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1200 indicates that the cutoff frequency will first rise 1 octave, then fall one octave.
11	modEnvToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence filter cutoff frequency. A positive number indicates an increase in cutoff frequency; a negative number indicates a decrease in filter cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1000 indicates that the cutoff frequency will rise one octave at the envelope attack peak.
12	endAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the End sample header parameter and the last sample data point to be played in this instrument. This parameter is added to the endAddrsOffset parameter. For example, if End were 65536, startAddrsOffset were -3 and startAddrsCoarseOffset were -1, the last sample data point played would be sample data point 32765.
13	modLfoToVolume	This is the degree, in centibels, to which a full scale excursion of the Modulation LFO will influence volume. A positive number indicates a positive LFO excursion increases volume; a negative number indicates a positive excursion decreases volume. Volume is always modified logarithmically, that is the deviation is in decibels rather than in linear amplitude. For example, a value of 100 indicates that the volume will first rise ten dB, then fall ten dB.
14	unused1	Unused, reserved. Should be ignored if encountered.
15	chorusEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the chorus effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the chorus effects processor.
16	reverbEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the reverb effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the reverb effects processor.

17	pan	This is the degree, in 0.1% units, to which the “dry” audio output of the note is positioned to the left or right output. A value of -50% or less indicates the signal is sent entirely to the left output and not sent to the right output; a value of +50% or more indicates the signal is sent entirely to the right and not sent to the left. A value of zero sends the signal equally to left and right. For example, a value of -250 indicates that the signal is sent at 75% of full level to the left output and 25% of full level to the right output.
18	unused2	Unused, reserved. Should be ignored if encountered.
19	unused3	Unused, reserved. Should be ignored if encountered.
20	unused4	Unused, reserved. Should be ignored if encountered.
21	delayModLFO	This is the delay time, in absolute timecents, from key on until the Modulation LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second and a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
22	freqModLFO	This is the frequency, in absolute cents, of the Modulation LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 MHz would be 1200log2(.01/8.176) = -11610.
23	delayVibLFO	This is the delay time, in absolute timecents, from key on until the Vibrato LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
24	freqVibLFO	This is the frequency, in absolute cents, of the Vibrato LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 mHz would be 1200log2(.01/8.176) = -11610.
25	delayModEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Modulation envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
26	attackModEnv	This is the time, in absolute timecents, from the end of the Modulation Envelope Delay Time until the point at which the Modulation Envelope value reaches its peak. Note that the attack is

		"convex"; the curve is nominally such that when applied to a decibel or semitone parameter, the result is linear in amplitude or Hz respectively. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
27	holdModEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
28	decayModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during decay phase. For the Modulation Envelope, the decay phase linearly ramps toward the sustain level. If the sustain level were zero, the Modulation Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
29	sustainModEnv	This is the decrease in level, expressed in 0.1% units, to which the Modulation Envelope value ramps during the decay phase. For the Modulation Envelope, the sustain level is properly expressed in percent of full scale. Because the volume envelope sustain level is expressed as an attenuation from full scale, the sustain level is analogously expressed as a decrease from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; values above 1000 are to be interpreted as 1000. For example, a sustain level which corresponds to an absolute value 40% of peak would be 600.
30	releaseModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during release phase. For the Modulation Envelope, the release phase linearly ramps toward zero from the current level. If the current level were full scale, the Modulation Envelope Release Time would be the time spent in release phase until zero value were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
31	keynumToModEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave

		causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
32	keynumToModEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
33	delayVolEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Volume envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
34	attackVolEnv	This is the time, in absolute timecents, from the end of the Volume Envelope Delay Time until the point at which the Volume Envelope value reaches its peak. Note that the attack is “convex”; the curve is nominally such that when applied to the decibel volume parameter, the result is linear in amplitude. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
35	holdVolEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the Volume envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
36	decayVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during decay phase. For the Volume Envelope, the decay phase linearly ramps toward the sustain level, causing a constant dB change for each time unit. If the sustain level were -96dB, the Volume Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
37	sustainVolEnv	This is the decrease in level, expressed in centibels, to which the Volume Envelope value ramps during the decay phase. For the

		Volume Envelope, the sustain level is best expressed in centibels of attenuation from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; conventionally 1000 indicates full attenuation. For example, a sustain level which corresponds to an absolute value 12dB below of peak would be 120.
38	releaseVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during release phase. For the Volume Envelope, the release phase linearly ramps toward zero from the current level, causing a constant dB change for each time unit. If the current level were full scale, the Volume Envelope Release Time would be the time spent in release phase until 96dB attenuation were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
39	keynumToVolEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
40	keynumToVolEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
41	instrument	This is the index into the INST subchunk providing the instrument to be used for the current preset zone. A value of zero indicates the first instrument in the list. The value should never exceed two less than the size of the instrument list. The instrument enumerator is the terminal generator for PGEN zones. As such, it should only appear in the PGEN subchunk, and it must appear as the last generator enumerator in all but the global preset zone.
42	reserved1	Unused, reserved. Should be ignored if encountered.
43	keyRange	This is the minimum and maximum MIDI key number values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid key. The

		keyRange enumerator is optional, but when it does appear, it must be the first generator in the zone generator list.
44	velRange	This is the minimum and maximum MIDI velocity values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid velocity. The velRange enumerator is optional, but when it does appear, it must be preceded only by keyRange in the zone generator list.
45	startloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Startloop sample header parameter and the first sample data point to be repeated in this instrument's loop. This parameter is added to the startloopAddrsOffset parameter. For example, if Startloop were 5, startloopAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point in the loop would be sample data point 65544.
46	keynum	This enumerator forces the MIDI key number to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
47	velocity	This enumerator forces the MIDI velocity to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
48	initialAttenuation	This is the attenuation, in .4 centibel units, by which a note is attenuated below full scale. A value of zero indicates no attenuation; the note will be played at full scale. For example, a value of 60 indicates the note will be played at 2.4 dB below full scale for the note.
49	reserved2	Unused, reserved. Should be ignored if encountered.
50	endloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. This parameter is added to the endloopAddrsOffset parameter. For example, if Endloop were 5, endloopAddrsOffset were 3 and endAddrsCoarseOffset were 2, sample data point 65544 would be considered equivalent to the Startloop sample data point, and hence sample data point 65543 would effectively precede Startloop during looping.
51	coarseTune	This is a pitch offset, in semitones, which should be applied to the note. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Coarse Tune value of -4 would cause the sound to be reproduced four semitones flat.
52	fineTune	This is a pitch offset, in cents, which should be applied to the note. It is additive with coarseTune. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Fine Tuning value of -5 would cause the sound to be reproduced five cents flat.

53	sampleID	This is the index into the SHDR subchunk providing the sample to be used for the current instrument zone. A value of zero indicates the first sample in the list. The value should never exceed two less than the size of the sample list. The sampleID enumerator is the terminal generator for IGEN zones. As such, it should only appear in the IGEN subchunk, and it must appear as the last generator enumerator in all but the global zone.
54	sampleModes	This enumerator indicates a value which gives a variety of Boolean flags describing the sample for the current instrument zone. The sampleModes should only appear in the IGEN subchunk, and should not appear in the global zone. The two LS bits of the value indicate the type of loop in the sample: 0 indicates a sound reproduced with no loop, 1 indicates a sound which loops continuously, 2 is unused but should be interpreted as indicating no loop, and 3 indicates a sound which loops for the duration of key depression then proceeds to play the remainder of the sample.
55	reserved3	Unused, reserved. Should be ignored if encountered.
56	scaleTuning	This parameter represents the degree to which MIDI key number influences pitch. A value of zero indicates that MIDI key number has no effect on pitch; a value of 100 represents the usual tempered semitone scale.
57	exclusiveClass	This parameter provides the capability for a key depression in a given instrument to terminate the playback of other instruments. This is particularly useful for percussive instruments such as a hi-hat cymbal. An exclusive class value of zero indicates no exclusive class; no special action is taken. Any other value indicates that when this note is initiated, any other sounding note with the same exclusive class value should be rapidly terminated. The exclusive class generator can only appear at the instrument level. The scope of the exclusive class is the entire preset. In other words, any other instrument zone within the same preset holding a corresponding exclusive class will be terminated.
58	overridingRootKey	This parameter represents the MIDI key number at which the sample is to be played back at its original sample rate. If not present, or if present with a value of -1, then the sample header parameter Original Key is used in its place. If it is present in the range 0-127, then the indicated key number will cause the sample to be played back at its sample header Sample Rate. For example, if the sample were a recording of a piano middle C (Original Key = 60) at a sample rate of 22.050 kHz, and Root Key were set to 69, then playing MIDI key number 69 (A above middle C) would cause a piano note of pitch middle C to be heard.
59	unused5	Unused, reserved. Should be ignored if encountered.
60	endOper	Unused, reserved. Should be ignored if encountered. Unique name provides value to end of defined list.

5.9.3.1.3 Generator Summary

The following tables give the ranges and default values for all SASBF defined generators.

#	Name	Unit	Abs Zero	Min	Min Useful	Max	Max Useful	De-fault	Def Value
0	startAddrsOffset +	smpls	0	0	None	*	*	0	None
1	endAddrsOffset +	smpls	0	*	*	0	None	0	None
2	startloopAddrsOffset +	smpls	0	*	*	*	*	0	None
3	endloopAddrsOffset +	smpls	0	*	*	*	*	0	None
4	startAddrsCoarseOffset +	32k	0	0	None	*	*	0	None
5	modLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
6	vibLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
7	modEnvToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
8	initialFilterFc	cent	8.176 Hz	1500	20 Hz	13500	20 kHz	13500	Open
9	initialFilterQ	cB	0	0	None	960	96 dB	0	None
10	modLfoToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
11	modEnvToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
12	endAddrsCoarseOffset +	32k	0	*	*	0	None	0	None
13	modLfoToVolume	cB fs	0	-960	-96 dB	960	96 dB	0	None
15	chorusEffectsSend	0.1%	0	0	None	1000	100%	0	None
16	reverbEffectsSend	0.1%	0	0	None	1000	100%	0	None
17	pan	0.1%	Cntr	-500	Left	+500	Right	0	Centre
21	delayModLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
22	freqModLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
23	delayVibLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
24	freqVibLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
25	delayModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
26	attackModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
27	holdModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
28	decayModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
29	sustainModEnv	-0.1%	attk peak	0	100%	1000	0%	0	attk pk
30	releaseModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
31	keynumToModEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
32	keynumToModEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
33	delayVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
34	attackVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
35	holdVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
36	decayVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
37	sustainVolEnv	cB attn	attk peak	0	0 dB	1440	144dB	0	attk pk
38	releaseVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
39	keynumToVolEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
40	keynumToVolEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
43	keyRange	MIDI ky#	key# 0	0	lo key	127	hi key	0-127	full kbd
44	velRange	MIDI vel	0	0	min vel	127	max vel	0-127	all vels
45	startloopAddrsCoarseOffset +	smpl	0	*	*	*	*	0	None

46	keynum +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None
47	velocity +	MIDI vel	0	1	min vel	127	max vel	-1	None
48	initialAttenuation	.4 cB	0	0	0 dB	1440	144dB	0	None
50	endloopAddrsCoarseOffset	smpls	0	*	*	*	*	0	None
51	CoarseTune	semitone	0	-120	-10 oct	120	10 oct	0	None
52	fineTune	cent	0	-99	-99cnt	99	99cent	0	None
54	sampleModes +	Bit Flags	Flags	**	**	**	**	0	No Loop
56	scaleTuning	cent/key	0	0	none	1200	oct/ky	100	semi-tone
57	exclusiveClass +	arbitrary #	0	1	--	127	--	0	None
58	overridingRootKey +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None

* Range depends on values of start, loop, and end points in sample header.

** Range has discrete values based on bit flags

\+ This generator is only valid at the instrument level.

5.9.3.2 Default Modulators

The “default” modulators are described below.

5.9.3.2.1 MIDI Key Velocity to Initial Attenuation

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.2 MIDI Key Velocity to Filter Cutoff

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is -2400 Cents.

The product of these values is added to the Initial Filter Cutoff generator summing node.

5.9.3.2.3 MIDI Channel Pressure to Vibrato LFO Pitch Depth

The MIDI Channel Pressure data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 50 cents per max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.4 MIDI Continuous Controller 1 to Vibrato LFO Pitch Depth

The MIDI Continuous Controller 1 data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. The MIDI Continuous Controller 33 data value may be optionally used for increased resolution of the controller input.

There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 50 cents/max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.5 MIDI Continuous Controller 7 to Initial Attenuation

The MIDI Continuous Controller 7 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.6 MIDI Continuous Controller 10 to Pan Position

The MIDI Continuous Controller 10 data value is used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 127 is mapped to 63/64 and all other values are mapped between -1 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 1000 tenths of a percent panned-right.

The product of these values is added to the Pan generator summing node.

5.9.3.2.7 MIDI Continuous Controller 11 to Initial Attenuation

The MIDI Continuous Controller 11 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.8 MIDI Continuous Controller 91 to Reverb Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added reverb send.

The product of these values is added to the Reverb Send generator summing node.

5.9.3.2.9 MIDI Continuous Controller 93 to Chorus Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added chorus send.

The product of these values is added to the Chorus Send generator summing node.

5.9.3.2.10 MIDI Pitch Wheel to Initial Pitch Controlled by MIDI Pitch Wheel Sensitivity

The MIDI Pitch Wheel data values are used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 16383 is mapped to 8191/8192 and all other values are mapped between -1 and 8191/8192 in a linear fashion.

The MIDI Pitch Wheel Sensitivity data values are used as a secondary source. This source is Positive Unipolar, thus an input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion.

The amount of this modulator is 12700 Cents.

The product of these values is added to the Initial Pitch generator summing node.

5.9.3.3 Precedence and Absolute and Relative values.

Most SASBF generators are available at both the Instrument and Preset Levels, as well as having a default value. Generators at the Instrument Level are considered “absolute” and determine an actual physical value for the associated synthesis parameter, which is used instead of the default. For example, a value of 1200 for the attackVolEnv generator would produce an absolute time of 1200 timecents or 2 seconds of attack time for the volume envelope, instead of the default value of -12000 timecents or 1 msec.

Generators at the Preset Level are instead considered “relative” and additive to all the default or instrument level generators within the Preset Zone. For example, a value of 2400 timecents for the attackVolEnv generator in a preset zone containing an instrument with two zones, one with the default attackVolEnv and one with an absolute attackVolEnv generator value of 1200 timecents would cause the default zone to actually have a value of -9600 timecents or 4 msec, and the other to have a value of 3600 timecents or 8 seconds attack time.

There are some generators which are not available at the Preset Level. These are:

#	Name
0	startAddrsOffset
1	endAddrsOffset
2	startloopAddrsOffset
3	endloopAddrsOffset
4	startAddrsCoarseOffset
12	endAddrsCoarseOffset
45	startloopAddrsCoarseOffset
46	keynum
47	velocity
50	endloopAddrsCoarseOffset
54	sampleModes

57	exclusiveClass
58	overridingRootKey

If these generators are encountered in the Preset Level, they should be ignored.

The effect of modulators on a given destination is always relative to the generator value at the Instrument level. However modulators may supersede or add to other modulators depending on their position within the hierarchy. Please see Subclause 8.4 for details on the Modulator implementation and the hierarchical details.

5.9.4 Parameters and Synthesis Model

The SASBF standard has been established with the intent of providing support for an expanding base of wavetable based synthesis models. The model supported by the SASBF specification originates with the EMU8000 wavetable synthesiser chip. The description below of the underlying synthesis model and the associated parameters are provided to allow mapping of this synthesis model onto other hardware platforms.

5.9.4.1 Synthesis Model

The SASBF specification Synthesis Model comprises a wavetable oscillator, a dynamic lowpass filter, an enveloping amplifier, and programmable sends to pan, reverb, and chorus effects units. An underlying modulation engine comprises two low frequency oscillators (LFOs) and two envelope generators with appropriate routing amplifiers.

5.9.4.1.1 Wavetable Oscillator/Interpolator

The SASBF specification wavetable oscillator model is capable of playing back a sample at an arbitrary sampling rate with an arbitrary pitch shift. In practice, the upward pitch shift (downward sample rate conversion) will be limited to a maximum value, typically at least two octaves. The pitch is described in terms of an initial pitch shift which is based on the sample's sampling rate, the root key at which the sample should be unshifted on the keyboard, the coarse, fine, and correction tunings, the effective MIDI key number, and the keyboard scale factor. All modulations in pitch are in octaves, semitones, and cents.

In general, interpolators have a symmetric impulse response, and thus a linear phase characteristic. The interpolation filter's magnitude response can be characterised by three regions - the pass band, the transition band, and the stop band.

The interpolation filter's pass band is the portion of its response which corresponds to the frequencies of the signal stored in waveform memory from DC to that signal's Nyquist frequency

The interpolation filter's transition band is the portion of its response which corresponds to the first image of the signal stored in waveform memory, that is from that signal's Nyquist frequency back to DC.

The interpolation filter's stop band is the portion of its response which corresponds to the remaining images above the transition band to the Nyquist frequency of the upsampled signal.

The guardband will be assumed to begin at 5/6 of the Nyquist frequency, as is the case for a 20 kHz bandwidth in a 48 kHz sample rate system.

Due to poor aliasing rejection in the stopband, linear interpolation does not satisfy this specification.

The specification constrains the Fourier transform of the impulse response of the interpolator. The impulse response is taken with a downward pitch shift of eight octaves. This gives a response which has been upsampled by a factor of 2**8 or 256. In other words, a frequency of the impulse response's Nyquist

frequency divided by 256 gives the filter frequency corresponding to the waveform memory's Nyquist frequency. In the specification below, Fn represents this waveform memory Nyquist frequency.

(All responses measured with downward pitch shift of 8 octaves.)	
Passband Response:	
Ripple:	No more than +/- 0.5 dB
Roll-off:	Minimal, but not to exceed 6 dB at 83.3% Fn.
Transition Band Response:	
Roll-off:	Monotonic and as rapid as possible to at least -80 dB attenuation
Return Lobes:	None above -80 dB
Attenuation:	Greater than 80 dB for all frequencies DC to at least 2% Fn in the first lobe.
Stop Band Response:	
Attenuation:	Greater than 90 dB for all frequencies DC to at least 1% Fn
	Greater than 80 dB for all frequencies DC to at least 20%Fn
	Greater than 60 dB for all frequencies

5.9.4.1.2 Sample Looping

The wavetable oscillator is playing a digital sample which is described in terms of a start point, end point, and two points describing a loop. The sound can be flagged as unlooped, in which case the loop points are ignored. If the sound is looped, it can be played in two ways. If it is flagged as "loop during release", the sound is played from the start point through the loop, and loops until the note becomes inaudible. If not, the sound is played from the start point through the loop, and loops until the key is released. At this point, the next time the loop end point is reached, the sound continues through the loop end point and plays until the end point is reached, at which time audio is terminated.

5.9.4.1.3 Lowpass Filter

The synthesis model contains a resonant lowpass filter, which is characterised by a dynamic cutoff frequency and a fixed resonance (Q).

The filter is idealised at zero resonance as having a flat passband to the cutoff frequency, then a rolloff at 12 dB per octave above that frequency. The resonance, when non-zero, comprises a peak at the cutoff frequency, superimposed on the above response. The resonance is measured as a dB ratio of the resonant peak to the DC gain. The DC gain at any resonance is half of the resonance value below the DC gain at zero resonance; hence the peak height is half the resonance value above DC gain at zero resonance.

All modulations in cutoff frequency are in cents.

Topology: 2nd order AR lowpass filter or equivalent. Filter coefficients are updated in a smooth manner at the sample rate.

Noise and Distortion: Less than 0.003% of full scale on any sinusoidal input for any parameter setting.

Control Parameters: Cutoff Frequency and Resonance.

Resonance Parameter: Specifies the ratio in decibels of the gain of a resonant peak to the gain at DC, when the filter cutoff frequency is set to 1.5 kHz and the modulation is zero. The DC gain of a filter is reduced from the DC gain of a filter with zero resonance by half the resonance value. The resonance parameter will remain fixed throughout the sounding of a musical note.

For a specified resonance, the actual gain ratio should be accurate within +/- 1 dB, and the DC gain accurate within +/-0.5 dB. The ratio of the gain at the resonant peak to the DC gain for all cutoff frequency values shall not deviate by more than 2dB per octave deviation from 1.5 kHz. Nominal gain at DC for a minimum resonance parameter is unity gain.

Cutoff Frequency Parameter: Specifies the frequency at which the filter achieves exactly 3dB of attenuation. The Cutoff Frequency is computed by the combination of the Initial Cutoff Frequency, specified in Hz, and the modulation, specified in semitones and fractions thereof. The Initial Cutoff Frequency is fixed throughout the duration of the a musical note; the modulation can vary at the sample rate. Within the region from 200 Hz to 6.4 kHz, the cutoff frequency parameter accuracy will be +/- 2 semitones.

Frequency Magnitude Response: When the cutoff frequency is at its maximum value and the resonance is at zero, the filter must pass audio without alteration. The filter must support cutoff frequencies down to 200 Hz. There must be a minimum of 2048 distinct cutoff frequencies per octave within the region from 200 Hz to 6.4 kHz, with a worst case spacing between distinct frequencies of 0.01 semitones.

5.9.4.1.4 Final Gain Amplifier

The final gain amplifier is a multiplier on the filter output, which is controlled by an initial gain in dB. This is added to the volume envelope. Additional modulation can also be added. The gain is always specified in dB.

5.9.4.1.5 Effects Sends

The output of the final gain amplifier can be routed into the effects unit. This unit causes the sound to be located (panned) in the stereo field, and a degree of reverberation and chorus to be added. The pan is specified in terms of percentage left and right, which also could be considered as an azimuth angle. The reverb and chorus sends are specified as a percentage of the signal amplitude to be sent to these units, from 0% to 100%.

5.9.4.1.6 Low Frequency Oscillators

The synthesis model provides for two low frequency oscillators (LFOs) for modulating pitch, filter cutoff, and amplitude. The “vibrato” LFO is only capable of modulating pitch. The “modulation” LFO can modulate any of the three parameters.

An LFO is defined as having a delay period during which its value remains zero, followed by a triangular waveshape ramping linearly to positive one, then downward again to negative 1, then upward again to positive one, etc.

Each parameter can be modulated to a varying degree, either positively or negatively, by the associated LFO. Modulations of pitch and cutoff are in octaves, semitones, and cents, while modulations of amplitude are in dB. The degree of modulation is specified in cents or dB for the full scale positive LFO excursion.

5.9.4.1.7 Envelope Generators

The synthesis model provides for two envelope generators. The volume envelope generator controls the final gain amplifier and hence determines the volume contour of the sound. The modulation envelope can control pitch and/or filter cutoff.

An envelope generates a control signal in six phases. When key-on occurs, a delay period begins during which the envelope value is zero. The envelope then rises in a convex curve to a value of one during the attack phase. When a value of one is reached, the envelope enters a hold phase during which it remains at one. When the hold phase ends, the envelope enters a decay phase during which its value decreases linearly to a sustain level. When the sustain level is reached, the envelope enters sustain phase, during which the envelope stays at the sustain level. Whenever a key-off occurs, the envelope immediately enters a release phase during which the value linearly ramps from the current value to zero. When zero is reached, the envelope value remains at zero.

Modulation of pitch and filter cutoff are in octaves, semitones, and cents. These parameters can be modulated to varying degree, either positively or negatively, by the modulation envelope. The degree of modulation is specified in cents for the full scale attack peak.

The volume envelope operates in dB, with the attack peak providing a full scale output, appropriately scaled by the initial volume. The zero value, however, is actually zero gain. When 96 dB of attenuation is reached in the final gain amplifier, an abrupt jump to zero gain (infinite dB of attenuation) occurs. In a 16-bit system, this jump is inaudible.

5.9.4.1.8 Modulation Interconnection Summary

The following diagram shows the interconnections expressed in the SASBF specification synthesis model:

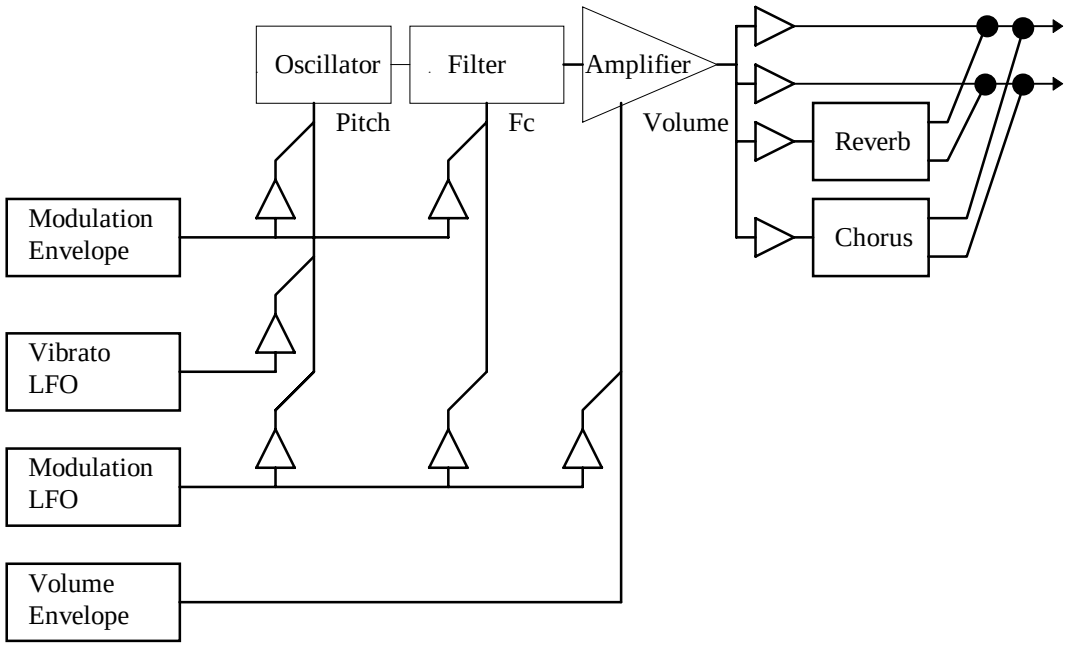


Figure 1: Generator Based Modulation Structure

5.9.4.2 MIDI Functions

The response to certain MIDI commands is defined within the MIDI specification, and therefore not considered to be part of the SASBF specification. These MIDI commands may not be used as sources for the Modulator implementation.

For completeness, the expected responses are given here.

MIDI CC0 Bank Select - When received, the following program change should select the MIDI program in this bank value instead of the default bank of 0.

MIDI CC6 - Data Entry MSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism depending on the Data Entry mode.

MIDI CC32 Bank Select LSB - When received, may behave in conjunction with CC0 Bank Select to provide a total of 16384 possible MIDI banks of programs.

MIDI CC38 Data Entry LSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism, depending on the Data Entry mode.

MIDI CC64 Sustain - ACTIVE when greater than or equal to 64. When the sustain function is active, all notes in the key-on state remain in the key-on state regardless of whether a key-off command for the note arrives. The key-off commands are stored, and when sustain becomes inactive, all stored key-off commands are executed.

MIDI CC66 Soft - ACTIVE when greater than or equal to 64. When active, all new key-ons are modulated in such a way to make the note sound “soft.” This typically affects initial attenuation and filter cutoff in a pre-defined manner.

MIDI CC67 Sostenuto - ACTIVE when greater than or equal to 64. When sostenuto becomes active, all notes currently in the key-on state remain in the key-on state until the sostenuto becomes inactive. All other notes behave normally. Notes maintained by sostenuto in key-on state remain in key-on state even if sustain is switched on and off.

MIDI CC98 NRPN LSB - When received, should be processed by the NRPN implementation mechanism.

MIDI CC99 NRPN MSB - When received, should put the synthesiser in NRPN Data Entry mode and then should be processed by the NRPN implementation mechanism.

MIDI CC100 RPN LSB - When received, should be processed by the RPN implementation mechanism.

MIDI CC101 RPN MSB - When received, should put the synthesiser in RPN Data Entry mode and then should be processed by the RPN implementation mechanism.

MIDI CC120 All Sound Off - When received with any data value, all notes playing in the key-on state bypass the release phase and are shut off, regardless of the sustain or sostenuto positions.

MIDI CC121 Reset All Controllers - Defined as Reset All Controllers as defined by the MIDI specification.

MIDI CC123 All Notes Off - When received with any data value, all notes playing in the key-on state immediately enter release phase, pending their status in SUSTAIN or SOSTENUTO state.

5.9.4.3 Parameter Units

The units with which SASBF generators are described are all well defined. The strict definitions appear below:

ABSOLUTE SAMPLE DATA POINTS - A numeric index of 16 bit sample data point words as stored in ROM or supplied in the smpl-ck, indexing the first sample data point word of memory or the chunk as zero.

RELATIVE SAMPLE DATA POINTS - A count of 16 bit sample data point words based on an absolute sample data point reference. A negative value implies a relative count toward the beginning of the data.

ABSOLUTE SEMITONES - An absolute logarithmic measure of frequency based on a reference of MIDI key numbers. A semitone is 1/12 of an octave, and value 69 is 440 Hz (A-440). Negative values and values above 127 are allowed.

RELATIVE SEMITONES - A relative logarithmic measure of frequency ratio based on units of 1/12 of an octave, which is the twelfth root of two, approximately 1.059463094.

ABSOLUTE CENTS - An absolute logarithmic measure of frequency based on a reference of MIDI key number scaled by 100. A cent is 1/1200 of an octave, and value 6900 is 440 Hz (A-440). Negative values and values above 12700 are allowed.

RELATIVE CENTS - A relative logarithmic measure of frequency ratio based on units of 1/1200 of an octave, which is the twelve hundredth root of two, approximately 1.000577790.

ABSOLUTE CENTIBELS - An absolute measure of the attenuation of a signal, based on a reference of zero being no attenuation. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

RELATIVE CENTIBELS - A relative measure of the attenuation of a signal. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

ABSOLUTE TIMECENTS - An absolute measure of time, based on a reference of zero being one second. A timecent represents a ratio in time of the twelve hundredth root of two, approximately 1.011579454.

RELATIVE TIMECENTS - A relative measure of time ratio, based on a unit size of the twelve hundredth root of two, approximately 1.011579454.

ABSOLUTE PERCENT - An absolute measure of gain, based on a reference of unity. In SASBF, absolute percent is measured in 0.1% units, so a value of zero is 0% and a value of 1000 is 100%.

RELATIVE PERCENT - A relative measure of gain difference. In SASBF, relative percent is measured in 0.1% units. When the gain goes below zero, zero is assumed; when the gain exceeds 100%, 100% is used.

5.9.4.4 The SASBF Generator Model

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

In case it is not clear in the general description of the SASBF hierarchy, the following is the precedence of SASBF generator in the hierarchy.

- A ‘generator’ sets or offsets the value of a destination or a synthesis parameter. In exception cases, it sets ranges (Range Generators), or sets values and never offsets values (Index Generators, Sample Generators, and Substitution Generators).
- A generator is defined as identical to another generator if its generator operator is the same in both generators.
- A generator in a global instrument zone which is identical to a default generator supersedes or replaces the default generator.
- A generator in a local instrument zone which is identical to a default generator or to a generator in a global instrument zone supersedes or replaces that generator.
- Points below (until noted) apply to Value Generators ONLY.
- A generator at the preset level adds to a generator at the instrument level if both generators are identical.
- A generator in a global preset zone which is identical to a default generator or to a generator in an instrument adds to that generator.
- A generator in a global preset zone which is not identical to a default generator and is not identical to a generator in an instrument has its effect added to the given synthesis parameter.
- A generator in a local preset zone which is identical to a generator in a global preset zone supersedes or replaces that generator in the global preset zone. That generator then has its effects added to the destination summing node of all zones in the given instrument.
- A generator in a local preset zone which is not identical to a default generator or a generator in a global preset zone has its effects added to the destination summing node of all zones in the given instrument.
  If the generator operator is a Range Generator, the generator values are NOT ADDED to those

in the instrument level, rather they serve as an intersection filter to those key number or velocity ranges in the instrument which is used in the preset zone.

- If the generator operator is a Substitution Generator or a Sample Generator, they are illegal at the preset level. The only Index Generator legal at the Preset Level is ‘instrumentID’, whereas the only Index Generator legal at the Instrument Level is ‘sampleID’

5.9.4.5 The SASBF Modulator Controller Model

SASBF Modulators are used to allow real-time control over the sound in sound designer programmable manner. Each instance of a SASBF modulator structure defines a real-time perceptually additive effect to be applied to a given destination or synthesiser parameter.

Modulators provide future extensibility to the SASBF standard. They are not used in this version.

5.9.5 Error Handling

5.9.5.1 Structural Errors

Structural Errors are errors which are determined from the implicit redundancy of the SASBF RIFF bitstream element structure, and indicate that the structure is not intact. Examples are incorrect lengths for the chunks or subchunks, pointers out of valid range, or missing required chunks or subchunks for which no error correction procedure exists.

In all cases, bitstream elements should be checked for structural errors at load time, and if any are found, the bitstream elements should be rejected. Separate tools or options can be used to “repair” structurally defective bitstream elements, but these tools should validate that the reconstructed bitstream element is not only a valid SASBF bank but also complies with the intended timbral results in all cases.

5.9.5.2 Unknown Chunks

In parsing the RIFF structure, unknown but well formed chunks or subchunks may be encountered. Unknown chunks within the INFO-list chunk should simply be ignored. Other unknown chunks or subchunks are illegal and should be treated as structural errors.

5.9.5.3 Unknown Enumerators

Unknown enumerators may be encountered in Generators, Modulator Sources, or Transforms. This is to be expected if the ifil field exceeds the specification to which the application was written. Even if unexpected, unknown enumerators should simply cause the associated Generator or Modulator to be ignored.

5.9.5.4 Illegal Parameter Values

Some SASBF parameters are defined for only a limited range of the possible values which can be expressed in their field. If the value of the field is not in the defined range, the parameter has an illegal value.

Illegal values for may be detected either at load or at run time. If detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the default value for the parameter should be used if the parameter is required, or the entire Generator or Modulator ignored if it is optional. Certain parameters may have more specific procedures for illegal values as expressed elsewhere in this specification.

5.9.5.5 Out-of-range Values

SASBF parameters have a specified minimum and useful range the span the perceptually relevant values for the associated sonic property. When the parameter value is exceeds this useful range, the parameter is said to have an out of range value.

Out of range values can result from two distinct causes. An out of range value can be actually present as a SASBF generator value, or the out of range value can be the result of the summation of instrument and preset values.

Out of range values should be handled by substituting the nearest perceptually relevant or realisable value. SASBF banks should not be created with out of range values in the instrument generators. While it is acceptable practice to create SASBF banks which produce out of range values as a result of summation, it is undesirable and should be avoided where practical.

5.9.5.6 Missing Required Parameter or Terminator

Certain parameters and terminators are required by the SASBF specification. If these are missing, the bitstream element is technically not within specification. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the instrument or zone for which the required parameter is missing should simply be ignored. If this causes no sound, the corresponding key-on event is ignored.

5.9.5.7 Illegal enumerator

Certain enumerators are illegal in certain contexts. For example, key and velocity ranges must be the first generators in a zone, instruments are not allowed in instrument zones, and sampleIDs are not allowed in preset zones. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the enumerator should simply be ignored.

5.9.6 Profile 2 (Sample Bank and MIDI decoding)

This Subclause describes the decoding process in which a bitstream conforming to Profile 2 is converted into sound. Profile 2 supports the Structured Audio Sample Bank Format and the General MIDI Format (Profile 1) for sound description. Profile 2 supports the MIDI Protocol and the Standard MIDI File Format for score specification.

5.9.6.1 Stream information header

The SASBF bitstream element is part of the bitstream header. Score elements in the form of MIDI files may be included in the bitstream header.

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class `SA_streaminfo` provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- MIDI file: The events in the MIDI file shall be time ordered, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed. The sample bank requires no processing for this preparation. Processing may be used to improve the computational

efficiency of a decoder. Banks shall be assigned consecutive increasing bank ID numbers in the order of the banks' position in the bitstream. The first bank in the bitstream shall be numbered 0 unless a bank resident in the terminal is being used. In that case, the resident bank shall be numbered 0, an other banks shall be numbered proceeding from there.

5.9.6.2 Bitstream data and sound creation

The MIDI Note On message is defined in the MIDI specification. When the SASBF decoder receives a Note On message, a voice shall be instantiated. Synthesis parameters for the voice shall be determined by the data in the sample bank containing the preset corresponding to the MIDI channel of the Note On message. The number of wavetable oscillators that are used by the voice is defined by the sample bank. Each required oscillator shall have the state variables required for synthesis allocated to it. The state variables shall be initialised according to the preset data from the sample bank.

The MIDI Program Change message is defined in the MIDI specification. When the SASBF decoder receives a Program Change message, an assignment shall be made in the scheduler between the specified MIDI channel and the specified preset. Until this assignment is changed by a subsequent Program Change message, subsequent voice instantiations using the specified MIDI channel shall use sample bank data corresponding to the assigned preset. In a MIDI program change message, if no preset exists in the specified bank with the specified preset number, a replacement preset is used. The replacement preset is the preset with the specified preset number in the bank with the highest bank ID number less than the specified bank ID number which contains a preset with the specified preset number.

The run time scheduler is used to issue MIDI messages to the SASBF decoder. MIDI messages from a MIDI standard file shall be issued by the scheduler when the scheduler clock equals or exceeds the time stamp associated with the message.

5.9.6.3 Conformance

Floating point computation is not required in Profile 2. Audio samples are stored in the bitstream as 16-bit integers. The resolution of the interpolator filter is constrained by the interpolator specification given in Subclause 0.

5.9.7 Profile 4 (Sample Bank decoding in SAOL instruments)

5.9.8 Sample Bank Format Glossary

absolute - Describes a parameter which gives a definitive real-world value. Contrast to relative.

additive - Describes a parameter which is to be numerically added to another parameter.

articulation - The process of modulation of amplitude, pitch, and timbre to produce an expressive musical note.

attack - That phase of an envelope or sound during which the amplitude increases from zero to a peak value.

attenuation - A decrease in volume or amplitude of a signal.

bag - A Sample Bank Format data structure element containing a list of zones.

balance - A form of stereo volume control in which both left and right channels are at maximum when the control is centred, and which attenuates only the opposite channel when taken to either extreme.

bank - A collection of presets. See also MIDI bank.

bipolar - In the SASBF standard, said of a modulator source whose minimum is -1 and whose maximum is 1. Contrast “unipolar”

big endian - Refers to the organisation in memory of bytes within a word such that the most significant byte occurs at the lowest address. Contrast “little endian.”

byte - A data structure element of eight bits without definition of meaning to those bits.

BYTE - A data structure element of eight bits which contains an unsigned value from 0 to 255.

case-insensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as identical. Contrast “case-sensitive.”

case-sensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as distinct. Contrast “case-insensitive.”

cent - A unit of pitch ratio corresponding to the twelve hundredth root of two, or one hundredth of a semitone, approximately 1.000577790.

centibel - A unit of amplitude ratio corresponding to the two hundredth root of ten, or one tenth of a decibel, approximately 1.011579454.

CHAR - A data structure of eight bits which contains a signed value from -128 to +127.

chorus - An effects processing algorithm which involves cyclically shifting the pitch of a signal and remixing it with itself to produce a time varying comb filter, giving a perception of motion and fullness to the resulting sound.

chunk - The top-level division of a RIFF file.

convex - A curve which is bowed in such a way that it is steeper on its lower portion.

concave - (1) A curve which is bowed in such a way that it is steeper on its upper portion. (2) In the SASBF standard, said of a modulator source whose shape is that of the amplitude squared characteristic. Contrast with “convex” and “linear.”

cutoff frequency - The frequency of a filter function at which the attenuation reaches a specified value.

data points - The individual values comprising a sample. Sometimes also called sample points. Contrast “sample.”

decay - The portion of an envelope or sound during which the amplitude declines from a peak to steady state value.

decibel - A unit of amplitude ratio corresponding to the twentieth root of ten, approximately 1.122018454.

delay - The portion of an envelope or LFO function which elapses from a key-on event until the amplitude becomes non-zero.

destination - The generator to which a modulator is applied.

DC gain - The degree of amplification or attenuation a system presents to a static or zero frequency signal.

digital audio - Audio represented as a sequence of quantised values spaced evenly over time. The values are called “sample data points.”

doubleword - A data structure element of 32 bits without definition of meaning to those bits.

downloadable - Said of samples which are loaded from a file into RAM, in contrast to samples which are maintained in ROM.

dry - Refers to audio which has not received any effects processing such as reverb or chorus.

DWORD - A data structure of 32 bits which contains an unsigned value from zero to 4,294,967,295.

envelope - A time varying signal which typically controls the pitch, volume, and/or filter cutoff frequency of a note, and comprises multiple phases including attack, decay, sustain, and release.

enumerated - Said of a data element whose symbols correspond to particular assigned functions.

flat - A. Said of a tone that is lower in pitch than another reference tone. B. Said of a frequency response that does not deviate significantly from a single fixed gain over the audio range.

generator - In the SASBF standard, a parameter which directly affects sound reproduction. Contrast with “modulator.”

global - Refers to parameters which affect all associated structures. See “global zone.”

global zone - A zone whose generators and modulators affect all other zones within the object.

header - A data structure element which describes several aspects of a SASBF element.

instrument - In the SASBF standard, a collection of zones which represents the sound of a single musical instrument or sound effect set.

instrument zone - A sample and associated articulation data defined to play over certain key numbers and velocities.

interpolator - A circuit or algorithm which computes intermediate points between existing sample data points. This is of particular use in the pitch shifting operation of a wavetable synthesiser, in which these intermediate points represent the output samples of the waveform at the desired pitch transposition.

key number - See MIDI key number.

LFO - Acronym for Low Frequency Oscillator. A slow periodic modulation source.

linear - In the SASBF standard, said of a modulator source whose shape is that of a straight line. Contrast with “concave.”

linear coding - The most common method of encoding amplitudes in digital audio in which each step is of equal size.

little endian - A method of ordering bytes within larger words in memory in which the least significant byte is at the lowest address. Contrast “big endian.”

loop - In wavetable synthesis, a portion of a sample which is repeated many times to increase the duration of the resulting sound.

loop points - The sample data points at which a loop begins and ends.

lowpass - Said of a filter which attenuates high frequencies but does not attenuate low frequencies.

modulator - In the SASBF standard, a parameter which routes an external controller to dynamically alter the setting of a “generator.” Contrast with “generator.”

monotonic - Continuously increasing or decreasing. Said of a sequence which never reverses direction.

MIDI - Acronym for Musical Instrument Digital Interface. The standard protocol for sending performance information to a musical synthesiser.

MIDI bank - A group of up to 128 presets selected by a MIDI “change bank” command.

MIDI continuous controller - A construct in the MIDI protocol.

MIDI key number - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the key of the musical instrument keyboard to which the command refers.

MIDI pitch bend - A special MIDI construct akin to the MIDI continuous controllers which controls the real-time value of the pitch of all notes played in a MIDI channel.

MIDI preset - A “preset” selected to be active in a particular MIDI channel by a MIDI “change preset” command.

MIDI velocity - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the speed with which the key was pressed or released.

modulator - In the SASBF standard, a set of parameters which affect a particular generator. Contrast with “generator.”

mono - Short for “monophonic.” Indicates a sound comprising only one channel or waveform. Contrast with “stereo.”

negative - In the SASBF standard, said of a modulator which has a negative sloping characteristic. Contrast with “positive.”

octave - A factor of two in ratio, typically applied to pitch or frequency.

orphan - Said of a data structure which under normal circumstances is referenced by a higher level, but in this particular instance is no longer linked. Specifically, it is an instrument which is not referenced by any preset zone, or a sample which is not referenced by any instrument zone.

oscillator - In wavetable synthesis, the wavetable interpolator is considered an oscillator.

pan - Short for “panorama.” This is the control of the apparent azimuth of a sound source over 180 degrees from left to right. It is generally implemented by varying the volume at the left and right speakers.

pitch - The perceived value of frequency. Generally can be used interchangeably with frequency.

pitch shift - A change in pitch. Wavetable synthesis relies on interpolators to cause pitch shift in a sample to produce the notes of the scale.

pole - A mathematical term used in filter transform analysis. Traditionally in synthesis, a pole is equated with a rolloff of 6dB per octave, and the rolloff of a filter is specified in “poles.”

positive - In the SASBF standard, said of a modulator source which has a positive sloping characteristic. Contrast “negative.”

preset - A keyboard full of sound. Typically the collection of samples and articulation data associated with a particular MIDI preset number.

preset zone - A subset of a preset containing generators, modulators, and an instrument.

proximal - Closest to. Proximal sample data points are the data points closest in either direction to the named point.

Q - A mathematical term used in filter transform analysis. Indicates the degree of resonance of the filter. In synthesis terminology, it is synonymous with resonance.

RAM - Random Access Memory. Conventionally, this term implies read-write memory. Contrast “ROM.”

record - A single instance of a data structure.

relative - Describes a parameter which merely indicates an offset from an otherwise established value. Contrast to absolute.

release - The portion of an envelope or sound during which the amplitude declines from a steady state to zero value or inaudibility.

resonance - Describes the aspect of a filter in which particular frequencies are given significantly more gain than others. The resonance can be measured in dB above the DC gain.

resonant frequency - The frequency at which resonance reaches its maximum.

reverb - Short for reverberation. In synthesis, a synthetic signal processor which adds artificial spaciousness and ambience to a sound.

RIFF - Acronym for Resource Interchange File Format.

ROM - Acronym for Read Only Memory. A memory whose contents are fixed at manufacture, and hence cannot be written by the user. Contrast with RAM.

sample - This term is often used both to indicate a “sample data point” and to indicate a collection of such points comprising a digital audio waveform. The latter meaning is exclusively used in this Subclause (the SASBF specification.)

sample rate - The frequency, in Hertz, at which sample data points are taken when recording a sample.

semitone - A unit of pitch ratio corresponding to the twelfth root of two, or one twelfth of an octave, approximately 1.059463094.

sharp - Said of a tone that is higher in pitch than another reference tone.

SHORT - A data structure element of sixteen bits which contains a signed value from -32,768 to +32,767.

soft - The pedal on a piano, so named because it causes the damper to be lowered in such a way as to soften the timbre and loudness of the notes. In MIDI, continuous controller #66, which behaves in a similar manner.

sostenuto - The pedal on a piano which causes the dampers on all keys depressed to be held until the pedal is released. In MIDI, continuous controller #67, which behaves in a similar manner.

sustain - The pedal on a piano which prevents all dampers on keys as they are depressed from being released. In MIDI, continuous controller #64, which behaves in a similar manner.

source - In a SASBF modulator, the enumerator indicating the particular real-time value which the modulator will transform, scale, and add to the destination generator.

stereo - Literally indicating three dimensions. In this specification, the term is used to mean two channel stereophonic, indicating that the sound is composed of two independent audio channels, dubbed left and right. Contrast monophonic.

subchunk - A division of a RIFF file below that of the chunk.

synthesis engine - The hardware and software associated with the signal processing and modulation path for a particular synthesiser.

synthesiser - A device ideally capable of producing arbitrary musical sound.

terminator - A data structure element indicating the final element in a sequence.

timecent - A unit of duration ratio corresponding to the twelve hundredth root of two, or one twelve hundredth of an octave, approximately 1.000577790.

transform - In a SASBF modulator, the enumerator indicating the particular transfer function through which the source will be passed prior to scaling and addition to the destination generator.

tremolo - A periodic change in amplitude of a sound, typically produced by applying a low frequency oscillator to the final volume amplifier.

triangular - A waveform which ramps upward to a positive limit, then downward at the opposite slope to the symmetrically negative limit periodically.

unipolar - In the SASBF standard, said of a modulator source whose minimum is 0 and whose maximum is 1. Contrast “bipolar.”

velocity - In synthesis, the speed with which a keyboard key is depressed, typically proportionally to the impact delivered by the musician. See also MIDI velocity.

vibrato - A periodic change in the pitch of a sound, typically produced by applying a low frequency oscillator to the oscillator pitch.

volume - The loudness or amplitude of a sound, or the control of this parameter.

wavetable - A music synthesis technique wherein musical sounds are recorded or computed mathematically and stored in a memory, then played back at a variable rate to produce the desired pitch. Additional timbral adjustments are often made to the sound thus produced using amplifiers, filters, and effects processing such as reverb and chorus.

WORD - A data structure of 16 bits which contains an unsigned value from zero to 65,535.

word - A data structure element of 16 bits without definition of meaning to those bits.

zone - An object and associated articulation data defined to play over certain key numbers and velocities.

5.10 MIDI semantics, and the decoding process for Profile 2 bitstreams in Subclause 0.

5.3.2 Decoder configuration header

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class **SA_decoder_config** provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- Orchestra file: The orchestra file shall be checked for syntactic conformance with the SAOL grammar and rate semantics as specified in Subclause 5.4 SAOL syntax and semantics. Whatever preprocessing (i.e., compilation, allocation of static storage, etc.) need be done to prepare for orchestra run-time execution shall be performed.

- Score file: Each event in the score file shall be registered with the scheduler. To “register” means to inform the scheduler of the presence of a particular parametrised event at a particular future time, and the scheduler’s associated actions.

- **MIDI file: Each event in the MIDI file shall be converted into an appropriate event as described in Subclause 5.9**

Sample Bank syntax and semantics

5.9.1 Introduction

This Subclause describes the operation of the Sample Bank synthesis method for both Profile 2 and Profile 4. In Profile 2, only Sample Bank and MIDI class types shall appear in the bitstream, and this Subclause describes the normative process of generating sound from a Sample Bank bitstream data element and a sequence of MIDI instructions. In Profile 4, Sample Banks are used in the context of a SAOL instrument as described in Subclause 5.4.6.6.8 Output, and this Subclause describes the normative process of generating sound and placing it on three busses, depending on the Sample Bank bitstream data element and the particular call to **sbsynth**.

5.9.2 Elements of bitstream

5.9.2.1 RIFF Structure

5.9.2.1.1 General RIFF File Structure

The RIFF (Resource Interchange File Format) is a tagged file structure developed for multimedia resource files, and is described in some detail in the Microsoft Windows 3.1 SDK Multimedia Programmer's Reference. The Tagged-file structure is useful because it helps prevent compatibility problems which can occur as the file definition changes over time. Because each piece of data in the file is identified by a standard header, an application that does not recognise a given data element can skip over the unknown information. The relevant information is provided in the bitstream description, Subclause 0.

A RIFF file is constructed from a basic building block called a "chunk." In 'C' syntax, a chunk is defined:

```
typedef DWORD FOURCC;      //  Four-character code
typedef struct {
  FOURCC ckID;             //  Chunk ID identifies type of data in the chunk.
  DWORD ckSize;            //  Size of chunk data in bytes, excluding pad byte.
  BYTE ckDATA[ckSize];     //  Actual data + a pad byte if req'd to word align.
};
```

Two types of chunks, the "RIFF" and "LIST" chunks, may contain nested chunks called subchunks as their data.

Within a given level of the hierarchy, the ordering of the chunks is arbitrary. Any chunk with an unknown chunk ID should be ignored.

5.9.2.1.2 The Sample Bank Chunks and Subchunks

A Structured Audio Sample Bank bitstream element comprises three chunks: an INFO-list chunk containing a number of required and optional subchunks describing the bitstream element, its history, and its intended use, an sdta-list chunk comprising a single subchunk containing any referenced digital audio samples, and a pdta-list chunk containing nine subchunks which define the articulation of the digital audio data.

5.9.2.1.3 Redundancy and Error Handling in the RIFF structure

The RIFF bitstream element structure contains redundant information regarding the length of the bitstream element and the length of the chunks and subchunks. This fact enables any reader of a SASBF bitstream element to determine if the bitstream element has been damaged by loss of data.

If any such loss is detected, the SASBF bitstream element is termed “structurally unsound” and in general should be rejected.

5.9.2.2 The INFO-list Chunk

The INFO-list chunk in a SASBF bitstream element contains three mandatory and a variety of optional subchunks as defined below. The INFO-list chunk gives basic information about the SASBF bank contained in the bitstream element.

5.9.2.2.1 The ifil Subchunk

The ifil subchunk is a mandatory subchunk identifying the SASBF specification version level to which the bitstream element complies. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the SASBF specification version. The WORD wMinor contains the value to the right of the decimal point. For example, version 2.11 would be implied if wMajor=2 and wMinor=11.

These values can be used by applications which read SASBF bitstream elements to determine if the format of the bitstream element is usable by the program. Within a fixed wMajor, the only changes to the format will be the addition of Generator, Source and Transform enumerators, and additional info subchunks. These are all defined as being ignored if unknown to the program.

If the ifil subchunk is missing, or its size is not four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.2.2 The isng Subchunk

The isng subchunk is a mandatory subchunk identifying the wavetable sound engine for which the bitstream element was optimised. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even.

The ASCII should be treated as case-sensitive. In other words “engine” is not the same as “ENGINE.”

The isng string may be optionally used by chip drivers to vary their synthesis algorithms to emulate the target sound engine.

If the isng subchunk is missing not terminated in a zero valued byte, or its contents are an unknown sound engine, the field should be ignored.

5.9.2.2.3 The INAM Subchunk

The INAM subchunk is a mandatory subchunk providing the name of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical inam subchunk would be the fourteen bytes representing “General MIDI” as twelve ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “General MIDI” is not the same as “GENERAL MIDI.”

If the inam subchunk is missing, or not terminated in a zero valued byte, the field should be ignored and the user supplied with an appropriate error message if the name is queried. If the bitstream element is re-written, a valid name should be placed in the INAM field.

5.9.2.2.4 The irom Subchunk

The irom subchunk is an optional subchunk identifying a particular wavetable sound data ROM to which any ROM samples refer. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical irom field would be the six bytes representing “1MGM” as four ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “1mgm” is not the same as “1MGM.”

The irom string is used by drivers to verify that the ROM data referenced by the bitstream element is available to the sound engine.

If the irom subchunk is missing, not terminated in a zero valued byte, or its contents are an unknown ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. A bitstream element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.5 The iver Subchunk

The iver subchunk is an optional subchunk identifying the particular wavetable sound data ROM revision to which any ROM samples refer. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the ROM version. The WORD wMinor contains the value to the right of the decimal point. For example, version 1.36 would be implied if wMajor=1 and wMinor=36.

The iver subchunk is used by drivers to verify that the ROM data referenced by the bitstream element is located in the exact locations specified by the sound headers.

If the iver subchunk is missing, not four bytes in length, or its contents indicate an unknown or incorrect ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. Note that for ROM samples to function correctly, both iver and irom must be present and valid. A bitstream

element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.6 The ICRD Subchunk

The ICRD subchunk is an optional subchunk identifying the creation date of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICRD field would be the twelve bytes representing “May 1, 1995” as eleven ASCII characters followed by a zero byte.

Conventionally, the format of the string is “Month Day, Year” where Month is initially capitalised and is the conventional full English spelling of the month, Day is the date in decimal followed by a comma, and Year is the full decimal year. Thus the field should conventionally never be longer than 32 bytes.

The ICRD string is provided for library management purposes.

If the ICRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.7 The IENG Subchunk

The IENG subchunk is an optional subchunk identifying the names of any sound designers or engineers responsible for the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IENG field would be the twelve bytes representing “Tim Swartz” as ten ASCII characters followed by two zero bytes.

The IENG string is provided for library management purposes.

If the IENG subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.8 The IPRD Subchunk

The IPRD subchunk is an optional subchunk identifying any specific product for which the SASBF bank is intended. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IPRD field would be the twelve bytes representing “MPEG SASBF” as ten ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “mpeg sasbf” is not the same as “MPEG SASBF.”

The IPRD string is provided for library management purposes.

If the IPRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.9 The ICOP Subchunk

The ICOP subchunk is an optional subchunk containing any copyright assertion string associated with the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICOP field would be the 38 bytes representing "Copyright (c) 1998 Content Developer" as 36 ASCII characters followed by two zero bytes.

The ICOP string is provided for intellectual property protection and management purposes.

If the ICOP subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.10 The ICMT Subchunk

The ICMT subchunk is an optional subchunk containing any comments associated with the SASBF bank. It contains an ASCII string of 65,536 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICMT field would be the 40 bytes representing "This space unintentionally left blank." as 38 ASCII characters followed by two zero bytes.

The ICMT string is provided for including comments.

If the ICMT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.11 The ISFT Subchunk

The ISFT subchunk is an optional subchunk identifying the SASBF tools used to create and most recently modify the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ISFT field would be the twenty-six bytes representing "Editor 2.00a:Editor 2.00a" as twenty-five ASCII characters followed by a zero byte.

The ASCII should be treated as case-sensitive. In other words "Editor" is not the same as "EDITOR."

Conventionally, the tool name and revision control number are included first for the creating tool and then for the most recent modifying tool. The two strings are separated by a colon. The string should be produced by the creating program with a null modifying tool field (e.g. "Editor 2.00a:), and each time a tool modifies the bank, it should replace the modifying tool field with its own name and revision control number.

The ISFT string is provided primarily for error tracing purposes.

If the ISFT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.3 The sdta-list Chunk

The sdta-list chunk in a SASBF bitstream element contains a single optional smpl subchunk which contains all the sound data associated with the SASBF bank. The smpl subchunk is of arbitrary length, and contains an even number of bytes.

5.9.2.3.1 Sample Data Format in the smpl Subchunk

The smpl subchunk, contains one or more samples of digital audio information in the form of linearly coded sixteen bit, signed, little endian (least significant byte first) words. Each sample is followed by a minimum of forty-six zero valued sample data points. These zero valued data points are necessary to guarantee that any reasonable upward pitch shift using any reasonable interpolator can loop on zero data at the end of the sound.

5.9.2.3.2 Sample Data Looping Rules

Within each sample, one or more loop point pairs may exist. The locations of these points are defined within the pdta-list chunk, but the sample data points themselves must comply with certain practices in order for the loop to be compatible across multiple platforms.

The loops are defined by “equivalent points” in the sample. This means that there are two sample data points which are logically equivalent, and a loop occurs when these points are spliced atop one another. In concept, the loop end point is never actually played during looping; instead the loop start point follows the point just prior to the loop end point. Because of the bandlimited nature of digital audio sampling, an artefact free loop will exhibit virtually identical data surrounding the equivalent points.

In actuality, because of the various interpolation algorithms used by wavetable synthesisers, the data surrounding both the loop start and end points may affect the sound of the loop. Hence both the loop start and end points must be surrounded by continuous audio data. For example, even if the sound is programmed to continue to loop throughout the decay, sample data points must be provided beyond the loop end point. This data will typically be identical to the data at the start of the loop. A minimum of eight valid data points are required to be present before the loop start and after the loop end.

The eight data points (four on each side) surrounding the two equivalent loop points should also be forced to be identical. By forcing the data to be identical, all interpolation algorithms are guaranteed to properly reproduce an artefact-free loop.

5.9.2.4 The pdta-list Chunk

5.9.2.4.1 The PHDR Subchunk

The PHDR subchunk is a required subchunk listing all presets within the SASBF bitstream element. It shall be a multiple of thirty-eight bytes in length, and shall contain a minimum of two records, one record for each preset and one for a terminal record according to the structure:

```
struct sfPresetHeader
{
       CHAR achPresetName[20];
       WORD wPreset;
       WORD wBank;
       WORD wPresetBagNdx;
       DWORD dwLibrary;
       DWORD dwGenre;
       DWORD dwMorphology;
};
```

The ASCII character field achPresetName shall contain the name of the preset expressed in ASCII, with unused terminal characters filled with zero valued bytes. Preset names are case-sensitive. A unique name shall always be assigned to each preset in the SASBF bank.

The WORD wPreset shall contain the MIDI Preset Number and the WORD wBank the MIDI Bank Number which apply to this preset. Note that the presets are not ordered within the SASBF bank. Presets

shall have a unique set of wPreset and wBank numbers. The special case of a General MIDI percussion bank is handled conventionally by a wBank value of 128. If the value in either field is not a valid MIDI value of 0 through 127, or 128 for wBank, the preset cannot be played but shall be maintained in memory for future renumbering.

The WORD wPresetBagNdx is an index to the preset's zone list in the PBAG subchunk. Because the preset zone list is in the same order as the preset header list, the preset bag indices shall be monotonically increasing with increasing preset headers. The size of the PBAG subchunk in bytes shall be equal to four times the terminal preset's wPresetBagNdx plus four. If the preset bag indices are non-monotonic or if the terminal preset's wPresetBagNdx does not match the PBAG subchunk size, the SASBF chunk is structurally defective and shall be rejected at load time. All presets except the terminal preset shall have at least one zone.

The DWORDs dwLibrary, dwGenre and dwMorphology are reserved for future implementation in a preset library management function and should be preserved as read, and created as zero.

The terminal sfPresetHeader record should never be accessed, and exists only to provide a terminal wPresetBagNdx with which to determine the number of zones in the last preset. All other values shall be conventionally zero, with the exception of achPresetName, which may be optionally be "EOP" indicating end of presets.

If the PHDR subchunk is missing, contains fewer than two records, or its size is not a multiple of 38 bytes, the SASBF bitstream element shall be rejected as structurally unsound.

5.9.2.4.2 The PBAG Subchunk

The PBAG subchunk is a required subchunk listing all preset zones within the SASBF bitstream element. It shall be a multiple of four bytes in length, and shall contain one record for each preset zone plus one record for a terminal zone according to the structure:

```
struct sfPresetBag
{
       WORD wGenNdx;
       WORD wModNdx;
};
```

The first zone in a given preset shall be located at that preset's wPresetBagNdx. The number of zones in the preset shall be determined by the difference between the next preset's wPresetBagNdx and the current wPresetBagNdx.

The WORD wGenNdx shall be an index to the preset's zone list of generators in the PGEN subchunk, and the wModNdx shall be an index to its list of modulators in the PMOD subchunk. Because both the generator and modulator lists are in the same order as the preset header and zone lists, these indices will be monotonically increasing with increasing preset zones. The size of the PMOD subchunk in bytes shall be equal to ten times the terminal preset's wModNdx plus ten and the size of the PGEN subchunk in bytes shall be equal to four times the terminal preset's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective PGEN or PMOD subchunks, the bitstream element is structurally defective and shall be rejected at load time.

If a preset has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not an Instrument generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks an Instrument generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the PBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.3 The PMOD Subchunk

The PMOD subchunk is a required subchunk listing all preset zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The preset zone's wModNdx points to the first modulator for that preset zone, and the number of modulators present for a preset zone is determined by the difference between the next higher preset zone's wModNdx and the current preset's wModNdx. A difference of zero indicates there are no modulators in this preset zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator with the same zone, the first modulator will be ignored.

Modulators in the PMOD subchunk act as additively relative modulators with respect to those in the IMOD subchunk. In other words, a PMOD modulator can increase or decrease the amount of an IMOD modulator.

In SASBF, no modulators have yet been defined, and the PMOD subchunk will always consist of ten zero valued bytes.

If the PMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.4 The PGEN Subchunk

The PGEN chunk is a required chunk containing a list of preset zone generators for each preset zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each preset zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined:

```
typedef struct
{
       BYTE byLo;
       BYTE byHi;
} rangesType;

typedef union
{
       rangesType ranges;
       SHORT shAmount;
       WORD wAmount;
} genAmountType;
```

The sfGenOper is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. This value indicates the type of generator being indicated. Note that this enumeration is two bytes in length.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers or MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The preset zone's wGenNdx points to the first generator for that preset zone. Unless the zone is a global zone, the last generator in the list is an "Instrument" generator, whose value is a pointer to the instrument associated with that zone. If a "key range" generator exists for the preset zone, it is always the first generator in the list for that preset zone. If a "velocity range" generator exists for the preset zone, it will only be preceded by a key range generator. If any generators follow an Instrument generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator with the same zone, the first generator will be ignored.

Generators in the PGEN subchunk are applied relative to generators in the IGEN subchunk in an additive manner. In other words, PGEN generators increase or decrease the value of an IGEN generator.

If the PGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in an instrument generator, the zone should be ignored. If the instrument generator value is equal to or greater than the terminal instrument, the bitstream element should be rejected as structurally unsound.

5.9.2.4.5 The INST Subchunk

The inst subchunk is a required subchunk listing all instruments within the SASBF bitstream element. It is always a multiple of twenty-two bytes in length, and contains a minimum of two records, one record for each instrument and one for a terminal record according to the structure:

```
struct sfInst
{
       CHAR achInstName[20];
       WORD wInstBagNdx;
};
```

The ASCII character field achInstName contains the name of the instrument expressed in ASCII, with unused terminal characters filled with zero valued bytes. Instrument names are case-sensitive. A unique name should always be assigned to each instrument in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of instruments with identical names, the instruments should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The WORD wInstBagNdx is an index to the instrument's zone list in the IBAG subchunk. Because the instrument zone list is in the same order as the instrument list, the instrument bag indices will be monotonically increasing with increasing instruments. The size of the IBAG subchunk in bytes will be four greater than four times the terminal (EOI) instrument's wInstBagNdx. If the instrument bag indices are non-monotonic or if the terminal instrument's wInstBagNdx does not match the IBAG subchunk size, the bitstream element is structurally defective and should be rejected at load time. All instruments except the terminal instrument must have at least one zone; any preset with no zones should be ignored.

The terminal sfInst record should never be accessed, and exists only to provide a terminal wInstBagNdx with which to determine the number of zones in the last instrument. All other values are conventionally zero, with the exception of achInstName, which should be "EOI" indicating end of instruments.

If the INST subchunk is missing, contains fewer than two records, or its size is not a multiple of 22 bytes, the bitstream element should be rejected as structurally unsound. All instruments present in the inst subchunk are typically referenced by a preset zone. However, a bitstream element containing any "orphaned" instruments need not be rejected. SASBF applications can optionally ignore or filter out these orphaned instruments based on user preference.

5.9.2.4.6 The IBAG Subchunk

The IBAG subchunk is a required subchunk listing all instrument zones within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one record for each instrument zone plus one record for a terminal zone according to the structure:

```
struct sfInstBag
{
       WORD wInstGenNdx;
       WORD wInstModNdx;
```

```
};
```

The first zone in a given instrument is located at that instrument's wInstBagNdx. The number of zones in the instrument is determined by the difference between the next instrument's wInstBagNdx and the current wInstBagNdx.

The WORD wInstGenNdx is an index to the instrument zone's list of generators in the IGEN subchunk, and the wInstModNdx is an index to its list of modulators in the IMOD subchunk. Because both the generator and modulator lists are in the same order as the instrument and zone lists, these indices will be monotonically increasing with increasing zones. The size of the IMOD subchunk in bytes will be equal to ten times the terminal instrument's wModNdx plus ten and the size of the IGEN subchunk in bytes will be equal to four times the terminal instrument's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective IGEN or IMOD subchunks, the bitstream element is structurally defective and should be rejected at load time.

If an instrument has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not a sampleID generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks a sampleID generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the IBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.7 The IMOD Subchunk

The IMOD subchunk is a required subchunk listing all instrument zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The zone's wInstModNdx points to the first modulator for that zone, and the number of modulators present for a zone is determined by the difference between the next higher zone's wInstModNdx and the current zone's wModNdx. A difference of zero indicates there are no modulators in this zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator within the same zone, the first modulator will be ignored.

Modulators in the IMOD subchunk are absolute. This means that an IMOD modulator replaces, rather than adds to, a default modulator.

In SASBF, no modulators have yet been defined, and the IMOD subchunk will always consist of ten zero valued bytes.

If the IMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.8 The IGEN Subchunk

The IGEN chunk is a required chunk containing a list of zone generators for each instrument zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfInstGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined as in the PGEN zone above.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers of MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The zone's wInstGenNdx points to the first generator for that zone. Unless the zone is a global zone, the last generator in the list is a "sampleID" generator, whose value is a pointer to the sample associated with that zone. If a "key range" generator exists for the zone, it is always the first generator in the list for that zone. If a "velocity range" generator exists for the zone, it will only be preceded by a key range generator. If any generators follow a sampleID generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator within the same zone, the first generator will be ignored.

Generators in the IGEN subchunk are absolute in nature. This means that an IGEN generator replaces, rather than adds to, the default value for the generator.

If the IGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in a sampleID generator, the zone should be ignored. If the sampleID generator value is equal to or greater than the terminal sampleID, the bitstream element should be rejected as structurally unsound.

5.9.2.4.9 The SHDR Subchunk

The SHDR chunk is a required subchunk listing all samples within the smpl subchunk and any referenced ROM samples. It is always a multiple of forty-six bytes in length, and contains one record for each sample plus a terminal record according to the structure:

```
struct sfSample
{
       CHAR achSampleName[20];
       DWORD dwStart;
       DWORD dwEnd;
       DWORD dwStartloop;
       DWORD dwEndloop;
       DWORD dwSampleRate;
       BYTE byOriginalPitch;
       CHAR chPitchCorrection;
       WORD wSampleLink;
       SFSampleLink sfSampleType;
};
```

The ASCII character field achSampleName contains the name of the sample expressed in ASCII, with unused terminal characters filled with zero valued bytes. Sample names are case-sensitive. A unique name should always be assigned to each sample in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of samples with identical names, the samples should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The DWORD dwStart contains the index, in sample data points, from the beginning of the sample data field to the first data point of this sample.

The DWORD dwEnd contains the index, in sample data points, from the beginning of the sample data field to the first of the set of 46 zero valued data points following this sample.

The DWORD dwStartloop contains the index, in sample data points, from the beginning of the sample data field to the first data point in the loop of this sample.

The DWORD dwEndloop contains the index, in sample data points, from the beginning of the sample data field to the first data point following the loop of this sample. Note that this is the data point “equivalent to” the first loop data point, and that to produce portable artefact free loops, the eight proximal data points surrounding both the Startloop and Endloop points should be identical.

The values of dwStart, dwEnd, dwStartloop, and dwEndloop must all be within the range of the sample data field included in the SASBF bank or referenced in the sound ROM. Also, to allow a variety of hardware platforms to be able to reproduce the data, the samples have a minimum length of 48 data points, a

minimum loop size of 32 data points and a minimum of 8 valid points prior to dwStartloop and after dwEndloop. Thus dwStart must be less than dwStartloop-7, dwStartloop must be less than dwEndloop-31, and dwEndloop must be less than dwEnd-7. If these constraints are not met, the sound may optionally not be played if the hardware cannot support artefact-free playback for the parameters given.

The DWORD dwSampleRate contains the sample rate, in hertz, at which this sample was acquired or to which it was most recently converted. Values of greater than 50000 or less than 400 may not be reproducible by some hardware platforms and should be avoided. A value of zero is illegal. If an illegal or impractical value is encountered, the nearest practical value should be used.

The BYTE byOriginalPitch contains the MIDI key number of the recorded pitch of the sample. For example, a recording of an instrument playing middle C (261.62 Hz) should receive a value of 60. This value is used as the default “root key” for the sample, so that in the example, a MIDI key-on command for note number 60 would reproduce the sound at its original pitch. For unpitched sounds, a conventional value of 255 should be used. Values between 128 and 254 are illegal. Whenever an illegal value or a value of 255 is encountered, the value 60 should be used.

The CHAR chPitchCorrection contains a pitch correction in cents which should be applied to the sample on playback. The purpose of this field is to compensate for any pitch errors during the sample recording process. The correction value is that of the correction to be applied. For example, if the sound is 4 cents sharp, a correction bringing it 4 cents flat is required; thus the value should be -4.

The value in sfSampleType is an enumeration with eight defined values: monoSample = 1, rightSample = 2, leftSample = 4, linkedSample = 8, RomMonoSample = 32769, RomRightSample = 32770, RomLeftSample = 32772, and RomLinkedSample = 32776. It can be seen that this is encoded such that bit 15 of the 16 bit value is set if the sample is in ROM, and reset if it is included in the SASBF bank. The four LS bits of the word are then exclusively set indicating mono, left, right, or linked.

If the sound is flagged as a ROM sample and no valid “irom” subchunk is included; the bitstream element is structurally defective and should be rejected at load time.

If sfSampleType indicates a mono sample, then wSampleLink is undefined and its value should be conventionally zero, but will be ignored regardless of value. If sfSampleType indicates a left or right sample, then wSampleLink is the sample header index of the associated right or left stereo sample respectively. Both samples should be played entirely synchronously, with their pitch controlled by the right sample’s generators. All non-pitch generators should apply as normal; in particular the panning of the individual samples to left and right should be accomplished via the pan generator. Left-right pairs should always be found within the same instrument. Note also that no instrument should be designed in which it is possible to activate more than one instance of a particular stereo pair. The linked sample type is not currently fully defined in the SASBF specification, but will ultimately support a circularly linked list of samples using wSampleLink. Note that this enumeration is two bytes in length.

The terminal sample record is never referenced, and is conventionally entirely zero with the exception of achSampleName, which should be “EOS” indicating end of samples. All samples present in the smpl subchunk are typically referenced by an instrument, however a bitstream element containing any “orphaned” samples need not be rejected. SASBF applications can optionally ignore or filter out these orphaned samples according to user preference.

If the SHDR subchunk is missing, or its is size is not a multiple of 46 bytes the bitstream element should be rejected as structurally unsound.

5.9.3 Enumerators

5.9.3.1 Generator Enumerators

Subclause 7.1 defines the generator and generator kinds. Subclause 8.4 defines the generator operation model.

5.9.3.1.1 Kinds of Generator Enumerators

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

An Index Generator's amount is an index into another data structure. The only two Index Generators are Instrument and sampleID.

A Range Generator defines a range of note-on parameters outside of which the zone is undefined. Two Range Generators are currently defined, keyRange and velRange.

Substitution Generators are generators which substitute a value for a note-on parameter. Two Substitution Generators are currently defined, overridingKeyNumber and overridingVelocity.

Sample Generators are generators which directly affect a sample's properties. These generators are undefined at the preset level. The currently defined Sample Generators are the eight address offset generators, the sampleModes generator, the Overriding Root Key generator and the Exclusive Class generator.

Value Generators are generators whose value directly affects a signal processing parameter. Most generators are value generators.

5.9.3.1.2 Generator Enumerators Defined

The following is an exhaustive list of SASBF generators and their strict definitions:

0	startAddrsOffset	The offset, in sample data points, beyond the Start sample header parameter to the first sample data point to be played for this instrument. For example, if Start were 7 and startAddrsOffset were 2, the first sample data point played would be sample data point 9.
1	endAddrsOffset	The offset, in sample data points, beyond the End sample header parameter to the last sample data point to be played for this instrument. For example, if End were 17 and endAddrsOffset were -2, the last sample data point played would be sample data point 15.
2	startloopAddrsOffset	The offset, in sample data points, beyond the Startloop sample header parameter to the first sample data point to be repeated in the loop for this instrument. For example, if Startloop were 10 and startloopAddrsOffset were -1, the first repeated loop sample data point would be sample data point 9.
3	endloopAddrsOffset	The offset, in sample data points, beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. For example, if Endloop were 15 and endloopAddrsOffset were 2, sample data point 17 would be considered equivalent to the

		Startloop sample data point, and hence sample data point 16 would effectively precede Startloop during looping.
4	startAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Start sample header parameter and the first sample data point to be played in this instrument. This parameter is added to the startAddrsOffset parameter. For example, if Start were 5, startAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point played would be sample data point 65544.
5	modLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
6	vibLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Vibrato LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
7	modEnvToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence pitch. A positive value indicates an increase in pitch; a negative value indicates a decrease in pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will rise 1 semitone at the envelope peak.
8	initialFilterFc	This is the cutoff and resonant frequency of the lowpass filter in absolute cent units. The lowpass filter is defined as a second order resonant pole pair whose pole frequency in Hz is defined by the Initial Filter Cutoff parameter. When the cutoff frequency exceeds 20kHz and the Q (resonance) of the filter is zero, the filter does not affect the signal.
9	initialFilterQ	This is the height above DC gain in centibels which the filter resonance exhibits at the cutoff frequency. A value of zero or less indicates the filter is not resonant; the gain at the cutoff frequency (pole angle) may be less than zero when zero is specified. The filter gain at DC is also affected by this parameter such that the gain at DC is reduced by half the specified gain. For example, for a value of 100, the filter gain at DC would be 5 dB below unity gain, and the height of the resonant peak would be 10 dB above the DC gain, or 5 dB above unity gain. Note also that if initialFilterQ is set to zero or less and the cutoff frequency exceeds 20 kHz, then the filter response is flat and unity gain.

10	modLfoToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence filter cutoff frequency. A positive number indicates a positive LFO excursion increases cutoff frequency; a negative number indicates a positive excursion decreases cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1200 indicates that the cutoff frequency will first rise 1 octave, then fall one octave.
11	modEnvToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence filter cutoff frequency. A positive number indicates an increase in cutoff frequency; a negative number indicates a decrease in filter cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1000 indicates that the cutoff frequency will rise one octave at the envelope attack peak.
12	endAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the End sample header parameter and the last sample data point to be played in this instrument. This parameter is added to the endAddrsOffset parameter. For example, if End were 65536, startAddrsOffset were -3 and startAddrsCoarseOffset were -1, the last sample data point played would be sample data point 32765.
13	modLfoToVolume	This is the degree, in centibels, to which a full scale excursion of the Modulation LFO will influence volume. A positive number indicates a positive LFO excursion increases volume; a negative number indicates a positive excursion decreases volume. Volume is always modified logarithmically, that is the deviation is in decibels rather than in linear amplitude. For example, a value of 100 indicates that the volume will first rise ten dB, then fall ten dB.
14	unused1	Unused, reserved. Should be ignored if encountered.
15	chorusEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the chorus effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the chorus effects processor.
16	reverbEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the reverb effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the reverb effects processor.

17	pan	This is the degree, in 0.1% units, to which the “dry” audio output of the note is positioned to the left or right output. A value of -50% or less indicates the signal is sent entirely to the left output and not sent to the right output; a value of +50% or more indicates the signal is sent entirely to the right and not sent to the left. A value of zero sends the signal equally to left and right. For example, a value of -250 indicates that the signal is sent at 75% of full level to the left output and 25% of full level to the right output.
18	unused2	Unused, reserved. Should be ignored if encountered.
19	unused3	Unused, reserved. Should be ignored if encountered.
20	unused4	Unused, reserved. Should be ignored if encountered.
21	delayModLFO	This is the delay time, in absolute timecents, from key on until the Modulation LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second and a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
22	freqModLFO	This is the frequency, in absolute cents, of the Modulation LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 MHz would be 1200log2(.01/8.176) = -11610.
23	delayVibLFO	This is the delay time, in absolute timecents, from key on until the Vibrato LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
24	freqVibLFO	This is the frequency, in absolute cents, of the Vibrato LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 mHz would be 1200log2(.01/8.176) = -11610.
25	delayModEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Modulation envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
26	attackModEnv	This is the time, in absolute timecents, from the end of the Modulation Envelope Delay Time until the point at which the Modulation Envelope value reaches its peak. Note that the attack is

		"convex"; the curve is nominally such that when applied to a decibel or semitone parameter, the result is linear in amplitude or Hz respectively. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
27	holdModEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
28	decayModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during decay phase. For the Modulation Envelope, the decay phase linearly ramps toward the sustain level. If the sustain level were zero, the Modulation Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
29	sustainModEnv	This is the decrease in level, expressed in 0.1% units, to which the Modulation Envelope value ramps during the decay phase. For the Modulation Envelope, the sustain level is properly expressed in percent of full scale. Because the volume envelope sustain level is expressed as an attenuation from full scale, the sustain level is analogously expressed as a decrease from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; values above 1000 are to be interpreted as 1000. For example, a sustain level which corresponds to an absolute value 40% of peak would be 600.
30	releaseModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during release phase. For the Modulation Envelope, the release phase linearly ramps toward zero from the current level. If the current level were full scale, the Modulation Envelope Release Time would be the time spent in release phase until zero value were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
31	keynumToModEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave

		causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
32	keynumToModEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
33	delayVolEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Volume envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
34	attackVolEnv	This is the time, in absolute timecents, from the end of the Volume Envelope Delay Time until the point at which the Volume Envelope value reaches its peak. Note that the attack is “convex”; the curve is nominally such that when applied to the decibel volume parameter, the result is linear in amplitude. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
35	holdVolEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the Volume envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
36	decayVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during decay phase. For the Volume Envelope, the decay phase linearly ramps toward the sustain level, causing a constant dB change for each time unit. If the sustain level were -96dB, the Volume Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
37	sustainVolEnv	This is the decrease in level, expressed in centibels, to which the Volume Envelope value ramps during the decay phase. For the

		Volume Envelope, the sustain level is best expressed in centibels of attenuation from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; conventionally 1000 indicates full attenuation. For example, a sustain level which corresponds to an absolute value 12dB below of peak would be 120.
38	releaseVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during release phase. For the Volume Envelope, the release phase linearly ramps toward zero from the current level, causing a constant dB change for each time unit. If the current level were full scale, the Volume Envelope Release Time would be the time spent in release phase until 96dB attenuation were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
39	keynumToVolEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
40	keynumToVolEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
41	instrument	This is the index into the INST subchunk providing the instrument to be used for the current preset zone. A value of zero indicates the first instrument in the list. The value should never exceed two less than the size of the instrument list. The instrument enumerator is the terminal generator for PGEN zones. As such, it should only appear in the PGEN subchunk, and it must appear as the last generator enumerator in all but the global preset zone.
42	reserved1	Unused, reserved. Should be ignored if encountered.
43	keyRange	This is the minimum and maximum MIDI key number values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid key. The

		keyRange enumerator is optional, but when it does appear, it must be the first generator in the zone generator list.
44	velRange	This is the minimum and maximum MIDI velocity values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid velocity. The velRange enumerator is optional, but when it does appear, it must be preceded only by keyRange in the zone generator list.
45	startloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Startloop sample header parameter and the first sample data point to be repeated in this instrument's loop. This parameter is added to the startloopAddrsOffset parameter. For example, if Startloop were 5, startloopAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point in the loop would be sample data point 65544.
46	keynum	This enumerator forces the MIDI key number to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
47	velocity	This enumerator forces the MIDI velocity to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
48	initialAttenuation	This is the attenuation, in .4 centibel units, by which a note is attenuated below full scale. A value of zero indicates no attenuation; the note will be played at full scale. For example, a value of 60 indicates the note will be played at 2.4 dB below full scale for the note.
49	reserved2	Unused, reserved. Should be ignored if encountered.
50	endloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. This parameter is added to the endloopAddrsOffset parameter. For example, if Endloop were 5, endloopAddrsOffset were 3 and endAddrsCoarseOffset were 2, sample data point 65544 would be considered equivalent to the Startloop sample data point, and hence sample data point 65543 would effectively precede Startloop during looping.
51	coarseTune	This is a pitch offset, in semitones, which should be applied to the note. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Coarse Tune value of -4 would cause the sound to be reproduced four semitones flat.
52	fineTune	This is a pitch offset, in cents, which should be applied to the note. It is additive with coarseTune. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Fine Tuning value of -5 would cause the sound to be reproduced five cents flat.

53	sampleID	This is the index into the SHDR subchunk providing the sample to be used for the current instrument zone. A value of zero indicates the first sample in the list. The value should never exceed two less than the size of the sample list. The sampleID enumerator is the terminal generator for IGEN zones. As such, it should only appear in the IGEN subchunk, and it must appear as the last generator enumerator in all but the global zone.
54	sampleModes	This enumerator indicates a value which gives a variety of Boolean flags describing the sample for the current instrument zone. The sampleModes should only appear in the IGEN subchunk, and should not appear in the global zone. The two LS bits of the value indicate the type of loop in the sample: 0 indicates a sound reproduced with no loop, 1 indicates a sound which loops continuously, 2 is unused but should be interpreted as indicating no loop, and 3 indicates a sound which loops for the duration of key depression then proceeds to play the remainder of the sample.
55	reserved3	Unused, reserved. Should be ignored if encountered.
56	scaleTuning	This parameter represents the degree to which MIDI key number influences pitch. A value of zero indicates that MIDI key number has no effect on pitch; a value of 100 represents the usual tempered semitone scale.
57	exclusiveClass	This parameter provides the capability for a key depression in a given instrument to terminate the playback of other instruments. This is particularly useful for percussive instruments such as a hi-hat cymbal. An exclusive class value of zero indicates no exclusive class; no special action is taken. Any other value indicates that when this note is initiated, any other sounding note with the same exclusive class value should be rapidly terminated. The exclusive class generator can only appear at the instrument level. The scope of the exclusive class is the entire preset. In other words, any other instrument zone within the same preset holding a corresponding exclusive class will be terminated.
58	overridingRootKey	This parameter represents the MIDI key number at which the sample is to be played back at its original sample rate. If not present, or if present with a value of -1, then the sample header parameter Original Key is used in its place. If it is present in the range 0-127, then the indicated key number will cause the sample to be played back at its sample header Sample Rate. For example, if the sample were a recording of a piano middle C (Original Key = 60) at a sample rate of 22.050 kHz, and Root Key were set to 69, then playing MIDI key number 69 (A above middle C) would cause a piano note of pitch middle C to be heard.
59	unused5	Unused, reserved. Should be ignored if encountered.
60	endOper	Unused, reserved. Should be ignored if encountered. Unique name provides value to end of defined list.

5.9.3.1.3 Generator Summary

The following tables give the ranges and default values for all SASBF defined generators.

#	Name	Unit	Abs Zero	Min	Min Useful	Max	Max Useful	De-fault	Def Value
0	startAddrsOffset +	smpls	0	0	None	*	*	0	None
1	endAddrsOffset +	smpls	0	*	*	0	None	0	None
2	startloopAddrsOffset +	smpls	0	*	*	*	*	0	None
3	endloopAddrsOffset +	smpls	0	*	*	*	*	0	None
4	startAddrsCoarseOffset +	32k	0	0	None	*	*	0	None
5	modLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
6	vibLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
7	modEnvToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
8	initialFilterFc	cent	8.176 Hz	1500	20 Hz	13500	20 kHz	13500	Open
9	initialFilterQ	cB	0	0	None	960	96 dB	0	None
10	modLfoToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
11	modEnvToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
12	endAddrsCoarseOffset +	32k	0	*	*	0	None	0	None
13	modLfoToVolume	cB fs	0	-960	-96 dB	960	96 dB	0	None
15	chorusEffectsSend	0.1%	0	0	None	1000	100%	0	None
16	reverbEffectsSend	0.1%	0	0	None	1000	100%	0	None
17	pan	0.1%	Cntr	-500	Left	+500	Right	0	Centre
21	delayModLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
22	freqModLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
23	delayVibLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
24	freqVibLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
25	delayModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
26	attackModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
27	holdModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
28	decayModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
29	sustainModEnv	-0.1%	attk peak	0	100%	1000	0%	0	attk pk
30	releaseModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
31	keynumToModEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
32	keynumToModEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
33	delayVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
34	attackVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
35	holdVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
36	decayVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
37	sustainVolEnv	cB attn	attk peak	0	0 dB	1440	144dB	0	attk pk
38	releaseVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
39	keynumToVolEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
40	keynumToVolEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
43	keyRange	MIDI ky#	key# 0	0	lo key	127	hi key	0-127	full kbd
44	velRange	MIDI vel	0	0	min vel	127	max vel	0-127	all vels
45	startloopAddrsCoarseOffset +	smpl	0	*	*	*	*	0	None

46	keynum +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None
47	velocity +	MIDI vel	0	1	min vel	127	max vel	-1	None
48	initialAttenuation	.4 cB	0	0	0 dB	1440	144dB	0	None
50	endloopAddrsCoarseOffset	smpls	0	*	*	*	*	0	None
51	CoarseTune	semitone	0	-120	-10 oct	120	10 oct	0	None
52	fineTune	cent	0	-99	-99cnt	99	99cent	0	None
54	sampleModes +	Bit Flags	Flags	**	**	**	**	0	No Loop
56	scaleTuning	cent/key	0	0	none	1200	oct/ky	100	semi-tone
57	exclusiveClass +	arbitrary #	0	1	--	127	--	0	None
58	overridingRootKey +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None

* Range depends on values of start, loop, and end points in sample header.

** Range has discrete values based on bit flags

\+ This generator is only valid at the instrument level.

5.9.3.2 Default Modulators

The “default” modulators are described below.

5.9.3.2.1 MIDI Key Velocity to Initial Attenuation

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.2 MIDI Key Velocity to Filter Cutoff

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is -2400 Cents.

The product of these values is added to the Initial Filter Cutoff generator summing node.

5.9.3.2.3 MIDI Channel Pressure to Vibrato LFO Pitch Depth

The MIDI Channel Pressure data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 50 cents per max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.4 MIDI Continuous Controller 1 to Vibrato LFO Pitch Depth

The MIDI Continuous Controller 1 data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. The MIDI Continuous Controller 33 data value may be optionally used for increased resolution of the controller input.

There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 50 cents/max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.5 MIDI Continuous Controller 7 to Initial Attenuation

The MIDI Continuous Controller 7 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.6 MIDI Continuous Controller 10 to Pan Position

The MIDI Continuous Controller 10 data value is used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 127 is mapped to 63/64 and all other values are mapped between -1 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 1000 tenths of a percent panned-right.

The product of these values is added to the Pan generator summing node.

5.9.3.2.7 MIDI Continuous Controller 11 to Initial Attenuation

The MIDI Continuous Controller 11 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.8 MIDI Continuous Controller 91 to Reverb Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added reverb send.

The product of these values is added to the Reverb Send generator summing node.

5.9.3.2.9 MIDI Continuous Controller 93 to Chorus Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added chorus send.

The product of these values is added to the Chorus Send generator summing node.

5.9.3.2.10 MIDI Pitch Wheel to Initial Pitch Controlled by MIDI Pitch Wheel Sensitivity

The MIDI Pitch Wheel data values are used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 16383 is mapped to 8191/8192 and all other values are mapped between -1 and 8191/8192 in a linear fashion.

The MIDI Pitch Wheel Sensitivity data values are used as a secondary source. This source is Positive Unipolar, thus an input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion.

The amount of this modulator is 12700 Cents.

The product of these values is added to the Initial Pitch generator summing node.

5.9.3.3 Precedence and Absolute and Relative values.

Most SASBF generators are available at both the Instrument and Preset Levels, as well as having a default value. Generators at the Instrument Level are considered “absolute” and determine an actual physical value for the associated synthesis parameter, which is used instead of the default. For example, a value of 1200 for the attackVolEnv generator would produce an absolute time of 1200 timecents or 2 seconds of attack time for the volume envelope, instead of the default value of -12000 timecents or 1 msec.

Generators at the Preset Level are instead considered “relative” and additive to all the default or instrument level generators within the Preset Zone. For example, a value of 2400 timecents for the attackVolEnv generator in a preset zone containing an instrument with two zones, one with the default attackVolEnv and one with an absolute attackVolEnv generator value of 1200 timecents would cause the default zone to actually have a value of -9600 timecents or 4 msec, and the other to have a value of 3600 timecents or 8 seconds attack time.

There are some generators which are not available at the Preset Level. These are:

#	Name
0	startAddrsOffset
1	endAddrsOffset
2	startloopAddrsOffset
3	endloopAddrsOffset
4	startAddrsCoarseOffset
12	endAddrsCoarseOffset
45	startloopAddrsCoarseOffset
46	keynum
47	velocity
50	endloopAddrsCoarseOffset
54	sampleModes

57	exclusiveClass
58	overridingRootKey

If these generators are encountered in the Preset Level, they should be ignored.

The effect of modulators on a given destination is always relative to the generator value at the Instrument level. However modulators may supersede or add to other modulators depending on their position within the hierarchy. Please see Subclause 8.4 for details on the Modulator implementation and the hierarchical details.

5.9.4 Parameters and Synthesis Model

The SASBF standard has been established with the intent of providing support for an expanding base of wavetable based synthesis models. The model supported by the SASBF specification originates with the EMU8000 wavetable synthesiser chip. The description below of the underlying synthesis model and the associated parameters are provided to allow mapping of this synthesis model onto other hardware platforms.

5.9.4.1 Synthesis Model

The SASBF specification Synthesis Model comprises a wavetable oscillator, a dynamic lowpass filter, an enveloping amplifier, and programmable sends to pan, reverb, and chorus effects units. An underlying modulation engine comprises two low frequency oscillators (LFOs) and two envelope generators with appropriate routing amplifiers.

5.9.4.1.1 Wavetable Oscillator/Interpolator

The SASBF specification wavetable oscillator model is capable of playing back a sample at an arbitrary sampling rate with an arbitrary pitch shift. In practice, the upward pitch shift (downward sample rate conversion) will be limited to a maximum value, typically at least two octaves. The pitch is described in terms of an initial pitch shift which is based on the sample's sampling rate, the root key at which the sample should be unshifted on the keyboard, the coarse, fine, and correction tunings, the effective MIDI key number, and the keyboard scale factor. All modulations in pitch are in octaves, semitones, and cents.

In general, interpolators have a symmetric impulse response, and thus a linear phase characteristic. The interpolation filter's magnitude response can be characterised by three regions - the pass band, the transition band, and the stop band.

The interpolation filter's pass band is the portion of its response which corresponds to the frequencies of the signal stored in waveform memory from DC to that signal's Nyquist frequency

The interpolation filter's transition band is the portion of its response which corresponds to the first image of the signal stored in waveform memory, that is from that signal's Nyquist frequency back to DC.

The interpolation filter's stop band is the portion of its response which corresponds to the remaining images above the transition band to the Nyquist frequency of the upsampled signal.

The guardband will be assumed to begin at 5/6 of the Nyquist frequency, as is the case for a 20 kHz bandwidth in a 48 kHz sample rate system.

Due to poor aliasing rejection in the stopband, linear interpolation does not satisfy this specification.

The specification constrains the Fourier transform of the impulse response of the interpolator. The impulse response is taken with a downward pitch shift of eight octaves. This gives a response which has been upsampled by a factor of 2**8 or 256. In other words, a frequency of the impulse response's Nyquist

frequency divided by 256 gives the filter frequency corresponding to the waveform memory's Nyquist frequency. In the specification below, Fn represents this waveform memory Nyquist frequency.

(All responses measured with downward pitch shift of 8 octaves.)	
Passband Response:	
Ripple:	No more than +/- 0.5 dB
Roll-off:	Minimal, but not to exceed 6 dB at 83.3% Fn.
Transition Band Response:	
Roll-off:	Monotonic and as rapid as possible to at least -80 dB attenuation
Return Lobes:	None above -80 dB
Attenuation:	Greater than 80 dB for all frequencies DC to at least 2% Fn in the first lobe.
Stop Band Response:	
Attenuation:	Greater than 90 dB for all frequencies DC to at least 1% Fn
	Greater than 80 dB for all frequencies DC to at least 20%Fn
	Greater than 60 dB for all frequencies

5.9.4.1.2 Sample Looping

The wavetable oscillator is playing a digital sample which is described in terms of a start point, end point, and two points describing a loop. The sound can be flagged as unlooped, in which case the loop points are ignored. If the sound is looped, it can be played in two ways. If it is flagged as "loop during release", the sound is played from the start point through the loop, and loops until the note becomes inaudible. If not, the sound is played from the start point through the loop, and loops until the key is released. At this point, the next time the loop end point is reached, the sound continues through the loop end point and plays until the end point is reached, at which time audio is terminated.

5.9.4.1.3 Lowpass Filter

The synthesis model contains a resonant lowpass filter, which is characterised by a dynamic cutoff frequency and a fixed resonance (Q).

The filter is idealised at zero resonance as having a flat passband to the cutoff frequency, then a rolloff at 12 dB per octave above that frequency. The resonance, when non-zero, comprises a peak at the cutoff frequency, superimposed on the above response. The resonance is measured as a dB ratio of the resonant peak to the DC gain. The DC gain at any resonance is half of the resonance value below the DC gain at zero resonance; hence the peak height is half the resonance value above DC gain at zero resonance.

All modulations in cutoff frequency are in cents.

Topology: 2nd order AR lowpass filter or equivalent. Filter coefficients are updated in a smooth manner at the sample rate.

Noise and Distortion: Less than 0.003% of full scale on any sinusoidal input for any parameter setting.

Control Parameters: Cutoff Frequency and Resonance.

Resonance Parameter: Specifies the ratio in decibels of the gain of a resonant peak to the gain at DC, when the filter cutoff frequency is set to 1.5 kHz and the modulation is zero. The DC gain of a filter is reduced from the DC gain of a filter with zero resonance by half the resonance value. The resonance parameter will remain fixed throughout the sounding of a musical note.

For a specified resonance, the actual gain ratio should be accurate within +/- 1 dB, and the DC gain accurate within +/-0.5 dB. The ratio of the gain at the resonant peak to the DC gain for all cutoff frequency values shall not deviate by more than 2dB per octave deviation from 1.5 kHz. Nominal gain at DC for a minimum resonance parameter is unity gain.

Cutoff Frequency Parameter: Specifies the frequency at which the filter achieves exactly 3dB of attenuation. The Cutoff Frequency is computed by the combination of the Initial Cutoff Frequency, specified in Hz, and the modulation, specified in semitones and fractions thereof. The Initial Cutoff Frequency is fixed throughout the duration of the a musical note; the modulation can vary at the sample rate. Within the region from 200 Hz to 6.4 kHz, the cutoff frequency parameter accuracy will be +/- 2 semitones.

Frequency Magnitude Response: When the cutoff frequency is at its maximum value and the resonance is at zero, the filter must pass audio without alteration. The filter must support cutoff frequencies down to 200 Hz. There must be a minimum of 2048 distinct cutoff frequencies per octave within the region from 200 Hz to 6.4 kHz, with a worst case spacing between distinct frequencies of 0.01 semitones.

5.9.4.1.4 Final Gain Amplifier

The final gain amplifier is a multiplier on the filter output, which is controlled by an initial gain in dB. This is added to the volume envelope. Additional modulation can also be added. The gain is always specified in dB.

5.9.4.1.5 Effects Sends

The output of the final gain amplifier can be routed into the effects unit. This unit causes the sound to be located (panned) in the stereo field, and a degree of reverberation and chorus to be added. The pan is specified in terms of percentage left and right, which also could be considered as an azimuth angle. The reverb and chorus sends are specified as a percentage of the signal amplitude to be sent to these units, from 0% to 100%.

5.9.4.1.6 Low Frequency Oscillators

The synthesis model provides for two low frequency oscillators (LFOs) for modulating pitch, filter cutoff, and amplitude. The “vibrato” LFO is only capable of modulating pitch. The “modulation” LFO can modulate any of the three parameters.

An LFO is defined as having a delay period during which its value remains zero, followed by a triangular waveshape ramping linearly to positive one, then downward to negative 1, then upward again to positive one, etc.

Each parameter can be modulated to a varying degree, either positively or negatively, by the associated LFO. Modulations of pitch and cutoff are in octaves, semitones, and cents, while modulations of amplitude are in dB. The degree of modulation is specified in cents or dB for the full scale positive LFO excursion.

5.9.4.1.7 Envelope Generators

The synthesis model provides for two envelope generators. The volume envelope generator controls the final gain amplifier and hence determines the volume contour of the sound. The modulation envelope can control pitch and/or filter cutoff.

An envelope generates a control signal in six phases. When key-on occurs, a delay period begins during which the envelope value is zero. The envelope then rises in a convex curve to a value of one during the attack phase. When a value of one is reached, the envelope enters a hold phase during which it remains at one. When the hold phase ends, the envelope enters a decay phase during which its value decreases linearly to a sustain level. When the sustain level is reached, the envelope enters sustain phase, during which the envelope stays at the sustain level. Whenever a key-off occurs, the envelope immediately enters a release phase during which the value linearly ramps from the current value to zero. When zero is reached, the envelope value remains at zero.

Modulation of pitch and filter cutoff are in octaves, semitones, and cents. These parameters can be modulated to varying degree, either positively or negatively, by the modulation envelope. The degree of modulation is specified in cents for the full scale attack peak.

The volume envelope operates in dB, with the attack peak providing a full scale output, appropriately scaled by the initial volume. The zero value, however, is actually zero gain. When 96 dB of attenuation is reached in the final gain amplifier, an abrupt jump to zero gain (infinite dB of attenuation) occurs. In a 16-bit system, this jump is inaudible.

5.9.4.1.8 Modulation Interconnection Summary

The following diagram shows the interconnections expressed in the SASBF specification synthesis model:

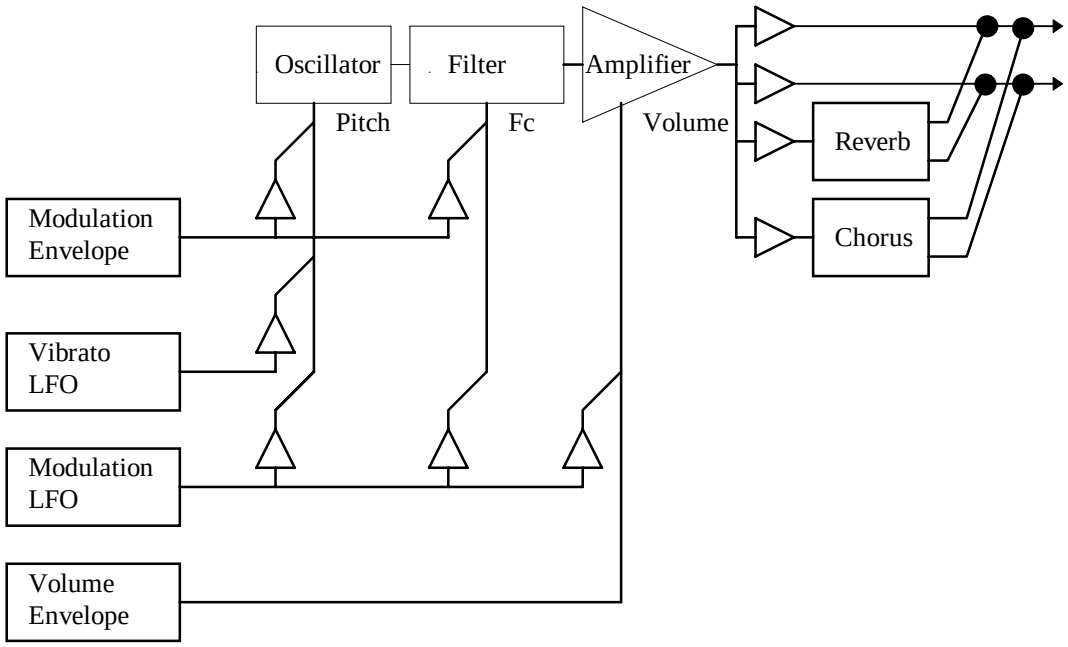


Figure 1: Generator Based Modulation Structure

5.9.4.2 MIDI Functions

The response to certain MIDI commands is defined within the MIDI specification, and therefore not considered to be part of the SASBF specification. These MIDI commands may not be used as sources for the Modulator implementation.

For completeness, the expected responses are given here.

MIDI CC0 Bank Select - When received, the following program change should select the MIDI program in this bank value instead of the default bank of 0.

MIDI CC6 - Data Entry MSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism depending on the Data Entry mode.

MIDI CC32 Bank Select LSB - When received, may behave in conjunction with CC0 Bank Select to provide a total of 16384 possible MIDI banks of programs.

MIDI CC38 Data Entry LSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism, depending on the Data Entry mode.

MIDI CC64 Sustain - ACTIVE when greater than or equal to 64. When the sustain function is active, all notes in the key-on state remain in the key-on state regardless of whether a key-off command for the note arrives. The key-off commands are stored, and when sustain becomes inactive, all stored key-off commands are executed.

MIDI CC66 Soft - ACTIVE when greater than or equal to 64. When active, all new key-ons are modulated in such a way to make the note sound “soft.” This typically affects initial attenuation and filter cutoff in a pre-defined manner.

MIDI CC67 Sostenuto - ACTIVE when greater than or equal to 64. When sostenuto becomes active, all notes currently in the key-on state remain in the key-on state until the sostenuto becomes inactive. All other notes behave normally. Notes maintained by sostenuto in key-on state remain in key-on state even if sustain is switched on and off.

MIDI CC98 NRPN LSB - When received, should be processed by the NRPN implementation mechanism.

MIDI CC99 NRPN MSB - When received, should put the synthesiser in NRPN Data Entry mode and then should be processed by the NRPN implementation mechanism.

MIDI CC100 RPN LSB - When received, should be processed by the RPN implementation mechanism.

MIDI CC101 RPN MSB - When received, should put the synthesiser in RPN Data Entry mode and then should be processed by the RPN implementation mechanism.

MIDI CC120 All Sound Off - When received with any data value, all notes playing in the key-on state bypass the release phase and are shut off, regardless of the sustain or sostenuto positions.

MIDI CC121 Reset All Controllers - Defined as Reset All Controllers as defined by the MIDI specification.

MIDI CC123 All Notes Off - When received with any data value, all notes playing in the key-on state immediately enter release phase, pending their status in SUSTAIN or SOSTENUTO state.

5.9.4.3 Parameter Units

The units with which SASBF generators are described are all well defined. The strict definitions appear below:

ABSOLUTE SAMPLE DATA POINTS - A numeric index of 16 bit sample data point words as stored in ROM or supplied in the smpl-ck, indexing the first sample data point word of memory or the chunk as zero.

RELATIVE SAMPLE DATA POINTS - A count of 16 bit sample data point words based on an absolute sample data point reference. A negative value implies a relative count toward the beginning of the data.

ABSOLUTE SEMITONES - An absolute logarithmic measure of frequency based on a reference of MIDI key numbers. A semitone is 1/12 of an octave, and value 69 is 440 Hz (A-440). Negative values and values above 127 are allowed.

RELATIVE SEMITONES - A relative logarithmic measure of frequency ratio based on units of 1/12 of an octave, which is the twelfth root of two, approximately 1.059463094.

ABSOLUTE CENTS - An absolute logarithmic measure of frequency based on a reference of MIDI key number scaled by 100. A cent is 1/1200 of an octave, and value 6900 is 440 Hz (A-440). Negative values and values above 12700 are allowed.

RELATIVE CENTS - A relative logarithmic measure of frequency ratio based on units of 1/1200 of an octave, which is the twelve hundredth root of two, approximately 1.000577790.

ABSOLUTE CENTIBELS - An absolute measure of the attenuation of a signal, based on a reference of zero being no attenuation. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

RELATIVE CENTIBELS - A relative measure of the attenuation of a signal. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

ABSOLUTE TIMECENTS - An absolute measure of time, based on a reference of zero being one second. A timecent represents a ratio in time of the twelve hundredth root of two, approximately 1.011579454.

RELATIVE TIMECENTS - A relative measure of time ratio, based on a unit size of the twelve hundredth root of two, approximately 1.011579454.

ABSOLUTE PERCENT - An absolute measure of gain, based on a reference of unity. In SASBF, absolute percent is measured in 0.1% units, so a value of zero is 0% and a value of 1000 is 100%.

RELATIVE PERCENT - A relative measure of gain difference. In SASBF, relative percent is measured in 0.1% units. When the gain goes below zero, zero is assumed; when the gain exceeds 100%, 100% is used.

5.9.4.4 The SASBF Generator Model

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

In case it is not clear in the general description of the SASBF hierarchy, the following is the precedence of SASBF generator in the hierarchy.

- A ‘generator’ sets or offsets the value of a destination or a synthesis parameter. In exception cases, it sets ranges (Range Generators), or sets values and never offsets values (Index Generators, Sample Generators, and Substitution Generators).
- A generator is defined as identical to another generator if its generator operator is the same in both generators.
- A generator in a global instrument zone which is identical to a default generator supersedes or replaces the default generator.
- A generator in a local instrument zone which is identical to a default generator or to a generator in a global instrument zone supersedes or replaces that generator.
- Points below (until noted) apply to Value Generators ONLY.
- A generator at the preset level adds to a generator at the instrument level if both generators are identical.
- A generator in a global preset zone which is identical to a default generator or to a generator in an instrument adds to that generator.
- A generator in a global preset zone which is not identical to a default generator and is not identical to a generator in an instrument has its effect added to the given synthesis parameter.
- A generator in a local preset zone which is identical to a generator in a global preset zone supersedes or replaces that generator in the global preset zone. That generator then has its effects added to the destination summing node of all zones in the given instrument.
- A generator in a local preset zone which is not identical to a default generator or a generator in a global preset zone has its effects added to the destination summing node of all zones in the given instrument.
  If the generator operator is a Range Generator, the generator values are NOT ADDED to those

in the instrument level, rather they serve as an intersection filter to those key number or velocity ranges in the instrument which is used in the preset zone.

- If the generator operator is a Substitution Generator or a Sample Generator, they are illegal at the preset level. The only Index Generator legal at the Preset Level is ‘instrumentID’, whereas the only Index Generator legal at the Instrument Level is ‘sampleID’

5.9.4.5 The SASBF Modulator Controller Model

SASBF Modulators are used to allow real-time control over the sound in sound designer programmable manner. Each instance of a SASBF modulator structure defines a real-time perceptually additive effect to be applied to a given destination or synthesiser parameter.

Modulators provide future extensibility to the SASBF standard. They are not used in this version.

5.9.5 Error Handling

5.9.5.1 Structural Errors

Structural Errors are errors which are determined from the implicit redundancy of the SASBF RIFF bitstream element structure, and indicate that the structure is not intact. Examples are incorrect lengths for the chunks or subchunks, pointers out of valid range, or missing required chunks or subchunks for which no error correction procedure exists.

In all cases, bitstream elements should be checked for structural errors at load time, and if any are found, the bitstream elements should be rejected. Separate tools or options can be used to “repair” structurally defective bitstream elements, but these tools should validate that the reconstructed bitstream element is not only a valid SASBF bank but also complies with the intended timbral results in all cases.

5.9.5.2 Unknown Chunks

In parsing the RIFF structure, unknown but well formed chunks or subchunks may be encountered. Unknown chunks within the INFO-list chunk should simply be ignored. Other unknown chunks or subchunks are illegal and should be treated as structural errors.

5.9.5.3 Unknown Enumerators

Unknown enumerators may be encountered in Generators, Modulator Sources, or Transforms. This is to be expected if the ifil field exceeds the specification to which the application was written. Even if unexpected, unknown enumerators should simply cause the associated Generator or Modulator to be ignored.

5.9.5.4 Illegal Parameter Values

Some SASBF parameters are defined for only a limited range of the possible values which can be expressed in their field. If the value of the field is not in the defined range, the parameter has an illegal value.

Illegal values for may be detected either at load or at run time. If detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the default value for the parameter should be used if the parameter is required, or the entire Generator or Modulator ignored if it is optional. Certain parameters may have more specific procedures for illegal values as expressed elsewhere in this specification.

5.9.5.5 Out-of-range Values

SASBF parameters have a specified minimum and useful range the span the perceptually relevant values for the associated sonic property. When the parameter value is exceeds this useful range, the parameter is said to have an out of range value.

Out of range values can result from two distinct causes. An out of range value can be actually present as a SASBF generator value, or the out of range value can be the result of the summation of instrument and preset values.

Out of range values should be handled by substituting the nearest perceptually relevant or realisable value. SASBF banks should not be created with out of range values in the instrument generators. While it is acceptable practice to create SASBF banks which produce out of range values as a result of summation, it is undesirable and should be avoided where practical.

5.9.5.6 Missing Required Parameter or Terminator

Certain parameters and terminators are required by the SASBF specification. If these are missing, the bitstream element is technically not within specification. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the instrument or zone for which the required parameter is missing should simply be ignored. If this causes no sound, the corresponding key-on event is ignored.

5.9.5.7 Illegal enumerator

Certain enumerators are illegal in certain contexts. For example, key and velocity ranges must be the first generators in a zone, instruments are not allowed in instrument zones, and sampleIDs are not allowed in preset zones. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the enumerator should simply be ignored.

5.9.6 Profile 2 (Sample Bank and MIDI decoding)

This Subclause describes the decoding process in which a bitstream conforming to Profile 2 is converted into sound. Profile 2 supports the Structured Audio Sample Bank Format and the General MIDI Format (Profile 1) for sound description. Profile 2 supports the MIDI Protocol and the Standard MIDI File Format for score specification.

5.9.6.1 Stream information header

The SASBF bitstream element is part of the bitstream header. Score elements in the form of MIDI files may be included in the bitstream header.

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class `SA_streaminfo` provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- MIDI file: The events in the MIDI file shall be time ordered, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed. The sample bank requires no processing for this preparation. Processing may be used to improve the computational

efficiency of a decoder. Banks shall be assigned consecutive increasing bank ID numbers in the order of the banks' position in the bitstream. The first bank in the bitstream shall be numbered 0 unless a bank resident in the terminal is being used. In that case, the resident bank shall be numbered 0, an other banks shall be numbered proceeding from there.

5.9.6.2 Bitstream data and sound creation

The MIDI Note On message is defined in the MIDI specification. When the SASBF decoder receives a Note On message, a voice shall be instantiated. Synthesis parameters for the voice shall be determined by the data in the sample bank containing the preset corresponding to the MIDI channel of the Note On message. The number of wavetable oscillators that are used by the voice is defined by the sample bank. Each required oscillator shall have the state variables required for synthesis allocated to it. The state variables shall be initialised according to the preset data from the sample bank.

The MIDI Program Change message is defined in the MIDI specification. When the SASBF decoder receives a Program Change message, an assignment shall be made in the scheduler between the specified MIDI channel and the specified preset. Until this assignment is changed by a subsequent Program Change message, subsequent voice instantiations using the specified MIDI channel shall use sample bank data corresponding to the assigned preset. In a MIDI program change message, if no preset exists in the specified bank with the specified preset number, a replacement preset is used. The replacement preset is the preset with the specified preset number in the bank with the highest bank ID number less than the specified bank ID number which contains a preset with the specified preset number.

The run time scheduler is used to issue MIDI messages to the SASBF decoder. MIDI messages from a MIDI standard file shall be issued by the scheduler when the scheduler clock equals or exceeds the time stamp associated with the message.

5.9.6.3 Conformance

Floating point computation is not required in Profile 2. Audio samples are stored in the bitstream as 16-bit integers. The resolution of the interpolator filter is constrained by the interpolator specification given in Subclause 0.

5.9.7 Profile 4 (Sample Bank decoding in SAOL instruments)

5.9.8 Sample Bank Format Glossary

absolute - Describes a parameter which gives a definitive real-world value. Contrast to relative.

additive - Describes a parameter which is to be numerically added to another parameter.

articulation - The process of modulation of amplitude, pitch, and timbre to produce an expressive musical note.

attack - That phase of an envelope or sound during which the amplitude increases from zero to a peak value.

attenuation - A decrease in volume or amplitude of a signal.

bag - A Sample Bank Format data structure element containing a list of zones.

balance - A form of stereo volume control in which both left and right channels are at maximum when the control is centred, and which attenuates only the opposite channel when taken to either extreme.

bank - A collection of presets. See also MIDI bank.

bipolar - In the SASBF standard, said of a modulator source whose minimum is -1 and whose maximum is 1. Contrast “unipolar”

big endian - Refers to the organisation in memory of bytes within a word such that the most significant byte occurs at the lowest address. Contrast “little endian.”

byte - A data structure element of eight bits without definition of meaning to those bits.

BYTE - A data structure element of eight bits which contains an unsigned value from 0 to 255.

case-insensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as identical. Contrast “case-sensitive.”

case-sensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as distinct. Contrast “case-insensitive.”

cent - A unit of pitch ratio corresponding to the twelve hundredth root of two, or one hundredth of a semitone, approximately 1.000577790.

centibel - A unit of amplitude ratio corresponding to the two hundredth root of ten, or one tenth of a decibel, approximately 1.011579454.

CHAR - A data structure of eight bits which contains a signed value from -128 to +127.

chorus - An effects processing algorithm which involves cyclically shifting the pitch of a signal and remixing it with itself to produce a time varying comb filter, giving a perception of motion and fullness to the resulting sound.

chunk - The top-level division of a RIFF file.

convex - A curve which is bowed in such a way that it is steeper on its lower portion.

concave - (1) A curve which is bowed in such a way that it is steeper on its upper portion. (2) In the SASBF standard, said of a modulator source whose shape is that of the amplitude squared characteristic. Contrast with “convex” and “linear.”

cutoff frequency - The frequency of a filter function at which the attenuation reaches a specified value.

data points - The individual values comprising a sample. Sometimes also called sample points. Contrast “sample.”

decay - The portion of an envelope or sound during which the amplitude declines from a peak to steady state value.

decibel - A unit of amplitude ratio corresponding to the twentieth root of ten, approximately 1.122018454.

delay - The portion of an envelope or LFO function which elapses from a key-on event until the amplitude becomes non-zero.

destination - The generator to which a modulator is applied.

DC gain - The degree of amplification or attenuation a system presents to a static or zero frequency signal.

digital audio - Audio represented as a sequence of quantised values spaced evenly over time. The values are called “sample data points.”

doubleword - A data structure element of 32 bits without definition of meaning to those bits.

downloadable - Said of samples which are loaded from a file into RAM, in contrast to samples which are maintained in ROM.

dry - Refers to audio which has not received any effects processing such as reverb or chorus.

DWORD - A data structure of 32 bits which contains an unsigned value from zero to 4,294,967,295.

envelope - A time varying signal which typically controls the pitch, volume, and/or filter cutoff frequency of a note, and comprises multiple phases including attack, decay, sustain, and release.

enumerated - Said of a data element whose symbols correspond to particular assigned functions.

flat - A. Said of a tone that is lower in pitch than another reference tone. B. Said of a frequency response that does not deviate significantly from a single fixed gain over the audio range.

generator - In the SASBF standard, a parameter which directly affects sound reproduction. Contrast with “modulator.”

global - Refers to parameters which affect all associated structures. See “global zone.”

global zone - A zone whose generators and modulators affect all other zones within the object.

header - A data structure element which describes several aspects of a SASBF element.

instrument - In the SASBF standard, a collection of zones which represents the sound of a single musical instrument or sound effect set.

instrument zone - A sample and associated articulation data defined to play over certain key numbers and velocities.

interpolator - A circuit or algorithm which computes intermediate points between existing sample data points. This is of particular use in the pitch shifting operation of a wavetable synthesiser, in which these intermediate points represent the output samples of the waveform at the desired pitch transposition.

key number - See MIDI key number.

LFO - Acronym for Low Frequency Oscillator. A slow periodic modulation source.

linear - In the SASBF standard, said of a modulator source whose shape is that of a straight line. Contrast with “concave.”

linear coding - The most common method of encoding amplitudes in digital audio in which each step is of equal size.

little endian - A method of ordering bytes within larger words in memory in which the least significant byte is at the lowest address. Contrast “big endian.”

loop - In wavetable synthesis, a portion of a sample which is repeated many times to increase the duration of the resulting sound.

loop points - The sample data points at which a loop begins and ends.

lowpass - Said of a filter which attenuates high frequencies but does not attenuate low frequencies.

modulator - In the SASBF standard, a parameter which routes an external controller to dynamically alter the setting of a “generator.” Contrast with “generator.”

monotonic - Continuously increasing or decreasing. Said of a sequence which never reverses direction.

MIDI - Acronym for Musical Instrument Digital Interface. The standard protocol for sending performance information to a musical synthesiser.

MIDI bank - A group of up to 128 presets selected by a MIDI “change bank” command.

MIDI continuous controller - A construct in the MIDI protocol.

MIDI key number - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the key of the musical instrument keyboard to which the command refers.

MIDI pitch bend - A special MIDI construct akin to the MIDI continuous controllers which controls the real-time value of the pitch of all notes played in a MIDI channel.

MIDI preset - A “preset” selected to be active in a particular MIDI channel by a MIDI “change preset” command.

MIDI velocity - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the speed with which the key was pressed or released.

modulator - In the SASBF standard, a set of parameters which affect a particular generator. Contrast with “generator.”

mono - Short for “monophonic.” Indicates a sound comprising only one channel or waveform. Contrast with “stereo.”

negative - In the SASBF standard, said of a modulator which has a negative sloping characteristic. Contrast with “positive.”

octave - A factor of two in ratio, typically applied to pitch or frequency.

orphan - Said of a data structure which under normal circumstances is referenced by a higher level, but in this particular instance is no longer linked. Specifically, it is an instrument which is not referenced by any preset zone, or a sample which is not referenced by any instrument zone.

oscillator - In wavetable synthesis, the wavetable interpolator is considered an oscillator.

pan - Short for “panorama.” This is the control of the apparent azimuth of a sound source over 180 degrees from left to right. It is generally implemented by varying the volume at the left and right speakers.

pitch - The perceived value of frequency. Generally can be used interchangeably with frequency.

pitch shift - A change in pitch. Wavetable synthesis relies on interpolators to cause pitch shift in a sample to produce the notes of the scale.

pole - A mathematical term used in filter transform analysis. Traditionally in synthesis, a pole is equated with a rolloff of 6dB per octave, and the rolloff of a filter is specified in “poles.”

positive - In the SASBF standard, said of a modulator source which has a positive sloping characteristic. Contrast “negative.”

preset - A keyboard full of sound. Typically the collection of samples and articulation data associated with a particular MIDI preset number.

preset zone - A subset of a preset containing generators, modulators, and an instrument.

proximal - Closest to. Proximal sample data points are the data points closest in either direction to the named point.

Q - A mathematical term used in filter transform analysis. Indicates the degree of resonance of the filter. In synthesis terminology, it is synonymous with resonance.

RAM - Random Access Memory. Conventionally, this term implies read-write memory. Contrast “ROM.”

record - A single instance of a data structure.

relative - Describes a parameter which merely indicates an offset from an otherwise established value. Contrast to absolute.

release - The portion of an envelope or sound during which the amplitude declines from a steady state to zero value or inaudibility.

resonance - Describes the aspect of a filter in which particular frequencies are given significantly more gain than others. The resonance can be measured in dB above the DC gain.

resonant frequency - The frequency at which resonance reaches its maximum.

reverb - Short for reverberation. In synthesis, a synthetic signal processor which adds artificial spaciousness and ambience to a sound.

RIFF - Acronym for Resource Interchange File Format.

ROM - Acronym for Read Only Memory. A memory whose contents are fixed at manufacture, and hence cannot be written by the user. Contrast with RAM.

sample - This term is often used both to indicate a “sample data point” and to indicate a collection of such points comprising a digital audio waveform. The latter meaning is exclusively used in this Subclause (the SASBF specification.)

sample rate - The frequency, in Hertz, at which sample data points are taken when recording a sample.

semitone - A unit of pitch ratio corresponding to the twelfth root of two, or one twelfth of an octave, approximately 1.059463094.

sharp - Said of a tone that is higher in pitch than another reference tone.

SHORT - A data structure element of sixteen bits which contains a signed value from -32,768 to +32,767.

soft - The pedal on a piano, so named because it causes the damper to be lowered in such a way as to soften the timbre and loudness of the notes. In MIDI, continuous controller #66, which behaves in a similar manner.

sostenuto - The pedal on a piano which causes the dampers on all keys depressed to be held until the pedal is released. In MIDI, continuous controller #67, which behaves in a similar manner.

sustain - The pedal on a piano which prevents all dampers on keys as they are depressed from being released. In MIDI, continuous controller #64, which behaves in a similar manner.

source - In a SASBF modulator, the enumerator indicating the particular real-time value which the modulator will transform, scale, and add to the destination generator.

stereo - Literally indicating three dimensions. In this specification, the term is used to mean two channel stereophonic, indicating that the sound is composed of two independent audio channels, dubbed left and right. Contrast monophonic.

subchunk - A division of a RIFF file below that of the chunk.

synthesis engine - The hardware and software associated with the signal processing and modulation path for a particular synthesiser.

synthesiser - A device ideally capable of producing arbitrary musical sound.

terminator - A data structure element indicating the final element in a sequence.

timecent - A unit of duration ratio corresponding to the twelve hundredth root of two, or one twelve hundredth of an octave, approximately 1.000577790.

transform - In a SASBF modulator, the enumerator indicating the particular transfer function through which the source will be passed prior to scaling and addition to the destination generator.

tremolo - A periodic change in amplitude of a sound, typically produced by applying a low frequency oscillator to the final volume amplifier.

triangular - A waveform which ramps upward to a positive limit, then downward at the opposite slope to the symmetrically negative limit periodically.

unipolar - In the SASBF standard, said of a modulator source whose minimum is 0 and whose maximum is 1. Contrast “bipolar.”

velocity - In synthesis, the speed with which a keyboard key is depressed, typically proportionally to the impact delivered by the musician. See also MIDI velocity.

vibrato - A periodic change in the pitch of a sound, typically produced by applying a low frequency oscillator to the oscillator pitch.

volume - The loudness or amplitude of a sound, or the control of this parameter.

wavetable - A music synthesis technique wherein musical sounds are recorded or computed mathematically and stored in a memory, then played back at a variable rate to produce the desired pitch. Additional timbral adjustments are often made to the sound thus produced using amplifiers, filters, and effects processing such as reverb and chorus.

WORD - A data structure of 16 bits which contains an unsigned value from zero to 65,535.

word - A data structure element of 16 bits without definition of meaning to those bits.

zone - An object and associated articulation data defined to play over certain key numbers and velocities.

5.10MIDI semantics, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed.

- Sample data: The data in the sample shall be stored, and whatever preprocessing necessary to prepare the data for reference from a SAOL wavetable generator shall be performed. If the sample data is represented as 16-bit integers in the bitstream, it shall be converted to floating-point format at this time.

If there is more than one orchestra file in the stream information header, the various files are combined together via concatenation and processed as one large orchestra file. That is, each orchestra file within the bitstream refers to the same global namespace, instrument namespace, and opcode namespace.

5.3.3 Bitstream data and sound creation

5.3.3.1 Relationship with systems layer

At each time step within the systems operation, the systems layer may present the Structured Audio decoder with an Access Unit containing data conforming to the **SA_access_unit** class. The run-time responsibility of the Structured Audio decoder is to receive these AU data elements, parse and understand them as the various Structured Audio bitstream data elements, execute the on-going SAOL orchestra, via the scheduler, to produce one Composition Unit of output, and present the systems layer with that Composition Unit.

5.3.3.2 Bitstream data elements

As Access Units are received from the systems demultiplexer, they are parsed and used by the Structured Audio decoder in various ways, as follows:

- Score line events shall be registered with the scheduler.

- MIDI events shall be converted into appropriate SAOL events (see Subclause 0) and then registered with the scheduler, if they have time stamps, or executed in the next k-cycle, if not.

- Sample data shall be stored, and whatever preprocessing is necessary for reference by forthcoming score lines containing references to that sample shall be performed. . If the sample data is represented as 16-bit integers in the bitstream, it shall be converted to floating-point format at this time.

5.3.3.3 Scheduler semantics

5.3.3.3.1 Purpose of scheduler

The scheduler is the central control mechanism of a Structured Audio decoding system. It is responsible for handling events by instantiating and terminating instruments, keeping track of what instrument instantiations are active, instructing the various instrument instantiations to perform synthesis, routing the output of instruments onto busses, and sending busses to effects instruments. Although there are many ways to perform these tasks, the exact nature of what must be done can be clearly specified. This Subclause provides normative bounds on the activities of the scheduler.

5.3.3.3.2 Instrument instantiation

To instantiate an instrument is to create data space for its variables and the data space required for any opcodes called by that instrument. When an instrument is instantiated, the following tasks shall be performed. First, space for any parameter fields shall be allocated and their values set according to the p-fields of the instantiating expression or event. Then, space for any locally declared variables shall be allocated and these variable values set to 0. Then, the current values of any imported i-rate variables shall be copied into the local storage space. Then, locally declared wavetables shall be created and filled with data according to their declaration and the appropriate rules in Subclause 5.6 SAOL core wavetable generator.

5.3.3.3.3 Instrument termination

To terminate an instrument instantiation is to destroy the data space for that instance.

5.3.3.3.4 Instrument execution

To execute an instrument instantiation at a particular rate is to calculate the results of the instructions given in that instrument definition. When an instrument instance is executed at a particular rate, the following steps shall be performed. First, the values of any global variables and wavetables imported by that instrument at that rate shall be copied into the storage space of the instrument. In addition, when executing at the a-rate an instrument instance which is the target of a **send** statement, the current value of the **input** standard name in the instance shall be set to the current value of the bus or busses referenced in the **send** statement. Then, the code block for that instrument shall be executed at the particular rate with regard to the data space of the instrument instantiation, as given by the rules in Subclause 5.4.6.6 Block of code statements. Then, the values of any global variables and wavetables exported by that instrument at that rate shall be copied into the global storage space. Finally, when executing an instrument instantiation at the a-rate, the value of the instance output shall be added to the bus onto which the instrument is routed according to the rules in Subclause 5.4.5.4 Route statement, unless the instance is the target of a **send** expression referencing the special bus **output_bus**, in which case the output of the instrument instance is the output of the orchestra and may be turned into sound

5.3.3.3.5 Orchestra startup and configuration

At orchestra startup time, before the first Composition Unit of audio samples is created in the scheduler, the following tasks shall be performed. First, space for any global signal variables (see Subclause 5.4.5.3 Global variable declaration) shall be allocated and their values set to zero. If there is an instrument called **startup** in the orchestra, that instrument shall be instantiated and executed at the i-rate. After this execution is complete, then all global wavetables are created and filled with data according to their definitions in the global block of the orchestra and the appropriate rules in Subclause 5.6 SAOL core wavetable generator.

After the global wavetable creation, the orchestra busses are initialised. Each bus's width is determined, in the order specified by the global sequencing rules (Subclause 5.4.5.6 Sequence specification), as the width of the output expression by instruments on that bus. For the purposes of calculating bus widths, any

instrument which does not receive any bus data according to the sequence rules shall have an **inchannels** width of 0 (this specification is needed since output widths may depend on the value of **inchannels**).

After busses are created, all instruments which are the targets of **send** statements as described in Subclause 5.4.5.5 Send statement shall be instantiated and executed at the i-rate in the order specified by the global sequencing rules described in the global block according to Subclause 5.4.5.6 Sequence specification. Finally, the global absolute orchestra time shall be set to 0.

NOTE

A time is called *absolute* if it is specified in seconds. When a tempo instruction is first decoded and the value of tempo changes from its default value, the score time and the absolute time are not identical anymore; all the times in the score, subsequent to a tempo line execution, are scaled according to the new tempo and enqueued in absolute dispatch and duration times as specified in Subclause 0, list item 3.

5.3.3.3.6 Decoder execution while streaming

In each orchestra cycle, one Composition Unit of samples is produced by the real-time synthesis process. This synthesis is performed according to the rules below and the resulting orchestra output, as described in list item 11, is presented to the Systems layer as a Composition Unit. To execute one orchestra cycle, the following tasks shall be performed in the order denoted:

1. If there is an end event whose dispatch time is earlier than the current absolute orchestra time, no further output is produced, and all future requests from the systems layer produce Composition Units are responded to with buffers of all 0s.

2. If there are any instrument events whose dispatch time is earlier than the current absolute orchestra time, an instrument instantiation is created for each such instrument event (see Subclause 0), and those instantiations are each executed at the i-rate (see Subclause 0) in the order prescribed by the global sequencing rules. If the instrument event specifies a duration for that instrument instantiation, the instrument instantiation shall be scheduled for termination at the time given by the sum of the current absolute orchestra time and the specified duration (scaled to absolute time units according to the actual tempo, if any).

   NOTE

   If the current orchestra time differs from the instrument dispatch time, the former shall be used to schedule instance termination.

3. If there are any active instrument instantiations whose termination time is earlier than the current absolute orchestra time, then the **released** standard name shall be set to 1 within each such instrument instance, and the instance is marked for termination in step 8, below.

4. If there are any control events whose dispatch time is earlier than the current absolute orchestra time, the global variables or instrument variables within instrument instantiations labelled by that control event shall have their values updated accordingly (see Subclause 5.7.4 Control line). Note that this implies that no more than one control change per variable per control cycle may be received by the orchestra. If multiple control changes are received in a single control cycle, the resulting value of the instrument or global variable is unspecified.

If there are any table events whose dispatch time is earlier than the current absolute orchestra time, global wavetables shall be created or

destroyed as specified by the table event (see Subclause 5.7.5 Tempo line

The **tempo** line in the score specifies the new tempo for the decoding process. The tempo is specified in beats-per-minute; the default tempo shall be sixty beats per minute, and thus by default the score time is measured in seconds.

The first number in the tempo line is the score time at which the tempo changes. When this time arrives, the tempo event shall be dispatched as described in Subclause 5.3.3.3 Scheduler semantics, list item 7.

The second number is the new tempo, specified in beats per minute. Consequently, one beat lasts 60/**tempo** seconds, so that a tempo of 120 beats per minute is twice as fast as the default. When a tempo line is decoded, the time numbers in the score continue progressively, with the increments now in accordance with the new time unit.

1. 5.7.6 Table line).

2. If there are any MIDI events whose timestamp is earlier than the current orchestra time, or which have been received without timestamps since the last execution of this rule, they are dispatched according to their semantics in Subclause 5.10.3 Mapping MIDI events into orchestra control.

3. If there are any tempo events whose dispatch time is earlier than the current absolute orchestra time, then the global **tempo** standard variable shall be set to the specified value, and all the score times after the current absolute time shall be scaled according to the new tempo value. The already scheduled times for terminations are also scaled in their remaining part, according to the ratio between the old and new tempo. Existing **extend** times are not affected, since they are specified in absolute time and are thus “outside” the score.

   NOTE

   If the current orchestra time differs from the tempo dispatch time, the former shall be used to calculate the new durations and future dispatch times of events.

4. If the **speed** field of the **AudioSource** scene node responsible for instantiating this decoder (see Subclause 5.11 Input sounds and relationship with AudioBIFS) has been changed in the last k-cycle, the **tempo** standard variable shall be set to the value specified in Subclause XXX of FCD ISO 14496-1, and events in the orchestra shall be rescaled as specified in (7) above.

5. The value of each channel of each bus shall be set to 0.

6. Each active instrument instance shall be executed once at the k-rate and *n* times at the a-rate, where *n* is the number of samples in the control period (see Subclause 0). Each execution at the k-rate shall be in the order given by the global sequencing rules, and each corresponding execution at the a-rate (that is, the first a-rate execution in a k-cycle of each, the second a-rate execution in a k-cycle of each, etc.) shall be in the order given by the global sequencing rules.

   NOTE 1

   If instrument **a** is sequenced before instrument **b** according to the rules in Subclause 5.4.5.6 Sequence specification, then the k-rate execution of **a** shall be strictly before the k-rate execution of **b**, and the k-rate execution of **a** shall be strictly before the first a-rate execution of **a**, and the first a-rate execution of **a** shall be strictly before the first a-rate execution of **b**.

However, there is no normative sequencing between the second a-rate execution of **a** and the first a-rate execution of **b**, or between the first a-rate execution of **a** and the k-rate execution of **b**, within a particular orchestra cycle.

NOTE 2

In accordance with to the conformance rules in Subclause 5.3.4 Conformance, the execution ordering described in this Subclause may be rearranged or ignored when it can be determined from examination of the orchestra that to do so will have no effect on the output of the decoding process. "Has no effect" shall be taken to mean that the output of the decoding process in rearranged order is sample-by-sample identical to the output of the decoding process performed strictly according to the rules in this Subclause.

7. If the special bus **output_bus** is sent to an instrument, the output of that instrument at each a-cycle is the orchestra output at that a-cycle. Otherwise, the value of the special bus **output_bus** after each instrument has been executed for an a-cycle is the orchestra output at that a-cycle. If the value of the current orchestra output is greater than 1 or less than –1, it shall be set to 1 or -1 respectively (hard clipping).

8. For each instance which was marked for termination in step 3, above: if that instrument instance called **extend** with a parameter greater than the amount of time in a control-cycle, the instrument is not terminated. All other instrument instances marked for termination in step 3 are terminated (see Subclause 0). As discussed in Subclause 5.10.3.2.9 All notes off, in the case of an "All Notes Off" MIDI message, instances may not extend themselves, and are destroyed at this time.

9. The current global absolute orchestra time is advanced by one control period.

5.3.4 Conformance

With regard to all normative language in this Sub-Part of ISO 14496-3, conformance to the normative language is measured at the time of orchestra output. Any optimisation of SAOL code or rearrangement of processing sequence may be performed as long as to do so has no effect on the output of the orchestra. "Has no effect" in this sense means that the output of the rearranged or optimised orchestra is sample-by-sample identical to the output of the original orchestra according to the decoding rules given in this Subpart.

5.4 SAOL syntax and semantics

5.4.1 Relationship with bitstream syntax

The bitstream syntax description as given in Subclause 5.1 Bitstream syntax and semantics specifies the representation of SAOL instruments and algorithms that shall be presented to the decoder in the bitstream. However, the tokenised description as presented there is not adequate to describe the SAOL language syntax and semantics. In addition, for purposes of enabling bitstream creation and exchange in robust manner, it is useful to have a standard human-readable textual representation of SAOL code in addition to the tokenised binary format.

The Backus-Naur Format (BNF) grammar presented in this Subclause denotes a language, or an infinite set of programs; the legal programs which may be transmitted in the bitstream are restricted to this set. Any program which cannot be parsed by this grammar is not a legal SAOL program – it has a *syntax error* – and a bitstream containing it is an invalid bitstream. Although the bitstream is made up of tokens, the grammar will be described in terms of lexical elements – a *textual representation* – for clarity of presentation. The syntactic rules expressed by the grammar which restrict the set of textual programs also normatively restrict the syntax of the bitstream, through the relationship of the bitstream and the textual format in the normative tokenisation process.

This Subclause thus describes a textual representation of SAOL which is standardised, but stands outside of the bitstream-decoder relationship. Subclause 0 describes the mapping between this textual representation and the bitstream representation. The exact normative *semantics* of SAOL will be described in reference to the textual representation, but also apply to the tokenised bitstream representation as created via the normative tokenisation mapping.

Annex C contains a grammar for the SAOL textual language, represented in the ‘lex’ and ‘yacc’ formats. Using these versions of the grammar, parsers can be automatically created using the ‘lex’ and ‘yacc’ tools. However, these versions are for informative purposes only; there is no requirement to use these tools in building a decoder.

Normative language regarding syntax in this Subclause provides bounds on syntactically legal SAOL programs, and by extension, the syntactically legal bitstream sequences which can appear in an **orchestra** bitstream class. That is, there are constructions which appear to be permissible upon reading only the BNF grammar, but are disallowed in the normative text accompanying the grammar. The status of such constructions is exactly that of those which are outside of the language defined by the grammar alone. In addition, normative language describing static rate semantics further bounds the set of syntactically legal SAOL programs, and by extension, the set of syntactically legal bitstream sequences.

The decoding process for bitstreams containing syntactically illegal SAOL programs (i.e., SAOL programs which do not conform to the BNF grammar, or contain syntax errors or rate mismatch errors) is unspecified.

Normative language regarding semantics in this Subclause describes the semantic bounds on the behaviour of the Structured Audio decoder. Certain constructions describe “run-time error” situations; the behaviour of the decoder in such circumstances is not normative, but implementations are encouraged to recover gracefully from such situations and continue decoding if possible.

5.4.2 Lexical elements

5.4.2.1 Concepts

The textual SAOL orchestra contains punctuation marks, which syntactically disambiguate the orchestra; identifiers, which denote symbols of the orchestra; numbers, which denote constant values; string constants, which are not currently used; comments, which allow internal documentation of the orchestra; and whitespace, which lexically separates the various textual elements. These elements do not occur in the bitstream – since each is represented there by a token – but we define them here to ground the subsequent discussion of SAOL. Within the rest of Subclause 5.4 SAOL syntax and semantics, when we discuss the semantics of "an identifier", this shall be taken to normatively refer to the semantics of the symbol denoted by that identifier; the language used is for clarity of presentation.

A lexical grammar for parsing SAOL, written in the 'lex' language, is provided for informative purposes in Annex 5.C.

5.4.2.2 Identifiers

An identifier is a series of one or more letters, digits and the underscore that begins with a letter or underscore; it denotes a symbol of the orchestra. Every identifier which consists of the same characters in the first 16 characters (is equivalent under string comparison to the first 16 characters) denotes the same symbol. Identifiers are case-sensitive, meaning that identifiers which differ only in the case of one or more characters denote different symbols.

A string of characters equivalent to one of the reserved words listed in Subclause 5.4.9 Reserved words, to one of the standard names listed in Subclause 5.4.6.8 Standard names, to the name of one of the core opcodes listed in Subclause 5.5.3 List of core opcodes, or to the name of one of the core wavetable generators listed in Subclause 5.6 SAOL core wavetable generator does not denote a symbol, but rather denotes that reserved word, standard name, core opcode, or core wavetable generator.

An identifier is denoted in the BNF grammar below by the terminal symbol **<ident>**.

5.4.2.3 Numbers

There are two kinds of symbolic constants which hold numeric values in SAOL: integer constants and floating-point constants.

The integer constant must occur in certain contexts, such as array definitions. An integer token is a series of one or more digits. Since the contexts in which integers must occur in SAOL do not allow negative values, there is no provision for negative integers. A string of characters which appears to be a negative integer shall be lexically analysed as a floating-point constant. No integer constant greater than 2^{32} (4294967296) shall occur in the orchestra.

An integer constant is denoted in the BNF grammar below by the terminal symbol **<int>**.

The floating-point constant occurs in SAOL expressions, and denotes a constant numeric value. A floating-point token consists of a base, optionally followed by an exponent. A base is either a series of one or more digits, optionally followed by a decimal point and a series of zero or more digits, or a decimal point followed by a series of one or more digits. An exponent is the letter **e**, optionally followed by either a **+** or **–** character, followed by a series of one or more digits. Since the floating-point constant appears in a SAOL expression, where the unary negation operator is always available, floating-point constants need not be

lexically negative. Every floating-point constant in the orchestra shall be representable by a 32-bit floating-point number.

A floating-point constant is denoted in the BNF grammar below by the terminal symbol **<number>**.

5.4.2.4 String constants

String constants are not used in the normative SAOL specification, but a description is provided here so that they may be treated consistently by implementors who choose to add functionality over and above normative requirements to their implementations.

A string constant denotes a constant string value, that is, a character sequence. A string constant is a series of characters enclosed in double quotation marks (**"**). The double quotation character may be included in the string constant by preceding it with a backslash (\) character. Any other character, including the line-break (newline) character, may be explicitly enclosed in the quotation marks.

The interpretation and use of string constants is left open to implementors.

5.4.2.5 Comments

Comments may be used in the textual SAOL representation to internally document an orchestra. However, they are not included in the bitstream, and so are lost on a tokenisation/detokenisation sequence.

A comment is any series of characters beginning with two slashes (**//**), and terminating with a new line. During lexical analysis, whenever the **//** element is found on a line, the rest of the line is ignored.

5.4.2.6 Whitespace

Whitespace serves to lexically separate the various elements of a textual SAOL orchestra. It has no syntactic function in SAOL, and is not represented in the bitstream, so the exact whitespacing of a textual orchestra is lost on a tokenisation/detokenisation sequence.

A whitespace is any series of one or more space, tab, and/or newline characters.

5.4.3 Variables and values

Each variable within the SAOL orchestra holds a value, or an ordered set of values for array variables, as an intermediate calculation by the orchestra. At any point in time, the value of a variable, sample in a wavetable, or single element of an array variable, shall be represented by a 32-bit floating-point value.

Conformance to this Subclause is in accordance with Subclause 5.3.4 Conformance; that is, implementations are free to use any internal representation for variable values, so long as the results calculated are identical to the results of the calculations using 32-bit floating-point values.

NOTE

For certain sensitive digital-filtering operations, the results of using greater precision in a calculation may be equivalently detrimental to orchestra output as the results of using less precision, as the stability of the filter may be critically dependent on the quantization error which is provided with 32-bit values. It is strongly deprecated for bitstreams to contain code which generates widely different results when calculated with 32-bit and 64-bit arithmetic.

At orchestra output, the values calculated by the orchestra should reside between a minimum value of –1 and a maximum value of 1. These values at orchestra output represent the maximum negatively- and positively-valued audio samples which can be produced by the terminal. If the values calculated by the orchestra fall outside that range, they are clipped to [-1,1] as described in Subclause 5.3.3.3 Scheduler semantics list item 7. When the terminal presents the sound to a listener, it is likely that further rescaling of the signal will be necessary, as required by the particular digital-analog converter present in the terminal. This scaling is not done by the orchestra, but is outside the scope of the standard and happens after all processing described in Subclause 5.3.3.3 Scheduler semantics is completed.

5.4.4 Orchestra

<orchestra> -> <orchestra element> <orchestra>
<orchestra> -> <orchestra element>

The orchestra is the collection of signal processing routines and declarations that make up a Structured Audio processing description. It shall consist of a list of one or more orchestra elements.

<orchestra element> -> <global block>
<orchestra element> -> <instrument declaration>
<orchestra element> -> <opcode declaration>
<orchestra element> -> <template declaration>
<orchestra element> -> **NULL**

There are four kinds of orchestra elements:

1. The global block contains instructions for global orchestra parameters, bus routings, global variable declarations, and instrument sequencing. It is not permissible to have more than one global block in an orchestra.

2. Instrument declarations describe sequences of processing instructions which can be parametrically controlled using SASL or MIDI score files.

3. Opcode declarations describe sequences of processing instruments which provide encapsulated functionality used by zero or more instruments in the orchestra.

4. Template declarations describe multiple instruments which differ only slightly using a concise parametric form.

Orchestra elements may appear in any order within the orchestra; in particular, opcode definitions may occur either syntactically before or after they are used in instruments or other opcodes.

5.4.5 Global block

5.4.5.1 Syntactic form

<global block> -> **global {** <global list> **}**
<global list> -> <global statement> <global list>
<global list> -> NULL

A global block shall contain a global list, which shall consist of a sequence of zero or more global statements.

<global statement> -> <global parameter>
<global statement> -> <global variable declaration>
<global statement> -> <route statement>
<global statement> -> <send statement>
<global statement> -> <sequence definition>

There are five kinds of global statement:

1. Global parameters set orchestra parameters such as sampling rate, control rate, and number of input and output channels of sound

2. Global variable declarations define global variables which can be shared by multiple instruments.

3. Route statements describe the routing of instrument outputs onto busses.

4. Send statements describe the sending of busses to effects instruments.

5. Sequence definitions describe the sequencing of instruments by the run-time scheduler.

5.4.5.2 Global parameter

5.4.5.2.1 srate parameter

<global parameter> -> **srate <int>;**

The **srate** global parameter specifies the audio sampling rate of the orchestra. The decoding process shall create audio internally at this sampling rate. It is not permissible to simplify orchestra complexity or account for terminal capability by generating audio internally at other sampling rates, for to do so may have seriously detrimental effects on certain processing elements of the orchestra.

The **srate** parameter shall be an integer value between 4000 and 96000 inclusive, specifying the audio sampling rate in Hz. If the **srate** parameter is not provided in an orchestra, the default shall be the fastest of the audio signals provided as input (see Subclause 5.11 Input sounds and relationship with AudioBIFS). If the sampling rate is not provided, and there are no input audio signals, the default sampling rate shall be 32000 Hz.

5.4.5.2.2 krate parameter

<global parameter> -> **krate <int>;**

The **krate** global parameter specifies the control rate of the orchestra. The decoding process shall execute k-rate processing internally at this rate. It is not permissible to simplify orchestra complexity or account for terminal capability by executing k-rate processing at other rates, unless it can be determined that to do so will have no effect on orchestra output. In this case, “no effect” means that the resulting output of the orchestra is sample-by-sample identical to the output created if the control rate is not altered.

The **krate** parameter shall be an integer value between 1 and the sampling rate inclusive, specifying the control rate in Hz. If the **krate** parameter is not provided in an orchestra, the default control rate shall be 100 Hz.

If the control rate as determined by the previous paragraph is not an even divisor of the sampling rate, then the control rate is the next larger integer which does evenly divide the sampling rate. The *control period* of the orchestra is the number of samples, or amount of time represented by these samples, in one control cycle.

5.4.5.2.3 inchannels parameter

<global parameter> -> **inchannels <int>;**

The **inchannels** global parameter specifies the number of input channels to process. If there are fewer than this many audio channels provided as input sources, the additional channels shall be set to continuous zero-valued signals. If there are more than this many audio channels provided as input sources, the extra channels are ignored.

If the **inchannels** parameter is not provided in an orchestra, the default shall be the sum of the numbers of channels provided by the input sources (see Subclause 5.11 Input sounds and relationship with AudioBIFS). If there are no input sources provided, the value shall be 0.

5.4.5.2.4 outchannels parameter

<global parameter> -> **outchannels <int>;**

The **outchannels** global parameter specifies the number of output channels of sound to produce. The run-time decoding process shall produce and render this number of channels internally. It is not permissible to simplify orchestra complexity or account for terminal capability by producing fewer channels.

If the **outchannels** parameter is not provided in an orchestra, the default shall be one channel.

5.4.5.3 Global variable declaration

5.4.5.3.1 Syntactic form

<global variable declaration> -> **ivar** <namelist> **;**
<global variable declaration> -> **ksig** <namelist> **;**
<global variable declaration> -> <table declaration> **;**

Global variable declarations declare variables which may be shared and accessed by all instruments and by a SASL score. Only **ivar** and **ksig** type variables, as well as wavetables, may be declared globally. A global variable declaration is either a table definition, or an allowed type name followed by a list of name declarations.

A global name declaration specifies that a name token shall be created and space equal to one signal value allocated for variable storage in the global context. A global array declaration specifies that a name token shall be created and space equal to the specified number of signal values allocated in the global context.

5.4.5.3.2 Signal variables

<namelist> -> <name>**,** <namelist>
<namelist> -> <name>

A namelist is a sequence of one or more name declarations.

<name> -> **<ident>**
<name> -> **<ident>[**<array length>**]**

<array length> -> **<int>**
<array length> -> **inchannels**
<array length> -> **outchannels**

A name declaration is an identifier (see Subclause 5.4.2.2 Identifiers), or an array declaration. For an array declaration, the parameter shall be either an integer strictly greater than 0, or one of the tokens **inchannels** or **outchannels**. If the latter, the array length shall be the same as the number of input channels or output channels to the instrument, respectively, as described in Subclause 5.4.5.2 Global parameter. It is illegal to use the token **inchannels** if the number of input channels to the instrument is 0.

Not every identifier may be used as a variable name; in particular, the reserved words listed in Subclause 5.4.8 Template declaration

5.4.8.1 Syntactic form

<template declaration> -> **template <** <identlist> **> (** <identlist> **)**
map { <identlist> **} with {** <maplist> **}**
{ <instr variable declarations> <block> **}**<maplist> -> **<** <expr list> **> ,** <maplist>
<maplist> -> **<** <expr list> **>**

<identlist> as given in Subclause 5.4.5.4 Route statement.
<namelist> as given in Subclause 5.4.5.3.2 Signal variables.
<instr variable declarations> as given in Subclause 5.4.6.5.1 Syntactic form.
<expr list> as given in Subclause 5.4.6.6.1 Syntactic form.
<block> as given in Subclause 5.4.6.6.1 Syntactic form.

A template declaration allows the concise declaration of multiple instruments which are similar in processing structure and syntax, but differ in only a few key expressions or wavetable names.

5.4.8.2 Semantics

The first identifier list contains the names for the instruments declared with the template. There shall be at least one identifier in this list. The second identifier list contains the pfields for the template declaration. Each instrument declared with the template has the same list of pfields. The third identifier list contains a list of template variables which are to be replaced in the subsequent code block with expressions from the map list. There may be no identifiers in this list, in which case each instrument declared by the template is exactly the same.

The map list takes the form of a list of lists. This list shall have as many elements as template variables declared in the third identifier list. Each sublist is a list of expressions, and shall have as many elements as instrument names in the first identifier list.

5.4.8.3 Template instrument definitions

As many instruments are defined by the template definition as there are names in the first identifier list. To describe each of the instruments, the identifiers described in the third (template variable) list are replaced in turn by each of the expressions from the map list.

That is, to construct the code for the first instrument, the code block given is processed by replacing the first template variable with the first expression from the first map list sublist, the second template variable with the first expression from the second map list sublist, the third template variable with the first expression from the third map list sublist, and so on. To construct the code for the second instrument, the code block given is processed by replacing the first template variable with the *second* expression from the first map list sublist, the second template variable with the *second* expression from the second map list sublist, the third template variable with the second expression from the third map list sublist, and so on.

This code-block processing occurs before any other syntax-checking or rate-checking of the elements of the instruments so defined. That is, the template variables are not true signal variables, and do not need to be declared in the variable declaration block. Once the code-block processing and template expansion is complete, the resulting instruments are treated as any other instruments in the orchestra.

EXAMPLE

The following template declaration:

```
template <oneharm, threeharm>(p)
  map {pitch,t,bar} with { <440, harm1, mysig>, <p, harm2, mysig * mysig + 2> }
{

  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(t,pitch,-1);

  mysig = bar *3;
  output(mysig);
}
```

declares exactly the same two instruments as the following two instrument declarations:

```
instr oneharm(p) {
  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(harm1,440,-1);

  mysig = mysig * 3;
  output(mysig);
  }

instr threeharm(p) {
  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(harm3,p,-1);

  mysig = (mysig * mysig + 2) * 3;  // notice embedding of template expression
  output(mysig);
  }
```

5.4.9 Reserved words, the standard names listed in Subclause 5.4.6.8 Standard names, the names of the core opcodes listed in Subclause 5.5 SAOL core opcode definitions and semantics, and the names of the core wavetable generators listed in Subclause 5.6 SAOL core wavetable generators shall not be declared as variable names.

5.4.5.3.3 Wavetable declarations

<table declaration> -> **table <ident> (<ident> ,** <expr> [**,** <expr list>] **) ;**
<expr> as defined in Subclause 5.4.6.7 Expression.
<expr list> as defined in Subclause 5.4.6.6.1 Syntactic form.

Wavetables are structures of memory allocated for the typical purpose of allowing rapid oscillation, looping, and playback. The wavetable declaration associates a name (the first identifier) with a wavetable created by a core wavetable generator referenced by the second identifier. It is a syntax error if the second identifier is not one of the core wavetable generators named in Subclause 0. The first expression in the comma-delimited parameter sequence is termed the *size expression*; the remaining zero or more expressions comprise the *wavetable parameter list*.

The semantics of the size expression and wavetable parameter list are determined by the particular core wavetable generator, see Subclause 0. Any expression which is i-rate (see Subclause 5.4.6.7.2 Properties of expressions) is legal as part of the table parameter list; in particular, reference to i-rate global variables is allowed (their values may be set by the special instrument **startup**). Each expression must be single-valued, except in the case of the **concat** generator (Subclause 5.6.16 Concat), in which case the expressions must be table references. The order of creation of wavetables is non-deterministic; it is not recommended for calls to the **tableread()** opcode to occur in the table parameter expressions, and to do so gives unspecified results.

A global wavetable may be referenced by a wavetable placeholder in any instrument or opcode. See Subclause 5.4.6.5.4 Wavetable placeholder. Global wavetables shall be created and initialised with data at orchestra initialisation time, immediately after the execution of the special instrument **startup.** They shall not be destroyed unless they are explicitly destroyed or replaced by a **table** line in a SASL score.

To create a wavetable, first, the expression fields are evaluated in the order they appear in the syntax according to the rules in Subclause 5.4.6.7 Expressions. Then, the particular wavetable generator named in the second identifier is executed; the normative semantics of each wavetable generator detail exactly how large a wavetable shall be created, and which values placed in the wavetable, for each generator.

5.4.5.4 Route statement

<route statement> -> **route (<ident> ,** <identlist> **) ;**

<identlist> -> **<ident> ,** <identlist>
<identlist> -> **<ident>**
<identlist> -> **<NULL>**

A **route** statement consists of a single identifier, which specifies a bus, and a sequence of one or more instrument names, which specify instruments. The route statement specifies that the instruments listed do not produce sound output directly, but instead their results are placed on the given bus. The output channels from the instruments listed each are placed on a separate channel of the bus. Multiple **route** statements onto the same bus indicate that the given instrument outputs should be summed on the bus. Multiple **route** statements with differing numbers of channels referencing the same bus are illegal, unless each statement

has either *n* channels or 1 channel. In this case, each of the one-channel **route** statements places the same signal on each channel of the bus, which is *n* channels wide.

There shall be at least one instrument name in the instrument list (the NULL Subclause in the grammar is provided so that constructions appearing later may use the same production).

EXAMPLES

Assume that instruments **a, b,** and **c** produce one, two, and three channels of output, respectively.

1. The sequence

```
route(bus1, a, b);
route(bus1, c);
```

is legal and specifies a three-channel bus. The first bus channel contains the sum of the output of **a** and the first channel of **c**; the second contains the sum of the first output channel of **b** and the second of **c**; and the third contains the sum of the second channel of **b** and the third channel of **c**.

2. The sequence

```
route(bus1,b);
route(bus1,c);
```

is illegal since the statements refer different numbers of channels to the same bus.

3. The sequence

```
route(bus1,a,c);
route(bus1,a);
route(bus1,b,b);
```

is legal and specifies a four-channel bus. The first and third **route** statements each refer to four channels of audio, and the second refers to one channel, which will be mapped to each of the four channels.

The resulting channel values are as follows, using array notation to indicate the channel outputs from each instrument:

Channel	Value
1	`a + a + b[1]`
2	`c[1] + a + b[2]`
3	`c[2] + a + b[1]`
4	`c[3] + a + b[2]`

It is illegal for a **route** statement to reference a bus which is not the special bus **output_bus** and which does not occur in a **send** statement. See Subclause 5.4.5.5 Send statement.

It is illegal for a **route** statement to refer to the special bus **input_bus** (see Subclause 5.11.2 Input sources and phaseGroup)**.**

All instruments which are not referred to in **route** statements place their output on the special bus **output_bus,** except for an effect instrument to which **output_bus** was sent (see Subclause 5.4.5.5 Send statement). The same rules for allowable channel combinations to the special bus **output_bus** apply as if the route statements were explicit; these rules are implicit in the rules for the **output** statement, see Subclause 5.4.6.6.8 Output.

5.4.5.5 Send statement

<send statement> -> **send (<ident> ;** <expr list> **;** <identlist> **);**
<identlist> as defined in Subclause 5.4.5.4 Route statement
<expr list> as defined in Subclause 5.4.6.6.1 Syntactic form

The **send** statement creates an instrument instantiation, defines busses, and specifies that the referenced instrument is used as an effects processor for those busses.

All busses in the orchestra are defined by using **send** statements. It is illegal for a statement referencing a bus to refer to a bus which is not defined in a **send** statement. The exception is the special bus **output_bus** which is always defined.

The identifier in the **send** statement references an instrument which will be used as a bus-processing instrument, also called *effect instrument*. There is no syntactic distinction between effect instruments and other instruments. The identifier list references one or more busses which shall be made available to the effect instrument through its **input** standard name, as follows:

> The first n_0 channels of **input,** channels 0 through n_0-1 are the n_0 channels of the first referenced bus;
> Channels n_0 through n_0+n_1-1 of **input** are the n_1 channels of the second bus,
> and so forth, with a total of $n_0 + n_1 + \ldots + n_k$ channels.

In addition, the grouping of busses in the **input** array shall be made available to the effect instrument through its **inGroup** standard name, as follows:

> The first n_0 values of **inGroup** have the value 1;
> Channels n_0 through n_0+n_1-1 of **inGroup** have the value 2,
> and so forth, through $n_0 + n_1 + \ldots + n_k$, with the last n_k having the value k.

The expression list is a list of zero or more i-rate expressions which are provided to the effect instrument as its parameter fields. Any expression which is i-rate (see Subclause 5.4.6.7.2 Properties of expressions) is legal as part of this list; in particular, reference to i-rate global variables is allowed. The number of expressions provided shall match the number of parameter fields defined in the instrument declaration; otherwise, it is a syntax error.

The effect instrument referred to in a **send** statement shall be instantiated no later than immediately after the first instantiation of an instrument which either is routed to a bus which is sent to the effect instrument or refers to the bus in an **outbus** or **sbsynth** statement. These instrument instantiations shall remain in effect until the orchestra synthesis process terminates. One instrument instantiation shall be created for each **send** statement in the orchestra. If such an instrument instantiation utilises the **turnoff** statement, the instantiation is destroyed (and sound is no longer routed to it). No other changes are made in the orchestra.

Any bus may be routed to more than one effect instrument, except for the special bus **output_bus**. The special bus **output_bus** represents the second-to-finalmost processing of a sound stream; it may only be sent to at most one effect instrument, and it is a syntax error if that instrument is itself routed or makes use of the **outbus** statement. If **output_bus** is not sent to an instrument, it is turned into sound at the end of an orchestra cycle (see Subclause 5.3.3.3 Scheduler semantics); if **output_bus** is sent to an instrument, the output of that instrument is turned into sound at the end of an orchestra pass. This instrument is not permitted to use the **turnoff** statement.

At least one bus name shall be provided in the **send** instruction.

5.4.5.6 Sequence specification

<sequence specification> -> **sequence (** <identlist> **) ;**
<identlist> as defined in Subclause 5.4.5.4 Route statement.

The **sequence** statement allows the specification of the ordering of execution of instrument instantiations by the run-time scheduler. The identlist references a list of instruments which describes a partial ordering on the set of instruments. If instrument **a** and instrument **b** are referenced in the same **sequence** statement with **a** preceding **b**, then instantiations of instrument **a** shall be executed strictly before instantiations of instrument **b**.

There are several default sequence rules:

1. The special instrument **startup** is instantiated and the instantiation executed at the i-rate at the very beginning of the orchestra.

2. Any instrument instances corresponding to the **startup** instrument are executed first in a particular orchestra cycle.

3. If **output_bus** is sent to an instrument, the instrument instantiation corresponding to that **send** statement is the last instantiation executed in the orchestra cycle.

4. For each instrument routed to a bus which is sent to an effect instrument, instantiations of the routed instrument are executed before instantiations of the effect instrument. If loops are created using **route** and **send** statements, the ordering is resolved syntactically: whichever **send** statement occurs latest, that instrument instantiation is executed latest.

Default rules 2, 3, and 4 may be overridden by use of the **sequence** statement. Rule 1 cannot be overridden.

It is a syntax error if explicit **sequence** statements create loops in ordering. Any **send** statements which are the “backward” part of an implicit **send** loop have no effect.

If the sequence of two instruments is not defined by the default or explicit sequence rules, their instantiations may be executed in any order or in parallel.

It is not possible to specify the ordering of multiple instantiations of the same instrument; these instantiations can be run in any order or in parallel.

EXAMPLES

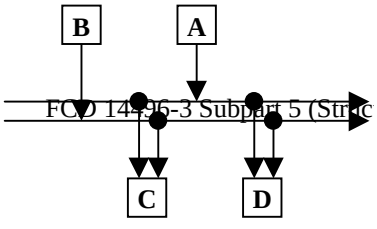
An orchestra consists of five instruments, **a, b, c, d,** and **e**.

1. The following code fragment

```
route(bus1, a, b);
send(c; ; bus1);
```

is legal and specifies (using the default sequencing rules) that instantiations of instruments **a** and **b** shall be executed strictly before instantiations of instrument **c**. This ordering applies to all instantiations of instrument **c**, not only to the one corresponding to the **send** statement. No ordering is specified between instruments **a** and **b**.

2. The following code fragment



```
route(bus1, a, b);
send(c; ; bus1);
sequence(c,a);
send(d; ; bus1);
```

is legal and specifies that instantiations of instrument **b** shall be executed first, followed by instantiations of instrument **c**, followed by instantiations of instrument **a**, followed by instances of instrument **d**. The ordering of **b** and **c**, and **a** and **b** with **d**, follows from default rule 3; the placement of instrument **c** follows from the explicit **sequence** statement, which overrides default rule 3. Due to this ordering, the output samples of instrument **a** are not provided to instrument **c** (they get put on the bus "too late"), and however many channels of output this represents are set to 0 in instrument **c**. The output samples of instrument **a** are provided to instrument **d**.

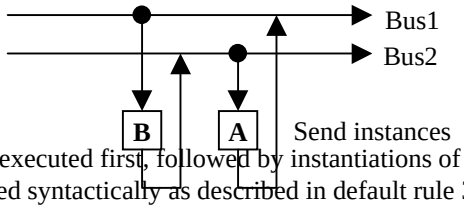
3. The following code fragment

```
sequence(a,b);
sequence(b,c,d);
sequence(c,e);
sequence(e,a);
```

is illegal, as it contains an explicit loop in sequencing.

4. The following code fragment

```
route(bus1, a);
send(b; ; bus1);
route(bus2, b);
send(a; ; bus2);
```



is legal, and specifies that instantiations of instrument **b** are executed first, followed by instantiations of instrument **a**. There is an implicit loop here which is resolved syntactically as described in default rule 3. Due to this ordering, the output values of instrument **a** are not provided to instrument **b**. Note that for deciding sequencing, only the order of **send** statements matters, not the order of **route** statements.

5.4.6 Instrument definition

5.4.6.1 Syntactic form

<instrument definition> -> **instr <ident> (** <identlist> **)** [**preset <int>**] [**channel <int>**] **{**
<instr variable declarations>
<block> **}**

An instrument definition has several elements. In order, they are

1. An identifier which defines the name of the instrument,
2. A list of zero or more identifiers which define names for the parameter fields, also called pfields, of the instrument,
3. An optional preset value for specifying a MIDI preset mapping,
4. An optional channel value for specifying a MIDI channel mapping,

5. A list of zero or more instrument variable declarations, and

6. A block of statements defining the executable functionality of the instrument.

5.4.6.2 Instrument name

Any identifier may serve as the instrument name except that the instrument name shall not be a reserved word (see Subclause 5.4.9 Reserved words), the name of a core opcode (see Subclause 5.5 SAOL core opcode definitions and semantics), or the name of a core wavetable generator (see Subclause 5.6 SAOL core wavetable generators). An instrument name may be the same as a variable in local or global scope; there is no ambiguity so created, since the contexts in which instrument names may occur are very restricted.

No two instruments or opcodes in an orchestra shall have the same name.

5.4.6.3 Parameter fields

<identlist> -> as given in Subclause 5.4.5.4 Route statement

The parameter fields, also called pfields, of the instrument, are the interface through which the instrument is instantiated. In the instrument code, the pfields have the rate semantics of i-rate local variables. Their values shall be set on instrument instantiation, before the creation of local variables, with the appropriate values as given in the score line, score event, MIDI event, **send** statement, or **instr** statement corresponding to the instrument instantiation.

5.4.6.4 Preset and channel tags

5.4.6.4.1 Preset tag

The **preset** tag specifies the preset number of the instrument. When MIDI program change events arrive in a MIDI stream or MIDI file controlling the orchestra, the program change numbers refer to the **preset** tags given to the various instruments. No more than one instrument may have the same preset number; if multiple instruments in an orchestra specify the same **preset** tag, the one occurring syntactically last is assigned that preset number. If a **preset** tag is not associated with a particular instrument, then that instrument has no preset number and cannot be referenced with a program change.

Preset values are fixed and do not change throughout an orchestra synthesis process.

See Subclause 5.10 MIDI semantics for more normative semantics on MIDI control of orchestras

5.4.6.4.2 Channel tag

The **channel** tag specifies the channel assignment of the instrument. When MIDI instructions arrive in a MIDI stream or MIDI file controlling the orchestra, the channel numbers refer to all instruments with the given channel assignment. When continuous control instructions arrive in a MIDI stream or MIDI file, the control instructions refer to all instrument instantiations created from instruments with the given channel assignment. Zero or more instruments may have the same channel value. If a **channel** tag is not associated with a particular instrument, then that instrument is on channel 1 by default.

Channel values may be changed with the MIDI program change instruction.

See Subclause 5.10 MIDI semantics for more normative semantics on MIDI control of orchestras.

5.4.6.5 Instrument variable declarations

5.4.6.5.1 Syntactic form

<instr variable declarations> -> <instr variable declarations> <instr variable declaration>
<instr variable declarations> -> **<NULL>**

<instr variable declaration> -> [<sharing tag>] **ivar** <namelist> **;**
<instr variable declaration> -> [<sharing tag>] **ksig** <namelist> **;**
<instr variable declaration> -> **asig** <namelist> **;**
<instr variable declaration> -> <table declaration> **;**
<instr variable declaration> -> <sharing tag> **table** <identlist> **;**
<instr variable declaration> -> **oparray <ident> [**<array length> **] ;**
<instr variable declaration> -> <tablemap declaration> **;**

<sharing tag> -> **imports**
<sharing tag> -> **exports**
<sharing tag> -> **imports exports**

<tablemap declaration> -> **tablemap <ident> (** <identlist> **) ;**

<array length> and <namelist> as defined in Subclause 5.4.5.3.2 Signal variables
<table declaration> as defined in Subclause 5.4.5.3.3 Wavetable declarations
<identlist> as defined in Subclause 5.4.5.4 Route statement

Instrument variable declarations declare variables which may be used within the scope of an instrument. Any rate type variable, as well as wavetables, tablemaps, and wavetable placeholders, may be declared in an instrument. An instrument variable declaration is either a wavetable declaration, or an type name, possibly preceded by a sharing tag followed by a list of name declarations, or a sharing tag followed by the token **table** followed by a list of identifiers referencing global or future wavetables, or an opcode-array declaration, or a table-map definition.

5.4.6.5.2 Wavetable declaration

The syntax and semantics of Subclause 5.4.5.3.3 Wavetable declarations hold for instrument local wavetables, with the following exceptions and additions:

An instrument local wavetable is available only within the local scope of a single instrument instantiation. As such, it shall be created and initialised with data at the instrument instantiation time, immediately after the pfield values are assigned from the calling parameters. It may be deleted and freed when that instrument instantiation terminates.

Not every expression which is i-rate is legal as part of the table parameter list. Reference to constants, pfields, and i-rate standard names is allowed. However, the instrument wavetable initialisation shall occur before the initialisation pass through the instrument code, and so reference to local i-rate variables is prohibited.

5.4.6.5.3 Signal variables

The syntax and semantics of Subclause 5.4.5.3.2 Signal variables hold for instrument local signal variables, with the following exceptions and additions:

A local name declaration specifies that a name token shall be created and space equal to one signal value allocated for variable storage in each instrument instantiation associated with the instrument definition. A

local array declaration specifies that a name token shall be created and space equal to the specified number of signal values allocated in each instrument instantiation associated with the instrument definition.

The sharing tags **imports** and/or **exports** may be used with local i-rate or k-rate signal variable declaration. They shall not be used with a-rate variables. If the **imports** tag is used, then the variable value shall be replaced with the value of the global variable of the same name at instrument initialisation time (for i-rate signal variables) or at the beginning of each control pass (for k-rate signal variables). The **imports** tag may be used for a local k-rate signal variable even if there is no global variable of the same name, in which case it is an indication that the k-rate variable so tagged may be modified with **control** lines in a SASL score. The **imports** tag shall not be used for local i-rate signal variables when there is no global variable of the same name.

If the **exports** tag is used, then the value of the global variable of the same name shall be replaced with the value of the local signal variable after instrument initialisation (for i-rate signal variables) or at the end of each control pass (for k-rate signal variables). The **exports** tag shall not be used if there is no global variable of the same name.

If, for a particular signal variable, the **imports** and/or **exports** tags are used, and there is a global variable with the same name, then the array width of the local and global variables must be the same.

If, for a particular local variable, the **imports** tag is not used, then its value is set to 0 before instrument initialisation.

If, for a particular local variable declaration, the **imports** and **exports** tags are not used, even if there is a global variable of the same name, there is no semantic relationship between the two variables. The construction is syntactically legal.

5.4.6.5.4 Wavetable placeholder

The sharing tags **imports** and **exports** may be used to reference global and future wavetables. In this case, the local declaration of the table reference is termed a wavetable placeholder. The wavetable placeholder definition does not contain a full wavetable definition, but only a reference to a global or future wavetable name.

If only the **imports** tag is used, and there is a global wavetable with the same name, then at instrument instantiation time, the current contents of the global wavetable are copied into a local wavetable with that name. If the contents of the global wavetable are modified after a particular instrument instantiation referencing that global wavetable is created, the new contents of the global wavetable shall not be copied into the instrument instantiation. Also, if the contents of the local wavetable are modified, these changes shall not be reflected in the global wavetable.

If the **imports** and **exports** tags are both used, and there is a global wavetable with the same name, then at instrument instantiation time and at the beginning of each control pass, the current contents of the global wavetable are made available to a local wavetable with that name. “Made available” in the preceding sentence means that access may be either in the form of copying data from one wavetable to another or by pointer reference to the same memory space, or by any equivalent implementation. Also, at the end of instrument instantiation and at the end of each control pass, the current contents of the local wavetable are similarly made available to the global wavetable with the same name.

It is not permissible to use the **exports** tag alone for a wavetable placeholder.

If the **imports** tag is used, and there is no global wavetable with the same name, then the reference is to a *future wavetable* which will be provided in the bitstream. When the instrument is instantiated, the contents of the most recent wavetable provided in the bitstream with the same name shall be copied into the local

wavetable. If no wavetable has been provided in the bitstream with the same name as the wavetable placeholder at the time of instrument instantiation, then the bitstream is invalid.

It is not permissible to use the **exports** tag if there is no global wavetable with the same name.

5.4.6.5.5 Opcode array declaration

An opcode array, or "oparray" declaration, declares several opcode states for a particular opcode that may be used by the current instrument or opcode. By declaring the states in this manner, access to them is available through the oparray expression, see Subclause 5.4.6.7.7 Oparray call. The identifier in the declaration shall be the name of a core opcode or an user-defined opcode declared elsewhere in the orchestra. The array length declares how many states are available for access to this oparray in the local code block; it shall be an integer value or the special tag **inchannels** or **outchannels.**

It is a syntax error if more than one oparray declaration references the same opcode name in a single instrument or opcode.

5.4.6.5.6 Table map definition

<table map definition> -> **tablemap <ident> (** <identlist> **)**

<identlist> as defined in Subclause 5.4.5.4 Route statement.

A table map is a data structure allowing indirect reference of wavetables via array notation. The identifier names the table map; it shall not be the same as the name of any other signal variable or other restricted word in the local scope. The identifier list gives a number of wavetable names for use with the table map. Each of these names shall correspond to a wavetable definition or wavetable placeholder within the current scope. The **tablemap** declaration may come before, after, or in the midst of wavetable declarations and wavetable placeholders in the instrument. All wavetables in the scope of the instrument may be referenced in a **tablemap**, regardless of the syntactic placement of the **tablemap**.

When the tablemap name is used in an array-reference expression (see Subclause 5.4.6.7.5 Array reference), the index of the expression determines to which of the wavetables in the list the expression refers. The first wavetable in the list is number 0, the second number 1, and so on.

EXAMPLE

For the following declarations

```
table t1(harm,2048,1);
imports table t2;
table t3(random,32,1);

tablemap tmap(t1,t2,t3,t2);
ivar i,x,y,z;
```

the following two code blocks are identical in semantics:

BLOCK 1

```
i = 3;
x = tableread(tmap[0],4);
y = tableread(tmap[i],3);
z = tableread(tmap[i > 4 ? 1 : 2],5);
```

BLOCK 2

```
x = tableread(t1,4);
y = tableread(t2,3);
z = tableread(t3,5);
```

Note that, like table references, array expressions using tablemaps may only occur in the context of an opcode or oparray call to an opcode accepting a wavetable reference.

5.4.6.6 Block of code statements

5.4.6.6.1 Syntactic form

<block> -> <statement> [<block>]
<block> -> **<NULL>**

<statement> -> <lvalue> **=** <expr> **;**
<statement> -> <expr> **;**
<statement> -> **if (** <expr> **) {** <block> **}**
<statement> -> **if (** <expr> **) {** <block> **} else {** <block> **}**
<statement> -> **while (** <expr> **) {** <block> **}**
<statement> -> **instr <ident> (** <expr list> **) ;**
<statement> -> **output (** <expr list> **) ;**
<statement> -> **sbsynth (** <expr list> **;** <identlist> **;** <expr list> **) ;**
<statement> -> **spatialize (** <expr list > **) ;**
<statement> -> **outbus (<ident> ,** <expr list> **) ;**
<statement> -> **extend (** <expr> **) ;**
<statement> -> **turnoff ;**

<expr list> -> <expr> [**,** <expr list>]
<expr list> -> **<NULL>**

<lvalue> as given in Subclause 5.4.6.6.2 Assignment.
<expr> as given in Subclause 5.4.6.7 Expression.

A block is a sequence of zero or more statements. A statement shall take one of 12 forms, which are enumerated and described in the subsequent Subclauses. Each statement has rate-semantics rules governing the rate of the statement, the rate contexts in which it is allowable, and the times at which various subcomponents shall be executed.

To execute a block of statements at a particular rate, the statements within the block shall be executed, each at that rate, in such order as to produce equivalent results to executing the statements sequentially in linear order, according to the semantics below governing each type of statement

5.4.6.6.2 Assignment

<statement> -> <lvalue> **=** <expr> **;**

<lvalue> -> **<ident>**
<lvalue> -> **<ident> [** <expr> **]**

<expr> as given in Subclause 5.4.6.7 Expression.

An assignment statement calculates the value of an expression and changes the value of a signal variable or variables to match that value.

The lvalue, or left-hand-side value, denotes the signal variable or variables whose values are to be changed. An lvalue may be a local variable name, in which case the denotation is to the storage space associated with that name. An lvalue may also be a local array name, in which case the denotation is to the entire array storage space. An lvalue may also be a single element of a local array denoted by indexing a local array name with an expression. An lvalue shall not be a table reference or tablemap expression.

If the lvalue denotes an entire array, the right-hand-side expression of the assignment shall denote an array-valued expression with the same array length, or a single value, otherwise the construction is syntactically illegal.

If the lvalue denotes a single value, the right-hand-side expression of the assignment shall denote a single value, otherwise the construction is syntactically illegal.

The rate of the lvalue is the rate of the signal variable, if there is no indexing expression, or the faster of the rate of the signal array denoted by the indexing expression and the rate of the indexing expression, if there is an indexing expression.

The rate of the right-hand side is the rate of the right-hand-side expression.

The rate of the statement is the rate of the lvalue, however, the statement is illegal if the rate of the right-hand side is faster than the rate of the lvalue.

The assignment shall be performed as follows:

At every pass through the statement occurring at lesser or equal rate to the rate of the assignment, the right-hand side expression shall be evaluated. At each pass equal in rate to the lvalue, the storage space denoted by the lvalue shall be updated to be equal to the value of the right-hand expression. If the lvalue denotes an entire array, and the right-hand-side expression a single value, then each of the values of each of the elements of the array shall be changed to the single right-hand-side value.

5.4.6.6.3 Null assignment

<statement> -> <expr> **;**

A null assignment contains only an expression; it is provided so that opcodes which do not have useful return values need not be used in the context of an assignment to a dummy variable.

The rate of the statement is the rate of the expression. The expression may be single-valued or array-valued; it shall not be a table reference.

The null assignment shall be performed as follows:

At every pass through the statement occurring at lesser or equal rate to the rate of the statement, the expression shall be evaluated.

5.4.6.6.4 If

<statement> -> **if (** <expr> **) {** <block> **}**

An **if** statement allows conditional evaluation of a block of code. The expression which is tested in the **if** statement is termed the guard expression.

The rate of the statement is the rate of the guard expression, or the rate of the fastest statement in the guarded code block, whichever is faster.

It is not permissible for the block of code governed by the **if** statement to contain statements slower than the guard expression. It is further not permissible for any of the statements in the governed block of code to contain calls to opcodes which would be executed slower than the guard expression. The guard expression shall be a single-valued expression.

EXAMPLE

The following code fragment is illegal:

```
asig a;
ksig k;

a = 0; while (a < 20) {
 k = kline(...);
}
```

The example is illegal because the **kline** assignment statement is slower than the guard **a < 20**. Even if the assignment were to an a-rate variable ("a2 = kline(...)"), thus making the assignment statement an a-rate statement, the example would be illegal, because the **kline** opcode itself is slower than the guard expression.

The **if** statement shall be executed as follows:

At every pass through the statement occurring at equal rate to the rate of the statement, the guard expression shall be evaluated. If the guard statement evaluates to any non-zero value in a particular pass, then the block of code shall be evaluated at the rate corresponding to that pass.

5.4.6.6.5 Else

<statement> -> **if (** <expr> **) {** <block> **} else {** <block> **}**

An **else** statement allows disjunctive evaluation of two blocks of code. The expression which is tested in the **else** statement is termed the guard expression.

The rate of the statement is the rate of the guard expression, or the rate of the fastest statement in the first guarded block of code, or the rate of the fastest statement in the second guarded block of code, whichever is fastest.

It is not permissible for the blocks of code governed by the **else** statement to contain statements slower than the guard expression. It is further not permissible for any of the statements in the governed blocks of code to contain calls to opcodes which would be executed slower than the guard expression. The guard expression shall be a single-valued expression.

The **else** statement shall be executed as follows:

At every pass through the statement occurring at equal rate to the rate of the statement, the guard expression shall be evaluated. If the guard expression evaluates to any non-zero value in a particular pass, then the first guarded block of code shall be at the rate corresponding to that pass. If the guard statement evaluates to zero in a particular pass, then each statement in the second guarded block of code shall be so evaluated.

5.4.6.6.6 While

<statement> -> **while (** <expr> **) {** <block> **}**

The **while** statement allows a block of code to be conditionally evaluated several times in a single rate pass. The expression which is tested in the **while** statement is termed the guard expression.

The rate of the **while** statement is the rate of the guard expression.

It is not permissible for the block of code governed by the **while** statement to contain statements which run at a rate other than the rate of the guard expression. It is further not permissible for any of the statements in the governed block of code to contain calls to opcodes which would be executed at a rate other than the rate of the guard expression. The guard expression shall be a single-valued expression.

The **while** statement shall be executed as follows:

At every pass through the statement occurring at equal rate to the rate of the statement, the guard expression shall be evaluated. If the guard expression evaluates to any non-zero value in a particular pass, then each statement in the guarded block of code shall be evaluated according to the particular rules for that statement, and then the guard expression re-evaluated, iterating until the guard expression evaluates to zero.

5.4.6.6.7 Instr

<statement> -> **instr <ident> (** <expr list> **) ;**

The **instr** statement allows an instrument instantiation to dynamically create other instrument instantiations, for layering or synthetic-performance techniques. It shall consist of an identifier referring to an instrument defined in the current orchestra, a duration, and a list of expressions defining parameters to pass to the referenced instrument.

It is a syntax error if the number of expressions in the expression list is not one greater than the number of pfields accepted by the referenced instrument (the first expression is the duration). Each expression in the expression list shall be a single-valued expression.

The rate of the **instr** statement is the rate of the fastest expression in the expression list.

It is not permissible for the rate of the **instr** statement to be a-rate.

The **instr** statement shall be executed as follows:

At every pass through the statement occurring at lesser or equal rate to the rate of the statement, each of the expressions in the expression list is evaluated. Then, at every pass through the statement occurring at equal rate to the rate of the statement, a new instrument instantiation is created, where the duration of the new instantiation is the value of the first expression in the expression list, and the values of the instrument p-fields in the new instantiation are set to the values of the remaining expressions.

The i-rate pass through the new instrument instantiation shall be executed immediately upon its creation, before any more statements from the block of code containing the **instr** statement are executed. However, any changes to global i-rate variables made in the new instance during its i-rate pass are not respected in this instrument (the “caller”). i-rate variables imported from the global context are set only during the initialisation pass of each instance, and never change afterward. The first k-rate and a-rate passes through the new instrument instantation shall be executed as appropriate to the sequencing relation between the instantiating and instantiated instruments; that is, if the new instrument is sequenced later than the instantiating instrument, the new instantiation shall be executed at some later time in the same orchestra pass, but if the new instrument is sequenced earlier than the instantiating instrument, then the new instantiation shall not be executed in k-time or a-time until the subsequent orchestra pass.

5.4.6.6.8 Output

<statement> -> **output (** <expr list> **) ;**

The **output** statement creates audio output from the instrument. This output does not get turned directly into sound, but rather gets buffered either on one or more busses based on instructions given in **route** statements (Subclause 5.4.5.4 Route statement) or on the special bus **output_bus** by default. However, if the current instrument instantiation is the one created with a **send** statement referencing the special bus **output_bus**, then the output of the current instantiation, created by summing its calls to **output**, may be turned directly into sound.

The expression list shall contain at least one expression.

The rate of the **output** statement is a-rate.

All statements within a orchestra which reference the same bus, whether through explicit sends, calls to **outbus** or **sbsynth**, or by default routing to the special bus **output_bus**, shall have compatible numbers of expression parameters representing output channels. "Compatible" means that if any calls to **output** for a particular bus reference more than one expression parameter, then all other calls to **output** referencing this bus shall have either the same number of expression parameters, or else only a single expression parameter. In addition, the number of channels of the special bus **output_bus** shall be the same as the global **outchannels** parameter and uses of **output** by instrument instances which are implicitly or explicitly routed to **output_bus** shall be compatible with this number of channels.

The **output** statement is executed as follows:

At each k-rate pass through the instrument, an output buffer, with number of channels determined by the compatibility rules above, shall be cleared to zero values. At every pass through the statement at any rate, the expression parameters shall each be evaluated. Then, if the pass is at a-rate, the expression parameter values shall be placed in the output buffer: if the **output** statement has more than one parameter expression, then the value of each parameter shall be added to the current value of the output buffer in the corresponding channel. If the **output** statement has only one parameter expression, then the value of that expression shall be added to the current value of the output buffer in each channel.

The expression parameters to the **output** statement may be array-valued, in which the mapping described in the preceding paragraph is not from expressions to buffer channels, but from array value channels to buffer channels.

EXAMPLE

The following code fragment

```
asig a[2], b;

. . .

output(a,b);
output(a[1],b,b);
output(b);
```

is legal and describes an instrument which outputs three channels of sound. The first channel of output contains the value `a[0] + a[1] + b`, the second `a[1] + b + b`, and the third `b + b + b.`

After each a-rate pass through the instrument instantiation during a particular orchestra pass, the values in the output buffer shall be added channel-by-channel to the current values of the bus or busses referenced by the **route** expression or expressions which also reference this instrument. If there are no such **route** statements, the values in the output buffer shall be added channel-by-channel to the current values of the special bus **output_bus**. If this is the instrument instantiation created by referencing the special bus

output_bus in a **send** statement, then the preceding two sentences do not hold, and instead the values in the output buffer are the output of the orchestra.

5.4.6.6.9 Wavetable bank synthesis (sbsynth)

<statement> -> **sbsynth (** <expr list> **;** <identlist> **;** <expr list> **) ;**

The **sbsynth** statement allows the use of the standardised bank synthesis procedure (see Subclause 5.9 Sample Bank syntax and semantics) within a SAOL instrument. There are three parameter lists to the **sbsynth** statement:

1. The first list shall be a list of three expressions, with the first expression corresponding to the sample bank number, the second to the MIDI pitch, and the third to the MIDI velocity. Each expression shall be single-valued.

2. The second list shall be a list of one, two, or three identifiers referencing busses which are defined in **send** statements, Subclause 5.4.5.5 Send statement. The first bus is the stereo output bus, and the **sbsynth** statement contains two parameter expressions to this bus for the purposes of the compatibility rules in Subclause 5.4.6.6.8 Output. The second bus, if given, is the mono reverb bus, and the **sbsynth** statement contains one parameter expression to this bus for the purposes of the compatibility rules in Subclause 5.4.6.6.8 Output. The third bus, if given, is the mono chorus bus, and the **sbsynth** statement contains one parameter expression to this bus for the purposes of the compatibility rules in Subclause 5.4.6.6.8 Output. It is a syntax error if busses are given in this list but are not defined in **send** statements in the global orchestra block.

3. The third list shall be a list of zero, one, two, or three expressions. The first expression, if given, corresponds to the MIDI bank number. If there are no expressions in the list, the MIDI bank number is the default value 1. The second expression, if given, corresponds to the MIDI channel number. If there are fewer than two expressions in the list, the MIDI channel number is the default value given by the instrument channel number (Subclause 5.4.6.4.2 Channel tag). The third expression, if given, corresponds to the MIDI preset number. If there are fewer than three expressions in the list, the MIDI preset number is the default value given by the instrument preset value (Subclause 5.4.6.4.1 Preset tag), if there is one, or else the default value 1. Each expression given shall be single-valued.

The rate of the **sbsynth** statement is a-rate.

The **sbsynth** statement shall be executed as follows:

At each pass through the instrument, the expressions in the first list, and any expressions in the third list, shall be evaluated. Then, at each pass through the instrument at a-rate, the wavetable bank synthesis procedure described in Subclause 5.9 Sample Bank syntax and semantics shall be executed using the six parameters as defined in Subclauses (1) and (3) of the numbered list in this Subclause. The output of the stereo output calculation in this procedure shall be added to the first bus referenced in the second list; then, if there is a second bus referenced in the second list, the output of the mono reverb calculation in this procedure shall be added to it; then, if there is a third bus referenced in the second list, the output of the mono chorus calculation shall be added to it.

The **sbsynth** statement shall not be used in an instrument which is the target of a **send** statement referencing the special bus **output_bus**.

5.4.6.6.10 Spatialize

<statement> -> **spatialize (** <expr list > **) ;**

The **spatialize** statement allows instruments to produce spatialised sound, using non-normative methods that are implementation-dependent.

The expression list shall contain four expressions. The second, third, and fourth shall not be a-rate expressions. The first expression represents the audio signal to be spatialised; the second, the azimuth (angle) from which the source sound should apparently come, measuring in radians clockwise from 0 azimuth directly in front of the listener; the third, the elevation angle from which the sound source should apparently come, measuring in radians upward from 0 elevation on the listener's horizontal place; and the fourth, the distance from which the sound source should apparently come, measuring in metres from the listener's position. Each of the four expressions shall be single-valued.

The rate of the **spatialize** statement is a-rate.

The **spatialize** statement shall be executed as follows:

At each pass through the instrument, the expressions in the expression list shall be evaluated. Then, at each pass through the instrument at a-rate, the sound signal in the first expression shall be presented to the listener as though it has arrived from the azimuth, elevation, and distance given in the second, third, and fourth expressions. No normative requirements are placed on this spatialisation capability, although terminal implementors are encouraged to provide the maximum sophistication possible.

The sound produced via the **spatialize** statement is turned directly into orchestra output; it shall not be affected by bus routings or further manipulation within the orchestra. If multiple calls to **spatialize** occur within an orchestra, the various sounds so produced shall be mixed via simple summation after spatialisation. Similarly, if both spatialised and non-spatialised sound is produced within an orchestra, the final orchestra output of all non-spatialised sound shall be mixed via simple summation with the various spatialised sounds for presentation. The sound produced via each **spatialize** statement shall have as many channels as the global orchestra number of output channels (see Subclause 5.4.5.2.4 outchannels parameter) in order to enable this mixing.

5.4.6.6.11 Outbus

<statement> -> **outbus (<ident> ,** <expr list> **) ;**

The **outbus** statement allows instruments to place dynamically-calculated signals on busses. The identifier parameter shall refer to the name of a bus defined with a **send** statement in the global block. The remaining expressions represent signals to place on the bus.

It is a syntax error if there are no expressions in the expression list, or if the identifier does not refer to a bus defined in the global block with a **send** statement. The number of expressions in the expression list shall be compatible with other statements making reference to the same bus, as defined in Subclause 5.4.6.6.8 Output.

The rate of the **outbus** statement is a-rate.

The **outbus** statement shall be executed as follows:

At each pass through the statement, the expression list shall be evaluated. Then, at each a-rate pass through the statement, the expression values shall be added to the current values of the referenced bus. If there is more than one expression in the expression list, then each expression value shall be added to the

corresponding channel of the referenced bus. If there is only one expression in the expression list, then the value of that expression shall be added to each channel of the referenced bus.

The expressions in the expression list may be array-valued, in which case the semantics are analogous to those in Subclause 5.4.6.6.8 Output.

The **outbus** statement shall not be used in an instrument which is the target of a **send** statement referencing the special bus **output_bus**.

5.4.6.6.12 Extend

<statement> -> **extend (** <expr> **) ;**

The **extend** statement allows an instrument instantiation to dynamically lengthen its duration.

The expression parameter shall not be a-rate. The expression shall be single-valued.

The rate of the **extend** statement is the rate of the expression parameter.

The **extend** statement shall be executed as follows:

At each pass through the statement at lesser or equal rate to the rate of the statement, the expression shall be evaluated. Then, at each pass through the statement at the rate of the statement, the duration of the instrument instantiation shall be extended by the amount of time, in seconds, given by the value of the expression. That is, if the instrument instance had been previously scheduled to be terminated at time *t*, then after a call to **extend** with an expression evaluating to *s*, the instrument instance shall be scheduled to terminate at time $t+s$**.** If the instrument instance had no scheduled termination time (its duration was –1 on instantiation), **extend** with an expression evaluating to *s* shall schedule termination of the instrument at time $T + s$, where *T* is the current orchestra time.

NOTE

The parameter of **extend** is specified in seconds, not in beats. If it is desirable to have time-extension dependant on tempo in a particular composition, the content author must enable this by rescaling the parameter by the current value of **tempo** (Subclause 5.4.6.8.8 tempo).

extend may be called with a negative argument to shorten the duration of a note; if $t+s < T$ (that is, if the negatively extended duration has already been exceeded in the instantiation), then the statement acts as the **turnoff** statement, see Subclause 5.4.6.6.13 Turnoff.

When the **extend** statement is called, the standard name **dur** shall not be updated to reflect the new duration, but keeps the value of the original duration.

5.4.6.6.13 Turnoff

<statement> -> **turnoff ;**

The **turnoff** statement allows an instrument instantiation to dynamically decide to terminate itself.

The rate of the **turnoff** statement is k-rate.

The **turnoff** statement shall be executed as follows:

When the **turnoff** statement is reached at k-rate, the instrument instance shall be scheduled to terminate after the following k-cycle; that is, if the current orchestra time is *T* and the k-pass duration *k*, the instrument instantiation shall be scheduled to terminate at time *T+k*.

The **turnoff** statement shall not update the **dur** standard name.

The **turnoff** statement shall not be executed in an instrument instance which is created as the result of a **send** statement referencing the special bus **output_bus**.

NOTE

turnoff does not destroy the instantiation immediately; the instantiation is executed for one more orchestra pass, to allow the instrument time to examine the **released** variable. Instruments may call **turnoff** and then "save" themselves on the subsequent k-cycle by calling **extend**.

5.4.6.7 Expressions

5.4.6.7.1 Syntactic form

<expr> -> **<ident>**
<expr> -> **<number>**
<expr> -> **<int>**
<expr> -> **<ident> [** <expr> **]**
<expr> -> **<ident> (** <expr list> **)**
<expr> -> **<ident> [** <expr> **] (** <expr list> **)**
<expr> -> <expr> **?** <expr> **:** <expr>
<expr> -> <expr> <binop> <expr>
<expr> -> **!** <expr>
<expr> -> **-** <expr>
<expr> -> **(** <expr> **)**

<binop>-> **+**
<binop>-> **-**
<binop>-> *****
<binop>-> **/**
<binop>-> **==**
<binop>-> **>=**
<binop>-> **<=**
<binop>-> **!=**
<binop>-> **>**
<binop>-> **<**
<binop>-> **&&**
<binop>-> **||**

An expression can take one of several forms, the semantics of which are enumerated in the Subclauses below. Each form has both rate semantics, which describe the rate of the expression in terms of the rates of the subexpressions, and value semantics, which describe the value of the expression in terms of the values of the subexpressions. The syntax above is ambiguous for many expressions; disambiguating precedence rules are given in Subclause 5.4.6.7.14 Order of operations.

5.4.6.7.2 Properties of expressions

Each expression is conceptually labelled with two properties: its rate and its width. The rate of an expression determines how fast the value of that expression might change; the width of an expression determines how many channels of sound or other data are represented by the expression. In each expression type, the rate and width of the expression are determined from the type of the expression, and perhaps from the rate and width of the component subexpressions.

NOTE

Any name declared as an array is an array-valued variable regardless of its length. That is, a variable declared as **asig name[1]** is not a single-valued variable.

5.4.6.7.3 Identifier

<expr> -> **<ident>**

An identifier expression denotes a storage location or locations which contain values stored in memory. It is illegal to reference an identifier which is not declared in the local instrument or opcode scope, and which is not a standard name (see Subclause

Expressions bind in the order prescribed in the following table. That is, operations listed higher in the table are performed before operations lower in the table whenever the ordering is syntactically ambiguous. Operations listed on the same row associate left-to-right. That is, the leftmost expression is performed first.

Operator	Function
!	not
-	unary negation
*, /	multiply, divide
+, -	add, subtract
<, >, <=, >=	relational
==, !=	equality
&&	logical and
||	logical or
?:	switch

5.4.6.8 Standard names).

The rate of an identifier expression is the rate type at which the identifier was declared, or is implicitly declared in the case of standard names. The rate of a **table** identifier is i-rate.

If the identifier denotes a single-valued name (i.e., one which is not an array type), then the value of the identifier expression is the value stored in memory associated with that identifier in the current scope, and the width of the expression is 1.

If the identifier denotes an array-valued name, then the value of the identifier expression is the ordered sequence of values stored in memory and associated with that identifier in the current scope, and the width of the expression is the width of the array so denoted.

If the identifier denotes a table, then the value of the identifier expression is a reference to the table with the given name. Table references may only appear in calls to opcodes. A table reference has width 1.

5.4.6.7.4 Constant value

<expr> -> **<number>**
<expr> -> **<int>**

A constant value expression denotes a single number.

The rate of a constant value expression is i-rate.

The width of a constant value expression is 1.

The value of a constant expression is the value of the number denoted by the constant. The value of a constant expression is always a floating-point value, whether the token or lexical expression denoting the value was an integer or floating-point token or expression.

5.4.6.7.5 Array reference

<expr> -> **<ident> [** <expr> **]**

An array reference expression allows the selection of one value from an array of several. The identifier in the array-reference syntax is termed the array name, and the expression the index expression. It is illegal to use an identifier in an array reference which is neither declared in the local instrument or opcode scope as an array, nor implicitly defined as an array-valued standard name or table map.

The index expression shall have width 1.

The rate of an array reference expression is the rate of the array name (which is the rate at which the array name was declared explicitly or implicitly), or the rate of the index expression, whichever is faster.

The width of an array reference expression is 1.

If the referenced array is an array-valued signal variable, then the value of the array reference expression is the value of that element of the sequence of values in the array storage corresponding to the value of the indexing expression, where element 0 corresponds to the first value in the sequence. It is a run-time error if the value of the indexing expression is less than 0, or equal to or greater than the declared size of the array. If the indexing expression is not an integer, it is rounded to the nearest integer.

If the referenced array is a table map, then the value of the array reference expression is a reference to that element of the sequence of tables corresponding to the value of the index expression, where element 0 corresponds to the first table in the sequence. It is a run-time error if the value of the indexing expression is less than 0, or equal to or greater than the declared size of the table map. If the indexing expression is not an integer, it is rounded to the nearest integer. Table references may only appear in calls to opcodes. See also the example in Subclause 5.4.6.5.6 Table map definition.

NOTE

The syntax `t[i]`, where **t** is a table rather than a table map, is illegal. The **tableread** core opcode is used to directly access elements of a wavetable. See Subclause 5.5.6 Table operations.

5.4.6.7.6 Opcode call

<expr> -> **<ident> (** <expr list> **)**

An opcode call expression allows the use of processing functionality encapsulated within an opcode.

The identifier is termed the opcode name, and the expression list the actual parameters of the opcode call expression. It is illegal to use an identifier which is not the name of a core opcode and is also not the name of a user-defined opcode declared elsewhere in the orchestra. For user-defined opcodes, the number of actual parameters shall be the same as the number of formal parameters in the opcode definition. For core opcodes without variable argument lists, the number of actual parameters required varies from opcode to opcode; see Subclause

The following words are reserved, and shall not be used as identifiers in a SAOL orchestra or score.

aopcode asig else exports extend global if imports inchannels inputmod instr iopcode ivar kopcode krate ksig map oparray opcode outbus outchannels output return route send sequence sbsynth spatialize srate table tablemap template turnoff while with xsig

Also, variable names starting with **_sym_** are reserved for implementation-specific use (for example, bitstream detokenisation – see Annex B).

5.5 SAOL core opcode definitions and semantics. If a particular formal parameter in an opcode definition is an array, then the corresponding actual parameter shall be an array-typed expression of equal width. If a particular formal parameter in an opcode definition is a table, then the corresponding actual parameter shall be a table reference.

If a particular formal parameter in an opcode definition is at a particular rate, then the corresponding actual parameter expression shall not be at a faster rate.

The rate of the opcode call expression is determined according to the rules in Subclause **Fehler! Verweisquelle konnte nicht gefunden werden.**.

The width of the opcode call expression is the number of channels provided in the **return** statements in the opcode's code block.

For calls to core opcodes (see Subclause 5.5 SAOL core opcode definitions and semantics), in the absence of normative language specifying otherwise for a particular opcode, it is a syntax error if any of the following statements apply:

- there are fewer actual parameters in the opcode call than required formal parameters
- there are more actual parameters in the opcode call than required and optional formal parameters, and the opcode definition does not include a varargs "..." Subclause
- a particular actual parameter expression is of faster rate than the corresponding formal parameter, or than the varargs formal parameter if that is the correspondence
- a particular actual parameter expression is not single-valued, or is not table-valued when the corresponding formal parameter specifies a table.

The context of the opcode call is restricted more than other expressions. When occurring within a block subsidiary to a guarding statement (**if**, **else**, or **while**), opcode calls shall not have a rate slower than the rate of the guarding expression (see Subclauses 5.4.6.6.4 If, 5.4.6.6.5 Else, 5.4.6.6.6 While). A call to an opcode with a particular name shall not occur within the code block of definition of that opcode, nor

within the code blocks of any of the opcodes called by that opcode, or any of the opcodes called by them, etc. That is, recursive and mutually-recursive opcodes are prohibited.

To calculate the value of an opcode call expression referencing a user-defined opcode at a particular rate, the values of the actual parameter expression shall be calculated in the order they appear in the expression list. The values of the formal parameters within the opcode scope shall be set to the values of the corresponding actual parameter expressions. If this is the first opcode call expression referencing this opcode scope, opcode storage space shall be created to store local signal variables and wavetables, the local signal variables set to 0, and the local wavetables created as discussed in Subclause 5.4.6.5.2

Wavetable declaration. Any global variables imported by the opcode at that rate shall be copied into the opcode storage space. The statement block of the opcode shall be executed at the current rate. The value of the opcode call expression is the value of the first **return** statement encountered when executing the opcode. The value of the opcode call expression may be array-valued (if the expression in the **return** statement is). After the end of opcode execution, any global variables exported by the opcode shall be copied into the global storage space.

NOTE

If an opcode changes and exports the value of a global variable which is imported by the calling instrument or opcode, the change in the global variable is not reflected in the caller until the next orchestra pass.

If a particular actual parameter expression in an opcode call expression is an identifier or an array-reference expression, then that parameter is a reference parameter in that call to that opcode. When the opcode statement block is executed, the final value of the formal parameter associated with that actual parameter shall be copied into the variable value denoted by the identifier or array-reference. This modification shall happen immediately after (but not until) the termination of the statement block, before any other calculation is done. Both single-value and array-value expressions may be reference parameters, but if an array-valued expression is used, the associated formal parameter shall be an array of the same length.

To calculate the value of an opcode call expression referencing a core opcode at a particular rate, the values of the actual parameter expressions shall be calculated in the order they appear in the expression list. Then, the return value of the core opcode shall be calculated according to the rules for the particular opcode given in Subclause

The following words are reserved, and shall not be used as identifiers in a SAOL orchestra or score.

aopcode asig else exports extend global if imports inchannels inputmod instr iopcode ivar kopcode krate ksig map oparray opcode outbus outchannels output return route send sequence sbsynth spatialize srate table tablemap template turnoff while with xsig

Also, variable names starting with **_sym_** are reserved for implementation-specific use (for example, bitstream detokenisation – see Annex B).

5.5 SAOL core opcode definitions and semantics.

NOTE

The variables declared within the scope of a user-defined opcode are static-valued; that is, they preserve their values from call to call. The values of variables within the scope of a user-defined opcode are set to 0 before the opcode is called the first time. Each syntactically distinct call to an opcode creates one and only one opcode scope (see example in next Subclause).

5.4.6.7.7 Oparray call

<expr> -> **<ident> [** <expr> **] (** <expr list> **)**

An oparray call expression allows the dynamic selection of an opcode state from a set of several, and the calculation of encapsulated functionality with respect to that opcode state.

The identifier is termed the opcode name, the expression in brackets is termed the index expression, and the expressions in the parameter list are termed the actual parameters. It is illegal to use an identifier which is not the name of a core opcode and is also not the name of a user-defined opcode declared elsewhere in the orchestra. It is also illegal to use an identifier for which oparray storage is not allocated in the local scope as described in Subclause 5.4.6.5.5 Opcode array declaration. For user-defined opcodes, the number of actual parameters shall be the same as the number of formal parameters in the opcode definition. For core, the number of actual parameters required varies from opcode to opcode; see Subclause

The following words are reserved, and shall not be used as identifiers in a SAOL orchestra or score.

aopcode asig else exports extend global if imports inchannels inputmod instr iopcode ivar kopcode krate ksig map oparray opcode outbus outchannels output return route send sequence sbsynth spatialize srate table tablemap template turnoff while with xsig

Also, variable names starting with **_sym_** are reserved for implementation-specific use (for example, bitstream detokenisation – see Annex B).

5.5 SAOL core opcode definitions and semantics.

The index expression shall be a single-valued expression.

The rate of the oparray call expression is the rate of the opcode referenced, as determined by the rules in Subclause 5.4.7 Opcode definition. The rate of the index expression shall not be faster than the rate of the opcode referenced.

The width of the oparray call expression is the number of channels returned by **return** statements within the opcode code block.

The context of the oparray call expression is restricted in the same way as described for the opcode call expression in Subclause 5.4.6.7.6 Opcode call.

The value of the oparray call expression is determined in the same way as described for the opcode call in Subclause 5.4.6.7.6 Opcode call, with the following exceptions and additions:

Before the values of the actual parameter expressions are calculated, the value of the index expression is calculated. It is a run-time error if the value of the index expression is not in the range [0..n-1], where n is the allocation size in the oparray definition for this oparray. If the index expression is not an integer, it is rounded to the nearest integer. The scope storage associated with the opcode name and the value of the index expression is selected from the set of oparray scopes in the local scope. The evaluation of the statement block in the referenced opcode is with regard to the selected scope. Within each oparray scope, local variables retain their values from call to call.

EXAMPLES

Some examples are provided to clarify the distinction between opcode calls and oparray calls.

The following user defined opcode

```
kopcode inc() {
  ksig ct;

  ct = ct + 1;
  return(ct);
  }
```

counts the number of times it is called.

1. After the first execution of the following code fragment

```
a = inc();
b = inc();
```

the value of **a** is 1, and the value of **b** is 1, since each call to **inc()** refers to a different scope.

2. After the first execution of the following code fragment

```
i = 0; while (i < 2) { a = inc(); i = i + 1; }
```

the value of **a** is 2, since there is only one scope for **inc().**

3. After the first execution of the following code fragment

```
oparray inc[2];

a = inc[0]();
b = inc[0]();
```

the value of **a** is 1, and the value of **b** is 2, since each call to **inc()** refers to the same scope (since the value of the indexing expression is the same in both calls).

4. After the first execution of the following code fragment

```
oparray inc[2];

i = 0; while (i < 2) { a = inc[i](); i = i + 1; }
```

the value of **a** is 1, since each iteration refers to a different scope in the call to **inc()** (since the value of the indexing expression is 0 on the first iteration, and 1 on the second).

NOTE

Opcode calls and oparray calls referencing the same opcode may be used in the same scope. In this case, the scopes referenced by each of the opcode calls are different from any of the scopes defined in the oparray definition.

5.4.6.7.8 Combination of vector and scalar elements in mathematical expressions

The subsequent Subclauses (Subclauses 5.4.6.7.9 Switch through 5.4.6.7.13 Parenthesis) describe mathematical expressions in SAOL. For each, the width of the expression is the maximum width of any of its subexpressions. For each expression type, each subexpression within an expression shall have the same width, or else width of 1. If subexpressions with width 1 and width different than 1 are combined in an expression, before the expression is computed, the subexpression(s) with width 1 shall be promoted to have the same width as the expression. That is, a width 1 expression with value **x** which is a subexpression of a width *n* expression shall be promoted to a width *n* expression where the value of each element is **x**.

For each expression type below, the semantics will be given for array-valued expressions. In each case, the semantics for the single-valued expression are the same as for an array-valued expression with width 1, except for the special cases of switch, logical AND, and logical OR, which will be described separately in those Subclauses.

5.4.6.7.9 Switch

<expr> -> <expr> **?** <expr> **:** <expr>

The switch expression combines values from two subexpressions based on the value of a third.

The rate of the switch expression is the rate of the fastest of the three subexpressions.

The value of the switch expression is calculated as follows: the three subexpressions are evaluated. Then, for each value of the first subexpression, if this value is non-zero, the corresponding value of the switch expression is the corresponding value of the second subexpression. If this value is zero, the corresponding value of the switch expression is the corresponding value of the third subexpression.

In the special case where all subexpressions have width 1, then the switch expression "short-circuits": the first subexpression is evaluated, and if its value is non-zero, then the second subexpression is evaluated, and its value is the value of the switch expression. If the value of the first subexpression is zero, then the third subexpression is evaluated, and its value is the value of the switch expression. If the width of the switch expression is 1, then in no case are both the second and third subexpressions evaluated.

5.4.6.7.10 Not

<expr> -> **!** <expr>

The not expression performs logical negation on a subexpression.

The rate of the not expression is the rate of the subexpression.

The value of the not expression is calculated as follows: the subexpression is evaluated. For each nonzero value in the subexpression, the corresponding value of the not expression is zero; for each zero value in the subexpression, the corresponding value of the not expression is 1.

5.4.6.7.11 Negation

<expr> -> **-** <expr>

The negation expression performs arithmetic negation on a subexpression.

The rate of the negation expression is the rate of the subexpression.

The value of the negation expression shall be calculated as follows: the subexpression is evaluated. For each value in the subexpression, the corresponding value of the negation expression is the arithmetic negative of the value.

5.4.6.7.12 Binary operators

<expr> -> <expr> <binop> <expr>

There are 12 binary operators. Each of them calculates a different function on binary subexpressions.

The value of the expression shall be calculated as follows. The two subexpressions shall be evaluated, and for each pair of values of the subexpressions, and the corresponding value of the binary expression shall be calculated according to the following table, where x_1 and x_2 are the values of the first and second subexpressions:

Operator	Value of expression
+	$x_1 + x_2$
-	$x_1 - x_2$
*	$x_1 x_2$
/	x_1 / x_2
==	if $x_1 = x_2$, then 1, otherwise 0
>	if $x_1 > x_2$, then 1, otherwise 0
<	if $x_1 < x_2$, then 1, otherwise 0
<=	if $x_1 \leq x_2$, then 1, otherwise 0
>=	if $x_1 \geq x_2$, then 1, otherwise 0
!=	if $x_1 \neq x_2$, then 1, otherwise 0

In each of these cases, if the particular operation would result in a NaN or Inf result (for example, division by 0), a run-time error shall result.

For the "logical and" operator **&&** in the special case where both subexpressions have width 1, the expression is calculated in a "short-circuit" fashion. The first subexpression shall be evaluated. If its value is 0, then the value of the expression is 0; if its value is nonzero, then the second subexpression shall be evaluated, and if its value is 0, then the value of the expression is 0, otherwise the value of the expression is 1.

For the "logical or" operator || in the special case where both subexpressions have width 1, the expression is calculated in a "short-circuit" fashion. The first subexpression shall be evaluated. If its value is nonzero, then the value of the expression is 1; if its value is 0, then the second subexpression shall be evaluated, and if its value is nonzero, then the value of the expression is 1, otherwise the value of the expression is 0.

5.4.6.7.13 Parenthesis

<expr> -> **(** <expr> **)**

The parenthesis operator performs no new calculation, but allows the specification of arithmetic grouping.

The rate of the parenthesis expression is the rate of the subexpression.

The width of the parenthesis expression is the width of the subexpression.

The value of the parenthesis expression is the value of the subexpression.

5.4.6.7.14 Order of operations

Expressions bind in the order prescribed in the following table. That is, operations listed higher in the table are performed before operations lower in the table whenever the ordering is syntactically ambiguous. Operations listed on the same row associate left-to-right. That is, the leftmost expression is performed first.

Operator	Function
!	not
-	unary negation
*, /	multiply, divide
+, -	add, subtract
<, >, <=, >=	relational
==, !=	equality
&&	logical and
||	logical or
?:	switch

5.4.6.8 Standard names

5.4.6.8.1 Definition

Not all identifiers to be referenced in an instrument or opcode are required to be declared as variables. Several identifiers, listed in this Subclause, are termed standard names, shall not be used as variables, and have fixed semantics which shall be implemented in a compliant SAOL decoder. Standard names may otherwise be used as variables, embedded in expressions, etc. in any SAOL instrument or opcode. However, the semantics of using a standard name as an lvalue are undefined.

The implicit definition of each standard name, showing the rate semantics and width of that standard name, is listed, and the semantics of the value of the standard name specified in the subsequent Subclauses.

5.4.6.8.2 k_rate

```
ivar k_rate
```

The standard name **k_rate** shall contain the control rate of the orchestra, in Hz.

5.4.6.8.3 s_rate

```
ivar s_rate
```

The standard name **s_rate** shall contain the sampling rate of the orchestra, in Hz.

5.4.6.8.4 inchan

```
ivar inchan
```

The standard name **inchan**, in each scope, shall contain the number of channels of input being provided to the instrument instantiation with which that scope is associated. “Associated” shall be taken to mean, for instrument code, the instrument instantiation for which the scope memory was created; for opcode code, the instrument instantiation which called the opcode, or called the opcode’s caller, etc.

Different instances of the same instrument may have different numbers of input channels if, for example, they are the targets of different send statements.

5.4.6.8.5 outchan

```
ivar outchan
```

The standard name **outchan** shall contain the number of channels of output being produced by the orchestra (not by the instrument instance).

5.4.6.8.6 time

```
ivar time
```

The standard name **time**, in each scope, shall contain the time at which the instrument instantiation associated with that scope was created.

NOTE

If the “event time” of an instrument (for example, a score event more precisely timed than one control period) and the actual instantiation time differ, the name time shall contain the latter time, not the former.

5.4.6.8.7 dur

```
ivar dur
```

The standard name **dur**, in each scope, shall contain the duration of the instrument instantiation as originally created, or –1 if the duration was not known at instantiation.

5.4.6.8.8 tempo

```
ksig tempo
```

The standard name **tempo** shall contain the value of the current global tempo, in beats per minute. The default value is 60 beats per minute.

5.4.6.8.9 MIDIctrl

```
ksig MIDIctrl[128]
```

The MIDIctrl **standard variable shall contain, for each scope, the current values of the MIDI controllers on the channel corresponding to the channel to which the instrument instantiation associated with that scope is assigned. See Subclause 5.9 Sample Bank syntax and semantics**

5.9.1 Introduction

This Subclause describes the operation of the Sample Bank synthesis method for both Profile 2 and Profile 4. In Profile 2, only Sample Bank and MIDI class types shall appear in the bitstream, and this Subclause describes the normative process of generating sound from a Sample Bank bitstream data element and a sequence of MIDI instructions. In Profile 4, Sample Banks are used in the context of a SAOL instrument as described in Subclause 5.4.6.6.8 Output, and this Subclause describes the normative process of generating sound and placing it on three busses, depending on the Sample Bank bitstream data element and the particular call to **sbsynth**.

5.9.2 Elements of bitstream

5.9.2.1 RIFF Structure

5.9.2.1.1 General RIFF File Structure

The RIFF (Resource Interchange File Format) is a tagged file structure developed for multimedia resource files, and is described in some detail in the Microsoft Windows 3.1 SDK Multimedia Programmer's Reference. The Tagged-file structure is useful because it helps prevent compatibility problems which can occur as the file definition changes over time. Because each piece of data in the file is identified by a standard header, an application that does not recognise a given data element can skip over the unknown information. The relevant information is provided in the bitstream description, Subclause 0.

A RIFF file is constructed from a basic building block called a "chunk." In 'C' syntax, a chunk is defined:

```
typedef DWORD FOURCC;      //  Four-character code
typedef struct {
  FOURCC ckID;             //  Chunk ID identifies type of data in the chunk.
  DWORD ckSize;            //  Size of chunk data in bytes, excluding pad byte.
  BYTE ckDATA[ckSize];     //  Actual data + a pad byte if req'd to word align.
};
```

Two types of chunks, the "RIFF" and "LIST" chunks, may contain nested chunks called subchunks as their data.

Within a given level of the hierarchy, the ordering of the chunks is arbitrary. Any chunk with an unknown chunk ID should be ignored.

5.9.2.1.2 The Sample Bank Chunks and Subchunks

A Structured Audio Sample Bank bitstream element comprises three chunks: an INFO-list chunk containing a number of required and optional subchunks describing the bitstream element, its history, and its intended use, an sdta-list chunk comprising a single subchunk containing any referenced digital audio samples, and a pdta-list chunk containing nine subchunks which define the articulation of the digital audio data.

5.9.2.1.3 Redundancy and Error Handling in the RIFF structure

The RIFF bitstream element structure contains redundant information regarding the length of the bitstream element and the length of the chunks and subchunks. This fact enables any reader of a SASBF bitstream element to determine if the bitstream element has been damaged by loss of data.

If any such loss is detected, the SASBF bitstream element is termed “structurally unsound” and in general should be rejected.

5.9.2.2 The INFO-list Chunk

The INFO-list chunk in a SASBF bitstream element contains three mandatory and a variety of optional subchunks as defined below. The INFO-list chunk gives basic information about the SASBF bank contained in the bitstream element.

5.9.2.2.1 The ifil Subchunk

The ifil subchunk is a mandatory subchunk identifying the SASBF specification version level to which the bitstream element complies. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the SASBF specification version. The WORD wMinor contains the value to the right of the decimal point. For example, version 2.11 would be implied if wMajor=2 and wMinor=11.

These values can be used by applications which read SASBF bitstream elements to determine if the format of the bitstream element is usable by the program. Within a fixed wMajor, the only changes to the format will be the addition of Generator, Source and Transform enumerators, and additional info subchunks. These are all defined as being ignored if unknown to the program.

If the ifil subchunk is missing, or its size is not four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.2.2 The isng Subchunk

The isng subchunk is a mandatory subchunk identifying the wavetable sound engine for which the bitstream element was optimised. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even.

The ASCII should be treated as case-sensitive. In other words “engine” is not the same as “ENGINE.”

The isng string may be optionally used by chip drivers to vary their synthesis algorithms to emulate the target sound engine.

If the isng subchunk is missing not terminated in a zero valued byte, or its contents are an unknown sound engine, the field should be ignored.

5.9.2.2.3 The INAM Subchunk

The INAM subchunk is a mandatory subchunk providing the name of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical inam subchunk would be the fourteen bytes representing “General MIDI” as twelve ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “General MIDI” is not the same as “GENERAL MIDI.”

If the inam subchunk is missing, or not terminated in a zero valued byte, the field should be ignored and the user supplied with an appropriate error message if the name is queried. If the bitstream element is re-written, a valid name should be placed in the INAM field.

5.9.2.2.4 The irom Subchunk

The irom subchunk is an optional subchunk identifying a particular wavetable sound data ROM to which any ROM samples refer. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical irom field would be the six bytes representing “1MGM” as four ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “1mgm” is not the same as “1MGM.”

The irom string is used by drivers to verify that the ROM data referenced by the bitstream element is available to the sound engine.

If the irom subchunk is missing, not terminated in a zero valued byte, or its contents are an unknown ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. A bitstream element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.5 The iver Subchunk

The iver subchunk is an optional subchunk identifying the particular wavetable sound data ROM revision to which any ROM samples refer. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the ROM version. The WORD wMinor contains the value to the right of the decimal point. For example, version 1.36 would be implied if wMajor=1 and wMinor=36.

The iver subchunk is used by drivers to verify that the ROM data referenced by the bitstream element is located in the exact locations specified by the sound headers.

If the iver subchunk is missing, not four bytes in length, or its contents indicate an unknown or incorrect ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. Note that for ROM samples to function correctly, both iver and irom must be present and valid. A bitstream

element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.6 The ICRD Subchunk

The ICRD subchunk is an optional subchunk identifying the creation date of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICRD field would be the twelve bytes representing “May 1, 1995” as eleven ASCII characters followed by a zero byte.

Conventionally, the format of the string is “Month Day, Year” where Month is initially capitalised and is the conventional full English spelling of the month, Day is the date in decimal followed by a comma, and Year is the full decimal year. Thus the field should conventionally never be longer than 32 bytes.

The ICRD string is provided for library management purposes.

If the ICRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.7 The IENG Subchunk

The IENG subchunk is an optional subchunk identifying the names of any sound designers or engineers responsible for the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IENG field would be the twelve bytes representing “Tim Swartz” as ten ASCII characters followed by two zero bytes.

The IENG string is provided for library management purposes.

If the IENG subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.8 The IPRD Subchunk

The IPRD subchunk is an optional subchunk identifying any specific product for which the SASBF bank is intended. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IPRD field would be the twelve bytes representing “MPEG SASBF” as ten ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “mpeg sasbf” is not the same as “MPEG SASBF.”

The IPRD string is provided for library management purposes.

If the IPRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.9 The ICOP Subchunk

The ICOP subchunk is an optional subchunk containing any copyright assertion string associated with the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICOP field would be the 38 bytes representing "Copyright (c) 1998 Content Developer" as 36 ASCII characters followed by two zero bytes.

The ICOP string is provided for intellectual property protection and management purposes.

If the ICOP subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.10 The ICMT Subchunk

The ICMT subchunk is an optional subchunk containing any comments associated with the SASBF bank. It contains an ASCII string of 65,536 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICMT field would be the 40 bytes representing "This space unintentionally left blank." as 38 ASCII characters followed by two zero bytes.

The ICMT string is provided for including comments.

If the ICMT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.11 The ISFT Subchunk

The ISFT subchunk is an optional subchunk identifying the SASBF tools used to create and most recently modify the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ISFT field would be the twenty-six bytes representing "Editor 2.00a:Editor 2.00a" as twenty-five ASCII characters followed by a zero byte.

The ASCII should be treated as case-sensitive. In other words "Editor" is not the same as "EDITOR."

Conventionally, the tool name and revision control number are included first for the creating tool and then for the most recent modifying tool. The two strings are separated by a colon. The string should be produced by the creating program with a null modifying tool field (e.g. "Editor 2.00a:), and each time a tool modifies the bank, it should replace the modifying tool field with its own name and revision control number.

The ISFT string is provided primarily for error tracing purposes.

If the ISFT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.3 The sdta-list Chunk

The sdta-list chunk in a SASBF bitstream element contains a single optional smpl subchunk which contains all the sound data associated with the SASBF bank. The smpl subchunk is of arbitrary length, and contains an even number of bytes.

5.9.2.3.1 Sample Data Format in the smpl Subchunk

The smpl subchunk, contains one or more samples of digital audio information in the form of linearly coded sixteen bit, signed, little endian (least significant byte first) words. Each sample is followed by a minimum of forty-six zero valued sample data points. These zero valued data points are necessary to guarantee that any reasonable upward pitch shift using any reasonable interpolator can loop on zero data at the end of the sound.

5.9.2.3.2 Sample Data Looping Rules

Within each sample, one or more loop point pairs may exist. The locations of these points are defined within the pdta-list chunk, but the sample data points themselves must comply with certain practices in order for the loop to be compatible across multiple platforms.

The loops are defined by “equivalent points” in the sample. This means that there are two sample data points which are logically equivalent, and a loop occurs when these points are spliced atop one another. In concept, the loop end point is never actually played during looping; instead the loop start point follows the point just prior to the loop end point. Because of the bandlimited nature of digital audio sampling, an artefact free loop will exhibit virtually identical data surrounding the equivalent points.

In actuality, because of the various interpolation algorithms used by wavetable synthesisers, the data surrounding both the loop start and end points may affect the sound of the loop. Hence both the loop start and end points must be surrounded by continuous audio data. For example, even if the sound is programmed to continue to loop throughout the decay, sample data points must be provided beyond the loop end point. This data will typically be identical to the data at the start of the loop. A minimum of eight valid data points are required to be present before the loop start and after the loop end.

The eight data points (four on each side) surrounding the two equivalent loop points should also be forced to be identical. By forcing the data to be identical, all interpolation algorithms are guaranteed to properly reproduce an artefact-free loop.

5.9.2.4 The pdta-list Chunk

5.9.2.4.1 The PHDR Subchunk

The PHDR subchunk is a required subchunk listing all presets within the SASBF bitstream element. It shall be a multiple of thirty-eight bytes in length, and shall contain a minimum of two records, one record for each preset and one for a terminal record according to the structure:

```
struct sfPresetHeader
{
       CHAR achPresetName[20];
       WORD wPreset;
       WORD wBank;
       WORD wPresetBagNdx;
       DWORD dwLibrary;
       DWORD dwGenre;
       DWORD dwMorphology;
};
```

The ASCII character field achPresetName shall contain the name of the preset expressed in ASCII, with unused terminal characters filled with zero valued bytes. Preset names are case-sensitive. A unique name shall always be assigned to each preset in the SASBF bank.

The WORD wPreset shall contain the MIDI Preset Number and the WORD wBank the MIDI Bank Number which apply to this preset. Note that the presets are not ordered within the SASBF bank. Presets

shall have a unique set of wPreset and wBank numbers. The special case of a General MIDI percussion bank is handled conventionally by a wBank value of 128. If the value in either field is not a valid MIDI value of 0 through 127, or 128 for wBank, the preset cannot be played but shall be maintained in memory for future renumbering.

The WORD wPresetBagNdx is an index to the preset's zone list in the PBAG subchunk. Because the preset zone list is in the same order as the preset header list, the preset bag indices shall be monotonically increasing with increasing preset headers. The size of the PBAG subchunk in bytes shall be equal to four times the terminal preset's wPresetBagNdx plus four. If the preset bag indices are non-monotonic or if the terminal preset's wPresetBagNdx does not match the PBAG subchunk size, the SASBF chunk is structurally defective and shall be rejected at load time. All presets except the terminal preset shall have at least one zone.

The DWORDs dwLibrary, dwGenre and dwMorphology are reserved for future implementation in a preset library management function and should be preserved as read, and created as zero.

The terminal sfPresetHeader record should never be accessed, and exists only to provide a terminal wPresetBagNdx with which to determine the number of zones in the last preset. All other values shall be conventionally zero, with the exception of achPresetName, which may be optionally be "EOP" indicating end of presets.

If the PHDR subchunk is missing, contains fewer than two records, or its size is not a multiple of 38 bytes, the SASBF bitstream element shall be rejected as structurally unsound.

5.9.2.4.2 The PBAG Subchunk

The PBAG subchunk is a required subchunk listing all preset zones within the SASBF bitstream element. It shall be a multiple of four bytes in length, and shall contain one record for each preset zone plus one record for a terminal zone according to the structure:

```
struct sfPresetBag
{
       WORD wGenNdx;
       WORD wModNdx;
};
```

The first zone in a given preset shall be located at that preset's wPresetBagNdx. The number of zones in the preset shall be determined by the difference between the next preset's wPresetBagNdx and the current wPresetBagNdx.

The WORD wGenNdx shall be an index to the preset's zone list of generators in the PGEN subchunk, and the wModNdx shall be an index to its list of modulators in the PMOD subchunk. Because both the generator and modulator lists are in the same order as the preset header and zone lists, these indices will be monotonically increasing with increasing preset zones. The size of the PMOD subchunk in bytes shall be equal to ten times the terminal preset's wModNdx plus ten and the size of the PGEN subchunk in bytes shall be equal to four times the terminal preset's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective PGEN or PMOD subchunks, the bitstream element is structurally defective and shall be rejected at load time.

If a preset has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not an Instrument generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks an Instrument generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the PBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.3 The PMOD Subchunk

The PMOD subchunk is a required subchunk listing all preset zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The preset zone's wModNdx points to the first modulator for that preset zone, and the number of modulators present for a preset zone is determined by the difference between the next higher preset zone's wModNdx and the current preset's wModNdx. A difference of zero indicates there are no modulators in this preset zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator with the same zone, the first modulator will be ignored.

Modulators in the PMOD subchunk act as additively relative modulators with respect to those in the IMOD subchunk. In other words, a PMOD modulator can increase or decrease the amount of an IMOD modulator.

In SASBF, no modulators have yet been defined, and the PMOD subchunk will always consist of ten zero valued bytes.

If the PMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.4 The PGEN Subchunk

The PGEN chunk is a required chunk containing a list of preset zone generators for each preset zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each preset zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined:

```
typedef struct
{
       BYTE byLo;
       BYTE byHi;
} rangesType;

typedef union
{
       rangesType ranges;
       SHORT shAmount;
       WORD wAmount;
} genAmountType;
```

The sfGenOper is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. This value indicates the type of generator being indicated. Note that this enumeration is two bytes in length.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers or MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The preset zone's wGenNdx points to the first generator for that preset zone. Unless the zone is a global zone, the last generator in the list is an "Instrument" generator, whose value is a pointer to the instrument associated with that zone. If a "key range" generator exists for the preset zone, it is always the first generator in the list for that preset zone. If a "velocity range" generator exists for the preset zone, it will only be preceded by a key range generator. If any generators follow an Instrument generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator with the same zone, the first generator will be ignored.

Generators in the PGEN subchunk are applied relative to generators in the IGEN subchunk in an additive manner. In other words, PGEN generators increase or decrease the value of an IGEN generator.

If the PGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in an instrument generator, the zone should be ignored. If the instrument generator value is equal to or greater than the terminal instrument, the bitstream element should be rejected as structurally unsound.

5.9.2.4.5 The INST Subchunk

The inst subchunk is a required subchunk listing all instruments within the SASBF bitstream element. It is always a multiple of twenty-two bytes in length, and contains a minimum of two records, one record for each instrument and one for a terminal record according to the structure:

```
struct sfInst
{
       CHAR achInstName[20];
       WORD wInstBagNdx;
};
```

The ASCII character field achInstName contains the name of the instrument expressed in ASCII, with unused terminal characters filled with zero valued bytes. Instrument names are case-sensitive. A unique name should always be assigned to each instrument in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of instruments with identical names, the instruments should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The WORD wInstBagNdx is an index to the instrument's zone list in the IBAG subchunk. Because the instrument zone list is in the same order as the instrument list, the instrument bag indices will be monotonically increasing with increasing instruments. The size of the IBAG subchunk in bytes will be four greater than four times the terminal (EOI) instrument's wInstBagNdx. If the instrument bag indices are non-monotonic or if the terminal instrument's wInstBagNdx does not match the IBAG subchunk size, the bitstream element is structurally defective and should be rejected at load time. All instruments except the terminal instrument must have at least one zone; any preset with no zones should be ignored.

The terminal sfInst record should never be accessed, and exists only to provide a terminal wInstBagNdx with which to determine the number of zones in the last instrument. All other values are conventionally zero, with the exception of achInstName, which should be "EOI" indicating end of instruments.

If the INST subchunk is missing, contains fewer than two records, or its size is not a multiple of 22 bytes, the bitstream element should be rejected as structurally unsound. All instruments present in the inst subchunk are typically referenced by a preset zone. However, a bitstream element containing any "orphaned" instruments need not be rejected. SASBF applications can optionally ignore or filter out these orphaned instruments based on user preference.

5.9.2.4.6 The IBAG Subchunk

The IBAG subchunk is a required subchunk listing all instrument zones within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one record for each instrument zone plus one record for a terminal zone according to the structure:

```
struct sfInstBag
{
       WORD wInstGenNdx;
       WORD wInstModNdx;
```

```
};
```

The first zone in a given instrument is located at that instrument's wInstBagNdx. The number of zones in the instrument is determined by the difference between the next instrument's wInstBagNdx and the current wInstBagNdx.

The WORD wInstGenNdx is an index to the instrument zone's list of generators in the IGEN subchunk, and the wInstModNdx is an index to its list of modulators in the IMOD subchunk. Because both the generator and modulator lists are in the same order as the instrument and zone lists, these indices will be monotonically increasing with increasing zones. The size of the IMOD subchunk in bytes will be equal to ten times the terminal instrument's wModNdx plus ten and the size of the IGEN subchunk in bytes will be equal to four times the terminal instrument's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective IGEN or IMOD subchunks, the bitstream element is structurally defective and should be rejected at load time.

If an instrument has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not a sampleID generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks a sampleID generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the IBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.7 The IMOD Subchunk

The IMOD subchunk is a required subchunk listing all instrument zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The zone's wInstModNdx points to the first modulator for that zone, and the number of modulators present for a zone is determined by the difference between the next higher zone's wInstModNdx and the current zone's wModNdx. A difference of zero indicates there are no modulators in this zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator within the same zone, the first modulator will be ignored.

Modulators in the IMOD subchunk are absolute. This means that an IMOD modulator replaces, rather than adds to, a default modulator.

In SASBF, no modulators have yet been defined, and the IMOD subchunk will always consist of ten zero valued bytes.

If the IMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.8 The IGEN Subchunk

The IGEN chunk is a required chunk containing a list of zone generators for each instrument zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfInstGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined as in the PGEN zone above.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers of MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The zone's wInstGenNdx points to the first generator for that zone. Unless the zone is a global zone, the last generator in the list is a "sampleID" generator, whose value is a pointer to the sample associated with that zone. If a "key range" generator exists for the zone, it is always the first generator in the list for that zone. If a "velocity range" generator exists for the zone, it will only be preceded by a key range generator. If any generators follow a sampleID generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator within the same zone, the first generator will be ignored.

Generators in the IGEN subchunk are absolute in nature. This means that an IGEN generator replaces, rather than adds to, the default value for the generator.

If the IGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in a sampleID generator, the zone should be ignored. If the sampleID generator value is equal to or greater than the terminal sampleID, the bitstream element should be rejected as structurally unsound.

5.9.2.4.9 The SHDR Subchunk

The SHDR chunk is a required subchunk listing all samples within the smpl subchunk and any referenced ROM samples. It is always a multiple of forty-six bytes in length, and contains one record for each sample plus a terminal record according to the structure:

```
struct sfSample
{
       CHAR achSampleName[20];
       DWORD dwStart;
       DWORD dwEnd;
       DWORD dwStartloop;
       DWORD dwEndloop;
       DWORD dwSampleRate;
       BYTE byOriginalPitch;
       CHAR chPitchCorrection;
       WORD wSampleLink;
       SFSampleLink sfSampleType;
};
```

The ASCII character field achSampleName contains the name of the sample expressed in ASCII, with unused terminal characters filled with zero valued bytes. Sample names are case-sensitive. A unique name should always be assigned to each sample in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of samples with identical names, the samples should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The DWORD dwStart contains the index, in sample data points, from the beginning of the sample data field to the first data point of this sample.

The DWORD dwEnd contains the index, in sample data points, from the beginning of the sample data field to the first of the set of 46 zero valued data points following this sample.

The DWORD dwStartloop contains the index, in sample data points, from the beginning of the sample data field to the first data point in the loop of this sample.

The DWORD dwEndloop contains the index, in sample data points, from the beginning of the sample data field to the first data point following the loop of this sample. Note that this is the data point “equivalent to” the first loop data point, and that to produce portable artefact free loops, the eight proximal data points surrounding both the Startloop and Endloop points should be identical.

The values of dwStart, dwEnd, dwStartloop, and dwEndloop must all be within the range of the sample data field included in the SASBF bank or referenced in the sound ROM. Also, to allow a variety of hardware platforms to be able to reproduce the data, the samples have a minimum length of 48 data points, a

minimum loop size of 32 data points and a minimum of 8 valid points prior to dwStartloop and after dwEndloop. Thus dwStart must be less than dwStartloop-7, dwStartloop must be less than dwEndloop-31, and dwEndloop must be less than dwEnd-7. If these constraints are not met, the sound may optionally not be played if the hardware cannot support artefact-free playback for the parameters given.

The DWORD dwSampleRate contains the sample rate, in hertz, at which this sample was acquired or to which it was most recently converted. Values of greater than 50000 or less than 400 may not be reproducible by some hardware platforms and should be avoided. A value of zero is illegal. If an illegal or impractical value is encountered, the nearest practical value should be used.

The BYTE byOriginalPitch contains the MIDI key number of the recorded pitch of the sample. For example, a recording of an instrument playing middle C (261.62 Hz) should receive a value of 60. This value is used as the default “root key” for the sample, so that in the example, a MIDI key-on command for note number 60 would reproduce the sound at its original pitch. For unpitched sounds, a conventional value of 255 should be used. Values between 128 and 254 are illegal. Whenever an illegal value or a value of 255 is encountered, the value 60 should be used.

The CHAR chPitchCorrection contains a pitch correction in cents which should be applied to the sample on playback. The purpose of this field is to compensate for any pitch errors during the sample recording process. The correction value is that of the correction to be applied. For example, if the sound is 4 cents sharp, a correction bringing it 4 cents flat is required; thus the value should be -4.

The value in sfSampleType is an enumeration with eight defined values: monoSample = 1, rightSample = 2, leftSample = 4, linkedSample = 8, RomMonoSample = 32769, RomRightSample = 32770, RomLeftSample = 32772, and RomLinkedSample = 32776. It can be seen that this is encoded such that bit 15 of the 16 bit value is set if the sample is in ROM, and reset if it is included in the SASBF bank. The four LS bits of the word are then exclusively set indicating mono, left, right, or linked.

If the sound is flagged as a ROM sample and no valid “irom” subchunk is included; the bitstream element is structurally defective and should be rejected at load time.

If sfSampleType indicates a mono sample, then wSampleLink is undefined and its value should be conventionally zero, but will be ignored regardless of value. If sfSampleType indicates a left or right sample, then wSampleLink is the sample header index of the associated right or left stereo sample respectively. Both samples should be played entirely synchronously, with their pitch controlled by the right sample’s generators. All non-pitch generators should apply as normal; in particular the panning of the individual samples to left and right should be accomplished via the pan generator. Left-right pairs should always be found within the same instrument. Note also that no instrument should be designed in which it is possible to activate more than one instance of a particular stereo pair. The linked sample type is not currently fully defined in the SASBF specification, but will ultimately support a circularly linked list of samples using wSampleLink. Note that this enumeration is two bytes in length.

The terminal sample record is never referenced, and is conventionally entirely zero with the exception of achSampleName, which should be “EOS” indicating end of samples. All samples present in the smpl subchunk are typically referenced by an instrument, however a bitstream element containing any “orphaned” samples need not be rejected. SASBF applications can optionally ignore or filter out these orphaned samples according to user preference.

If the SHDR subchunk is missing, or its is size is not a multiple of 46 bytes the bitstream element should be rejected as structurally unsound.

5.9.3 Enumerators

5.9.3.1 Generator Enumerators

Subclause 7.1 defines the generator and generator kinds. Subclause 8.4 defines the generator operation model.

5.9.3.1.1 Kinds of Generator Enumerators

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

An Index Generator's amount is an index into another data structure. The only two Index Generators are Instrument and sampleID.

A Range Generator defines a range of note-on parameters outside of which the zone is undefined. Two Range Generators are currently defined, keyRange and velRange.

Substitution Generators are generators which substitute a value for a note-on parameter. Two Substitution Generators are currently defined, overridingKeyNumber and overridingVelocity.

Sample Generators are generators which directly affect a sample's properties. These generators are undefined at the preset level. The currently defined Sample Generators are the eight address offset generators, the sampleModes generator, the Overriding Root Key generator and the Exclusive Class generator.

Value Generators are generators whose value directly affects a signal processing parameter. Most generators are value generators.

5.9.3.1.2 Generator Enumerators Defined

The following is an exhaustive list of SASBF generators and their strict definitions:

0	startAddrsOffset	The offset, in sample data points, beyond the Start sample header parameter to the first sample data point to be played for this instrument. For example, if Start were 7 and startAddrsOffset were 2, the first sample data point played would be sample data point 9.
1	endAddrsOffset	The offset, in sample data points, beyond the End sample header parameter to the last sample data point to be played for this instrument. For example, if End were 17 and endAddrsOffset were -2, the last sample data point played would be sample data point 15.
2	startloopAddrsOffset	The offset, in sample data points, beyond the Startloop sample header parameter to the first sample data point to be repeated in the loop for this instrument. For example, if Startloop were 10 and startloopAddrsOffset were -1, the first repeated loop sample data point would be sample data point 9.
3	endloopAddrsOffset	The offset, in sample data points, beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. For example, if Endloop were 15 and endloopAddrsOffset were 2, sample data point 17 would be considered equivalent to the

		Startloop sample data point, and hence sample data point 16 would effectively precede Startloop during looping.
4	startAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Start sample header parameter and the first sample data point to be played in this instrument. This parameter is added to the startAddrsOffset parameter. For example, if Start were 5, startAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point played would be sample data point 65544.
5	modLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
6	vibLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Vibrato LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
7	modEnvToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence pitch. A positive value indicates an increase in pitch; a negative value indicates a decrease in pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will rise 1 semitone at the envelope peak.
8	initialFilterFc	This is the cutoff and resonant frequency of the lowpass filter in absolute cent units. The lowpass filter is defined as a second order resonant pole pair whose pole frequency in Hz is defined by the Initial Filter Cutoff parameter. When the cutoff frequency exceeds 20kHz and the Q (resonance) of the filter is zero, the filter does not affect the signal.
9	initialFilterQ	This is the height above DC gain in centibels which the filter resonance exhibits at the cutoff frequency. A value of zero or less indicates the filter is not resonant; the gain at the cutoff frequency (pole angle) may be less than zero when zero is specified. The filter gain at DC is also affected by this parameter such that the gain at DC is reduced by half the specified gain. For example, for a value of 100, the filter gain at DC would be 5 dB below unity gain, and the height of the resonant peak would be 10 dB above the DC gain, or 5 dB above unity gain. Note also that if initialFilterQ is set to zero or less and the cutoff frequency exceeds 20 kHz, then the filter response is flat and unity gain.

10	modLfoToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence filter cutoff frequency. A positive number indicates a positive LFO excursion increases cutoff frequency; a negative number indicates a positive excursion decreases cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1200 indicates that the cutoff frequency will first rise 1 octave, then fall one octave.
11	modEnvToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence filter cutoff frequency. A positive number indicates an increase in cutoff frequency; a negative number indicates a decrease in filter cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1000 indicates that the cutoff frequency will rise one octave at the envelope attack peak.
12	endAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the End sample header parameter and the last sample data point to be played in this instrument. This parameter is added to the endAddrsOffset parameter. For example, if End were 65536, startAddrsOffset were -3 and startAddrsCoarseOffset were -1, the last sample data point played would be sample data point 32765.
13	modLfoToVolume	This is the degree, in centibels, to which a full scale excursion of the Modulation LFO will influence volume. A positive number indicates a positive LFO excursion increases volume; a negative number indicates a positive excursion decreases volume. Volume is always modified logarithmically, that is the deviation is in decibels rather than in linear amplitude. For example, a value of 100 indicates that the volume will first rise ten dB, then fall ten dB.
14	unused1	Unused, reserved. Should be ignored if encountered.
15	chorusEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the chorus effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the chorus effects processor.
16	reverbEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the reverb effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the reverb effects processor.

17	pan	This is the degree, in 0.1% units, to which the “dry” audio output of the note is positioned to the left or right output. A value of -50% or less indicates the signal is sent entirely to the left output and not sent to the right output; a value of +50% or more indicates the signal is sent entirely to the right and not sent to the left. A value of zero sends the signal equally to left and right. For example, a value of -250 indicates that the signal is sent at 75% of full level to the left output and 25% of full level to the right output.
18	unused2	Unused, reserved. Should be ignored if encountered.
19	unused3	Unused, reserved. Should be ignored if encountered.
20	unused4	Unused, reserved. Should be ignored if encountered.
21	delayModLFO	This is the delay time, in absolute timecents, from key on until the Modulation LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second and a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
22	freqModLFO	This is the frequency, in absolute cents, of the Modulation LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 MHz would be 1200log2(.01/8.176) = -11610.
23	delayVibLFO	This is the delay time, in absolute timecents, from key on until the Vibrato LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
24	freqVibLFO	This is the frequency, in absolute cents, of the Vibrato LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 mHz would be 1200log2(.01/8.176) = -11610.
25	delayModEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Modulation envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
26	attackModEnv	This is the time, in absolute timecents, from the end of the Modulation Envelope Delay Time until the point at which the Modulation Envelope value reaches its peak. Note that the attack is

		"convex"; the curve is nominally such that when applied to a decibel or semitone parameter, the result is linear in amplitude or Hz respectively. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
27	holdModEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
28	decayModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during decay phase. For the Modulation Envelope, the decay phase linearly ramps toward the sustain level. If the sustain level were zero, the Modulation Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
29	sustainModEnv	This is the decrease in level, expressed in 0.1% units, to which the Modulation Envelope value ramps during the decay phase. For the Modulation Envelope, the sustain level is properly expressed in percent of full scale. Because the volume envelope sustain level is expressed as an attenuation from full scale, the sustain level is analogously expressed as a decrease from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; values above 1000 are to be interpreted as 1000. For example, a sustain level which corresponds to an absolute value 40% of peak would be 600.
30	releaseModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during release phase. For the Modulation Envelope, the release phase linearly ramps toward zero from the current level. If the current level were full scale, the Modulation Envelope Release Time would be the time spent in release phase until zero value were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
31	keynumToModEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave

		causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
32	keynumToModEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
33	delayVolEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Volume envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
34	attackVolEnv	This is the time, in absolute timecents, from the end of the Volume Envelope Delay Time until the point at which the Volume Envelope value reaches its peak. Note that the attack is “convex”; the curve is nominally such that when applied to the decibel volume parameter, the result is linear in amplitude. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
35	holdVolEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the Volume envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
36	decayVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during decay phase. For the Volume Envelope, the decay phase linearly ramps toward the sustain level, causing a constant dB change for each time unit. If the sustain level were -96dB, the Volume Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
37	sustainVolEnv	This is the decrease in level, expressed in centibels, to which the Volume Envelope value ramps during the decay phase. For the

		Volume Envelope, the sustain level is best expressed in centibels of attenuation from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; conventionally 1000 indicates full attenuation. For example, a sustain level which corresponds to an absolute value 12dB below of peak would be 120.
38	releaseVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during release phase. For the Volume Envelope, the release phase linearly ramps toward zero from the current level, causing a constant dB change for each time unit. If the current level were full scale, the Volume Envelope Release Time would be the time spent in release phase until 96dB attenuation were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
39	keynumToVolEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
40	keynumToVolEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
41	instrument	This is the index into the INST subchunk providing the instrument to be used for the current preset zone. A value of zero indicates the first instrument in the list. The value should never exceed two less than the size of the instrument list. The instrument enumerator is the terminal generator for PGEN zones. As such, it should only appear in the PGEN subchunk, and it must appear as the last generator enumerator in all but the global preset zone.
42	reserved1	Unused, reserved. Should be ignored if encountered.
43	keyRange	This is the minimum and maximum MIDI key number values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid key. The

		keyRange enumerator is optional, but when it does appear, it must be the first generator in the zone generator list.
44	velRange	This is the minimum and maximum MIDI velocity values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid velocity. The velRange enumerator is optional, but when it does appear, it must be preceded only by keyRange in the zone generator list.
45	startloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Startloop sample header parameter and the first sample data point to be repeated in this instrument's loop. This parameter is added to the startloopAddrsOffset parameter. For example, if Startloop were 5, startloopAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point in the loop would be sample data point 65544.
46	keynum	This enumerator forces the MIDI key number to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
47	velocity	This enumerator forces the MIDI velocity to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
48	initialAttenuation	This is the attenuation, in .4 centibel units, by which a note is attenuated below full scale. A value of zero indicates no attenuation; the note will be played at full scale. For example, a value of 60 indicates the note will be played at 2.4 dB below full scale for the note.
49	reserved2	Unused, reserved. Should be ignored if encountered.
50	endloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. This parameter is added to the endloopAddrsOffset parameter. For example, if Endloop were 5, endloopAddrsOffset were 3 and endAddrsCoarseOffset were 2, sample data point 65544 would be considered equivalent to the Startloop sample data point, and hence sample data point 65543 would effectively precede Startloop during looping.
51	coarseTune	This is a pitch offset, in semitones, which should be applied to the note. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Coarse Tune value of -4 would cause the sound to be reproduced four semitones flat.
52	fineTune	This is a pitch offset, in cents, which should be applied to the note. It is additive with coarseTune. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Fine Tuning value of -5 would cause the sound to be reproduced five cents flat.

53	sampleID	This is the index into the SHDR subchunk providing the sample to be used for the current instrument zone. A value of zero indicates the first sample in the list. The value should never exceed two less than the size of the sample list. The sampleID enumerator is the terminal generator for IGEN zones. As such, it should only appear in the IGEN subchunk, and it must appear as the last generator enumerator in all but the global zone.
54	sampleModes	This enumerator indicates a value which gives a variety of Boolean flags describing the sample for the current instrument zone. The sampleModes should only appear in the IGEN subchunk, and should not appear in the global zone. The two LS bits of the value indicate the type of loop in the sample: 0 indicates a sound reproduced with no loop, 1 indicates a sound which loops continuously, 2 is unused but should be interpreted as indicating no loop, and 3 indicates a sound which loops for the duration of key depression then proceeds to play the remainder of the sample.
55	reserved3	Unused, reserved. Should be ignored if encountered.
56	scaleTuning	This parameter represents the degree to which MIDI key number influences pitch. A value of zero indicates that MIDI key number has no effect on pitch; a value of 100 represents the usual tempered semitone scale.
57	exclusiveClass	This parameter provides the capability for a key depression in a given instrument to terminate the playback of other instruments. This is particularly useful for percussive instruments such as a hi-hat cymbal. An exclusive class value of zero indicates no exclusive class; no special action is taken. Any other value indicates that when this note is initiated, any other sounding note with the same exclusive class value should be rapidly terminated. The exclusive class generator can only appear at the instrument level. The scope of the exclusive class is the entire preset. In other words, any other instrument zone within the same preset holding a corresponding exclusive class will be terminated.
58	overridingRootKey	This parameter represents the MIDI key number at which the sample is to be played back at its original sample rate. If not present, or if present with a value of -1, then the sample header parameter Original Key is used in its place. If it is present in the range 0-127, then the indicated key number will cause the sample to be played back at its sample header Sample Rate. For example, if the sample were a recording of a piano middle C (Original Key = 60) at a sample rate of 22.050 kHz, and Root Key were set to 69, then playing MIDI key number 69 (A above middle C) would cause a piano note of pitch middle C to be heard.
59	unused5	Unused, reserved. Should be ignored if encountered.
60	endOper	Unused, reserved. Should be ignored if encountered. Unique name provides value to end of defined list.

5.9.3.1.3 Generator Summary

The following tables give the ranges and default values for all SASBF defined generators.

#	Name	Unit	Abs Zero	Min	Min Useful	Max	Max Useful	De-fault	Def Value
0	startAddrsOffset +	smpls	0	0	None	*	*	0	None
1	endAddrsOffset +	smpls	0	*	*	0	None	0	None
2	startloopAddrsOffset +	smpls	0	*	*	*	*	0	None
3	endloopAddrsOffset +	smpls	0	*	*	*	*	0	None
4	startAddrsCoarseOffset +	32k	0	0	None	*	*	0	None
5	modLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
6	vibLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
7	modEnvToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
8	initialFilterFc	cent	8.176 Hz	1500	20 Hz	13500	20 kHz	13500	Open
9	initialFilterQ	cB	0	0	None	960	96 dB	0	None
10	modLfoToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
11	modEnvToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
12	endAddrsCoarseOffset +	32k	0	*	*	0	None	0	None
13	modLfoToVolume	cB fs	0	-960	-96 dB	960	96 dB	0	None
15	chorusEffectsSend	0.1%	0	0	None	1000	100%	0	None
16	reverbEffectsSend	0.1%	0	0	None	1000	100%	0	None
17	pan	0.1%	Cntr	-500	Left	+500	Right	0	Centre
21	delayModLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
22	freqModLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
23	delayVibLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
24	freqVibLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
25	delayModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
26	attackModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
27	holdModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
28	decayModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
29	sustainModEnv	-0.1%	attk peak	0	100%	1000	0%	0	attk pk
30	releaseModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
31	keynumToModEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
32	keynumToModEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
33	delayVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
34	attackVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
35	holdVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
36	decayVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
37	sustainVolEnv	cB attn	attk peak	0	0 dB	1440	144dB	0	attk pk
38	releaseVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
39	keynumToVolEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
40	keynumToVolEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
43	keyRange	MIDI ky#	key# 0	0	lo key	127	hi key	0-127	full kbd
44	velRange	MIDI vel	0	0	min vel	127	max vel	0-127	all vels
45	startloopAddrsCoarseOffset +	smpl	0	*	*	*	*	0	None

46	keynum +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None
47	velocity +	MIDI vel	0	1	min vel	127	max vel	-1	None
48	initialAttenuation	.4 cB	0	0	0 dB	1440	144dB	0	None
50	endloopAddrsCoarseOffset	smpls	0	*	*	*	*	0	None
51	CoarseTune	semitone	0	-120	-10 oct	120	10 oct	0	None
52	fineTune	cent	0	-99	-99cnt	99	99cent	0	None
54	sampleModes +	Bit Flags	Flags	**	**	**	**	0	No Loop
56	scaleTuning	cent/key	0	0	none	1200	oct/ky	100	semi-tone
57	exclusiveClass +	arbitrary #	0	1	--	127	--	0	None
58	overridingRootKey +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None

* Range depends on values of start, loop, and end points in sample header.

** Range has discrete values based on bit flags

\+ This generator is only valid at the instrument level.

5.9.3.2 Default Modulators

The “default” modulators are described below.

5.9.3.2.1 MIDI Key Velocity to Initial Attenuation

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.2 MIDI Key Velocity to Filter Cutoff

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is -2400 Cents.

The product of these values is added to the Initial Filter Cutoff generator summing node.

5.9.3.2.3 MIDI Channel Pressure to Vibrato LFO Pitch Depth

The MIDI Channel Pressure data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 50 cents per max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.4 MIDI Continuous Controller 1 to Vibrato LFO Pitch Depth

The MIDI Continuous Controller 1 data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. The MIDI Continuous Controller 33 data value may be optionally used for increased resolution of the controller input.

There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 50 cents/max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.5 MIDI Continuous Controller 7 to Initial Attenuation

The MIDI Continuous Controller 7 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.6 MIDI Continuous Controller 10 to Pan Position

The MIDI Continuous Controller 10 data value is used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 127 is mapped to 63/64 and all other values are mapped between -1 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 1000 tenths of a percent panned-right.

The product of these values is added to the Pan generator summing node.

5.9.3.2.7 MIDI Continuous Controller 11 to Initial Attenuation

The MIDI Continuous Controller 11 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.8 MIDI Continuous Controller 91 to Reverb Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added reverb send.

The product of these values is added to the Reverb Send generator summing node.

5.9.3.2.9 MIDI Continuous Controller 93 to Chorus Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added chorus send.

The product of these values is added to the Chorus Send generator summing node.

5.9.3.2.10 MIDI Pitch Wheel to Initial Pitch Controlled by MIDI Pitch Wheel Sensitivity

The MIDI Pitch Wheel data values are used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 16383 is mapped to 8191/8192 and all other values are mapped between -1 and 8191/8192 in a linear fashion.

The MIDI Pitch Wheel Sensitivity data values are used as a secondary source. This source is Positive Unipolar, thus an input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion.

The amount of this modulator is 12700 Cents.

The product of these values is added to the Initial Pitch generator summing node.

5.9.3.3 Precedence and Absolute and Relative values.

Most SASBF generators are available at both the Instrument and Preset Levels, as well as having a default value. Generators at the Instrument Level are considered “absolute” and determine an actual physical value for the associated synthesis parameter, which is used instead of the default. For example, a value of 1200 for the attackVolEnv generator would produce an absolute time of 1200 timecents or 2 seconds of attack time for the volume envelope, instead of the default value of -12000 timecents or 1 msec.

Generators at the Preset Level are instead considered “relative” and additive to all the default or instrument level generators within the Preset Zone. For example, a value of 2400 timecents for the attackVolEnv generator in a preset zone containing an instrument with two zones, one with the default attackVolEnv and one with an absolute attackVolEnv generator value of 1200 timecents would cause the default zone to actually have a value of -9600 timecents or 4 msec, and the other to have a value of 3600 timecents or 8 seconds attack time.

There are some generators which are not available at the Preset Level. These are:

#	Name
0	startAddrsOffset
1	endAddrsOffset
2	startloopAddrsOffset
3	endloopAddrsOffset
4	startAddrsCoarseOffset
12	endAddrsCoarseOffset
45	startloopAddrsCoarseOffset
46	keynum
47	velocity
50	endloopAddrsCoarseOffset
54	sampleModes

57	exclusiveClass
58	overridingRootKey

If these generators are encountered in the Preset Level, they should be ignored.

The effect of modulators on a given destination is always relative to the generator value at the Instrument level. However modulators may supersede or add to other modulators depending on their position within the hierarchy. Please see Subclause 8.4 for details on the Modulator implementation and the hierarchical details.

5.9.4 Parameters and Synthesis Model

The SASBF standard has been established with the intent of providing support for an expanding base of wavetable based synthesis models. The model supported by the SASBF specification originates with the EMU8000 wavetable synthesiser chip. The description below of the underlying synthesis model and the associated parameters are provided to allow mapping of this synthesis model onto other hardware platforms.

5.9.4.1 Synthesis Model

The SASBF specification Synthesis Model comprises a wavetable oscillator, a dynamic lowpass filter, an enveloping amplifier, and programmable sends to pan, reverb, and chorus effects units. An underlying modulation engine comprises two low frequency oscillators (LFOs) and two envelope generators with appropriate routing amplifiers.

5.9.4.1.1 Wavetable Oscillator/Interpolator

The SASBF specification wavetable oscillator model is capable of playing back a sample at an arbitrary sampling rate with an arbitrary pitch shift. In practice, the upward pitch shift (downward sample rate conversion) will be limited to a maximum value, typically at least two octaves. The pitch is described in terms of an initial pitch shift which is based on the sample's sampling rate, the root key at which the sample should be unshifted on the keyboard, the coarse, fine, and correction tunings, the effective MIDI key number, and the keyboard scale factor. All modulations in pitch are in octaves, semitones, and cents.

In general, interpolators have a symmetric impulse response, and thus a linear phase characteristic. The interpolation filter's magnitude response can be characterised by three regions - the pass band, the transition band, and the stop band.

The interpolation filter's pass band is the portion of its response which corresponds to the frequencies of the signal stored in waveform memory from DC to that signal's Nyquist frequency

The interpolation filter's transition band is the portion of its response which corresponds to the first image of the signal stored in waveform memory, that is from that signal's Nyquist frequency back to DC.

The interpolation filter's stop band is the portion of its response which corresponds to the remaining images above the transition band to the Nyquist frequency of the upsampled signal.

The guardband will be assumed to begin at 5/6 of the Nyquist frequency, as is the case for a 20 kHz bandwidth in a 48 kHz sample rate system.

Due to poor aliasing rejection in the stopband, linear interpolation does not satisfy this specification.

The specification constrains the Fourier transform of the impulse response of the interpolator. The impulse response is taken with a downward pitch shift of eight octaves. This gives a response which has been upsampled by a factor of 2**8 or 256. In other words, a frequency of the impulse response's Nyquist

frequency divided by 256 gives the filter frequency corresponding to the waveform memory's Nyquist frequency. In the specification below, Fn represents this waveform memory Nyquist frequency.

(All responses measured with downward pitch shift of 8 octaves.)		
Passband Response:		
	Ripple:	No more than +/- 0.5 dB
	Roll-off:	Minimal, but not to exceed 6 dB at 83.3% Fn.
Transition Band Response:		
	Roll-off:	Monotonic and as rapid as possible to at least -80 dB attenuation
	Return Lobes:	None above -80 dB
	Attenuation:	Greater than 80 dB for all frequencies DC to at least 2% Fn in the first lobe.
Stop Band Response:		
	Attenuation:	Greater than 90 dB for all frequencies DC to at least 1% Fn
		Greater than 80 dB for all frequencies DC to at least 20%Fn
		Greater than 60 dB for all frequencies

5.9.4.1.2 Sample Looping

The wavetable oscillator is playing a digital sample which is described in terms of a start point, end point, and two points describing a loop. The sound can be flagged as unlooped, in which case the loop points are ignored. If the sound is looped, it can be played in two ways. If it is flagged as "loop during release", the sound is played from the start point through the loop, and loops until the note becomes inaudible. If not, the sound is played from the start point through the loop, and loops until the key is released. At this point, the next time the loop end point is reached, the sound continues through the loop end point and plays until the end point is reached, at which time audio is terminated.

5.9.4.1.3 Lowpass Filter

The synthesis model contains a resonant lowpass filter, which is characterised by a dynamic cutoff frequency and a fixed resonance (Q).

The filter is idealised at zero resonance as having a flat passband to the cutoff frequency, then a rolloff at 12 dB per octave above that frequency. The resonance, when non-zero, comprises a peak at the cutoff frequency, superimposed on the above response. The resonance is measured as a dB ratio of the resonant peak to the DC gain. The DC gain at any resonance is half of the resonance value below the DC gain at zero resonance; hence the peak height is half the resonance value above DC gain at zero resonance.

All modulations in cutoff frequency are in cents.

Topology: 2nd order AR lowpass filter or equivalent. Filter coefficients are updated in a smooth manner at the sample rate.

Noise and Distortion: Less than 0.003% of full scale on any sinusoidal input for any parameter setting.

Control Parameters: Cutoff Frequency and Resonance.

Resonance Parameter: Specifies the ratio in decibels of the gain of a resonant peak to the gain at DC, when the filter cutoff frequency is set to 1.5 kHz and the modulation is zero. The DC gain of a filter is reduced from the DC gain of a filter with zero resonance by half the resonance value. The resonance parameter will remain fixed throughout the sounding of a musical note.

For a specified resonance, the actual gain ratio should be accurate within +/- 1 dB, and the DC gain accurate within +/-0.5 dB. The ratio of the gain at the resonant peak to the DC gain for all cutoff frequency values shall not deviate by more than 2dB per octave deviation from 1.5 kHz. Nominal gain at DC for a minimum resonance parameter is unity gain.

Cutoff Frequency Parameter: Specifies the frequency at which the filter achieves exactly 3dB of attenuation. The Cutoff Frequency is computed by the combination of the Initial Cutoff Frequency, specified in Hz, and the modulation, specified in semitones and fractions thereof. The Initial Cutoff Frequency is fixed throughout the duration of the a musical note; the modulation can vary at the sample rate. Within the region from 200 Hz to 6.4 kHz, the cutoff frequency parameter accuracy will be +/- 2 semitones.

Frequency Magnitude Response: When the cutoff frequency is at its maximum value and the resonance is at zero, the filter must pass audio without alteration. The filter must support cutoff frequencies down to 200 Hz. There must be a minimum of 2048 distinct cutoff frequencies per octave within the region from 200 Hz to 6.4 kHz, with a worst case spacing between distinct frequencies of 0.01 semitones.

5.9.4.1.4 Final Gain Amplifier

The final gain amplifier is a multiplier on the filter output, which is controlled by an initial gain in dB. This is added to the volume envelope. Additional modulation can also be added. The gain is always specified in dB.

5.9.4.1.5 Effects Sends

The output of the final gain amplifier can be routed into the effects unit. This unit causes the sound to be located (panned) in the stereo field, and a degree of reverberation and chorus to be added. The pan is specified in terms of percentage left and right, which also could be considered as an azimuth angle. The reverb and chorus sends are specified as a percentage of the signal amplitude to be sent to these units, from 0% to 100%.

5.9.4.1.6 Low Frequency Oscillators

The synthesis model provides for two low frequency oscillators (LFOs) for modulating pitch, filter cutoff, and amplitude. The “vibrato” LFO is only capable of modulating pitch. The “modulation” LFO can modulate any of the three parameters.

An LFO is defined as having a delay period during which its value remains zero, followed by a triangular waveshape ramping linearly to positive one, then downward again to negative 1, then upward again to positive one, etc.

Each parameter can be modulated to a varying degree, either positively or negatively, by the associated LFO. Modulations of pitch and cutoff are in octaves, semitones, and cents, while modulations of amplitude are in dB. The degree of modulation is specified in cents or dB for the full scale positive LFO excursion.

5.9.4.1.7 Envelope Generators

The synthesis model provides for two envelope generators. The volume envelope generator controls the final gain amplifier and hence determines the volume contour of the sound. The modulation envelope can control pitch and/or filter cutoff.

An envelope generates a control signal in six phases. When key-on occurs, a delay period begins during which the envelope value is zero. The envelope then rises in a convex curve to a value of one during the attack phase. When a value of one is reached, the envelope enters a hold phase during which it remains at one. When the hold phase ends, the envelope enters a decay phase during which its value decreases linearly to a sustain level. When the sustain level is reached, the envelope enters sustain phase, during which the envelope stays at the sustain level. Whenever a key-off occurs, the envelope immediately enters a release phase during which the value linearly ramps from the current value to zero. When zero is reached, the envelope value remains at zero.

Modulation of pitch and filter cutoff are in octaves, semitones, and cents. These parameters can be modulated to varying degree, either positively or negatively, by the modulation envelope. The degree of modulation is specified in cents for the full scale attack peak.

The volume envelope operates in dB, with the attack peak providing a full scale output, appropriately scaled by the initial volume. The zero value, however, is actually zero gain. When 96 dB of attenuation is reached in the final gain amplifier, an abrupt jump to zero gain (infinite dB of attenuation) occurs. In a 16-bit system, this jump is inaudible.

5.9.4.1.8 Modulation Interconnection Summary

The following diagram shows the interconnections expressed in the SASBF specification synthesis model:

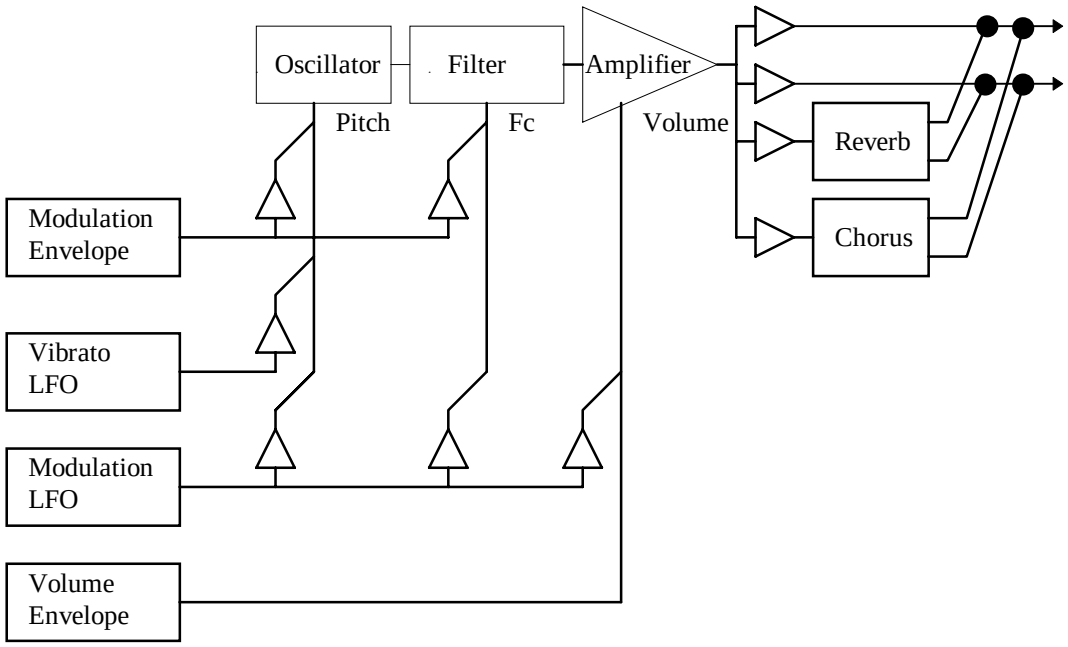


Figure 1: Generator Based Modulation Structure

5.9.4.2 MIDI Functions

The response to certain MIDI commands is defined within the MIDI specification, and therefore not considered to be part of the SASBF specification. These MIDI commands may not be used as sources for the Modulator implementation.

For completeness, the expected responses are given here.

MIDI CC0 Bank Select - When received, the following program change should select the MIDI program in this bank value instead of the default bank of 0.

MIDI CC6 - Data Entry MSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism depending on the Data Entry mode.

MIDI CC32 Bank Select LSB - When received, may behave in conjunction with CC0 Bank Select to provide a total of 16384 possible MIDI banks of programs.

MIDI CC38 Data Entry LSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism, depending on the Data Entry mode.

MIDI CC64 Sustain - ACTIVE when greater than or equal to 64. When the sustain function is active, all notes in the key-on state remain in the key-on state regardless of whether a key-off command for the note arrives. The key-off commands are stored, and when sustain becomes inactive, all stored key-off commands are executed.

MIDI CC66 Soft - ACTIVE when greater than or equal to 64. When active, all new key-ons are modulated in such a way to make the note sound “soft.” This typically affects initial attenuation and filter cutoff in a pre-defined manner.

MIDI CC67 Sostenuto - ACTIVE when greater than or equal to 64. When sostenuto becomes active, all notes currently in the key-on state remain in the key-on state until the sostenuto becomes inactive. All other notes behave normally. Notes maintained by sostenuto in key-on state remain in key-on state even if sustain is switched on and off.

MIDI CC98 NRPN LSB - When received, should be processed by the NRPN implementation mechanism.

MIDI CC99 NRPN MSB - When received, should put the synthesiser in NRPN Data Entry mode and then should be processed by the NRPN implementation mechanism.

MIDI CC100 RPN LSB - When received, should be processed by the RPN implementation mechanism.

MIDI CC101 RPN MSB - When received, should put the synthesiser in RPN Data Entry mode and then should be processed by the RPN implementation mechanism.

MIDI CC120 All Sound Off - When received with any data value, all notes playing in the key-on state bypass the release phase and are shut off, regardless of the sustain or sostenuto positions.

MIDI CC121 Reset All Controllers - Defined as Reset All Controllers as defined by the MIDI specification.

MIDI CC123 All Notes Off - When received with any data value, all notes playing in the key-on state immediately enter release phase, pending their status in SUSTAIN or SOSTENUTO state.

5.9.4.3 Parameter Units

The units with which SASBF generators are described are all well defined. The strict definitions appear below:

ABSOLUTE SAMPLE DATA POINTS - A numeric index of 16 bit sample data point words as stored in ROM or supplied in the smpl-ck, indexing the first sample data point word of memory or the chunk as zero.

RELATIVE SAMPLE DATA POINTS - A count of 16 bit sample data point words based on an absolute sample data point reference. A negative value implies a relative count toward the beginning of the data.

ABSOLUTE SEMITONES - An absolute logarithmic measure of frequency based on a reference of MIDI key numbers. A semitone is 1/12 of an octave, and value 69 is 440 Hz (A-440). Negative values and values above 127 are allowed.

RELATIVE SEMITONES - A relative logarithmic measure of frequency ratio based on units of 1/12 of an octave, which is the twelfth root of two, approximately 1.059463094.

ABSOLUTE CENTS - An absolute logarithmic measure of frequency based on a reference of MIDI key number scaled by 100. A cent is 1/1200 of an octave, and value 6900 is 440 Hz (A-440). Negative values and values above 12700 are allowed.

RELATIVE CENTS - A relative logarithmic measure of frequency ratio based on units of 1/1200 of an octave, which is the twelve hundredth root of two, approximately 1.000577790.

ABSOLUTE CENTIBELS - An absolute measure of the attenuation of a signal, based on a reference of zero being no attenuation. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

RELATIVE CENTIBELS - A relative measure of the attenuation of a signal. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

ABSOLUTE TIMECENTS - An absolute measure of time, based on a reference of zero being one second. A timecent represents a ratio in time of the twelve hundredth root of two, approximately 1.011579454.

RELATIVE TIMECENTS - A relative measure of time ratio, based on a unit size of the twelve hundredth root of two, approximately 1.011579454.

ABSOLUTE PERCENT - An absolute measure of gain, based on a reference of unity. In SASBF, absolute percent is measured in 0.1% units, so a value of zero is 0% and a value of 1000 is 100%.

RELATIVE PERCENT - A relative measure of gain difference. In SASBF, relative percent is measured in 0.1% units. When the gain goes below zero, zero is assumed; when the gain exceeds 100%, 100% is used.

5.9.4.4 The SASBF Generator Model

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

In case it is not clear in the general description of the SASBF hierarchy, the following is the precedence of SASBF generator in the hierarchy.

- A ‘generator’ sets or offsets the value of a destination or a synthesis parameter. In exception cases, it sets ranges (Range Generators), or sets values and never offsets values (Index Generators, Sample Generators, and Substitution Generators).
- A generator is defined as identical to another generator if its generator operator is the same in both generators.
- A generator in a global instrument zone which is identical to a default generator supersedes or replaces the default generator.
- A generator in a local instrument zone which is identical to a default generator or to a generator in a global instrument zone supersedes or replaces that generator.
- Points below (until noted) apply to Value Generators ONLY.
- A generator at the preset level adds to a generator at the instrument level if both generators are identical.
- A generator in a global preset zone which is identical to a default generator or to a generator in an instrument adds to that generator.
- A generator in a global preset zone which is not identical to a default generator and is not identical to a generator in an instrument has its effect added to the given synthesis parameter.
- A generator in a local preset zone which is identical to a generator in a global preset zone supersedes or replaces that generator in the global preset zone. That generator then has its effects added to the destination summing node of all zones in the given instrument.
- A generator in a local preset zone which is not identical to a default generator or a generator in a global preset zone has its effects added to the destination summing node of all zones in the given instrument.
  If the generator operator is a Range Generator, the generator values are NOT ADDED to those

in the instrument level, rather they serve as an intersection filter to those key number or velocity ranges in the instrument which is used in the preset zone.

- If the generator operator is a Substitution Generator or a Sample Generator, they are illegal at the preset level. The only Index Generator legal at the Preset Level is ‘instrumentID’, whereas the only Index Generator legal at the Instrument Level is ‘sampleID’

5.9.4.5 The SASBF Modulator Controller Model

SASBF Modulators are used to allow real-time control over the sound in sound designer programmable manner. Each instance of a SASBF modulator structure defines a real-time perceptually additive effect to be applied to a given destination or synthesiser parameter.

Modulators provide future extensibility to the SASBF standard. They are not used in this version.

5.9.5 Error Handling

5.9.5.1 Structural Errors

Structural Errors are errors which are determined from the implicit redundancy of the SASBF RIFF bitstream element structure, and indicate that the structure is not intact. Examples are incorrect lengths for the chunks or subchunks, pointers out of valid range, or missing required chunks or subchunks for which no error correction procedure exists.

In all cases, bitstream elements should be checked for structural errors at load time, and if any are found, the bitstream elements should be rejected. Separate tools or options can be used to “repair” structurally defective bitstream elements, but these tools should validate that the reconstructed bitstream element is not only a valid SASBF bank but also complies with the intended timbral results in all cases.

5.9.5.2 Unknown Chunks

In parsing the RIFF structure, unknown but well formed chunks or subchunks may be encountered. Unknown chunks within the INFO-list chunk should simply be ignored. Other unknown chunks or subchunks are illegal and should be treated as structural errors.

5.9.5.3 Unknown Enumerators

Unknown enumerators may be encountered in Generators, Modulator Sources, or Transforms. This is to be expected if the ifil field exceeds the specification to which the application was written. Even if unexpected, unknown enumerators should simply cause the associated Generator or Modulator to be ignored.

5.9.5.4 Illegal Parameter Values

Some SASBF parameters are defined for only a limited range of the possible values which can be expressed in their field. If the value of the field is not in the defined range, the parameter has an illegal value.

Illegal values for may be detected either at load or at run time. If detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the default value for the parameter should be used if the parameter is required, or the entire Generator or Modulator ignored if it is optional. Certain parameters may have more specific procedures for illegal values as expressed elsewhere in this specification.

5.9.5.5 Out-of-range Values

SASBF parameters have a specified minimum and useful range the span the perceptually relevant values for the associated sonic property. When the parameter value is exceeds this useful range, the parameter is said to have an out of range value.

Out of range values can result from two distinct causes. An out of range value can be actually present as a SASBF generator value, or the out of range value can be the result of the summation of instrument and preset values.

Out of range values should be handled by substituting the nearest perceptually relevant or realisable value. SASBF banks should not be created with out of range values in the instrument generators. While it is acceptable practice to create SASBF banks which produce out of range values as a result of summation, it is undesirable and should be avoided where practical.

5.9.5.6 Missing Required Parameter or Terminator

Certain parameters and terminators are required by the SASBF specification. If these are missing, the bitstream element is technically not within specification. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the instrument or zone for which the required parameter is missing should simply be ignored. If this causes no sound, the corresponding key-on event is ignored.

5.9.5.7 Illegal enumerator

Certain enumerators are illegal in certain contexts. For example, key and velocity ranges must be the first generators in a zone, instruments are not allowed in instrument zones, and sampleIDs are not allowed in preset zones. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the enumerator should simply be ignored.

5.9.6 Profile 2 (Sample Bank and MIDI decoding)

This Subclause describes the decoding process in which a bitstream conforming to Profile 2 is converted into sound. Profile 2 supports the Structured Audio Sample Bank Format and the General MIDI Format (Profile 1) for sound description. Profile 2 supports the MIDI Protocol and the Standard MIDI File Format for score specification.

5.9.6.1 Stream information header

The SASBF bitstream element is part of the bitstream header. Score elements in the form of MIDI files may be included in the bitstream header.

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class `SA_streaminfo` provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- MIDI file: The events in the MIDI file shall be time ordered, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed. The sample bank requires no processing for this preparation. Processing may be used to improve the computational

efficiency of a decoder. Banks shall be assigned consecutive increasing bank ID numbers in the order of the banks' position in the bitstream. The first bank in the bitstream shall be numbered 0 unless a bank resident in the terminal is being used. In that case, the resident bank shall be numbered 0, an other banks shall be numbered proceeding from there.

5.9.6.2 Bitstream data and sound creation

The MIDI Note On message is defined in the MIDI specification. When the SASBF decoder receives a Note On message, a voice shall be instantiated. Synthesis parameters for the voice shall be determined by the data in the sample bank containing the preset corresponding to the MIDI channel of the Note On message. The number of wavetable oscillators that are used by the voice is defined by the sample bank. Each required oscillator shall have the state variables required for synthesis allocated to it. The state variables shall be initialised according to the preset data from the sample bank.

The MIDI Program Change message is defined in the MIDI specification. When the SASBF decoder receives a Program Change message, an assignment shall be made in the scheduler between the specified MIDI channel and the specified preset. Until this assignment is changed by a subsequent Program Change message, subsequent voice instantiations using the specified MIDI channel shall use sample bank data corresponding to the assigned preset. In a MIDI program change message, if no preset exists in the specified bank with the specified preset number, a replacement preset is used. The replacement preset is the preset with the specified preset number in the bank with the highest bank ID number less than the specified bank ID number which contains a preset with the specified preset number.

The run time scheduler is used to issue MIDI messages to the SASBF decoder. MIDI messages from a MIDI standard file shall be issued by the scheduler when the scheduler clock equals or exceeds the time stamp associated with the message.

5.9.6.3 Conformance

Floating point computation is not required in Profile 2. Audio samples are stored in the bitstream as 16-bit integers. The resolution of the interpolator filter is constrained by the interpolator specification given in Subclause 0.

5.9.7 Profile 4 (Sample Bank decoding in SAOL instruments)

5.9.8 Sample Bank Format Glossary

absolute - Describes a parameter which gives a definitive real-world value. Contrast to relative.

additive - Describes a parameter which is to be numerically added to another parameter.

articulation - The process of modulation of amplitude, pitch, and timbre to produce an expressive musical note.

attack - That phase of an envelope or sound during which the amplitude increases from zero to a peak value.

attenuation - A decrease in volume or amplitude of a signal.

bag - A Sample Bank Format data structure element containing a list of zones.

balance - A form of stereo volume control in which both left and right channels are at maximum when the control is centred, and which attenuates only the opposite channel when taken to either extreme.

bank - A collection of presets. See also MIDI bank.

bipolar - In the SASBF standard, said of a modulator source whose minimum is -1 and whose maximum is 1. Contrast “unipolar”

big endian - Refers to the organisation in memory of bytes within a word such that the most significant byte occurs at the lowest address. Contrast “little endian.”

byte - A data structure element of eight bits without definition of meaning to those bits.

BYTE - A data structure element of eight bits which contains an unsigned value from 0 to 255.

case-insensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as identical. Contrast “case-sensitive.”

case-sensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as distinct. Contrast “case-insensitive.”

cent - A unit of pitch ratio corresponding to the twelve hundredth root of two, or one hundredth of a semitone, approximately 1.000577790.

centibel - A unit of amplitude ratio corresponding to the two hundredth root of ten, or one tenth of a decibel, approximately 1.011579454.

CHAR - A data structure of eight bits which contains a signed value from -128 to +127.

chorus - An effects processing algorithm which involves cyclically shifting the pitch of a signal and remixing it with itself to produce a time varying comb filter, giving a perception of motion and fullness to the resulting sound.

chunk - The top-level division of a RIFF file.

convex - A curve which is bowed in such a way that it is steeper on its lower portion.

concave - (1) A curve which is bowed in such a way that it is steeper on its upper portion. (2) In the SASBF standard, said of a modulator source whose shape is that of the amplitude squared characteristic. Contrast with “convex” and “linear.”

cutoff frequency - The frequency of a filter function at which the attenuation reaches a specified value.

data points - The individual values comprising a sample. Sometimes also called sample points. Contrast “sample.”

decay - The portion of an envelope or sound during which the amplitude declines from a peak to steady state value.

decibel - A unit of amplitude ratio corresponding to the twentieth root of ten, approximately 1.122018454.

delay - The portion of an envelope or LFO function which elapses from a key-on event until the amplitude becomes non-zero.

destination - The generator to which a modulator is applied.

DC gain - The degree of amplification or attenuation a system presents to a static or zero frequency signal.

digital audio - Audio represented as a sequence of quantised values spaced evenly over time. The values are called “sample data points.”

doubleword - A data structure element of 32 bits without definition of meaning to those bits.

downloadable - Said of samples which are loaded from a file into RAM, in contrast to samples which are maintained in ROM.

dry - Refers to audio which has not received any effects processing such as reverb or chorus.

DWORD - A data structure of 32 bits which contains an unsigned value from zero to 4,294,967,295.

envelope - A time varying signal which typically controls the pitch, volume, and/or filter cutoff frequency of a note, and comprises multiple phases including attack, decay, sustain, and release.

enumerated - Said of a data element whose symbols correspond to particular assigned functions.

flat - A. Said of a tone that is lower in pitch than another reference tone. B. Said of a frequency response that does not deviate significantly from a single fixed gain over the audio range.

generator - In the SASBF standard, a parameter which directly affects sound reproduction. Contrast with “modulator.”

global - Refers to parameters which affect all associated structures. See “global zone.”

global zone - A zone whose generators and modulators affect all other zones within the object.

header - A data structure element which describes several aspects of a SASBF element.

instrument - In the SASBF standard, a collection of zones which represents the sound of a single musical instrument or sound effect set.

instrument zone - A sample and associated articulation data defined to play over certain key numbers and velocities.

interpolator - A circuit or algorithm which computes intermediate points between existing sample data points. This is of particular use in the pitch shifting operation of a wavetable synthesiser, in which these intermediate points represent the output samples of the waveform at the desired pitch transposition.

key number - See MIDI key number.

LFO - Acronym for Low Frequency Oscillator. A slow periodic modulation source.

linear - In the SASBF standard, said of a modulator source whose shape is that of a straight line. Contrast with “concave.”

linear coding - The most common method of encoding amplitudes in digital audio in which each step is of equal size.

little endian - A method of ordering bytes within larger words in memory in which the least significant byte is at the lowest address. Contrast “big endian.”

loop - In wavetable synthesis, a portion of a sample which is repeated many times to increase the duration of the resulting sound.

loop points - The sample data points at which a loop begins and ends.

lowpass - Said of a filter which attenuates high frequencies but does not attenuate low frequencies.

modulator - In the SASBF standard, a parameter which routes an external controller to dynamically alter the setting of a “generator.” Contrast with “generator.”

monotonic - Continuously increasing or decreasing. Said of a sequence which never reverses direction.

MIDI - Acronym for Musical Instrument Digital Interface. The standard protocol for sending performance information to a musical synthesiser.

MIDI bank - A group of up to 128 presets selected by a MIDI “change bank” command.

MIDI continuous controller - A construct in the MIDI protocol.

MIDI key number - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the key of the musical instrument keyboard to which the command refers.

MIDI pitch bend - A special MIDI construct akin to the MIDI continuous controllers which controls the real-time value of the pitch of all notes played in a MIDI channel.

MIDI preset - A “preset” selected to be active in a particular MIDI channel by a MIDI “change preset” command.

MIDI velocity - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the speed with which the key was pressed or released.

modulator - In the SASBF standard, a set of parameters which affect a particular generator. Contrast with “generator.”

mono - Short for “monophonic.” Indicates a sound comprising only one channel or waveform. Contrast with “stereo.”

negative - In the SASBF standard, said of a modulator which has a negative sloping characteristic. Contrast with “positive.”

octave - A factor of two in ratio, typically applied to pitch or frequency.

orphan - Said of a data structure which under normal circumstances is referenced by a higher level, but in this particular instance is no longer linked. Specifically, it is an instrument which is not referenced by any preset zone, or a sample which is not referenced by any instrument zone.

oscillator - In wavetable synthesis, the wavetable interpolator is considered an oscillator.

pan - Short for “panorama.” This is the control of the apparent azimuth of a sound source over 180 degrees from left to right. It is generally implemented by varying the volume at the left and right speakers.

pitch - The perceived value of frequency. Generally can be used interchangeably with frequency.

pitch shift - A change in pitch. Wavetable synthesis relies on interpolators to cause pitch shift in a sample to produce the notes of the scale.

pole - A mathematical term used in filter transform analysis. Traditionally in synthesis, a pole is equated with a rolloff of 6dB per octave, and the rolloff of a filter is specified in “poles.”

positive - In the SASBF standard, said of a modulator source which has a positive sloping characteristic. Contrast “negative.”

preset - A keyboard full of sound. Typically the collection of samples and articulation data associated with a particular MIDI preset number.

preset zone - A subset of a preset containing generators, modulators, and an instrument.

proximal - Closest to. Proximal sample data points are the data points closest in either direction to the named point.

Q - A mathematical term used in filter transform analysis. Indicates the degree of resonance of the filter. In synthesis terminology, it is synonymous with resonance.

RAM - Random Access Memory. Conventionally, this term implies read-write memory. Contrast “ROM.”

record - A single instance of a data structure.

relative - Describes a parameter which merely indicates an offset from an otherwise established value. Contrast to absolute.

release - The portion of an envelope or sound during which the amplitude declines from a steady state to zero value or inaudibility.

resonance - Describes the aspect of a filter in which particular frequencies are given significantly more gain than others. The resonance can be measured in dB above the DC gain.

resonant frequency - The frequency at which resonance reaches its maximum.

reverb - Short for reverberation. In synthesis, a synthetic signal processor which adds artificial spaciousness and ambience to a sound.

RIFF - Acronym for Resource Interchange File Format.

ROM - Acronym for Read Only Memory. A memory whose contents are fixed at manufacture, and hence cannot be written by the user. Contrast with RAM.

sample - This term is often used both to indicate a “sample data point” and to indicate a collection of such points comprising a digital audio waveform. The latter meaning is exclusively used in this Subclause (the SASBF specification.)

sample rate - The frequency, in Hertz, at which sample data points are taken when recording a sample.

semitone - A unit of pitch ratio corresponding to the twelfth root of two, or one twelfth of an octave, approximately 1.059463094.

sharp - Said of a tone that is higher in pitch than another reference tone.

SHORT - A data structure element of sixteen bits which contains a signed value from -32,768 to +32,767.

soft - The pedal on a piano, so named because it causes the damper to be lowered in such a way as to soften the timbre and loudness of the notes. In MIDI, continuous controller #66, which behaves in a similar manner.

sostenuto - The pedal on a piano which causes the dampers on all keys depressed to be held until the pedal is released. In MIDI, continuous controller #67, which behaves in a similar manner.

sustain - The pedal on a piano which prevents all dampers on keys as they are depressed from being released. In MIDI, continuous controller #64, which behaves in a similar manner.

source - In a SASBF modulator, the enumerator indicating the particular real-time value which the modulator will transform, scale, and add to the destination generator.

stereo - Literally indicating three dimensions. In this specification, the term is used to mean two channel stereophonic, indicating that the sound is composed of two independent audio channels, dubbed left and right. Contrast monophonic.

subchunk - A division of a RIFF file below that of the chunk.

synthesis engine - The hardware and software associated with the signal processing and modulation path for a particular synthesiser.

synthesiser - A device ideally capable of producing arbitrary musical sound.

terminator - A data structure element indicating the final element in a sequence.

timecent - A unit of duration ratio corresponding to the twelve hundredth root of two, or one twelve hundredth of an octave, approximately 1.000577790.

transform - In a SASBF modulator, the enumerator indicating the particular transfer function through which the source will be passed prior to scaling and addition to the destination generator.

tremolo - A periodic change in amplitude of a sound, typically produced by applying a low frequency oscillator to the final volume amplifier.

triangular - A waveform which ramps upward to a positive limit, then downward at the opposite slope to the symmetrically negative limit periodically.

unipolar - In the SASBF standard, said of a modulator source whose minimum is 0 and whose maximum is 1. Contrast “bipolar.”

velocity - In synthesis, the speed with which a keyboard key is depressed, typically proportionally to the impact delivered by the musician. See also MIDI velocity.

vibrato - A periodic change in the pitch of a sound, typically produced by applying a low frequency oscillator to the oscillator pitch.

volume - The loudness or amplitude of a sound, or the control of this parameter.

wavetable - A music synthesis technique wherein musical sounds are recorded or computed mathematically and stored in a memory, then played back at a variable rate to produce the desired pitch. Additional timbral adjustments are often made to the sound thus produced using amplifiers, filters, and effects processing such as reverb and chorus.

WORD - A data structure of 16 bits which contains an unsigned value from zero to 65,535.

word - A data structure element of 16 bits without definition of meaning to those bits.

zone - An object and associated articulation data defined to play over certain key numbers and velocities.

5.10 MIDI semantics for more details on MIDI control of orchestras.

5.4.6.8.10 MIDItouch

```
ksig MIDItouch
```

The MIDItouch **standard variable shall contain, for each scope, the current value of the MIDI aftertouch on the note which caused the associated instrument instantiation to be created. See Subclause 5.9 Sample Bank syntax and semantics**

5.9.1 Introduction

This Subclause describes the operation of the Sample Bank synthesis method for both Profile 2 and Profile 4. In Profile 2, only Sample Bank and MIDI class types shall appear in the bitstream, and this Subclause describes the normative process of generating sound from a Sample Bank bitstream data element and a sequence of MIDI instructions. In Profile 4, Sample Banks are used in the context of a SAOL instrument as described in Subclause 5.4.6.6.8 Output, and this Subclause describes the normative process of generating sound and placing it on three busses, depending on the Sample Bank bitstream data element and the particular call to **sbsynth**.

5.9.2 Elements of bitstream

5.9.2.1 RIFF Structure

5.9.2.1.1 General RIFF File Structure

The RIFF (Resource Interchange File Format) is a tagged file structure developed for multimedia resource files, and is described in some detail in the Microsoft Windows 3.1 SDK Multimedia Programmer's Reference. The Tagged-file structure is useful because it helps prevent compatibility problems which can occur as the file definition changes over time. Because each piece of data in the file is identified by a standard header, an application that does not recognise a given data element can skip over the unknown information. The relevant information is provided in the bitstream description, Subclause 0.

A RIFF file is constructed from a basic building block called a "chunk." In 'C' syntax, a chunk is defined:

```
typedef DWORD FOURCC;      //  Four-character code
typedef struct {
  FOURCC ckID;             //  Chunk ID identifies type of data in the chunk.
  DWORD ckSize;            //  Size of chunk data in bytes, excluding pad byte.
  BYTE ckDATA[ckSize];     //  Actual data + a pad byte if req'd to word align.
};
```

Two types of chunks, the "RIFF" and "LIST" chunks, may contain nested chunks called subchunks as their data.

Within a given level of the hierarchy, the ordering of the chunks is arbitrary. Any chunk with an unknown chunk ID should be ignored.

5.9.2.1.2 The Sample Bank Chunks and Subchunks

A Structured Audio Sample Bank bitstream element comprises three chunks: an INFO-list chunk containing a number of required and optional subchunks describing the bitstream element, its history, and its intended use, an sdta-list chunk comprising a single subchunk containing any referenced digital audio samples, and a pdta-list chunk containing nine subchunks which define the articulation of the digital audio data.

5.9.2.1.3 Redundancy and Error Handling in the RIFF structure

The RIFF bitstream element structure contains redundant information regarding the length of the bitstream element and the length of the chunks and subchunks. This fact enables any reader of a SASBF bitstream element to determine if the bitstream element has been damaged by loss of data.

If any such loss is detected, the SASBF bitstream element is termed “structurally unsound” and in general should be rejected.

5.9.2.2 The INFO-list Chunk

The INFO-list chunk in a SASBF bitstream element contains three mandatory and a variety of optional subchunks as defined below. The INFO-list chunk gives basic information about the SASBF bank contained in the bitstream element.

5.9.2.2.1 The ifil Subchunk

The ifil subchunk is a mandatory subchunk identifying the SASBF specification version level to which the bitstream element complies. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the SASBF specification version. The WORD wMinor contains the value to the right of the decimal point. For example, version 2.11 would be implied if wMajor=2 and wMinor=11.

These values can be used by applications which read SASBF bitstream elements to determine if the format of the bitstream element is usable by the program. Within a fixed wMajor, the only changes to the format will be the addition of Generator, Source and Transform enumerators, and additional info subchunks. These are all defined as being ignored if unknown to the program.

If the ifil subchunk is missing, or its size is not four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.2.2 The isng Subchunk

The isng subchunk is a mandatory subchunk identifying the wavetable sound engine for which the bitstream element was optimised. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even.

The ASCII should be treated as case-sensitive. In other words “engine” is not the same as “ENGINE.”

The isng string may be optionally used by chip drivers to vary their synthesis algorithms to emulate the target sound engine.

If the isng subchunk is missing not terminated in a zero valued byte, or its contents are an unknown sound engine, the field should be ignored.

5.9.2.2.3 The INAM Subchunk

The INAM subchunk is a mandatory subchunk providing the name of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical inam subchunk would be the fourteen bytes representing “General MIDI” as twelve ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “General MIDI” is not the same as “GENERAL MIDI.”

If the inam subchunk is missing, or not terminated in a zero valued byte, the field should be ignored and the user supplied with an appropriate error message if the name is queried. If the bitstream element is re-written, a valid name should be placed in the INAM field.

5.9.2.2.4 The irom Subchunk

The irom subchunk is an optional subchunk identifying a particular wavetable sound data ROM to which any ROM samples refer. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical irom field would be the six bytes representing “1MGM” as four ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “1mgm” is not the same as “1MGM.”

The irom string is used by drivers to verify that the ROM data referenced by the bitstream element is available to the sound engine.

If the irom subchunk is missing, not terminated in a zero valued byte, or its contents are an unknown ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. A bitstream element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.5 The iver Subchunk

The iver subchunk is an optional subchunk identifying the particular wavetable sound data ROM revision to which any ROM samples refer. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the ROM version. The WORD wMinor contains the value to the right of the decimal point. For example, version 1.36 would be implied if wMajor=1 and wMinor=36.

The iver subchunk is used by drivers to verify that the ROM data referenced by the bitstream element is located in the exact locations specified by the sound headers.

If the iver subchunk is missing, not four bytes in length, or its contents indicate an unknown or incorrect ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. Note that for ROM samples to function correctly, both iver and irom must be present and valid. A bitstream

element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.6 The ICRD Subchunk

The ICRD subchunk is an optional subchunk identifying the creation date of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICRD field would be the twelve bytes representing “May 1, 1995” as eleven ASCII characters followed by a zero byte.

Conventionally, the format of the string is “Month Day, Year” where Month is initially capitalised and is the conventional full English spelling of the month, Day is the date in decimal followed by a comma, and Year is the full decimal year. Thus the field should conventionally never be longer than 32 bytes.

The ICRD string is provided for library management purposes.

If the ICRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.7 The IENG Subchunk

The IENG subchunk is an optional subchunk identifying the names of any sound designers or engineers responsible for the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IENG field would be the twelve bytes representing “Tim Swartz” as ten ASCII characters followed by two zero bytes.

The IENG string is provided for library management purposes.

If the IENG subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.8 The IPRD Subchunk

The IPRD subchunk is an optional subchunk identifying any specific product for which the SASBF bank is intended. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IPRD field would be the twelve bytes representing “MPEG SASBF” as ten ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “mpeg sasbf” is not the same as “MPEG SASBF.”

The IPRD string is provided for library management purposes.

If the IPRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.9 The ICOP Subchunk

The ICOP subchunk is an optional subchunk containing any copyright assertion string associated with the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICOP field would be the 38 bytes representing "Copyright (c) 1998 Content Developer" as 36 ASCII characters followed by two zero bytes.

The ICOP string is provided for intellectual property protection and management purposes.

If the ICOP subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.10 The ICMT Subchunk

The ICMT subchunk is an optional subchunk containing any comments associated with the SASBF bank. It contains an ASCII string of 65,536 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICMT field would be the 40 bytes representing "This space unintentionally left blank." as 38 ASCII characters followed by two zero bytes.

The ICMT string is provided for including comments.

If the ICMT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.11 The ISFT Subchunk

The ISFT subchunk is an optional subchunk identifying the SASBF tools used to create and most recently modify the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ISFT field would be the twenty-six bytes representing "Editor 2.00a:Editor 2.00a" as twenty-five ASCII characters followed by a zero byte.

The ASCII should be treated as case-sensitive. In other words "Editor" is not the same as "EDITOR."

Conventionally, the tool name and revision control number are included first for the creating tool and then for the most recent modifying tool. The two strings are separated by a colon. The string should be produced by the creating program with a null modifying tool field (e.g. "Editor 2.00a:), and each time a tool modifies the bank, it should replace the modifying tool field with its own name and revision control number.

The ISFT string is provided primarily for error tracing purposes.

If the ISFT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.3 The sdta-list Chunk

The sdta-list chunk in a SASBF bitstream element contains a single optional smpl subchunk which contains all the sound data associated with the SASBF bank. The smpl subchunk is of arbitrary length, and contains an even number of bytes.

5.9.2.3.1 Sample Data Format in the smpl Subchunk

The smpl subchunk, contains one or more samples of digital audio information in the form of linearly coded sixteen bit, signed, little endian (least significant byte first) words. Each sample is followed by a minimum of forty-six zero valued sample data points. These zero valued data points are necessary to guarantee that any reasonable upward pitch shift using any reasonable interpolator can loop on zero data at the end of the sound.

5.9.2.3.2 Sample Data Looping Rules

Within each sample, one or more loop point pairs may exist. The locations of these points are defined within the pdta-list chunk, but the sample data points themselves must comply with certain practices in order for the loop to be compatible across multiple platforms.

The loops are defined by “equivalent points” in the sample. This means that there are two sample data points which are logically equivalent, and a loop occurs when these points are spliced atop one another. In concept, the loop end point is never actually played during looping; instead the loop start point follows the point just prior to the loop end point. Because of the bandlimited nature of digital audio sampling, an artefact free loop will exhibit virtually identical data surrounding the equivalent points.

In actuality, because of the various interpolation algorithms used by wavetable synthesisers, the data surrounding both the loop start and end points may affect the sound of the loop. Hence both the loop start and end points must be surrounded by continuous audio data. For example, even if the sound is programmed to continue to loop throughout the decay, sample data points must be provided beyond the loop end point. This data will typically be identical to the data at the start of the loop. A minimum of eight valid data points are required to be present before the loop start and after the loop end.

The eight data points (four on each side) surrounding the two equivalent loop points should also be forced to be identical. By forcing the data to be identical, all interpolation algorithms are guaranteed to properly reproduce an artefact-free loop.

5.9.2.4 The pdta-list Chunk

5.9.2.4.1 The PHDR Subchunk

The PHDR subchunk is a required subchunk listing all presets within the SASBF bitstream element. It shall be a multiple of thirty-eight bytes in length, and shall contain a minimum of two records, one record for each preset and one for a terminal record according to the structure:

```
struct sfPresetHeader
{
       CHAR achPresetName[20];
       WORD wPreset;
       WORD wBank;
       WORD wPresetBagNdx;
       DWORD dwLibrary;
       DWORD dwGenre;
       DWORD dwMorphology;
};
```

The ASCII character field achPresetName shall contain the name of the preset expressed in ASCII, with unused terminal characters filled with zero valued bytes. Preset names are case-sensitive. A unique name shall always be assigned to each preset in the SASBF bank.

The WORD wPreset shall contain the MIDI Preset Number and the WORD wBank the MIDI Bank Number which apply to this preset. Note that the presets are not ordered within the SASBF bank. Presets

shall have a unique set of wPreset and wBank numbers. The special case of a General MIDI percussion bank is handled conventionally by a wBank value of 128. If the value in either field is not a valid MIDI value of 0 through 127, or 128 for wBank, the preset cannot be played but shall be maintained in memory for future renumbering.

The WORD wPresetBagNdx is an index to the preset's zone list in the PBAG subchunk. Because the preset zone list is in the same order as the preset header list, the preset bag indices shall be monotonically increasing with increasing preset headers. The size of the PBAG subchunk in bytes shall be equal to four times the terminal preset's wPresetBagNdx plus four. If the preset bag indices are non-monotonic or if the terminal preset's wPresetBagNdx does not match the PBAG subchunk size, the SASBF chunk is structurally defective and shall be rejected at load time. All presets except the terminal preset shall have at least one zone.

The DWORDs dwLibrary, dwGenre and dwMorphology are reserved for future implementation in a preset library management function and should be preserved as read, and created as zero.

The terminal sfPresetHeader record should never be accessed, and exists only to provide a terminal wPresetBagNdx with which to determine the number of zones in the last preset. All other values shall be conventionally zero, with the exception of achPresetName, which may be optionally be "EOP" indicating end of presets.

If the PHDR subchunk is missing, contains fewer than two records, or its size is not a multiple of 38 bytes, the SASBF bitstream element shall be rejected as structurally unsound.

5.9.2.4.2 The PBAG Subchunk

The PBAG subchunk is a required subchunk listing all preset zones within the SASBF bitstream element. It shall be a multiple of four bytes in length, and shall contain one record for each preset zone plus one record for a terminal zone according to the structure:

```
struct sfPresetBag
{
       WORD wGenNdx;
       WORD wModNdx;
};
```

The first zone in a given preset shall be located at that preset's wPresetBagNdx. The number of zones in the preset shall be determined by the difference between the next preset's wPresetBagNdx and the current wPresetBagNdx.

The WORD wGenNdx shall be an index to the preset's zone list of generators in the PGEN subchunk, and the wModNdx shall be an index to its list of modulators in the PMOD subchunk. Because both the generator and modulator lists are in the same order as the preset header and zone lists, these indices will be monotonically increasing with increasing preset zones. The size of the PMOD subchunk in bytes shall be equal to ten times the terminal preset's wModNdx plus ten and the size of the PGEN subchunk in bytes shall be equal to four times the terminal preset's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective PGEN or PMOD subchunks, the bitstream element is structurally defective and shall be rejected at load time.

If a preset has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not an Instrument generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks an Instrument generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the PBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.3 The PMOD Subchunk

The PMOD subchunk is a required subchunk listing all preset zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The preset zone's wModNdx points to the first modulator for that preset zone, and the number of modulators present for a preset zone is determined by the difference between the next higher preset zone's wModNdx and the current preset's wModNdx. A difference of zero indicates there are no modulators in this preset zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator with the same zone, the first modulator will be ignored.

Modulators in the PMOD subchunk act as additively relative modulators with respect to those in the IMOD subchunk. In other words, a PMOD modulator can increase or decrease the amount of an IMOD modulator.

In SASBF, no modulators have yet been defined, and the PMOD subchunk will always consist of ten zero valued bytes.

If the PMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.4 The PGEN Subchunk

The PGEN chunk is a required chunk containing a list of preset zone generators for each preset zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each preset zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined:

```
typedef struct
{
       BYTE byLo;
       BYTE byHi;
} rangesType;

typedef union
{
       rangesType ranges;
       SHORT shAmount;
       WORD wAmount;
} genAmountType;
```

The sfGenOper is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. This value indicates the type of generator being indicated. Note that this enumeration is two bytes in length.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers or MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The preset zone's wGenNdx points to the first generator for that preset zone. Unless the zone is a global zone, the last generator in the list is an "Instrument" generator, whose value is a pointer to the instrument associated with that zone. If a "key range" generator exists for the preset zone, it is always the first generator in the list for that preset zone. If a "velocity range" generator exists for the preset zone, it will only be preceded by a key range generator. If any generators follow an Instrument generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator with the same zone, the first generator will be ignored.

Generators in the PGEN subchunk are applied relative to generators in the IGEN subchunk in an additive manner. In other words, PGEN generators increase or decrease the value of an IGEN generator.

If the PGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in an instrument generator, the zone should be ignored. If the instrument generator value is equal to or greater than the terminal instrument, the bitstream element should be rejected as structurally unsound.

5.9.2.4.5 The INST Subchunk

The inst subchunk is a required subchunk listing all instruments within the SASBF bitstream element. It is always a multiple of twenty-two bytes in length, and contains a minimum of two records, one record for each instrument and one for a terminal record according to the structure:

```
struct sfInst
{
       CHAR achInstName[20];
       WORD wInstBagNdx;
};
```

The ASCII character field achInstName contains the name of the instrument expressed in ASCII, with unused terminal characters filled with zero valued bytes. Instrument names are case-sensitive. A unique name should always be assigned to each instrument in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of instruments with identical names, the instruments should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The WORD wInstBagNdx is an index to the instrument's zone list in the IBAG subchunk. Because the instrument zone list is in the same order as the instrument list, the instrument bag indices will be monotonically increasing with increasing instruments. The size of the IBAG subchunk in bytes will be four greater than four times the terminal (EOI) instrument's wInstBagNdx. If the instrument bag indices are non-monotonic or if the terminal instrument's wInstBagNdx does not match the IBAG subchunk size, the bitstream element is structurally defective and should be rejected at load time. All instruments except the terminal instrument must have at least one zone; any preset with no zones should be ignored.

The terminal sfInst record should never be accessed, and exists only to provide a terminal wInstBagNdx with which to determine the number of zones in the last instrument. All other values are conventionally zero, with the exception of achInstName, which should be "EOI" indicating end of instruments.

If the INST subchunk is missing, contains fewer than two records, or its size is not a multiple of 22 bytes, the bitstream element should be rejected as structurally unsound. All instruments present in the inst subchunk are typically referenced by a preset zone. However, a bitstream element containing any "orphaned" instruments need not be rejected. SASBF applications can optionally ignore or filter out these orphaned instruments based on user preference.

5.9.2.4.6 The IBAG Subchunk

The IBAG subchunk is a required subchunk listing all instrument zones within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one record for each instrument zone plus one record for a terminal zone according to the structure:

```
struct sfInstBag
{
       WORD wInstGenNdx;
       WORD wInstModNdx;
```

```
};
```

The first zone in a given instrument is located at that instrument's wInstBagNdx. The number of zones in the instrument is determined by the difference between the next instrument's wInstBagNdx and the current wInstBagNdx.

The WORD wInstGenNdx is an index to the instrument zone's list of generators in the IGEN subchunk, and the wInstModNdx is an index to its list of modulators in the IMOD subchunk. Because both the generator and modulator lists are in the same order as the instrument and zone lists, these indices will be monotonically increasing with increasing zones. The size of the IMOD subchunk in bytes will be equal to ten times the terminal instrument's wModNdx plus ten and the size of the IGEN subchunk in bytes will be equal to four times the terminal instrument's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective IGEN or IMOD subchunks, the bitstream element is structurally defective and should be rejected at load time.

If an instrument has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not a sampleID generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks a sampleID generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the IBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.7 The IMOD Subchunk

The IMOD subchunk is a required subchunk listing all instrument zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The zone's wInstModNdx points to the first modulator for that zone, and the number of modulators present for a zone is determined by the difference between the next higher zone's wInstModNdx and the current zone's wModNdx. A difference of zero indicates there are no modulators in this zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator within the same zone, the first modulator will be ignored.

Modulators in the IMOD subchunk are absolute. This means that an IMOD modulator replaces, rather than adds to, a default modulator.

In SASBF, no modulators have yet been defined, and the IMOD subchunk will always consist of ten zero valued bytes.

If the IMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.8 The IGEN Subchunk

The IGEN chunk is a required chunk containing a list of zone generators for each instrument zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfInstGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined as in the PGEN zone above.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers of MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The zone's wInstGenNdx points to the first generator for that zone. Unless the zone is a global zone, the last generator in the list is a "sampleID" generator, whose value is a pointer to the sample associated with that zone. If a "key range" generator exists for the zone, it is always the first generator in the list for that zone. If a "velocity range" generator exists for the zone, it will only be preceded by a key range generator. If any generators follow a sampleID generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator within the same zone, the first generator will be ignored.

Generators in the IGEN subchunk are absolute in nature. This means that an IGEN generator replaces, rather than adds to, the default value for the generator.

If the IGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in a sampleID generator, the zone should be ignored. If the sampleID generator value is equal to or greater than the terminal sampleID, the bitstream element should be rejected as structurally unsound.

5.9.2.4.9 The SHDR Subchunk

The SHDR chunk is a required subchunk listing all samples within the smpl subchunk and any referenced ROM samples. It is always a multiple of forty-six bytes in length, and contains one record for each sample plus a terminal record according to the structure:

```
struct sfSample
{
       CHAR achSampleName[20];
       DWORD dwStart;
       DWORD dwEnd;
       DWORD dwStartloop;
       DWORD dwEndloop;
       DWORD dwSampleRate;
       BYTE byOriginalPitch;
       CHAR chPitchCorrection;
       WORD wSampleLink;
       SFSampleLink sfSampleType;
};
```

The ASCII character field achSampleName contains the name of the sample expressed in ASCII, with unused terminal characters filled with zero valued bytes. Sample names are case-sensitive. A unique name should always be assigned to each sample in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of samples with identical names, the samples should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The DWORD dwStart contains the index, in sample data points, from the beginning of the sample data field to the first data point of this sample.

The DWORD dwEnd contains the index, in sample data points, from the beginning of the sample data field to the first of the set of 46 zero valued data points following this sample.

The DWORD dwStartloop contains the index, in sample data points, from the beginning of the sample data field to the first data point in the loop of this sample.

The DWORD dwEndloop contains the index, in sample data points, from the beginning of the sample data field to the first data point following the loop of this sample. Note that this is the data point “equivalent to” the first loop data point, and that to produce portable artefact free loops, the eight proximal data points surrounding both the Startloop and Endloop points should be identical.

The values of dwStart, dwEnd, dwStartloop, and dwEndloop must all be within the range of the sample data field included in the SASBF bank or referenced in the sound ROM. Also, to allow a variety of hardware platforms to be able to reproduce the data, the samples have a minimum length of 48 data points, a

minimum loop size of 32 data points and a minimum of 8 valid points prior to dwStartloop and after dwEndloop. Thus dwStart must be less than dwStartloop-7, dwStartloop must be less than dwEndloop-31, and dwEndloop must be less than dwEnd-7. If these constraints are not met, the sound may optionally not be played if the hardware cannot support artefact-free playback for the parameters given.

The DWORD dwSampleRate contains the sample rate, in hertz, at which this sample was acquired or to which it was most recently converted. Values of greater than 50000 or less than 400 may not be reproducible by some hardware platforms and should be avoided. A value of zero is illegal. If an illegal or impractical value is encountered, the nearest practical value should be used.

The BYTE byOriginalPitch contains the MIDI key number of the recorded pitch of the sample. For example, a recording of an instrument playing middle C (261.62 Hz) should receive a value of 60. This value is used as the default “root key” for the sample, so that in the example, a MIDI key-on command for note number 60 would reproduce the sound at its original pitch. For unpitched sounds, a conventional value of 255 should be used. Values between 128 and 254 are illegal. Whenever an illegal value or a value of 255 is encountered, the value 60 should be used.

The CHAR chPitchCorrection contains a pitch correction in cents which should be applied to the sample on playback. The purpose of this field is to compensate for any pitch errors during the sample recording process. The correction value is that of the correction to be applied. For example, if the sound is 4 cents sharp, a correction bringing it 4 cents flat is required; thus the value should be -4.

The value in sfSampleType is an enumeration with eight defined values: monoSample = 1, rightSample = 2, leftSample = 4, linkedSample = 8, RomMonoSample = 32769, RomRightSample = 32770, RomLeftSample = 32772, and RomLinkedSample = 32776. It can be seen that this is encoded such that bit 15 of the 16 bit value is set if the sample is in ROM, and reset if it is included in the SASBF bank. The four LS bits of the word are then exclusively set indicating mono, left, right, or linked.

If the sound is flagged as a ROM sample and no valid “irom” subchunk is included; the bitstream element is structurally defective and should be rejected at load time.

If sfSampleType indicates a mono sample, then wSampleLink is undefined and its value should be conventionally zero, but will be ignored regardless of value. If sfSampleType indicates a left or right sample, then wSampleLink is the sample header index of the associated right or left stereo sample respectively. Both samples should be played entirely synchronously, with their pitch controlled by the right sample’s generators. All non-pitch generators should apply as normal; in particular the panning of the individual samples to left and right should be accomplished via the pan generator. Left-right pairs should always be found within the same instrument. Note also that no instrument should be designed in which it is possible to activate more than one instance of a particular stereo pair. The linked sample type is not currently fully defined in the SASBF specification, but will ultimately support a circularly linked list of samples using wSampleLink. Note that this enumeration is two bytes in length.

The terminal sample record is never referenced, and is conventionally entirely zero with the exception of achSampleName, which should be “EOS” indicating end of samples. All samples present in the smpl subchunk are typically referenced by an instrument, however a bitstream element containing any “orphaned” samples need not be rejected. SASBF applications can optionally ignore or filter out these orphaned samples according to user preference.

If the SHDR subchunk is missing, or its is size is not a multiple of 46 bytes the bitstream element should be rejected as structurally unsound.

5.9.3 Enumerators

5.9.3.1 Generator Enumerators

Subclause 7.1 defines the generator and generator kinds. Subclause 8.4 defines the generator operation model.

5.9.3.1.1 Kinds of Generator Enumerators

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

An Index Generator's amount is an index into another data structure. The only two Index Generators are Instrument and sampleID.

A Range Generator defines a range of note-on parameters outside of which the zone is undefined. Two Range Generators are currently defined, keyRange and velRange.

Substitution Generators are generators which substitute a value for a note-on parameter. Two Substitution Generators are currently defined, overridingKeyNumber and overridingVelocity.

Sample Generators are generators which directly affect a sample's properties. These generators are undefined at the preset level. The currently defined Sample Generators are the eight address offset generators, the sampleModes generator, the Overriding Root Key generator and the Exclusive Class generator.

Value Generators are generators whose value directly affects a signal processing parameter. Most generators are value generators.

5.9.3.1.2 Generator Enumerators Defined

The following is an exhaustive list of SASBF generators and their strict definitions:

0	startAddrsOffset	The offset, in sample data points, beyond the Start sample header parameter to the first sample data point to be played for this instrument. For example, if Start were 7 and startAddrsOffset were 2, the first sample data point played would be sample data point 9.
1	endAddrsOffset	The offset, in sample data points, beyond the End sample header parameter to the last sample data point to be played for this instrument. For example, if End were 17 and endAddrsOffset were -2, the last sample data point played would be sample data point 15.
2	startloopAddrsOffset	The offset, in sample data points, beyond the Startloop sample header parameter to the first sample data point to be repeated in the loop for this instrument. For example, if Startloop were 10 and startloopAddrsOffset were -1, the first repeated loop sample data point would be sample data point 9.
3	endloopAddrsOffset	The offset, in sample data points, beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. For example, if Endloop were 15 and endloopAddrsOffset were 2, sample data point 17 would be considered equivalent to the

		Startloop sample data point, and hence sample data point 16 would effectively precede Startloop during looping.
4	startAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Start sample header parameter and the first sample data point to be played in this instrument. This parameter is added to the startAddrsOffset parameter. For example, if Start were 5, startAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point played would be sample data point 65544.
5	modLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
6	vibLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Vibrato LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
7	modEnvToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence pitch. A positive value indicates an increase in pitch; a negative value indicates a decrease in pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will rise 1 semitone at the envelope peak.
8	initialFilterFc	This is the cutoff and resonant frequency of the lowpass filter in absolute cent units. The lowpass filter is defined as a second order resonant pole pair whose pole frequency in Hz is defined by the Initial Filter Cutoff parameter. When the cutoff frequency exceeds 20kHz and the Q (resonance) of the filter is zero, the filter does not affect the signal.
9	initialFilterQ	This is the height above DC gain in centibels which the filter resonance exhibits at the cutoff frequency. A value of zero or less indicates the filter is not resonant; the gain at the cutoff frequency (pole angle) may be less than zero when zero is specified. The filter gain at DC is also affected by this parameter such that the gain at DC is reduced by half the specified gain. For example, for a value of 100, the filter gain at DC would be 5 dB below unity gain, and the height of the resonant peak would be 10 dB above the DC gain, or 5 dB above unity gain. Note also that if initialFilterQ is set to zero or less and the cutoff frequency exceeds 20 kHz, then the filter response is flat and unity gain.

10	modLfoToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence filter cutoff frequency. A positive number indicates a positive LFO excursion increases cutoff frequency; a negative number indicates a positive excursion decreases cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1200 indicates that the cutoff frequency will first rise 1 octave, then fall one octave.
11	modEnvToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence filter cutoff frequency. A positive number indicates an increase in cutoff frequency; a negative number indicates a decrease in filter cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1000 indicates that the cutoff frequency will rise one octave at the envelope attack peak.
12	endAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the End sample header parameter and the last sample data point to be played in this instrument. This parameter is added to the endAddrsOffset parameter. For example, if End were 65536, startAddrsOffset were -3 and startAddrsCoarseOffset were -1, the last sample data point played would be sample data point 32765.
13	modLfoToVolume	This is the degree, in centibels, to which a full scale excursion of the Modulation LFO will influence volume. A positive number indicates a positive LFO excursion increases volume; a negative number indicates a positive excursion decreases volume. Volume is always modified logarithmically, that is the deviation is in decibels rather than in linear amplitude. For example, a value of 100 indicates that the volume will first rise ten dB, then fall ten dB.
14	unused1	Unused, reserved. Should be ignored if encountered.
15	chorusEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the chorus effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the chorus effects processor.
16	reverbEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the reverb effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the reverb effects processor.

17	pan	This is the degree, in 0.1% units, to which the “dry” audio output of the note is positioned to the left or right output. A value of -50% or less indicates the signal is sent entirely to the left output and not sent to the right output; a value of +50% or more indicates the signal is sent entirely to the right and not sent to the left. A value of zero sends the signal equally to left and right. For example, a value of -250 indicates that the signal is sent at 75% of full level to the left output and 25% of full level to the right output.
18	unused2	Unused, reserved. Should be ignored if encountered.
19	unused3	Unused, reserved. Should be ignored if encountered.
20	unused4	Unused, reserved. Should be ignored if encountered.
21	delayModLFO	This is the delay time, in absolute timecents, from key on until the Modulation LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second and a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
22	freqModLFO	This is the frequency, in absolute cents, of the Modulation LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 MHz would be 1200log2(.01/8.176) = -11610.
23	delayVibLFO	This is the delay time, in absolute timecents, from key on until the Vibrato LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
24	freqVibLFO	This is the frequency, in absolute cents, of the Vibrato LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 mHz would be 1200log2(.01/8.176) = -11610.
25	delayModEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Modulation envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
26	attackModEnv	This is the time, in absolute timecents, from the end of the Modulation Envelope Delay Time until the point at which the Modulation Envelope value reaches its peak. Note that the attack is

		"convex"; the curve is nominally such that when applied to a decibel or semitone parameter, the result is linear in amplitude or Hz respectively. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
27	holdModEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
28	decayModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during decay phase. For the Modulation Envelope, the decay phase linearly ramps toward the sustain level. If the sustain level were zero, the Modulation Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
29	sustainModEnv	This is the decrease in level, expressed in 0.1% units, to which the Modulation Envelope value ramps during the decay phase. For the Modulation Envelope, the sustain level is properly expressed in percent of full scale. Because the volume envelope sustain level is expressed as an attenuation from full scale, the sustain level is analogously expressed as a decrease from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; values above 1000 are to be interpreted as 1000. For example, a sustain level which corresponds to an absolute value 40% of peak would be 600.
30	releaseModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during release phase. For the Modulation Envelope, the release phase linearly ramps toward zero from the current level. If the current level were full scale, the Modulation Envelope Release Time would be the time spent in release phase until zero value were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
31	keynumToModEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave

		causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
32	keynumToModEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
33	delayVolEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Volume envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
34	attackVolEnv	This is the time, in absolute timecents, from the end of the Volume Envelope Delay Time until the point at which the Volume Envelope value reaches its peak. Note that the attack is “convex”; the curve is nominally such that when applied to the decibel volume parameter, the result is linear in amplitude. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
35	holdVolEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the Volume envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
36	decayVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during decay phase. For the Volume Envelope, the decay phase linearly ramps toward the sustain level, causing a constant dB change for each time unit. If the sustain level were -96dB, the Volume Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
37	sustainVolEnv	This is the decrease in level, expressed in centibels, to which the Volume Envelope value ramps during the decay phase. For the

		Volume Envelope, the sustain level is best expressed in centibels of attenuation from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; conventionally 1000 indicates full attenuation. For example, a sustain level which corresponds to an absolute value 12dB below of peak would be 120.
38	releaseVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during release phase. For the Volume Envelope, the release phase linearly ramps toward zero from the current level, causing a constant dB change for each time unit. If the current level were full scale, the Volume Envelope Release Time would be the time spent in release phase until 96dB attenuation were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
39	keynumToVolEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
40	keynumToVolEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
41	instrument	This is the index into the INST subchunk providing the instrument to be used for the current preset zone. A value of zero indicates the first instrument in the list. The value should never exceed two less than the size of the instrument list. The instrument enumerator is the terminal generator for PGEN zones. As such, it should only appear in the PGEN subchunk, and it must appear as the last generator enumerator in all but the global preset zone.
42	reserved1	Unused, reserved. Should be ignored if encountered.
43	keyRange	This is the minimum and maximum MIDI key number values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid key. The

		keyRange enumerator is optional, but when it does appear, it must be the first generator in the zone generator list.
44	velRange	This is the minimum and maximum MIDI velocity values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid velocity. The velRange enumerator is optional, but when it does appear, it must be preceded only by keyRange in the zone generator list.
45	startloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Startloop sample header parameter and the first sample data point to be repeated in this instrument's loop. This parameter is added to the startloopAddrsOffset parameter. For example, if Startloop were 5, startloopAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point in the loop would be sample data point 65544.
46	keynum	This enumerator forces the MIDI key number to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
47	velocity	This enumerator forces the MIDI velocity to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
48	initialAttenuation	This is the attenuation, in .4 centibel units, by which a note is attenuated below full scale. A value of zero indicates no attenuation; the note will be played at full scale. For example, a value of 60 indicates the note will be played at 2.4 dB below full scale for the note.
49	reserved2	Unused, reserved. Should be ignored if encountered.
50	endloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. This parameter is added to the endloopAddrsOffset parameter. For example, if Endloop were 5, endloopAddrsOffset were 3 and endAddrsCoarseOffset were 2, sample data point 65544 would be considered equivalent to the Startloop sample data point, and hence sample data point 65543 would effectively precede Startloop during looping.
51	coarseTune	This is a pitch offset, in semitones, which should be applied to the note. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Coarse Tune value of -4 would cause the sound to be reproduced four semitones flat.
52	fineTune	This is a pitch offset, in cents, which should be applied to the note. It is additive with coarseTune. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Fine Tuning value of -5 would cause the sound to be reproduced five cents flat.

53	sampleID	This is the index into the SHDR subchunk providing the sample to be used for the current instrument zone. A value of zero indicates the first sample in the list. The value should never exceed two less than the size of the sample list. The sampleID enumerator is the terminal generator for IGEN zones. As such, it should only appear in the IGEN subchunk, and it must appear as the last generator enumerator in all but the global zone.
54	sampleModes	This enumerator indicates a value which gives a variety of Boolean flags describing the sample for the current instrument zone. The sampleModes should only appear in the IGEN subchunk, and should not appear in the global zone. The two LS bits of the value indicate the type of loop in the sample: 0 indicates a sound reproduced with no loop, 1 indicates a sound which loops continuously, 2 is unused but should be interpreted as indicating no loop, and 3 indicates a sound which loops for the duration of key depression then proceeds to play the remainder of the sample.
55	reserved3	Unused, reserved. Should be ignored if encountered.
56	scaleTuning	This parameter represents the degree to which MIDI key number influences pitch. A value of zero indicates that MIDI key number has no effect on pitch; a value of 100 represents the usual tempered semitone scale.
57	exclusiveClass	This parameter provides the capability for a key depression in a given instrument to terminate the playback of other instruments. This is particularly useful for percussive instruments such as a hi-hat cymbal. An exclusive class value of zero indicates no exclusive class; no special action is taken. Any other value indicates that when this note is initiated, any other sounding note with the same exclusive class value should be rapidly terminated. The exclusive class generator can only appear at the instrument level. The scope of the exclusive class is the entire preset. In other words, any other instrument zone within the same preset holding a corresponding exclusive class will be terminated.
58	overridingRootKey	This parameter represents the MIDI key number at which the sample is to be played back at its original sample rate. If not present, or if present with a value of -1, then the sample header parameter Original Key is used in its place. If it is present in the range 0-127, then the indicated key number will cause the sample to be played back at its sample header Sample Rate. For example, if the sample were a recording of a piano middle C (Original Key = 60) at a sample rate of 22.050 kHz, and Root Key were set to 69, then playing MIDI key number 69 (A above middle C) would cause a piano note of pitch middle C to be heard.
59	unused5	Unused, reserved. Should be ignored if encountered.
60	endOper	Unused, reserved. Should be ignored if encountered. Unique name provides value to end of defined list.

5.9.3.1.3 Generator Summary

The following tables give the ranges and default values for all SASBF defined generators.

#	Name	Unit	Abs Zero	Min	Min Useful	Max	Max Useful	De-fault	Def Value
0	startAddrsOffset +	smpls	0	0	None	*	*	0	None
1	endAddrsOffset +	smpls	0	*	*	0	None	0	None
2	startloopAddrsOffset +	smpls	0	*	*	*	*	0	None
3	endloopAddrsOffset +	smpls	0	*	*	*	*	0	None
4	startAddrsCoarseOffset +	32k	0	0	None	*	*	0	None
5	modLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
6	vibLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
7	modEnvToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
8	initialFilterFc	cent	8.176 Hz	1500	20 Hz	13500	20 kHz	13500	Open
9	initialFilterQ	cB	0	0	None	960	96 dB	0	None
10	modLfoToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
11	modEnvToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
12	endAddrsCoarseOffset +	32k	0	*	*	0	None	0	None
13	modLfoToVolume	cB fs	0	-960	-96 dB	960	96 dB	0	None
15	chorusEffectsSend	0.1%	0	0	None	1000	100%	0	None
16	reverbEffectsSend	0.1%	0	0	None	1000	100%	0	None
17	pan	0.1%	Cntr	-500	Left	+500	Right	0	Centre
21	delayModLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
22	freqModLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
23	delayVibLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
24	freqVibLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
25	delayModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
26	attackModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
27	holdModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
28	decayModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
29	sustainModEnv	-0.1%	attk peak	0	100%	1000	0%	0	attk pk
30	releaseModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
31	keynumToModEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
32	keynumToModEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
33	delayVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
34	attackVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
35	holdVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
36	decayVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
37	sustainVolEnv	cB attn	attk peak	0	0 dB	1440	144dB	0	attk pk
38	releaseVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
39	keynumToVolEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
40	keynumToVolEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
43	keyRange	MIDI ky#	key# 0	0	lo key	127	hi key	0-127	full kbd
44	velRange	MIDI vel	0	0	min vel	127	max vel	0-127	all vels
45	startloopAddrsCoarseOffset +	smpl	0	*	*	*	*	0	None

46	keynum +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None
47	velocity +	MIDI vel	0	1	min vel	127	max vel	-1	None
48	initialAttenuation	.4 cB	0	0	0 dB	1440	144dB	0	None
50	endloopAddrsCoarseOffset	smpls	0	*	*	*	*	0	None
51	CoarseTune	semitone	0	-120	-10 oct	120	10 oct	0	None
52	fineTune	cent	0	-99	-99cnt	99	99cent	0	None
54	sampleModes +	Bit Flags	Flags	**	**	**	**	0	No Loop
56	scaleTuning	cent/key	0	0	none	1200	oct/ky	100	semi-tone
57	exclusiveClass +	arbitrary #	0	1	--	127	--	0	None
58	overridingRootKey +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None

* Range depends on values of start, loop, and end points in sample header.

** Range has discrete values based on bit flags

\+ This generator is only valid at the instrument level.

5.9.3.2 Default Modulators

The “default” modulators are described below.

5.9.3.2.1 MIDI Key Velocity to Initial Attenuation

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.2 MIDI Key Velocity to Filter Cutoff

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is -2400 Cents.

The product of these values is added to the Initial Filter Cutoff generator summing node.

5.9.3.2.3 MIDI Channel Pressure to Vibrato LFO Pitch Depth

The MIDI Channel Pressure data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 50 cents per max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.4 MIDI Continuous Controller 1 to Vibrato LFO Pitch Depth

The MIDI Continuous Controller 1 data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. The MIDI Continuous Controller 33 data value may be optionally used for increased resolution of the controller input.

There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 50 cents/max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.5 MIDI Continuous Controller 7 to Initial Attenuation

The MIDI Continuous Controller 7 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.6 MIDI Continuous Controller 10 to Pan Position

The MIDI Continuous Controller 10 data value is used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 127 is mapped to 63/64 and all other values are mapped between -1 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 1000 tenths of a percent panned-right.

The product of these values is added to the Pan generator summing node.

5.9.3.2.7 MIDI Continuous Controller 11 to Initial Attenuation

The MIDI Continuous Controller 11 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.8 MIDI Continuous Controller 91 to Reverb Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added reverb send.

The product of these values is added to the Reverb Send generator summing node.

5.9.3.2.9 MIDI Continuous Controller 93 to Chorus Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added chorus send.

The product of these values is added to the Chorus Send generator summing node.

5.9.3.2.10 MIDI Pitch Wheel to Initial Pitch Controlled by MIDI Pitch Wheel Sensitivity

The MIDI Pitch Wheel data values are used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 16383 is mapped to 8191/8192 and all other values are mapped between -1 and 8191/8192 in a linear fashion.

The MIDI Pitch Wheel Sensitivity data values are used as a secondary source. This source is Positive Unipolar, thus an input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion.

The amount of this modulator is 12700 Cents.

The product of these values is added to the Initial Pitch generator summing node.

5.9.3.3 Precedence and Absolute and Relative values.

Most SASBF generators are available at both the Instrument and Preset Levels, as well as having a default value. Generators at the Instrument Level are considered “absolute” and determine an actual physical value for the associated synthesis parameter, which is used instead of the default. For example, a value of 1200 for the attackVolEnv generator would produce an absolute time of 1200 timecents or 2 seconds of attack time for the volume envelope, instead of the default value of -12000 timecents or 1 msec.

Generators at the Preset Level are instead considered “relative” and additive to all the default or instrument level generators within the Preset Zone. For example, a value of 2400 timecents for the attackVolEnv generator in a preset zone containing an instrument with two zones, one with the default attackVolEnv and one with an absolute attackVolEnv generator value of 1200 timecents would cause the default zone to actually have a value of -9600 timecents or 4 msec, and the other to have a value of 3600 timecents or 8 seconds attack time.

There are some generators which are not available at the Preset Level. These are:

#	Name
0	startAddrsOffset
1	endAddrsOffset
2	startloopAddrsOffset
3	endloopAddrsOffset
4	startAddrsCoarseOffset
12	endAddrsCoarseOffset
45	startloopAddrsCoarseOffset
46	keynum
47	velocity
50	endloopAddrsCoarseOffset
54	sampleModes

57	exclusiveClass
58	overridingRootKey

If these generators are encountered in the Preset Level, they should be ignored.

The effect of modulators on a given destination is always relative to the generator value at the Instrument level. However modulators may supersede or add to other modulators depending on their position within the hierarchy. Please see Subclause 8.4 for details on the Modulator implementation and the hierarchical details.

5.9.4 Parameters and Synthesis Model

The SASBF standard has been established with the intent of providing support for an expanding base of wavetable based synthesis models. The model supported by the SASBF specification originates with the EMU8000 wavetable synthesiser chip. The description below of the underlying synthesis model and the associated parameters are provided to allow mapping of this synthesis model onto other hardware platforms.

5.9.4.1 Synthesis Model

The SASBF specification Synthesis Model comprises a wavetable oscillator, a dynamic lowpass filter, an enveloping amplifier, and programmable sends to pan, reverb, and chorus effects units. An underlying modulation engine comprises two low frequency oscillators (LFOs) and two envelope generators with appropriate routing amplifiers.

5.9.4.1.1 Wavetable Oscillator/Interpolator

The SASBF specification wavetable oscillator model is capable of playing back a sample at an arbitrary sampling rate with an arbitrary pitch shift. In practice, the upward pitch shift (downward sample rate conversion) will be limited to a maximum value, typically at least two octaves. The pitch is described in terms of an initial pitch shift which is based on the sample's sampling rate, the root key at which the sample should be unshifted on the keyboard, the coarse, fine, and correction tunings, the effective MIDI key number, and the keyboard scale factor. All modulations in pitch are in octaves, semitones, and cents.

In general, interpolators have a symmetric impulse response, and thus a linear phase characteristic. The interpolation filter's magnitude response can be characterised by three regions - the pass band, the transition band, and the stop band.

The interpolation filter's pass band is the portion of its response which corresponds to the frequencies of the signal stored in waveform memory from DC to that signal's Nyquist frequency

The interpolation filter's transition band is the portion of its response which corresponds to the first image of the signal stored in waveform memory, that is from that signal's Nyquist frequency back to DC.

The interpolation filter's stop band is the portion of its response which corresponds to the remaining images above the transition band to the Nyquist frequency of the upsampled signal.

The guardband will be assumed to begin at 5/6 of the Nyquist frequency, as is the case for a 20 kHz bandwidth in a 48 kHz sample rate system.

Due to poor aliasing rejection in the stopband, linear interpolation does not satisfy this specification.

The specification constrains the Fourier transform of the impulse response of the interpolator. The impulse response is taken with a downward pitch shift of eight octaves. This gives a response which has been upsampled by a factor of 2**8 or 256. In other words, a frequency of the impulse response's Nyquist

frequency divided by 256 gives the filter frequency corresponding to the waveform memory's Nyquist frequency. In the specification below, Fn represents this waveform memory Nyquist frequency.

(All responses measured with downward pitch shift of 8 octaves.)	
Passband Response:	
Ripple:	No more than +/- 0.5 dB
Roll-off:	Minimal, but not to exceed 6 dB at 83.3% Fn.
Transition Band Response:	
Roll-off:	Monotonic and as rapid as possible to at least -80 dB attenuation
Return Lobes:	None above -80 dB
Attenuation:	Greater than 80 dB for all frequencies DC to at least 2% Fn in the first lobe.
Stop Band Response:	
Attenuation:	Greater than 90 dB for all frequencies DC to at least 1% Fn
	Greater than 80 dB for all frequencies DC to at least 20%Fn
	Greater than 60 dB for all frequencies

5.9.4.1.2 Sample Looping

The wavetable oscillator is playing a digital sample which is described in terms of a start point, end point, and two points describing a loop. The sound can be flagged as unlooped, in which case the loop points are ignored. If the sound is looped, it can be played in two ways. If it is flagged as "loop during release", the sound is played from the start point through the loop, and loops until the note becomes inaudible. If not, the sound is played from the start point through the loop, and loops until the key is released. At this point, the next time the loop end point is reached, the sound continues through the loop end point and plays until the end point is reached, at which time audio is terminated.

5.9.4.1.3 Lowpass Filter

The synthesis model contains a resonant lowpass filter, which is characterised by a dynamic cutoff frequency and a fixed resonance (Q).

The filter is idealised at zero resonance as having a flat passband to the cutoff frequency, then a rolloff at 12 dB per octave above that frequency. The resonance, when non-zero, comprises a peak at the cutoff frequency, superimposed on the above response. The resonance is measured as a dB ratio of the resonant peak to the DC gain. The DC gain at any resonance is half of the resonance value below the DC gain at zero resonance; hence the peak height is half the resonance value above DC gain at zero resonance.

All modulations in cutoff frequency are in cents.

Topology: 2nd order AR lowpass filter or equivalent. Filter coefficients are updated in a smooth manner at the sample rate.

Noise and Distortion: Less than 0.003% of full scale on any sinusoidal input for any parameter setting.

Control Parameters: Cutoff Frequency and Resonance.

Resonance Parameter: Specifies the ratio in decibels of the gain of a resonant peak to the gain at DC, when the filter cutoff frequency is set to 1.5 kHz and the modulation is zero. The DC gain of a filter is reduced from the DC gain of a filter with zero resonance by half the resonance value. The resonance parameter will remain fixed throughout the sounding of a musical note.

For a specified resonance, the actual gain ratio should be accurate within +/- 1 dB, and the DC gain accurate within +/-0.5 dB. The ratio of the gain at the resonant peak to the DC gain for all cutoff frequency values shall not deviate by more than 2dB per octave deviation from 1.5 kHz. Nominal gain at DC for a minimum resonance parameter is unity gain.

Cutoff Frequency Parameter: Specifies the frequency at which the filter achieves exactly 3dB of attenuation. The Cutoff Frequency is computed by the combination of the Initial Cutoff Frequency, specified in Hz, and the modulation, specified in semitones and fractions thereof. The Initial Cutoff Frequency is fixed throughout the duration of the a musical note; the modulation can vary at the sample rate. Within the region from 200 Hz to 6.4 kHz, the cutoff frequency parameter accuracy will be +/- 2 semitones.

Frequency Magnitude Response: When the cutoff frequency is at its maximum value and the resonance is at zero, the filter must pass audio without alteration. The filter must support cutoff frequencies down to 200 Hz. There must be a minimum of 2048 distinct cutoff frequencies per octave within the region from 200 Hz to 6.4 kHz, with a worst case spacing between distinct frequencies of 0.01 semitones.

5.9.4.1.4 Final Gain Amplifier

The final gain amplifier is a multiplier on the filter output, which is controlled by an initial gain in dB. This is added to the volume envelope. Additional modulation can also be added. The gain is always specified in dB.

5.9.4.1.5 Effects Sends

The output of the final gain amplifier can be routed into the effects unit. This unit causes the sound to be located (panned) in the stereo field, and a degree of reverberation and chorus to be added. The pan is specified in terms of percentage left and right, which also could be considered as an azimuth angle. The reverb and chorus sends are specified as a percentage of the signal amplitude to be sent to these units, from 0% to 100%.

5.9.4.1.6 Low Frequency Oscillators

The synthesis model provides for two low frequency oscillators (LFOs) for modulating pitch, filter cutoff, and amplitude. The “vibrato” LFO is only capable of modulating pitch. The “modulation” LFO can modulate any of the three parameters.

An LFO is defined as having a delay period during which its value remains zero, followed by a triangular waveshape ramping linearly to positive one, then downward to negative 1, then upward again to positive one, etc.

Each parameter can be modulated to a varying degree, either positively or negatively, by the associated LFO. Modulations of pitch and cutoff are in octaves, semitones, and cents, while modulations of amplitude are in dB. The degree of modulation is specified in cents or dB for the full scale positive LFO excursion.

5.9.4.1.7 Envelope Generators

The synthesis model provides for two envelope generators. The volume envelope generator controls the final gain amplifier and hence determines the volume contour of the sound. The modulation envelope can control pitch and/or filter cutoff.

An envelope generates a control signal in six phases. When key-on occurs, a delay period begins during which the envelope value is zero. The envelope then rises in a convex curve to a value of one during the attack phase. When a value of one is reached, the envelope enters a hold phase during which it remains at one. When the hold phase ends, the envelope enters a decay phase during which its value decreases linearly to a sustain level. When the sustain level is reached, the envelope enters sustain phase, during which the envelope stays at the sustain level. Whenever a key-off occurs, the envelope immediately enters a release phase during which the value linearly ramps from the current value to zero. When zero is reached, the envelope value remains at zero.

Modulation of pitch and filter cutoff are in octaves, semitones, and cents. These parameters can be modulated to varying degree, either positively or negatively, by the modulation envelope. The degree of modulation is specified in cents for the full scale attack peak.

The volume envelope operates in dB, with the attack peak providing a full scale output, appropriately scaled by the initial volume. The zero value, however, is actually zero gain. When 96 dB of attenuation is reached in the final gain amplifier, an abrupt jump to zero gain (infinite dB of attenuation) occurs. In a 16-bit system, this jump is inaudible.

5.9.4.1.8 Modulation Interconnection Summary

The following diagram shows the interconnections expressed in the SASBF specification synthesis model:

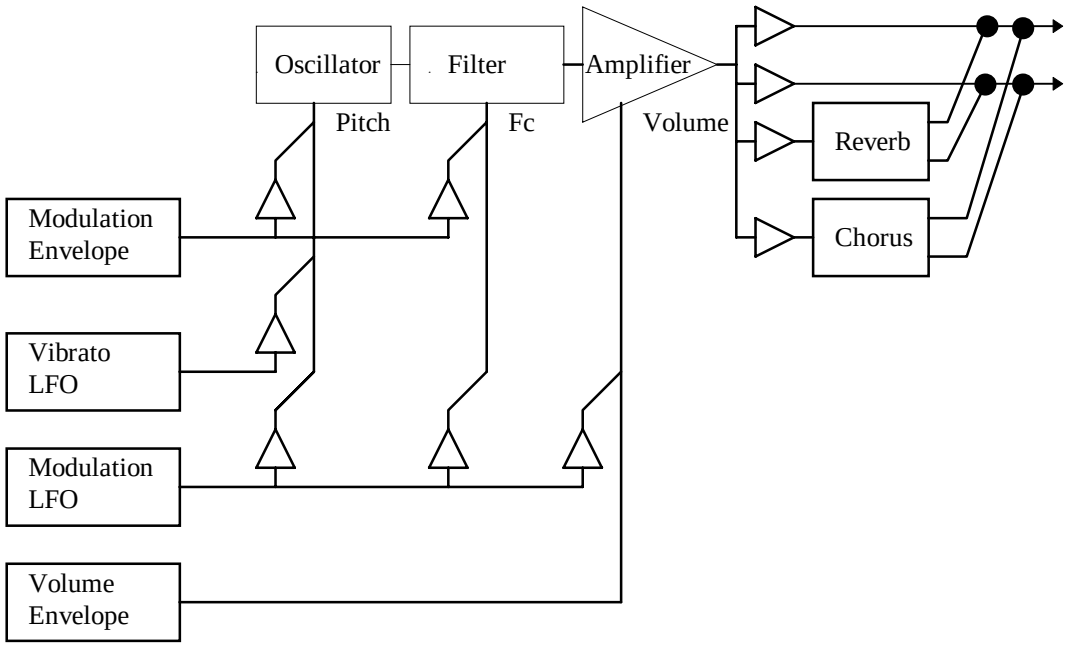


Figure 1: Generator Based Modulation Structure

5.9.4.2 MIDI Functions

The response to certain MIDI commands is defined within the MIDI specification, and therefore not considered to be part of the SASBF specification. These MIDI commands may not be used as sources for the Modulator implementation.

For completeness, the expected responses are given here.

MIDI CC0 Bank Select - When received, the following program change should select the MIDI program in this bank value instead of the default bank of 0.

MIDI CC6 - Data Entry MSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism depending on the Data Entry mode.

MIDI CC32 Bank Select LSB - When received, may behave in conjunction with CC0 Bank Select to provide a total of 16384 possible MIDI banks of programs.

MIDI CC38 Data Entry LSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism, depending on the Data Entry mode.

MIDI CC64 Sustain - ACTIVE when greater than or equal to 64. When the sustain function is active, all notes in the key-on state remain in the key-on state regardless of whether a key-off command for the note arrives. The key-off commands are stored, and when sustain becomes inactive, all stored key-off commands are executed.

MIDI CC66 Soft - ACTIVE when greater than or equal to 64. When active, all new key-ons are modulated in such a way to make the note sound “soft.” This typically affects initial attenuation and filter cutoff in a pre-defined manner.

MIDI CC67 Sostenuto - ACTIVE when greater than or equal to 64. When sostenuto becomes active, all notes currently in the key-on state remain in the key-on state until the sostenuto becomes inactive. All other notes behave normally. Notes maintained by sostenuto in key-on state remain in key-on state even if sustain is switched on and off.

MIDI CC98 NRPN LSB - When received, should be processed by the NRPN implementation mechanism.

MIDI CC99 NRPN MSB - When received, should put the synthesiser in NRPN Data Entry mode and then should be processed by the NRPN implementation mechanism.

MIDI CC100 RPN LSB - When received, should be processed by the RPN implementation mechanism.

MIDI CC101 RPN MSB - When received, should put the synthesiser in RPN Data Entry mode and then should be processed by the RPN implementation mechanism.

MIDI CC120 All Sound Off - When received with any data value, all notes playing in the key-on state bypass the release phase and are shut off, regardless of the sustain or sostenuto positions.

MIDI CC121 Reset All Controllers - Defined as Reset All Controllers as defined by the MIDI specification.

MIDI CC123 All Notes Off - When received with any data value, all notes playing in the key-on state immediately enter release phase, pending their status in SUSTAIN or SOSTENUTO state.

5.9.4.3 Parameter Units

The units with which SASBF generators are described are all well defined. The strict definitions appear below:

ABSOLUTE SAMPLE DATA POINTS - A numeric index of 16 bit sample data point words as stored in ROM or supplied in the smpl-ck, indexing the first sample data point word of memory or the chunk as zero.

RELATIVE SAMPLE DATA POINTS - A count of 16 bit sample data point words based on an absolute sample data point reference. A negative value implies a relative count toward the beginning of the data.

ABSOLUTE SEMITONES - An absolute logarithmic measure of frequency based on a reference of MIDI key numbers. A semitone is 1/12 of an octave, and value 69 is 440 Hz (A-440). Negative values and values above 127 are allowed.

RELATIVE SEMITONES - A relative logarithmic measure of frequency ratio based on units of 1/12 of an octave, which is the twelfth root of two, approximately 1.059463094.

ABSOLUTE CENTS - An absolute logarithmic measure of frequency based on a reference of MIDI key number scaled by 100. A cent is 1/1200 of an octave, and value 6900 is 440 Hz (A-440). Negative values and values above 12700 are allowed.

RELATIVE CENTS - A relative logarithmic measure of frequency ratio based on units of 1/1200 of an octave, which is the twelve hundredth root of two, approximately 1.000577790.

ABSOLUTE CENTIBELS - An absolute measure of the attenuation of a signal, based on a reference of zero being no attenuation. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

RELATIVE CENTIBELS - A relative measure of the attenuation of a signal. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

ABSOLUTE TIMECENTS - An absolute measure of time, based on a reference of zero being one second. A timecent represents a ratio in time of the twelve hundredth root of two, approximately 1.011579454.

RELATIVE TIMECENTS - A relative measure of time ratio, based on a unit size of the twelve hundredth root of two, approximately 1.011579454.

ABSOLUTE PERCENT - An absolute measure of gain, based on a reference of unity. In SASBF, absolute percent is measured in 0.1% units, so a value of zero is 0% and a value of 1000 is 100%.

RELATIVE PERCENT - A relative measure of gain difference. In SASBF, relative percent is measured in 0.1% units. When the gain goes below zero, zero is assumed; when the gain exceeds 100%, 100% is used.

5.9.4.4 The SASBF Generator Model

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

In case it is not clear in the general description of the SASBF hierarchy, the following is the precedence of SASBF generator in the hierarchy.

- A ‘generator’ sets or offsets the value of a destination or a synthesis parameter. In exception cases, it sets ranges (Range Generators), or sets values and never offsets values (Index Generators, Sample Generators, and Substitution Generators).
- A generator is defined as identical to another generator if its generator operator is the same in both generators.
- A generator in a global instrument zone which is identical to a default generator supersedes or replaces the default generator.
- A generator in a local instrument zone which is identical to a default generator or to a generator in a global instrument zone supersedes or replaces that generator.
- Points below (until noted) apply to Value Generators ONLY.
- A generator at the preset level adds to a generator at the instrument level if both generators are identical.
- A generator in a global preset zone which is identical to a default generator or to a generator in an instrument adds to that generator.
- A generator in a global preset zone which is not identical to a default generator and is not identical to a generator in an instrument has its effect added to the given synthesis parameter.
- A generator in a local preset zone which is identical to a generator in a global preset zone supersedes or replaces that generator in the global preset zone. That generator then has its effects added to the destination summing node of all zones in the given instrument.
- A generator in a local preset zone which is not identical to a default generator or a generator in a global preset zone has its effects added to the destination summing node of all zones in the given instrument.
  If the generator operator is a Range Generator, the generator values are NOT ADDED to those

in the instrument level, rather they serve as an intersection filter to those key number or velocity ranges in the instrument which is used in the preset zone.

- If the generator operator is a Substitution Generator or a Sample Generator, they are illegal at the preset level. The only Index Generator legal at the Preset Level is ‘instrumentID’, whereas the only Index Generator legal at the Instrument Level is ‘sampleID’

5.9.4.5 The SASBF Modulator Controller Model

SASBF Modulators are used to allow real-time control over the sound in sound designer programmable manner. Each instance of a SASBF modulator structure defines a real-time perceptually additive effect to be applied to a given destination or synthesiser parameter.

Modulators provide future extensibility to the SASBF standard. They are not used in this version.

5.9.5 Error Handling

5.9.5.1 Structural Errors

Structural Errors are errors which are determined from the implicit redundancy of the SASBF RIFF bitstream element structure, and indicate that the structure is not intact. Examples are incorrect lengths for the chunks or subchunks, pointers out of valid range, or missing required chunks or subchunks for which no error correction procedure exists.

In all cases, bitstream elements should be checked for structural errors at load time, and if any are found, the bitstream elements should be rejected. Separate tools or options can be used to “repair” structurally defective bitstream elements, but these tools should validate that the reconstructed bitstream element is not only a valid SASBF bank but also complies with the intended timbral results in all cases.

5.9.5.2 Unknown Chunks

In parsing the RIFF structure, unknown but well formed chunks or subchunks may be encountered. Unknown chunks within the INFO-list chunk should simply be ignored. Other unknown chunks or subchunks are illegal and should be treated as structural errors.

5.9.5.3 Unknown Enumerators

Unknown enumerators may be encountered in Generators, Modulator Sources, or Transforms. This is to be expected if the ifil field exceeds the specification to which the application was written. Even if unexpected, unknown enumerators should simply cause the associated Generator or Modulator to be ignored.

5.9.5.4 Illegal Parameter Values

Some SASBF parameters are defined for only a limited range of the possible values which can be expressed in their field. If the value of the field is not in the defined range, the parameter has an illegal value.

Illegal values for may be detected either at load or at run time. If detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the default value for the parameter should be used if the parameter is required, or the entire Generator or Modulator ignored if it is optional. Certain parameters may have more specific procedures for illegal values as expressed elsewhere in this specification.

5.9.5.5 Out-of-range Values

SASBF parameters have a specified minimum and useful range the span the perceptually relevant values for the associated sonic property. When the parameter value is exceeds this useful range, the parameter is said to have an out of range value.

Out of range values can result from two distinct causes. An out of range value can be actually present as a SASBF generator value, or the out of range value can be the result of the summation of instrument and preset values.

Out of range values should be handled by substituting the nearest perceptually relevant or realisable value. SASBF banks should not be created with out of range values in the instrument generators. While it is acceptable practice to create SASBF banks which produce out of range values as a result of summation, it is undesirable and should be avoided where practical.

5.9.5.6 Missing Required Parameter or Terminator

Certain parameters and terminators are required by the SASBF specification. If these are missing, the bitstream element is technically not within specification. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the instrument or zone for which the required parameter is missing should simply be ignored. If this causes no sound, the corresponding key-on event is ignored.

5.9.5.7 Illegal enumerator

Certain enumerators are illegal in certain contexts. For example, key and velocity ranges must be the first generators in a zone, instruments are not allowed in instrument zones, and sampleIDs are not allowed in preset zones. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the enumerator should simply be ignored.

5.9.6 Profile 2 (Sample Bank and MIDI decoding)

This Subclause describes the decoding process in which a bitstream conforming to Profile 2 is converted into sound. Profile 2 supports the Structured Audio Sample Bank Format and the General MIDI Format (Profile 1) for sound description. Profile 2 supports the MIDI Protocol and the Standard MIDI File Format for score specification.

5.9.6.1 Stream information header

The SASBF bitstream element is part of the bitstream header. Score elements in the form of MIDI files may be included in the bitstream header.

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class `SA_streaminfo` provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- MIDI file: The events in the MIDI file shall be time ordered, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed. The sample bank requires no processing for this preparation. Processing may be used to improve the computational

efficiency of a decoder. Banks shall be assigned consecutive increasing bank ID numbers in the order of the banks' position in the bitstream. The first bank in the bitstream shall be numbered 0 unless a bank resident in the terminal is being used. In that case, the resident bank shall be numbered 0, an other banks shall be numbered proceeding from there.

5.9.6.2 Bitstream data and sound creation

The MIDI Note On message is defined in the MIDI specification. When the SASBF decoder receives a Note On message, a voice shall be instantiated. Synthesis parameters for the voice shall be determined by the data in the sample bank containing the preset corresponding to the MIDI channel of the Note On message. The number of wavetable oscillators that are used by the voice is defined by the sample bank. Each required oscillator shall have the state variables required for synthesis allocated to it. The state variables shall be initialised according to the preset data from the sample bank.

The MIDI Program Change message is defined in the MIDI specification. When the SASBF decoder receives a Program Change message, an assignment shall be made in the scheduler between the specified MIDI channel and the specified preset. Until this assignment is changed by a subsequent Program Change message, subsequent voice instantiations using the specified MIDI channel shall use sample bank data corresponding to the assigned preset. In a MIDI program change message, if no preset exists in the specified bank with the specified preset number, a replacement preset is used. The replacement preset is the preset with the specified preset number in the bank with the highest bank ID number less than the specified bank ID number which contains a preset with the specified preset number.

The run time scheduler is used to issue MIDI messages to the SASBF decoder. MIDI messages from a MIDI standard file shall be issued by the scheduler when the scheduler clock equals or exceeds the time stamp associated with the message.

5.9.6.3 Conformance

Floating point computation is not required in Profile 2. Audio samples are stored in the bitstream as 16-bit integers. The resolution of the interpolator filter is constrained by the interpolator specification given in Subclause 0.

5.9.7 Profile 4 (Sample Bank decoding in SAOL instruments)

5.9.8 Sample Bank Format Glossary

absolute - Describes a parameter which gives a definitive real-world value. Contrast to relative.

additive - Describes a parameter which is to be numerically added to another parameter.

articulation - The process of modulation of amplitude, pitch, and timbre to produce an expressive musical note.

attack - That phase of an envelope or sound during which the amplitude increases from zero to a peak value.

attenuation - A decrease in volume or amplitude of a signal.

bag - A Sample Bank Format data structure element containing a list of zones.

balance - A form of stereo volume control in which both left and right channels are at maximum when the control is centred, and which attenuates only the opposite channel when taken to either extreme.

bank - A collection of presets. See also MIDI bank.

bipolar - In the SASBF standard, said of a modulator source whose minimum is -1 and whose maximum is 1. Contrast “unipolar”

big endian - Refers to the organisation in memory of bytes within a word such that the most significant byte occurs at the lowest address. Contrast “little endian.”

byte - A data structure element of eight bits without definition of meaning to those bits.

BYTE - A data structure element of eight bits which contains an unsigned value from 0 to 255.

case-insensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as identical. Contrast “case-sensitive.”

case-sensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as distinct. Contrast “case-insensitive.”

cent - A unit of pitch ratio corresponding to the twelve hundredth root of two, or one hundredth of a semitone, approximately 1.000577790.

centibel - A unit of amplitude ratio corresponding to the two hundredth root of ten, or one tenth of a decibel, approximately 1.011579454.

CHAR - A data structure of eight bits which contains a signed value from -128 to +127.

chorus - An effects processing algorithm which involves cyclically shifting the pitch of a signal and remixing it with itself to produce a time varying comb filter, giving a perception of motion and fullness to the resulting sound.

chunk - The top-level division of a RIFF file.

convex - A curve which is bowed in such a way that it is steeper on its lower portion.

concave - (1) A curve which is bowed in such a way that it is steeper on its upper portion. (2) In the SASBF standard, said of a modulator source whose shape is that of the amplitude squared characteristic. Contrast with “convex” and “linear.”

cutoff frequency - The frequency of a filter function at which the attenuation reaches a specified value.

data points - The individual values comprising a sample. Sometimes also called sample points. Contrast “sample.”

decay - The portion of an envelope or sound during which the amplitude declines from a peak to steady state value.

decibel - A unit of amplitude ratio corresponding to the twentieth root of ten, approximately 1.122018454.

delay - The portion of an envelope or LFO function which elapses from a key-on event until the amplitude becomes non-zero.

destination - The generator to which a modulator is applied.

DC gain - The degree of amplification or attenuation a system presents to a static or zero frequency signal.

digital audio - Audio represented as a sequence of quantised values spaced evenly over time. The values are called “sample data points.”

doubleword - A data structure element of 32 bits without definition of meaning to those bits.

downloadable - Said of samples which are loaded from a file into RAM, in contrast to samples which are maintained in ROM.

dry - Refers to audio which has not received any effects processing such as reverb or chorus.

DWORD - A data structure of 32 bits which contains an unsigned value from zero to 4,294,967,295.

envelope - A time varying signal which typically controls the pitch, volume, and/or filter cutoff frequency of a note, and comprises multiple phases including attack, decay, sustain, and release.

enumerated - Said of a data element whose symbols correspond to particular assigned functions.

flat - A. Said of a tone that is lower in pitch than another reference tone. B. Said of a frequency response that does not deviate significantly from a single fixed gain over the audio range.

generator - In the SASBF standard, a parameter which directly affects sound reproduction. Contrast with “modulator.”

global - Refers to parameters which affect all associated structures. See “global zone.”

global zone - A zone whose generators and modulators affect all other zones within the object.

header - A data structure element which describes several aspects of a SASBF element.

instrument - In the SASBF standard, a collection of zones which represents the sound of a single musical instrument or sound effect set.

instrument zone - A sample and associated articulation data defined to play over certain key numbers and velocities.

interpolator - A circuit or algorithm which computes intermediate points between existing sample data points. This is of particular use in the pitch shifting operation of a wavetable synthesiser, in which these intermediate points represent the output samples of the waveform at the desired pitch transposition.

key number - See MIDI key number.

LFO - Acronym for Low Frequency Oscillator. A slow periodic modulation source.

linear - In the SASBF standard, said of a modulator source whose shape is that of a straight line. Contrast with “concave.”

linear coding - The most common method of encoding amplitudes in digital audio in which each step is of equal size.

little endian - A method of ordering bytes within larger words in memory in which the least significant byte is at the lowest address. Contrast “big endian.”

loop - In wavetable synthesis, a portion of a sample which is repeated many times to increase the duration of the resulting sound.

loop points - The sample data points at which a loop begins and ends.

lowpass - Said of a filter which attenuates high frequencies but does not attenuate low frequencies.

modulator - In the SASBF standard, a parameter which routes an external controller to dynamically alter the setting of a “generator.” Contrast with “generator.”

monotonic - Continuously increasing or decreasing. Said of a sequence which never reverses direction.

MIDI - Acronym for Musical Instrument Digital Interface. The standard protocol for sending performance information to a musical synthesiser.

MIDI bank - A group of up to 128 presets selected by a MIDI “change bank” command.

MIDI continuous controller - A construct in the MIDI protocol.

MIDI key number - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the key of the musical instrument keyboard to which the command refers.

MIDI pitch bend - A special MIDI construct akin to the MIDI continuous controllers which controls the real-time value of the pitch of all notes played in a MIDI channel.

MIDI preset - A “preset” selected to be active in a particular MIDI channel by a MIDI “change preset” command.

MIDI velocity - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the speed with which the key was pressed or released.

modulator - In the SASBF standard, a set of parameters which affect a particular generator. Contrast with “generator.”

mono - Short for “monophonic.” Indicates a sound comprising only one channel or waveform. Contrast with “stereo.”

negative - In the SASBF standard, said of a modulator which has a negative sloping characteristic. Contrast with “positive.”

octave - A factor of two in ratio, typically applied to pitch or frequency.

orphan - Said of a data structure which under normal circumstances is referenced by a higher level, but in this particular instance is no longer linked. Specifically, it is an instrument which is not referenced by any preset zone, or a sample which is not referenced by any instrument zone.

oscillator - In wavetable synthesis, the wavetable interpolator is considered an oscillator.

pan - Short for “panorama.” This is the control of the apparent azimuth of a sound source over 180 degrees from left to right. It is generally implemented by varying the volume at the left and right speakers.

pitch - The perceived value of frequency. Generally can be used interchangeably with frequency.

pitch shift - A change in pitch. Wavetable synthesis relies on interpolators to cause pitch shift in a sample to produce the notes of the scale.

pole - A mathematical term used in filter transform analysis. Traditionally in synthesis, a pole is equated with a rolloff of 6dB per octave, and the rolloff of a filter is specified in “poles.”

positive - In the SASBF standard, said of a modulator source which has a positive sloping characteristic. Contrast “negative.”

preset - A keyboard full of sound. Typically the collection of samples and articulation data associated with a particular MIDI preset number.

preset zone - A subset of a preset containing generators, modulators, and an instrument.

proximal - Closest to. Proximal sample data points are the data points closest in either direction to the named point.

Q - A mathematical term used in filter transform analysis. Indicates the degree of resonance of the filter. In synthesis terminology, it is synonymous with resonance.

RAM - Random Access Memory. Conventionally, this term implies read-write memory. Contrast “ROM.”

record - A single instance of a data structure.

relative - Describes a parameter which merely indicates an offset from an otherwise established value. Contrast to absolute.

release - The portion of an envelope or sound during which the amplitude declines from a steady state to zero value or inaudibility.

resonance - Describes the aspect of a filter in which particular frequencies are given significantly more gain than others. The resonance can be measured in dB above the DC gain.

resonant frequency - The frequency at which resonance reaches its maximum.

reverb - Short for reverberation. In synthesis, a synthetic signal processor which adds artificial spaciousness and ambience to a sound.

RIFF - Acronym for Resource Interchange File Format.

ROM - Acronym for Read Only Memory. A memory whose contents are fixed at manufacture, and hence cannot be written by the user. Contrast with RAM.

sample - This term is often used both to indicate a “sample data point” and to indicate a collection of such points comprising a digital audio waveform. The latter meaning is exclusively used in this Subclause (the SASBF specification.)

sample rate - The frequency, in Hertz, at which sample data points are taken when recording a sample.

semitone - A unit of pitch ratio corresponding to the twelfth root of two, or one twelfth of an octave, approximately 1.059463094.

sharp - Said of a tone that is higher in pitch than another reference tone.

SHORT - A data structure element of sixteen bits which contains a signed value from -32,768 to +32,767.

soft - The pedal on a piano, so named because it causes the damper to be lowered in such a way as to soften the timbre and loudness of the notes. In MIDI, continuous controller #66, which behaves in a similar manner.

sostenuto - The pedal on a piano which causes the dampers on all keys depressed to be held until the pedal is released. In MIDI, continuous controller #67, which behaves in a similar manner.

sustain - The pedal on a piano which prevents all dampers on keys as they are depressed from being released. In MIDI, continuous controller #64, which behaves in a similar manner.

source - In a SASBF modulator, the enumerator indicating the particular real-time value which the modulator will transform, scale, and add to the destination generator.

stereo - Literally indicating three dimensions. In this specification, the term is used to mean two channel stereophonic, indicating that the sound is composed of two independent audio channels, dubbed left and right. Contrast monophonic.

subchunk - A division of a RIFF file below that of the chunk.

synthesis engine - The hardware and software associated with the signal processing and modulation path for a particular synthesiser.

synthesiser - A device ideally capable of producing arbitrary musical sound.

terminator - A data structure element indicating the final element in a sequence.

timecent - A unit of duration ratio corresponding to the twelve hundredth root of two, or one twelve hundredth of an octave, approximately 1.000577790.

transform - In a SASBF modulator, the enumerator indicating the particular transfer function through which the source will be passed prior to scaling and addition to the destination generator.

tremolo - A periodic change in amplitude of a sound, typically produced by applying a low frequency oscillator to the final volume amplifier.

triangular - A waveform which ramps upward to a positive limit, then downward at the opposite slope to the symmetrically negative limit periodically.

unipolar - In the SASBF standard, said of a modulator source whose minimum is 0 and whose maximum is 1. Contrast “bipolar.”

velocity - In synthesis, the speed with which a keyboard key is depressed, typically proportionally to the impact delivered by the musician. See also MIDI velocity.

vibrato - A periodic change in the pitch of a sound, typically produced by applying a low frequency oscillator to the oscillator pitch.

volume - The loudness or amplitude of a sound, or the control of this parameter.

wavetable - A music synthesis technique wherein musical sounds are recorded or computed mathematically and stored in a memory, then played back at a variable rate to produce the desired pitch. Additional timbral adjustments are often made to the sound thus produced using amplifiers, filters, and effects processing such as reverb and chorus.

WORD - A data structure of 16 bits which contains an unsigned value from zero to 65,535.

word - A data structure element of 16 bits without definition of meaning to those bits.

zone - An object and associated articulation data defined to play over certain key numbers and velocities.

5.10 MIDI semantics for more details on MIDI control of orchestras.

5.4.6.8.11 MIDIbend

```
ksig MIDIbend
```

The **MIDIbend** standard variable shall contain, for each scope, the current value of the MIDI pitchbend on the channel corresponding to the channel to which the instrument instantiation associated with that scope is assigned.

5.4.6.8.12 input

```
asig input[inchannels]
```

The **input** standard variable shall contain, for each scope, the input signal or signals being provided to the instrument instantiation through the send instruction.

5.4.6.8.13 inGroup

```
ivar inGroup[inchannels]
```

The **inGroup** standard variable shall contain, for each scope, the grouping of the input signals being provided to the instrument instantiation. See Subclause 5.4.5.5 Send statement.

5.4.6.8.14 released

```
ksig released
```

The **released** standard name shall contain, for each scope, 1 if and only if the instrument instantiation associated with the scope is scheduled to be destroyed at the end of the current orchestra pass. Otherwise, released shall contain 0.

5.4.6.8.15 cpuload

```
ksig cpuload
```

The **cpuload** standard name shall contain, for each scope, a measure of the recent CPU load on the CPU most strongly associated with the instrument instantiation associated with the scope. If the instrument instantiation is running entirely on one CPU, then that CPU shall be measured; if the instrument instantiation is running on multiple CPUs, then the exact measurement procedure is nonnormative. The measure of CPU load shall be as a percentage of real-time capability: if the CPU is entirely loaded and cannot perform any more calculations without slipping out of real-time performance, the value of **cpuload** shall be 1 on that CPU at that k-cycle. If the CPU is entirely unloaded and is not performing any calculations, the value of **cpuload** shall be 0 on that CPU at that k-cycle. If the CPU is half-loaded, and could perform twice as many calculations in real-time as it is currently performing, the value of **cpuload** shall be 0.5 on that CPU at that k-cycle.

The exact calculation method, time window, recency, etc. of the CPU load is left to implementors.

5.4.6.8.16 position

```
imports ksig position[3]
```

The **position** name contains the absolute position of the node responsible for creating the current orchestra in the BIFS scene graph (see ISO 14496-1 Subclause XXX). The position is given by the current value of the **position** field of the **Sound** node which is the ancestor of this node in the scene graph, as transformed by its ancestors (that is, the final position in world co-ordinates of the **Sound** node). The value is global and shared by all instruments; it may not be changed by the orchestra.

5.4.6.8.17 direction

```
ksig direction[3]
```

The **direction** name contains the orientation of the node responsible for creating the current orchestra in the BIFS scene graph (see ISO 14496-1 Subclause XXX). The direction is given by the current value of the **direction** field of the **Sound** node which is the ancestor of this node in the scene graph, as transformed by its ancestors (that is, the final direction in world co-ordinates of the **Sound** node). The value is global and shared by all instruments; it may not be changed by the orchestra.

5.4.6.8.18 listenerPosition

```
ksig listenerPosition[3]
```

The **listenerPosition** name contains the absolute position of the listener in the BIFS scene graph (see ISO 14496-1 Subclause XXX). The position is given by the current value of the **position** field of the active **ListeningPoint** node in the scene graph, as transformed by its ancestors (that is, the final position in world co-ordinates of the **ListeningPoint** node).

5.4.6.8.19 listenerDirection

```
ksig listenerDirection[3]
```

The **listenerDirection** name contains the orientation of the listener in the BIFS scene graph (see ISO 14496-1 Subclause XXX). The direction is given by the current value of the **direction** field of the active **ListeningPoint** node in the scene graph, as transformed by its ancestors (that is, the final direction in world co-ordinates of the **ListeningPoint** node).

5.4.6.8.20 minFront

```
ksig minFront
```

The **minFront** standard name gives one parameter of the sound radiation pattern of the sound which the current node is a part of. This parameter, and its semantics, are defined by the **minFront** field of the **Sound** node of which this node is an ancestor (see ISO 14496-1 Subclause XXX).

5.4.6.8.21 maxFront

```
ksig maxFront
```

The **maxFront** standard name gives one parameter of the sound radiation pattern of the sound which the current node is a part of. This parameter, and its semantics, are defined by the **maxFront** field of the **Sound** node of which this node is an ancestor (see ISO 14496-1 Subclause XXX).

5.4.6.8.22 minBack

```
ksig minBack
```

The **minBack** standard name gives one parameter of the sound radiation pattern of the sound which the current node is a part of. This parameter, and its semantics, are defined by the **minBack** field of the **Sound** node of which this node is an ancestor (see ISO 14496-1 Subclause XXX).

5.4.6.8.23 maxBack

```
ksig maxBack
```

The **maxBack** standard name gives one parameter of the sound radiation pattern of the sound which the current node is a part of. This parameter, and its semantics, are defined by the **maxBack** field of the **Sound** node of which this node is an ancestor (see ISO 14496-1 Subclause XXX).

5.4.6.8.24 params

```
imports exports ksig params[128]
```

The **params** standard name is shared globally by all instruments. At each k-cycle of the orchestra, it shall contain the current values of the **params** field of the BIFS **AudioFX** node responsible for instantiating the current orchestra. If the orchestra is created by an **AudioSource** node rather than an **AudioFX** node, the value of **params** shall be 0 on every channel. See Subclause 5.11.3 The AudioFX node for more details.

5.4.7 Opcode definition

5.4.7.1 Syntactic Form

This Subclause describes the definition of new opcodes. Bitstream authors may create their own opcodes according to these rules in order to encapsulate functionality and simplify instruments and the content authoring process.

<opcode definition> -> <opcode rate> **<ident> (** <formal param list> **) {**
<opcode var declarations>
<opcode statement block>
}

<opcode rate> -> **aopcode**
<opcode rate> -> **kopcode**
<opcode rate> -> **iopcode**
<opcode rate> -> **opcode**

An opcode definition has several elements. In order, they are

1. A rate tag which defines the rate at which the opcode executes, or indicates that the opcode is rate-polymorphic,

2. An identifier which defines the name of the opcode,

3. A list of zero or more formal parameters of the opcode,

4. A list of zero or more opcode variable declarations,

5. A block of statements defining the executable functionality of the opcode.

5.4.7.2 Rate tag

The rate tag describes the rate at which the opcode is to run, or else indicates that the opcode is rate-polymorphic. The four rate tags are

1. **iopcode**, indicating that the opcode runs at i-rate,

2. **kopcode**, indicating that the opcode runs at k-rate,

3. **aopcode**, indicating that the opcode runs at i-rate,

4. **opcode**, indicating that the opcode is rate-polymorphic.

See Subclause **Fehler! Verweisquelle konnte nicht gefunden werden.** for instructions on determining the rate of a rate-polymorphic opcode.

5.4.7.3 Opcode name

Any identifier may serve as the opcode name except that the opcode name shall not be a reserved word (see Subclause 5.4.8 Template declaration

5.4.8.1 Syntactic form

<template declaration> -> **template <** <identlist> **> (** <identlist> **)**
map { <identlist> **} with {** <maplist> **}**
{ <instr variable declarations> <block> **}**<maplist> -> **<** <expr list> **> ,** <maplist>
<maplist> -> **<** <expr list> **>**

<identlist> as given in Subclause 5.4.5.4 Route statement.
<namelist> as given in Subclause 5.4.5.3.2 Signal variables.
<instr variable declarations> as given in Subclause 5.4.6.5.1 Syntactic form.
<expr list> as given in Subclause 5.4.6.6.1 Syntactic form.
<block> as given in Subclause 5.4.6.6.1 Syntactic form.

A template declaration allows the concise declaration of multiple instruments which are similar in processing structure and syntax, but differ in only a few key expressions or wavetable names.

5.4.8.2 Semantics

The first identifier list contains the names for the instruments declared with the template. There shall be at least one identifier in this list. The second identifier list contains the pfields for the template declaration. Each instrument declared with the template has the same list of pfields. The third identifier list contains a list of template variables which are to be replaced in the subsequent code block with expressions from the

map list. There may be no identifiers in this list, in which case each instrument declared by the template is exactly the same.

The map list takes the form of a list of lists. This list shall have as many elements as template variables declared in the third identifier list. Each sublist is a list of expressions, and shall have as many elements as instrument names in the first identifier list.

5.4.8.3 Template instrument definitions

As many instruments are defined by the template definition as there are names in the first identifier list. To describe each of the instruments, the identifiers described in the third (template variable) list are replaced in turn by each of the expressions from the map list.

That is, to construct the code for the first instrument, the code block given is processed by replacing the first template variable with the first expression from the first map list sublist, the second template variable with the first expression from the second map list sublist, the third template variable with the first expression from the third map list sublist, and so on. To construct the code for the second instrument, the code block given is processed by replacing the first template variable with the *second* expression from the first map list sublist, the second template variable with the *second* expression from the second map list sublist, the third template variable with the second expression from the third map list sublist, and so on.

This code-block processing occurs before any other syntax-checking or rate-checking of the elements of the instruments so defined. That is, the template variables are not true signal variables, and do not need to be declared in the variable declaration block. Once the code-block processing and template expansion is complete, the resulting instruments are treated as any other instruments in the orchestra.

EXAMPLE

The following template declaration:

```
template <oneharm, threeharm>(p)
  map {pitch,t,bar} with { <440, harm1, mysig>, <p, harm2, mysig * mysig + 2> }
{

  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(t,pitch,-1);

  mysig = bar *3;
  output(mysig);
}
```

declares exactly the same two instruments as the following two instrument declarations:

```
instr oneharm(p) {
  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(harm1,440,-1);

  mysig = mysig * 3;
  output(mysig);
  }

instr threeharm(p) {
  table harm1(harm,4096,1);
```

```
table harm3(harm,4096,3,2,1);
asig mysig;

mysig = oscil(harm3,p,-1);

mysig = (mysig * mysig + 2) * 3;  // notice embedding of template expression
output(mysig);
}
```

5.4.9 Reserved words)., the name of one of the core opcodes listed in Subclause 5.5 SAOL core opcode definitions and semantics, or the name of one of the core wavetable generators listed in Subclause 5.6 SAOL core wavetable generators An opcode name may be the same as the name of a variable in local or global score; there is no ambiguity so created, since the contexts in which opcode names may occur are very restricted.

No two instruments or opcodes in an orchestra shall have the same name.

5.4.7.4 Formal parameter list

5.4.7.4.1 Syntactic form

<formal param list> -> <formal param> [**,** <formal param list>]
<formal param list> -> **<NULL>**

<formal param> -> <opcode variable rate> <name>
<formal param> -> **table <ident>**

<opcode variable rate> -> **asig**
<opcode variable rate> -> **ksig**
<opcode variable rate> -> **ivar**
<opcode variable rate> -> **xsig**

<name> as defined in Subclause 5.4.5.3.2 Signal variables.

The formal parameter list defines the calling interface to the opcode. Each formal parameter in the list has a name, a rate type, and may have an array width. If the array width is the special token **inchannels**, then the array width shall be the same as the number of input channels to the associated instrument instantiation (in the sense of Subclause 0); if the array width is the special token **outchannels**, then the array width shall be the same as the number of orchestra output channels as defined in the global block.

There is no way to create user-defined opcodes with variable number of arguments in SAOL, although certain of the core opcodes have this property.

Within the opcode statement block, formal parameters may be used like any other variable. The rate tag of each formal parameter defines the rate of the variable. If an opcode is declared to be at a particular rate, then no formal parameter shall be declared faster than that rate.

There is a special rate tag **xsig** which allows formal parameters to be rate-polymorphic, see Subclause **Fehler! Verweisquelle konnte nicht gefunden werden.**. **xsig** shall not be the rate tag of any formal parameter unless the opcode is of type **opcode**.

5.4.7.5 Opcode variable declarations

5.4.7.5.1 Syntactic form

<opcode var declarations> -> <opcode var declaration> [<opcode var declarations>]
<opcode var declarations> -> **<NULL>**

<opcode var declaration> -> <instr variable declaration>
<opcode var declaration> -> **xsig** <namelist> **;**

<instr variable declaration> as defined in Subclause 5.4.6.5.1 Syntactic form.
<namelist> as defined in Subclause 5.4.5.3.2 Signal variables.

The syntax and semantics of opcode variable declarations are the same as those of instrument variable declarations as given in Subclause **Fehler! Verweisquelle konnte nicht gefunden werden.**, with the following exceptions and additions:

The opcode variable names are available only within the scope of the opcode containing them. The instrument variable declarations for the instrument instantiation associated with the opcode call are not within the scope of an opcode, and references to these names shall not be made unless the names are also explicitly declared within the opcode, in which case the variable denoted is a different one. However, standard names (Subclause 5.4.6.8 Standard names) are within the scope of every opcode, may be referenced within opcodes, and shall have the semantics given in Subclause 5.4.6.8 Standard names as applied to the instrument instantiation associated with the opcode call.

The values of opcode variables are static, and are preserved from call to call referencing a particular opcode state. The values of opcode variables shall be set to 0 in an opcode state before the first call referencing that state is executed.

The **tablemap** declaration may reference any tables declared in the local scope, as well as any formal parameters which are tables.

The values of opcode variables in different states of the same opcode (due to different syntactic uses of opcode expressions, or different indexing expressions in oparray expressions) are separate and have no relationship to one another.

There is a special rate tag called **xsig** which may be used to declare opcode variables in rate-polymorphic opcodes, see Subclause **Fehler! Verweisquelle konnte nicht gefunden werden.**. **xsig** shall not be the rate tag of any variable unless the opcode type is **opcode**.

5.4.7.6 Opcode statement block

5.4.7.6.1 Syntactic form

<opcode statement block> -> <opcode statement> [<opcode statement block>]
<opcode statement block> -> **<NULL>**

<opcode statement> -> <statement>
<opcode statement> -> **return (** <expr list> **) ;**

<statement> as defined in Subclause 5.4.6.6.1 Syntactic form.
<expr list> as defined in Subclause 5.4.6.6 Block of code statements.

The syntax and semantics of statements in opcodes are the same as the syntax and semantics of statements in instruments, with the following exceptions and additions:

No statement in an opcode shall be faster than the rate of the opcode, as defined in Subclause **Fehler! Verweisquelle konnte nicht gefunden werden.**.

The assignment statement and the values of all variables refer to the opcode state associated with this particular call to this opcode, or associated with a particular indexing expression in an oparray call.

There is a special statement called **return** which is used in opcodes. This statement allows opcodes to return values back to their callers.

5.4.7.6.2 Return statement

The **return** statement allows opcodes to return values back to their callers.

The expression parameter list may contain both single-valued and array-valued expressions.

The rate of the **return** statement is the rate of the opcode containing it. No expression in the expression parameter list shall be faster than the rate of the opcode.

The **return** statement shall be evaluated as follows. Each expression in the expression parameter list is evaluated, in the order they occur in the list. The return value of the opcode is the array-value formed by sequencing the values of the expression parameters. In the case that there is only one expression parameter which is a single-valued expression, then the return value of the opcode is the single value of that expression. The return value denoted by every **return** statement within an opcode shall have the same width (although it is permissible for them to differ in the number of expressions, so long as the sum of the widths of the expressions is equal).

After a **return** statement is encountered, no further statements in the opcode are evaluated, and control returns immediately to the calling instrument or opcode.

5.4.7.7 Opcode rate

5.4.7.7.1 Introduction

This Subclause describes the rules for determining the rate of a call to an opcode, and the semantics of the special tags **opcode** and **xsig.**

The rate of an opcode call depends on the type of the opcode, as follows:

1. If the opcode type is **aopcode**, calls to the opcode are a-rate.

2. If the opcode type is **kopcode**, calls to the opcode are k-rate.

3. If the opcode type is **iopcode**, calls to the opcode are i-rate.

4. If the opcode type is **opcode**, the opcode is rate-polymorphic, and the rate is as described in the next Subclause.

5.4.7.7.2 Rate-polymorphic opcodes

Opcodes which are rate-polymorphic take their rates from the context in which they are called. This allows the same opcode statement block to apply to multiple calling rate contexts. Without such a construct, three versions of each opcode of this sort would have to be created and used, depending on the context.

The rate of an **opcode** opcode for a particular call is the rate of the fastest actual parameter expression (not formal parameter expression) in that call, or the rate of the fastest formal parameter in the opcode definition, or the rate of the fastest guarding **if**, **while**, or **else** expression surrounding the opcode call, or the rate of the opcode enclosing the opcode call, whichever is fastest.

Rate-polymorphic opcodes may contain variable declarations and formal parameter declarations using the special rate tag **xsig**. A formal parameter of type **xsig**, for a particular call to that opcode, has the same rate as the actual parameter expression in the calling expression to which it corresponds. A variable of type **xsig**, for a particular call to that opcode, has the same rate as the opcode.

EXAMPLES

Given the following opcode definition:

```
opcode xop(ksig p1, xsig p2) {
  xsig v1;
  . . .
}
```

1. For the following code fragment

```
ksig k;

k = xop(1,2);
```

the rate of the opcode call is k-rate, since the formal parameter **p1** is faster than either of the actual parameters. The rate of **p2** within the call to **xop()** is i-rate, matching the actual parameter. The rate of **v1** within **xop()** is k-rate.

2. For the following code fragment

```
asig a1, a2;

a1 = xop(1,a2);
```

the rate of the opcode call is a-rate, since the actual parameter **a2** is faster than either of the formal parameters. The rates of **p2** and **v1** within the call to **xop()** are k-rate and a-rate respectively.

3. For the following code fragment

```
ksig k;
asig a;

k = xop(1,a)
```

there is a rate mismatch error, since the opcode call is a-rate, and thus shall not be assigned to a k-rate lvalue.

4. For the following code fragment

```
ksig k;
asig a1,a2;
oparray xop[10];

a1 = 0; while (a1 < 10) {
  a2 = a2 + xop[a1](1,k);
  a1 = a1 + 1;
}
```

the rate of the oparray call is a-rate, since the rate of the guarding expression is faster than any of the formal parameters or actual parameters. The rates of **p2** and **v1** within **xop()** are a-rate as well.

5.4.8 Template declaration

5.4.8.1 Syntactic form

<template declaration> -> **template <** <identlist> **> (** <identlist> **)**
map { <identlist> **} with {** <maplist> **}**
{ <instr variable declarations> <block> **}**<maplist> -> **<** <expr list> **> ,** <maplist>
<maplist> -> **<** <expr list> **>**

<identlist> as given in Subclause 5.4.5.4 Route statement.
<namelist> as given in Subclause 5.4.5.3.2 Signal variables.
<instr variable declarations> as given in Subclause 5.4.6.5.1 Syntactic form.
<expr list> as given in Subclause 5.4.6.6.1 Syntactic form.
<block> as given in Subclause 5.4.6.6.1 Syntactic form.

A template declaration allows the concise declaration of multiple instruments which are similar in processing structure and syntax, but differ in only a few key expressions or wavetable names.

5.4.8.2 Semantics

The first identifier list contains the names for the instruments declared with the template. There shall be at least one identifier in this list. The second identifier list contains the pfields for the template declaration. Each instrument declared with the template has the same list of pfields. The third identifier list contains a list of template variables which are to be replaced in the subsequent code block with expressions from the map list. There may be no identifiers in this list, in which case each instrument declared by the template is exactly the same.

The map list takes the form of a list of lists. This list shall have as many elements as template variables declared in the third identifier list. Each sublist is a list of expressions, and shall have as many elements as instrument names in the first identifier list.

5.4.8.3 Template instrument definitions

As many instruments are defined by the template definition as there are names in the first identifier list. To describe each of the instruments, the identifiers described in the third (template variable) list are replaced in turn by each of the expressions from the map list.

That is, to construct the code for the first instrument, the code block given is processed by replacing the first template variable with the first expression from the first map list sublist, the second template variable with the first expression from the second map list sublist, the third template variable with the first expression from the third map list sublist, and so on. To construct the code for the second instrument, the code block given is processed by replacing the first template variable with the *second* expression from the first map list sublist, the second template variable with the *second* expression from the second map list sublist, the third template variable with the second expression from the third map list sublist, and so on.

This code-block processing occurs before any other syntax-checking or rate-checking of the elements of the instruments so defined. That is, the template variables are not true signal variables, and do not need to be declared in the variable declaration block. Once the code-block processing and template expansion is complete, the resulting instruments are treated as any other instruments in the orchestra.

EXAMPLE

The following template declaration:

```
template <oneharm, threeharm>(p)
  map {pitch,t,bar} with { <440, harm1, mysig>, <p, harm2, mysig * mysig + 2> }
{

  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(t,pitch,-1);

  mysig = bar *3;
  output(mysig);
}
```

declares exactly the same two instruments as the following two instrument declarations:

```
instr oneharm(p) {
  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(harm1,440,-1);

  mysig = mysig * 3;
  output(mysig);
  }

instr threeharm(p) {
  table harm1(harm,4096,1);
  table harm3(harm,4096,3,2,1);
  asig mysig;

  mysig = oscil(harm3,p,-1);

  mysig = (mysig * mysig + 2) * 3;  // notice embedding of template expression
  output(mysig);
  }
```

5.4.9 Reserved words

The following words are reserved, and shall not be used as identifiers in a SAOL orchestra or score.

aopcode asig else exports extend global if imports inchannels inputmod instr iopcode ivar kopcode krate ksig map oparray opcode outbus outchannels output return route send sequence sbsynth spatialize srate table tablemap template turnoff while with xsig

Also, variable names starting with **_sym_** are reserved for implementation-specific use (for example, bitstream detokenisation – see Annex B).

5.5 SAOL core opcode definitions and semantics

5.5.1 Introduction

This Subclause describes the definitions and normative semantics for each of the core opcodes in SAOL. All core opcodes shall be implemented in every terminal complying to Profile 4.

For each core opcode, the following is described:

- The prototype, showing the rate of the opcode, the parameters which must be provided in a call to this opcode, and the rates of these parameters.
- The normative semantics of the return value. These semantics describe how to calculate the return value for each call to that opcode.
- The normative semantics of any side effects of the core opcode.

5.5.2 Specialop type

There is a special rate type for certain core opcodes called **specialop**. This rate type tag is not an actual lexical element of the SAOL language, and shall not appear in a SAOL orchestra, but is used in subsequent Subclauses as a shorthand for core opcodes with these particular semantics.

Core opcodes with rate type **specialop** describe functions which map from one or more a-rate signals into a k-rate signal. That is, they have one or more parameters which vary at the a-rate, and they have normative semantics described at the a-rate, but they only return values and/or have side effects at the k-rate. When using these opcodes in expressions, they are treated as **kopcode** opcodes for the purposes of determining the rate of the expression (although it is not a rate-mismatch error to pass them an a-rate signal), and as **aopcode** opcodes for the purposes of determining when to execute them.

The core opcodes with this type are: **fft**, **rms, sblock, downsamp**, and **decimate**.

5.5.3 List of core opcodes

The several core opcodes are described in the subsequent Subclauses. They are divided by category into major Subclauses, but there is no normative significance in this division; it is only for clarity of presentation.

Math functions	int, frac, dbamp, ampdb, abs, sgn, exp, log, sqrt, sin, cos, atan, pow, log10, asin, acos, floor, ceil, min, max
Pitch converters	gettune, settune, octpch, pchoct, cpspch, pchcps, cpsoct, octcps, midipch, pchmidi, midioct, octmidi, midicps, cpsmidi
Table operations	ftlen, ftloop, ftloopend, ftsr, ftbasecps, ftsetloop, ftsetend, ftsetbase, tableread, tablewrite, oscil, loscil, doscil, koscil
Signal generators	kline, aline, kexpon, aexpon, kphasor, aphasor, pluck, buzz, fof
Noise generators	irand, krand, arand, ilinrand, klinrand, alinrand, iexprand, kexprand, iexprand, kpoissonrand, apoissonrand, igaussrand, kgaussrand, agaussrand

Filters	port, hipass, lopass, bandpass, bandstop, biquad, allpass, comb, fir, iir, firt, iirt
Spectral analysis	fft, ifft
Gain control	rms, gain, balance, compress, pcompress
Sample conversion	decimate, upsamp, downsamp, samphold, sblock
Delays	delay, delay1, fracdelay
Effects	reverb, chorus, flange

For each core opcode, an opcode prototype is given. This shows the rate of the opcode, the number of required and optional formal parameters and the rate of each of the formal parameters. Certain parameters to certain core opcodes are presented in brackets, in which case that formal parameter is optional. Certain opcodes use the "…" notation, which means that the opcode can process an arbitrary number of parameters. The "…" is tagged with a rate for such opcodes, which is then the rate type of all of the parameters matching the varargs parameter. If there is not normative language for a particular opcode which specifies otherwise, it is a syntax error if any of the following statements apply:

- there are fewer actual parameters in the opcode call than required formal parameters
- there are more actual parameters in the opcode call than required and optional formal parameters, and the opcode definition does not include a varargs "..." Subclause
- a particular actual parameter expression is of faster rate than the corresponding formal parameter, or than the varargs formal parameter if that is the correspondence
- a particular actual parameter expression is not single-valued, or is not table-valued when the corresponding formal parameter specifies a table.

The names associated with the formal parameters in the core opcode prototypes have no normative significance, but are used for clarity of exposition to refer to the values passed as the corresponding actual parameters when describing how to calculate the return value of the core opcode.

5.5.4 Math functions

5.5.4.1 Introduction

Each of the opcodes in this Subclause computes a mathematical function. Whenever the result of calculating the function on the argument or arguments provided results in a NaN or Inf value, a run-time error shall result..

5.5.4.2 int

```
opcode int(xsig x)
```

The **int** core opcode calculates the integer part of its parameter.

The return value shall be the integer part of **x**.

5.5.4.3 frac

```
opcode frac(xsig x)
```

The **frac** core opcode calculates the fractional part of its parameter.

The return value shall be the fractional part of **x**, i.e., **x** – int(**x**). If **x** is negative, then **frac(x)** is also negative.

5.5.4.4 dbamp

```
opcode dbamp(xsig x)
```

The **dbamp** core opcode calculates the amplitude equivalent of a decibel-valued parameter, where the maximum amplitude of 1 corresponds to a decibel level of 90 dB.

The return value shall be $10^{(\mathbf{x} - 90) / 10}$.

5.5.4.5 ampdb

```
opcode ampdb(xsig x)
```

The **ampdb** core opcode calculates the decibel equivalent of an amplitude parameter, where the maximum amplitude of 1 corresponds to a decibel level of 90 dB.

The return value shall be $90 + 10 \log_{10} \mathbf{x}$.

5.5.4.6 abs

```
opcode abs(xsig x)
```

The **abs** core opcode calculates the absolute value of a parameter.

The return value shall be **–x** if $\mathbf{x} < 0$, or **x** otherwise.

5.5.4.7 sgn

```
opcode sgn(xsig x)
```

The **sgn** core opcode calculates the signum (sign function) of a parameter.

The return value shall be –1 if $\mathbf{x} < 0$, 0 if $\mathbf{x} = 0$, or 1 if $\mathbf{x} > 0$.

5.5.4.8 exp

```
opcode exp(xsig x)
```

The **exp** core opcode calculates the exponential function.

The return value shall be $e^{\mathbf{x}}$.

5.5.4.9 log

```
opcode log(xsig x)
```

The **log** core opcode calculates the natural logarithm of a parameter.

It is a run-time error if **x** is not strictly positive.

The return value shall be log **x**.

5.5.4.10 sqrt

```
opcode sqrt(xsig x)
```

The **sqrt** core opcode calculates the square root of a parameter.

It is a run-time error if **x** is negative.

The return value shall be .

5.5.4.11 sin

```
opcode sin(xsig x)
```

The **sin** core opcode calculates the sine of a parameter given in radians.

The return value shall be sin **x**.

5.5.4.12 cos

```
opcode cos(xsig x)
```

The **cos** core opcode calculates the cosine of a parameter given in radians.

The return value shall be cos **x**.

5.0.1.13 atan

```
opcode atan(xsig x)
```

The **atan** core opcode calculates the arctangent of a parameter , in radians.

The return value shall be $\tan^{-1}$ **x**, in the range $[0, \pi)$.

5.5.4.14 pow

```
opcode pow(xsig x, xsig y)
```

The **pow** core opcode calculates the to-the-power-of operation.

It shall be a run-time error if **x** is negative and **y** is not an integer.

The return value shall be $\mathbf{x}^{\mathbf{y}}$**.**

5.5.4.15 log10

```
opcode log10(xsig x)
```

The **log10** core opcode calculates the base-10 logarithm of a parameter.

It is a run-time error if **x** is not strictly positive.

The return value shall be $\log_{10}$ **x**.

5.5.4.16 asin

```
opcode asin(xsig x)
```

The **asin** core opcode calculates the arcsine of a parameter, in radians.

It is a run-time error if **x** is not in the range [-1, 1].

The return value shall be $\sin^{-1}$ **x**, in the range $[0, \pi)$.

5.5.4.17 acos

```
opcode acos(xsig x)
```

The **acos** core opcode calculates the arccosine of a parameter, in radians.

It is a run-time error if **x** is not in the range [-1, 1].

The return value shall be $\cos^{-1}$ **x**, in the range $[0, \pi)$.

5.5.4.18 ceil

```
opcode ceil(xsig x)
```

The **ceil** core opcode calculates the ceiling of a parameter.

The return value shall be the smallest integer *y* such that $\mathbf{x} \leq y$.

5.5.4.19 floor

```
opcode floor(xsig x)
```

The **floor** core opcode calculates the floor of a parameter.

The return value shall be the greatest integer *y* such that $y \leq \mathbf{x}$.

5.5.4.20 min

```
opcode min(xsig x1[, xsig ...])
```

The **min** core opcode finds the minimum of a number of parameters.

The return value shall be the minimum value out of the parameter values.

5.5.4.21 max

```
opcode max(xsig x1[, xsig ...])
```

The **max** core opcode finds the maximum out of the parameter values.

The return value shall be the maximum value out of the parameter values.

5.5.5 Pitch converters

5.5.5.1 Introduction to pitch representations

There are four representations for pitch in a SAOL orchestra; the following twelve functions (after **gettune** and **settune**) convert them from one to another. The four representations are as follows:

- pitch-class, or **pch** representation. A pitch is represented as an integer part, which represents the octave number, where 8 shall be the octave containing middle C (C4); plus a fractional part, which represents the pitch-class, where .00 shall be C, .01 shall be C#, .02 shall be D, and so forth. Fractional parts larger than .11 (B) have no meaning in this representation; fractional parts between the pitch-class steps are rounded to the nearest pitch-class.

  For example, 7.09 is the A below middle C.

- octave-fraction, or **oct** representation. A pitch is represented as an integer part, which represents the octave number, where 8 shall be the octave containing middle C (C4); plus a fractional part, which represents a fraction of an octave, where each step of 0.16667 represents a semitone.

  For example, 7.75 is the A below middle C, in equal-tempered tuning.

- MIDI pitch number representation. A pitch is represented as an integer number of semitones from the bottom of the piano keyboard, where 60 shall be middle C (C4).

  For example, 57 is the A below middle C.

- Frequency, or **cps** representation. A pitch is represented as some number of cycles per second.

  For example, 220 Hz is the A below middle C.

Each of the pitch converters represents the conversion which is done by its name, with the new representation first and the original (parameter) representation second. Thus, **cpsmidi** is the converter which returns the frequency corresponding to a particular MIDI pitch.

5.5.5.2 gettune

```
opcode gettune()
```

The **gettune** core opcode returns the value in Hz of the current orchestra global tuning, which is the frequency of A above middle C. The global tuning shall be set by default to 440, but can be changed using the **settune** core opcode, Subclause 5.5.5.3 settune.

5.5.5.3 settune

```
kopcode settune(ksig x)
```

The **settune** core opcode sets and returns the value of the current orchestra global tuning. The global tuning is used by several pitch converters when converting between symbolic pitch representations and cycles-per-second representation.

It is a run-time error if **x** is not strictly positive. (Allowing a wide range for tuning parameters allows unusual "pitch" representations to be used).

This core opcode has side-effects, as follows: The global tuning variable shall be set to the value **x**.

The return value shall be **x**.

5.5.5.4 octpch

```
opcode octpch(xsig x)
```

The **octpch** core opcode converts pitch-class representation to octave representation, with regard to equal scale tempering.

It is a run-time error if **x** is not strictly positive.

Let the integer part of **x** be *y* and the fractional part of **x** be *z***.** Then, the return value shall be calculated as follows:

z shall be "rounded" to the nearest value such that 100*z* is an integer. If $z < 0$ or $z > 0.11$, then *z* shall be set to 0 instead.

Then, the return value shall be $y + 100z / 12$.

5.5.5.5 pchoct

```
opcode pchoct(xsig x)
```

The **pchoct** core opcode converts octave representation to pitch-class representation.

It is a run-time error if **x** is not strictly positive.

Let the integer part of **x** be *y* and the fractional part of **x** be *z***.** Then, the return value shall be calculated as follows:

z shall be rounded to the nearest value such that 12 *z* is an integer. Then, the return value shall be $y + 12z / 100$.

5.5.5.6 cpspch

```
opcode cpspch(xsig x)
```

The **cpspch** core opcode converts pitch-class representation to cycles-per-second representation, with regard to equal scale tempering and the global tuning.

It is a run-time error if **x** is not strictly positive.

Let the integer part of **x** be *y* and the fractional part of **x** be *z***.** Then, the return value shall be calculated as follows:

z shall be "rounded" to the nearest value such that 100*z* is an integer. If $z < 0$ or $z > 11$, then *z* shall be set to 0 instead.

Further let *t* be the global tuning parameter. Then, the return value shall be $t \times 2^{(y + 100z/12 - 8.75)}$.

5.5.5.7 pchcps

```
opcode pchcps(xsig x)
```

The **pchcps** core opcode converts cycles-per-second representation to pitch-class representation, with regard to the global tuning.

It is a run-time error if **x** is not strictly positive.

The return value shall be calculated as follows.

Let *t* be the global tuning parameter. Then, let *k* be $\log_2(\mathbf{x} / t) + 8.75$. Then, let the integer part of *k* be *y* and the fractional part of *k* be *z*, "rounded" to the nearest value such that 12*z* is an integer. The return value shall be $y + 12z / 100$.

5.5.5.8 cpsoct

```
opcode cpsoct(xsig x)
```

The **cpsoct** core opcode converts octave representation to cycles-per-second representation, with regard to the global tuning.

It is a run-time error if **x** is not strictly positive.

Let *t* be the global tuning value; then, the return value shall be $t \times 2^{(\mathbf{x} - 8.75)}$.

5.5.5.9 octcps

```
opcode octcps(asig x)
```

The **octcps** core opcode converts cycles-per-second representation to octave representation, with regard to the global tuning.

It is a run-time error if **x** is not strictly positive.

Let *t* be the global tuning value; then, the return value shall be $\log_2(\mathbf{x} / t) + 8.75$.

5.5.5.10 midipch

```
opcode midipch(asig x)
```

The **midipch** core opcode converts pitch-class representation to MIDI representation.

It is a run-time error if **x** is not strictly positive.

Let the integer part of **x** be *y* and the fractional part of **x** be *z***.** Then, the return value shall be calculated as follows:

z shall be "rounded" to the nearest value such that 100*z* is an integer. If $z < 0$ or $z > 0.11$, then *z* shall be set to 0 instead.

The return value shall be $60 + 100z + 12 (y - 8)$.

5.5.5.11 pchmidi

```
opcode pchmidi(asig x)
```

The **midipch** core opcode converts MIDI representation to pitch-class representation.

It is a run-time error if **x** is not strictly positive.

The return value shall be calculated as follows: **x** shall be rounded to the nearest integer, then let *k* be (**x** – 60) / 12, and let *y* be the integer part of *k*, and let *z* be the fractional part of *k*. Then, the return value shall be $y + 8 + 12z / 100$.

5.5.5.12 midioct

```
opcode midioct(asig x)
```

The **midioct** core opcode converts octave representation to MIDI representation.

It is a run-time error if **x** is not strictly positive.

The return value shall be calculated as follows. Let *k* be 12 (**x** – 8) + 60. Then, the value of *k* rounded to the nearest integer shall be the return value.

5.5.5.13 octmidi

```
opcode octmidi(xsig x)
```

The **octmidi** core opcode converts MIDI representation to octave representation.

It is a run-time error if **x** is not strictly positive.

The return value shall be (**x** – 60) / 12 + 8.

5.5.5.14 midicps

```
opcode midicps(xsig x)
```

The **midicps** core opcode converts cycles-per-second representation to MIDI representation, with regard to the global tuning.

It is a run-time error if **x** is not strictly positive.

Let *t* be the global tuning parameter, and let *k* be $12 \log_2 (\mathbf{x} / t) + 69$. Then, the return value shall be *k* rounded to the nearest integer.

5.5.5.15 cpsmidi

```
opcode cpsmidi(xsig x)
```

The **cpsmidi** core opcode converts MIDI representation to cycles-per-second representation, with regard to the global tuning and equal scale temperament.

It is a run-time error if **x** is not strictly positive.

Let *t* be the global tuning parameter. Then, the return value shall be $t \times 2^{(\mathbf{x} - \mathbf{69})/12}$.

5.5.6 Table operations

5.5.6.1 ftlen

```
opcode ftlen(table t)
```

The **ftlen** core opcode returns the length of a table. The length of a table is the value calculated based on the **size** parameter in the particular core wavetable generator as described in Subclause 5.6 SAOL core wavetable generator.

The return value shall be the length of the table referenced by **t**.

5.5.6.2 ftloop

```
opcode ftloop(table t)
```

The **ftloop** core opcode returns the loop start point of a wavetable. The loop point is set either in a sound sample data block in the bitstream, or by the **ftsetloop** core opcode (see Subclause 5.5.6.6 ftsetloop), or else it is 0.

The return value shall be the loop start point (in samples) of the wavetable referenced by **t**.

5.5.6.3 ftloopend

```
opcode ftloopend(table t)
```

The **ftloopend** core opcode returns the loop end point of a wavetable. The loop point is set either in a sound sample data block in the bitstream, or by the **ftsetend** core opcode (see Subclause 5.5.6.7 ftsetend), or else it is 0.

The return value shall be the loop end point (in samples) of the wavetable referenced by **t**.

5.5.6.4 ftsr

```
opcode ftsr(table t)
```

The **ftsr** core opcode returns the sampling rate of a wavetable. The sampling rate is set in a sound sample data block in the bitstream, or else it is 0.

The return value shall be the sampling rate, in Hz, of the wavetable referenced by **t**.

5.5.6.5 ftbasecps

```
opcode ftbasecps(table t)
```

The **ftbasecps** core opcode returns the base frequency of a wavetable, in cycles per second (Hz). The base frequency is set either in a sound sample data block in the bitstream, or in the core wavetable generator **sample** (Subclause 5.6.2 Sample), or by the core opcode **ftsetbase** (Subclause 5.5.6.8 ftsetbase), or else it is 0.

The return value shall be the base frequency, in Hz, of the wavetable referenced by **t**.

5.5.6.6 ftsetloop

```
kopcode ftsetloop(table t, ksig x)
```

The **ftbasecps** core opcode sets the loop start point of a wavetable to a new value, and returns the new value.

It is a run-time error if **x** < 0, or if **x** is larger than the size of the wavetable referenced by **t**.

This core opcode has side effects, as follows: the loop start point of the wavetable **t** shall be set to sample number **x**.

The return value shall be **x**.

5.5.6.7 ftsetend

```
kopcode ftsetend(table t, ksig x)
```

The **ftsetend** core opcode sets the loop end point of a wavetable to a new value, and returns the new value. It is a run-time error if **x** < 0, or if **x** is larger than the size of the wavetable referenced by **t**.

This core opcode has side effects, as follows: the loop end point of the wavetable **t** shall be set to sample number **x**.

The return value shall be **x**.

5.5.6.8 ftsetbase

```
kopcode ftsetbase(table t, ksig x)
```

The **ftsetbase** core opcode sets the base frequency of a wavetable to a new value, and returns the new value. It is a run-time error if **x** is not strictly positive.

This core opcode has side effects, as follows: the base frequency of the wavetable **t** shall be set to **x**, where **x** is a value in Hz.

The return value shall be **x**.

5.5.6.9 tableread

```
opcode tableread(table t, xsig index)
```

The **tableread** core opcode returns a single value from a wavetable. It is a run-time error if **x** < 0, or if **x** is larger than the size of the wavetable referenced by **t**.

The return value shall be the value of the wavetable **t** at sample number **index**, where sample number 0 is the first sample in the wavetable. If **index** is not an integer, then the return value shall be interpolated from nearby points of the wavetable, as described in Subclause 1.X, with a maximum passband ripple of 1 dB, a minimum stopband attenuation of 80 dB, and a maximum transition width of 10% of Nyquist.

5.5.6.10 tablewrite

```
opcode tablewrite(table t, xsig index, xsig val)
```

The **tablewrite** core opcode sets a single value in a wavetable, and returns that value. It is a run-time error if **index** < 0, or if **index** is larger than the size of the wavetable referenced by **t**.

This core opcode has side effects, as follows: **index** shall be rounded to the nearest integer, and the value of sample number **index** in the wavetable **t** shall be set to the new value **val**, where sample number 0 is the first sample in the wavetable.

The return value shall be **val**.

5.5.6.11 oscil

```
aopcode oscil(table t, sig freq[, ivar loops])
```

The **oscil** core opcode loops several times around the wavetable **t** at a rate of **freq** loops per second, returning values at the audio-rate. **loops** shall be rounded to the nearest integer when the opcode is evaluated. If **loops** is not provided, its value shall be –1.

It is a run-time error if **loops** is not strictly positive and is also not –1.

The return value is calculated according to the following procedure.

On the first a-rate call to **oscil** relative to a particular state, the *internal phase* shall be set to 0, and the *internal number of loops* set to **loops**. On subsequent calls, the internal phase shall be incremented by **freq/SR,** where **SR** is the orchestra sampling rate. If, after the incrementation, the internal phase is not in the interval [0,1] and the internal loop count is strictly positive, the phase shall be set to the fractional portion of its value ($p := p - \text{floor}(p)$) and the loop count decremented.

If the internal loop count is zero, the return value shall be 0. Otherwise the return value shall be the value of sample number x in the wavetable, where $x = p * l$, where p is the current internal phase, and l is the length of table **t**. If x is not an integer, then the value must be interpolated from the nearby table values, as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist.

NOTE

The **oscil** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **oscil** is referenced twice in the same a-cycle, then the effective loop frequency is twice as high as given by **freq**.

5.5.6.12 loscil

```
aopcode loscil(table t, sig freq[, ivar basefreq, ivar loopstart, ivar loopend])
```

This opcode loops around the wavetable **t**, returning values at the audio-rate. The looping continues as long as the opcode is active, and is performed at a special rate which depends on the base frequency **basefreq** and the sampling rate of the table. In this way, samples which were recorded at a particular known pitch may be interpolated to any other pitch.

If **basefreq** is not provided, it shall be set to the base frequency of the table **t** by default. If the table **t** has base frequency 0 and **basefreq** is not provided, it is a run-time error. If **basefreq** is not strictly positive, it is a runtime error. The **basefreq** parameter shall be specified in Hz.

If **loopstart** and **loopend** are not provided, they shall be set to the loop start point and loop end point of the table **t**, respectively. If **loopend** is not provided and the loop end point of **t** is 0, then it shall be set to the end of the table ($l - 1$, where l is the length of the table in sample points). If **loopstart** is not strictly less than **loopend**, or either is negative, it is a runtime error.

The return value is calculated according to the following procedure.

Let l be the length of the table, m be the value **loopstart** / l, and n be the value **loopend** / l. On the first a-rate call to **oscil** relative to a particular state, the *internal phase* shall be set to m. On subsequent calls, the internal phase shall be incremented by **freq * TSR / (basefreq *SR),** where **TSR** is the sampling rate of the table and **SR** is the orchestra sampling rate. If, after the incrementation, the internal phase is not in the interval [m,n], the phase shall be set to $m + \boldsymbol{p - kn}$, where p is the internal phase and k is the value floor(ph/n).

Otherwise the return value shall be the value of sample number x in the wavetable, where $x = \boldsymbol{p * l}$, where p is the current internal phase, and l is the length of table **t**. If x is not an integer, then the value must be interpolated from the nearby table values, as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist.

NOTE

The **loscil** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **loscil** is referenced twice in the same a-cycle, then the effective loop frequency is twice as high as given by **freq**.

5.5.6.13 doscil

```
aopcode doscil(table t)
```

The **doscil** core opcode plays back a sample once, with no frequency control or looping. It is useful for sample-rate matching sampled drum sounds to an orchestra rate.

The return value is calculated according to the following procedure.

On the first a-rate call to **oscil** relative to a particular state, the *internal phase* shall be set to 0. On subsequent calls, the internal phase shall be incremented by **TSR/SR,** where **TSR** is the sampling rate of the table **t** and **SR** is the orchestra sampling rate. If, after the incrementation, the internal phase is greater than 1, then the opcode is done.

If the opcode is done, the return value shall be 0. Otherwise the return value shall be the value of sample number *x* in the wavetable, where $x = p * l$, where *p* is the current internal phase, and *l* is the length of table **t**. If *x* is not an integer, then the value must be interpolated from the nearby table values, as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist.

NOTE

The **doscil** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **doscil** is referenced twice in the same a-cycle, then the sample is played back at twice its original frequency.

5.5.6.14 koscil

```
kopcode koscil(table t, ksig freq[, ivar loops])
```

This opcode loops several times around the wavetable **t** at a rate of **freq** loops per second, returning values at the control-rate. **loops** shall be rounded to the nearest integer when the opcode is evaluated. If **loops** is not provided, its value shall be set to –1.

It is a run-time error if **loops** is not strictly positive and is also not –1.

The return value is calculated according to the following procedure.

On the first k-rate call to **oscil** relative to a particular state, the *internal phase* shall be set to 0, and the *internal number of loops* set to **loops**. On subsequent calls, the internal phase shall be incremented by **freq/KR**, where **KR** is the orchestra control rate. If, after the incrementation, the phase is not in the interval [0,1]and the internal loop count is strictly positive, the phase shall be set to the fractional portion of its value ($p := p - \text{floor}(p)$) and the loop count decremented.

If the internal loop count is zero, the return value shall be 0. Otherwise the return value shall be the value of sample number *x* in the wavetable, where $x = p * l$, where *p* is the current internal phase, and *l* is the length of table **t**. If *x* is not an integer, then the value must be interpolated from the nearby table values, as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist.

NOTE

The **koscil** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **koscil** is referenced twice in the same k-cycle, then the effective loop frequency is twice as high as given by **freq**.

5.5.7 Signal generators

5.5.7.1 kline

```
kopcode kline(ivar x1, ivar dur1, ivar x2[, ivar dur2, ivar x3, …]);
```

The **kline** core opcode produces a line-segmented or "ramp" function, with values changing at the k-rate. This function takes **dur1** seconds to go from **x1** to **x2**, **dur2** seconds to go from **x2** to **x3**, and so on.

It is a run-time error if any of the following conditions apply:

- there are an even number of parameters
- any of the **dur** values are negative

The return value shall be calculated as follows:

On the first call to **kline** with regard to a particular state, the *internal time* shall be set to 0, the *current left point* to **x1**, the *current right point* to **x2**, and the *current duration* to **dur1**. On subsequent calls, the internal time shall be incremented by 1/**KR**, where **KR** is the orchestra control rate. So long as the internal time is thereby greater than the current duration and there is another duration parameter, the internal time shall be decremented by the current duration, the current duration shall be set to the next duration parameter, the current left point to the current right point, and the current right point to the next control point (x-value) (these steps repeat if necessary so long as the internal time is greater than the current duration). If there is no additional duration parameter, the generator is done.

If the generator is done, then the return value is 0. Otherwise, the return value is $l + (r - l)t/d$, where l is the current left point, r is the current right point, t is the internal time, and d is the current duration.

NOTE

The **kline** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **kline** is referenced twice in the same k-cycle, then the effective segment duration is half as long as given by the corresponding duration value.

5.5.7.2 aline

```
kopcode aline(ivar x1, ivar dur1, ivar x2[, ivar dur2, ivar x3, …]);
```

The **aline** core opcode produces a line-segmented or "ramp" function, with values changing at the a-rate. This function takes **dur1** seconds to go from **x1** to **x2**, **dur2** seconds to go from **x2** to **x3**, and so on.

It is a run-time error if any of the following conditions apply:

- there are an even number of parameters
- any of the **dur** values are negative

The return value shall be calculated as follows:

On the first call to **aline** with regard to a particular state, the *internal time* shall be set to 0, the *current left point* to **x1**, the *current right point* to **x2**, and the *current duration* to **dur1**. On subsequent calls, the internal time shall be incremented by 1/**SR**, where **SR** is the orchestra sampling rate. So long as the internal time is thereby greater than the current duration and there is another duration parameter, the internal time shall be decremented by the current duration, the current duration shall be set to the next duration parameter, the current left point to the current right point, and the current right point to the next control point (x-value) (these steps repeat if necessary so long as the internal time is greater than the current duration). If there is no additional duration parameter, the generator is done.

If the generator is done, then the return value is 0. Otherwise, the return value is $l + (r - l) \times t/d$, where l is the current left point, r is the current right point, t is the internal time, and d is the current duration.

NOTE

The **aline** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **aline** is referenced twice in the same a-cycle, then the effective segment duration is half as long as given by the corresponding duration value.

5.5.7.3 kexpon

```
kopcode kexpon(ivar x1, ivar dur1, ivar x2[, ivar dur2, ivar x3, …]);
```

The **kexpon** core opcode produces a segmented function made out of exponential curves, with values changing at the k-rate. This function takes **dur1** seconds to go from **x1** to **x2**, **dur2** seconds to go from **x2** to **x3**, and so on.

It is a run-time error if any of the following conditions apply:

- there are an even number of parameters
- any of the **dur** values are negative
- the **x** values are not all the same sign
- any **x** value is 0

The return value shall be calculated as follows:

On the first call to **kexpon** with regard to a particular state, the *internal time* shall be set to 0, the *current left point* to **x1**, the *current right point* to **x2**, and the *current duration* to **dur1**. On subsequent calls, the internal time shall be incremented by 1/**KR**, where **KR** is the orchestra control rate. . So long as the internal time is thereby greater than the current duration and there is another duration parameter, the internal time shall be decremented by the current duration, the current duration shall be set to the next duration parameter, the current left point to the current right point, and the current right point to the next control point (x-value) (these steps repeat if necessary so long as the internal time is greater than the current duration). If there is no additional duration parameter, the generator is done.

If the generator is done, then the return value is 0. Otherwise, the return value is $l\,(r / l)^{t/d}$, where l is the current left point, r is the current right point, t is the internal time, and d is the current duration.

NOTE

The **kexpon** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **kexpon** is referenced twice in the same k-cycle, then the effective segment duration is half as long as given by the corresponding duration value.

5.5.7.4 aexpon

```
aopcode aexpon(ivar x1, ivar dur1, ivar x2[, ivar dur2, ivar x3, …]);
```

The **aexpon** core opcode produces a segmented function made out of exponential curves, with values changing at the a-rate. This function takes **dur1** seconds to go from **x1** to **x2**, **dur2** seconds to go from **x2** to **x3**, and so on.

It is a run-time error if any of the following conditions apply:

- there are an even number of parameters
- any of the **dur** values are negative
- the **x** values are not all the same sign
- any **x** value is 0

The return value shall be calculated as follows:

On the first call to **aexpon** with regard to a particular state, the *internal time* shall be set to 0, the *current left point* to **x1**, the *current right point* to **x2**, and the *current duration* to **dur1**. On subsequent calls, the internal time shall be incremented by 1/**SR**, where **SR** is the orchestra sampling rate. . So long as the internal time is thereby greater than the current duration and there is another duration parameter, the internal time shall be decremented by the current duration, the current duration shall be set to the next duration parameter, the current left point to the current right point, and the current right point to the next control point (x-value) (these steps repeat if necessary so long as the internal time is greater than the current duration). If there is no additional duration parameter, the generator is done.

If the generator is done, then the return value is 0. Otherwise, the return value is $l\,(r / l)^{t/d}$, where l is the current left point, r is the current right point, t is the internal time, and d is the current duration.

NOTE

The **aexpon** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **aexpon** is referenced twice in the same a-cycle, then the effective segment duration is half as long as given by the corresponding duration value.

5.5.7.5 kphasor

```
kopcode kphasor(ksig cps)
```

The **kphasor** core opcode produces a moving phase value, looping from 0 to 1 repeatedly, **cps** times per second.

The return value shall be calculated as follows:

On the first call to **kphasor** with regard to a particular state, the *internal phase* shall be set to 0. On subsequent calls, the internal phase shall be incremented by **cps/KR**, where **R** is the orchestra control rate. If the internal phase is thereby not in the interval [0,1], the internal phase shall be set to the fractional part of its value (p = frac(p)). The return value is the internal phase.

NOTE

The **kphasor** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **kphasor** is referenced twice in the same k-cycle, then the effective frequency is twice as fast as given by **cps**.

5.5.7.6 aphasor

```
aopcode aphasor(asig cps)
```

The **aphasor** opcode produces a moving phase value, looping from 0 to 1 repeatedly, **cps** times per second.

The return value shall be calculated as follows:

On the first call to **aphasor** with regard to a particular state, the *internal phase* shall be set to 0. On subsequent calls, the internal phase shall be incremented by **cps**/**SR**, where **SR** is the orchestra sampling rate. If the internal phase is thereby not in the interval [0,1], the internal phase shall be set to the fractional part of its value (p = frac(p)). The return value is the internal phase.

NOTE

The **aphasor** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **aphasor** is referenced twice in the same a-cycle, then the effective frequency is twice as fast as given by **cps**.

5.5.7.7 pluck

```
aopcode pluck(asig cps, ivar buflen, table init, ksig atten, ksig smoothrate)
```

This opcode uses a simple form of the Karplus-Strong algorithm to generate plucked-string sounds by repeated sampling and smoothing of a buffer.

It is a run-time error if **buflen** is not strictly positive.

The return value is calculated as follows:

On the first call to **pluck** with regard to a particular opcode state, a *buffer* of length **buflen** shall be created and filled with the values from the table **init**, as follows. Let x be the length of the table **init**. If x is less than **bufval**, then the values of the buffer shall be set to the first **buflen** sample values of the table **init**. If x is greater than or equal to **buflen**, then the first **buflen** values of the buffer shall be set to the sample values in the table **init**, and the remainder of the buffer filled as described in this paragraph for the whole table. That is, as many full and partial cycles of the table are used as necessary to fill the buffer.

Also on the first call to **pluck** with regard to a particular state, the *internal phase* shall be set to 0, and the *smooth count* shall be set to 0.

On subsequent calls to **pluck** with regard to a state, the smooth count is incremented. If the smooth count is equal to **smoothrate**, the buffer shall be smoothed, as follows. A new buffer of length **buflen** shall be created, and its values set by averaging over the current buffer. Each sample value in the new buffer shall be set to the value of the attenuated mean of the five surrounding samples of the current buffer. That is, for each sample x of the new buffer, its value shall be set to **atten** * (b[x-2] + b[x-1] + b[x] + b[x+1] + b[x+2])/5, where the b[.] notation refers to values of the current buffer, and the indices are calculated modulo **buflen** (that is, they "wrap around"). Then, the values of the current buffer shall be set to the values of the new buffer.

Whether or not the buffer has just been smoothed, the internal phase shall be incremented by **cps/SR**, where **SR** is the orchestra sampling rate, and if the resulting value is not in the interval [0,1], then the internal phase shall be set to the fractional part of the internal phase ($p = p$ – floor(p)).

The return value shall be the value of the buffer at the point p * **buflen**, where p is the internal phase. If this index is not an integer, the value must be interpolated from nearby buffer values, as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist.

NOTE

The **pluck** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **pluck** is referenced twice in the same a-cycle, then the effective frequency is twice as fast as given by **cps**.

5.5.7.8 buzz

```
aopcode buzz(asig cps, ksig nharm, ksig lowharm, ksig rolloff)
```

The **buzz** opcode produces a band-limited pulse train formed by adding together cosine overtones of a fundamental frequency **cps** given in Hz. These noisy sounds are useful as complex sound sources for subtractive synthesis.

lowharm gives the lowest harmonic used, where 0 is the fundamental, at frequency **cps**. It is a runtime error if **lowharm** is negative.

nharm gives the number of harmonics used starting from **lowharm**. If **nharm** is not strictly positive, then every overtone up to the orchestra Nyquist frequency is used (**nharm** shall be set to **SR** / 2 / **cps** – **lowharm**).

rolloff gives the multiplicative rolloff which defines the spectral shape. If **rolloff** is negative, then the partials alternate in phase; if |**rolloff**| > 1, then the partials increase in amplitude rather than attenuating.

The return value is calculated as follows. On the first call to **buzz** with regard to a particular scope, the *internal phase* shall be set to 0. On subsequent calls, the internal phase shall be incremented by **cps / SR**, where **SR** is the orchestra sampling rate. If, after this incrementation, the internal phase is greater than 1, the internal phase shall be set to the fractional part of its value (*p := frac(p)*).

The return value shall be

$$\sum_{f=\mathbf{lowharm}}^{\mathbf{lowharm}+\mathbf{nharm}} \mathbf{rolloff}^{(f-\mathbf{lowharm})} \cos 2\pi f p$$

where *p* is the internal phase.

5.5.7.9 fof

```
aopcode fof(asig f0, asig formant, table wave, table amp, ksig dur)
```

The **fof** opcode uses Rodet's FOF method [ref XXX] to synthesise a voiced vowel formant.

5.5.8 Noise generators

5.5.8.1 Note on noise generators and pseudo-random sequences

The following core opcodes generate noise, that is, pseudo-random sequences of various statistical properties. In order to provide maximum decorrelation among multiple noise generators, it is important that all references to pseudo-random generation share a single feedback state. That is, all random values required by the various states of various noise generators should make use of sequential values from a single "master" pseudo-random sequence.

It is strictly prohibited for an implementation to maintain multiple pseudo-random sequences to draw from (using the same algorithm) for various states of noise generation opcodes, because to do so may result in strong correlations between multiple noise generators.

This point does not apply to implementations which do not use “linear congruential”, “modulo feedback”, or similar mathematical structures to generate pseudo-random numbers.

The standard mathematical description of probability density functions is used in this Subclause. This means that if the pdf of a random variable *x* is *f*(*x*), then the probability of it taking a value in the range [*y*,*z*] is $\int_y^z f(x)dx$.

5.5.8.2 irand

```
iopcode irand(ivar p)
```

The **irand** core opcode generates a random number from a linear distribution.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} 1/2\mathbf{p} : x \in [-\mathbf{p}, \mathbf{p}] \\ 0 : otherwise \end{cases}$$

5.5.8.3 krand

```
kopcode krand(ksig p)
```

The **krand** core opcode generates random numbers from a linear distribution.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} 1/2\mathbf{p} : x \in [-\mathbf{p}, \mathbf{p}] \\ 0 : otherwise \end{cases}$$

5.5.8.4 arand

```
aopcode arand(asig p)
```

The **arand** core opcode generates random noise according to a linear distribution.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} 1/2\mathbf{p} : x \in [-\mathbf{p}, \mathbf{p}] \\ 0 : otherwise \end{cases}$$

5.5.8.5 ilinrand

```
iopcode ilinrand(ivar p1, ivar p2)
```

The **ilinrand** core opcode generates a random number from a linearly-ramped distribution.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} \text{abs}(2 / (\mathbf{p2} - \mathbf{p1}) * [(x - \mathbf{p1}) / (\mathbf{p2} - \mathbf{p1})]) & \text{if } x \in [\mathbf{p1}, \mathbf{p2}] \\ 0 & \text{otherwise} \end{cases}$$

5.5.8.6 klinrand

```
kopcode klinrand(ksig p1, ksig p2)
```

The **klinrand** core opcode generates random numbers from a linearly-ramped distribution.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} \text{abs}(2 / (\mathbf{p2} - \mathbf{p1}) * [(x - \mathbf{p1}) / (\mathbf{p2} - \mathbf{p1})]) & \text{if } x \in [\mathbf{p1}, \mathbf{p2}] \\ 0 & \text{otherwise} \end{cases}$$

5.5.8.7 alinrand

```
aopcode alinrand(asig p1, asig p2)
```

The **alinrand** core opcode generates random noise from a linearly-ramped distribution.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} \text{abs}(2 / (\mathbf{p2} - \mathbf{p1}) * [(x - \mathbf{p1}) / (\mathbf{p2} - \mathbf{p1})]) & \text{if } x \in [\mathbf{p1}, \mathbf{p2}] \\ 0 & \text{otherwise} \end{cases}$$

5.5.8.8 iexprand

```
iopcode iexprand(ivar p1)
```

The **iexprand** core opcode generates a random number from a exponential distribution with mean **p1**. It is a run-time error if **p1** is not strictly positive.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} 0 & \text{if } x \le 0, \text{ or} \\ k \exp(-kx), \text{ where } k = 1 / \mathbf{p1}, & \text{otherwise.} \end{cases}$$

5.5.8.9 kexprand

```
kopcode kexprand(ksig p1)
```

The **kexprand** core opcode generates random numbers from an exponential distribution with mean **p1**. It is a run-time error if **p1** is not strictly positive.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \begin{cases} 0 & \text{if } x \le 0, \text{ or} \\ k \exp(-kx), \text{ where } k = 1 / \mathbf{p1}, & \text{otherwise.} \end{cases}$$

5.5.8.10 aexprand

```
aopcode aexprand(asig p1)
```

The **aexprand** core opcode generates random noise according to an exponential distribution with mean **p1**. It is a run-time error if **p1** is not strictly positive.

The return value shall be a random number *x* chosen according to the pdf

$p(x) =$	0	if $x \leq 0$, or
	$k \exp(-kx)$, where $k = 1 /$ **p1**,	otherwise.

5.5.8.11 kpoissonrand

```
kopcode kpoissonrand(ksig p1)
```

The **kpoissonrand** core opcode generates a random binary (0/1) sequence of numbers such that the mean time between 1's is **p1** seconds. It is a run-time error if **p1** is not strictly positive.

On the first call to **kpoissonrand** with regard to a particular opcode state, a random number *x* shall be chosen according to the pdf

$p(x) =$	0	If $x \leq 0$, or
	$k \exp(-kx)$, where $k = 1/$ (**p1 * KR)**,	otherwise.

where **KR** is the orchestra control rate.

The return value shall be 0 and the floor of this random value shall be stored.

On subsequent calls, the stored value shall be decremented by 1. If the decremented value is -1, the return value shall be 1 and a new random value shall be generated and stored as described above. Otherwise, the return value shall be 0.

NOTE

The **kpoissonrand** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **kpoissonrand** is referenced twice in the same k-cycle, then the effective mean time between 1 values is half as long as given by **t**.

5.5.8.12 apoissonrand

```
aopcode apoissonrand(asig p1)
```

The **apoissonrand** core opcode generates random binary (0/1) noise such that the mean time between 1's is **p1** seconds. It is a run-time error if **p1** is not strictly positive.

On the first call to **apoissonrand** with regard to a particular opcode state, a random number *x* shall be chosen according to the pdf

$p(x) =$	0	If $x \leq 0$, or
	$k \exp(-kx)$, where $k = 1 /$ (**p1 * SR)**,	otherwise.

where **SR** is the orchestra sampling rate.

The return value shall be 0 and the floor of this random value shall be stored.

On subsequent calls, the stored value shall be decremented by 1. If the decremented value is -1, the return value shall be 1 and a new random value shall be generated and stored as described above. Otherwise, the return value shall be 0.

NOTE

The **apoissonrand** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **apoissonrand** is referenced twice in the same a-cycle, then the effective mean time between 1 values is half as long as given by **t**.

5.5.8.13 igaussrand

```
iopcode igaussrand(ivar mean, ivar var)
```

The **igaussrand** core opcode generates a random number drawn from a Gaussian (normal) distribution with mean **mean** and variance **var**.

It is a run-time error if **var** is not strictly positive.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \frac{e^{-(\mathbf{mean}-x)^2/(2\mathbf{var})}}{\sqrt{2\pi \times \mathbf{var}}}$$

that is, *p(x)* ~ N(**mean**, **var**) where **mean** is the mean and **var** the variance of a normal distribution.

5.5.8.14 kgaussrand

```
kopcode kgaussrand(ksig mean, ksig var)
```

The **kgaussrand** core opcode generates random numbers drawn from a Gaussian (normal) distribution with mean **mean** and variance **var**.

It is a run-time error if **var** is not strictly positive.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \frac{e^{-(\mathbf{mean}-x)^2/(2\mathbf{var})}}{\sqrt{2\pi \times \mathbf{var}}},$$

that is, *p(x)* ~ N(**mean**, **var**) where **mean** is the mean and **var** the variance of a normal distribution.

5.5.8.15 agaussrand

```
aopcode agaussrand(asig mean, asig var)
```

The **agaussrand** core opcode generates random noise drawn from a Gaussian (normal) distribution with mean **mean** and variance **var**.

It is a run-time error if **var** is not strictly positive.

The return value shall be a random number *x* chosen according to the pdf

$$p(x) = \frac{e^{-(\mathbf{mean}-x)^2/(2\mathbf{var})}}{\sqrt{2\pi \times \mathbf{var}}},$$

that is, *p(x)* ~ N(**mean**, **var**) where **mean** is the mean and **var** the variance of a normal distribution.

5.5.9 Filters

5.5.9.1 port

```
kopcode port(ksig ctrl, ksig htime)
```

The **port** core opcode converts a step-valued control signal into a portamento signal. **ctrl** is an incoming control signal, and **htime** is the half-transition time in seconds to slide from one value to the next.

The return value is calculated as follows. On the first call to **port** with regard to a particular state, the *current value* and *old value* are both set to **ctrl**. On subsequent calls, if **ctrl** is not equal to the new value, then the old value is set to the current value, the *new value* is set to **ctrl** and the *current time* is set to 0. It is a run-time error if **ctrl** is 0 or has opposite sign from the current value.

If **htime** is 0, the current value is set to the new value.

The return value is calculated as follows. If the current value and new value are equal, then the return value is the new value. Otherwise, the current time is incremented by 1/**KR**, where **KR** is the orchestra control rate. Then, the current value shall be set to $o + (1 - [n - o]\, 2^{-t / \mathbf{htime}})$, where *t* is the current time, *n* is the new value, and *o* is the old value.

NOTE

The **port** opcode does not have a "proper" representation of time, but infers it from the number of calls. If the same state of **port** is referenced twice in the same k-cycle, then the effective half-transition time is half as long as given by **htime**.

5.5.9.2 hipass

```
aopcode hipass(asig input, ksig cut)
```

The **hipass** core opcode high-pass filters its input signal. **cut** is the –6 dB cutoff point of the filter, and is specified in Hz. It is a run-time error if **cut** is not strictly positive.

The particular method of high-pass filtering is not normative. Any filter with the specified characteristic may be used.

The return value shall be the result of filtering **input** with a high-pass filter at **cut**.

NOTE

The **hipass** opcode is not required to have a "proper" representation of time, but is permitted to infer it from the number of calls. If the same state of **hipass** is referenced twice in the same a-cycle, the result is unspecified.

5.5.9.3 lopass

```
aopcode lopass(asig input, ksig cut)
```

The **lopass** core opcode low-pass filters its input signal. **cut** is the –6 dB cutoff point of the filter, and is specified in Hz. It is a runtime error if **cut** is not strictly positive.

The particular method of low-pass filtering is not normative. Any filter with the specified characteristic may be used.

The return value shall be the result of filtering **input** with a low-pass filter at **cut**.

NOTE

The **lopass** opcode is not required to have a "proper" representation of time, but is permitted to infer it from the number of calls. If the same state of **lopass** is referenced twice in the same a-cycle, the result is unspecified.

5.5.9.4 bandpass

```
aopcode bandpass(asig input, ksig cf, ksig bw)
```

The **bandpass** core opcode band-pass filters its input signal. **cf** is the centre frequency of the passband, and is specified in Hz. **bw** is the bandwidth of the filter, measuring from the –6 dB cutoff point below the centre frequency to the –6 dB point above, and is specified in Hz. It is a runtime error if **cf** and **bw** are not both strictly positive.

The particular method of bandpass filtering is not normative. Any filter with the specified characteristic may be used.

The return value shall be the result of filtering **input** with a bandpass filter with centre frequency **cf** and bandwidth **bw**.

NOTE

The **bandpass** opcode is not required to have a "proper" representation of time, but is permitted to infer it from the number of calls. If the same state of **bandpass** is referenced twice in the same a-cycle, the result is unspecified.

5.5.9.5 bandstop

```
aopcode bandstop(asig input, ksig cf, ksig bw)
```

The **bandstop** core opcode band-stop (notch) filters its input signal. **cf** is the centre frequency of the stopband, and is specified in Hz. **bw** is the bandwidth of the filter, measuring from the –6 dB cutoff point below the centre frequency to the –6 dB point above, and is specified in Hz. It is a runtime error if **cf** and **bw** are not both strictly positive.

The particular method of notch filtering is not normative. Any filter with the specified characteristic may be used.

The return value shall be the result of filtering **input** with a bandstop filter with centre frequency **cf** and bandwidth **bw**.

NOTE

The **bandstop** opcode is not required to have a "proper" representation of time, but is permitted to infer it from the number of calls. If the same state of **bandstop** is referenced twice in the same a-cycle, the result is unspecified.

5.5.9.6 biquad

```
aopcode biquad(asig input, ivar b0, ivar b1, ivar b2, ivar a1, ivar a2)
```

The **biquad** core opcode performs exactly normative filtering using the canonical second-order filter in a "Transposed Direct Form II" structure. Using cascades of **biquad** opcodes allows the construction of arbitrary filters with exactly normative results.

The return value is calculated as follows. On the first call to **biquad** with regard to a particular state, the intermediate variables **ti, to, w0, w1,** and **w2** are set to 0. Then, on the first call and each subsequent call, the following pseudo-code defines the functionality:

```
ti := input + a1 * w1 + a2 * w2
to := b0 * ti + b1 * w1 + b2 * w2
w2 := w1
w1 := w0
w0 := ti
```

and the return value is **to**.

A runtime error is produced if this process produces out-of-bounds values (i.e., the filter is unstable).

The **biquad** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **biquad** is referenced twice in the same a-cycle, the effective sampling rate of the filter is twice as high as the orchestra sampling rate.

5.5.9.7 allpass

```
aopcode allpass(asig input, ivar time, ivar gain)
```

The **allpass** core opcode performs allpass filtering on an input signal. The length of the feedback delay is **time** and is specified in seconds. It is a run-time error if **time** is not strictly positive.

Let **t** be the value **time * SR**, where **SR** is the orchestra sampling rate. On the first call to **comb** with regard to a particular state, a delay line of length **t** is initialised and set to all zeros.

On the first and each subsequent call, let **x** be the value which was inserted into the delay line **t** calls ago, or 0 if there have not been **t** calls to this state. Insert the value **x** * **gain + input** into the beginning of the delay line. The output shall be **x – input * gain**.

NOTE

The **allpass** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **allpass** is referenced twice in the same a-cycle, then the effective allpass length is half as long as **len**.

5.5.9.8 comb

```
aopcode comb(asig input, ivar time, ivar gain)
```

The **comb** core opcode performs comb filtering on an input signal. The length of the feedback delay is **time** and is specified in seconds. It is a run-time error if **time** is not strictly positive.

Let **t** be the value **time * SR**, where **SR** is the orchestra sampling rate. On the first call to **comb** with regard to a particular state, a delay line of length **t** is initialised and set to all zeros.

On the first and each subsequent call, let **x** be the value which was inserted into the delay line **t** calls ago, or 0 if there have not been **t** calls to this state. Insert the value **x** * **gain** + **input** into the beginning of the delay line. The return value shall be **x**.

NOTE

The **comb** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. That is, if the same state of **comb** is referenced twice in the same a-cycle, the effective length is half of **t**.

5.5.9.9 fir

```
aopcode fir(asig input, ksig b0[, ksig b1, ksig b2, ksig …])
```

The **fir** core opcode applies a specified FIR filter of arbitrary order to an input signal. The particular method of implementing FIR filters is not specified and left open to implementors.

The parameters **b0, b1, b2, …** specify a FIR filter

$$H(z) = \mathbf{b0} + \mathbf{b1}\,\mathbf{z}^{-1} + \mathbf{b2}\,\mathbf{z}^{-2} + \ldots$$

The return value shall be the successive values given by the application of this filter to the signal given by the value of **input** in successive calls to **fir**.

NOTE

The **fir** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **fir** is referenced twice in the same a-cycle, the effective filter sampling rate is double that of the orchestra sampling rate.

5.5.9.10 iir

```
aopcode iir(asig input, ksig b0[, ksig a1, ksig b1, ksig a2, ksig b2, ksig …])
```

The **iir** core opcode applies a specified IIR filter of arbitrary order to an input signal. The particular method of implementing IIR filters is not specified and left open to implementors.

The parameters **b0, b1, b2, …** and **a1, a2, …** specify an IIR filter

$$H(z) = \frac{\mathbf{b0} + \mathbf{b1}z^{-1} + \mathbf{b2}z^{-2} + \cdots}{\mathbf{a1}z^{-1} + \mathbf{a2}z^{-2} + \cdots}.$$

The return value shall be the successive values of the signal given by the application of this filter to the signal given by **input** in successive calls to **iir**. It is a run-time error if this application produces out-of-range values (that is, if the filter is unstable).

NOTE

The **iir** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **iir** is referenced twice in the same a-cycle, the effective filter sampling rate is double that of the orchestra sampling rate.

5.5.9.11 firt

```
aopcode firt(asig input, table t[, ksig order])
```

The **firt** core opcode applies a specified FIR filter of arbitrary order, given in a table, to an input signal. The particular method of implementing FIR filters is not specified and left open to implementors.

The values stored in samples 0, 1, 2, … **order** of table **t** specify a FIR filter

$$H(z) = \mathbf{t}[0] + \mathbf{t}[1]\,\mathbf{z}^{-1} + \mathbf{t}[2]\,\mathbf{z}^{-2} + \ldots\ \mathbf{t}[\mathbf{order}\text{-}1]\,\mathbf{z}^{-\mathbf{order}+1}$$

where array notation is used to indicate wavetable samples. If **order** is not given or is greater than the size of the wavetable **t**, then **order** shall be set to the size of the wavetable. It is a run-time error if **order** is zero or negative.

The return values shall be the successive values of the signal given by the application of this filter to the signal given by the value of **input** in successive calls to **firt**.

NOTE

The **firt** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **firt** is referenced twice in the same a-cycle, the effective filter sampling rate is double that of the orchestra sampling rate.

5.5.9.12 iirt

```
aopcode iirt(asig input, table a, table b, ksig order)
```

The **iirt** core opcode applies a specified IIR filter of arbitrary order, given in two tables, to an input signal. The particular method of implementing IIR filters is not specified and left open to implementors.

The values stored in samples 1, 2, … **order** of table **a** and samples 0, 1, 2, …, **order** of wavetable **b** specify a IIR filter

$$H(z) = \frac{\mathbf{b}[0] + \mathbf{b}[1]z^{-1} + \mathbf{b}[2]z^{-2} + \cdots}{\mathbf{a}[1]z^{-1} + \mathbf{a}[2]z^{-2} + \cdots}$$

where array notation is used to indicate wavetable samples. (Note that sample 0 of wavetable **a** is not used). If **order** is not given or is greater than the size of the larger of the two wavetables, then **order** shall be set to the size of the greater of the two wavetables. If one wavetable is smaller than given by **order**, then the "extra" values shall be taken as zero coefficients It is a run-time error if **order** is zero or negative.

The return values shall be the successive values of the signal given by the application of this filter to the signal given by the value of **input** in successive calls to **iirt**. It is a run-time error if this application produces out-of-range values (that is, if the filter is unstable).

NOTE

The **iirt** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **iirt** is referenced twice in the same a-cycle, the effective filter sampling rate is double that of the orchestra sampling rate.

5.5.10 Spectral analysis

5.5.10.1 fft

```
specialop fft(asig input, table re, table im[, ivar len, ivar shift, ivar size,
table win])
```

The **fft** core opcode calculates windowed and overlapped DFT frames and places the complex-valued result in two tables. It is a "special opcode"; that is, it accepts values at the audio rate, but only returns then at the control rate.

There are several optional parameters. **len** specifies the length of the sample frame (the number of input samples to use). If **len** is zero or not provided, it is set to **SR/KR**, where **SR** is the orchestra sampling rate and **KR** is the orchestra control rate. **shift** specifies the number of samples by which to shift the analysis window. If **shift** is zero or not provided, it is set to **len**. **size** is the length of the DFT calculated by the opcode. If **size** is zero or not provided, it is set to **len**. **win** is the analysis window to apply to the analysis. If **win** is not provided, a boxcar window of length **len** is used.

It is a runtime error if any of the following apply: **len** is negative, **shift** is negative, **size** is negative, **win** has fewer than **len** samples, **re** has fewer than **size** samples, or **im** has fewer than **size** samples.

The calculation of this opcode is as follows: On the first call to the **fft** opcode with respect to a particular state, a holding buffer of length **len** is created. On each a-rate call to the opcode, the **input** sample is inserted into the buffer. When there are **len** samples in the buffer, the following steps are performed:

1. A new buffer is created of length **size**, for which each value **new[i]** is set to the value **buf[i]** * **win[i]**, where **new[i]** is the **i**th value of the new buffer, **buf[i]** is the value of the holding buffer, and **win[i]** is the value of the **i**th sample in the analysis-window wavetable. (The new buffer contains the pointwise product of the holding buffer and the analysis window). If **size > len**, then the values of **new[i]** for **i > len** are set to zero. If **size < len**, then only the first **size** values of the holding buffer are used.

2. The first **shift** samples are removed from the holding buffer and the remaining **len-shift** samples shifted to the front of the holding buffer. The **shift** samples at the end of the buffer after this shift are set to zero. If **shift** > **len**, the holding buffer is cleared.

3. The real DFT of the new buffer is calculated, resulting in a length-**size** complex vector of frequency-domain values. The real components of the DFT are placed in the first **size** samples of table **re**; the imaginary components of the DFT are placed in the first **size** samples of table **im**. The DFT is arranged such that the lowest frequencies, starting with DC, are at the zero point of the output tables, going up to the Nyquist frequency at **size/2**; the reflection of the spectrum from the Nyquist to the sampling frequency is placed in the second half of the tables.

The DFT is defined as

$$d[i] = \frac{\sum_{k=0}^{len} e^{-ijk/2\pi} x[k]}{\sqrt{2\pi}}$$

where **d[i]** are the resulting complex components of the DFT, 0 < **i** < **size**;
x[k] are the input samples, 0 < **k** < **len**;
and ***j*** is the square root of –1.

The return value on a particular k-cycle is 1 if a DFT was calculated since the last k-cycle, or 0 if one was not.

The **fft** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **fft** is referenced twice in the same a-cycle, then the effective FFT period is half as long as given by **len** and **shift**.

5.5.10.2 ifft

```
aopcode ifft(table re, table im[, ivar len, ivar shift, ivar size, table win])
```

The **ifft** core opcode calculates windowed and overlapped IDFTs and streams the result out as audio. **re** and **im** are wavetables which contain the real and imaginary parts of a DFT, respectively. There are several optional arguments which control the synthesis procedure. **len** is the number of output samples to use as audio; if **len** is zero or not given, it is taken as **SR**/**KR**, where **SR** is the orchestra sampling rate and **KR** is the orchestra control rate. **shift** is the number of samples by which the analysis window is shifted between frames; if **shift** is not given or is zero, it is taken as **len**. **size** is the size of the IDFT; if **size** is not given or is zero, it is taken as **len**. **win** is the synthesis window; if it is not given, a boxcar window of length **len** is assumed.

It is a run-time error if any of the following apply: **re** or **im** are shorter than length **size**, **win,** if given, is shorter than length **len**, or **len**, **shift**, or **size** are negative.

The calculation for this opcode is as follows. On the first call to **ifft** with respect to a particular state, the size **size** IDFT of the tables **re** and **im** is calculated. If **re** and/or **im** are longer than **size** samples, only the first **size** samples of these tables shall be used. The result of this IDFT is a sequence of **size** values, potentially complex-valued. The real components of the first **len** elements of this sequence are multiplied point-by-point by the corresponding samples of the window **win** and placed in an output buffer of length **len**. (**out[i] = seq[i] * win[i]** for 0 < **i** < **len**).

The IDFT is calculated with the assumption that the lowest-numbered elements of the tables **re** and **im** are the lowest frequencies of the audio signal, beginning with DC in sample 0, proceeding up to the Nyquist frequency in sample **size**/2, and then the reflected spectrum in samples **size**/2 up to **size**-1.

The IDFT is defined as

$$x[i] = \frac{\sum_{k=0}^{len} e^{ijk/2\pi} d[k]}{\sqrt{2\pi}}$$

where **d[i]** are the complex frequency components of the DFT, 0 < **i** < **size** (**d[i] = re[i] + *j* im[i]**)
x[k] are the input samples, 0 < **k** < **len**;
and ***j*** is the square root of –1.

Also on the first call to **ifft** with respect to a particular state, the output point of the synthesis is set to 0.

At each call to **ifft**, the following calculation is performed. The output value of the opcode is the value of the output buffer at the output point. Then, the output point is incremented. If the output point is thereby equal to **shift**, then the following steps shall be performed:

1. The first **shift** samples of the output buffer are discarded, the remaining **len-shift** samples of the output buffer are shifted into the beginning of the buffer, and the last **shift** samples are set to 0.

2. The IDFT of the current values of the **re** and **im** wavetables is calculated as described above. The first **len** values of the real part of the resulting audio sequence are multiplied point-by-point by the synthesis window **win**, and the result is added point-by-point to the output buffer (**out[i] = out[i] + seq[i] * win[i]** for $0 <$ **i** $<$ **len**).

3. The output point is set to 0.

NOTE

The **ifft** opcode shall not have a “proper” representation of table, but shall infer it from the number of calls. If the same state of **ifft** is referenced twice in the same a-cycle, the result is undefined.

EXAMPLE

The **ifft** and **fft** opcodes can be used together to write instruments that use FFT-based spectral modification techniques, with logical syntax at the instrument level, in which the FFT frame rate is asynchronous with the control rate. The structure of such an instrument is:

```
instr spec_mod() {
  asig out;
  ksig new_fft;
  ivar length;
  table t(empty,1025);

  length = 256;
  new_fft = fft(input,t,1024,length); // no windowing; place FFT in “t”
  if (new_fft) { // modify table data if there’s a new spectrum
     .
     .
     .
  }

  // and output IFFT
  out = ifft(out,t,1024,length);
  output(out);
  }
```

Thus, every 256 samples (assuming 256 is greater than the number of samples in the control period), we compute the 1024-point IFFT. On those k-cycles during which we compute the FFT, we modify the table values in some interesting way. The IFFT operator produces continuous output, where every 256 samples the new table data is inspected and the IFFT calculated

There is nothing preventing us from manipulating the table data every control period, but only those values present in the table at the IFFT times will actually be turned into sound.

FFTs and IFFTs do not need to be implemented in pairs; other methods (such as table calculations) can be used to generate spectra to be turned into sound with IFFT, or rudimentary audio-pattern-recognition tools can be constructed which compute functions of the FFT and return or export the results.

5.5.11 Gain control

5.5.11.1 rms

```
specialop rms(asig x[, ivar length])
```

The **rms** core opcode calculates the power in a signal. It is a “special opcode”; that is, it accepts values at the audio rate, but only returns them at the k-rate.

If **length** is not provided, it shall be set to the length of the control period. It is a run-time error if **length** is provided and is negative. The **length** parameter is specified in seconds.

The return value is calculated as follows. Let *l* be the value floor(**length * SR)**, where **SR** is the orchestra sampling rate. A buffer *b***[],** of length *l,* is maintained of the most recent values provided as the **x** parameter. Each control period, the RMS of these values is calculated as

$$p = \sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}$$

and the return value is *p*.

NOTE

The **rms** opcode shall not have a “proper” representation of time, but shall infer it from the number of calls. If the same state of **rms** is referenced twice in the same a-cycle, then the effective length is half as long as given by **length**.

5.5.11.2 gain

```
aopcode gain(asig x, ksig gain[, ivar length])
```

The **gain** core opcode attenuates or increases the amplitude of a signal to make its power equal to a specified power level.

If **length** is not provided, it shall be set to the length of the control period. It is a run-time error if **length** is provided and is not strictly positive. The **length** parameter is specified in seconds.

The return value is calculated as follows. Let *l* be the value floor(**length * SR)**, where **SR** is the orchestra sampling rate.

At the first call to the opcode, the attenuation level is set to 1. At each subsequent call, the input value **x** shall be stored in a buffer **b[]** of length ***l*.** When the buffer is full, the attenuation level is recalculated as

$$\frac{\text{gain}}{\sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}}$$

and the buffer is cleared.

The return value at each call is **x** * **A**, where **A** is the current attenuation level.

NOTE

The **gain** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **gain** is referenced twice in the same a-cycle, then the effective buffer length is half as long as given by **length**.

5.5.11.3 balance

```
aopcode balance(asig x, asig ref[, ivar length])
```

The **balance** core opcode attenuates or increases the amplitude of a signal to make its power equal to the power in a reference signal.

If **length** is not provided, it shall be set to the length of the control period. It is a run-time error if **length** is provided and is not strictly positive. The **length** parameter is specified in seconds.

The return value is calculated as follows. Let *l* be the value floor(**length * SR)**, where **SR**, is the orchestra sampling rate.

At the first call to the opcode, the attenuation level is set to 1. At each subsequent call, the input value **x** shall be stored in a buffer **b[]** of length *l***,** and the input value **ref** stored in a buffer **r**[] of length *l*. When the buffers are full, the attenuation level is recalculated as

$$\frac{\sqrt{\frac{\sum_{i=0}^{l-1} r[i]^2}{l}}}{\sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}}$$

and the buffer is cleared.

The return value at each call is **x** * **A**, where **A** is the current attenuation level.

NOTE

The **balance** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **balance** is referenced twice in the same a-cycle, then the effective buffer length is half as long as given by **length**.

5.5.11.4 compress

```
aopcode compress(asig x, table t)
```

The **compress** core opcode performs waveform compression, expansion, waveshaping, or other waveform manipulation of an input signal, by passing it through a lookup table.

The return value is calculated as follows: let *l* be the length of table **t**. Then, if the value **x** is greater than 1 or less than –1, it shall be clipped to 1 or –1 respectively. Then, let *p* be the value *l* * (**x** / 2 + 0.5). The

return value is the value of table **t** at sample *p*. If *p* is not an integer, the return value is interpolated from nearby points as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist..

5.5.11.5 pcompress

```
aopcode pcompress(asig x, table t[, ivar length])
```

The **pcompress** core opcode performs power-level compression, expansion, or other manipulation on an input signal.

If **length** is not provided, it shall be set to the length of the control period. It is a run-time error if **length** is provided and is not strictly positive. The **length** parameter is specified in seconds.

The return value is calculated as follows. Let *l* be the value floor(**length * SR)**, where **SR**, is the orchestra sampling rate.

At the first call to the opcode, the attenuation level is set to 1. At each subsequent call, the input value **x** shall be stored in a buffer **b[]** of length ***l*.** When the buffer is full, the value *p* is calculated as

$$p = \sqrt{\frac{\sum_{i=0}^{l-1} b[i]^2}{l}}$$

and the buffer is cleared. Let *k* be the length of table **t**. Then, if *p* is greater than 1 or less than –1, it is clipped to 1 or –1 respectively, and the new attenuation level shall be set as the value of the table **t** at the sample *k* * *p*. If this value is not an integer, the attenuation level shall be interpolated from nearby points of **t**, as described in Subclause 1.X, with a maximum passband ripple of 2.5 dB, a minimum stopband attenuation of 60 dB, and a maximum transition width of 10% of Nyquist..

The return value at each call is **x** * **A**, where **A** is the current attenuation level.

NOTE

The **pcompress** opcode shall not have a “proper” representation of time, but shall infer it from the number of calls. If the same state of **pcompress** is referenced twice in the same a-cycle, then the effective buffer length shall be half as long as given by **length**.

5.5.11.6 sblock

```
specialop sblock(asig x, table t)
```

The **sblock** core opcode creates control-rate blocks of samples and places them in a wavetable. It is a “special opcode”; that is, it accepts values at the audio rate, but only returns them at the k-rate.

It is a run-time error if the table **t** is not allocated with as much space as there are samples in the control period of the orchestra.

The return value of this opcode is always 0.

This opcode has side effects, as follows. Let *k* be the number of samples in a control period. At each k-cycle, the most recent *k* values of **x** are placed in table **t** such that the oldest value is placed in sample 0.

NOTE

The **sblock** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **sblock** is referenced twice in the same a-cycle, then the samples placed in the table shall be the interleaved values given in the two calls during the second half of the k-period.

5.5.12 Sample conversion

5.5.12.1 decimate

```
specialop decimate(asig input)
```

The **decimate** core opcode decimates a signal from the audio rate to the control rate. It is a "special opcode"; that is, it accepts values at the audio rate, but only returns them at the k-rate.

The return value is calculated as follows. Each k-cycle, one of the values given as **input** in the preceding k-period of a-samples shall be returned.

NOTE

The **decimate** opcode is not required to have a "proper" representation of time, but is allowed to infer it from the number of calls. If the same state of **decimate** is referenced twice in the same a-cycle, then the return value for each call at the subsequent k-cycle may be taken from any of the values provided to the state during the preceding k-period.

EXAMPLE

```
oparray decimate[2];
ksig a,b,c;

a = decimate[0](1);
b = decimate[0](0);
c = decimate[1](2);
```

The value of **a** and **b** at each k-cycle shall be either 0 or 1, in an implementation-dependant manner. The value of **c** shall be 2.

5.5.12.2 upsamp

```
asig upsamp(ksig input[, table win])
```

The **upsamp** core opcode upsamples a control signal to an audio signal. **win** is an optional interpolation window. If **win** is not provided, it is taken to be a boxcar window (all values equal 1) of length **SR / KR**, where **SR** is the orchestra sampling rate and **KR** is the orchestra control rate. If **win** is provided and is shorter than **SR / KR** samples, it is zero-padded at the end to length **SR/KR** for use in this opcode.

On the first call to **upsamp** with regard to a particular state, an output buffer of length **win** is created and set to zero. Also, the *output point* is set to 0.

On the first call to **upsamp** in a particular k-cycle with regard to a particular cycle, the output buffer is shifted by **SR / KR** samples: the first **SR / KR** samples are discarded, the remaining samples are shifted to the front of the output buffer, and the last **SR / KR** samples are set to 0. Then, the window function is scaled by **input** and added into the output buffer (**buf[i] = buf[i] + input * win[i],** 0 < **i** < length(**win**)). Then, the output point is set to 0.

On the first call and each subsequent call to **upsamp**, the return value is the value of the output buffer at the current output point. Then, the output point shall be incremented.

It is a run-time error if the same state of **upsamp** is referenced more times than the length of **win** in a single k-cycle.

5.5.12.3 downsamp

```
specialop downsamp(asig input[, table win])
```

The **downsamp** core opcode downsamples an audio signal to a control signal. It is a "special opcode"; that is, it accepts samples at the audio rate but only returns values at the control rate. **win** is an optional analysis window.

It is a run-time error if **win** is shorter than **SR * KR** samples, where **SR** is the orchestra sampling rate and **KR** is the orchestra control rate.

The return value is calculated as follows: at each k-cycle, the values of each sample of **input** provided in the previous a-cycle are placed in a buffer. If **win** is not provided, then the return value is the mean of the samples in the buffer. If **win** is provided, then the return value is calculated by multiplying the analysis window point-by-point with the input signal (**rtn** = Σ **input[i] * win[i]** for 0 < **i** < **SR * KR**, where **SR** is the orchestra sampling rate and **KR** is the orchestra control rate).

NOTE

The **decimate** opcode does not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **decimate** is referenced twice in the same a-cycle, then the return value is calculated from the input values in the second half of the k-cycle.

5.5.12.4 samphold

```
opcode samphold(sig input, ksig gate)
```

The **samphold** core opcode gates a signal with a control signal.

The return value is calculated as follows. On the first call to **samphold** with regard to a particular state, the last passed value is set to 0. If the value of **gate** is non-zero, then the last passed value is set to **input**. The last passed value is returned.

5.5.13 Delays

5.5.13.1 delay

```
aopcode delay(asig x, ivar t)
```

The **delay** opcode implements a fixed-length end-to-end (i.e., untapped) delay line. **t** gives the length of the delay line in seconds. It is a run-time error if **t** < 0, unless the terminal is running in a negative-time universe.

Let *y* be floor(**t** * **SR**) samples, where **SR** is the orchestra sampling rate. At each call to **delay** with respect to a particular opcode state, the value of **x** is inserted into a FIFO buffer of length *y*. The return value is the value which was inserted into the delay line *y* calls ago to **delay** with regard to the same state.

NOTE

The **delay** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **delay** is referenced twice in the same a-cycle, then the effective delay line is half as long as given by **t**.

5.5.13.2 delay1

```
aopcode delay1(asig x)
```

The **delay1** opcode implements a single-sample delay.

At each call to **delay1** with regard to a particular state, the value of **x** is stored, and the return value is the value stored on the previous call.

NOTE

The **delay1** opcode shall not have a "proper" representation of time, but shall infer it from the number of calls. If the same state of **delay1** is referenced twice in the same a-cycle, then the return value for the second call is the parameter value of the first.

5.5.13.3 fracdelay

```
aopcode fracdelay(ksig method[, xsig p1, xsig p2])
```

The **fracdelay** core opcode implements fractional, variable-length, and/or multitap delay lines. Several methods for manipulating the delay line are provided; in this way, **fracdelay** is like an object-oriented delay-line "class".

The semantics of **p1**and **p2** and the calculation of the return value differ depending on the value of **method**. It is a run-time error if **method** is less than 1 or greater than 4.

If **method** is 1, the "initialise" method is specified. In this case, **p1** is the length of the delay line in seconds. It is a run-time error if **p1** is not provided, or is less than 0. Any currently existing delay line in this opcode state shall be destroyed, a new delay line with the specified length (floor(**p1*** **SR**), where **SR** is the orchestra sampling rate) shall be created, and all values on this delay line shall be initialised to 0. The return value is 0. **p2** is not used, and is ignored if provided.

If **method** is 2, the "tap" method is specified. In this case, **p1** is the position of the tap in seconds. It is a run-time error if method 1 has not yet been called for this opcode state, or if **p1** is not provided, or if **p1** is less than 0, or if **p1** is greater than the most recent initialisation length. The return value is the current value of the delay line at position **p1** * **SR**, where **SR** is the orchestra sampling rate. If **p1 * SR** is not an integer, the return value must be interpolated from the nearby values, as described in Subclause 1.X, with a maximum passband ripple of 1 dB, a minimum stopband attenuation of 80 dB, and a maximum transition width of 10% of Nyquist. **p2** is not used, and is ignored if provided.

If **method** is 3, the "set" method is specified. In this case, **p1** is the position of the insertion in seconds, and **p2** is the value to insert. It is a run-time error if method 1 has not yet been called for this opcode state, or if **p1** is not provided, or if **p1** is less than 0, or if **p1**is greater than the most recent initialisation length, or if **p2** is not provided. The value of the delay line at position floor(**p1 * SR)**, where **SR** is the orchestra sampling rate, is updated to **p2**. The return value is 0.

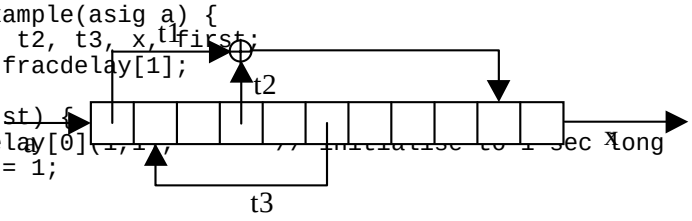
If **method** is 4, the "add into" method is specified. In this case, **p1** is the position of the insertion in seconds, and **p2** is the value to add in. It is a run-time error if method 1 has not yet been called for this

opcode state, or if **p2** is not provided. Let *x* be the current value of the delay line at position floor(**p1 * SR**), where **SR** is the orchestra sampling rate; then, the value of the delay line at this position is updated to *x* + **p2**. The return value is *x* + **p2**.

If **method** is 5, the "shift" method is specified. It is a run-time error if method 1 has not yet been called for this opcode state. All values of the delay line are shifted forward by one sample; that is, for each sample *x* where 0 < *x* <= L, where L is the length of the delay line, the new value of sample *x* of the delay line is the current value of sample *x* – 1. Sample 0 is set to value 0. The return value is the value shifted "off the end" of the delay line, that is the current value of sample L. **p1**and **p2** are not used, and are ignored if provided.

EXAMPLE

The following block diagram is implemented by the user-defined opcode which follows it. We assume that the orchestra sampling rate is 10 Hz for clarity.




```
aopcode example(asig a) {
  asig t1, t2, t3, x, first;
  oparray fracdelay[1];

  if (!first) {
    fracdelay[0](1,1,0);        // initialise to 1 sec long
    first = 1;
    }

  // flow network
  fracdelay[0](3,0,a);       // insert a at beginning
  t1 = fracdelay[0](2,0);    // tap at 0
  t2 = fracdelay[0](2,0.3);  // tap at 0.3
  t3 = fracdelay[0](2,0.5);  // tap at 0.5
  fracdelay[0](4,0.1,t3);    // feedback
  fracdelay[0](4,0.8,t1+t2); // feedforward
  x = fracdelay[0](5);       // shift and get output
  return(x);
}
```

Notice the use of the oparray construction to implement this network. If an oparray is not used, then each call to **fracdelay** refers to a different delay line, and the algorithm makes no sense. Also note that **fracdelay**, unlike **delay**, does not shift automatically. For "typical" operations, method 5 should be called once per a-cycle.

5.5.14 Effects

5.5.14.1 reverb

```
aopcode reverb(asig x, ivar f0[, ivar r0, ivar f1, ivar r1, ivar …])
```

The **reverb** core opcode produces a reverberation effect according to the given parameters.

It is a run-time error if any **f** or **r** value is negative, or if there are an even number of parameters greater than 2.

If only one value **f0** is given as an argument, it is taken as a full-range reverberation time, that is, the amount of time delay until the sound amplitude is attenuated 60 dB compared to the source sound (RT60).

If more values are given, the **f** – **r** pairs represent responses at different frequencies. At each frequency **f** given as a parameter, the reverberation time (RT60) at that frequency is given by the corresponding **r** value.

The exact method of calculating the reverberation according to the specified parameters is not normative. If content authors wish to have exactly normative reverberations, they can easily be authored using the **comb**, **allpass**, **biquad**, **delay**, **fracdelay**, and other strictly normative core opcodes (q.v.).

The output shall be the reverberated sound signal.

5.5.14.2 chorus

```
asig chorus(asig x, ksig rate, ksig depth)
```

The **chorus** core opcode creates a sound with a chorusing effect, with rate **rate** and depth **depth**, from the input sound **x**. **rate** is specified in cycles per second; **depth** is specified as percent excursion.

The exact method of chorusing is non-normative and left open to implementors.

5.5.14.3 flange

```
asig flange(asig x, ksig rate, ksig depth)
```

The **flange** core opcode creates a sound with a flanged effect, with rate **rate** and depth **depth**, from the input sound **x**. **rate** is specified in cycles per second; **depth** is specified as percent excursion.

The exact method of flanging is non-normative and left open to implementors.

5.6 SAOL core wavetable generators

5.6.1 Introduction

This Subclause describes each of the core wavetable generators in SAOL. All core wavetable generators shall be implemented in every terminal complying to Profile 4.

For each core wavetable generator, the following is described:

- A usage description, showing the parameters which must be provided in a table definition utilising this core wavetable generator.
- The normative semantics of the generator. These semantics describe how to calculate values and place them in the wavetable for each table definition using this generator.

For each core wavetable generator, the first field in the table definition is the name of the generator, and the value of the expression in the second field is the size of the wavetable. Many wavetable generators also allow the value –1 in this field to signify dynamic calculation of the wavetable size. If the size is not –1, and is also not strictly greater than zero, then the syntax of the generator call is illegal. In each case, the **size** parameter shall be rounded to the nearest integer before evaluating the semantics as described below.

The subsequent expressions are the required and optional parameters to the generator. For ease of exposition, each of these parameter fields will be given a name in the description of the generators, but there is no normative significance to these names. Parameter fields enclosed in brackets are optional and may or may not occur in a table definition using that generator.

Each wavetable, as well as a block of data, has four parameters associated with it: the sampling rate loop start, loop end, and base frequency. For all wavetable generators except **sample**, these parameters shall be set to zero initially.

5.6.2 Sample

```
t1 table(sample, size, which[, skip])
```

The **sample** core wavetable generator allows the inclusion of audio samples (or other blocks of data) in the bitstream and subsequent access in the orchestra.

If **size** is –1, then the size of the table shall be the length of the audio sample. If **size** is given, and larger than the length of the audio sample, then the audio sample shall be zero-padded at the end to length **size**. If **size** is given, and smaller than the length of the audio sample, only the first **size** samples shall be used.

The **which** field identifies a sample. It is either a symbol, in which case the generator refers to a sample in the bitstream, by symbol number; or a number, in which case the generator refers to a sample stored as an **AudioClip** in the BIFS scene graph (ISO 14496-1 Subclause XXX).

In the case where the generator refers to a sample in the bitstream, for compliant bitstream implementations, the sample data is simply a stream of raw floating-point values. This sample block of data shall be placed in the wavetable. If the bitstream sample data block contains sampling rate, loop start, loop end, and/or base frequency values, these parameters of the wavetable shall be set accordingly. If the sampling rate is not provided, it shall be set to the orchestra sampling rate by default. Any other parameters not so provided shall be set to 0.

In the case where the generator refers to a sample stored as an **AudioClip**, other audio coders described in this Part of ISO 14496 may be used to compress samples. The **children** fields of the **AudioSource** node responsible for instantiation of this orchestra refer to **AudioClip** nodes. Each **AudioClip** contains, after buffering as described in ISO 14496-1 Subclause XXX, several channels of audio data. If the first child has n_1 channels, the second n_2 channels, and so forth up to child *k*, then this **AudioSource** node has $K = n_0 + n_1 + ... + n_k$ channels in all, and **which** shall be a value between 0 and *K-1*. Channel **which** (where **which** is rounded to the nearest integer if necessary), numbering in order across children and their channels, shall be placed in the bitstream. The sampling rate of the wavetable shall be set to the sampling rate of the **AudioClip** node from which channel **which** is taken. The loop start, loop end, and base frequency values shall be set to 0.

If the **isReady** flag of the selected **AudioClip** node is not set when the generator is executed, then the bitstream is in error. That is, this form of this generator must only be used in cases where there is time allotted in the bitstream for the other decoders to produce samples (in real-time) before the generator executes. This is likely done by including the table generator in a score line scheduled to execute after the Composition Time (see ISO 14496-1 Subclause XXX) of the last audio Access Unit needed in the **AudioClip** node.

For standalone systems such as authoring tools, implementors are encouraged to provide access to other audio file formats and disk file access using this field (for example, to allow a filename as a string constant here). However, the only normative aspect is that in which the tokenised bitstream element for the generator refers to a **sample** element in the bitstream.

If **skip** is provided and is a positive value, it is rounded to the nearest integer, and the data placed in the wavetable begins with sample **skip**+1 of the bitstream or **AudioClip** sample data.

5.6.3 Data

```
t1 table(data, size, p1, p2, p3, ...)
```

The **data** core wavetable generator allows the orchestra to place data values directly into a wavetable.

If **size** is –1, then the size of the table shall be the number of data values specified. If **size** is given, and smaller than the number of data values, then the wavetable shall be zero-padded at the end to length **size**. If **size** is given, and larger than the number of data values, then only the first **size** values shall be used.

The **p1, p2, p3 ...** fields are floating-point values which shall be placed in the wavetable

5.6.4 Random

```
t1 table(random, size, dist, p1[, p2])
```

The **random** core wavetable generator fills a wavetable with pseudo-random numbers according to a given distribution. For all pseudo-random number generation algorithms, they shall be reseeded upon orchestra startup such that each execution of an orchestra containing these instructions generates different numbers.

If **size** is –1, the generator is illegal and a run-time error generated. If the **size** field is a positive value, then this shall be the length of the table, and this many independent random numbers shall be computed to place in the table.

The **dist** field specifies which random distribution to use, and the meanings of the **p1** and **p2** fields vary accordingly.

If **dist** is 1, then a uniform distribution is used. Pseudo-random numbers are computed such that all floating-point values between **p1** and **p2** inclusive have equal probability of being chosen for any sample.

If **dist** is 2, then a linearly ramped distribution is used. Pseudo-random numbers are computed such that the probability distribution function of choosing *x* for any sample is given by

p(x) = 0 if $x \leq$ **p1** or $x >$ **p2**, or
abs(2 / (**p2** – **p1**) x [(*x* – **p1**) / (**p2** – **p1**)]) otherwise.

A run-time error is generated if **dist** is 2 and **p1 = p2**.

If **dist** is 3, then an exponential distribution is used. Pseudo-random numbers are computed such that the probability distribution function of choosing *x* for any sample is

p(x) = 0 if $x \leq 0$, or
k exp(-*kx*), where $k = 1$ / **p1**, otherwise.

If **dist** is 3, then **p2** is not used and is ignored if it is provided.

If **dist** is 4, then a Gaussian distribution is used. Pseudo-random numbers are computed such that the probability distribution function of choosing x for any sample is

$$p(x) = \frac{e^{-(\mathbf{mean}-x)^2/(2\,\mathbf{var})}}{\sqrt{2\pi \times \mathbf{var}}},$$

that is, $p(x) \sim N(\mathbf{p1}, \mathbf{p2})$ where **p1** is the mean and **p2** the variance of a normal distribution.

If **dist** is 4, then **p2** shall be strictly greater than 0, otherwise a run-time error is generated.

If **dist** is 5, then a Poisson process is modelled, where the mean number of samples between 1's is given by an exponential distribution with mean **p1**. A pseudo-random value is computed according to $p(x)$ as given for **dist =** 3 (the exponential distribution), above. This value is rounded to the nearest integer *y*. The first *y* values of the table (elements 0 through *y*-1) are set to 0, and the next value (element *y*) to 1. Another pseudo-random value is computed as if **dist** =3, and rounded to the nearest integer *z*. The next *z* values (elements $y + 1$ through $y + z$ in the table) are set to 0, and the next value (element $y + z + 1$) to 1. This process is repeated until the table is full through element **size**. The resulting table has length **size** regardless of the values generated in the pseudo-random process; the last element may be a zero or 1.

If **dist** is 5, then **p2** is not used and is ignored if provided.

If **dist** is less than 0 or greater than 5, a run-time error is generated.

5.6.5 Step

```
t1 table(step, size, x1, y1, x2, y2, ...)
```

The **step** core wavetable generator allows arbitrary step functions to be placed in a wavetable. The step function is computed from pairs of (**x, y**) values.

If **size** is –1, then the size of the wavetable shall be the size of the largest x-value parameter. If **size** is larger than the largest x-value parameter, then the wavetable shall be padded with 0 values at the end to size **size**. If **size** is smaller than the largest x-value provided, then only the first **size** values shall be computed and used.

It is a run-time error if:

- **x1** is not 0,
- the x-values are not a non-decreasing sequence, or
- there are an even number of parameters, not counting the **size** parameter.

For the **step** generator, sample values 0 through **x2-**1 shall be set to **y1**, **x2** through **x3**-1 shall be set to **y2**, **x3** through **x4-1** shall be set to **y3**, and so forth.

5.6.6 Lineseg

```
t1 table(lineseg, size, x1, y1, x2, y2, ...)
```

The **lineseg** core wavetable generator allows arbitrary line-segment functions to be placed in a wavetable. The line segment function is computed from pairs of (**x**, **y**) values.

If **size** is –1, then the size of the wavetable shall be the size of the largest x-value parameter. If **size** is larger than the largest x-value parameter, then the wavetable shall be padded with 0 values at the end to size **size**. If **size** is smaller than the largest x-value provided, then only the first **size** values shall be computed and used.

It is a run-time error if:

- **x1** is not 0,
- the x-values are not a non-decreasing sequence, or
- there are an odd number of parameters, not counting the **size** parameter.

For the **step** generator, sample values for samples
x in the range **x1** through **x2** shall be set to **y1 + (y2-y1)**(*x* – **x1**) / (**x2** – **x1**)**,**
x in the range **x2** through **x3** shall be set to **y2 + (y3-y2)**(*x* **–** **x2**) / (**x3** – **x2**),

and so forth.

If any two successive x-values are equal, a discontinuous function is generated, and no values shall be calculated for the “range” corresponding to those values.

5.6.7 Expseg

```
t1 table(expseg, size, x1, y1, x2, y2, ...)
```

The **expseg** core wavetable generator allows arbitrary exponential-segment functions to be placed in a wavetable. The function is computed from pairs of (**x**, **y**) values.

If **size** is –1, then the size of the wavetable shall be the size of the largest x-value parameter. If **size** is larger than the largest x-value parameter, then the wavetable shall be padded with 0 values at the end to size **size**. If **size** is smaller than the largest x-value provided, then only the first **size** values shall be computed and used.

It is a run-time error if:

- **x1** is not 0,
- the x-values are not a non-decreasing sequence,
- the y-values are not all of the same sign,
- any y-value is equal to 0, or
- there are an odd number of parameters, not counting the **size** parameter.

For the **expseg** generator, sample values for samples
x in the range **x1** through **x2** shall be set to $\mathbf{y1}(\mathbf{y2}/\mathbf{y1})^{(x-\mathbf{x1})/(\mathbf{x2}-\mathbf{x1})}$,
x in the range **x2** through **x3** shall be set to $\mathbf{y2}(\mathbf{y3}/\mathbf{y2})^{(x-\mathbf{x1})/(\mathbf{x2}-\mathbf{x1})}$,

and so forth.

If any two successive x-values are equal, a discontinuous function is generated, and no values shall be calculated for the "range" corresponding to those values.

5.6.8 Cubicseg

```
t1 table(cubicseg, size, infl1, y1, x1, y2, infl2, y3, x2, y4, infl3,...)
```

The **cubicseg** core wavetable generator creates a function made up of segments of cubic polynomials. Each segment is specified in terms of endpoints and an inflection point. If, for successive segments, the y-values at the inflection points are between the y-values at the endpoints, then the function is smooth; otherwise, the function is pointy or "comb-like".

If **size** is –1, then the size of the wavetable shall be the size of the largest x-value parameter. If **size** is larger than the largest x-value parameter, then the wavetable shall be padded with 0 values at the end to size **size**. If **size** is smaller than the largest x-value provided, then only the first **size** values shall be computed and used.

It is a run-time error if:

- **infl1** is not 0,
- the x-values are not a non-decreasing sequence,
- any infl-value is not strictly between the two surrounding x-values,
- there are less than two x-values, or
- the sequence of control values does not end with an infl-value

For the **cubicseg** generator, sample values for samples
x in the range **infl1** to **infl2** shall be set to $\mathbf{a}x^3 + \mathbf{b}x^2 + \mathbf{c}x + \mathbf{d}$, where **a**, **b**, **c**, and **d** are the coefficients of a cubic polynomial which passes through (**infl1,y1**), (**x1,y2**), and (**infl2,y3**) and which has 0 derivative at **x1**;
x in the range **infl2** to **infl3** shall be set to $\mathbf{a}x^3 + \mathbf{b}x^2 + \mathbf{c}x + \mathbf{d}$, where **a**, **b**, **c**, and **d** are the coefficients of a cubic polynomial which passes through (**infl2,y3**), (**x2,y4**), and (**infl3,y5**) and which has 0 derivative at **x2**;
and so on.

If, for any segment, such a cubic polynomial does not exist or does not have real values through the segment range, it is a run-time error.

5.6.9 Spline

```
t1 table(spline, size, x1, y1, x2, y2, ...)
```

The **spline** core wavetable generator creates a smoothly varying "spline" function for a set of control points.

If **size** is –1, then the size of the wavetable shall be the size of the largest x-value parameter. If **size** is larger than the largest x-value parameter, then the wavetable shall be padded with 0 values at the end to size **size**. If **size** is smaller than the largest x-value provided, then only the first **size** values shall be computed and used.

It is a run-time error if:

- **x1** is not 0,
- the x-values are not a non-decreasing sequence,
– there are less than two x-values, or
- there are an odd number of parameters, not counting the **size** parameter.

For the **spline** generator, sample values for samples
x in the range **x1** to **x2** shall be set to $\mathbf{a}x^3 + \mathbf{b}x^2 + \mathbf{c}x + \mathbf{d}$, where **a**, **b**, **c**, and **d** are the coefficients of a cubic polynomial which passes through (**x1,y1**), and (**x2,y2**) and which has derivative 0 at **x1** and derivative **k2** at **x2**;
x in the range **x2** to **x3** shall be set to $\mathbf{a}x^3 + \mathbf{b}x^2 + \mathbf{c}x + \mathbf{d}$, where **a**, **b**, **c**, and **d** are the coefficients of a cubic polynomial which passes through (**x2,y2**), and (**x3,y3**) and which has derivative **k2** at **x2** and derivative **k3** at **x3**;
x in the range **x3** to **x4** shall be set to $\mathbf{a}x^3 + \mathbf{b}x^2 + \mathbf{c}x + \mathbf{d}$, where **a**, **b**, **c**, and **d** are the coefficients of a cubic polynomial which passes through (**x3,y3**), and (**x4,y4**) and which has derivative **k3** at **x3** and derivative **k4** at **x4**; and so on.

The derivative of the last cubic Subclause shall be zero at **xn**, the last x-point of the sequence.

If, for any segment, such a cubic polynomial does not exist or is not real-valued over the segment range, it is a run-time error.

5.6.10 Polynomial

```
t1 table(polynomial, size, xmin, xmax, a0, a1, a2, ...)
```

The **polynomial** core wavetable generator allows an arbitrary section of an arbitrary polynomial function to be placed in a wavetable. The polynomial function used is $p(x) = \mathbf{a0} + \mathbf{a1}x + \mathbf{a2}x^2 + ...$; it is evaluated over the range [**xmin**, **xmax**].

It is a run-time error if size is not strictly positive, or if there are not at least 3 parameters, not counting the **size** parameter, or if **xmin = xmax**.

For the **polynomial** generator, the sample value for sample *x* in the range [0,**size-1**] inclusive shall be set to

$\mathbf{a0} + \mathbf{a1}y + \mathbf{a2}y^2 + ...$, where y = **xmin** + (**size -** *x*) / **size** $\times$ (**xmax** – **xmin**).

5.6.11 Window

```
t1 table(window, size, type[, p])
```

The **window** core wavetable generator allows a windowing function to be placed in a table.

It is a run-time error if the **size** parameter is not strictly positive, or if **type** = 5 and the **p** parameter is not included.

The window type is specified by the **type** parameter. This parameter shall be rounded to the nearest integer, and then interpreted as follows:

If **type**=1, a Hamming window shall be used. For sample number *x* in the range [0, **size** – 1], the value placed in the table shall be

[XXX to be completed in editing]

If **type**=2, a Hanning (raised cosine) window shall be used. For sample number *x* in the range [0, **size** – 1], the value placed in the table shall be

$$\cos\left((x - \textbf{size}) / \textbf{size} \times \pi/2 \right).$$

If **type**=3, a Bartlett (triangular) window shall be used. For sample number *x* in the range [0, **size** – 1], the value placed in the table shall be

$$1 - | (\textbf{size}/2 - x) / (\textbf{size}/2) |.$$

If **type**=4, a Gaussian window shall be used. For sample number *x* in the range [0, **size** – 1], the value placed in the table shall be

$$\frac{e^{-(m-x)^2/(2v)}}{\sqrt{2\pi \times v}}$$, where m = **size**/2 and $v = (\textbf{size}/6)^{1/2}$.

If **type**=5, a Kaiser window shall be used, with parameter **p**. For sample number *x* in the range [0, **size** – 1], the value placed in the table shall be

[XXX to be completed in editing]

If **type**=6, a boxcar window shall be used. Each sample in the range [0, **size** – 1] shall be given the value 1.

5.6.12 Harm

```
t1 table(harm, size, f1, f2, f3...)
```

The **harm** generator creates one cycle of a composite waveform made up of a weighted sum of zero-phase sinusoids.

It is a run-time error if **size** is not strictly positive.

For each sample *x* in the range [0, **size** –1], the sample shall be assigned the value

$$\textbf{f1} \sin(2\pi x/\textbf{size}) + \textbf{f2} \sin(4\pi x/\textbf{size}) + \textbf{f3} \sin(6\pi x/\textbf{size}) + ...$$

5.6.13 Harm_phase

```
t1 table(harm_phase, size, f1, ph1, f2, ph2, ...)
```

The **harm_phase** core wavetable generator creates one cycle of a composite waveform made up of a weighted sum of zero-DC sinusoids, each with specified initial phase in radians.

It is a run-time error if **size** is not strictly positive, or if there are an odd number of parameters, not counting the **size** parameter.

For each sample *x* in the range [0, **size** –1], the sample shall be assigned the value

$$\textbf{f1} \sin(2\pi x/\textbf{size} + \textbf{ph1}) + \textbf{f2} \sin(4\pi x/\textbf{size} + \textbf{ph2}) + \textbf{f3} \sin(6\pi x/\textbf{size} + \textbf{ph3}) + ...$$

5.6.14 Periodic

```
t1 table(periodic, size, p1, f1, ph1, p2, f2, ph2, ...)
```

The **periodic** core wavetable generator creates one cycle of an arbitrary periodic waveform, parametrised as the sum of several sinusoids with arbitrary frequency, magnitude and phase. The phase values (**ph1, ph2, ...**) are specified in radians.

It is a run-time error if **size** is not strictly positive, or if the number of parameters, not counting the **size** parameter, is not evenly divisible by three.

For each sample *x* in the range [0, **size** –1], the sample shall be assigned the value

$$\mathbf{f1} \sin(2\ \mathbf{p1}\pi\ x/\mathbf{size} + \mathbf{ph1}) + \mathbf{f2} \sin(2\ \mathbf{p2}\ \pi\ x/\mathbf{size} + \mathbf{ph2}) + \mathbf{f3} \sin(2\ \mathbf{p3}\ \pi\ x/\mathbf{size} + \mathbf{ph3}) + ...$$

Any of the **p1**, **p2**, **p3**, etc. values may be zero, in which case the corresponding term of the calculation is a DC term; or non-integral, in which case there is a discontinuity at the table wrap point, or negative, which means the corresponding term evolves as a negative phase term. In all cases, the above value expression holds as specified.

5.6.15 Buzz

```
t1 table(buzz, size, numh, lowh, fl, r)
```

The **buzz** core wavetable generator creates one cycle of the sum of a series of spectrally-sloped cosine partials (band-limited pulse train). This waveform is a good source for subtractive synthesis.

It is a run-time error if **size** is not strictly positive, and **numh** is also not strictly positive.

lowh and **numh** shall be rounded to the nearest integer before further processing.

If **size** is not strictly positive, then the size of the table is given by the highest harmonic included, such that **size** = 2 (**lowh** + **numh**).

If **numh** is not strictly positive, then the number of harmonics shall be given by the size of the table, such that **numh** is the greatest integer smaller than **size/2** – **lowh**.

For each sample *x* in the range [0, **size** –1], the sample shall be assigned the value

If **fl** is negative, then alternating partials alternate phase direction; if |**r**| < 1, then partials attenuate as they get higher in frequency; otherwise, they stay the same or grow in magnitude; in all cases, the above value expression holds as specified.

5.6.16 Concat

```
table t1(concat, size, ft1, ft2, ...)
```

The **concat** generator allows several tables to be concatenated together into a new table.

It is a runtime error if no tables are provided as arguments.

If **size** is not strictly positive, the size of the wavetable shall be the sum of the sizes of the parameter wavetables. If **size** is strictly positive, but smaller than the sum of the sizes of the parameter wavetables, then only the first **size** points of the parameter wavetables shall be used. If **size** is larger than the sum of the sizes of the parameter wavetables, then the generated wavetables shall be zero-padded at the end to size **size**.

The values of the wavetable shall be calculated as follows: for each sample x in the range [0, s_1-1], where s_1 is the size of the wavetable referenced by **p1**, the sample shall be assigned the same value as sample x of **p1**; for each sample x in the range [s_1, s_1+s_2-1], where s_2 is the size of the wavetable referenced by **p2**, the sample shall be assigned the same value as sample $x - s_1$ of **p2**; and so on, up to sample **size**.

5.6.17 Empty

```
t1 table(empty,size)
```

The **empty** generator allocates space and fills it with zeros.

It is a run-time error if **size** is not strictly positive.

For each sample in the range [0,**size**-1], the sample is assigned value 0.

This generator is useful in conjunction with user-defined opcodes that fill up a table with data.

5.7 SASL syntax and semantics

5.7.1 Introduction

This Subclause describes the syntax and semantics of the score language SASL. SASL allows the simple parametric description of events which use an orchestra to generate sound, including notes, controllers, and dynamic wavetable generation. SASL is simpler than many previously existing score languages; this is intentional, as it enables easier cross-coding of score data from other formats into SASL. Since in many cases, SASL code is automatically generated by authoring tools, it is not a great disadvantage to have relatively simple syntax and few "defaults".

As with the SAOL description in Subclause 5.4 SAOL syntax and semantics, this Subclause describes a textual representation of SASL which is standardised, but stands outside of the bitstream-decoder relationship. It also describes the mapping between the textual representation and the bitstream representation. The exact normative semantics of SASL will be described in reference to the textual representation, but also apply to the tokenised bitstream representation as created via the normative tokenisation mapping.

All times in the score file (start times and durations) are specified in **score time**, which is measured in **beats**. By default, the score time is equivalent to the absolute time, and thus events with duration of one beat last one second, and an event dispatched two beats of score time after another is dispatched two seconds later by the scheduler. However, this mapping can be changed with the **tempo** command, see below.

[What happens if an event gets received in streaming score data and the time has already gone by?]

5.7.2 Syntactic Form

<score file> -> <score line> [<score file>]
<score file> -> <score line>

<score line> -> <instr line> **<newline>**
<score line> -> <control line> **<newline>**
<score line> -> <tempo line> **<newline>**
<score line> -> <table line> **<newline>**
<score line> -> <end line> **<newline>**

<instr line> -> [**<ident> :**] **<number> <ident> <number>** <pflist>

<control line> -> **<number>** [**<ident>**] **control <ident> <number>**

<tempo line> -> **<number> tempo <number>**

<table line> -> **<number> table <ident> <ident>** <pflist>

<end line> -> **<number> end**

<pflist> -> **<number>** [<pflist>]
<pflist> -> **<NULL>**

<number> as given in Subclause 5.4.2.3 Numbers.

<ident> as given in Subclause 5.4.2.2 Identifiers.

5.7.3 Instr line

The **instr** line specifies the construction of an instrument instantiation at a given time.

The first identifier, if given, is a label which is used to identify the instantiation for use with further control events.

The first number is the score time of the event. As much precision as desired may be used to specify times; however, instruments are only dispatched as fast as the orchestra control rate, as described in Subclause 5.3.3.3 Scheduler semantics. Event times do not have to be received, or present in the score file, in temporal order.

The second identifier (the first required identifier) is the name of the instrument, used to select one instrument from the orchestra described in the SAOL file. It is a syntax error if there is not an instrument with this name in the orchestra when the orchestra is started.

The second number is the score duration of the instrument instance. When the instrument instantiation is created, a termination event shall be scheduled (see Subclause 5.3.3.3 Scheduler semantics) at sum of the instantiation time and the duration. If this field is –1, then the instrument shall have no scheduled duration.

The pflist is the list of parameter fields to be passed to the instrument instance when it is created. If there are more pfields specified in the instrument declaration than elements of this list, the remaining pfields shall be set to 0 upon instantiation. If there are fewer pfields than elements, the extra elements shall be ignored.

5.7.4 Control line

The **control** line specifies a control instruction to be passed to the orchestra, or to a set of running instruments.

The first number is the score time of the control event. When this time arrives in the orchestra, the control event is dispatched according to its particular semantics.

The first identifier, if provided, is a label specifying which instrument instances are to receive the event. If this label is provided, when the control event is dispatched, any active instrument instances which were created by **instr** events with the same label receive the control event. If the label is provided, and there are no such active instrument instances, the control event shall be ignored. If the label is not provided, then the control event references a global variable of the orchestra.

The second identifier (the first required identifier) is the name of a variable which will receive the event. For labelled control lines, the name references a variable in instruments which were created based upon **instr** events with the same label. If there is no such name in a particular instrument instance, then the control event shall be ignored for that instance. For unlabelled lines, the name references a global variable of the orchestra with the same name. If there is no such global variable, then the control event shall be ignored.

The second number is the new value for the control variable. When the control event is dispatched, variables in the orchestra as identified in the preceding paragraph shall have their values set to this value.

5.7.5 Tempo line

The **tempo** line in the score specifies the new tempo for the decoding process. The tempo is specified in beats-per-minute; the default tempo shall be sixty beats per minute, and thus by default the score time is measured in seconds.

The first number in the tempo line is the score time at which the tempo changes. When this time arrives, the tempo event shall be dispatched as described in Subclause 5.3.3.3 Scheduler semantics, list item 7.

The second number is the new tempo, specified in beats per minute. Consequently, one beat lasts 60/**tempo** seconds, so that a tempo of 120 beats per minute is twice as fast as the default. When a tempo line is decoded, the time numbers in the score continue progressively, with the increments now in accordance with the new time unit.

5.7.6 Table line

The **table** line in the score specifies the creation or destruction of a wavetable.

The first number in the score line is the score time at which the wavetable is created or destroyed. For creation events, the wavetable shall be created at this time. For destruction events, the wavetable shall not be destroyed before this time.

The first identifier is the name of the wavetable. This name references a wavetable in the global orchestra scope.

The second identifier is either the name of the table generator, or the special name **destroy**. It is a syntax error if this identifier is not the name of one of the core wavetable generators listed in Subclause 5.6 SAOL core wavetable generator, or the special name **destroy**.

The pfield list is the list of parameters to the particular core wavetable generator. Not every sequence of parameters is legal for every table generator; see the definitions in Subclause 5.6 SAOL core wavetable generator.

The **sample** core wavetable generator refers to a sound sample (see Subclause 5.6.2 Sample). Implementations providing textual interfaces are suggested to provide access to commonly-used “soundfile” formats in the first pfield as a string constant. However, this is non-normative; the only normative aspect is as follows. In a bitstream **table** score line object, the **refers_to_sample** bit may be set. If this is the case, then the **sample** token of that score line object shall refer to another bitstream object containing the sample data, and it is this sample data which shall be placed in the wavetable.

When the dispatch time of the table event is received, if the table line references the **destroy** name, then any global wavetable with that name may be destroyed and its memory freed. If the table line specifies creation of a wavetable, and there is already a global wavetable with the same name, the new wavetable replaces the existing wavetable. That is, the global wavetable with that name may be destroyed and its memory freed.

When a new table is to be created, memory space is allocated for the table and filled with data according to the particular wavetable generator. Any reference to a wavetable with this name (including indirect references through import into a instrument instance) in existing or new instrument instances shall be taken as direction to the new wavetable.

NOTE

According to this paragraph, the wavetables referenced by running instrument instances shall be replaced upon dispatch of a **table** score line using the same name. That is, in the midst of the sound generation

process, when the **table** score line is dispatched, any table-reference opcodes in an instrument referencing that name shift reference to the new wavetable.

5.7.7 End line

The **end** line in the score specifies the end of the sound-generation process. The number given is the end time, in score time, for the orchestra. When this time is reached, the orchestra ceases, and all future Composition Buffers based on this Structured Audio decoding process contain only 0 values.

5.8 SAOL/SASL tokenisation

5.8.1 Introduction

This Subclause describes the normative process of mapping between the SAOL textual format used to describe syntax and semantics in Subclause 5.4 SAOL syntax and semantics, and the tokenised bitstream representation used in the bitstream definition in Subclause 5.1 Bitstream syntax and semantics. The textual representation stands outside of the bitstream-decoder relationship, and as such is not required to be implemented or used. The only aspect of SAOL decoding which is strictly normative is the process of turning a tokenised bitstream representation into sound as described in Subclause 5.3 Decoding process. However, it is highly recommended that implementations which allow access to bitstream contents use the textual representation described in Subclause 5.4 SAOL syntax and semantics rather than the tokenised representation. It is nearly impossible for a human reader to understand a SAOL program presented in tokenised format.

Annex D describes the analogous detokenisation process, for informative purposes only.

5.8.2 SAOL tokenisation

To tokenise a textual SAOL orchestra, the following steps shall be performed. First, the orchestra shall be divided into lexical elements, where a lexical element is one of the following:

1. A punctuation mark,
2. A reserved word (see Subclause 5.4.9 Reserved words),
3. A standard name (see Subclause 5.4.6.8 Standard names),
4. A core opcode name (see Subclause 5.5.3 List of core opcodes),
5. A core wavetable generator name (see Subclause 5.6 SAOL core wavetable generator),
6. A symbolic constant (a string, integer, or floating-point constant; see Subclause 5.4.2.3 Numbers), or
7. An identifier (see Subclause 5.4.2.2 Identifiers).

Whitespace (see Subclause 5.4.2.6 Whitespace) may be used to separate lexical elements as desired; in some cases, it is required in order to lexically disambiguate the orchestra. In neither case shall whitespace be treated as a lexical element of the orchestra. Comments (see Subclause 5.4.2.5 Comments) may be used in the textual SAOL orchestra but are removed upon lexical analysis; comments are not preserved through a tokenisation/detokenisation sequence.

After lexical analysis, all identifiers in the orchestra shall be numbered with symbol values, so that a single symbol is associated with a particular textual identifier. All identifiers which are textually equivalent (equal under string comparison) shall be associated with the same symbol regardless of their syntactic scope. This association of symbols to identifiers is called the *symbol table***.**

Using the lexical analysis and the symbol table, a tokenised representation of the orchestra may be produced. The lexical analysis is scanned in the order it was presented in the textual representation, and for each lexical element:

- If the element is of type (1) – (5) from above, the token value associated in the table in Annex A with that element shall be produced.

- If the element is of type (6) from above, one of the special tokens 0xF1, 0xF2, or 0xF3 shall be produced, depending on the type of the symbolic constant, and the succeeding bitstream element shall be the bitstream representation of the value.

- If the element is of type 7, the special token 0xF0 shall be produced, and the succeeding bitstream element shall be the symbol associated with the identifier in the symbol table.

After the sequence of lexical elements presented in the textual orchestra is tokenised, the special token 0xFF, representing end-of-orchestra, shall be produced.

5.8.3 SASL Tokenisation

A SASL score must be tokenised with respect to a particular SAOL orchestra, since the symbol values must correspond in order for the semantics to be according to the author's intent.

To tokenise a SASL file, the following steps are taken. First, the SASL file is divided into lexical elements, where each element is either an identifier, a reserved word, the name of a core wavetable generator, or a number. After lexical analysis, each identifier shall be associated with the appropriate symbol number from the SAOL orchestra reference. That is, for the associated SAOL orchestra, if there is an identifier in the orchestra equivalent to the identifier in the score, the identifier in the score shall receive the same symbol number that it received in the orchestra. If there is no such identifier in the orchestra, any unused symbol number may be assigned to the identifier in the score.

Using the lexical analysis and the symbol table, a tokenised representation of the orchestra may be produced. Each score line is taken in turn, in the order presented in the textual representation, and used to produce a **score_line** bitstream element, according to the semantics in Subclause 5.7 SASL syntax and semantics and the bitstream syntax for the various score elements, as given in Subclause 5.1.2 Bitstream syntax.

5.9 Sample Bank syntax and semantics

5.9.1 Introduction

This Subclause describes the operation of the Sample Bank synthesis method for both Profile 2 and Profile 4. In Profile 2, only Sample Bank and MIDI class types shall appear in the bitstream, and this Subclause describes the normative process of generating sound from a Sample Bank bitstream data element and a sequence of MIDI instructions. In Profile 4, Sample Banks are used in the context of a SAOL instrument as described in Subclause 5.4.6.6.8 Output, and this Subclause describes the normative process of generating sound and placing it on three busses, depending on the Sample Bank bitstream data element and the particular call to **sbsynth**.

5.9.2 Elements of bitstream

5.9.2.1 RIFF Structure

5.9.2.1.1 General RIFF File Structure

The RIFF (Resource Interchange File Format) is a tagged file structure developed for multimedia resource files, and is described in some detail in the Microsoft Windows 3.1 SDK Multimedia Programmer's Reference. The Tagged-file structure is useful because it helps prevent compatibility problems which can occur as the file definition changes over time. Because each piece of data in the file is identified by a standard header, an application that does not recognise a given data element can skip over the unknown information. The relevant information is provided in the bitstream description, Subclause 0.

A RIFF file is constructed from a basic building block called a "chunk." In 'C' syntax, a chunk is defined:

```
typedef DWORD FOURCC;      //  Four-character code
typedef struct {
  FOURCC ckID;             //  Chunk ID identifies type of data in the chunk.
  DWORD ckSize;            //  Size of chunk data in bytes, excluding pad byte.
  BYTE ckDATA[ckSize];     //  Actual data + a pad byte if req'd to word align.
};
```

Two types of chunks, the "RIFF" and "LIST" chunks, may contain nested chunks called subchunks as their data.

Within a given level of the hierarchy, the ordering of the chunks is arbitrary. Any chunk with an unknown chunk ID should be ignored.

5.9.2.1.2 The Sample Bank Chunks and Subchunks

A Structured Audio Sample Bank bitstream element comprises three chunks: an INFO-list chunk containing a number of required and optional subchunks describing the bitstream element, its history, and its intended use, an sdta-list chunk comprising a single subchunk containing any referenced digital audio samples, and a pdta-list chunk containing nine subchunks which define the articulation of the digital audio data.

5.9.2.1.3 Redundancy and Error Handling in the RIFF structure

The RIFF bitstream element structure contains redundant information regarding the length of the bitstream element and the length of the chunks and subchunks. This fact enables any reader of a SASBF bitstream element to determine if the bitstream element has been damaged by loss of data.

If any such loss is detected, the SASBF bitstream element is termed “structurally unsound” and in general should be rejected.

5.9.2.2 The INFO-list Chunk

The INFO-list chunk in a SASBF bitstream element contains three mandatory and a variety of optional subchunks as defined below. The INFO-list chunk gives basic information about the SASBF bank contained in the bitstream element.

5.9.2.2.1 The ifil Subchunk

The ifil subchunk is a mandatory subchunk identifying the SASBF specification version level to which the bitstream element complies. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the SASBF specification version. The WORD wMinor contains the value to the right of the decimal point. For example, version 2.11 would be implied if wMajor=2 and wMinor=11.

These values can be used by applications which read SASBF bitstream elements to determine if the format of the bitstream element is usable by the program. Within a fixed wMajor, the only changes to the format will be the addition of Generator, Source and Transform enumerators, and additional info subchunks. These are all defined as being ignored if unknown to the program.

If the ifil subchunk is missing, or its size is not four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.2.2 The isng Subchunk

The isng subchunk is a mandatory subchunk identifying the wavetable sound engine for which the bitstream element was optimised. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even.

The ASCII should be treated as case-sensitive. In other words “engine” is not the same as “ENGINE.”

The isng string may be optionally used by chip drivers to vary their synthesis algorithms to emulate the target sound engine.

If the isng subchunk is missing not terminated in a zero valued byte, or its contents are an unknown sound engine, the field should be ignored.

5.9.2.2.3 The INAM Subchunk

The INAM subchunk is a mandatory subchunk providing the name of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical inam subchunk would be the fourteen bytes representing “General MIDI” as twelve ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “General MIDI” is not the same as “GENERAL MIDI.”

If the inam subchunk is missing, or not terminated in a zero valued byte, the field should be ignored and the user supplied with an appropriate error message if the name is queried. If the bitstream element is re-written, a valid name should be placed in the INAM field.

5.9.2.2.4 The irom Subchunk

The irom subchunk is an optional subchunk identifying a particular wavetable sound data ROM to which any ROM samples refer. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical irom field would be the six bytes representing “1MGM” as four ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “1mgm” is not the same as “1MGM.”

The irom string is used by drivers to verify that the ROM data referenced by the bitstream element is available to the sound engine.

If the irom subchunk is missing, not terminated in a zero valued byte, or its contents are an unknown ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. A bitstream element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.5 The iver Subchunk

The iver subchunk is an optional subchunk identifying the particular wavetable sound data ROM revision to which any ROM samples refer. It is always four bytes in length, and contains data according to the structure:

```
struct sfVersionTag
{
       WORD wMajor;
       WORD wMinor;
};
```

The WORD wMajor contains the value to the left of the decimal point in the ROM version. The WORD wMinor contains the value to the right of the decimal point. For example, version 1.36 would be implied if wMajor=1 and wMinor=36.

The iver subchunk is used by drivers to verify that the ROM data referenced by the bitstream element is located in the exact locations specified by the sound headers.

If the iver subchunk is missing, not four bytes in length, or its contents indicate an unknown or incorrect ROM, the field should be ignored and the bitstream element assumed to reference no ROM samples. If ROM samples are accessed, any accesses to such instruments should be terminated and not sound. Note that for ROM samples to function correctly, both iver and irom must be present and valid. A bitstream

element should not be written which attempts to access ROM samples without both irom and iver present and valid.

5.9.2.2.6 The ICRD Subchunk

The ICRD subchunk is an optional subchunk identifying the creation date of the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICRD field would be the twelve bytes representing “May 1, 1995” as eleven ASCII characters followed by a zero byte.

Conventionally, the format of the string is “Month Day, Year” where Month is initially capitalised and is the conventional full English spelling of the month, Day is the date in decimal followed by a comma, and Year is the full decimal year. Thus the field should conventionally never be longer than 32 bytes.

The ICRD string is provided for library management purposes.

If the ICRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.7 The IENG Subchunk

The IENG subchunk is an optional subchunk identifying the names of any sound designers or engineers responsible for the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IENG field would be the twelve bytes representing “Tim Swartz” as ten ASCII characters followed by two zero bytes.

The IENG string is provided for library management purposes.

If the IENG subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.8 The IPRD Subchunk

The IPRD subchunk is an optional subchunk identifying any specific product for which the SASBF bank is intended. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical IPRD field would be the twelve bytes representing “MPEG SASBF” as ten ASCII characters followed by two zero bytes.

The ASCII should be treated as case-sensitive. In other words “mpeg sasbf” is not the same as “MPEG SASBF.”

The IPRD string is provided for library management purposes.

If the IPRD subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field’s contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.9 The ICOP Subchunk

The ICOP subchunk is an optional subchunk containing any copyright assertion string associated with the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICOP field would be the 38 bytes representing "Copyright (c) 1998 Content Developer" as 36 ASCII characters followed by two zero bytes.

The ICOP string is provided for intellectual property protection and management purposes.

If the ICOP subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.10 The ICMT Subchunk

The ICMT subchunk is an optional subchunk containing any comments associated with the SASBF bank. It contains an ASCII string of 65,536 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ICMT field would be the 40 bytes representing "This space unintentionally left blank." as 38 ASCII characters followed by two zero bytes.

The ICMT string is provided for including comments.

If the ICMT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.2.11 The ISFT Subchunk

The ISFT subchunk is an optional subchunk identifying the SASBF tools used to create and most recently modify the SASBF bank. It contains an ASCII string of 256 or fewer bytes including one or two terminators of value zero, so as to make the total byte count even. A typical ISFT field would be the twenty-six bytes representing "Editor 2.00a:Editor 2.00a" as twenty-five ASCII characters followed by a zero byte.

The ASCII should be treated as case-sensitive. In other words "Editor" is not the same as "EDITOR."

Conventionally, the tool name and revision control number are included first for the creating tool and then for the most recent modifying tool. The two strings are separated by a colon. The string should be produced by the creating program with a null modifying tool field (e.g. "Editor 2.00a:), and each time a tool modifies the bank, it should replace the modifying tool field with its own name and revision control number.

The ISFT string is provided primarily for error tracing purposes.

If the ISFT subchunk is missing, not terminated in a zero valued byte, or for some reason incapable of being faithfully copied as an ASCII string, the field should be ignored and if re-written, should not be copied. If the field's contents are not seemingly meaningful but can faithfully reproduced, this should be done.

5.9.2.3 The sdta-list Chunk

The sdta-list chunk in a SASBF bitstream element contains a single optional smpl subchunk which contains all the sound data associated with the SASBF bank. The smpl subchunk is of arbitrary length, and contains an even number of bytes.

5.9.2.3.1 Sample Data Format in the smpl Subchunk

The smpl subchunk, contains one or more samples of digital audio information in the form of linearly coded sixteen bit, signed, little endian (least significant byte first) words. Each sample is followed by a minimum of forty-six zero valued sample data points. These zero valued data points are necessary to guarantee that any reasonable upward pitch shift using any reasonable interpolator can loop on zero data at the end of the sound.

5.9.2.3.2 Sample Data Looping Rules

Within each sample, one or more loop point pairs may exist. The locations of these points are defined within the pdta-list chunk, but the sample data points themselves must comply with certain practices in order for the loop to be compatible across multiple platforms.

The loops are defined by “equivalent points” in the sample. This means that there are two sample data points which are logically equivalent, and a loop occurs when these points are spliced atop one another. In concept, the loop end point is never actually played during looping; instead the loop start point follows the point just prior to the loop end point. Because of the bandlimited nature of digital audio sampling, an artefact free loop will exhibit virtually identical data surrounding the equivalent points.

In actuality, because of the various interpolation algorithms used by wavetable synthesisers, the data surrounding both the loop start and end points may affect the sound of the loop. Hence both the loop start and end points must be surrounded by continuous audio data. For example, even if the sound is programmed to continue to loop throughout the decay, sample data points must be provided beyond the loop end point. This data will typically be identical to the data at the start of the loop. A minimum of eight valid data points are required to be present before the loop start and after the loop end.

The eight data points (four on each side) surrounding the two equivalent loop points should also be forced to be identical. By forcing the data to be identical, all interpolation algorithms are guaranteed to properly reproduce an artefact-free loop.

5.9.2.4 The pdta-list Chunk

5.9.2.4.1 The PHDR Subchunk

The PHDR subchunk is a required subchunk listing all presets within the SASBF bitstream element. It shall be a multiple of thirty-eight bytes in length, and shall contain a minimum of two records, one record for each preset and one for a terminal record according to the structure:

```
struct sfPresetHeader
{
       CHAR achPresetName[20];
       WORD wPreset;
       WORD wBank;
       WORD wPresetBagNdx;
       DWORD dwLibrary;
       DWORD dwGenre;
       DWORD dwMorphology;
};
```

The ASCII character field achPresetName shall contain the name of the preset expressed in ASCII, with unused terminal characters filled with zero valued bytes. Preset names are case-sensitive. A unique name shall always be assigned to each preset in the SASBF bank.

The WORD wPreset shall contain the MIDI Preset Number and the WORD wBank the MIDI Bank Number which apply to this preset. Note that the presets are not ordered within the SASBF bank. Presets

shall have a unique set of wPreset and wBank numbers. The special case of a General MIDI percussion bank is handled conventionally by a wBank value of 128. If the value in either field is not a valid MIDI value of 0 through 127, or 128 for wBank, the preset cannot be played but shall be maintained in memory for future renumbering.

The WORD wPresetBagNdx is an index to the preset's zone list in the PBAG subchunk. Because the preset zone list is in the same order as the preset header list, the preset bag indices shall be monotonically increasing with increasing preset headers. The size of the PBAG subchunk in bytes shall be equal to four times the terminal preset's wPresetBagNdx plus four. If the preset bag indices are non-monotonic or if the terminal preset's wPresetBagNdx does not match the PBAG subchunk size, the SASBF chunk is structurally defective and shall be rejected at load time. All presets except the terminal preset shall have at least one zone.

The DWORDs dwLibrary, dwGenre and dwMorphology are reserved for future implementation in a preset library management function and should be preserved as read, and created as zero.

The terminal sfPresetHeader record should never be accessed, and exists only to provide a terminal wPresetBagNdx with which to determine the number of zones in the last preset. All other values shall be conventionally zero, with the exception of achPresetName, which may be optionally be "EOP" indicating end of presets.

If the PHDR subchunk is missing, contains fewer than two records, or its size is not a multiple of 38 bytes, the SASBF bitstream element shall be rejected as structurally unsound.

5.9.2.4.2 The PBAG Subchunk

The PBAG subchunk is a required subchunk listing all preset zones within the SASBF bitstream element. It shall be a multiple of four bytes in length, and shall contain one record for each preset zone plus one record for a terminal zone according to the structure:

```
struct sfPresetBag
{
       WORD wGenNdx;
       WORD wModNdx;
};
```

The first zone in a given preset shall be located at that preset's wPresetBagNdx. The number of zones in the preset shall be determined by the difference between the next preset's wPresetBagNdx and the current wPresetBagNdx.

The WORD wGenNdx shall be an index to the preset's zone list of generators in the PGEN subchunk, and the wModNdx shall be an index to its list of modulators in the PMOD subchunk. Because both the generator and modulator lists are in the same order as the preset header and zone lists, these indices will be monotonically increasing with increasing preset zones. The size of the PMOD subchunk in bytes shall be equal to ten times the terminal preset's wModNdx plus ten and the size of the PGEN subchunk in bytes shall be equal to four times the terminal preset's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective PGEN or PMOD subchunks, the bitstream element is structurally defective and shall be rejected at load time.

If a preset has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not an Instrument generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks an Instrument generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the PBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.3 The PMOD Subchunk

The PMOD subchunk is a required subchunk listing all preset zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The preset zone's wModNdx points to the first modulator for that preset zone, and the number of modulators present for a preset zone is determined by the difference between the next higher preset zone's wModNdx and the current preset's wModNdx. A difference of zero indicates there are no modulators in this preset zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator with the same zone, the first modulator will be ignored.

Modulators in the PMOD subchunk act as additively relative modulators with respect to those in the IMOD subchunk. In other words, a PMOD modulator can increase or decrease the amount of an IMOD modulator.

In SASBF, no modulators have yet been defined, and the PMOD subchunk will always consist of ten zero valued bytes.

If the PMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.4 The PGEN Subchunk

The PGEN chunk is a required chunk containing a list of preset zone generators for each preset zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each preset zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined:

```
typedef struct
{
       BYTE byLo;
       BYTE byHi;
} rangesType;

typedef union
{
       rangesType ranges;
       SHORT shAmount;
       WORD wAmount;
} genAmountType;
```

The sfGenOper is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. This value indicates the type of generator being indicated. Note that this enumeration is two bytes in length.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers or MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The preset zone's wGenNdx points to the first generator for that preset zone. Unless the zone is a global zone, the last generator in the list is an "Instrument" generator, whose value is a pointer to the instrument associated with that zone. If a "key range" generator exists for the preset zone, it is always the first generator in the list for that preset zone. If a "velocity range" generator exists for the preset zone, it will only be preceded by a key range generator. If any generators follow an Instrument generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator with the same zone, the first generator will be ignored.

Generators in the PGEN subchunk are applied relative to generators in the IGEN subchunk in an additive manner. In other words, PGEN generators increase or decrease the value of an IGEN generator.

If the PGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in an instrument generator, the zone should be ignored. If the instrument generator value is equal to or greater than the terminal instrument, the bitstream element should be rejected as structurally unsound.

5.9.2.4.5 The INST Subchunk

The inst subchunk is a required subchunk listing all instruments within the SASBF bitstream element. It is always a multiple of twenty-two bytes in length, and contains a minimum of two records, one record for each instrument and one for a terminal record according to the structure:

```
struct sfInst
{
       CHAR achInstName[20];
       WORD wInstBagNdx;
};
```

The ASCII character field achInstName contains the name of the instrument expressed in ASCII, with unused terminal characters filled with zero valued bytes. Instrument names are case-sensitive. A unique name should always be assigned to each instrument in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of instruments with identical names, the instruments should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The WORD wInstBagNdx is an index to the instrument's zone list in the IBAG subchunk. Because the instrument zone list is in the same order as the instrument list, the instrument bag indices will be monotonically increasing with increasing instruments. The size of the IBAG subchunk in bytes will be four greater than four times the terminal (EOI) instrument's wInstBagNdx. If the instrument bag indices are non-monotonic or if the terminal instrument's wInstBagNdx does not match the IBAG subchunk size, the bitstream element is structurally defective and should be rejected at load time. All instruments except the terminal instrument must have at least one zone; any preset with no zones should be ignored.

The terminal sfInst record should never be accessed, and exists only to provide a terminal wInstBagNdx with which to determine the number of zones in the last instrument. All other values are conventionally zero, with the exception of achInstName, which should be "EOI" indicating end of instruments.

If the INST subchunk is missing, contains fewer than two records, or its size is not a multiple of 22 bytes, the bitstream element should be rejected as structurally unsound. All instruments present in the inst subchunk are typically referenced by a preset zone. However, a bitstream element containing any "orphaned" instruments need not be rejected. SASBF applications can optionally ignore or filter out these orphaned instruments based on user preference.

5.9.2.4.6 The IBAG Subchunk

The IBAG subchunk is a required subchunk listing all instrument zones within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one record for each instrument zone plus one record for a terminal zone according to the structure:

```
struct sfInstBag
{
       WORD wInstGenNdx;
       WORD wInstModNdx;
```

```
};
```

The first zone in a given instrument is located at that instrument's wInstBagNdx. The number of zones in the instrument is determined by the difference between the next instrument's wInstBagNdx and the current wInstBagNdx.

The WORD wInstGenNdx is an index to the instrument zone's list of generators in the IGEN subchunk, and the wInstModNdx is an index to its list of modulators in the IMOD subchunk. Because both the generator and modulator lists are in the same order as the instrument and zone lists, these indices will be monotonically increasing with increasing zones. The size of the IMOD subchunk in bytes will be equal to ten times the terminal instrument's wModNdx plus ten and the size of the IGEN subchunk in bytes will be equal to four times the terminal instrument's wGenNdx plus four. If the generator or modulator indices are non-monotonic or do not match the size of the respective IGEN or IMOD subchunks, the bitstream element is structurally defective and should be rejected at load time.

If an instrument has more than one zone, the first zone may be a global zone. A global zone is determined by the fact that the last generator in the list is not a sampleID generator. All generator lists must contain at least one generator with one exception - if a global zone exists for which there are no generators but only modulators. The modulator lists can contain zero or more modulators.

If a zone other than the first zone lacks a sampleID generator as its last generator, that zone should be ignored. A global zone with no modulators and no generators should also be ignored.

If the IBAG subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.7 The IMOD Subchunk

The IMOD subchunk is a required subchunk listing all instrument zone modulators within the SASBF bitstream element. It is always a multiple of ten bytes in length, and contains zero or more modulators plus a terminal record according to the structure:

```
struct sfModList
{
       SFModulator sfModSrcOper;
       SFGenerator sfModDestOper;
       SHORT modAmount;
       SFModulator sfModAmtSrcOper;
       SFTransform sfModTransOper;
};
```

The zone's wInstModNdx points to the first modulator for that zone, and the number of modulators present for a zone is determined by the difference between the next higher zone's wInstModNdx and the current zone's wModNdx. A difference of zero indicates there are no modulators in this zone.

The sfModSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. Modulators with sfModAmtSrcOper set to 'link' which have no other modulator linked to it are ignored. This value indicates the source of data for the modulator. Note that this enumeration is two bytes in length.

The sfModDestOper indicates the destination of the modulator. The destination is a value of one of the SFGenerator enumeration type values. Unknown or undefined values are ignored. Modulators with links which point to modulators which would exceed the total number of modulators for a given zone are ignored. Linked modulators that are part of circular links are ignored. Note that this enumeration is two bytes in length.

The SHORT modAmount is a signed value indicating the degree to which the source modulates the destination. A zero value indicates there is no fixed amount.

The sfModAmtSrcOper is one of the SFModulator enumeration type values. Unknown or undefined values are ignored. This enumerator indicates that the specified modulation source controls the degree to which the source modulates the destination. Note that this enumeration is two bytes in length.

The sfModTransOper is one of the SFTransform enumeration type values. Unknown or undefined values are ignored. This value indicates that a transform of the specified type will be applied to the modulation source before application to the modulator. Note that this enumeration is two bytes in length.

The terminal record conventionally contains zero in all fields, and is always ignored.

A modulator is defined by its sfModSrcOper, its sfModDestOper, and its sfModSrcAmtOper. All modulators within a zone must have a unique set of these three enumerators. If a second modulator is encountered with the same three enumerators as a previous modulator within the same zone, the first modulator will be ignored.

Modulators in the IMOD subchunk are absolute. This means that an IMOD modulator replaces, rather than adds to, a default modulator.

In SASBF, no modulators have yet been defined, and the IMOD subchunk will always consist of ten zero valued bytes.

If the IMOD subchunk is missing, or its size is not a multiple of ten bytes, the bitstream element should be rejected as structurally unsound.

5.9.2.4.8 The IGEN Subchunk

The IGEN chunk is a required chunk containing a list of zone generators for each instrument zone within the SASBF bitstream element. It is always a multiple of four bytes in length, and contains one or more generators for each zone (except a global zone containing only modulators) plus a terminal record according to the structure:

```
struct sfInstGenList
{
       SFGenerator sfGenOper;
       genAmountType genAmount;
};
```

where the types are defined as in the PGEN zone above.

The genAmount is the value to be assigned to the specified generator. Note that this can be of three formats. Certain generators specify a range of MIDI key numbers of MIDI velocities, with a minimum and maximum value. Other generators specify an unsigned WORD value. Most generators, however, specify a signed 16 bit SHORT value.

The zone's wInstGenNdx points to the first generator for that zone. Unless the zone is a global zone, the last generator in the list is a "sampleID" generator, whose value is a pointer to the sample associated with that zone. If a "key range" generator exists for the zone, it is always the first generator in the list for that zone. If a "velocity range" generator exists for the zone, it will only be preceded by a key range generator. If any generators follow a sampleID generator, they will be ignored.

A generator is defined by its sfGenOper. All generators within a zone must have a unique sfGenOper enumerator. If a second generator is encountered with the same sfGenOper enumerator as a previous generator within the same zone, the first generator will be ignored.

Generators in the IGEN subchunk are absolute in nature. This means that an IGEN generator replaces, rather than adds to, the default value for the generator.

If the IGEN subchunk is missing, or its size is not a multiple of four bytes, the bitstream element should be rejected as structurally unsound. If a key range generator is present and not the first generator, it should be ignored. If a velocity range generator is present, and is preceded by a generator other than a key range generator, it should be ignored. If a non-global list does not end in a sampleID generator, the zone should be ignored. If the sampleID generator value is equal to or greater than the terminal sampleID, the bitstream element should be rejected as structurally unsound.

5.9.2.4.9 The SHDR Subchunk

The SHDR chunk is a required subchunk listing all samples within the smpl subchunk and any referenced ROM samples. It is always a multiple of forty-six bytes in length, and contains one record for each sample plus a terminal record according to the structure:

```
struct sfSample
{
       CHAR achSampleName[20];
       DWORD dwStart;
       DWORD dwEnd;
       DWORD dwStartloop;
       DWORD dwEndloop;
       DWORD dwSampleRate;
       BYTE byOriginalPitch;
       CHAR chPitchCorrection;
       WORD wSampleLink;
       SFSampleLink sfSampleType;
};
```

The ASCII character field achSampleName contains the name of the sample expressed in ASCII, with unused terminal characters filled with zero valued bytes. Sample names are case-sensitive. A unique name should always be assigned to each sample in the SASBF bank to enable identification. However, if a bank is read containing the erroneous state of samples with identical names, the samples should not be discarded. They should either be preserved as read or, preferably, uniquely renamed.

The DWORD dwStart contains the index, in sample data points, from the beginning of the sample data field to the first data point of this sample.

The DWORD dwEnd contains the index, in sample data points, from the beginning of the sample data field to the first of the set of 46 zero valued data points following this sample.

The DWORD dwStartloop contains the index, in sample data points, from the beginning of the sample data field to the first data point in the loop of this sample.

The DWORD dwEndloop contains the index, in sample data points, from the beginning of the sample data field to the first data point following the loop of this sample. Note that this is the data point “equivalent to” the first loop data point, and that to produce portable artefact free loops, the eight proximal data points surrounding both the Startloop and Endloop points should be identical.

The values of dwStart, dwEnd, dwStartloop, and dwEndloop must all be within the range of the sample data field included in the SASBF bank or referenced in the sound ROM. Also, to allow a variety of hardware platforms to be able to reproduce the data, the samples have a minimum length of 48 data points, a

minimum loop size of 32 data points and a minimum of 8 valid points prior to dwStartloop and after dwEndloop. Thus dwStart must be less than dwStartloop-7, dwStartloop must be less than dwEndloop-31, and dwEndloop must be less than dwEnd-7. If these constraints are not met, the sound may optionally not be played if the hardware cannot support artefact-free playback for the parameters given.

The DWORD dwSampleRate contains the sample rate, in hertz, at which this sample was acquired or to which it was most recently converted. Values of greater than 50000 or less than 400 may not be reproducible by some hardware platforms and should be avoided. A value of zero is illegal. If an illegal or impractical value is encountered, the nearest practical value should be used.

The BYTE byOriginalPitch contains the MIDI key number of the recorded pitch of the sample. For example, a recording of an instrument playing middle C (261.62 Hz) should receive a value of 60. This value is used as the default “root key” for the sample, so that in the example, a MIDI key-on command for note number 60 would reproduce the sound at its original pitch. For unpitched sounds, a conventional value of 255 should be used. Values between 128 and 254 are illegal. Whenever an illegal value or a value of 255 is encountered, the value 60 should be used.

The CHAR chPitchCorrection contains a pitch correction in cents which should be applied to the sample on playback. The purpose of this field is to compensate for any pitch errors during the sample recording process. The correction value is that of the correction to be applied. For example, if the sound is 4 cents sharp, a correction bringing it 4 cents flat is required; thus the value should be -4.

The value in sfSampleType is an enumeration with eight defined values: monoSample = 1, rightSample = 2, leftSample = 4, linkedSample = 8, RomMonoSample = 32769, RomRightSample = 32770, RomLeftSample = 32772, and RomLinkedSample = 32776. It can be seen that this is encoded such that bit 15 of the 16 bit value is set if the sample is in ROM, and reset if it is included in the SASBF bank. The four LS bits of the word are then exclusively set indicating mono, left, right, or linked.

If the sound is flagged as a ROM sample and no valid “irom” subchunk is included; the bitstream element is structurally defective and should be rejected at load time.

If sfSampleType indicates a mono sample, then wSampleLink is undefined and its value should be conventionally zero, but will be ignored regardless of value. If sfSampleType indicates a left or right sample, then wSampleLink is the sample header index of the associated right or left stereo sample respectively. Both samples should be played entirely synchronously, with their pitch controlled by the right sample’s generators. All non-pitch generators should apply as normal; in particular the panning of the individual samples to left and right should be accomplished via the pan generator. Left-right pairs should always be found within the same instrument. Note also that no instrument should be designed in which it is possible to activate more than one instance of a particular stereo pair. The linked sample type is not currently fully defined in the SASBF specification, but will ultimately support a circularly linked list of samples using wSampleLink. Note that this enumeration is two bytes in length.

The terminal sample record is never referenced, and is conventionally entirely zero with the exception of achSampleName, which should be “EOS” indicating end of samples. All samples present in the smpl subchunk are typically referenced by an instrument, however a bitstream element containing any “orphaned” samples need not be rejected. SASBF applications can optionally ignore or filter out these orphaned samples according to user preference.

If the SHDR subchunk is missing, or its is size is not a multiple of 46 bytes the bitstream element should be rejected as structurally unsound.

5.9.3 Enumerators

5.9.3.1 Generator Enumerators

Subclause 7.1 defines the generator and generator kinds. Subclause 8.4 defines the generator operation model.

5.9.3.1.1 Kinds of Generator Enumerators

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

An Index Generator's amount is an index into another data structure. The only two Index Generators are Instrument and sampleID.

A Range Generator defines a range of note-on parameters outside of which the zone is undefined. Two Range Generators are currently defined, keyRange and velRange.

Substitution Generators are generators which substitute a value for a note-on parameter. Two Substitution Generators are currently defined, overridingKeyNumber and overridingVelocity.

Sample Generators are generators which directly affect a sample's properties. These generators are undefined at the preset level. The currently defined Sample Generators are the eight address offset generators, the sampleModes generator, the Overriding Root Key generator and the Exclusive Class generator.

Value Generators are generators whose value directly affects a signal processing parameter. Most generators are value generators.

5.9.3.1.2 Generator Enumerators Defined

The following is an exhaustive list of SASBF generators and their strict definitions:

0	startAddrsOffset	The offset, in sample data points, beyond the Start sample header parameter to the first sample data point to be played for this instrument. For example, if Start were 7 and startAddrsOffset were 2, the first sample data point played would be sample data point 9.
1	endAddrsOffset	The offset, in sample data points, beyond the End sample header parameter to the last sample data point to be played for this instrument. For example, if End were 17 and endAddrsOffset were -2, the last sample data point played would be sample data point 15.
2	startloopAddrsOffset	The offset, in sample data points, beyond the Startloop sample header parameter to the first sample data point to be repeated in the loop for this instrument. For example, if Startloop were 10 and startloopAddrsOffset were -1, the first repeated loop sample data point would be sample data point 9.
3	endloopAddrsOffset	The offset, in sample data points, beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. For example, if Endloop were 15 and endloopAddrsOffset were 2, sample data point 17 would be considered equivalent to the

		Startloop sample data point, and hence sample data point 16 would effectively precede Startloop during looping.
4	startAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Start sample header parameter and the first sample data point to be played in this instrument. This parameter is added to the startAddrsOffset parameter. For example, if Start were 5, startAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point played would be sample data point 65544.
5	modLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
6	vibLfoToPitch	This is the degree, in cents, to which a full scale excursion of the Vibrato LFO will influence pitch. A positive value indicates a positive LFO excursion increases pitch; a negative value indicates a positive excursion decreases pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will first rise 1 semitone, then fall one semitone.
7	modEnvToPitch	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence pitch. A positive value indicates an increase in pitch; a negative value indicates a decrease in pitch. Pitch is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 100 indicates that the pitch will rise 1 semitone at the envelope peak.
8	initialFilterFc	This is the cutoff and resonant frequency of the lowpass filter in absolute cent units. The lowpass filter is defined as a second order resonant pole pair whose pole frequency in Hz is defined by the Initial Filter Cutoff parameter. When the cutoff frequency exceeds 20kHz and the Q (resonance) of the filter is zero, the filter does not affect the signal.
9	initialFilterQ	This is the height above DC gain in centibels which the filter resonance exhibits at the cutoff frequency. A value of zero or less indicates the filter is not resonant; the gain at the cutoff frequency (pole angle) may be less than zero when zero is specified. The filter gain at DC is also affected by this parameter such that the gain at DC is reduced by half the specified gain. For example, for a value of 100, the filter gain at DC would be 5 dB below unity gain, and the height of the resonant peak would be 10 dB above the DC gain, or 5 dB above unity gain. Note also that if initialFilterQ is set to zero or less and the cutoff frequency exceeds 20 kHz, then the filter response is flat and unity gain.

10	modLfoToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation LFO will influence filter cutoff frequency. A positive number indicates a positive LFO excursion increases cutoff frequency; a negative number indicates a positive excursion decreases cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1200 indicates that the cutoff frequency will first rise 1 octave, then fall one octave.
11	modEnvToFilterFc	This is the degree, in cents, to which a full scale excursion of the Modulation Envelope will influence filter cutoff frequency. A positive number indicates an increase in cutoff frequency; a negative number indicates a decrease in filter cutoff frequency. Filter cutoff frequency is always modified logarithmically, that is the deviation is in cents, semitones, and octaves rather than in Hz. For example, a value of 1000 indicates that the cutoff frequency will rise one octave at the envelope attack peak.
12	endAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the End sample header parameter and the last sample data point to be played in this instrument. This parameter is added to the endAddrsOffset parameter. For example, if End were 65536, startAddrsOffset were -3 and startAddrsCoarseOffset were -1, the last sample data point played would be sample data point 32765.
13	modLfoToVolume	This is the degree, in centibels, to which a full scale excursion of the Modulation LFO will influence volume. A positive number indicates a positive LFO excursion increases volume; a negative number indicates a positive excursion decreases volume. Volume is always modified logarithmically, that is the deviation is in decibels rather than in linear amplitude. For example, a value of 100 indicates that the volume will first rise ten dB, then fall ten dB.
14	unused1	Unused, reserved. Should be ignored if encountered.
15	chorusEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the chorus effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the chorus effects processor.
16	reverbEffectsSend	This is the degree, in 0.1% units, to which the audio output of the note is sent to the reverb effects processor. A value of 0% or less indicates no signal is sent from this note; a value of 100% or more indicates the note is sent at full level. Note that this parameter has no effect on the amount of this signal sent to the “dry” or unprocessed portion of the output. For example, a value of 250 indicates that the signal is sent at 25% of full level (attenuation of 12 dB from full level) to the reverb effects processor.

17	pan	This is the degree, in 0.1% units, to which the “dry” audio output of the note is positioned to the left or right output. A value of -50% or less indicates the signal is sent entirely to the left output and not sent to the right output; a value of +50% or more indicates the signal is sent entirely to the right and not sent to the left. A value of zero sends the signal equally to left and right. For example, a value of -250 indicates that the signal is sent at 75% of full level to the left output and 25% of full level to the right output.
18	unused2	Unused, reserved. Should be ignored if encountered.
19	unused3	Unused, reserved. Should be ignored if encountered.
20	unused4	Unused, reserved. Should be ignored if encountered.
21	delayModLFO	This is the delay time, in absolute timecents, from key on until the Modulation LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second and a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
22	freqModLFO	This is the frequency, in absolute cents, of the Modulation LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 MHz would be 1200log2(.01/8.176) = -11610.
23	delayVibLFO	This is the delay time, in absolute timecents, from key on until the Vibrato LFO begins its upward ramp from zero value. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
24	freqVibLFO	This is the frequency, in absolute cents, of the Vibrato LFO’s triangular period. A value of zero indicates a frequency of 8.176 Hz. A negative value indicates a frequency less than 8.176 Hz; a positive value a frequency greater than 8.176 Hz. For example, a frequency of 10 mHz would be 1200log2(.01/8.176) = -11610.
25	delayModEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Modulation envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
26	attackModEnv	This is the time, in absolute timecents, from the end of the Modulation Envelope Delay Time until the point at which the Modulation Envelope value reaches its peak. Note that the attack is

		"convex"; the curve is nominally such that when applied to a decibel or semitone parameter, the result is linear in amplitude or Hz respectively. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
27	holdModEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
28	decayModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during decay phase. For the Modulation Envelope, the decay phase linearly ramps toward the sustain level. If the sustain level were zero, the Modulation Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
29	sustainModEnv	This is the decrease in level, expressed in 0.1% units, to which the Modulation Envelope value ramps during the decay phase. For the Modulation Envelope, the sustain level is properly expressed in percent of full scale. Because the volume envelope sustain level is expressed as an attenuation from full scale, the sustain level is analogously expressed as a decrease from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; values above 1000 are to be interpreted as 1000. For example, a sustain level which corresponds to an absolute value 40% of peak would be 600.
30	releaseModEnv	This is the time, in absolute timecents, for a 100% change in the Modulation Envelope value during release phase. For the Modulation Envelope, the release phase linearly ramps toward zero from the current level. If the current level were full scale, the Modulation Envelope Release Time would be the time spent in release phase until zero value were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
31	keynumToModEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave

		causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
32	keynumToModEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Modulation Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Modulation Envelope Hold Time were -7973 = 10 msec and the Key Number to Mod Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
33	delayVolEnv	This is the delay time, in absolute timecents, between key on and the start of the attack phase of the Volume envelope. A value of 0 indicates a 1 second delay. A negative value indicates a delay less than one second; a positive value a delay longer than one second. The most negative number (-32768) conventionally indicates no delay. For example, a delay of 10 msec would be 1200log2(.01) = -7973.
34	attackVolEnv	This is the time, in absolute timecents, from the end of the Volume Envelope Delay Time until the point at which the Volume Envelope value reaches its peak. Note that the attack is “convex”; the curve is nominally such that when applied to the decibel volume parameter, the result is linear in amplitude. A value of 0 indicates a 1 second attack time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates instantaneous attack. For example, an attack time of 10 msec would be 1200log2(.01) = -7973.
35	holdVolEnv	This is the time, in absolute timecents, from the end of the attack phase to the entry into decay phase, during which the Volume envelope value is held at its peak. A value of 0 indicates a 1 second hold time. A negative value indicates a time less than one second; a positive value a time longer than one second. The most negative number (-32768) conventionally indicates no hold phase. For example, a hold time of 10 msec would be 1200log2(.01) = -7973.
36	decayVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during decay phase. For the Volume Envelope, the decay phase linearly ramps toward the sustain level, causing a constant dB change for each time unit. If the sustain level were -96dB, the Volume Envelope Decay Time would be the time spent in decay phase. A value of 0 indicates a 1 second decay time for a zero sustain level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a decay time of 10 msec would be 1200log2(.01) = -7973.
37	sustainVolEnv	This is the decrease in level, expressed in centibels, to which the Volume Envelope value ramps during the decay phase. For the

		Volume Envelope, the sustain level is best expressed in centibels of attenuation from full scale. A value of 0 indicates the sustain level is full level; this implies a zero duration of decay phase regardless of decay time. A positive value indicates a decay to the corresponding level. Values less than zero are to be interpreted as zero; conventionally 1000 indicates full attenuation. For example, a sustain level which corresponds to an absolute value 12dB below of peak would be 120.
38	releaseVolEnv	This is the time, in absolute timecents, for a 100% change in the Volume Envelope value during release phase. For the Volume Envelope, the release phase linearly ramps toward zero from the current level, causing a constant dB change for each time unit. If the current level were full scale, the Volume Envelope Release Time would be the time spent in release phase until 96dB attenuation were reached. A value of 0 indicates a 1 second decay time for a release from full level. A negative value indicates a time less than one second; a positive value a time longer than one second. For example, a release time of 10 msec would be 1200log2(.01) = -7973.
39	keynumToVolEnvHold	This is the degree, in timecents per KeyNumber units, to which the hold time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
40	keynumToVolEnvDecay	This is the degree, in timecents per KeyNumber units, to which the decay time of the Volume Envelope is decreased by increasing MIDI key number. The hold time at key number 60 is always unchanged. The unit scaling is such that a value of 100 provides a hold time which tracks the keyboard; that is, an upward octave causes the hold time to halve. For example, if the Volume Envelope Hold Time were -7973 = 10 msec and the Key Number to Vol Env Hold were 50 when key number 36 was played, the hold time would be 20 msec.
41	instrument	This is the index into the INST subchunk providing the instrument to be used for the current preset zone. A value of zero indicates the first instrument in the list. The value should never exceed two less than the size of the instrument list. The instrument enumerator is the terminal generator for PGEN zones. As such, it should only appear in the PGEN subchunk, and it must appear as the last generator enumerator in all but the global preset zone.
42	reserved1	Unused, reserved. Should be ignored if encountered.
43	keyRange	This is the minimum and maximum MIDI key number values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid key. The

		keyRange enumerator is optional, but when it does appear, it must be the first generator in the zone generator list.
44	velRange	This is the minimum and maximum MIDI velocity values for which this preset zone or instrument zone is active. The LS byte indicates the highest and the MS byte the lowest valid velocity. The velRange enumerator is optional, but when it does appear, it must be preceded only by keyRange in the zone generator list.
45	startloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Startloop sample header parameter and the first sample data point to be repeated in this instrument's loop. This parameter is added to the startloopAddrsOffset parameter. For example, if Startloop were 5, startloopAddrsOffset were 3 and startAddrsCoarseOffset were 2, the first sample data point in the loop would be sample data point 65544.
46	keynum	This enumerator forces the MIDI key number to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
47	velocity	This enumerator forces the MIDI velocity to effectively be interpreted as the value given. This generator can only appear at the instrument level. Valid values are from 0 to 127.
48	initialAttenuation	This is the attenuation, in .4 centibel units, by which a note is attenuated below full scale. A value of zero indicates no attenuation; the note will be played at full scale. For example, a value of 60 indicates the note will be played at 2.4 dB below full scale for the note.
49	reserved2	Unused, reserved. Should be ignored if encountered.
50	endloopAddrsCoarseOffset	The offset, in 32768 sample data point increments beyond the Endloop sample header parameter to the sample data point considered equivalent to the Startloop sample data point for the loop for this instrument. This parameter is added to the endloopAddrsOffset parameter. For example, if Endloop were 5, endloopAddrsOffset were 3 and endAddrsCoarseOffset were 2, sample data point 65544 would be considered equivalent to the Startloop sample data point, and hence sample data point 65543 would effectively precede Startloop during looping.
51	coarseTune	This is a pitch offset, in semitones, which should be applied to the note. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Coarse Tune value of -4 would cause the sound to be reproduced four semitones flat.
52	fineTune	This is a pitch offset, in cents, which should be applied to the note. It is additive with coarseTune. A positive value indicates the sound is reproduced at a higher pitch; a negative value indicates a lower pitch. For example, a Fine Tuning value of -5 would cause the sound to be reproduced five cents flat.

53	sampleID	This is the index into the SHDR subchunk providing the sample to be used for the current instrument zone. A value of zero indicates the first sample in the list. The value should never exceed two less than the size of the sample list. The sampleID enumerator is the terminal generator for IGEN zones. As such, it should only appear in the IGEN subchunk, and it must appear as the last generator enumerator in all but the global zone.
54	sampleModes	This enumerator indicates a value which gives a variety of Boolean flags describing the sample for the current instrument zone. The sampleModes should only appear in the IGEN subchunk, and should not appear in the global zone. The two LS bits of the value indicate the type of loop in the sample: 0 indicates a sound reproduced with no loop, 1 indicates a sound which loops continuously, 2 is unused but should be interpreted as indicating no loop, and 3 indicates a sound which loops for the duration of key depression then proceeds to play the remainder of the sample.
55	reserved3	Unused, reserved. Should be ignored if encountered.
56	scaleTuning	This parameter represents the degree to which MIDI key number influences pitch. A value of zero indicates that MIDI key number has no effect on pitch; a value of 100 represents the usual tempered semitone scale.
57	exclusiveClass	This parameter provides the capability for a key depression in a given instrument to terminate the playback of other instruments. This is particularly useful for percussive instruments such as a hi-hat cymbal. An exclusive class value of zero indicates no exclusive class; no special action is taken. Any other value indicates that when this note is initiated, any other sounding note with the same exclusive class value should be rapidly terminated. The exclusive class generator can only appear at the instrument level. The scope of the exclusive class is the entire preset. In other words, any other instrument zone within the same preset holding a corresponding exclusive class will be terminated.
58	overridingRootKey	This parameter represents the MIDI key number at which the sample is to be played back at its original sample rate. If not present, or if present with a value of -1, then the sample header parameter Original Key is used in its place. If it is present in the range 0-127, then the indicated key number will cause the sample to be played back at its sample header Sample Rate. For example, if the sample were a recording of a piano middle C (Original Key = 60) at a sample rate of 22.050 kHz, and Root Key were set to 69, then playing MIDI key number 69 (A above middle C) would cause a piano note of pitch middle C to be heard.
59	unused5	Unused, reserved. Should be ignored if encountered.
60	endOper	Unused, reserved. Should be ignored if encountered. Unique name provides value to end of defined list.

5.9.3.1.3 Generator Summary

The following tables give the ranges and default values for all SASBF defined generators.

#	Name	Unit	Abs Zero	Min	Min Useful	Max	Max Useful	De-fault	Def Value
0	startAddrsOffset +	smpls	0	0	None	*	*	0	None
1	endAddrsOffset +	smpls	0	*	*	0	None	0	None
2	startloopAddrsOffset +	smpls	0	*	*	*	*	0	None
3	endloopAddrsOffset +	smpls	0	*	*	*	*	0	None
4	startAddrsCoarseOffset +	32k	0	0	None	*	*	0	None
5	modLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
6	vibLfoToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
7	modEnvToPitch	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
8	initialFilterFc	cent	8.176 Hz	1500	20 Hz	13500	20 kHz	13500	Open
9	initialFilterQ	cB	0	0	None	960	96 dB	0	None
10	modLfoToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
11	modEnvToFilterFc	cent fs	0	-12000	-10 oct	1200	10 oct	0	None
12	endAddrsCoarseOffset +	32k	0	*	*	0	None	0	None
13	modLfoToVolume	cB fs	0	-960	-96 dB	960	96 dB	0	None
15	chorusEffectsSend	0.1%	0	0	None	1000	100%	0	None
16	reverbEffectsSend	0.1%	0	0	None	1000	100%	0	None
17	pan	0.1%	Cntr	-500	Left	+500	Right	0	Centre
21	delayModLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
22	freqModLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
23	delayVibLFO	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
24	freqVibLFO	cent	8.176 Hz	-16000	1 mHz	4500	100 Hz	0	8.176 Hz
25	delayModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
26	attackModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
27	holdModEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
28	decayModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
29	sustainModEnv	-0.1%	attk peak	0	100%	1000	0%	0	attk pk
30	releaseModEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
31	keynumToModEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
32	keynumToModEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
33	delayVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
34	attackVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
35	holdVolEnv	timecent	1 sec	-12000	1 msec	5000	20 sec	-12000	<1 msec
36	decayVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
37	sustainVolEnv	cB attn	attk peak	0	0 dB	1440	144dB	0	attk pk
38	releaseVolEnv	timecent	1 sec	-12000	1 msec	8000	100sec	-12000	<1 msec
39	keynumToVolEnvHold	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
40	keynumToVolEnvDecay	tcent/key	0	-1200	-oct/ky	1200	oct/ky	0	None
43	keyRange	MIDI ky#	key# 0	0	lo key	127	hi key	0-127	full kbd
44	velRange	MIDI vel	0	0	min vel	127	max vel	0-127	all vels
45	startloopAddrsCoarseOffset +	smpl	0	*	*	*	*	0	None

46	keynum +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None
47	velocity +	MIDI vel	0	1	min vel	127	max vel	-1	None
48	initialAttenuation	.4 cB	0	0	0 dB	1440	144dB	0	None
50	endloopAddrsCoarseOffset	smpls	0	*	*	*	*	0	None
51	CoarseTune	semitone	0	-120	-10 oct	120	10 oct	0	None
52	fineTune	cent	0	-99	-99cnt	99	99cent	0	None
54	sampleModes +	Bit Flags	Flags	**	**	**	**	0	No Loop
56	scaleTuning	cent/key	0	0	none	1200	oct/ky	100	semi-tone
57	exclusiveClass +	arbitrary #	0	1	--	127	--	0	None
58	overridingRootKey +	MIDI ky#	key# 0	0	lo key	127	hi key	-1	None

* Range depends on values of start, loop, and end points in sample header.

** Range has discrete values based on bit flags

\+ This generator is only valid at the instrument level.

5.9.3.2 Default Modulators

The “default” modulators are described below.

5.9.3.2.1 MIDI Key Velocity to Initial Attenuation

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.2 MIDI Key Velocity to Filter Cutoff

The MIDI key number is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is -2400 Cents.

The product of these values is added to the Initial Filter Cutoff generator summing node.

5.9.3.2.3 MIDI Channel Pressure to Vibrato LFO Pitch Depth

The MIDI Channel Pressure data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 50 cents per max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.4 MIDI Continuous Controller 1 to Vibrato LFO Pitch Depth

The MIDI Continuous Controller 1 data value is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. The MIDI Continuous Controller 33 data value may be optionally used for increased resolution of the controller input.

There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 50 cents/max excursion of vibrato modulation.

The product of these values is added to the Vibrato LFO to Pitch generator summing node.

5.9.3.2.5 MIDI Continuous Controller 7 to Initial Attenuation

The MIDI Continuous Controller 7 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.6 MIDI Continuous Controller 10 to Pan Position

The MIDI Continuous Controller 10 data value is used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 127 is mapped to 63/64 and all other values are mapped between -1 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 1000 tenths of a percent panned-right.

The product of these values is added to the Pan generator summing node.

5.9.3.2.7 MIDI Continuous Controller 11 to Initial Attenuation

The MIDI Continuous Controller 11 data value is used as a Negative Unipolar source, thus the input value of 0 is mapped to a value of 127/128, an input value of 127 is mapped to 0 and all other values are mapped between 127/128 and 0 in a concave fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1. The amount of this modulator is 960 cB (or 96 dB) of attenuation.

The product of these values is added to the initial attenuation generator.

5.9.3.2.8 MIDI Continuous Controller 91 to Reverb Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added reverb send.

The product of these values is added to the Reverb Send generator summing node.

5.9.3.2.9 MIDI Continuous Controller 93 to Chorus Effects Send

The MIDI key number is used as a Positive Unipolar source, thus the input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 128 in a linear fashion. There is no secondary source for this modulator; thus its effect is the same as the effect of multiplying the amount by 1.

The amount of this modulator is 200 tenths of a percent added chorus send.

The product of these values is added to the Chorus Send generator summing node.

5.9.3.2.10 MIDI Pitch Wheel to Initial Pitch Controlled by MIDI Pitch Wheel Sensitivity

The MIDI Pitch Wheel data values are used as a Positive Bipolar source, thus the input value of 0 is mapped to a value of -1, an input value of 16383 is mapped to 8191/8192 and all other values are mapped between -1 and 8191/8192 in a linear fashion.

The MIDI Pitch Wheel Sensitivity data values are used as a secondary source. This source is Positive Unipolar, thus an input value of 0 is mapped to a value of 0, an input value of 127 is mapped to 127/128 and all other values are mapped between 0 and 127/128 in a linear fashion.

The amount of this modulator is 12700 Cents.

The product of these values is added to the Initial Pitch generator summing node.

5.9.3.3 Precedence and Absolute and Relative values.

Most SASBF generators are available at both the Instrument and Preset Levels, as well as having a default value. Generators at the Instrument Level are considered “absolute” and determine an actual physical value for the associated synthesis parameter, which is used instead of the default. For example, a value of 1200 for the attackVolEnv generator would produce an absolute time of 1200 timecents or 2 seconds of attack time for the volume envelope, instead of the default value of -12000 timecents or 1 msec.

Generators at the Preset Level are instead considered “relative” and additive to all the default or instrument level generators within the Preset Zone. For example, a value of 2400 timecents for the attackVolEnv generator in a preset zone containing an instrument with two zones, one with the default attackVolEnv and one with an absolute attackVolEnv generator value of 1200 timecents would cause the default zone to actually have a value of -9600 timecents or 4 msec, and the other to have a value of 3600 timecents or 8 seconds attack time.

There are some generators which are not available at the Preset Level. These are:

#	Name
0	startAddrsOffset
1	endAddrsOffset
2	startloopAddrsOffset
3	endloopAddrsOffset
4	startAddrsCoarseOffset
12	endAddrsCoarseOffset
45	startloopAddrsCoarseOffset
46	keynum
47	velocity
50	endloopAddrsCoarseOffset
54	sampleModes

57	exclusiveClass
58	overridingRootKey

If these generators are encountered in the Preset Level, they should be ignored.

The effect of modulators on a given destination is always relative to the generator value at the Instrument level. However modulators may supersede or add to other modulators depending on their position within the hierarchy. Please see Subclause 8.4 for details on the Modulator implementation and the hierarchical details.

5.9.4 Parameters and Synthesis Model

The SASBF standard has been established with the intent of providing support for an expanding base of wavetable based synthesis models. The model supported by the SASBF specification originates with the EMU8000 wavetable synthesiser chip. The description below of the underlying synthesis model and the associated parameters are provided to allow mapping of this synthesis model onto other hardware platforms.

5.9.4.1 Synthesis Model

The SASBF specification Synthesis Model comprises a wavetable oscillator, a dynamic lowpass filter, an enveloping amplifier, and programmable sends to pan, reverb, and chorus effects units. An underlying modulation engine comprises two low frequency oscillators (LFOs) and two envelope generators with appropriate routing amplifiers.

5.9.4.1.1 Wavetable Oscillator/Interpolator

The SASBF specification wavetable oscillator model is capable of playing back a sample at an arbitrary sampling rate with an arbitrary pitch shift. In practice, the upward pitch shift (downward sample rate conversion) will be limited to a maximum value, typically at least two octaves. The pitch is described in terms of an initial pitch shift which is based on the sample's sampling rate, the root key at which the sample should be unshifted on the keyboard, the coarse, fine, and correction tunings, the effective MIDI key number, and the keyboard scale factor. All modulations in pitch are in octaves, semitones, and cents.

In general, interpolators have a symmetric impulse response, and thus a linear phase characteristic. The interpolation filter's magnitude response can be characterised by three regions - the pass band, the transition band, and the stop band.

The interpolation filter's pass band is the portion of its response which corresponds to the frequencies of the signal stored in waveform memory from DC to that signal's Nyquist frequency

The interpolation filter's transition band is the portion of its response which corresponds to the first image of the signal stored in waveform memory, that is from that signal's Nyquist frequency back to DC.

The interpolation filter's stop band is the portion of its response which corresponds to the remaining images above the transition band to the Nyquist frequency of the upsampled signal.

The guardband will be assumed to begin at 5/6 of the Nyquist frequency, as is the case for a 20 kHz bandwidth in a 48 kHz sample rate system.

Due to poor aliasing rejection in the stopband, linear interpolation does not satisfy this specification.

The specification constrains the Fourier transform of the impulse response of the interpolator. The impulse response is taken with a downward pitch shift of eight octaves. This gives a response which has been upsampled by a factor of 2**8 or 256. In other words, a frequency of the impulse response's Nyquist

frequency divided by 256 gives the filter frequency corresponding to the waveform memory's Nyquist frequency. In the specification below, Fn represents this waveform memory Nyquist frequency.

(All responses measured with downward pitch shift of 8 octaves.)	
Passband Response:	
Ripple:	No more than +/- 0.5 dB
Roll-off:	Minimal, but not to exceed 6 dB at 83.3% Fn.
Transition Band Response:	
Roll-off:	Monotonic and as rapid as possible to at least -80 dB attenuation
Return Lobes:	None above -80 dB
Attenuation:	Greater than 80 dB for all frequencies DC to at least 2% Fn in the first lobe.
Stop Band Response:	
Attenuation:	Greater than 90 dB for all frequencies DC to at least 1% Fn
	Greater than 80 dB for all frequencies DC to at least 20%Fn
	Greater than 60 dB for all frequencies

5.9.4.1.2 Sample Looping

The wavetable oscillator is playing a digital sample which is described in terms of a start point, end point, and two points describing a loop. The sound can be flagged as unlooped, in which case the loop points are ignored. If the sound is looped, it can be played in two ways. If it is flagged as "loop during release", the sound is played from the start point through the loop, and loops until the note becomes inaudible. If not, the sound is played from the start point through the loop, and loops until the key is released. At this point, the next time the loop end point is reached, the sound continues through the loop end point and plays until the end point is reached, at which time audio is terminated.

5.9.4.1.3 Lowpass Filter

The synthesis model contains a resonant lowpass filter, which is characterised by a dynamic cutoff frequency and a fixed resonance (Q).

The filter is idealised at zero resonance as having a flat passband to the cutoff frequency, then a rolloff at 12 dB per octave above that frequency. The resonance, when non-zero, comprises a peak at the cutoff frequency, superimposed on the above response. The resonance is measured as a dB ratio of the resonant peak to the DC gain. The DC gain at any resonance is half of the resonance value below the DC gain at zero resonance; hence the peak height is half the resonance value above DC gain at zero resonance.

All modulations in cutoff frequency are in cents.

Topology: 2nd order AR lowpass filter or equivalent. Filter coefficients are updated in a smooth manner at the sample rate.

Noise and Distortion: Less than 0.003% of full scale on any sinusoidal input for any parameter setting.

Control Parameters: Cutoff Frequency and Resonance.

Resonance Parameter: Specifies the ratio in decibels of the gain of a resonant peak to the gain at DC, when the filter cutoff frequency is set to 1.5 kHz and the modulation is zero. The DC gain of a filter is reduced from the DC gain of a filter with zero resonance by half the resonance value. The resonance parameter will remain fixed throughout the sounding of a musical note.

For a specified resonance, the actual gain ratio should be accurate within +/- 1 dB, and the DC gain accurate within +/-0.5 dB. The ratio of the gain at the resonant peak to the DC gain for all cutoff frequency values shall not deviate by more than 2dB per octave deviation from 1.5 kHz. Nominal gain at DC for a minimum resonance parameter is unity gain.

Cutoff Frequency Parameter: Specifies the frequency at which the filter achieves exactly 3dB of attenuation. The Cutoff Frequency is computed by the combination of the Initial Cutoff Frequency, specified in Hz, and the modulation, specified in semitones and fractions thereof. The Initial Cutoff Frequency is fixed throughout the duration of the a musical note; the modulation can vary at the sample rate. Within the region from 200 Hz to 6.4 kHz, the cutoff frequency parameter accuracy will be +/- 2 semitones.

Frequency Magnitude Response: When the cutoff frequency is at its maximum value and the resonance is at zero, the filter must pass audio without alteration. The filter must support cutoff frequencies down to 200 Hz. There must be a minimum of 2048 distinct cutoff frequencies per octave within the region from 200 Hz to 6.4 kHz, with a worst case spacing between distinct frequencies of 0.01 semitones.

5.9.4.1.4 Final Gain Amplifier

The final gain amplifier is a multiplier on the filter output, which is controlled by an initial gain in dB. This is added to the volume envelope. Additional modulation can also be added. The gain is always specified in dB.

5.9.4.1.5 Effects Sends

The output of the final gain amplifier can be routed into the effects unit. This unit causes the sound to be located (panned) in the stereo field, and a degree of reverberation and chorus to be added. The pan is specified in terms of percentage left and right, which also could be considered as an azimuth angle. The reverb and chorus sends are specified as a percentage of the signal amplitude to be sent to these units, from 0% to 100%.

5.9.4.1.6 Low Frequency Oscillators

The synthesis model provides for two low frequency oscillators (LFOs) for modulating pitch, filter cutoff, and amplitude. The “vibrato” LFO is only capable of modulating pitch. The “modulation” LFO can modulate any of the three parameters.

An LFO is defined as having a delay period during which its value remains zero, followed by a triangular waveshape ramping linearly to positive one, then downward to negative 1, then upward again to positive one, etc.

Each parameter can be modulated to a varying degree, either positively or negatively, by the associated LFO. Modulations of pitch and cutoff are in octaves, semitones, and cents, while modulations of amplitude are in dB. The degree of modulation is specified in cents or dB for the full scale positive LFO excursion.

5.9.4.1.7 Envelope Generators

The synthesis model provides for two envelope generators. The volume envelope generator controls the final gain amplifier and hence determines the volume contour of the sound. The modulation envelope can control pitch and/or filter cutoff.

An envelope generates a control signal in six phases. When key-on occurs, a delay period begins during which the envelope value is zero. The envelope then rises in a convex curve to a value of one during the attack phase. When a value of one is reached, the envelope enters a hold phase during which it remains at one. When the hold phase ends, the envelope enters a decay phase during which its value decreases linearly to a sustain level. When the sustain level is reached, the envelope enters sustain phase, during which the envelope stays at the sustain level. Whenever a key-off occurs, the envelope immediately enters a release phase during which the value linearly ramps from the current value to zero. When zero is reached, the envelope value remains at zero.

Modulation of pitch and filter cutoff are in octaves, semitones, and cents. These parameters can be modulated to varying degree, either positively or negatively, by the modulation envelope. The degree of modulation is specified in cents for the full scale attack peak.

The volume envelope operates in dB, with the attack peak providing a full scale output, appropriately scaled by the initial volume. The zero value, however, is actually zero gain. When 96 dB of attenuation is reached in the final gain amplifier, an abrupt jump to zero gain (infinite dB of attenuation) occurs. In a 16-bit system, this jump is inaudible.

5.9.4.1.8 Modulation Interconnection Summary

The following diagram shows the interconnections expressed in the SASBF specification synthesis model:

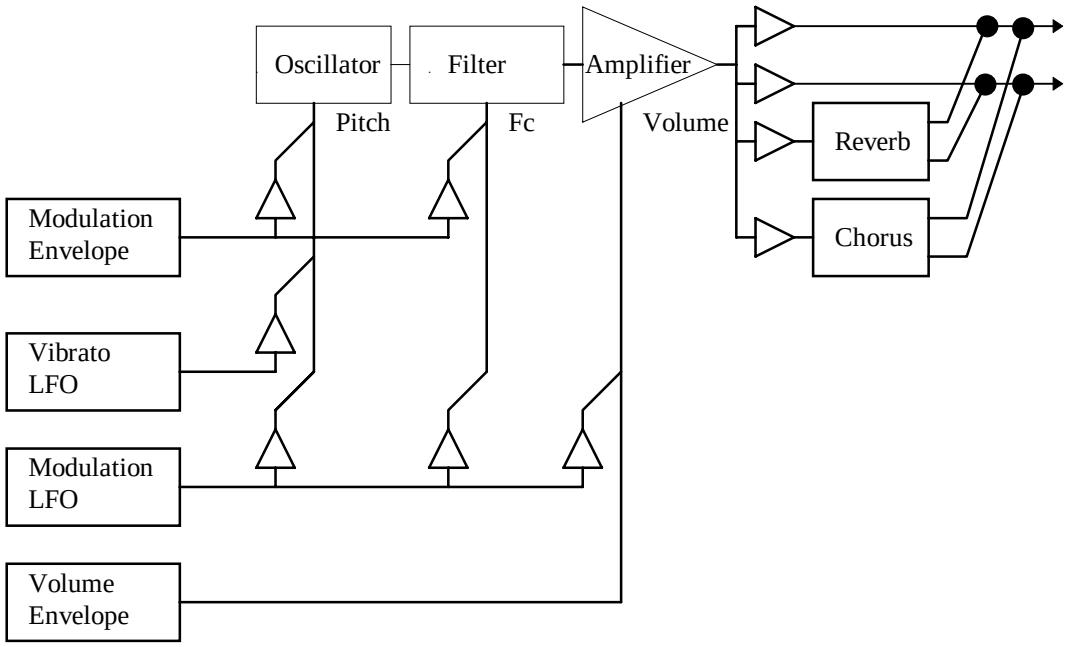


Figure 1: Generator Based Modulation Structure

5.9.4.2 MIDI Functions

The response to certain MIDI commands is defined within the MIDI specification, and therefore not considered to be part of the SASBF specification. These MIDI commands may not be used as sources for the Modulator implementation.

For completeness, the expected responses are given here.

MIDI CC0 Bank Select - When received, the following program change should select the MIDI program in this bank value instead of the default bank of 0.

MIDI CC6 - Data Entry MSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism depending on the Data Entry mode.

MIDI CC32 Bank Select LSB - When received, may behave in conjunction with CC0 Bank Select to provide a total of 16384 possible MIDI banks of programs.

MIDI CC38 Data Entry LSB - When received, its value should be sent to either the RPN or NRPN implementation mechanism, depending on the Data Entry mode.

MIDI CC64 Sustain - ACTIVE when greater than or equal to 64. When the sustain function is active, all notes in the key-on state remain in the key-on state regardless of whether a key-off command for the note arrives. The key-off commands are stored, and when sustain becomes inactive, all stored key-off commands are executed.

MIDI CC66 Soft - ACTIVE when greater than or equal to 64. When active, all new key-ons are modulated in such a way to make the note sound “soft.” This typically affects initial attenuation and filter cutoff in a pre-defined manner.

MIDI CC67 Sostenuto - ACTIVE when greater than or equal to 64. When sostenuto becomes active, all notes currently in the key-on state remain in the key-on state until the sostenuto becomes inactive. All other notes behave normally. Notes maintained by sostenuto in key-on state remain in key-on state even if sustain is switched on and off.

MIDI CC98 NRPN LSB - When received, should be processed by the NRPN implementation mechanism.

MIDI CC99 NRPN MSB - When received, should put the synthesiser in NRPN Data Entry mode and then should be processed by the NRPN implementation mechanism.

MIDI CC100 RPN LSB - When received, should be processed by the RPN implementation mechanism.

MIDI CC101 RPN MSB - When received, should put the synthesiser in RPN Data Entry mode and then should be processed by the RPN implementation mechanism.

MIDI CC120 All Sound Off - When received with any data value, all notes playing in the key-on state bypass the release phase and are shut off, regardless of the sustain or sostenuto positions.

MIDI CC121 Reset All Controllers - Defined as Reset All Controllers as defined by the MIDI specification.

MIDI CC123 All Notes Off - When received with any data value, all notes playing in the key-on state immediately enter release phase, pending their status in SUSTAIN or SOSTENUTO state.

5.9.4.3 Parameter Units

The units with which SASBF generators are described are all well defined. The strict definitions appear below:

ABSOLUTE SAMPLE DATA POINTS - A numeric index of 16 bit sample data point words as stored in ROM or supplied in the smpl-ck, indexing the first sample data point word of memory or the chunk as zero.

RELATIVE SAMPLE DATA POINTS - A count of 16 bit sample data point words based on an absolute sample data point reference. A negative value implies a relative count toward the beginning of the data.

ABSOLUTE SEMITONES - An absolute logarithmic measure of frequency based on a reference of MIDI key numbers. A semitone is 1/12 of an octave, and value 69 is 440 Hz (A-440). Negative values and values above 127 are allowed.

RELATIVE SEMITONES - A relative logarithmic measure of frequency ratio based on units of 1/12 of an octave, which is the twelfth root of two, approximately 1.059463094.

ABSOLUTE CENTS - An absolute logarithmic measure of frequency based on a reference of MIDI key number scaled by 100. A cent is 1/1200 of an octave, and value 6900 is 440 Hz (A-440). Negative values and values above 12700 are allowed.

RELATIVE CENTS - A relative logarithmic measure of frequency ratio based on units of 1/1200 of an octave, which is the twelve hundredth root of two, approximately 1.000577790.

ABSOLUTE CENTIBELS - An absolute measure of the attenuation of a signal, based on a reference of zero being no attenuation. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

RELATIVE CENTIBELS - A relative measure of the attenuation of a signal. A centibel is a tenth of a decibel, or a ratio in signal amplitude of the two hundredth root of 10, approximately 1.011579454.

ABSOLUTE TIMECENTS - An absolute measure of time, based on a reference of zero being one second. A timecent represents a ratio in time of the twelve hundredth root of two, approximately 1.011579454.

RELATIVE TIMECENTS - A relative measure of time ratio, based on a unit size of the twelve hundredth root of two, approximately 1.011579454.

ABSOLUTE PERCENT - An absolute measure of gain, based on a reference of unity. In SASBF, absolute percent is measured in 0.1% units, so a value of zero is 0% and a value of 1000 is 100%.

RELATIVE PERCENT - A relative measure of gain difference. In SASBF, relative percent is measured in 0.1% units. When the gain goes below zero, zero is assumed; when the gain exceeds 100%, 100% is used.

5.9.4.4 The SASBF Generator Model

Five kinds of Generator Enumerators exist: Index Generators, Range Generators, Substitution Generators, Sample Generators, and Value Generators.

In case it is not clear in the general description of the SASBF hierarchy, the following is the precedence of SASBF generator in the hierarchy.

- A ‘generator’ sets or offsets the value of a destination or a synthesis parameter. In exception cases, it sets ranges (Range Generators), or sets values and never offsets values (Index Generators, Sample Generators, and Substitution Generators).
- A generator is defined as identical to another generator if its generator operator is the same in both generators.
- A generator in a global instrument zone which is identical to a default generator supersedes or replaces the default generator.
- A generator in a local instrument zone which is identical to a default generator or to a generator in a global instrument zone supersedes or replaces that generator.
- Points below (until noted) apply to Value Generators ONLY.
- A generator at the preset level adds to a generator at the instrument level if both generators are identical.
- A generator in a global preset zone which is identical to a default generator or to a generator in an instrument adds to that generator.
- A generator in a global preset zone which is not identical to a default generator and is not identical to a generator in an instrument has its effect added to the given synthesis parameter.
- A generator in a local preset zone which is identical to a generator in a global preset zone supersedes or replaces that generator in the global preset zone. That generator then has its effects added to the destination summing node of all zones in the given instrument.
- A generator in a local preset zone which is not identical to a default generator or a generator in a global preset zone has its effects added to the destination summing node of all zones in the given instrument.
  If the generator operator is a Range Generator, the generator values are NOT ADDED to those

in the instrument level, rather they serve as an intersection filter to those key number or velocity ranges in the instrument which is used in the preset zone.

- If the generator operator is a Substitution Generator or a Sample Generator, they are illegal at the preset level. The only Index Generator legal at the Preset Level is ‘instrumentID’, whereas the only Index Generator legal at the Instrument Level is ‘sampleID’

5.9.4.5 The SASBF Modulator Controller Model

SASBF Modulators are used to allow real-time control over the sound in sound designer programmable manner. Each instance of a SASBF modulator structure defines a real-time perceptually additive effect to be applied to a given destination or synthesiser parameter.

Modulators provide future extensibility to the SASBF standard. They are not used in this version.

5.9.5 Error Handling

5.9.5.1 Structural Errors

Structural Errors are errors which are determined from the implicit redundancy of the SASBF RIFF bitstream element structure, and indicate that the structure is not intact. Examples are incorrect lengths for the chunks or subchunks, pointers out of valid range, or missing required chunks or subchunks for which no error correction procedure exists.

In all cases, bitstream elements should be checked for structural errors at load time, and if any are found, the bitstream elements should be rejected. Separate tools or options can be used to “repair” structurally defective bitstream elements, but these tools should validate that the reconstructed bitstream element is not only a valid SASBF bank but also complies with the intended timbral results in all cases.

5.9.5.2 Unknown Chunks

In parsing the RIFF structure, unknown but well formed chunks or subchunks may be encountered. Unknown chunks within the INFO-list chunk should simply be ignored. Other unknown chunks or subchunks are illegal and should be treated as structural errors.

5.9.5.3 Unknown Enumerators

Unknown enumerators may be encountered in Generators, Modulator Sources, or Transforms. This is to be expected if the ifil field exceeds the specification to which the application was written. Even if unexpected, unknown enumerators should simply cause the associated Generator or Modulator to be ignored.

5.9.5.4 Illegal Parameter Values

Some SASBF parameters are defined for only a limited range of the possible values which can be expressed in their field. If the value of the field is not in the defined range, the parameter has an illegal value.

Illegal values for may be detected either at load or at run time. If detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the default value for the parameter should be used if the parameter is required, or the entire Generator or Modulator ignored if it is optional. Certain parameters may have more specific procedures for illegal values as expressed elsewhere in this specification.

5.9.5.5 Out-of-range Values

SASBF parameters have a specified minimum and useful range the span the perceptually relevant values for the associated sonic property. When the parameter value is exceeds this useful range, the parameter is said to have an out of range value.

Out of range values can result from two distinct causes. An out of range value can be actually present as a SASBF generator value, or the out of range value can be the result of the summation of instrument and preset values.

Out of range values should be handled by substituting the nearest perceptually relevant or realisable value. SASBF banks should not be created with out of range values in the instrument generators. While it is acceptable practice to create SASBF banks which produce out of range values as a result of summation, it is undesirable and should be avoided where practical.

5.9.5.6 Missing Required Parameter or Terminator

Certain parameters and terminators are required by the SASBF specification. If these are missing, the bitstream element is technically not within specification. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the instrument or zone for which the required parameter is missing should simply be ignored. If this causes no sound, the corresponding key-on event is ignored.

5.9.5.7 Illegal enumerator

Certain enumerators are illegal in certain contexts. For example, key and velocity ranges must be the first generators in a zone, instruments are not allowed in instrument zones, and sampleIDs are not allowed in preset zones. If such a problem is detected at load time, the bitstream element may optionally be rejected as structurally unsound. If detected at run time, the enumerator should simply be ignored.

5.9.6 Profile 2 (Sample Bank and MIDI decoding)

This Subclause describes the decoding process in which a bitstream conforming to Profile 2 is converted into sound. Profile 2 supports the Structured Audio Sample Bank Format and the General MIDI Format (Profile 1) for sound description. Profile 2 supports the MIDI Protocol and the Standard MIDI File Format for score specification.

5.9.6.1 Stream information header

The SASBF bitstream element is part of the bitstream header. Score elements in the form of MIDI files may be included in the bitstream header.

At the creation of a Structured Audio Elementary Stream, a Structured Audio decoder is instantiated and a bitstream object of class `SA_streaminfo` provided to that decoder as configuration information. At this time, the decoder shall initialise a run-time scheduler, and then parse the stream information object into its component parts and use them as follows:

- MIDI file: The events in the MIDI file shall be time ordered, and those events registered with the scheduler.

- Sample bank: The data in the bank shall be stored, and whatever preprocessing necessary to prepare for using the bank for synthesis shall be performed. The sample bank requires no processing for this preparation. Processing may be used to improve the computational

efficiency of a decoder. Banks shall be assigned consecutive increasing bank ID numbers in the order of the banks' position in the bitstream. The first bank in the bitstream shall be numbered 0 unless a bank resident in the terminal is being used. In that case, the resident bank shall be numbered 0, an other banks shall be numbered proceeding from there.

5.9.6.2 Bitstream data and sound creation

The MIDI Note On message is defined in the MIDI specification. When the SASBF decoder receives a Note On message, a voice shall be instantiated. Synthesis parameters for the voice shall be determined by the data in the sample bank containing the preset corresponding to the MIDI channel of the Note On message. The number of wavetable oscillators that are used by the voice is defined by the sample bank. Each required oscillator shall have the state variables required for synthesis allocated to it. The state variables shall be initialised according to the preset data from the sample bank.

The MIDI Program Change message is defined in the MIDI specification. When the SASBF decoder receives a Program Change message, an assignment shall be made in the scheduler between the specified MIDI channel and the specified preset. Until this assignment is changed by a subsequent Program Change message, subsequent voice instantiations using the specified MIDI channel shall use sample bank data corresponding to the assigned preset. In a MIDI program change message, if no preset exists in the specified bank with the specified preset number, a replacement preset is used. The replacement preset is the preset with the specified preset number in the bank with the highest bank ID number less than the specified bank ID number which contains a preset with the specified preset number.

The run time scheduler is used to issue MIDI messages to the SASBF decoder. MIDI messages from a MIDI standard file shall be issued by the scheduler when the scheduler clock equals or exceeds the time stamp associated with the message.

5.9.6.3 Conformance

Floating point computation is not required in Profile 2. Audio samples are stored in the bitstream as 16-bit integers. The resolution of the interpolator filter is constrained by the interpolator specification given in Subclause 0.

5.9.7 Profile 4 (Sample Bank decoding in SAOL instruments)

5.9.8 Sample Bank Format Glossary

absolute - Describes a parameter which gives a definitive real-world value. Contrast to relative.

additive - Describes a parameter which is to be numerically added to another parameter.

articulation - The process of modulation of amplitude, pitch, and timbre to produce an expressive musical note.

attack - That phase of an envelope or sound during which the amplitude increases from zero to a peak value.

attenuation - A decrease in volume or amplitude of a signal.

bag - A Sample Bank Format data structure element containing a list of zones.

balance - A form of stereo volume control in which both left and right channels are at maximum when the control is centred, and which attenuates only the opposite channel when taken to either extreme.

bank - A collection of presets. See also MIDI bank.

bipolar - In the SASBF standard, said of a modulator source whose minimum is -1 and whose maximum is 1. Contrast “unipolar”

big endian - Refers to the organisation in memory of bytes within a word such that the most significant byte occurs at the lowest address. Contrast “little endian.”

byte - A data structure element of eight bits without definition of meaning to those bits.

BYTE - A data structure element of eight bits which contains an unsigned value from 0 to 255.

case-insensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as identical. Contrast “case-sensitive.”

case-sensitive - Indicates that an ASCII character or string treats alphabetic characters of upper or lower case as distinct. Contrast “case-insensitive.”

cent - A unit of pitch ratio corresponding to the twelve hundredth root of two, or one hundredth of a semitone, approximately 1.000577790.

centibel - A unit of amplitude ratio corresponding to the two hundredth root of ten, or one tenth of a decibel, approximately 1.011579454.

CHAR - A data structure of eight bits which contains a signed value from -128 to +127.

chorus - An effects processing algorithm which involves cyclically shifting the pitch of a signal and remixing it with itself to produce a time varying comb filter, giving a perception of motion and fullness to the resulting sound.

chunk - The top-level division of a RIFF file.

convex - A curve which is bowed in such a way that it is steeper on its lower portion.

concave - (1) A curve which is bowed in such a way that it is steeper on its upper portion. (2) In the SASBF standard, said of a modulator source whose shape is that of the amplitude squared characteristic. Contrast with “convex” and “linear.”

cutoff frequency - The frequency of a filter function at which the attenuation reaches a specified value.

data points - The individual values comprising a sample. Sometimes also called sample points. Contrast “sample.”

decay - The portion of an envelope or sound during which the amplitude declines from a peak to steady state value.

decibel - A unit of amplitude ratio corresponding to the twentieth root of ten, approximately 1.122018454.

delay - The portion of an envelope or LFO function which elapses from a key-on event until the amplitude becomes non-zero.

destination - The generator to which a modulator is applied.

DC gain - The degree of amplification or attenuation a system presents to a static or zero frequency signal.

digital audio - Audio represented as a sequence of quantised values spaced evenly over time. The values are called “sample data points.”

doubleword - A data structure element of 32 bits without definition of meaning to those bits.

downloadable - Said of samples which are loaded from a file into RAM, in contrast to samples which are maintained in ROM.

dry - Refers to audio which has not received any effects processing such as reverb or chorus.

DWORD - A data structure of 32 bits which contains an unsigned value from zero to 4,294,967,295.

envelope - A time varying signal which typically controls the pitch, volume, and/or filter cutoff frequency of a note, and comprises multiple phases including attack, decay, sustain, and release.

enumerated - Said of a data element whose symbols correspond to particular assigned functions.

flat - A. Said of a tone that is lower in pitch than another reference tone. B. Said of a frequency response that does not deviate significantly from a single fixed gain over the audio range.

generator - In the SASBF standard, a parameter which directly affects sound reproduction. Contrast with “modulator.”

global - Refers to parameters which affect all associated structures. See “global zone.”

global zone - A zone whose generators and modulators affect all other zones within the object.

header - A data structure element which describes several aspects of a SASBF element.

instrument - In the SASBF standard, a collection of zones which represents the sound of a single musical instrument or sound effect set.

instrument zone - A sample and associated articulation data defined to play over certain key numbers and velocities.

interpolator - A circuit or algorithm which computes intermediate points between existing sample data points. This is of particular use in the pitch shifting operation of a wavetable synthesiser, in which these intermediate points represent the output samples of the waveform at the desired pitch transposition.

key number - See MIDI key number.

LFO - Acronym for Low Frequency Oscillator. A slow periodic modulation source.

linear - In the SASBF standard, said of a modulator source whose shape is that of a straight line. Contrast with “concave.”

linear coding - The most common method of encoding amplitudes in digital audio in which each step is of equal size.

little endian - A method of ordering bytes within larger words in memory in which the least significant byte is at the lowest address. Contrast “big endian.”

loop - In wavetable synthesis, a portion of a sample which is repeated many times to increase the duration of the resulting sound.

loop points - The sample data points at which a loop begins and ends.

lowpass - Said of a filter which attenuates high frequencies but does not attenuate low frequencies.

modulator - In the SASBF standard, a parameter which routes an external controller to dynamically alter the setting of a “generator.” Contrast with “generator.”

monotonic - Continuously increasing or decreasing. Said of a sequence which never reverses direction.

MIDI - Acronym for Musical Instrument Digital Interface. The standard protocol for sending performance information to a musical synthesiser.

MIDI bank - A group of up to 128 presets selected by a MIDI “change bank” command.

MIDI continuous controller - A construct in the MIDI protocol.

MIDI key number - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the key of the musical instrument keyboard to which the command refers.

MIDI pitch bend - A special MIDI construct akin to the MIDI continuous controllers which controls the real-time value of the pitch of all notes played in a MIDI channel.

MIDI preset - A “preset” selected to be active in a particular MIDI channel by a MIDI “change preset” command.

MIDI velocity - A construct in the MIDI protocol which accompanies a MIDI key-on or key-off command and specifies the speed with which the key was pressed or released.

modulator - In the SASBF standard, a set of parameters which affect a particular generator. Contrast with “generator.”

mono - Short for “monophonic.” Indicates a sound comprising only one channel or waveform. Contrast with “stereo.”

negative - In the SASBF standard, said of a modulator which has a negative sloping characteristic. Contrast with “positive.”

octave - A factor of two in ratio, typically applied to pitch or frequency.

orphan - Said of a data structure which under normal circumstances is referenced by a higher level, but in this particular instance is no longer linked. Specifically, it is an instrument which is not referenced by any preset zone, or a sample which is not referenced by any instrument zone.

oscillator - In wavetable synthesis, the wavetable interpolator is considered an oscillator.

pan - Short for “panorama.” This is the control of the apparent azimuth of a sound source over 180 degrees from left to right. It is generally implemented by varying the volume at the left and right speakers.

pitch - The perceived value of frequency. Generally can be used interchangeably with frequency.

pitch shift - A change in pitch. Wavetable synthesis relies on interpolators to cause pitch shift in a sample to produce the notes of the scale.

pole - A mathematical term used in filter transform analysis. Traditionally in synthesis, a pole is equated with a rolloff of 6dB per octave, and the rolloff of a filter is specified in “poles.”

positive - In the SASBF standard, said of a modulator source which has a positive sloping characteristic. Contrast “negative.”

preset - A keyboard full of sound. Typically the collection of samples and articulation data associated with a particular MIDI preset number.

preset zone - A subset of a preset containing generators, modulators, and an instrument.

proximal - Closest to. Proximal sample data points are the data points closest in either direction to the named point.

Q - A mathematical term used in filter transform analysis. Indicates the degree of resonance of the filter. In synthesis terminology, it is synonymous with resonance.

RAM - Random Access Memory. Conventionally, this term implies read-write memory. Contrast “ROM.”

record - A single instance of a data structure.

relative - Describes a parameter which merely indicates an offset from an otherwise established value. Contrast to absolute.

release - The portion of an envelope or sound during which the amplitude declines from a steady state to zero value or inaudibility.

resonance - Describes the aspect of a filter in which particular frequencies are given significantly more gain than others. The resonance can be measured in dB above the DC gain.

resonant frequency - The frequency at which resonance reaches its maximum.

reverb - Short for reverberation. In synthesis, a synthetic signal processor which adds artificial spaciousness and ambience to a sound.

RIFF - Acronym for Resource Interchange File Format.

ROM - Acronym for Read Only Memory. A memory whose contents are fixed at manufacture, and hence cannot be written by the user. Contrast with RAM.

sample - This term is often used both to indicate a “sample data point” and to indicate a collection of such points comprising a digital audio waveform. The latter meaning is exclusively used in this Subclause (the SASBF specification.)

sample rate - The frequency, in Hertz, at which sample data points are taken when recording a sample.

semitone - A unit of pitch ratio corresponding to the twelfth root of two, or one twelfth of an octave, approximately 1.059463094.

sharp - Said of a tone that is higher in pitch than another reference tone.

SHORT - A data structure element of sixteen bits which contains a signed value from -32,768 to +32,767.

soft - The pedal on a piano, so named because it causes the damper to be lowered in such a way as to soften the timbre and loudness of the notes. In MIDI, continuous controller #66, which behaves in a similar manner.

sostenuto - The pedal on a piano which causes the dampers on all keys depressed to be held until the pedal is released. In MIDI, continuous controller #67, which behaves in a similar manner.

sustain - The pedal on a piano which prevents all dampers on keys as they are depressed from being released. In MIDI, continuous controller #64, which behaves in a similar manner.

source - In a SASBF modulator, the enumerator indicating the particular real-time value which the modulator will transform, scale, and add to the destination generator.

stereo - Literally indicating three dimensions. In this specification, the term is used to mean two channel stereophonic, indicating that the sound is composed of two independent audio channels, dubbed left and right. Contrast monophonic.

subchunk - A division of a RIFF file below that of the chunk.

synthesis engine - The hardware and software associated with the signal processing and modulation path for a particular synthesiser.

synthesiser - A device ideally capable of producing arbitrary musical sound.

terminator - A data structure element indicating the final element in a sequence.

timecent - A unit of duration ratio corresponding to the twelve hundredth root of two, or one twelve hundredth of an octave, approximately 1.000577790.

transform - In a SASBF modulator, the enumerator indicating the particular transfer function through which the source will be passed prior to scaling and addition to the destination generator.

tremolo - A periodic change in amplitude of a sound, typically produced by applying a low frequency oscillator to the final volume amplifier.

triangular - A waveform which ramps upward to a positive limit, then downward at the opposite slope to the symmetrically negative limit periodically.

unipolar - In the SASBF standard, said of a modulator source whose minimum is 0 and whose maximum is 1. Contrast “bipolar.”

velocity - In synthesis, the speed with which a keyboard key is depressed, typically proportionally to the impact delivered by the musician. See also MIDI velocity.

vibrato - A periodic change in the pitch of a sound, typically produced by applying a low frequency oscillator to the oscillator pitch.

volume - The loudness or amplitude of a sound, or the control of this parameter.

wavetable - A music synthesis technique wherein musical sounds are recorded or computed mathematically and stored in a memory, then played back at a variable rate to produce the desired pitch. Additional timbral adjustments are often made to the sound thus produced using amplifiers, filters, and effects processing such as reverb and chorus.

WORD - A data structure of 16 bits which contains an unsigned value from zero to 65,535.

word - A data structure element of 16 bits without definition of meaning to those bits.

zone - An object and associated articulation data defined to play over certain key numbers and velocities.

5.10 MIDI semantics

5.10.1 Introduction

This Subclause describes the normative decoding process for Profile 1 implementations, and the normative mapping from MIDI events in the stream information header and bitstream data into SAOL semantics for Profile 4 implementations.

The MIDI standards referenced are standardised externally by the MIDI Manufacturers Association. In particular, we reference the Standard MIDIFile format, the MIDI protocol, and the General MIDI patch mapping, all standardised in **[MIDI]**. The MIDI terminology used in this Subclause is defined in that document.

5.10.2 Profile 1 decoding process

Little normative needs be said about the Profile 1 decoding process. The rules given in **[MIDI]** apply as standardised in those documents. As described in Subclause 5.2 Profiles, only **midi** and **midi_file** bitstream elements shall occur in a Profile 1 bitstream.

There are no normative aspects to producing sound in Profile 1.

5.10.3 Mapping MIDI events into orchestra control

5.10.3.1 Introduction

For Profile 4, events coded as MIDI data must be converted, when they are received in the terminal as part of a Standard MIDIFile or MIDI event, into the appropriate scheduler semantics. This Subclause lists the various MIDI events and their corresponding semantics in MPEG-4.

5.10.3.2 MIDI events

5.10.3.2.1 Introduction

This Subclause describes the semantics of the various types of events that may arrive in a continuous bitstream as a **MIDI_event** object. The syntax of these objects is standardised externally in **[MIDI]**.

5.10.3.2.2 NoteOn

```
noteon channel note velocity
```

When a **noteon** event is received, each instrument in the orchestra currently assigned to channel **channel** shall be instantiated with duration –1 and the first two p-fields set to **note** and **velocity**. Each value of **MIDIctrl[]** within the instrument instance is set to the most recent value of a controller change for that controller on channel **channel** or to the default value (see Subclause 5.10.3.4 Default controller values) if there have been no controller changes for that controller on that channel. The value of **MIDIbend** is set to the most recent value of the MIDI pitch bend. The value of **MIDItouch** is set to the most recent aftertouch value on the channel.

An instrument instance created in response to a **noteon** message on a particular channel is referred to as being “on” that channel.

5.10.3.2.3 NoteOff

```
noteoff channel note velocity
```

When a **noteoff** event is received, each instrument instance on channel **channel** which was instantiated with note number **note** is scheduled for termination at the end of the k-cycle; that is, its **released** flag is set, and if the instrument does not call **extend**, it shall be de-instantiated after the current k-cycle of computation.

5.10.3.2.4 Control change

```
cc channel controller value
```

When a **cc** or control change event is received, the new value of the specified controller is set to **value**. This value shall be cached so that future instrument instances on the given channel have access to it; also, all currently active instrument instances on the channel **channel** shall have the standard name **MIDIctrl[controller]** updated to **value**.

5.10.3.2.5 Aftertouch

```
touch channel note velocity
```

When a **touch** event is received, the value of the **MIDItouch** variable of each instrument instance on channel **channel** which was instantiated with note number **note** is set to **velocity**.

5.10.3.2.6 Channel aftertouch

```
ctouch channel velocity
```

When a **ctouch** event is received, the value of the **MIDItouch** variable of each instrument instance on channel **channel** is set to **velocity**.

5.10.3.2.7 Program change

```
pchange channel program
```

When a **pchange** event is received, the current instrument receiving events on channel **channel** shall be changed to the instrument with preset number **program** (see Subclause 5.4.6.4.1 Preset tag). If there is no instrument with this preset number, then future events on the channel, until another program change is received, shall be ignored.

5.10.3.2.8 Pitch wheel change

```
pwheel channel value
```

When a **pwheel** event is received, the **MIDIbend** value for each instrument instance on channel **channel** shall be set to **value**.

5.10.3.2.9 All notes off

```
notesoff
```

When a **notesoff** event is received, all instrument instances in the orchestra are terminated at the end of the current k-cycle. Instruments may not save themselves from termination by using the **extend** statement in this case.

5.10.3.2.10 MIDI messages not respected

The following MIDI messages have no meaning in MPEG-4 Profiles 3 and 4:

Local Control
Omni Mode On/Off
Mono Mode On/Off
Poly Mode On/Off
System Exclusive
Tune Request
Timing Clock
Song Select/Continue/Stop
Song Position
Active Sensing
Reset

5.10.3.3 Standard MIDI Files

MIDI files have data with the same semantics as the MIDI messages described above; however, the timing semantics are more complicated due to the use of chunks, sequences, and varying tempo.

To process a **smf** stream information element, the following steps must be taken. First, the entire stream element is parsed and cached. Then, using the sequence instructions, the tempo commands, and the sequencer model described in **[MIDI]**, the delta-times of the various events are converted into absolute timestamps. To perform this step requires converting each MIDI track chunk into a timelist, and then using the song select and song position commands to interleave the various tracks as required.

Then, for each event labelled with its proper absolute time, the event is registered with the scheduler and dispatched according to the semantics in the preceding Subclause when the appropriate time arrives.

5.10.3.4 Default controller values

The following table gives the default values for certain continuous controllers. If a particular controller is not listed here, then its default value shall be zero.

There is no normative significance to these “function names”; however, content authors who wish to use General MIDI score files with SAOL orchestras are advised to consult **[MIDI]** for the normative meaning of the controllers and controller values within General MIDI bitstreams and MIDIfiles.

Controller	Function	Default
1	Mod Wheel	0
5	Portamento Speed	0
7	Volume	100
10	Pan	64
11	Expression	127
65	Portamento	0
66	Sus Pedal	0
67	Soft Pedal	0
84	Portamento Control	0
Pitch Bend	Pitch Bend	8192

5.11 Input sounds and relationship with AudioBIFS

5.11.1 Introduction

This Subclause describes the use of SAOL orchestras as the effects-processing functionality in the AudioBIFS (Binary Format for Scene Description) system, described in ISO 14496-1 Subclause XXX. In ISO/IEC 14496, SAOL is used not only as a sound-synthesis description method, but also as a description method for sound-effects and other post-production algorithms. The BIFS **AudioFX** node allows the inclusion of signal-processing algorithms described in SAOL which are applied to the outputs of the sound nodes subsidiary to that node in the scene graph. This functionality fits well into the bus-send methodology in Structured Audio, but requires some additional normative text to exactly describe the process.

5.11.2 Input sources and phaseGroup

Each node in a BIFS scene graph that contains SAOL code is either an **AudioSource** node or an **AudioFX** node. If the former, there are no input sources to the SAOL orchestra, and so the default orchestra global **inchannels** value is 0 (see Subclause 5.4.5.2.3 inchannels parameter). In this case, the special bus **input_bus** may not be sent to an instrument or otherwise used in the orchestra.

If the latter, the child nodes of the **AudioFX** node provide several channels of input sound to the orchestra. These channels of input sound, calculated as described in ISO 14496-1 Subclause XXX, are placed on the special bus **input_bus**. From this bus, they may be sent to any instrument(s) desired and the audio data thereby provided shall be treated normally. The number of orchestra input channels---the default value of orchestra global **inchannels**---is the sum of the numbers of channels of sound provided by each of the children.

In any instrument that receives a **send** from the special bus **input_bus**, the value of the **inGroup** standard name (see Subclause 5.4.6.8.13 inGroup) shall be constructed using the **phaseGroup** values of the child nodes in the scene graph, as follows. The **inGroup**[] values, when non-zero, shall have the property that **inGroup**[**i**] **= inGroup**[**j**] when **i** ≠ **j** exactly when **input** channel **i** is output channel **n** of child **c1**, **input** channel **j** is output channel **m** of child **c2**, **c1 = c2**, and **phaseGroup**[**n**] = **phaseGroup**[**m**] within **c1**. (That is, when the two channels come from the same child and are phase-grouped in that child's output).

This rule applies in addition to the usual **inGroup** rules as given in Subclause 5.4.6.8.13 inGroup.

EXAMPLE

Assume that the two child nodes of an **AudioFX** node produce two and three channels of output respectively, and their **phaseGroup** fields are [1,1] and [1,0,1] respectively. That is, in the first child, the two channels form a stereo pair; and in the second, the first and third channels form a stereo pair which has no phase relationships with the second channel.

For the following global orchestra definitions:

```
send(input_bus ; ; a);
route(a, bus2);
send(bus2,input_bus,bus2 ; ; b);
```

Assume that instrument **a** produces two channels of output. Then, a legal value for the **inGroup** name within **a** is [1,1,2,0,2], and a legal value for the **inGroup** name within **b** is [1,1,2,2,3,0,3,4,4]. The value for the **inGroup** name within **a** shall not be [1,1,1,0,1], and the value for the **inGroup** name within **b** shall not be [1,1,2,2,2,2,2,3,3] (among other illegal possibilities).

5.11.3 The AudioFX node

When a SAOL orchestra is instantiated due to an **AudioFX** BIFS node, only an orchestra file (the **orch** field in the node) and, optionally, a SASL score file (the **score** field) are provided. These files correspond to tokenised sequences of orchestra and score data forming legal **orchestra** and **score_file** bitstream elements as described in Subclause 5.1.2 Bitstream syntax. Further, a score may not contain new instrument events, but only control parameters for the **send** instruments defined in the orchestra.

To instantiate the orchestra for the AudioFX node requires the following steps:

1. Decoding of the **orch** and **score** (if any) elements in the node

2. Parsing and syntax-checking of these elements

3. Instantiation of **send** instruments in the orchestra (as described in Subclause 5.3.2 Decoder configuration header).

Each of these **send** instances shall be maintained until it is turned off by the **turnoff** statement, or the node containing the orchestra is deleted from the scene graph. If the **turnoff** statement is used in one of these instruments, it shall be taken as producing zero values for all future time.

The run-time synthesis process proceeds according to the rules cited in Subclause 5.3.3.3 Scheduler semantics for a standard SA decoding process, with the following exceptions and additions:

As no run-time events will be received by an **AudioFX** process, no communication with the systems layer need be maintained for this purpose. The only events used are those which are in the **score** field of the node itself. At each time step, the **AudioFX** orchestra shall request from the systems layer the input audio buffers which correspond to the child nodes. These audio buffers shall be placed on the special bus **input_bus** and then sent to whatever instruments are specified in the global orchestra header.

Also, at each control-rate step, the **params[]** fields of the **AudioFX** node shall be copied into the global **params[]** array of the orchestra. These fields are exposed in the scene graph so that interactive aspects of other parts of the scene graph may be used to control the orchestra. At the end of each control cycle, the **params[]** array values shall be copied back into the corresponding fields of the **AudioFX** node and then routed to other nodes as specified within the scene graph. (It is not possible to give a more semantically meaningful field name than **params** since the purpose of the field may vary greatly from application to application, depending on the needs of the content).

At every point in time, the output of the orchestra is the output of the **AudioFX** node.

5.11.4 Interactive 3-D spatial audio scenes

When an **AudioSource** or **AudioFX** node is the child of a **Sound** node, the spatial location, direction, and propagation pattern of the sound subtree represented at the position of the **Sound** node, and the spatial location and direction of the listener, are provided to the SAOL code in the node. In this way, subtle spatial effects such as source directivity modelling may be written in SAOL.

The standard names **position**, **direction**, **listenerPosition, listenerDirection, minFront, maxFront, minBack,** and **maxBack** (see Subclauses 5.4.6.8.16 position-5.4.6.8.23 maxBack) are used for this purpose.

It is not recommended that content providing 3-D spatial audio in the context of audio-visual virtual reality applications in BIFS use the **spatialize** statement within SAOL to provide this functionality. In most terminals, the scene-composition 3-D audio functionality will be able to use more information about the

interaction process to provide the best-quality audio rendering. In particular, spatial positioning and source directivity are implemented at the end terminal with a sophistication suitable for the terminal itself (see Systems CD, Sound node specification, Subclause XXX). Content authors can use SAOL and the **AudioFX** node to creat*e* enhanced spatial effects that include reverberation, environmental attributes and complex attenuation functions, and then let the terminal-level spatial audio presentation be used to any available rendering method. The **spatialize** statement in SAOL is provided for the creation of non-interactive spatial audio effects in musical compositions, so that composers may tightly integrate the spatial presentation with other aspects of the musical material.

Annex A Tables (normative)

This Annex contains the bitstream token table as referenced in Subclause 5.1 Bitstream syntax and semantics and Subclause 5.8 SAOL/SASL . Certain tokens are indicated as **(reserved),** which means they are not currently used in the bitstream, but may be used in future versions of the standard. Tokens 0xE2 through 0xEF may be used by implementors for implementation-dependent purposes.

Bitstream token table

Token	Text
0x00	aopcode
0x01	asig
0x02	else
0x03	exports
0x04	extend
0x05	global
0x06	if
0x07	imports
0x08	inchannels
0x09	inputmod
0x0A	instr
0x0B	iopcode
0x0C	ivar
0x0D	kopcode
0x0E	krate
0x0F	ksig
0x10	map
0x11	oparray
0x12	opcode
0x13	outbus
0x14	outchannels
0x15	output
0x16	return
0x17	route
0x18	send
0x19	sequence
0x1A	sbsynth
0x1B	spatialize
0x1C	srate
0x1D	table
0x1E	tablemap
0x1F	template
0x20	turnoff
0x21	while
0x22	with
0x23	xsig
0x24	channel
0x25	preset
0x26-0x2F	**(reserved)**
0x30	k_rate
0x31	s_rate
0x32	inchan
0x33	outchan
0x34	time
0x35	dur
0x36	MIDIctrl
0x37	MIDItouch
0x38	MIDIbend
0x39	input
0x3A	inGroup
0x3B	released
0x3C	cpuload
0x3D	position
0x3E	direction
0x3F	listenerPosition
0x40	minFront
0x41	minBack
0x42	maxFront
0x43	maxBack
0x44	params
0x45-0x4F	**(reserved)**
0x50	&&
0x51	||
0x52	>=
0x53	<=
0x54	!=
0x55	==
0x56	-
0x57	*
0x58	/
0x59	+
0x5A	>
0x5B	<
0x5C	?
0x5D	:
0x5E	(
0x5F	)
0x60	{
0x61	}
0x62	[
0x63	]
0x64	;

0x65	'
0x66	=
0x67	!
0x68-0x6F	**(reserved)**
0x70	sample
0x71	data
0x72	random
0x73	step
0x74	lineseg
0x75	expseg
0x76	cubicseg
0x77	polynomial
0x78	window
0x79	harm
0x7A	harm_phase
0x7B	periodic
0x7C	buzz
0x7D	concat
0x7E	empty
0x7F	**(reserved)**
0x80	int
0x81	frac
0x82	dbamp
0x83	ampdb
0x84	abs
0x85	exp
0x86	log
0x87	sqrt
0x88	sin
0x89	cos
0x8A	atan
0x8B	pow
0x8C	log10
0x8D	asin
0x8E	acos
0x8F	floor
0x90	ceil
0x91	min
0x92	max
0x93	pchoct
0x94	octpch
0x95	cpspch
0x96	pchcps
0x97	cpsoct
0x98	octcps
0x99	pchmidi
0x9A	midipch
0x9B	octmidi
0x9C	midioct
0x9D	cpsmidi
0x9E	midicps
0x9F	**(reserved)**
0xA0	ftlen
0xA1	ftloop
0xA2	ftloopend
0xA3	ftsetloop
0xA4	ftsetend
0xA5	ftbasecps
0xA6	ftsetbase
0xA7	tableread
0xA8	tablewrite
0xA9	oscil
0xAA	loscil
0xAB	doscil
0xAC	koscil
0xAD	kline
0xAE	aline
0xAF	sblock
0xB0	kexpon
0xB1	aexpon
0xB2	kphasor
0xB3	aphasor
0xB4	pluck
0xB5	buzz
0xB6	fof
0xB7	irand
0xB8	krand
0xB9	arand
0xBA	ilinrand
0xBB	klinrand
0xBC	alinrand
0xBD	iexprand
0xBE	kexprand
0xBF	aexprand
0xC0	kpoissonrand
0xC1	apoissonrand
0xC2	igaussrand
0xC3	kgaussrand
0xC4	agaussrand
0xC5	port
0xC6	hipass
0xC7	lopass
0xC8	bandpass
0xC9	bandstop
0xCA	fir
0xCB	iir
0xCC	firt
0xCD	iirt
0xCE	biquad
0xCF	fft
0xD0	ifft
0xD1	rms
0xD2	gain
0xD3	balance
0xD4	decimate
0xD5	upsamp
0xD6	downsamp
0xD7	samphold

0xD8	`delay`	0xE2	`gettune`
0xD9	`delay1`	0xE3	`settune`
0xDA	`fracdelay`	0xE4	`ftsr`
0xDB	`comb`	0xE5-0xEF	**(free)**
0xDC	`allpass`	0xF0	`<symbol>`
0xDD	`chorus`	0xF1	`<integer>`
0xDE	`flange`	0xF2	`<number>`
0xDF	`reverb`	0xF3	`<string>`
0xE0	`compress`	0xF4-0xFF	**(reserved)**
0xE1	`pcompress`	0xFF	`<EOO>`

Annex B Encoding (informative)

This Annex, provided for informative purposes only, provides guidelines as to the functioning of a typical Structured Audio encoder. As of this writing, the capabilities of audio-processing algorithms for performing automatic source separation, identification of instruments, identification of spatial qualities, etc., are not sufficiently advanced to perform fully automatic structured-audio encoding. Instead, for the near future, encoding Structured Audio bitstreams will likely occur in conjunction with authoring tools, where sound designers compose and orchestrate soundtracks and design instruments, and the results are then packaged into a legal Structured Audio bitstream.

[to be expanded]

Annex C lex/yacc grammars for SAOL (informative)

C.1 Introduction

This Annex provides grammars using the widely-available tools ‘lex’ and ‘yacc’ which conform to the SAOL specification in this document. They are provided for informative purposes only; implementors are free to use whichever tools they desire, or no tools, in building an implementation.

The reference software for Structured Audio in ISO 14496-5 builds the lexer and parser for SAOL out of these grammars by augmenting them with more processing and data-structure construction.

C.2 Lexical grammar for SAOL in lex

```
STRCONST     \"(\\.|[^\\"])*\"
IDENT        [a-zA-Z_][a-zA-Z0-9_]*
INTGR        [0-9]+
NUMBER [0-9]+(\.[0-9]*)?(e[-+]?[0-9]+)?|-?\.[0-9]*(e-+?[0-9]+)?

%%

"//"         { }
"aopcode"    { return(AOPCODE) ; }
"asig"       { return(ASIG) ; }
"else"       { return(ELSE) ; }
"exports"    { return(EXPORTS) ; }
"extend"     { return(EXTEND) ; }
"global"     { return(GLOBAL) ; }
"if"         { return(IF) ; }
"imports"    { return(IMPORTS); }
"inchannels" { return(INCHANNELS) ; }
"inputmod"   { return(INPUTMOD) ; }
"instr"      { return(INSTR); }
"iopcode"    { return(IOPCODE); }
"ivar"       { return(IVAR) ; }
"kopcode"    { return(KOPCODE); }
"krate"      { return(KRATE) ; }
"ksig"       { return(KSIG) ; }
"map"        { return(MAP) ; }
"oparray"    { return(OPARRAY) ; }
"opcode"     { return(OPCODE) ; }
"outbus"     { return(OUTBUS) ; }
"outchannels"{ return(OUTCHANNELS) ; }
"output"     { return(OUTPUT) ; }
"return"     { return(RETURN) ; }
"route"      { return(ROUTE) ; }
"send"       { return(SEND) ; }
"sequence"   { return(SEQUENCE) ; }
"sbsynth"    { return(SFSYNTH) ; }
"spatialize" { return(SPATIALIZE) ; }
"srate"      { return(SRATE); }
"table"      { return(TABLE); }
"tablemap"   { return(TABLEMAP); }
"template"   { return(TEMPLATE); }
"turnoff"    { return(TURNOFF); }
"while"      { return(WHILE); }
"with"       { return(WITH); }
"xsig"       { return(XSIG) ; }
"&&"         { return(AND); }
"||"         { return(OR); }
">="         { return(GEQ); }
"<="         { return(LEQ); }
```

```
"!="          { return(NEQ); }
"=="          { return(EQEQ); }
"-"           { return(MINUS); }
"*"           { return(STAR); }
"/"           { return(SLASH); }
"+"           { return(PLUS); }
">"           { return(GT); }
"<"           { return(LT); }
"?"           { return(Q); }
":"           { return(COL); }
"("           { return(LP); }
")"           { return(RP); }
"{"           { return(LC); }
"}"           { return(RC); }
"["           { return(LB); }
"]"           { return(RB); }
";"           { return(SEM); }
","           { return(COM); }
"="           { return(EQ); }
"!"           { return(NOT); }

{STRCONST}    { return(STRCONST); }
{IDENT}      { return(IDENT) ; }
{INTGR}       { return(INTGR) ; }
{NUMBER}      { return(NUMBER) ; }
[ \t\n]       { }
.                 { printf("Line %d: Unknown character: '%s'\n",yyline,yytext);
}

%%
```

C.3 Syntactic grammar for SAOL in yacc

```
%start orcfile
%left AND OR
%nonassoc LT GT LEQ GEQ EQEQ NEQ
%left PLUS MINUS
%left STAR SLASH
%right UNOT
%right UMINUS
%token HIGHEST

%%

orcfile         : proclist
                ;
proclist        : proclist instrdecl
                | proclist opcodedecl
                | proclist
                | proclist
                | /* null */
                | error
                ;
instrdecl       : INSTR IDENT LP identlist RP LC vardecls block
                | error
                ;
opcodedecl      : optype IDENT LP paramlist RP LC opvardecls block RC
                | error
                ;
globaldecl      : GLOBAL LC globalblock
                | error
                ;
templatedecl    : TEMPLATE LT identlist GT LP identlist RP
                   MAP LC identlist RC
                   WITH LC mapblock RC LC
                   vardecls block
                | error
                ;
mapblock        : mapblock COM LT terminal_list
                | LT terminal_list GT
                | /* null */
                | error
                ;
terminal_list   : terminal_list COM
                | terminal
                | error
                ;
terminal        : IDENT
                | const
                | STRCONST
                ;
globalblock     : globalblock globaldef
                | /* null */
                | error
                ;
globaldef       : rtparam
                | vardecl
                | routedef
                | senddef
                | inputmoddef
                | seqdef
                | error
                ;
rtparam         : SRATE INTGR SEM
                | KRATE INTGR SEM
                | INCHANNELS INTGR SEM
                | OUTCHANNELS INTGR SEM
```

```
                | error
                ;
routedef        : ROUTE LP IDENT COM identlist RP SEM
senddef         : SEND LP IDENT SEM exprlist SEM identlist RP SEM
                | error
                ;
inputmoddef     : INPUTMOD LP IDENT SEM exprlist RP SEM
                | error
                ;
seqdef          : SEQUENCE LP identlist RP SEM
                | error
                ;
block           : block statement
                | /* null */
                | error
                ;
statement       : lvalue EQ expr SEM
                | expr SEM
                | IF LP expr RP LC block RC
                | IF LP expr RP LC block RC ELSE LC block RC

                | WHILE LP expr RP LC block RC
                | INSTR IDENT LP exprlist RP SEM
                | OUTPUT LP exprlist RP SEM
                | SFSYNTH LP exprlist SEM identlist SEM exprlist RP SEM
                | SPATIALIZE LP exprlist RP SEM
                | OUTBUS LP IDENT COM exprlist RP SEM
                | EXTEND LP expr RP SEM
                | TURNOFF SEM

                | RETURN LP exprlist RP SEM

                | error
                ;
lvalue          : IDENT
                | IDENT LB expr RB
                | error
                ;
identlist       : identlist COM IDENT
                | IDENT
                | /* null */
                | error
                ;
paramlist       : paramlist COM paramdecl
                | paramdecl
                |   /* null */
                | error
                ;
vardecls        : vardecls vardecl
                | /* null */
                | error
                ;
vardecl               : taglist stype namelist SEM
                | tabledecl SEM
                | TABLEMAP IDENT LP identlist RP SEM
                | error
                ;
opvardecls      : opvardecls opvardecl
                | /* null */
                | error
                ;
opvardecl       : taglist otype namelist SEM
                | tabledecl SEM
                | error
                ;
paramdecl       : otype name
                | error
```

```
                ;
namelist        : namelist COM name
                | name
                | error
                ;
name            : IDENT
                | IDENT LB INTGR RB
                | error
                ;
stype           : IVAR
                | KSIG
                | ASIG
                | TABLE
                | OPARRAY
                | error
                ;
otype           : XSIG
                | stype
                | error
                ;
tabledecl       : TABLE IDENT LP IDENT COM exprstrlist RP
                | error
                ;
taglist               : IMPORTS
                | EXPORTS
                | IMPORTS EXPORTS
                | EXPORTS IMPORTS
                |
                | error
                ;
optype          : AOPCODE
                | KOPCODE
                | IOPCODE
                | OPCODE
                | error
                ;
expr            : IDENT
                | const
                | IDENT LB expr RB
                | IDENT LP exprlist RP
                | IDENT LB expr RB LP exprlist RP
                | expr Q expr COL expr
                | expr LEQ expr
                | expr GEQ expr
                | expr NEQ expr
                | expr EQEQ expr
                | expr GT expr
                | expr LT expr
                | expr AND expr
                | expr OR expr
                | expr PLUS expr
                | expr MINUS expr
                | expr STAR expr
                | expr SLASH expr
                | NOT expr     %prec UNOT
                | MINUS expr
                | LP expr RP
                | error
                ;
exprlist        : exprlist COM expr
                | expr
                | /* null */
                | error
                ;
exprstrlist     : exprstrlist COM expr
                | exprstrlist COM STRCONST
                | STRCONST
```

```
                | expr
                | error
                ;
const           : INTGR
                | NUMBER
                | error
                ;
%%
```

Annex D Detokenisation (informative)

This Annex describes the process of converting a tokenised bitstream representation into a human-readable program in the textual SAOL format. It is provided for informative purposes only. The result of this process is much more satisfying if the bitstream contains the optional symbol table element, see Subclause 5.1 Bitstream syntax and semantics.

[Needs to be completed or removed].

Index to Subpart 5

Page numbers in **boldface** refer to definitions in the Glossary, Subclause 5.0.3 Glossary of Terms.

ISO/IEC JTC 1/SC 29 N 2203

Date: 1997-10-31

ISO/IEC CD 14496-3 Subpart 6

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information Technology - Coding of Audiovisual Objects – Part 3: Audio

Subpart 6 : Text-to-Speech

Document type: International Standard
Document subtype: Not applicable
Document stage: (30) Committee
Document language: E

C:\MPEG-CD\1903TTS.DOC ISOSTD Version 1.8 1996-10-30

Contents

Foreword

Boilerplate text for ISO standards:

ISO (the International Organisation for Standardisation) is a world-wide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organisations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardisation.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Boilerplate text for ISO/IEC standards:

ISO (the International Organisation for Standardisation) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Introduction

This document is the 6th subpart of the MPEG-4 Audio phase 1 CD (ISO Document Number: CD14496-3). The MPEG-4 Audio phase 1 CD is divided into six sections: section 1 (Main Document), section 2 (MPEG-4 Audio Parametric Coding), section 3 (MPEG-4 Audio Code Excited Linear Prediction (CELP) Coding), section 4 (MPEG-4 Audio Transform Coding), section 5 (MPEG-4 Audio Structured Audio), and section 6 (MPEG-4 Audio Text-to-Speech Coding, this file).

Information Technology - Coding of Audiovisual Objects

Part 3 : Audio

Subpart 6 : Text-to-Speech

1 Scope

This subpart of ISO/IEC CD 14496-3 specifies the coded representation of MPEG-4 Audio Text-to-Speech (M-TTS) and its decoder for high quality synthesized speech and for enabling various applications. The exact synthesis method is not a standardization issue partly because there are already various speech synthesis techniques.

This subpart of ISO/IEC CD 14496-3 is intended for application to M-TTS functionalities such as those for facial animation (FA) and moving picture (MP) interoperability with a coded bitstream. The M-TTS functionalities include a capability of utilizing prosodic information extracted from natural speech. They also include the applications to the speaking device for FA tools and a dubbing device for moving pictures by utilizing lip shape and input text information.

The text-to-speech (TTS) synthesis technology is recently becoming a rather common interface tool and begins to play an important role in various multimedia application areas. For instance, by using TTS synthesis functionality, multimedia contents with narration can be easily composed without recording natural speech sound. Moreover, TTS synthesis with facial animation (FA) / moving picture (MP) functionalities would possibly make the contents much richer. In other words, TTS technology can be used as a speech output device for FA tools and can also be used for MP dubbing with lip shape information. In MPEG-4, common interfaces only for the TTS synthesizer and for FA/MP interoperability are proposed. The proposed MPEG-4 Audio TTS functionalities can be considered as a superset of the conventional TTS framework. This TTS synthesizer can also utilize prosodic information of natural speech in addition to input text and can generate much higher quality synthetic speech. The interface bitstream format is strongly user-friendly: if some parameters of the prosodic information are not available, the missed parameters are generated by utilizing pre- established rules. The functionalities of the M-TTS thus range from conventional TTS synthesis function to natural speech coding and its application areas, i.e., from a simple TTS synthesis function to those for FA and MP.

2 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 10646-1:1993 *Information Technology - Universal Multiple-Octet Coded Character Set (UCS) - Prt 1: Architecture and Basic Multilingual Plane*.

The Unicode Consortium: 1996 *The Unicode Standard Version 2.0* (Published by Addison-Wesley Developers Press)

3 Definitions

International Phonetic Alphabet; IPA

The worldwide agreed symbol set to represent various phonemes appearing in human speech.

lip shape pattern

A number that specifies a particular pattern of the preclassified lip shape.

lip synchronization
A functionality that synchronizes speech with corresponding lip shapes.

MPEG-4 Audio Text-to-Speech Decoder
A device that produces synthesized speech by utilizing the M-TTS bitstream while supporting all the M-TTS functionalities such as speech synthesis for FA and MP dubbing.

moving picture dubbing
A functionality that assigns synthetic speech to the corresponding moving picture while utilizing lip shape pattern information for synchronization.

M-TTS sentence
This defines the information such as prosody, gender, and age for only the corresponding sentence to be synthesized.

M-TTS sequence
This sequence can contain multiple M-TTS sentences and additional information that affects all M-TTS sentences in the sequence.

phoneme-to-FAP converter
A device that converts phoneme information to FAPs.

text-to-speech synthesizer
A device producing synthesized speech according to the input sentence character strings.

trick mode
A set of functions that enables stop, play, forward, and backward operations for users.

4 Symbols and Abbreviations

The following symbols and the abbreviations are defined to describe this International Standard.

F0: fundamental frequency (pitch frequency)

CELP: code excited linear prediction

DEMUX: demultiplexer

FA: facial animation

FAP: facial animation parameter

ID: identifier

IPA: International Phonetic Alphabet

MP: moving picture

M-TTS: MPEG-4 Audio TTS

STOD: story teller on demand

TTS: text-to-speech

5 Method of Describing Syntax

The bitstream retrieved by the decoder is described in 6. Each data item in the bitstream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

6 MPEG-4 Audio Text-to-Speech Bitstream Syntax

6.1 MPEG-4 Audio TTSSpecificConfig

Syntax	No. of bits	Mnemonic
TTSSpecificConfig() {		
Language_Code	18	uimsbf
Video_Enable	1	bslbf
Lip_Shape_Enable	1	bslbf
Trick_Mode_Enable	1	bslbf
}		

6.2 MPEG-4 Audio Text-to-Speech Payload

Syntax	No. of bits	Mnemonic
AlPduPayload {		
TTS_Sequence()		
}		
TTS_Sequence() {		
TTS_Sequence_Start_Code	sc+8=32	bslbf
TTS_Sequence_ID	5	uimsbf
Language_Code	18	uimsbf
Gender_Enable	1	bslbf
Age_Enable	1	bslbf
Speech_Rate_Enable	1	bslbf
Prosody_Enable	1	bslbf
Video_Enable	1	bslbf
Lip_Shape_Enable	1	bslbf
Trick_Mode_Enable	1	bslbf
do {		
TTS_Sentence()		
} while (next_bits()==TTS_Sentence_Start_Code)		
}		

6.3 MPEG-4 Audio Text-to-Speech Sentence

Syntax	No. of bits	Mnemonic
TTS_Sentence() {		
TTS_Sentence_Start_Code	sc+8=32	bslbf
TTS_Sentence_ID	10	uimsbf

Silence	1	bslbf
if (Silence) {		
Silence_Duration	12	uimsbf
}		
else {		
if (Gender_Enable) {		
Gender	1	bslbf
}		
if (Age_Enable) {		
Age	3	uimsbf
}		
if (!Video_Enable && Speech_Rate_Enable) {		
Speech_Rate	4	uimsbf
}		
Length_of_Text	12	uimsbf
for (j=0; j<Length_of_Text; j++) {		
TTS_Text	8	bslbf
}		
if (Prosody_Enable) {		
Dur_Enable	1	bslbf
F0_Contour_Enable	1	bslbf
Energy_Contour_Enable	1	bslbf
Number_of_Phonemes	10	uimsbf
Phoneme_Symbols_Length	13	uimsbf
for (j=0 ; j<Phoneme_Symbols_Length ; j++) {		
Phoneme_Symbols	8	bslbf
}		
for (j=0 ; j<Number_of_Phonemes ; j++) {		
if(Dur_Enable) {		
Dur_each_Phoneme	12	uimsbf
}		
if (F0_Contour_Enable) {		
num_F0	5	uimsbf
for (j=0; j<num_F0;j++) {		
F0_Contour_each_Phoneme	8	uimsbf
F0_Contour_each_Phoneme_Time	12	uimsbf
}		
}		
if (Energy_Contour_Enable) {		
Energy_Contour_each_Phoneme	8*3=24	uimsbf
}		
}		
}		

if (Video_Enable) {		
Sentence_Duration	16	uimsbf
Position_in_Sentence	16	uimsbf
Offset	10	uimsbf
}		
if (Lip_Shape_Enable) {		
Number_of_Lip_Shape	10	uimsbf
for (j=0 ; j<Number_of_Lip_Shape ; j++) {		
Lip_Shape_in_Sentence	16	uimsbf
Lip_Shape	8	uimsbf
}		
}		
}		
}		

7 Semantics for MPEG-4 Audio Text-to-Speech Bitstream

7.1 General

The M-TTS bitstream is described in three parts; M-TTS specific configuration in 7.2, M-TTS sequence in 7.3, and M-TTS sentence in 7.4. M-TTS system interface contains information that specific M-TTS decoder(s) should be used. M-TTS sequence can contain multiple M-TTS sentences. The information in the M-TTS sequence affects all M-TTS sentences while the information in the M-TTS sentence affects only the corresponding M-TTS sentence.

7.2 MPEG-4 Audio TTSSpecificConfig

Language_Code - When this is „00“ (00110000 00110000 in binary), the IPA is to be sent. In all other languages, this is the ISO 639 Language Code. In addition to this 16 bits, two bits that represent dialects of each language is added is added at the end (user defined).

Video_Enable - This is a one-bit flag which is set to `1` when the MPEG-4 Audio TTS decoder works with MP.

Lip_Shape_Enable - This is a one-bit flag that is set to `1` when the coded input bitstream has lip shape information.

Trick_Mode_Enable - This is a one-bit flag which is set to `1` when the coded input bitstream permits trick mode functions such as stop, play, forward, and backward.

7.3 MPEG-4 Audio Text-to-Speech Payload

TTS_Sequence_Start_Code - This is a bit string `000000XX` in hexadecimal that identifies the start of the M-TTS coded sequence.

TTS_Sequence_ID - This is a five-bit ID to uniquely identify each object appearing in one scene.

Language_Code - When this is „00“ (00110000 00110000 in binary), the IPA is to be sent. In all other languages, this is the ISO 639 Language Code. In addition to this 16 bits, two bits that represent dialects of each language is added is added at the end (user defined).

Gender_Enable - This is a one-bit flag which is set to ‘1‘ when the gender information exists.

Age_Enable - This is a one-bit flag which is set to ‘1‘ when the age information exists.

Speech_Rate_Enable - This is a one-bit flag which is set to ‘1‘ when the speech rate information exists.

Prosody_Enable - This is a one-bit flag which is set to ‘1‘ when the prosody information exists.

VideoEnable – This is a one-bit flag which is set to ‘1’ when the M-TTS decoder works with MP. In this case, M-TTS should synchronize synthetic speech to MP and accommodate the functionality of ttsForward and ttsBackward. When VideoEnable flag is set, M-TTS decoder uses system clock to select adequate TTS-Sentence frame and fetches Sentence_Duration, Position_in_Sentence, Offset data. TTS synthesizer assigns appropriate duration for each phoneme to meet Sentence_Duration. The starting point of speech in a sentence is decided by Position_in_Sentence. If Position_in_Sentence equals 0 (the starting point is the initial of sentence), TTS uses Offset as a delay time to synchronize synthetic speech to MP.

LipshapeEnable – This is a one-bit flag which is set to ‘1’ when the coded input bitstream has lip shape information. With lip shape information, M-TTS can request FA tool to change lip shape according to timing information (Lip_Shape_in_Sentence) and predifined lip shape pattern.

Trick_Mode_Enable - This is a one-bit flag which is set to `1` when the coded input bitstream permits trick mode functions such as stop, play, forward, and backward.

*****Ed : Right position of this syntax element should be considered later with system group. Object_Descriptor or node definition in BIFS?***

7.4 MPEG-4 Audio Text-to-Speech Sentence

TTS_Sentence_Start_Code - This is a bit string `000000XX` in hexadecimal which identifies the start of the sentence to be processed.

TTS_Sentence_ID - This is a ten-bit ID to uniquely identify a sentence in the M-TTS text data sequence for indexing purpose.

Silence - This is a one-bit flag which is set to `1` when the current position is silence.

Silence_Duration - This defines the time duration of the current silence segment in milliseconds. It has a value from 1 to 1023. The value `0` is prohibited.

Gender - This is a one-bit which is set to `1` if the gender of the synthetic speech producer is male and `0`, if female.

Age - This represents the age of the speaker for synthetic speech. The meaning of age is defined in the table below.

Age	age of the speaker
000	below 6
001	6 - 12
010	13 - 18
011	19 - 25
100	26 - 34
101	35 - 45
110	45 - 60
111	over 60

SpeechRate – This defines the synthetic speech rate in 16 levels. This is user-defined since the average speaking speed of each language differs.

Length_of_Text - This identifies the length of the TTS_Text data in bytes.

TTS_Text - This is a character string containing the input text.

Dur_Enable - This is a one bit flag which is set to `1`when the duration information for each phoneme exists.

F0_Contour_Enable - This is a one bit flag which is set to `1` when the pitch contour information for each phoneme exists.

Energy_Contour_Enable - This is a one bit flag which is set to `1` when the energy contour information for each phoneme exists.

Number_of_Phonemes - This defines the number of phonemes needed for speech synthesis of the input text.

Phonemes_Symbols_Length – This identifies the length of Phonemes_Symbols (IPA code) data in bytes since the IPA code has optional modifiers and dialect codes.

Phoneme_Symbols - This defines the indexing number for the current phoneme by using the Unicode 2.0 numbering system. Each phoneme symbol is represented as a number for the corresponding IPA. Three two-byte numbers can be used for each IPA representation including a two-byte integer for the character, and an optional two-byte integer for the spacing modifier, and another optional two-byte integer for the diacritical mark.

Dur_each_Phoneme - This defines the duration of each phoneme in msec.

num_F0 – This defines the number of F0 values specified for the current phoneme.

F0_Contour_each_Phoneme – This defines F0 value in Hz at time instant F0_Contour_each_Phoneme_Time.

F0_Contour_each_Phoneme_Time – This defines the integer number of the time in ms for the position of the F0_Contour_each_Phoneme.

Energy_Contour_each_Phoneme - This defines the energy level of current phoneme in integer dB for three points (0%, 50%, 100% positions of the phoneme are for three points).

Sentence_Duration - This defines the duration of the sentence in msec.

Position_in_Sentence - This defines the position of the current stop in a sentence as an elapsed time in msec.

Offset - This defines the duration of a very short pause before the start of synthesized speech output in msec.

Number_of_Lip_Shape - This defines the number of lip-shape patterns to be processed.

Lip_Shape_in_Sentence - This defines the position of each lip shape from the beginning of the sentence in msec.

Lip_Shape - This defines the indexing number for the current lip-shape pattern to be processed that is defined in the table 12-5 in CD14496-2.

8 MSDL-S Representation of the Bitstream

8.1 ttsSpecificConfig

class ttsSpecificConfig {

```
        unsigned int(18) languageCode;
        bit(1) videoEnable;
        bit(1) lipShapeEnable;
        bit(1) trickModeEnable;
}
```

8.2 alPduPayload

```
class alPduPayload {
        ttsSequence sequence;
}
class ttsSequence : aligned bit(32) tts_sequence_start_code {
        unsigned int(5) ttsSequenceID;
        unsigned int(18) languageCode;
        bit(1) genderEnable;
        bit(1) ageEnable;
        bit(1) speechRateEnable;
        bit(1) prosodyEnable;
        bit(1) videoEnable;
        bit(1) lipShapeEnable;
        bit(1) trickModeEnable;
        ttsSentence sentence;
}
```

8.3 ttsSentence

```
class ttsSentence : aligned bit(32) tts_sentence_start_code {
        unsigned int(10) ttsSentenceID;
        bit(1) silence;
        if (silence) {
                unsigned int(12) silenceDuration;
        } else {
                if (genderEnable) {
                        unsigned int(1) gender;
                }
```

```
                if (ageEnable) {
                        unsigned int(3) age;
                }
                if (!videoEnable && speechRateEnable) {
                        unsigned int(4) speechRate;
                }
                unsigned int(12) lengthText;
                unsigned int(8) ttsText[lengthText];
                if (prosodyEnable) {
                        ttsProsody prosody;
                }
                if (videoEnable) {
                        unsigned int(16) sentenceDuration;
                        unsigned int(16) positionInSentence;
                        unsigned int(10) offset;
                }
                if (lipShapeEnable) {
                        unsigned int(10) numberLipShape
                        unsigned int(16) LipShapeInSentence[numberLipShape]
                        unsigned int(8) lipShape[numberLipShape];
                }
        }
}
```

8.4 ttsProsody

```
class ttsProsody {
        bit(1) durEnable;
        bit(1) f0ContourEnable;
        bit(1) energyContourEnable;
        unsigned int(10) numberPhonemes;
        unsigned int(13) phonemeSymbolsLength;
```

```
        unsigned int(8) phonemeSymbols[phonemeSymbolsLength] ;
        for (j=0 ; j<NumberofPhonemes ; j++) {
                if (durEnable) {
                        unsigned int(12) durEachPhoneme ;
                }
                if (f0ContourEnable) {
                        unsigned int(5) numf0 ;
                        unsigned int(8) f0ContourEachPhoneme[numf0] ;
                        unsigned int(12) f0ContourEachPhonemeTime[numf0] ;
                }
                if (energyContourEnable) {
                        unsigned int(8) energyContourEachPhoneme[3];
                }
        }
}
```

9 MPEG-4 Audio Text-to-Speech Decoding Process

The architecture of the M-TTS decoder is described below and only the interfaces relevant to the M-TTS decoder are the subjects of standardization. The number above each arrow indicates the section describing each interface.

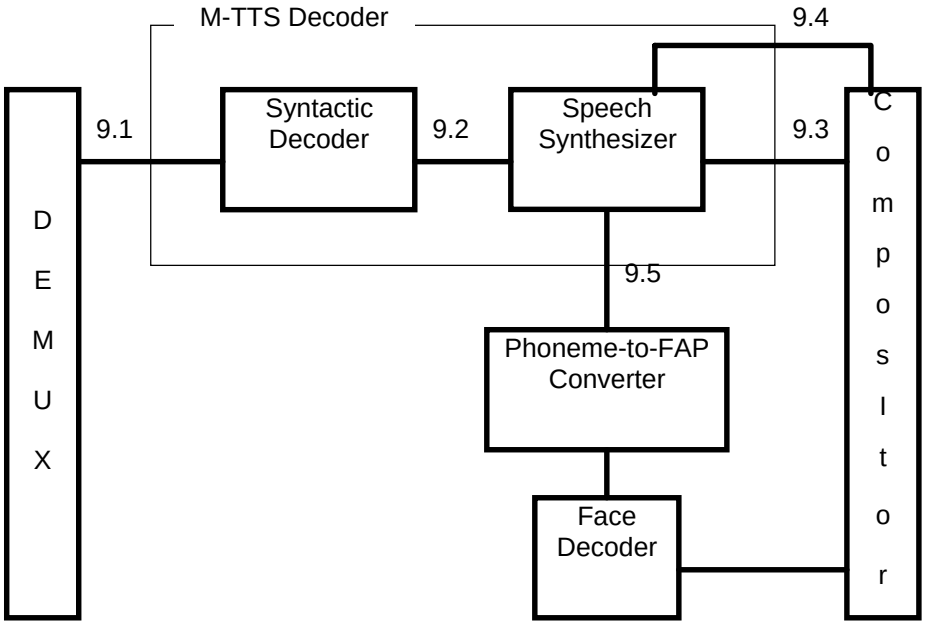


Figure 1 — MPEG-4 Audio TTS decoder architecture.

In this architecture the following types of interfaces can be distinguished:

Interface between DEMUX and the syntactic decoder

Interface between the syntactic decoder and the speech synthesizer

Interface from the speech synthesizer to the compositor

Interface from the compositor to the speech synthesizer

Interface between the speech synthesizer and the phoneme-to-FAP converter

9.1 Interface between DEMUX and Syntactic Decoder

Receiving a bitstream, DEMUX passes coded M-TTS bitstreams to the syntactic decoder.

9.2 Interface between Syntactic Decoder and Speech Synthesizer

Receiving a coded M-TTS bitstream, the syntactic decoder passes some of the following bitstreams to the speech synthesizer.

Input type of the M-TTS data: specifies synchronized operation with FA or MP

Control commands stream: Control command sequence

Input text: character string(s) for the text to be synthesized

Auxiliary information: Prosodic parameters including phoneme symbols

Lip shape patterns

Information for trick mode operation

The MSDL-S representations of the classes for this interface are defined in 8.

9.3 Interface from Speech Synthesizer to Compositor

This interface is identical to the interface for digitized natural speech to the compositor. The class ttsAudioInterface defines the data structure for the interface of encoded speech from the speech synthesizer to the compositor.

```
class ttsAudioInterface {

    DigitalSpeech   *dgspeech;

}
```

9.4 Interface from Compositor to Speech Synthesizer

This interface is defined to allow the local control of the synthesized speech by users. This user interface can support trick mode of the synthesized speech in synchronization with MP and can change some prosodic properties of the synthesized speech by using the class **ttsControl** defined as follows:

```
class ttsControl  {
        void ttsPlay():
        void ttsForward():
        void ttsBackward():
        void ttsStopSyllable():
        void ttsStopWord():
        void ttsStopPhrase():
        void ttsChangeSpeechRate(unsigned int(4) speed):
        void ttsChangePitchDynamicRange(unsigned int(4) level):
        void ttsChangePitchHeight(unsigned int(4) height):
        void ttsChangeGender(unsigned int(1) gender):
        void ttsChangeAge(unsigned int(3) age):
}
```

The member function `ttsPlay` allows a user to start speech synthesis in the forward direction while `ttsForward` and `ttsBackward` enable the user to change the starting play position in forward and backward direction, respectively. The `ttsStopSyllable, ttsStopWord,` and `ttsStopPhrase` functions define the interface for users to stop speech synthesis at the specified boundary such as syllable, word, and phrase. The member function `ttsChangeSpeechRate` is an interface to change the synthesized speech rate. The argument `speed` can have the numbers from 1 to 16. The member function `ttsChangePitchDynamicRange` is an interface to change the dynamic range of the pitch of synthesized speech. By using the argument of this function, `level`, a user can change the dynamic range from 1 to 16. Also a user can change the pitch height from 1 to 16 by using the argument `height` in the member function `ttsChangePitchHeight`. The member functions `ttsChangeGender` and `ttsChangeAge` allow a user to change the gender and the age of the synthetic speech producer by assigning numbers, as defined in 7.3, to their arguments, gender and age, respectively.

9.5 Interface between Speech Synthesizer and Phoneme-to-FAP Converter

In the MPEG-4 framework, the speech synthesizer and the face decoder are driven synchronously. The speech synthesizer generates synthetic speech according to the phonemeDuration. At the same time, TTS gives phonemeSymbol and phonemeDuration to Phoneme-to-FAP converter. The face decoder generates relevant facial animation according to the phonemeSymbol and the phonemeDuration. In this way, the synthesized speech and facial animation have relative synchronization except the absolute composition time. The synchronization of the absolute composition time comes from the same composition time stamp of TTS bitstream and facial animation bitstream. When the lipShapeEnable is set, the lipShapeInSentence can be used to generate the phonemeDuration.

The class ttsFAPinterface defines the data structure for the interface between the speech synthesizer and the phoneme-to-FAP converter.

```
class ttsFAPinterface {
        unsigned int(8)  phonemeSymbol;
        unsigned int(12) phonemeDuration;
        unsigned int(8)  f0Average;
        unisgned int(1)  stress;
        unsigned int(1)  word_begin ;
}
```

Annex A
(informative)
Applications of MPEG-4 Audio Text-to-Speech Decoder

A.1 General

This annex part describes application scenarios for the M-TTS decoder.

A.2 Application Scenario: MPEG-4 Story Teller on Demand (STOD)

In the STOD application, users can select a story from a huge database of story libraries which are stored in hard disks or compact disks. The STOD system reads aloud the story via the M-TTS decoder with the MPEG-4 facial animation tool or with appropriately selected images. The user can stop and resume speaking at any moment he wants through user interfaces of the local machine (for example, mouse or keyboard).The user can also select the gender, age, and the speech rate of the electronic story teller.

The synchronization between the M-TTS decoder with the MPEG-4 facial animation tool is realized by using the same composition time of the M-TTS decoder for the MPEG-4 facial animation tool.

A.3 Application Scenario : MPEG-4 Audio Text-to-Speech with Moving Picture

In this application, synchronized playback of the M-TTS decoder and encoded moving picture is the most important issue. The architecture of the M-TTS decoder can provide several granularities of synchronization. Aligning the composition time of each TTS_Sentence(), coarse granularity of synchronzation and trick mode functionality can be easily achieved. To get finer granularity of synchronization, the information about the lip shape events would be utilized. The finest granularity of syncronization can be achieved by using the prosody information and the video-related information such as Sentence_Duration, Position_in_Sentence, and Offset.

With this synchronization capability, the M-TTS decoder can be used for moving picture dubbing by utilizing the lip shape pattern information.