Doc. CS/T3-613
2 April 2004

# Candidate Standard Amendment No. 1 to ATSC Digital Audio Compression (AC-3) Standard, Doc. A/52A

# Annex E: Enhanced AC-3 Bit Stream Syntax

**Advanced Television Systems Committee**
1750 K Street, N.W.
Suite 1200
Washington, D.C. 20006
www.atsc.org

The Advanced Television Systems Committee, Inc., is an international, non-profit organization developing voluntary standards for digital television. The ATSC member organizations represent the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries.

Specifically, ATSC is working to coordinate television standards among different communications media focusing on digital television, interactive systems, and broadband multimedia communications. ATSC is also developing digital television implementation strategies and presenting educational seminars on the ATSC standards.

ATSC was formed in 1982 by the member organizations of the Joint Committee on InterSociety Coordination (JCIC): the Electronic Industries Association (EIA), the Institute of Electrical and Electronic Engineers (IEEE), the National Association of Broadcasters (NAB), the National Cable Television Association (NCTA), and the Society of Motion Picture and Television Engineers (SMPTE). Currently, there are approximately 140 members representing the broadcast, broadcast equipment, motion picture, consumer electronics, computer, cable, satellite, and semiconductor industries.

ATSC Digital TV Standards include digital high definition television (HDTV), standard definition television (SDTV), data broadcasting, multichannel surround-sound audio, and satellite direct-to-home broadcasting.

## About the Candidate Standard

This specification is being put forth as a Candidate Standard by the T3/S6 Specialist Group on Video and Audio Coding. ATSC members and non-members are encouraged to review and implement this specification and return comments to cs_amend_editor@atsc.org. ATSC Members can also send comments directly to the T3/S6 Specialist Group. The ATSC believes this specification is stable. It is expected to progress to Proposed Standard within a period of time ending 23 September 2004.

*This Amendment adds the following Annex E to A/52A*:

# Annex E:
# Enhanced AC-3 Bit Stream Syntax – Version 1.70 (Normative)

## 1. SCOPE

This Annex to ATSC document A/52 describes the bit stream syntax used by Enhanced AC-3 bit streams and decoders. Enhanced AC-3 bit streams are similar in nature to standard AC-3 bit streams, but are not backwards compatible (i.e., they are not decodable by standard AC-3 decoders). This Annex outlines the differences between the stream types, and specifies the decoding of Enhanced AC-3 bit streams. This Annex is normative in applications that specify the use of Enhanced AC-3.

## 2. SPECIFICATION

### 2.1 Indication of Enhanced AC-3 bit stream syntax

An AC-3 bit stream is indicated as using the Enhanced AC-3 bit stream syntax described in this Annex when the bit stream identification (bsid) field is set to 12.

### 2.2 Syntax Specification

A continuous audio bit stream consists of a sequence of synchronization frames:

| **Syntax** |
|---|
| bit stream()<br>{<br>  while(true)<br>  {<br>  **syncframe()** ;<br>  }<br>} /* end of bit stream */ |

The syncframe consists of the **syncinfo**, **bsi** and **audfrm** fields, up to 6 coded **audblk** fields, the **auxdata** field, and the **errorcheck** field.

| **Syntax** |
|---|
| syncframe()<br>{<br>  **syncinfo()** ;<br>  **bsi()** ;<br>  **audfrm()** ;<br>  for(blk = 0; blk < number_of_blocks_per_syncframe; blk++)<br>  {<br>  **audblk()** ;<br>  }<br>  **auxdata()** ;<br>  **errorcheck()** ;<br>} /* end of syncframe */ |

Each of the bit stream elements, and their length, are itemized in the following pseudo code. Note that all bit stream elements arrive most significant bit first, or left bit first, in time.

### 2.2.1 syncinfo - Synchronization Information

| Syntax | Word Size |
|---|---|
| syncinfo() | |
| { | |
| **syncword** | 16 |
| } /* end of syncinfo */ | |

### 2.2.2 bsi – Bit Stream Information

| Syntax | Word Size |
|---|---|
| bsi() | |
| { | |
| **strmtyp** | 2 |
| **substreamid** | 3 |
| **frmsiz** | 11 |
| **fscod** | 2 |
| if(fscod == 0x3) | |
| { | |
| **fscod2** | 2 |
| numblkscod = 0x3 /* six blocks per frame */ | |
| } | |
| else | |
| { | |
| **numblkscod** | 2 |
| } | |
| **acmod** | 3 |
| **lfeon** | 1 |
| **bsid** | 5 |
| **dialnorm** | 5 |
| **compre** | 1 |
| if(compre) **{compr}** | 8 |
| if(acmod == 0x0) /* if 1+1 mode (dual mono, so some items need a second value) */ | |
| { | |
| **dialnorm2** | 5 |
| **compr2e** | 1 |
| if(compr2e) **{compr2}** | 8 |
| } | |
| if(strmtyp == 0x1) /* if dependent stream */ | |
| { | |
| **chanmape** | 1 |
| if(chanmape) **{chanmap}** | 16 |
| } | |
| **mixmdate** | 1 |
| if(mixmdate) /* Mixing metadata */ | |
| { | |
| if(acmod > 0x2) /* if more than 2 channels */ **{dmixmod}** | 2 |
| if((acmod & 0x1) && (acmod > 0x2)) /* if three front channels exist */ | |
| { | |
| **ltrtcmixlev** | 3 |
| **lorocmixlev** | 3 |
| } | |

```
        if(acmod & 0x4) /* if a surround channel exists */
        {
            ltrtsurmixlev ..... 3
            lorosurmixlev ..... 3
        }
        if(strmtyp == 0x0) /* if independent stream */
        {
            pgmscle ..... 1
            if(pgmscle) {pgmscl} ..... 6
            if(acmod == 0x0) /* if 1+1 mode (dual mono, so some items need a second value) */
            {
                pgmscl2e ..... 1
                if(pgmscl2e) {pgmscl2} ..... 6
            }
            extpgmscle ..... 1
            if(extpgmscle) {extpgmscl} ..... 6
            mixdef ..... 2
            if(mixdef == 0x1) /* mixing option 2 */ {mixdata} ..... 5
            else if(mixdef == 0x2) /* mixing option 3 */ {mixdata} ..... 12
            else if(mixdef == 0x3) /* mixing option 4 */
            {
                mixdeflen ..... 5
                mixdata ..... 8*(mixdeflen+2)
            }
            if(acmod < 0x2) /* if mono or dual mono source */
            {
                paninfoe ..... 1
                if(paninfoe) {paninfo} ..... 12
                if(acmod == 0x0) /* if 1+1 mode (dual mono, so some items need a second value) */
                {
                    paninfo2e ..... 1
                    if(paninfo2e) {paninfo2} ..... 12
                }
            }
        }
    }
    infomdate ..... 1
    if(infomdate) /* Informational metadata */
    {
        bsmod ..... 3
        copyrightb ..... 1
        origbs ..... 1
        if(acmod == 0x2) /* if in 2/0 mode */
        {
            dsurmod ..... 2
            dheadphonmod ..... 2
        }
        if(acmod >= 0x6) /* if both surround channels exist */ {dsurexmod} ..... 2
        audprodie ..... 1
        if(audprodie)
        {
            mixlevel ..... 5
            roomtyp ..... 2
            adconvtyp ..... 1
```

```
        }
        if(acmod == 0x0) /* if 1+1 mode (dual mono, so some items need a second value) */
        {
            audprodi2e ..... 1
            if(audprodi2e)
            {
                mixlevel2 ..... 5
                roomtyp2 ..... 2
                adconvtyp2 ..... 1
            }
        }
        if(fscod < 0x3) /* if not half sample rate */  {sourcefscod} ..... 1
    }
    if( (strmtyp == 0x0) && (numblkscod != 0x3) ) {convsync} ..... 1
    if(strmtyp == 0x2) /* if bit stream converted from AC-3 */
    {
        if(numblkscod == 0x3) /* 6 blocks per frame */ {blkid = 1}
        else {blkid} ..... 1
        if(blkid) {frmsizecod} ..... 6
    }
    addbsie ..... 1
    if(addbsie)
    {
        addbsil ..... 6
        addbsi ..... (addbsil+1)  8
    }
} /* end of bsi */
```

### 2.2.3 audfrm – Audio Frame

| Syntax | Word Size |
|---|---|

```
audfrm()
{
/* These fields for audio frame exist flags and strategy data */
    expstre ..... 1
    if(numblkscod == 0x3) /* six blocks per frame */ {ahte} ..... 1
    else {ahte = 0}
    snroffststr ..... 2
    transproce ..... 1
    blkswe ..... 1
    dithflage ..... 1
    bamode ..... 1
    frmfgaincode ..... 1
    dbaflde ..... 1
    skipflde ..... 1

/* These fields for coupling data */
    if(acmod > 0x1)
    {
        cplstre[0] = 1
        cplinu[0] ..... 1
        for(blk = 1; blk < number_of_blocks_per_sync_frame; blk++)
        {
            cplstre[blk] ..... 1
```

```
            if(cplstre[blk] == 1) {cplinu[blk]} ..... 1
            else {cplinu[blk] = cplinu[blk-1]}
        }
    }
    else
    {
        for(blk = 0; blk < number_of_blocks_per_sync_frame; blk++) {cplinu[blk] = 0}
    }

/* These fields for exponent strategy data */
    if(expstre)
    {
        for(blk = 0; blk < number_of_blocks_per_sync_frame; blk++)
        {
            if(cplinu[blk] == 1) {cplexpstr[blk]} ..... 2
            for(ch = 0; ch < nfchans; ch++) {chexpstr[blk][ch]} ..... 2
        }
    }
    else
    {
        ncplblks = 0
        for(blk = 0; blk < number_of_blocks_per_sync_frame; blk++) {ncplblks += cplinu[blk]}
        if( (acmod > 0x1) && (ncplblks > 0) ) {frmcplexpstr} ..... 5
        for(ch = 0; ch < nfchans; ch++) {frmchexpstr[ch]} ..... 5
        /* cplexpstr[blk] and chexpstr[blk][ch] derived from table lookups – see Table 2.8 */
    }
    if(lfeon)
    {
        for(blk = 0; blk < number_of_blocks_per_sync_frame; blk++) {lfeexpstr[blk]} ..... 1
    }

/* These fields for AHT data */
    if(ahte)
    {
    /* coupling can use AHT only when coupling in use for all blocks */
        /* ncplregs derived from cplstre and cplexpstr – see section 3.3.2 */
        if( (ncplblks == 6) && (ncplregs ==1) ) {cplahtinu}..... 1
        else {cplahtinu = 0}
        for(ch = 0; ch < nfchans; ch++)
        {
            /* nchregs derived from chexpstr – see section 3.3.2 */
            if(nchregs[ch] == 1) {chahtinu[ch]} ..... 1
            else {chahtinu[ch] == 0}
        }
        if(lfeon)
        {
            /* nlferegs derived from lfeexpstr – see section 3.3.2 */
            if(nlferegs == 1) {lfeahtinu}..... 1
            else {lfeahtinu == 0}
        }
    }

/* These fields for audio frame SNR offset data */
    if(snroffststr == 0x0)
```

```
    {
        frmcsnroffst ..... 6
        frmfsnroffst ..... 4
    }

/* These fields for audio frame transient pre-noise processing data */
    if(transproce)
    {
        for(ch = 0; ch < nfchans; ch++)
        {
            chintransproc[ch] ..... 1
            if(chintransproc[ch])
            {
                transprocloc[ch] ..... 10
                transproclen[ch] ..... 8
            }
        }
    }

/* These fields for block start information */
    if (numblkscod != 0x0) {blkstrtinfoe} ..... 1
    else {blkstrtinfoe = 0}
    if(blkstrtinfoe)
    {
        /* nblkstrtbits determined from frmsiz (see section 2.3.2.22) */
        blkstrtinfo ..... nblkstrtbits
    }

/* These fields for syntax state initialization */
    firstspxbndstrc = 1
    firstcplbndstrc = 1
    for(ch = 0; ch < nfchans; ch++)
    {
        firstspxcos[ch] = 1
        firstcplcos[ch] = 1
    }
    firstcplleak = 1
} /* end of audfrm */
```

### 2.2.4 audblk - Audio Block

| Syntax | Word Size |
|---|---|

```
audblk()
{
/* These fields for block switch and dither flags */
    if(blkswe)
    {
        for(ch = 0; ch < nfchans; ch++) {blksw[ch]} ..... 1
    }
    else
    {
        for(ch = 0; ch < nfchans; ch++) {blksw[ch] = 0}
    }
    if(dithflage)
    {
```

```
        for(ch = 0; ch < nfchans; ch++) {dithflag[ch]}.....1
    }
    else
    {
        for(ch = 0; ch < nfchans; ch++) {dithflag[ch] = 1} /* dither on */
    }

/* These fields for dynamic range control */
    dynrnge.....1
    if(dynrnge) {dynrng}.....8
    if(acmod == 0x0) /* if 1+1 mode */
    {
        dynrng2e.....1
        if(dynrng2e) {dynrng2}.....8
    }

/* These fields for spectral extension strategy information */
    if(blk == 0) {spxstre = 1}
    else {spxstre}.....1
    if(spxstre)
    {
        spxinu.....1
        if(spxinu)
        {
            if(acmod == 0x1)
            {
                chinspx[0] = 1
            }
            else
            {
                for(ch = 0; ch < nfchans; ch++) {chinspx[ch]}.....1
            }
            spxstrtf.....2
            spxbegf.....3
            spxendf.....3
            if(spxbegf < 6)  {spxbegf += 2}
            else  {spxbegf = spxbegf * 2 – 3}
            if(spxendf < 3)  {spxendf += 5}
            else  {spxendf = spxendf * 2 + 3}
            spxbndstrce.....1
            if(spxbndstrce)
            {
                for(bnd = spxbegf+1; bnd < spxendf; bnd++) {spxbndstrc[bnd]}.....1
            }
            else
            {
                if(firstspxbndstrc) {spxbndstrc[] = defspxbndstrc[]}
            }
            firstspxbndstrc = 0
        }
        else /* !spxinu */
        {
            for(ch = 0; ch < nfchans; ch++)
            {
```

```
                    chinspx[ch] = 0
                    firstspxcos[ch] = 1
                }
            }
        }

/* These fields for spectral extension coordinates */
        if(spxinu)
        {
            for(ch = 0; ch < nfchans; ch++)
            {
                if(chinspx[ch])
                {
                    if(firstspxcos[ch])
                    {
                        spxcoe[ch] = 1
                        firstspxcos[ch] = 0
                    }
                    else /* !firstspxcos[ch] */ {spxcoe[ch]} ..... 1
                    if(spxcoe[ch])
                    {
                        spxblnd[ch] ..... 5
                        mstrspxco[ch] ..... 2
                        /* nspxbnds determined from spxbegf, spxendf, and spxbndstrc[ ] */
                        for(bnd = 0; bnd < nspxbnds; bnd++)
                        {
                            spxcoexp[ch][bnd] ..... 4
                            spxcomant[ch][bnd] ..... 2
                        }
                    }
                }
                else /* !chinspx[ch] */
                {
                    firstspxcos[ch] = 1
                }
            }
        }

/* These fields for coupling strategy and enhanced coupling strategy information */
        if( (cplstre[blk]) && (cplinu[blk]) )
        {
            ecplinu ..... 1
            if (acmod == 0x2)
            {
                chincpl[0] = 1
                chincpl[1] = 1
            }
            else
            {
                for(ch = 0; ch < nfchans; ch++) {chincpl[ch]} ..... 1
            }
            if (ecplinu == 0) /* standard coupling in use */
            {
                if(acmod == 0x2) {phsflginu} /* if in 2/0 mode */ ..... 1
```

```
            cplbegf ..... 4
            if (spxinu == 0) /* if SPX not in use */
            {
                cplendf ..... 4
                cplendf = cplendf + 3
            }
            else /* SPX in use */
            {
                cplendf = spxbegf - 1
            }
            cplbndstrce ..... 1
            if(cplbndstrce)
            {
                for(bnd = cplbegf+1; bnd < cplendf; bnd++) {cplbndstrc[bnd]} ..... 1
            }
            else
            {
                if(firstcplbndstrc) {cplbndstrc[] = defcplbndstrc[]}
            }
            firstcplbndstrc = 0
        }
        else /* enhanced coupling in use */
        {
            ecplbegf ..... 4
            if(ecplbegf < 3) {ecpl_start_subbnd = ecplbegf * 2}
            else if(ecplbegf < 13) {ecpl_start_subbnd = ecplbegf + 2}
            else {ecpl_start_subbnd = ecplbegf * 2 - 10}
            if (spxinu == 0) /* if SPX not in use */
            {
                ecplendf ..... 4
                ecpl_end_subbnd = ecplendf + 7
            }
            else /* SPX in use */
            {
                if(spxbegf < 6) {ecpl_end_subbnd = spxbegf + 5}
                else {ecpl_end_subbnd = spxbegf * 2}
            }
            ecplbndstrce ..... 1
            if (ecplbndstrce)
            {
                for(sbnd = max(9, ecpl_start_subbnd+1); sbnd < ecpl_end_subbnd; sbnd++)
                {
                    ecplbndstrc[sbnd] ..... 1
                }
            }
            else
            {
                ecplbndstrc[] = defcplbndstrc[]
            }
        }
    }
    else /*(!cplstre[blk]) || (!cplinu[blk]) */
    {
        for(ch = 0; ch < nfchans; ch++)
```

```
        {
            chincpl[ch] = 0
            firstcplcos[ch] = 1
        }
        firstcplleak = 1
        phsflginu = 0
        ecplinu = 0;
    }

/* These fields for coupling coordinates */
    if(cplinu[blk])
    {
        if(ecplinu == 1) /* enhanced coupling in use */
        {
            firstchincpl = -1
            ecplangleintrp ..... 1
        }
        for(ch = 0; ch < nfchans; ch++)
        {
            if(chincpl[ch])
            {
                if(ecplinu == 0) /* standard coupling in use */
                {
                    if (firstcplcos[ch])
                    {
                        cplcoe[ch] = 1
                        firstcplcos[ch] = 0
                    }
                    else /* !firstcplcos[ch] */ {cplcoe[ch]} ..... 1
                    if(cplcoe[ch])
                    {
                        mstrcplco[ch] ..... 2
                        /* ncplbnd derived from cplbegf, cplendf, and cplbndstrc */
                        for(bnd = 0; bnd < ncplbnd; bnd++)
                        {
                            cplcoexp[ch][bnd] ..... 4
                            cplcomant[ch][bnd] ..... 4
                        }
                    }
                }
                else /* enhanced coupling in use */
                {
                    if(firstchincpl == -1) {firstchincpl = ch}
                    if(firstcplcos[ch])
                    {
                        ecplparam1e[ch] = 1
                        if (ch > firstchincpl) {ecplparam2e[ch] = 1}
                        else {ecplparam2e[ch] = 0}
                        firstcplcos[ch] = 0
                    }
                    else /* !firstcplcos[ch] */
                        {
                            ecplparam1e[ch] ..... 1
                            if(ch > firstchincpl) {ecplparam2e[ch]} ..... 1
```

```
                    else {ecplparam2e[ch] = 0}
                }
                if(ecplparam1e[ch])
                {
                    /* necplbnd derived from ecpl_start_subbnd, ecpl_end_subbnd, and ecplbndstrc */
                    for(bnd = 0; bnd < necplbnd; bnd++) {ecplamp[ch][bnd]} ..... 5
                }
                if(ecplparam2e[ch])
                {
                    /* necplbnd derived from ecpl_start_subbnd, ecpl_end_subbnd, and ecplbndstrc */
                    for(bnd = 0; bnd < necplbnd; bnd++)
                    {
                        ecplangle[ch][bnd] ..... 6
                        ecplchaos[ch][bnd] ..... 3
                    }
            }
            if(ch > firstchincpl) {ecpltrans[ch]} ..... 1
        }
    }
    else /* !chincpl[ch] */
    {
        firstcplcos[ch] = 1
    }
}
if((acmod == 0x2) && phsflginu && (cplcoe[0] || cplcoe[1]))
{
    for(bnd = 0; bnd < ncplbnd; bnd++) {phsflg[bnd]} ..... 1
}
}

/* These fields for rematrixing operation in the 2/0 mode */
if(acmod == 0x2) /* if in 2/0 mode */
{
    if (blk == 0) {rematstr = 1}
    else {rematstr} ..... 1
    if(rematstr)
    {
        /* nrematbnds determined from cplinu, ecplinu, spxinu, cplbegf, ecplbegf and spxbegf */
        for(bnd = 0; bnd < nrematbnds; bnd++) {rematflg[bnd]} ..... 1
    }
}

/* This field for channel bandwidth code */
for(ch = 0; ch < nfchans; ch++)
{
    if(chexpstr[blk][ch] != reuse)
    {
        if((!chincpl[ch]) && (!chinspx[ch])) {chbwcod[ch]} ..... 6
    }
}

/* These fields for exponents */
if(cplinu[blk]) /* exponents for the coupling channel */
{
```

```
        if(cplexpstr[blk] != reuse)
        {
            cplabsexp ..... 4
            /* ncplgrps derived from cplbegf, ecplbegf, cplendf, ecplendf, and cplexpstr */
            for(grp = 0; grp< ncplgrps; grp++) {cplexps[grp]}..... 7
        }
    }
    for(ch = 0; ch < nfchans; ch++) /* exponents for full bandwidth channels */
    {
        if(chexpstr[blk][ch] != reuse)
        {
            exps[ch][0] ..... 4
            /* nchgrps derived from chexpstr[ch], and cplbegf or chbwcod[ch] */
            for(grp = 1; grp <= nchgrps[ch]; grp++) {exps[ch][grp]} ..... 7
            gainrng[ch]..... 2
        }
    }
    if(lfeon) /* exponents for the low frequency effects channel */
    {
        if(lfeexpstr[blk] != reuse)
        {
            lfeexps[0]..... 4
            nlfegrps = 2
            for(grp = 1; grp <= nlfegrps; grp++) {lfeexps[grp]} ..... 7
        }
    }

/* These fields for bit-allocation parametric information */
    if(bamode)
    {
        baie..... 1
        if(baie)
        {
            sdcycod ..... 2
            fdcycod ..... 2
            sgaincod ..... 2
            dbpbcod ..... 2
            floorcod ..... 3
        }
    }
    else
    {
        sdcycod = 0x2
        fdcycod = 0x1
        sgaincod = 0x1
        dbpbcod = 0x2
        floorcod = 0x7
    }
    if( (snroffststr == 0x1) || (snroffststr == 0x2) )
    {
        if(blk == 0) {snroffste = 1}
        else {snroffste} ..... 1
        if(snroffste)
        {
```

```
            csnroffst ..... 6
            if(snroffststr == 0x1)
            {
                blkfsnroffst ..... 4
                if(cplinu[blk]) {cplfsnroffst = blkfsnroffst}
                for(ch=0; ch < nfchans; ch++) {fsnroffst[ch] = blkfsnroffst}
                if(lfeon) {lfefsnroffst = blkfsnroffst}
            }
            else if(snroffststr == 0x2)
            {
                if(cplinu[blk]) {cplfsnroffst} ..... 4
                for(ch = 0; ch < nfchans; ch++) {fsnroffst[ch]} ..... 4
                if(lfeon) {lfefsnroffst} ..... 4
            }
        }
    }
    if(frmfgaincode) {fgaincode} ..... 1
    else {fgaincode = 0}
    if(fgaincode)
    {
        if(cplinu[blk]) {cplfgaincod} ..... 3
        for(ch = 0; ch < nfchans; ch++) {fgaincod[ch]} ..... 3
        if(lfeon) {lfefgaincod} ..... 3
    }
    else
    {
        if(cplinu[blk]) {cplfgaincod = 0x4}
        for(ch= 0; ch < nfchans; ch++) {fgaincod[ch] = 0x4}
        if(lfeon) {lfefgaincod = 0x4}
    }
    convsnroffste ..... 1
    if(convsnroffste) {convsnroffst} ..... 10
    if(cplinu[blk])
    {
        if (firstcplleak)
        {
            cplleake = 1
            firstcplleak = 0
        }
        else /* !firstcplleak */
        {
            cplleake ..... 1
        }
        if(cplleake)
        {
            cplfleak ..... 3
            cplsleak ..... 3
        }
    }

/* These fields for delta bit allocation information */
    if(dbaflde)
    {
        deltbaie ..... 1
```

```
        if(deltbaie)
        {
            if(cplinu[blk]) {cpldeltbae} ..... 2
            for(ch = 0; ch < nfchans; ch++) {deltbae[ch]} ..... 2
            if(cplinu[blk])
            {
                if(cpldeltbae==new info follows)
                {
                    cpldeltnseg ..... 3
                    for(seg = 0; seg <= cpldeltnseg; seg++)
                    {
                        cpldeltoffst[seg] ..... 5
                        cpldeltlen[seg] ..... 4
                        cpldeltba[seg] ..... 3
                    }
                }
            }
            for(ch = 0; ch < nfchans; ch++)
            {
                if(deltbae[ch]==new info follows)
                {
                    deltnseg[ch] ..... 3
                    for(seg = 0; seg <= deltnseg[ch]; seg++)
                    {
                        deltoffst[ch][seg] ..... 5
                        deltlen[ch][seg] ..... 4
                        deltba[ch][seg] ..... 3
                    }
                }
            }
        } /* if(deltbaie) */
    }/* if(dbaflde) */

/* These fields for inclusion of unused dummy data */
    if(skipflde)
    {
        skiple ..... 1
        if(skiple)
        {
            skipl ..... 9
            skipfld ..... skipl 8
        }
    }

/* These fields for quantized mantissa values */
    got_cplchan = 0
    for(ch = 0; ch < nfchans; ch++)
    {
        if(chahtinu[ch] == 0)
        {
            for(bin = 0; bin < nchmant[ch]; bin++) {chmant[ch][bin]} ..... (0-16)
        }
        else if(chahtinu[ch] == 1)
        {
```

```
            chgaqmod[ch] ..... 2
            if( (chgaqmod[ch] > 0x0) && (chgaqmod[ch] < 0x3) )
            {
                for(n = 0; n < chgaqsections[ch]; n++) {chgaqgain[ch][n]} ..... 1
            }
            else if(chgaqmod[ch] == 0x3)
            {
                for(n = 0; n < chgaqsections[ch]; n++) {chgaqgain[ch][n]} ..... 5
            }
            for(bin = 0; bin < nchmant[ch]; bin++)
            {
                if(chgaqbin[ch][bin])
                {
                    for(n = 0; n < 6; n++) {pre_chmant[n][ch][bin]} ..... (0-16)
                }
                else {pre_chmant[0][ch][bin]} ..... (0-9)
            }
            chahtinu[ch] = -1 /* AHT info for this frame has been read – do not read again */
        }
        if(cplinu[blk] && chincpl[ch] && !got_cplchan)
        {
            if(cplahtinu == 0)
            {
                for(bin = 0; bin < ncplmant; bin++) {cplmant[bin]} ..... (0-16)
                got_cplchan = 1
            }
            else if(cplahtinu == 1)
            {
                cplgaqmod ..... 2
                if( (cplgaqmod > 0x0) && (cplgaqmod < 0x3) )
                {
                    for(n = 0; n < cplgaqsections; n++) {cplgaqgain[n]} ..... 1
                }
                else if(cplgaqmod == 0x3)
                {
                    for(n = 0; n < cplgaqsections; n++) {cplgaqgain[n]} ..... 5
                }
                for(bin = 0; bin < ncplmant; bin++)
                {
                    if(cplgaqbin[bin])
                    {
                        for(n = 0; n < 6; n++) {pre_cplmant[n][bin]} ..... (0-16)
                    }
                    else {pre_cplmant[0][bin]} ..... (0-9)
                }
                got_cplchan = 1
                cplahtinu = -1 /* AHT info for this frame has been read – do not read again */
            }
            else {got_cplchan = 1}
        }
    }
    if(lfeon) /* mantissas of low frequency effects channel */
    {
        if(lfeahtinu == 0)
```

```
        {
            for(bin = 0; bin < nlfemant; bin++) {lfemant[bin]} ..... (0-16)
        }
        else if(lfeahtinu == 1)
        {
            lfegaqmod ..... 2
            if( (lfegaqmod > 0x0) && (lfegaqmod < 0x3) )
            {
                for(n = 0; n < lfegaqsections; n++) {lfegaqgain[n]}..... 1
            }
            else if(lfegaqmod == 0x3)
            {
                for(n = 0; n < lfegaqsections; n++) {lfegaqgain[n]}..... 5
            }
            for(bin = 0; bin < nlfemant; bin++)
            {
                if(lfegaqbin[bin])
                {
                    for(n = 0; n < 6; n++) {pre_lfemant[n][bin]} ..... (0-16)
                }
                else {pre_lfemant[0][bin]}..... (0-9)
            }
            lfeahtinu = -1 /* AHT info for this frame has been read – do not read again */
        }
    }
} /* end of audblk */
```

### 2.2.5 auxdata - Auxiliary Data

| Syntax | Word Size |
|---|---|
| auxdata() | |
| { | |
| **auxbits** | nauxbits |
| if(auxdatae) | |
| { | |
| **auxdatal** | 14 |
| } | |
| **auxdatae** | 1 |
| } /* end of auxdata */ | |

### 2.2.6 errorcheck - Error Detection Code

| Syntax | Word Size |
|---|---|
| errorcheck() | |
| { | |
| **encinfo** | 1 |
| **crc2** | 16 |
| } /* end of errorcheck */ | |

## 2.3 Description of Enhanced AC-3 bit stream elements

Unless otherwise indicated, all bit stream elements retain the same meaning and purpose as described in ATSC Document A/52, including enhancements described in the Alternate Bit stream Syntax annex.

### 2.3.1 bsi - Bit Stream Information

#### 2.3.1.1 strmtyp – Stream Type – 2 bits

This 2-bit code, as shown in **Table 2.1**, indicates the stream type.

**Table 2.1** Stream Type

| strmtyp | Indication |
|---|---|
| '00' | Type 0 |
| '01' | Type 1 |
| '10' | Type 2 |
| '11' | Type 3 |

The stream types are defined as follows:

**Type 0**: These frames comprise an independent stream or substream. The program may be decoded independently of any other substreams that might exist in the bit stream.

**Type 1**: These frames comprise a dependent substream. The program must be decoded in conjunction with the independent substream with which it is associated.

**Type 2**: These frames comprise an independent stream or substream that was previously coded in AC-3. Type 2 streams must be independently decodable, and may not have any dependent streams associated with them.

**Type 3**: Reserved.

#### 2.3.1.2 substreamid – Substream Identification – 3 bits

This field indicates the substream identification parameter. The substream identification parameter can be used, in conjunction with additional bit stream metadata, to enable carriage of a single program of more than 5.1 channels, multiple programs of up to 5.1 channels, or a mixture of programs with up to 5.1 channels and programs with greater than 5.1 channels.

All Enhanced AC-3 bit streams must contain an independent substream assigned substream ID 0. The independent substream assigned substream ID 0 must be the first substream present in the bit stream.

Enhanced AC-3 bit streams may also contain up to 7 additional independent substreams assigned substream ID's 1 – 7. Independent substream ID's must be assigned sequentially in the order the independent substreams are present in the bit stream.

Each independent substream may have up to 8 dependent substreams associated with it. Dependent substreams must immediately follow the independent substream with which they are associated. Dependent substreams are assigned substream ID's 0 – 7, which must be assigned sequentially according to the order the dependent substreams are present in the bit stream.

For more information about usage of the substreamid parameter, please refer to Section 3.6.

#### 2.3.1.3 frmsiz – Frame Size – 11 bits

This field indicates a value one less than the overall size of the coded frame in 16-bit words. That is, this field may assume a value ranging from 0 to 2047, and these values correspond to frame sizes ranging from 1 to 2048. Note that some values at the lower end of this range may not be valid, as they may not represent enough words to convey a complete frame. It is the responsibility of the encoder to ensure that this does not occur in practice.

#### 2.3.1.4 fscod – Sample Rate Code – 2 bits

This is a 2-bit code indicating sample rate according to **Table 2.2**. If the fscod2 is indicated, the decoder should interpret the following 2-bits as fscod2.

**Table 2.2** Sample Rate Codes

| fscod | Sampling Rate, kHz |
|---|---|
| '00' | 48 |
| '01' | 44.1 |
| '10' | 32 |
| '11' | fscod2 |

#### 2.3.1.5 numblkscod / fscod2 – Number of Audio Blocks / Sample Rate Code 2 – 2 bits

**numblkscod**: This 2-bit code, as shown in **Table 2.3**, indicates the number of audio blocks per syncframe if the fscod indicates 32, 44.1, or 48 kHz sampling rate:

**Table 2.3** Number of Audio Blocks Per Syncframe

| numblkscod | Indication |
|---|---|
| '00' | 1 block per syncframe |
| '01' | 2 blocks per syncframe |
| '10' | 3 blocks per syncframe |
| '11' | 6 blocks per syncframe |

**fscod2**: If the fscod field indicates fscod2 then this 2-bit code indicates the reduced sample rate, as shown in **Table 2.4**. When using reduced sample rates, numblkscod shall be '11' (6 blocks per syncframe).

**Table 2.4** Reduced Sampling Rates

| fscod2 | Sampling Rate, kHz |
|---|---|
| '00' | 24 |
| '01' | 22.05 |
| '10' | 16 |
| '11' | reserved |

#### 2.3.1.6 bsid – Bit Stream Identification – 5 bits

This bit field has a value of '01100' (=12) in this version of this standard.

#### 2.3.1.7 chanmape – Custom Channel Map Exists – 1 bit

If this bit is a 0, the channel map for a dependent substream is defined by the audio coding mode (acmod) and LFE on (lfeon) parameters. If this bit is a 1, the following 16 bits define the custom channel map for this dependent substream.

Only dependent substreams can have a custom channel map.

#### 2.3.1.8 chanmap – Custom Channel Map – 16 bits

This 16-bit field specifies the custom channel map for a dependent substream. The channel locations supported by the custom channel map are defined in **Table 2.5**. Shaded entries in **Table 2.5** represent channel locations present in the independent substream with which the

dependent substream is associated. Non-shaded entries in **Table 2.5** represent channel locations not present in the independent substream with which the dependent substream is associated.

**Table 2.5** Custom Channel Map Locations

| Bit | Location | Bit | Location |
|---|---|---|---|
| 0 | Left | 8 | TBD |
| 1 | Center | 9 | TBD |
| 2 | Right | 10 | TBD |
| 3 | Left Surround | 11 | TBD |
| 4 | Right Surround | 12 | TBD |
| 5 | TBD | 13 | TBD |
| 6 | TBD | 14 | TBD |
| 7 | TBD | 15 | LFE |

The custom channel map indicates both which coded channels are present in the dependent substream and the order of the coded channels in the dependent substream. For each channel present in the dependent substream, the corresponding location bit in the chanmap is set to 1. The order of the coded channels in the dependent substream is the same as the order of the enabled location bits in the chanmap. For example, if bits 1, 3, and 4 of the chanmap field are set to 1, and the dependent stream is coded with acmod = 3 and lfeon = 0, the first coded channel in the dependent stream is the Left channel, the second coded channel is the Left Surround channel, and the third coded channel is the Right Surround channel. Note that the number of location bits set to 1 in the chanmap field must equal the total number of coded channels present in the dependent substream, as indicated by the acmod and lfeon bit stream parameters.

For more information about usage of the chanmap parameter, please refer to ection 3.6.

#### 2.3.1.9 mixmdate – Mixing Meta-Data Exists – 1 bit

If this bit is a 1, mixing and mapping information follows in the bit stream.

#### 2.3.1.10 pgmscle – Program Scale Factor Exists – 1 bit

If this bit is a 1, the program scale factor word follows in the bit stream. If it is 0, the program scale factor word is defaulted to 0 dB (no scaling).

#### 2.3.1.11 pgmscl – Program Scale Factor – 6 bits

This field specifies a scale factor that should be applied to the program during decoding. Valid values are 0-63, with 0 interpreted as mute, and 1–63 interpreted as –50 dB to +12 dB of scaling in 1 dB steps.

#### 2.3.1.12 pgmscl2e – Program Scale Factor #2 Exists – 1 bit

If this bit is a 1, the program scale factor #2 word follows in the bit stream. If it is 0, the program scale factor #2 word is defaulted to 0 dB (no scaling).

#### 2.3.1.13 pgmscl2 – Program Scale Factor #2 – 6 bits

This field has the same meaning as pgmscl, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono 1+1 mode).

#### 2.3.1.14 extpgmscle – External Program Scale Factor Exists – 1 bit

If this bit is a 1, the external program scale factor word follows in the bit stream. If it is 0, the external program scale factor word is defaulted to 0 dB (no scaling).

#### 2.3.1.15 extpgmscl – External Program Scale Factor – 6 bits

In some applications, two bit streams may be decoded and mixed together. This field specifies a scale factor that should be applied to the external program (i.e., the program that is not carried in this bit stream) during mixing. This field uses the same scale as pgmscl.

#### 2.3.1.16 mixdef – Mix Control Type – 2 bits

This 2-bit code, as shown in **Table 2.6**, indicates the mode and parameter field lengths for program mixing.

**Table 2.6** Mix Control

| mixdef | Indication |
|---|---|
| '00' | mixing option 1, no additional bits |
| '01' | mixing option 2, 5 bits reserved |
| '10' | mixing option 3, 12 bits reserved |
| '11' | mixing option 4, 16-264 bits reserved by mixdeflen |

#### 2.3.1.17 mixdeflen – Length of Mixing Parameter Data Field – 5 bits

This defines the mixing data field size for the most flexible mode. The length is given in bytes: mixdeflen = {0,1 2,3 … 31) represents mixdata lengths = {2,3,4,5 … 33} bytes.

#### 2.3.1.18 mixdata – Mixing Parameter Data – (5-264) bits

This data field contains control parameters for mixing the program and external program streams during decoding.

#### 2.3.1.19 paninfoe – Pan Information Exists – 1 bit

If this bit is a 1, panning information follows in the bit stream. If it is 0, the pan position word is defaulted to "center".

#### 2.3.1.20 paninfo – Pan Information – 12 bits

This 12-bit word defines how a single channel is reproduced in a two dimensional sound field.

#### 2.3.1.21 paninfo2e – Pan Information Exists - 1 bit

If this bit is a 1, panning information #2 follows in the bit stream. If it is 0, the pan position word is defaulted to "center".

#### 2.3.1.22 paninfo2 – Pan Information – 12 bits

This field has the same meaning as paninfo, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono 1+1 mode).

#### 2.3.1.23 infomdate – Informational Meta-Data Exists – 1 bit

If this bit is a 1, informational meta-data follows in the bit stream.

#### 2.3.1.24 sourcefscod – Source Sample Rate Code – 1 bit

If the sourcefscod bit is a 1, the source material was sampled at twice the rate indicated by fscod.

#### 2.3.1.25 convsync – Converter Synchronization Flag – 1 bit

This bit is used for synchronization by a device that converts an Enhanced AC-3 bit stream to an A/52 compliant bit stream.

#### 2.3.1.26 blkid – Block Identification – 1 bit

If strmtyp indicates a Type 2 bit stream, this bit is set to 1 to indicate that the first block in this Enhanced AC-3 frame was the first block in the original standard AC-3 frame.

### 2.3.2 audfrm – Audio Frame

#### 2.3.2.1 expstre – Exponent Strategy Syntax Enabled – 1 bit

If this bit is a 1, full exponent strategy syntax exists in each audio block. If this bit is a 0, then the exponent strategy is specified by the frame-based exponent strategy defined in Sections 2.3.2.11 and 2.3.2.12.

#### 2.3.2.2 ahte – Adaptive Hybrid Transform Enabled – 1 bit

If this bit is a 1, an Adaptive Hybrid Transform is used to code at least one of the independent channels, the coupling channel, or the LFE channel in the current frame. If this bit is a 0, the entire frame is coded using the standard bit allocation and quantization model as described in ATSC document A/52.

#### 2.3.2.3 snroffststr – SNR Offset Strategy – 2 bits

This field indicates how SNR offsets are transmitted

**Table 2.7** SNR Offset Strategy

| snroffststr | Indication |
|---|---|
| '00' | SNR offset strategy 1 |
| '01' | SNR offset strategy 2 |
| '10' | SNR offset strategy 3 |
| '11' | Reserved |

**SNR Offset Strategy 1**: When SNR Offset Strategy 1 is used, one coarse SNR offset value and one fine SNR offset value are transmitted in the bit stream. These SNR offset values are used for every channel of every block in the frame, including the coupling and LFE channels.

**SNR Offset Strategy 2**: When SNR Offset Strategy 2 is used, one coarse SNR offset value and one fine SNR offset value are transmitted in the bit stream as often as once per block. The fine SNR offset value is used for every channel in the block, including the coupling and LFE channels. For blocks in which coarse and fine SNR offset values are not transmitted in the bit stream, the decoder must reuse the coarse and fine SNR offset values from the previous block. One coarse and one fine SNR offset value must be transmitted in block 0.

**SNR Offset Strategy 3**: When SNR Offset Strategy 3 is used, coarse and fine SNR offset values are transmitted in the bit stream as often as once per block. Separate fine SNR offset values are transmitted for each channel, including the coupling and LFE channels. For blocks in which coarse and fine SNR offset values are not transmitted in the bit stream, the decoder

must reuse the coarse and fine SNR offset values from the previous block. Coarse and fine SNR offset values must be transmitted in block 0.

#### 2.3.2.4 transproce – Transient Pre-Noise Processing Enabled – 1 bit

If this bit is a 1, at least one channel in the current frame contains transient pre-noise processing data. If it is 0, transient pre-noise processing is not being utilized in this frame.

#### 2.3.2.5 blkswe – Block Switch Syntax Enabled – 1 bit

If this bit is a 1, full block switch syntax exists in each audio block.

#### 2.3.2.6 dithflage – Dither Flag Syntax Enabled – 1 bit

If this bit is a 1, full dither flag syntax exists in each audio block.

#### 2.3.2.7 bamode – Bit Allocation Model Syntax Enabled – 1 bit

If this bit is a 1, full bit allocation syntax exists in each audio block.

#### 2.3.2.8 frmfgaincode – Fast Gain Codes Enabled – 1 bit

If this bit is a 1, fast gain codes are transmitted in the bit stream as often as once per audio block. If this bit is a 0, no fast gain codes are transmitted in the bit stream, and default fast gain code values are used for every channel of every block in the frame.

#### 2.3.2.9 dbaflde – Delta Bit Allocation Syntax Enabled – 1 bit

If this bit is a 1, full delta bit allocation syntax exists in each audio block.

#### 2.3.2.10 skipflde – Skip Field Syntax Enabled – 1 bit

If this bit is a 1, full skip field syntax exists in each audio block.

#### 2.3.2.11 frmcplexpstr – Frame Based Coupling Exponent Strategy – 5 bits

This 5-bit code specifies the coupling channel exponent strategy for all audio blocks, as defined in **Table 2.8**. The number of blocks per frame is required to be six. Note that exponent strategies D15, D25, and D45 are as defined in ATSC Document A/52, while 'R' indicates that exponents from the previous block are reused.

#### 2.3.2.12 frmchexpstr[ch] – Frame Based Channel Exponent Strategy – 5 bits

This 5-bit code specifies the channel exponent strategy for all audio blocks, as defined in **Table 2.8**. The number of blocks per frame is required to be six. Note that exponent strategies D15, D25, and D45 are as defined in ATSC Document A/52, while 'R' indicates that exponents from the previous block are reused.

**Table 2.8** Frame Exponent Strategy Combinations

| frmcplexpstr | Audio Block Number | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| 0 | D15 | R | R | R | R | R |
| 1 | D15 | R | R | R | R | D45 |
| 2 | D15 | R | R | R | D25 | R |
| 3 | D15 | R | R | R | D45 | D45 |
| 4 | D25 | R | R | D25 | R | R |
| 5 | D25 | R | R | D25 | R | D45 |
| 6 | D25 | R | R | D45 | D25 | R |
| 7 | D25 | R | R | D45 | D45 | D45 |
| 8 | D25 | R | D15 | R | R | R |
| 9 | D25 | R | D25 | R | R | D45 |
| 10 | D25 | R | D25 | R | D25 | R |
| 11 | D25 | R | D25 | R | D45 | D45 |
| 12 | D25 | R | D45 | D25 | R | R |
| 13 | D25 | R | D45 | D25 | R | D45 |
| 14 | D25 | R | D45 | D45 | D25 | R |
| 15 | D25 | R | D45 | D45 | D45 | D45 |
| 16 | D45 | D15 | R | R | R | R |
| 17 | D45 | D15 | R | R | R | D45 |
| 18 | D45 | D25 | R | R | D25 | R |
| 19 | D45 | D25 | R | R | D45 | D45 |
| 20 | D45 | D25 | R | D25 | R | R |
| 21 | D45 | D25 | R | D25 | R | D45 |
| 22 | D45 | D25 | R | D45 | D25 | R |
| 23 | D45 | D25 | R | D45 | D45 | D45 |
| 24 | D45 | D45 | D15 | R | R | R |
| 25 | D45 | D45 | D25 | R | R | D45 |
| 26 | D45 | D45 | D25 | R | D25 | R |
| 27 | D45 | D45 | D25 | R | D45 | D45 |
| 28 | D45 | D45 | D45 | D25 | R | R |
| 29 | D45 | D45 | D45 | D25 | R | D45 |
| 30 | D45 | D45 | D45 | D45 | D25 | R |
| 31 | D45 | D45 | D45 | D45 | D45 | D45 |

#### 2.3.2.13 cplahtinu – Coupling Channel AHT in Use – 1bit

If this bit is a 1, the coupling channel is coded using an Adaptive Hybrid Transform. If this bit is a 0, conventional coupling channel coding is used for that region.

#### 2.3.2.14 chahtinu[ch] – Channel AHT in Use – 1 bit

If this bit is a 1, channel ch is coded using an Adaptive Hybrid Transform. If this bit is a 0, conventional channel coding is used for that region.

#### 2.3.2.15 lfeahtinu – LFE Channel AHT in Use – 1 bit

If this bit is a 1, the LFE channel is coded using an Adaptive Hybrid Transform. If this bit is a 0, conventional LEF channel coding is used for that region.

#### 2.3.2.16 frmcsnroffst – Frame Coarse SNR Offset – 6 bits

This field contains the frame coarse SNR offset value. This coarse SNR offset value is used for every block in the frame.

#### 2.3.2.17 frmfsnroffst – Frame Fine SNR Offset – 4 bits

This field contains the frame fine SNR offset value. This fine SNR offset value is used for every channel of every block in the frame, including the coupling and LFE channels.

#### 2.3.2.18 chintransproc[ch] – Channel in Transient Pre-Noise Processing – 1 bit

Transient pre-noise processing exist bit for each full bandwidth channel. If set to 1, then the corresponding channel has associated transient pre-noise processing data.

#### 2.3.2.19 transprocloc[ch] – Transient Location Relative to Start of Frame – 10 bits

This field provides the location of the transient relative to the start of the current frame. The transient location (in samples) is calculated by multiplying this value by 4. It is possible for the transient to be located in a later audio frame and therefore this number can exceed the number of PCM samples contained within the current frame.

#### 2.3.2.20 transproclen[ch] – Transient Processing Length – 8 bits

This field provides the transient pre-noise processing length in samples, relative to the location of the transient provided by the value of transprocloc[ch].

#### 2.3.2.21 blkstrtinfoe – Block Start Information Exists – 1 bit

If this bit is a 1, block start information follows in the bit stream. If this bit is a 0, no block start information follows in the bit stream.

#### 2.3.2.22 blkstrtinfo – Block Start Information – nblkstrtbits

This field contains the block start information. The number of bits of block start information is given by the formula:

$$nblkstrtbits = (numblks - 1) * (4 + ceiling\ (\log_2 (words\_per\_frame)))$$

where:

numblks is derived from the numblkscod in **Table 2.3**

ceiling(n) is a function which rounds the fractional number n up to the next higher integer.

For example,

$$ceiling(2.1) = 3$$

$\log_2(n)$ is the base 2 logarithm of n

$$words\_per\_frame = \mathbf{frmsiz} + 1$$

#### 2.3.2.23 firstspxbndstrc – First Spectral Extension Band Structure State

When spxbndstrce is set to 0 and firstspxbndstrc is set to 1, the default spectral extension band structure is used. When spxbndstrce is set to 0 and firstspxbndstrc is set to 0, the previous spectral extension band structure is used.

#### 2.3.2.24 firstcplbndstrc – First Coupling Band Structure State

When cplbndstrce is set to 0 and firstcplbndstrc is set to 1, the default coupling band structure is used. When cplbndstrce is set to 0 and firstcplbndstrc is set to 0, the previous coupling band structure is used.

#### 2.3.2.25 firstspxcos[ch] – First Spectral Extension Coordinates States

The firstspxcos[ch] state determines when new spectral extension coordinates can be assumed to exist in the bit stream. If firstspxcos[ch] is set to 1, the spxcoe[ch] bit is assumed to be 1 for the current block and is not transmitted in the bit stream.

#### 2.3.2.26 firstcplcos[ch] – First Coupling Coordinates States

The firstcplcos[ch] state determines when new coupling coordinates can be assumed to exist in the bit stream. If firstcplcos[ch] is set to 1, the cplcoe[ch] bit is assumed to be 1 for the current block and is not transmitted in the bit stream.

#### 2.3.2.27 firstcplleak – First Coupling Leak State

The firstcplleak state determines when new coupling leak values can be assumed to exist in the bit stream. If firstcplleak is set to 1, the cplleake bit is assumed to be 1 for the current block and is not transmitted in the bit stream.

### 2.3.3 audblk – Audio Block

#### 2.3.3.1 spxstre – Spectral Extension Strategy Exists – 1 bit

If this bit is a 1, spectral extension information follows in the bit stream. If it is 0, new spectral extension information is not present, and spectral extension parameters previously sent are reused.

#### 2.3.3.2 spxinu – Spectral Extension in Use – 1 bit

If this bit is a 1, then the spectral extension technique is used in this block. If this bit is a 0, then the spectral extension technique is not used in this block.

#### 2.3.3.3 chinspx[ch] – Channel Using Spectral Extension – 1 bit

If this bit is a 1, then the channel indicated by the index [ch] is utilizing spectral extension. If the bit is a 0, then this channel is not utilizing spectral extension.

#### 2.3.3.4 spxstrtf – Spectral Extension Start Copy Frequency Code – 2 bits

This 2-bit code is used to derive the number of the lowest frequency sub-band of the spectral extension copy region. See **Table 3.13** for the definition of the spectral extension sub-bands.

#### 2.3.3.5 spxbegf – Spectral Extension Begin Frequency Code – 3 bits

This 3-bit code is used to derive the number of the lowest frequency sub-band of the spectral extension region.

#### 2.3.3.6 spxendf – Spectral Extension End Frequency Code – 3 bits

This 3-bit code is used to derive a number one greater than the highest frequency sub-band of the spectral extension region.

#### 2.3.3.7 spxbndstrce – Spectral Extension Band Structure Exist – 1 bit

If this parameter is one, the spectral extension band structure follows. If it is zero in the first block, a default spectral extension band structure is used. If it is zero in any other block, the band structure from the previous block is reused. The default banding structure defspxbndstrc[] is shown in **Table 2.9**. Note that the first sub-band is not transmitted.

**Table 2.9** Default Spectral Extension Banding Structure

| spx sub-band # | defspxbndstrc[] |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 1 |
| 9 | 0 |
| 10 | 1 |
| 11 | 0 |
| 12 | 1 |
| 13 | 0 |
| 14 | 1 |
| 15 | 0 |
| 16 | 1 |

#### 2.3.3.8 spxbndstrc[bnd] – Spectral Extension Band Structure – 1 – 14 bits

This data structure determines the grouping of subbands in spectral extension, and operates in the same fashion as the coupling band structure. For each subband:

- A zero represents the beginning of a new band
- A one indicates that the subband should be combined into the previous band.

Note that it is assumed that the first band begins at the first subband. Therefore, the first band is assumed to be zero and not sent. The first band in the structure corresponds to the second subband.

#### 2.3.3.9 spxcoe[ch] – Spectral Extension Coordinates Exist – 1 bit

If this parameter is one, spectral extension coordinate information follows. If it is zero, the spectral extension coordinates from the previous block are used.

#### 2.3.3.10 spxblnd[ch] – Spectral Extension Blend – 5 bits

This per channel parameter determines the per channel noise blending factor (translated signal mixed with random noise) for the spectral extension process.

#### 2.3.3.11 mstrspxco[ch] – Master Spectral Extension Coordinate – 2 bits

This per channel parameter establishes a per channel gain factor (increasing the dynamic range) for the spectral extension coordinates as shown in Table 5.9 from the mstrcplco[ch] element from the A/52 document.

#### 2.3.3.12 spxcoexp[ch][bnd] – Spectral Extension Coordinate Exponent – 4 bits

Each spectral extension coordinate is composed of a 4-bit exponent and a 2-bit mantissa. This element is the value of the spectral extension coordinate exponent for channel [ch] and band [bnd]. The index [ch] only will exist for those channels that are in spectral extension. The index [bnd] will range from zero to nspxbnds.

#### 2.3.3.13 spxcomant[ch][bnd] – Spectral Extension Coordinate Mantissa – 2 bits

This element is the 2-bit spectral extension coordinate mantissa for the channel [ch] and band [bnd].

#### 2.3.3.14 ecplinu – Enhanced Coupling in Use – 1 bit

If this bit is a 1, enhanced coupling is used for the current block. If this bit is a 0, standard coupling is used for the current block.

#### 2.3.3.15 cplbndstrce – Coupling Band Structure Exist – 1 bit

If this parameter is one, the coupling band structure follows. If it is zero in the first block, a default coupling band structure is used. If it is zero in any other block, the band structure from the previous block is reused. The default coupling banding structure defcplbndstrc[] is shown in **Table 2.10**. Note that the first sub-band is not transmitted.

**Table 2.10** Default Coupling Banding Structure

| couple sub-band # | defcplbndstrc[] |
|---|---|
| 0 | |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 1 |
| 9 | 0 |
| 10 | 1 |
| 11 | 1 |
| 12 | 0 |
| 13 | 1 |
| 14 | 1 |
| 15 | 1 |
| 16 | 1 |
| 17 | 1 |

#### 2.3.3.16 ecplbegf – Enhanced Coupling Begin Frequency Code – 4 bits

This 4-bit code is used to derive the number of the lowest frequency edge of the enhanced coupling channel (or the first active enhanced coupling sub-band) as shown in **Table 3.8**. The index of the first active enhanced coupling sub-band is equal to ecpl_start_subbnd and is calculated as:

```
if (ecplbegf < 3) {ecpl_start_subbnd = ecplbegf * 2}
else if (ecplbegf < 13) {ecpl_start_subbnd = ecplbegf + 2}
else {ecpl_start_subbnd = ecplbegf * 2 - 10}
```

#### 2.3.3.17 ecplendf – Enhanced Coupling End Frequency Code – 4 bits

This 4-bit code is used to derive a number one greater than the highest frequency sub-band of the enhanced coupling region. See **Table 3.8**. The index of one greater than the highest active enhanced coupling sub-band is equal to ecpl_end_subbnd and is calculated as:

```
if (spxinu == 0) {ecpl_end_subbnd = ecplendf + 7}
else if (spxbegf < 6) {ecpl_end_subbnd = spxbegf + 5}
else {ecpl_end_subbnd = spxbegf * 2}
```

#### 2.3.3.18 ecplbndstrce – Enhanced Coupling Band Structure Exists – 1 bit

If this bit is a 1, the enhanced coupling band structure follows. If this bit is a 0 in block 0, the default enhanced coupling band structure is used. If this bit is 0 in any other block, the band structure from the previous block is reused. The default enhanced coupling banding structure defecplbndstrc[] is shown in **Table 2.11**. Note that sub-bands 0 to 8 are equal to 0 and are not transmitted. See ecplbndstrc below.

**Table 2.11** Default Enhanced Coupling Banding Structure

| Enhanced Coupling Sub-Band # | defecplbndstrc[] |
|---|---|
| 0 to 8 | 0 |
| 9 | 1 |
| 10 | 0 |
| 11 | 1 |
| 12 | 0 |
| 13 | 1 |
| 14 | 0 |
| 15 | 1 |
| 16 | 1 |
| 17 | 1 |
| 18 | 0 |
| 19 | 1 |
| 20 | 1 |
| 21 | 1 |

2.3.3.19 ecplbndstrc[sbnd] – Enhanced Coupling Band Structure – 1 bit

There are 22 enhanced coupling sub-bands defined in **Table 3.7**, each containing either 6 or 12 frequency coefficients. The fixed 12-bin wide enhanced coupling sub-bands 8 and above are converted into enhanced coupling bands, each of which may be wider than (a multiple of) 12 frequency bins. Sub-bands 0 through 7 are never grouped together to form larger enhanced coupling bands, and are thus each considered enhanced coupling bands. Each enhanced coupling band may contain one or more enhanced coupling sub-bands. Enhanced coupling coordinates are transmitted for each enhanced coupling band. Each band's enhanced coupling coordinate must be applied to all the coefficients in the enhanced coupling band.

The enhanced coupling band structure indicates which enhanced coupling sub-bands are combined into wider enhanced coupling bands. When ecplbndstrc[sbnd] is a 0, the sub-band number [sbnd] is not combined into the previous band to form a wider band, but starts a new 12-bin wide enhanced coupling band. When cplbndstrc[sbnd] is a 1, then the sub-band [sbnd] is combined with the previous band, making the previous band 12 bins wider. Each successive value of ecplbndstrc which is a 1 will continue to combine sub-bands into the current band. When another ecplbndstrc value of 0 is received, then a new band will be formed, beginning with the 12 bins of the current sub-band. The set of ecplbndstrc[sbnd] values is typically considered an array.

Each bit in the array corresponds to a specific enhanced coupling sub-band in ascending frequency order. The elements of the array corresponding to the sub-bands up to and including ecpl_start_subbnd or 8 (whichever is greater), are always 0, and are not transmitted. (There is no reason to send an ecplbndstrc bit for these sub-bands, since these bits are always 0.) If there is only one enhanced coupling sub-band above sub-band 7, then no ecplbndstrc bits are sent.

The total number of enhanced coupling bands, necplbnd, may be computed as follows:

```
necplbnd = ecpl_end_subbnd - ecpl_start_subbnd;
necplbnd -= ecplbndstrc[ecpl_start_subbnd] + … + ecplbndstrc[ecpl_end_subbnd -1]
```

A default setting of ecplbndstrc[], when all bands are used in enhanced coupling, is given in **Table 2.11**.

#### 2.3.3.20 ecplangleintrp – Enhanced Coupling Angle Interpolation Flag – 1 bit

If this element is set to 1, then interpolation is used to derive enhanced coupling bin angle values between band angle values according to the pseudo-code specified in Section 3.4.5.3. If this element is set to 0, then interpolation is not used and the each enhanced coupling band value should be applied to all the bin angle values within the band.

#### 2.3.3.21 ecplparam1e[ch] – Enhanced Coupling Parameters 1 Exist – 1 bit

Enhanced coupling parameters are used to derive the enhanced coupling coordinates which indicate, for a given channel and within a given enhanced coupling band, the fraction of the enhanced coupling channel frequency coefficients to use to re-create the individual channel frequency coefficients. Enhanced coupling parameters are conditionally transmitted in the bit stream. If new values are not delivered, the previously sent values remain in effect. See Section 3.4 for further information on enhanced coupling.

Each enhanced coupling coordinate is derived from a 5-bit amplitude, a 6-bit angle, a 3-bit chaos measure and a 1-bit transient present flag. With the exception of the transient present flag, enhanced coupling parameters are signaled by two exist bits.

If ecplparam1e[ch] is 1, the amplitudes for the corresponding channel [ch] exist and follow in the bit stream. If the bit is 0, the previously transmitted amplitudes for this channel are reused. All amplitudes are always transmitted in the first block in which enhanced coupling is enabled.

#### 2.3.3.22 ecplparam2e[ch] – Enhanced Coupling Parameters 2 Exist – 1 bit

If ecplparam2e[ch] is 1, the angle and chaos values for the corresponding channel [ch] exist and follow in the bit stream. If the bit is 0, the previously transmitted angle and chaos values for this channel are reused. The angle and chaos parameters are always transmitted in the first block in which enhanced coupling is enabled.

#### 2.3.3.23 ecplamp[ch][bnd] – Enhanced Coupling Amplitude Scaling – 5 bits

This element is the value of the enhanced coupling amplitude for channel [ch] and band [bnd]. The index [ch] will only exist for those channels in enhanced coupling. The index [bnd] will range from 0 to necplbnds-1. See Section 3.4.4 for more information on how to interpret enhanced coupling parameters.

#### 2.3.3.24 ecplangle[ch][bnd] – Enhanced Coupling Angle – 6 bits

This element is the 6-bit enhanced coupling angle for channel [ch] and band [bnd]. The enhanced coupling angle is assumed to be 0 for the first channel [ch] in enhanced coupling, and is not transmitted in the bit stream.

#### 2.3.3.25 ecplchaos[ch][bnd] – Enhanced Coupling Chaos – 3 bits

This element is the 3-bit enhanced coupling chaos for channel [ch] and band [bnd]. The enhanced coupling chaos is assumed to be 0 for the first channel [ch] in enhanced coupling, and is not transmitted in the bit stream.

#### 2.3.3.26 ecpltrans[ch] – Enhanced Coupling Transient Present – 1 bit

This element is the 1-bit enhanced coupling transient present indication for channel [ch]. The enhanced coupling transient present bit is not transmitted in the bit stream for the first channel [ch] in enhanced coupling.

#### 2.3.3.27 blkfsnroffst – Block Fine SNR Offset – 4 bits

This 4-bit code specifies the fine SNR offset value used by all channels, including the coupling and LFE channels.

#### 2.3.3.28 fgaincode – Fast Gain Codes Exist – 1 bit

If this parameter is set to 1, fast gain codes for each channel are transmitted in the bit stream. If this parameter is set to 0 in block 0, no fast gain codes are transmitted in the bit stream, and default fast gain codes are used. If parameter is set to 0 in any other block, no fast gain codes are transmitted in the bit stream, and fast gain codes from the previous block are re-used.

#### 2.3.3.29 convsnroffste – Converter SNR Offset Exists – 1 bit

If this parameter is one, a SNR offset for the converter follows.

#### 2.3.3.30 convsnroffst – Converter SNR Offset – 10 bits

This 10 bit word is the SNR offset required to convert the current frame to a compliant A/52 frame.

#### 2.3.3.31 chgaqmod[ch] – Channel Gain Adaptive Quantization Mode – 2 bits

This 2-bit code specifies which one of four possible quantization modes is used for mantissas in the given channel. If chgaqmod[ch] is 0, conventional scalar quantization is used for channel ch. Otherwise, gain adaptive quantization is used and chgaqgain[ch][n] words follow in the bit stream.

#### 2.3.3.32 chgaqgain[ch][n] – Channel Gain Adaptive Quantization gain – 1 or 5 bits

This code signals the adaptive quantizer gain value or values associated with one or more exponents. If chgaqmod[ch] is either 1 or 2, chgaqgain[ch][n] is 1 bit in length, signaling two possible gain states. If chgaqmod[ch] is 3, chgaqgain[ch][n] is 5 bits in length, representing a triplet of gains coded compositely. In this case, each gain signals three possible gain states.

#### 2.3.3.33 pre_chmant[n][ch][bin] – Pre Channel Mantissas – 0 to 16 bits

These values represent the channel mantissas coded either with scalar, vector or gain adaptive quantization.

#### 2.3.3.34 cplgaqmod – Coupling Channel Gain Adaptive Quantization Mode – 2 bits

This 2-bit code specifies which one of four possible quantization modes is used for mantissas in the coupling channel. If cplgaqmod is 0, conventional scalar quantization is used. Otherwise, gain adaptive quantization is used and cplgaqgain[n] words follow in the bit stream.

#### 2.3.3.35 cplgaqgain[n] – Coupling Gain Adaptive Quantization Gain – 1 or 5 bits

This code signals the adaptive quantizer gain value or values associated with one or more exponents. If cplgaqmod is either 1 or 2, cplgaqgain[n] is 1 bit in length, signaling two possible gain states. If cplgaqmod is 3, cplgaqgain[n] is 5 bits in length, representing a triplet of gains coded compositely. In this case, each gain signals three possible gain states.

#### 2.3.3.36 pre_cplmant[n][bin] – Pre Coupling Channel Mantissas – 0 to 16 bits

These values represent the coupling channel mantissas coded either with scalar, vector or gain adaptive quantization.

#### 2.3.3.37 lfegaqmod – LFE Channel Gain Adaptive Quantization Mode – 2 bits

This 2-bit code specifies which one of four possible quantization modes is used for mantissas in the LFE channel. If lfegaqmod is 0, conventional scalar quantization is used. Otherwise, gain adaptive quantization is used and lfegaqgain[n] words follow in the bit stream.

#### 2.3.3.38 lfegaqgain[n] – LFE Gain Adaptive Quantization Gain – 1 or 5 bits

This code signals the adaptive quantizer gain value or values associated with one or more exponents. If lfegaqmod is either 1 or 2, lfegaqgain[n] is 1 bit in length, signaling two possible gain states. If lfegaqmod is 3, lfegaqgain[n] is 5 bits in length, representing a triplet of gains coded compositely. In this case, each gain signals three possible gain states.

#### 2.3.3.39 pre_lfemant[n][bin] – Pre LFE Channel Mantissas – 0 to 16 bits

These values represent the LFE channel mantissas coded either with scalar, vector or gain adaptive quantization.

## 3. DECODER PROCESSING

This section specifies how Enhanced AC-3 decoders will process bit streams that use the Enhanced AC-3 bit stream syntax.

### 3.1 Glitch-Free Switching Between Different Stream Types

Enhanced AC-3 decoders should be designed to switch between all supported bit stream types without introducing audible clicks or pops.

### 3.2 Error Detection and Concealment

Enhanced AC-3 decoders are required to implement error detection based on the bit stream CRC word. Enhanced AC-3 bit streams contain only one CRC word, which covers the entire frame. When decoding bit streams that use the Enhanced AC-3 bit stream syntax, Enhanced AC-3 decoders must verify the CRC word prior to decoding any of the blocks in the frame.

If the CRC word for an Enhanced AC-3 bit stream is found to be invalid, all blocks in the frame must be substituted with an appropriate error concealment signal. For most applications, this can be easily accomplished by simply repeating the last known-good block (before the overlap-add window process).

### 3.3 Adaptive Hybrid Transform Processing

#### 3.3.1 Overview

The AHT is composed of two linear transforms connected in cascade. The first transform is identical to that employed in A/52 AC-3 – a windowed Modified Discrete Cosine Transform (MDCT) of length 128 or 256 frequency samples. This feature provides compatibility with A/52 AC-3 without the need to return to the time domain in the decoder. For frames containing audio signals which are not time-varying in nature (stationary), a second transform can optionally be applied by the encoder, and inverted by the decoder. The second transform is composed of a non-windowed, non-overlapped Discrete Cosine Transform (DCT Type II). When the DCT is employed, the effective audio transform length increases from 256 to 1536 audio samples. This results in significantly improved coding gain and perceptual coding performance for stationary signals.

The AHT transform is enabled by setting the ahte bit stream parameter to 1. If ahte is 1, at least one of the independent channels, the coupling channel, or the LFE channel has been coded with AHT. The chahtinu[ch], cplahtinu, and lfeahtinu bit stream parameters are used to indicate which channels are channels coded with AHT.

In order to realize gain made available by the AHT, the A/52 AC-3 scalar quantizers have been augmented with two new coding tools. When AHT is in use, both 6-dimensional vector quantization (VQ) and gain-adaptive quantization (GAQ) are employed. VQ is employed for the largest step sizes (coarsest quantization), and GAQ is employed for the smallest stepsizes (finest quantization). The selection of quantizer step size is performed using the same parametric bit allocation method as A/52 AC-3, except the conventional bit allocation pointer (bap) table is replaced with a high-efficiency bap table (hebap[]). The hebap[] table employs finer-granularity than the conventional bap table, enabling more efficient allocation of bits.

### 3.3.2 Bit Stream Helper Variables

Several helper variables must be computed during the decode process in order to decode a frame containing at least one channel using AHT (ahte = 1). These variables are not transmitted in the bit stream itself, but are computed from other bit stream parameters. The first helper variables of this type are denoted in the bit stream syntax as ncplregs, nchregs[ch], and nlferegs. The method for computing these variables is presented in the following three sections of pseudo code. Generally speaking, the nregs variables are set equal to the number of times exponents are transmitted in the frame.

**Pseudo Code**

```
/* Only compute ncplregs if coupling in use for all 6 blocks */
ncplregs = 0;
/* AHT is only available in 6 block mode (numblkscod ==0x3) */
for(blk = 0; blk < 6; blk++)
{
    if( (cplstre[blk] == 1) || (cplexpstr[blk] != reuse) )
    {
        ncplregs++;
    }
}
```

**Pseudo Code**

```
for(ch = 0; ch < nfchans; ch++)
{
    nchregs[ch] = 0;
    /* AHT is only available in 6 block mode (numblkscod ==0x3) */
    for(blk = 0; blk < 6; blk++)
    {
        if(chexpstr[blk][ch] != reuse)
        {
            nchregs[ch]++;
        }
    }
}
```

**Pseudo Code**

```
nlferegs = 0;
/* AHT is only available in 6 block mode (numblkscod ==0x3) */
for(blk = 0; blk < 6; blk++)
{
    if( lfeexpstr[blk] != reuse)
    {
        nlferegs++;
    }
}
```

A second set of helper variables are required for identifying which and how many mantissas employ GAQ. The arrays identifying which bins are GAQ coded are called chgaqbin[ch][bin], cplgaqbin[bin], and lfegaqbin[bin]. Since the number and position of GAQ-coded mantissas varies from frame to frame, these variables need to be computed after the corresponding hebap[] array is available, but prior to mantissa unpacking. This procedure is shown in pseudo-code below.

**Pseudo Code**

```
if(cplahtinu == 0)
{
    for(bin = cplstrtmant; bin < cplendmant; bin++)
    {
        cplgaqbin[bin] = 0;
    }
}
else
{
    if (cplgaqmod < 2)
    {
        endbap = 12;
    }
    else
    {
        endbap = 17;
    }
    cplactivegaqbins = 0;
    for(bin = cplstrtmant; bin < cplendmant; bin++)
    {
        if(cplhebap[bin] > 7 && cplhebap[bin] < endbap)
        {
            cplgaqbin[bin] = 1;                    /* Gain word is present */
            cplactivegaqbins++;
        }
        else if (cplhebap[bin] >= endbap)
        {
            cplgaqbin[bin] = -1;                   /* Gain word is not present */
        }
        else
        {
            cplgaqbin[bin] = 0;
        }
    }
}
```

**Pseudo Code**

```
for(ch = 0; ch < nfchans; ch++)
{
    if(chahtinu[ch] == 0)
    {
        for(bin = 0; bin < endmant[ch]; bin++)
        {
            chgaqbin[ch][bin] = 0;
        }
    }
    else
    {
        if (chgaqmod < 2)
        {
            endbap = 12;
        }
        else
        {
            endbap = 17;
        }
        chactivegaqbins[ch] = 0;
        for(bin = 0; bin < endmant[ch]; bin++)
        {
            if(chhebap[ch][bin] > 7 && chhebap[ch][bin] < endbap)
            {
                chgaqbin[ch][bin] = 1;          /* Gain word is present */
                chactivegaqbins[ch]++;
            }
            else if (chhebap[ch][bin] >= endbap)
            {
                chgaqbin[ch][bin] = -1;         /* Gain word not present */
            }
            else
            {
                chgaqbin[ch][bin] = 0;
            }
        }
    }
}
```

**Pseudo Code**

```
if(lfeahtinu == 0)
{
    for(bin = 0; bin < lfeendmant; bin++)
    {
        lfegaqbin[bin] = 0;
    }
}
else
{
    if (lfegaqmod < 2)
    {
        endbap = 12;
    }
    else
    {
        endbap = 17;
    }
    lfeactivegaqbins = 0;
    for(bin = 0; bin < lfeendmant; bin++)
    {
        if(lfehebap[bin] > 7 && lfehebap[bin] < endbap)
        {
            lfegaqbin[bin] = 1;                    /* Gain word is present */
            lfeactivegaqbins++;
        }
        else if (lfehebap[bin] >= endbap)
        {
            lfegaqbin[bin] = -1;                   /* Gain word is not present */
        }
        else
        {
            lfegaqbin[bin] = 0;
        }
    }
}
```

In a final set of helper variables, the number of gain words to be read from the bitstream is computed. These variables are called chgaqsections[ch], cplgaqsections, and lfegaqsections for the independent channels, coupling channel, and LFE channel, respectively. They denote the number of GAQ gain words transmitted in the bit stream, and are computed as shown in the following pseudo code.

**Pseudo Code**

```
if(cplahtinu == 0)
{
      cplgaqsections = 0;
}
else
{
switch(cplgaqmod)
      {
            case 0: /* No GAQ gains present */
            {
                  cplgaqsections = 0;
                  break;
            }
            case 1: /* GAQ gains 1 and 2 */
            case 2: /* GAQ gains 1 and 4 */
            {
                  cplgaqsections = cplactivegaqbins;        /* cplactivegaqbins was computed earlier */
                  break;
            }
            case 3: /* GAQ gains 1, 2, and 4 */
            {
                  cplgaqsections = cplactivegaqbins / 3;
if (cplactivegaqbins % 3) cplgaqsections++;
                  break;
            }
      }
}
```

**Pseudo Code**

```
for(ch = 0; ch <nfchans; ch ++)
{
    if(chahtinu[ch] == 0)
    {
        chgaqsections[ch] = 0;
    }
    else
    {
        switch(chgaqmod[ch])
        {
            case 0: /* No GAQ gains present */
            {
                chgaqsections[ch] = 0;
                break;
            }
            case 1: /* GAQ gains 1 and 2 */
            case 2: /* GAQ gains 1 and 4 */
            {
        chgaqsections[ch] = chactivegaqbins[ch];   /* chactivegaqbins[ch] was computed earlier */
                break;
            }
            case 3: /* GAQ gains 1, 2, and 4 */
{
        chgaqsections[ch] = chactivegaqbins[ch] / 3;
        if (chactivegaqbins[ch] % 3) chgaqsections[ch]++;
        break;
}
        }
    }
}
```

**Pseudo Code**

```
if(lfeahtinu == 0)
{
    lfegaqsections = 0;
}
else
{
    sumgaqbins = 0;
    for(bin = 0; bin < lfeendmant; bin++)
    {
        sumgaqbins += lfegaqbin[bin];
    }
    switch(lfegaqmod)
    {
        case 0: /* No GAQ gains present */
        {
            lfegaqsections = 0;
            break;
        }
        case 1: /* GAQ gains 1 and 2 */
        case 2: /* GAQ gains 1 and 4 */
        {
            lfegaqsections = lfeactivegaqbins;        /* lfeactivegaqbins was computed earlier */
            break;
        }
        case 3: /* GAQ gains 1, 2, and 4 */
        {
            lfegaqsections = lfeactivegaqbins / 3;
            if(lfeactivegaqbins % 3) lfegaqsections++;
            break;
        }
    }
}
```

If the gaqmod bit stream parameter bits are set to 0, conventional scalar quantization is used in place of GAQ coding. If the gaqmod bits are set to 1 or 2, a 1-bit gain is present for each mantissa coded with GAQ. If the gaqmod bits are set to 3, the GAQ gains for three individual mantissas are compositely coded as a 5-bit word.

### 3.3.3 Bit Allocation

When AHT is in use for any independent channel, the coupling channel, or the LFE channel, higher coding efficiency is achieved by allowing quantization noise to be allocated with higher precision. The high precision allocation is achieved using a combination of a new bit allocation pointer look up table and vector quantization. The following section describes the changes to the bit allocation routines defined in the A/52 document in order to achieve higher precision allocation.

#### 3.3.3.1 Parametric Bit Allocation

If the ahtinu flag is set for any independent channel, the coupling channel, or the LFE channel then the bit allocation routine (defined in A/52) for that channel is modified to incorporate the new high efficiency bit allocation pointers. When AHT is in use, the exponents are first decoded and the PSD, excitation function, and masking curve are calculated. The delta bit allocation, if present in the bit stream, is then applied (in accordance with A/52). The final computation of the bit allocation, however, is modified as follows:

The high efficiency bit allocation array (hebap[]) is now computed. The masking curve, adjusted by the snroffset and then truncated, is subtracted from the fine-grain psd[] array. The difference is right shifted by 5 bits, limited, and then used as an address into the hebaptab[] to find the final bit allocation and quantizer type applied to the mantissas. The hebaptab[] array is shown in **Table 3.1**.

At the end of the bit allocation procedure, shown in the following pseudo-code, the hebap[] array contains a series of 5-bit pointers. The pointers indicate how many bits have been allocated to each mantissa and the type of quantizer applied to the mantissas. The correspondence between the hebap pointer and quantizer type and quantizer levels is shown in **Table 3.2**.

Note that if AHT is not in use for a given independent channel, the coupling channel, or the LFE channel, then the bit allocation procedure and resulting bap[] arrays for that channel are the same as described in A/52.

**Pseudo Code**

```
if(ahtinu == 1) /* cplAHTinu, chAHTinu[ch], or lfeAHTinu */
{
    i = start ;
    j = masktab[start] ;
    do
    {
        lastbin = min(bndtab[j] + bndsz[j]), end);
        mask[j] -= snroffset ;
        mask[j] -= floor ;
        if (mask[j] < 0)
        {
            mask[j] = 0 ;
        }
        mask[j] &= 0x1fe0 ;
        mask[j] += floor ;
        for (k = i; k < lastbin; k++)
        {
            address = (psd[i] - mask[j]) >> 5 ;
            address = min(63, max(0, address)) ;
            hebap[i] = hebaptab[address] ;
            i++ ;
        }
        j++;
    }
    while (end > lastbin) ;
}
else
{
    i = start ;
    j = masktab[start] ;
    do
    {
        lastbin = min(bndtab[j] + bndsz[j], end);
        mask[j] -= snroffset ;
        mask[j] -= floor ;
        if (mask[j] < 0)
        {
            mask[j] = 0 ;
        }
        mask[j] &= 0x1fe0 ;
        mask[j] += floor ;
        for (k = i; k < lastbin; k++)
        {
            address = (psd[i] - mask[j]) >> 5 ;
            address = min(63, max(0, address)) ;
            bap[i] = baptab[address] ;
            i++ ;
        }
        j++;
    }
    while (end > lastbin) ;
}
```

3.3.3.2 Bit Allocation Tables

**Table 3.1** High Efficiency Bit Allocation Pointers, hebaptab[]

| Address | hebaptab[address] | Address | hebaptab[address] |
|---|---|---|---|
| 0 | 0 | 32 | 14 |
| 1 | 1 | 33 | 14 |
| 2 | 2 | 34 | 14 |
| 3 | 3 | 35 | 15 |
| 4 | 4 | 36 | 15 |
| 5 | 5 | 37 | 15 |
| 6 | 6 | 38 | 15 |
| 7 | 7 | 39 | 16 |
| 8 | 8 | 40 | 16 |
| 9 | 8 | 41 | 16 |
| 10 | 8 | 42 | 16 |
| 11 | 8 | 43 | 17 |
| 12 | 9 | 44 | 17 |
| 13 | 9 | 45 | 17 |
| 14 | 9 | 46 | 17 |
| 15 | 10 | 47 | 18 |
| 16 | 10 | 48 | 18 |
| 17 | 10 | 49 | 18 |
| 18 | 10 | 50 | 18 |
| 19 | 11 | 51 | 18 |
| 20 | 11 | 52 | 18 |
| 21 | 11 | 53 | 18 |
| 22 | 11 | 54 | 18 |
| 23 | 12 | 55 | 19 |
| 24 | 12 | 56 | 19 |
| 25 | 12 | 57 | 19 |
| 26 | 12 | 58 | 19 |
| 27 | 13 | 59 | 19 |
| 28 | 13 | 60 | 19 |
| 29 | 13 | 61 | 19 |
| 30 | 13 | 62 | 19 |
| 31 | 14 | 63 | 19 |

**Table 3.2** Quantizer Type, Quantizer Level, and Mantissa Bits vs. hebap

| hebap | Quantizer Type | Levels | Mantissa Bits |
|---|---|---|---|
| 0 | NA | NA | 0 |
| 1 | VQ | NA | (2/6) |
| 2 | VQ | NA | (3/6) |
| 3 | VQ | NA | (4/6) |
| 4 | VQ | NA | (5/6) |
| 5 | VQ | NA | (7/6) |
| 6 | VQ | NA | (8/6) |
| 7 | VQ | NA | (9/6) |
| 8 | symmetric + GAQ | 7 | 3 |
| 9 | symmetric + GAQ | 15 | 4 |
| 10 | symmetric + GAQ | 31 | 5 |
| 11 | symmetric + GAQ | 63 | 6 |
| 12 | symmetric + GAQ | 127 | 7 |
| 13 | symmetric + GAQ | 255 | 8 |
| 14 | symmetric + GAQ | 511 | 9 |
| 15 | symmetric + GAQ | 1023 | 10 |
| 16 | symmetric + GAQ | 2047 | 11 |
| 17 | symmetric + GAQ | 4095 | 12 |
| 18 | symmetric + GAQ | 16,383 | 14 |
| 19 | symmetric + GAQ | 65,535 | 16 |

### 3.3.4 Quantization

Depending on the bit allocation pointer (hebap) calculated in Section 3.3.2, the mantissa values are either coded using vector quantization or gain adaptive quantization. The following section describes both of these coding techniques.

#### 3.3.4.1 Vector Quantization

Vector quantization is a quantization technique that takes advantage of similarities and patterns in an ordered series of values, or vector, to reduce redundancy and hence improve coding efficiency. For AHT processing, 6 mantissa values across blocks within a single spectral bin are grouped together to create a 6-dimensional Euclidean space.

If AHT is in use and the bit allocation pointer is between 1 and 7 inclusive, then vector quantization (VQ) is used to encode the mantissas. The range of hebap values that use VQ are shown in **Table 3.2**. If VQ is applied to a set of 6 mantissa values then the data in the bit stream represents an N bit index into a 6-dimensional look up table, where N is dependent on the hebap value as defined in **Table 3.2**. The vector tables are shown in Section 4; the values in the vector tables are represented as 16-bit, signed values.

If a hebap value is within the VQ range, the encoder selects the best vector to transmit to the decoder by locating the vector which minimizes the Euclidean distance between the actual mantissa vector and the table vector. The index of the closest matching vector is then transmitted to the decoder.

In the decoder, the index is read from the bit stream and the mant values are replaced with the values from the appropriate vector table.

#### 3.3.4.2 Gain Adaptive Quantization

Gain-adaptive quantization (GAQ) is a method for quantizing mantissas using variable-length codewords. In the encoder, the technique is based upon conditionally amplifying one or more of the smaller and typically more frequently occurring transform coefficient mantissas in one DCT block, and representing these with a shorter length code. Larger transform coefficients are not gain amplified, but are transmitted using longer codes since these occur relatively infrequently for typical audio signals. The gain words selected by the encoder, one per GAQ-coded DCT block of length six, are packed together with the mantissa codewords and transmitted as side information. With this system, the encoder can adapt to changing local signal statistics from frame to frame, and/or from channel to channel. Since a coding mode using constant-length output symbols is included as a subset, gain-adaptive quantization cannot cause a noticeable coding loss compared to the fixed-length codes used in A/52 AC-3.

In the decoder, the individual gain words are unpacked first, followed by a bit stream parsing operation (using the gains) to reconstruct the individual transform coefficient mantissas. To compensate for amplification applied in the encoder, the decoder applies an attenuation factor to the small mantissas. The level of large mantissas is unaffected by these gain factors in both the encoder and decoder.

The decoder structure for gain-adaptive quantization is presented in **Figure 3.1**. Decoder processing consists of a bit stream deformatter connected in cascade with the switched gain attenuation element, labeled as $1/G_k$ in the figure. The three inputs to the deformatter are the packed mantissa bit stream, the hebap[] output from the parametric bit allocation, and the gaqgain[] array received from the encoder. The hebap[] array is used by the deformatter to determine if the current ($k^{th}$) DCT block of six mantissas to be unpacked is coded with GAQ, and if so, what the small and large mantissa bit lengths are. The gaqgain[] array is processed by the deformatter to produce the gain attenuation element corresponding to each DCT mantissa block identified in the bit stream. The switch position is also derived by the deformatter for each GAQ-coded mantissa. The switch position is determined from the presence or absence of a unique bit stream tag, as discussed in the next paragraph. When the deformatting operation is complete, the dequantized and level-adjusted mantissas are available for the next stage of processing.

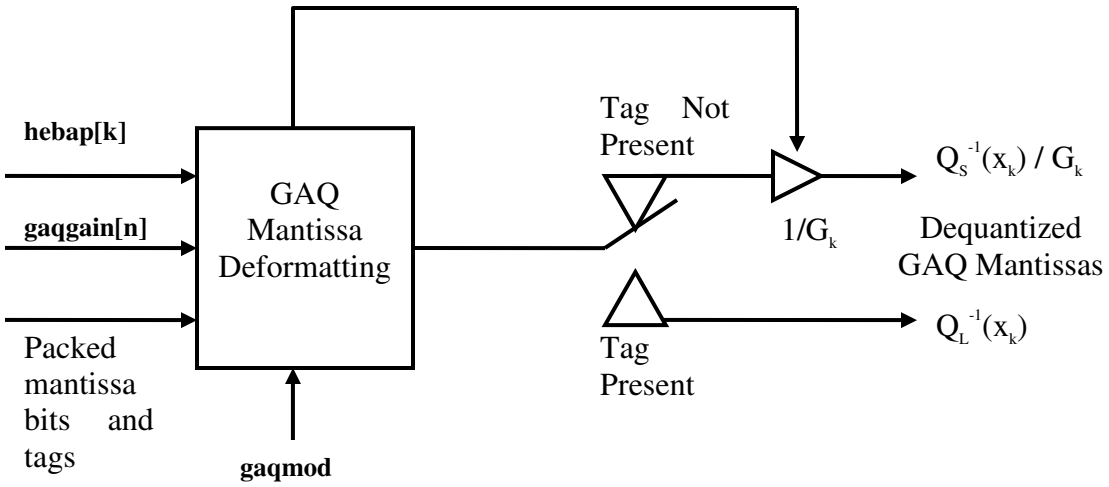


**Figure 3.1** Flow diagram for GAQ mantissa dequantization.

As a means for signaling the two mantissa lengths to the decoder, quantizer output symbols for large mantissas are flagged in the bit stream using a unique identifier tag. In Enhanced AC-3, the identifier tag is the quantizer symbol representing a full-scale negative output (e.g., the '100' symbol for a 3-bit two's complement quantizer). In a conventional mid-tread quantizer, this symbol is often deliberately unused since it results in an asymmetric quantizer characteristic. In gain-adaptive quantization, this symbol is employed to indicate the presence of a large mantissa. The tag length is equal to the length of the small mantissa codeword (computed from hebap[] and gaqgain[]), allowing unique bit stream decoding. If an identifier tag is found, additional bits immediately following the tag (also of known length) convey the quantizer output level for the corresponding large mantissas.

Four different gain transmission modes are available for use in the encoder. The different modes employ switched 0, 1 or 1.67-bit gains. For each independent, coupling, and LFE channel in which AHT is in use, a 2-bit parameter called gaqmod is transmitted once per frame to the decoder. The bitstream parameters, values, and active hebap range are shown for each mode in **Table 3.3**. If gaqmod = 0x0, GAQ is not in use and no gains are present in the bitstream. If gaqmod = 0x1, a 1-bit gain value is present for each block of DCT coefficients having an hebap value between 8 and 11, inclusive. Coefficients with hebap higher than 11 are decoded using the same quantizer as gaqmod 0x0. If gaqmod = 0x2 or 0x3, gain values are present for each block of DCT coefficients having an hebap value between 8 and 16, inclusive. Coefficients with hebap higher than 16 are decoded using the same quantizer as gaqmod 0x0. The difference between the two last modes lies in the gain word length, as shown in the table.

**Table 3.3** Gain Adaptive Quantization Modes

| chgaqmod[ch], cplgaqmod, and lfegaqmod | GAQ Mode for Frame | Active hebap Range (for which gains are transmitted) |
|---|---|---|
| 0x0 | GAQ not in use | None |
| 0x1 | 1-bit gains ($G_k$ = 1 or 2) | 8 ≤ hebap ≤ 11 |
| 0x2 | 1-bit gains ($G_k$ = 1 or 4) | 8 ≤ hebap ≤ 16 |
| 0x3 | 1.67 bit gains ($G_k$ = 1, 2, or 4) | 8 ≤ hebap ≤ 16 |

For the case of gaqmod = 0x1 and 0x2, the gains are coded using binary 0 to signal $G_k$ = 1, and binary 1 to signal $G_k$ = 2 or 4. For the case of gaqmod = 0x3, the gains are composite-coded in triplets (three 3-state gains packed into 5-bit words). The gains are unpacked in a manner similar to exponent unpacking as described in the A/52 document. For example, for a 5-bit composite gain triplet grpgain:

M1 = truncate (grpgain / 9)
M2 = truncate ((grpgain % 9) / 3)
M3 = (grpgain % 9) % 3

In this example, M1, M2, and M3 correspond to mapped values derived from consecutive gains in three ascending frequency blocks, respectively, each ranging in value from 0 to 2 inclusive as shown in **Table 3.4** below.

**Table 3.4** Mapping of Gain Elements, gaqmod = 0x3

| Gain, $G_k$ | Mapped Value |
|---|---|
| 1 | 0 |
| 2 | 1 |
| 4 | 2 |

Details of the GAQ quantizer characteristics are shown in **Table 3.5**. If the received gain is 1, or no gain was received at all, a single quantizer with no tag is used. If the received gain is either 2 or 4, both the small and large mantissas (and associated tags) must be decoded using the quantizer characteristics shown. Both small and large mantissas are decoded by interpreting them as signed two's complement fractional values. The variable m in the table represents the number of mantissa bits associated with a given hebap value as shown in **Table 3.2**.

**Table 3.5** Gain Adaptive Quantizer Characteristics

| | $G_k = 1$ | $G_k = 2$ | | $G_k = 4$ | |
|---|---|---|---|---|---|
| | **Quantizer** | **Small Quantizer** | **Large Quantizer** | **Small Quantizer** | **Large Quantizer** |
| Length of quantizer codeword | m | m–1 | m–1 | m–2 | m |
| Number of reconstruction (output) points | $2^m - 1$ | $2^{m-1} - 1$ | $2^{m-1}$ | $2^{m-2} - 1$ | $2^m$ |
| Step size | $2/(2^{m-1})$ | $1/(2^{m-1})$ | $1/(2^{m-1} - 1)$ | $1/(2^{m-1})$ | $3/(2^{m+1} - 2)$ |

Since the large mantissas are coded using a dead-zone quantizer, a post-processing step is required to transform (remap) large mantissa codewords received by the decoder into a reconstructed mantissa. This remapping is applied when $G_k = 2$ or 4. An identical post-processing step is required to implement the symmetric characteristic for all gaqmod = 0x0 quantizers. The post-process is a computation of the form $y = x + ax + b$. In this equation, x represents a mantissa codeword (interpreted as a signed two's complement fractional value), and the constants a and b are provided in **Table 3.6**. The constants are also interpreted as 16-bit signed two's complement fractional values. The expression for y was arranged for implementation convenience so that all constants will have magnitude less than one. For decoders where this is not a concern, the remapping can be implemented as $y = a'x + b$, where the new coefficient $a' = 1 + a$. The sign of x must be tested prior to retrieving b from the table. Remapping is not applicable to the table entries marked N/A.

**Table 3.6** Large Mantissa Inverse Quantization (Remapping) Constants

| hebap | | $G_k = 1$ | | $G_k = 2$ | | $G_k = 4$ | |
|---|---|---|---|---|---|---|---|
| | | a | b | a | b | a | b |
| 8 | x ≥ 0 | 0x1249 | 0x0000 | 0xd555 | 0x4000 | 0xedb7 | 0x2000 |
| | x < 0 | 0xe186 | 0xf3cf | 0xd555 | 0xeaab | 0xedb7 | 0xfb6e |
| 9 | x ≥ 0 | 0x0889 | 0x0000 | 0xc925 | 0x4000 | 0xe666 | 0x2000 |
| | x < 0 | 0xf507 | 0xfd90 | 0xc925 | 0xd249 | 0xe666 | 0xeccd |
| 10 | x ≥ 0 | 0x0421 | 0x0000 | 0xc444 | 0x4000 | 0xe319 | 0x2000 |
| | x < 0 | 0xfb52 | 0xff73 | 0xc444 | 0xc889 | 0xe319 | 0xe632 |
| 11 | x ≥ 0 | 0x0208 | 0x0000 | 0xc211 | 0x4000 | 0xe186 | 0x2000 |
| | x < 0 | 0xfdd6 | 0xffde | 0xc211 | 0xc421 | 0xe186 | 0xe30c |
| 12 | x ≥ 0 | 0x0102 | 0x0000 | 0xc104 | 0x4000 | 0xe0c2 | 0x2000 |
| | x < 0 | 0xfef6 | 0xfff8 | 0xc104 | 0xc208 | 0xe0c2 | 0xe183 |
| 13 | x ≥ 0 | 0x0081 | 0x0000 | 0xc081 | 0x4000 | 0xe060 | 0x2000 |
| | x < 0 | 0xff7d | 0xfffe | 0xc081 | 0xc102 | 0xe060 | 0xe0c1 |
| 14 | x ≥ 0 | 0x0040 | 0x0000 | 0xc040 | 0x4000 | 0xe030 | 0x2000 |
| | x < 0 | 0xffbf | 0x0000 | 0xc040 | 0xc081 | 0xe030 | 0xe060 |
| 15 | x ≥ 0 | 0x0020 | 0x0000 | 0xc020 | 0x4000 | 0xe018 | 0x2000 |
| | x < 0 | 0xffe0 | 0x0000 | 0xc020 | 0xc040 | 0xe018 | 0xe030 |
| 16 | x ≥ 0 | 0x0010 | 0x0000 | 0xc010 | 0x4000 | 0xe00c | 0x2000 |
| | x < 0 | 0xfff0 | 0x0000 | 0xc010 | 0xc020 | 0xe00c | 0xe018 |
| 17 | x ≥ 0 | 0x0008 | 0x0000 | N/A | N/A | N/A | N/A |
| | x < 0 | 0xfff8 | 0x0000 | N/A | N/A | N/A | N/A |
| 18 | x ≥ 0 | 0x0002 | 0x0000 | N/A | N/A | N/A | N/A |
| | x < 0 | 0xfffe | 0x0000 | N/A | N/A | N/A | N/A |
| 19 | x ≥ 0 | 0x0000 | 0x0000 | N/A | N/A | N/A | N/A |
| | x < 0 | 0x0000 | 0x0000 | N/A | N/A | N/A | N/A |

### 3.3.5 Transform Equations

The AHT processing uses a DCT to achieve higher coding efficiency. Hence, if AHT is in use, the DCT must be inverted prior to applying the exponents. The inverse DCT (IDCT) for AHT is given in the following equation. Any fast technique may be used to invert the DCT in Enhanced AC-3 decoders. In the following equation, $C(k,m)$ is the MDCT spectrum for the $k$th bin and $m$th block, and $X(k,j)$ is the AHT spectrum for the $k$th bin and $j$th block.

$$C(k,m) = \sqrt{2}\sum_{j=0}^{5} R_j X(k,j)\cos\left(\frac{j(2m+1)\pi}{12}\right) \quad m = 0,1,...,5$$

where

$$R_j = \begin{cases} 1 & j \neq 0 \\ 1/\sqrt{2} & j = 0 \end{cases}$$

and $k$ is the bin index, $m$ is the block index, and $j$ is the AHT transform index.

## 3.4 Enhanced Channel Coupling

### 3.4.1 Overview

Enhanced channel coupling is a spatial coding technique that elaborates on conventional channel coupling, principally by adding phase compensation, a de-correlation mechanism, variable time constants, and more compact amplitude representation. The intent is to reduce coupling cancellation artifacts in the encode process by adjusting inter-channel phase before downmixing, and to improve dimensionality of the reproduced signal by restoring the phase angles and degrees of correlation in the decoder. This also allows the process to be used at lower frequencies than conventional channel coupling.

The decoder converts the enhanced coupling channel back into individual channels principally by applying an amplitude scaling and phase adjustment for each channel and frequency sub-band. Additional processing occurs when transients are indicated in one or more channels.

### 3.4.2 Sub-Band Structure for Enhanced Coupling

Enhanced coupling transform coefficients are transmitted in exactly the same manner as conventional coupling. That is, coefficients are reconstructed from exponents and quantized mantissas. Transform coefficients # 13 through # 252 are grouped into 22 sub-bands of either 6 or 12 coefficients each, as shown in **Table 3.7**. The parameter ecplbegf is used to derive the value ecpl_start_subbnd which indicates the number of the enhanced coupling sub-band which is the first to be included in the enhanced coupling process. Below the frequency (or transform coefficient number) indicated by ecplbegf, all channels are independently coded. Above the frequency indicated by ecplbegf, channels included in the enhanced coupling process (chincpl[ch] = 1) share the common enhanced coupling channel up to the frequency (or tc #) indicated by ecplendf. The enhanced coupling channel is coded up to the frequency (or tc #) indicated by ecplendf, which is used to derive ecpl_end_subbnd. The value ecpl_end_subbnd is one greater than the last coupling sub-band which is coded.

**Table 3.7** Enhanced Coupling Sub-bands

| enhanced coupling sub-band # | low tc # | high tc # | lf cutoff (kHz) @ fs=48 kHz | hf cutoff (kHz) @ fs=48 kHz | lf cutoff (kHz) @ fs=44.1 kHz | hf cutoff (kHz) @ fs=44.1 kHz |
|---|---|---|---|---|---|---|
| 0 | 13 | 18 | 1.17 | 1.73 | 1.08 | 1.59 |
| 1 | 19 | 24 | 1.73 | 2.30 | 1.59 | 2.11 |
| 2 | 25 | 30 | 2.30 | 2.86 | 2.11 | 2.63 |
| 3 | 31 | 36 | 2.86 | 3.42 | 2.63 | 3.14 |
| 4 | 37 | 48 | 3.42 | 4.55 | 3.14 | 4.18 |
| 5 | 49 | 60 | 4.55 | 5.67 | 4.18 | 5.21 |
| 6 | 61 | 72 | 5.67 | 6.80 | 5.21 | 6.24 |
| 7 | 73 | 84 | 6.80 | 7.92 | 6.24 | 7.28 |
| 8 | 85 | 96 | 7.92 | 9.05 | 7.28 | 8.31 |
| 9 | 97 | 108 | 9.05 | 10.17 | 8.31 | 9.35 |
| 10 | 109 | 120 | 10.17 | 11.30 | 9.35 | 10.38 |
| 11 | 121 | 132 | 11.30 | 12.42 | 10.38 | 11.41 |
| 12 | 133 | 144 | 12.42 | 13.55 | 11.41 | 12.45 |
| 13 | 145 | 156 | 13.55 | 14.67 | 12.45 | 13.48 |
| 14 | 157 | 168 | 14.67 | 15.80 | 13.48 | 14.51 |
| 15 | 169 | 180 | 15.80 | 16.92 | 14.51 | 15.55 |
| 16 | 181 | 192 | 16.92 | 18.05 | 15.55 | 16.58 |
| 17 | 193 | 204 | 18.05 | 19.17 | 16.58 | 17.61 |
| 18 | 205 | 216 | 19.17 | 20.30 | 17.61 | 18.65 |
| 19 | 217 | 228 | 20.30 | 21.42 | 18.65 | 19.68 |
| 20 | 229 | 240 | 21.42 | 22.55 | 19.68 | 20.71 |
| 21 | 241 | 252 | 22.55 | 23.67 | 20.71 | 21.75 |

Note: At 32 kHz sampling rate the sub-band frequency ranges are 2/3 the values of those for 48 kHz.

**Table 3.8** Enhanced Coupling Start and End Indexes

| ecpl sub-band # | low tc # | high tc # | ecplbegf | ecplendf |
|---|---|---|---|---|
| 0 | 13 | 18 | 0 | |
| 1 | 19 | 24 | | |
| 2 | 25 | 30 | 1 | |
| 3 | 31 | 36 | | |
| 4 | 37 | 48 | 2 | |
| 5 | 49 | 60 | 3 | |
| 6 | 61 | 72 | 4 | |
| 7 | 73 | 84 | 5 | 0 |
| 8 | 85 | 96 | 6 | 1 |
| 9 | 97 | 108 | 7 | 2 |
| 10 | 109 | 120 | 8 | 3 |
| 11 | 121 | 132 | 9 | 4 |
| 12 | 133 | 144 | 10 | 5 |
| 13 | 145 | 156 | 11 | 6 |
| 14 | 157 | 168 | 12 | 7 |
| 15 | 169 | 180 | | 8 |
| 16 | 181 | 192 | 13 | 9 |
| 17 | 193 | 204 | | 10 |
| 18 | 205 | 216 | 14 | 11 |
| 19 | 217 | 228 | | 12 |
| 20 | 229 | 240 | 15 | 13 |
| 21 | 241 | 252 | | 14 |
| 22 | 253 | | | 15 |

The enhanced coupling sub-bands are combined into enhanced coupling bands for which coupling coordinates are generated (and included in the bit stream). The coupling band structure is indicated by ecplbndstrc[sbnd]. Each bit of the ecplbndstrc[] array indicates whether the sub-band indicated by the index is combined into the previous (lower in frequency) enhanced coupling band. Enhanced coupling bands are thus made from integral numbers of enhanced coupling sub-bands. (See Section 2.3.3.19.)

### 3.4.3 Enhanced coupling tables

The following tables are used to lookup various parameter values used by the enhanced coupling process.

**Table 3.9** Sub-band Transform Start Coefficients: ecplsubbndtab[]

| sbnd | ecplsubbndtab[sbnd] |
|---|---|
| 0 | 13 |
| 1 | 19 |
| 2 | 25 |
| 3 | 31 |
| 4 | 37 |
| 5 | 49 |
| 6 | 61 |
| 7 | 73 |
| 8 | 85 |
| 9 | 97 |
| 10 | 109 |
| 11 | 121 |
| 12 | 133 |
| 13 | 145 |
| 14 | 157 |
| 15 | 169 |
| 16 | 181 |
| 17 | 193 |
| 18 | 205 |
| 19 | 217 |
| 20 | 229 |
| 21 | 241 |
| 22 | 253 |

**Table 3.10** Amplitudes: ecplampexptab[], ecplampmanttab[]

| ecplamp | ecplampexptab[ecplamp] | ecplampmanttab[ecplamp] |
|---|---|---|
| 0 | 0 | 0x20 |
| 1 | 0 | 0x1b |
| 2 | 0 | 0x17 |
| 3 | 0 | 0x13 |
| 4 | 0 | 0x10 |
| 5 | 1 | 0x1b |
| 6 | 1 | 0x17 |
| 7 | 1 | 0x13 |
| 8 | 1 | 0x10 |
| 9 | 2 | 0x1b |
| 10 | 2 | 0x17 |
| 11 | 2 | 0x13 |
| 12 | 2 | 0x10 |
| 13 | 3 | 0x1b |
| 14 | 3 | 0x17 |
| 15 | 3 | 0x13 |
| 16 | 3 | 0x10 |
| 17 | 4 | 0x1b |
| 18 | 4 | 0x17 |
| 19 | 4 | 0x13 |
| 20 | 4 | 0x10 |
| 21 | 5 | 0x1b |
| 22 | 5 | 0x17 |
| 23 | 5 | 0x13 |
| 24 | 5 | 0x10 |
| 25 | 6 | 0x1b |
| 26 | 6 | 0x17 |
| 27 | 6 | 0x13 |
| 28 | 6 | 0x10 |
| 29 | 7 | 0x1b |
| 30 | 7 | 0x17 |
| 31 | - | 0x00 |

**Table 3.11** Angles: ecplangletab[]

| ecplangle | ecplangletab[ecplangle] |
|---|---|
| 0 | 0.00000 |
| 1 | 0.03125 |
| 2 | 0.06250 |
| 3 | 0.09375 |
| 4 | 0.12500 |
| 5 | 0.15625 |
| 6 | 0.18750 |
| 7 | 0.21875 |
| 8 | 0.25000 |
| 9 | 0.28125 |
| 10 | 0.31250 |
| 11 | 0.34375 |
| 12 | 0.37500 |
| 13 | 0.40625 |
| 14 | 0.43750 |
| 15 | 0.46875 |
| 16 | 0.50000 |
| 17 | 0.53125 |
| 18 | 0.56250 |
| 19 | 0.59375 |
| 20 | 0.62500 |
| 21 | 0.65625 |
| 22 | 0.68750 |
| 23 | 0.71875 |
| 24 | 0.75000 |
| 25 | 0.78125 |
| 26 | 0.81250 |
| 27 | 0.84375 |
| 28 | 0.87500 |
| 29 | 0.90625 |
| 30 | 0.93750 |
| 31 | 0.96875 |

| ecplangle | ecplangletab[ecplangle] |
|---|---|
| 32 | -1.00000 |
| 33 | -0.96875 |
| 34 | -0.93750 |
| 35 | -0.90625 |
| 36 | -0.87500 |
| 37 | -0.84375 |
| 38 | -0.81250 |
| 39 | -0.78125 |
| 40 | -0.75000 |
| 41 | -0.71875 |
| 42 | -0.68750 |
| 43 | -0.65625 |
| 44 | -0.62500 |
| 45 | -0.59375 |
| 46 | -0.56250 |
| 47 | -0.53125 |
| 48 | -0.50000 |
| 49 | -0.46875 |
| 50 | -0.43750 |
| 51 | -0.40625 |
| 52 | -0.37500 |
| 53 | -0.34375 |
| 54 | -0.31250 |
| 55 | -0.28125 |
| 56 | -0.25000 |
| 57 | -0.21875 |
| 58 | -0.18750 |
| 59 | -0.15625 |
| 60 | -0.12500 |
| 61 | -0.09375 |
| 62 | -0.06250 |
| 63 | -0.03125 |

**Table 3.12** Chaos Scaling: ecplchaostab[]

| ecplchaos | ecplchaostab[ecplchaos] |
|---|---|
| 0 | 0.000000 |
| 1 | -0.142857 |
| 2 | -0.285714 |
| 3 | -0.428571 |
| 4 | -0.571429 |
| 5 | -0.714286 |
| 6 | -0.857143 |
| 7 | -1.000000 |

### 3.4.4 Enhanced Coupling Coordinate Format

Enhanced coupling coordinates exist for each enhanced coupling band [bnd] in each channel [ch] which is coupled (chincp[ch]==1). Enhanced coupling coordinates are derived from three parameters; a 5-bit amplitude scaling value (ecplamp[ch][bnd]), a 6-bit phase angle value (ecplangle[ch][bnd]) and a 3-bit chaos measure (ecplchaos[ch][bnd]). These values will always be transmitted in the first block containing a coupled channel and are optionally transmitted in subsequent blocks, as indicated by the enhanced coupling parameter exists flags (ecplparam1e[ch] and ecplparam2e[ch]). If ecplparam1e[ch] or ecplparam2e[ch] are set to 0, corresponding coordinate values from the previous block are reused.

The ecplamp values 0 to 30 represent gains between 0 dB and –45.16 dB quantized to increments of 1.5015 dB, and the value 31 represents minus infinity dB. The ecplangle values represent angles between 0 and 2pi radians, quantized to increments of 2pi/64 radians. The ecplchaos values each represent a scaling value between 0.0 and –1.0.

### 3.4.5 Enhanced Coupling Processing

This section describes the processing steps required to recover transform coefficients for each coupled channel from the enhanced coupling data.

The following steps are performed for each block.

- Process the enhanced coupling channel
- Prepare amplitudes for each channel and band
- Prepare angles for each channel and band
- Generate transform coefficients for each channel from the processed enhanced coupling channel, amplitudes and angles

#### 3.4.5.1 Process Enhanced Coupling Channel

This section assumes that the enhanced coupling channel mantissas and exponents have been extracted from the bitstream and have been denormalized into fixed point transform coefficients.

Angle adjustment of the enhanced coupling channel requires that time domain aliasing not be present. Therefore the non-aliased enhanced coupling channel must be reconstructed using the enhanced coupling transform coefficients from the previous, current and next blocks. If enhanced coupling is not in use in the previous block, enhanced coupling transform coefficients for the previous block shall be set to zero. Likewise if enhanced coupling is not in use in the next block, enhanced coupling transform coefficients for the next block shall be set to zero.

The following procedure describes how the non-aliased coupling channel is obtained.

1) Define the MDCT transform coefficient buffers for the previous, current and next blocks (of length k=0,1,…,N/2-1 where N=512) as:

$X_{PREV}[k]$ = $ecplmant_{PREV}[k]$ where k = $ecplstartmant_{PREV}$ to $ecplendmant_{PREV}$ - 1
= 0 elsewhere

$X_{CURR}[k]$ = $ecplmant_{CURR}[k]$ where k = $ecplstartmant_{CURR}$ to $ecplendmant_{CURR}$ - 1
= 0 elsewhere

$X_{NEXT}[k]$ = $ecplmant_{NEXT}[k]$ where k = $ecplstartmant_{NEXT}$ to $ecplendmant_{NEXT}$ - 1
= 0 elsewhere

where ecplstartmant = ecplsubbndtab[ecplbegf]
ecplendmant = ecplsubbndtab[ecplendf]

2) Compute the windowed time domain samples xPREV[n], xCURR[n] and xNEXT[n] using the 512-sample IMDCT (as described in steps 1 to 5 of clause 7.9.4.1 in A/52).
3) Overlap and add the second half of the previous sample block and the first half of the next sample block with the current sample block as follows:

**Pseudo Code**

```
for(n=0; n<N/2; n++)
{
    pcm[n] =  xPREV[n+N/2] + xCURR[n];
    pcm[n+N/2] =  xCURR[n+N/2] + xNEXT [n];
}
```

4) Adjust the enhanced coupling channel samples such that the following DFT (FFT) output is an oddly stacked filterbank (as per the MDCT). The window w[n] is defined in Table 7.33 in A/52.

**Pseudo Code**

```
for(n=0; n<N/2; n++)
{
    pcm_real[n] = pcm[n] * w[n] * xcos3[n];
    pcm_real[n+N/2] = pcm[n+N/2] * w[N/2-n-1] * xcos3[n+N/2];
    pcm_imag[n] = pcm[n] * w[n] * xsin3[n];
    pcm_imag[n+N/2] = pcm[n+N/2] * w[N/2-n-1] * xsin3[n+N/2];
}
```

where

```
xcos3[n] = cos(pi * n / N) ;
xsin3[n] = -sin(pi * n / N) ;
```

5) Perform a Discrete Fourier Transform (as an FFT) on the complex samples to create the complex frequency coefficients Z[k], k=0,1,…,N-1

$$Z[k] = \frac{1}{N}\sum_{n=0}^{N-1}\left(pcm\_real[n] + j.pcm\_imag[n]\right).\left(\cos(2\pi kn/N) - j.\sin(2\pi kn/N)\right)$$

3.4.5.2 Process Amplitude Parameters

Amplitude values for each enhanced coupling band [bnd] in each channel [ch] are obtained from the ecplamp parameters as:

**Pseudo Code**

```
if (ecplamp[ch][bnd] == 31)
{
    amp[ch][bnd] = 0;
}
else
{
    amp[ch][bnd] = ( ecplampmanttab[ecplamp[ch][bnd]] / 32 ) >> ecplampexptab[ecplamp[ch][bnd]];
}
```

Modifications are made to the amplitude values using the transmitted chaos measure and transient parameter. Firstly, chaos values for each enhanced coupling band [bnd] in each channel [ch] are obtained from the ecplchaos parameters as follows.

**Pseudo Code**

```
if (ch == firstchincpl)
{
    chaos[ch][bnd] = 0;
}
else
{
    chaos[ch][bnd] = ecplchaostab[ecplchaos[ch][bnd]];
}
```

The chaos modification is then performed as:

**Pseudo Code**

```
if( (ecpltrans[ch] == 0) && (ch != firstchincpl) )
{
    amp[ch][bnd] *= 1 + 0.38 * chaos[ch][bnd];
}
```

Using the ecplbndstrc[] array, the amplitude values amp[ch][bnd] which apply to enhanced coupling bands are converted to values which apply to enhanced coupling sub-bands amp[ch][sbnd] by duplicating values as indicated by values of '1' in ecplbndstrc[]. Amplitude values for individual transform coefficients [bin] are then reconstructed as follows.

**Pseudo Code**

```
bnd = -1;
for(sbnd=ecpl_start_sbnd; sbnd<ecpl_end_sbnd; sbnd++)
{
    if(ecplbndstrc[sbnd] == 0)
    {
        bnd++;
    }
    for(bin=ecplsubbndtab[sbnd]; bin<ecplsubbndtab[sbnd+1]; bin++)
    {
        amp[ch][bin] = amp[ch][bnd];
    }
}
```

#### 3.4.5.3 Process Angle Parameters

Angle values for each enhanced coupling band [bnd] in each channel [ch] are obtained from the ecplangle parameters as follows. Each angle has a value in the range –1.0 to 1.0 (representing –pi to pi). Arithmetic operations performed on these angles "wrap around" such that the results are within the range –1.0 to 1.0. The following pseudo code derives the band angle value associated with a given channel and enhanced coupling angle, ecplangle[ch][bnd].

**Pseudo Code**

```
if (ch == firstchincpl)
{
    angle[ch][bnd] = 0;
}
else
{
    angle[ch][bnd] = ecplangletab[ecplangle[ch][bnd]];
}
```

The above band angle values are used to derive bin angle values associated with individual transform coefficients in one of two ways depending on the ecplangleintrp flag.

If ecplangleintrp is set to 0, then no interpolation is used and the band angle values are applied to bin angle values according to the ecplbndstrc[] array.

If ecplangleintrp is set to 1, then the band angle values are converted to bin angle values using linear interpolation between the centers of each band. The following pseudo code interpolates the band angles (angle[ch][bnd]) into bin angles (angle[ch][bin]) for channel [ch].

**Pseudo Code**

```
if (ecpangleintrp == 1)
{
    bin = ecplsubbndtab[ecpl_start_subbnd];
    for (bnd = 1; bnd < nbands; bnd++)
    {
        nbins_prev = nbins_per_bnd_array[bnd-1];        /* array of length nbands containing band sizes */
        nbins_curr = nbins_per_bnd_array[bnd];
        angle_prev = angle[ch][bnd-1];
        angle_curr = angle[ch][bnd];
        while ((angle_curr – angle_prev) > 1.0) angle_curr -= 2.0;
        while ((angle_prev – angle_curr) > 1.0) angle_curr += 2.0;
        slope = (angle_curr – angle_prev)/((nbins_curr + nbins_prev)/2.0);      /* floating point calculation*/

        / * do lower half of first band */
        if ((bnd == 1) && (nbins_prev > 1))
        {
            if (iseven(nbins_prev))                     /* iseven() returns 1 if value is even, 0 if value is odd */
            {
                y = angle_prev - slope/2;
                bin = nbins_prev/2 - 1;
            }
            else
            {
                y = angle_prev - slope;
                bin = (nbins_prev - 3)/2;
            }
            count = bin + 1;
            for (j = 0; j < count; j++)
            {
                ytmp = y;
                while (y > 1.0) y -= 2.0;
                while (y < (-1.0)) y += 2.0;
                angle[ch][bin--] = y;
                y = ytmp;
                y -= slope;
            }
            bin = count;
        }
        if (iseven(nbins_prev))
        {
            y = angle_prev + slope/2;
            count = nbins_curr/2 + nbins_prev/2;        /* integer calculation */
        }
        else {
            y = angle_prev;
            count = nbins_curr/2 + (nbins_prev + 1)/2;  /* integer calculation */
        }
        for (j = 0; j < count; j++) {
            ytmp = y;
            while (y > 1.0) y -= 2.0;
            while (y < (-1.0)) y += 2.0;
            angle[ch][bin++] = y;
            y = ytmp;
            y += slope;
        }
    }

    /* Finish last band */
    if (iseven(nbins_curr))
```

```
        count = nbins_curr/2;                    /* integer calculation */
    else
        count = nbins_curr/2 + 1;                /* integer calculation */
    for (j = 0; j < count; j++)
    {
        ytmp = y;
        while (y > 1.0) y -= 2.0;
        while (y < (-1.0)) y += 2.0;
        angle[ch][bin++] = y;
        y = ytmp;
        y += slope;
    }
}
```

To assist in de-correlating complex continuous signals, a scaled array of random values is added to each bin angle. The random values depend on whether or not a transient is present in the channel being processed as indicated by ecpltrans[ch].

For channels without a transient, the random values rand_notrans[ch][bin] have the following properties:

- They are uniformly distributed between -1.0 and 1.0.
- They must be unique for each bin [bin] and channel [ch].
- They must only be generated once (for example during decoder initialization) and must stay the same for every block of every frame.

For channels with a transient, the random values rand_trans[ch][bnd] have the following properties:

- They are uniformly distributed between –1.0 and 1.0.
- They must be unique for each band [bnd] and channel [ch].
- New values must be generated for each block.

Using the ecplbndstrc[] array, the banded values for chaos[ch][bnd] and for rand_trans[ch][bnd] are converted to individual bin values by duplicating the band values across each subband and then across each bin within a subband. The chaos and random values are then used to modify each angle value as follows.

**Pseudo Code**

```
if(ecpltrans[ch] == 0)
{
    rand[ch][bin] = rand_notrans[ch][bin]
}
else
{
    rand[ch][bin] = rand_trans[ch][bin]
}

angle[ch][bin] += chaos[ch][bin] * rand[ch][bin];
if(angle[ch][bin] < -1.0)
{
    angle[ch][bin] += 2.0;
}
else if(angle[ch][bin] >= 1.0)
{
    angle[ch][bin] -= 2.0;
}
```

#### 3.4.5.4 Generate Channel Transform Coefficients

Individual channel transform coefficients are then reconstructed from the coupling channel by computing the following complex products.

**Pseudo Code**

```
Zr[ch][bin] = Zr[bin] * amp[ch][bin] * cos(pi * angle[ch][bin])   -  Zi[bin] * amp[ch][bin] * sin(pi * angle[ch][bin]);
Zi[ch][bin] = Zi[bin] * amp[ch][bin] * cos(pi * angle[ch][bin])  +  Zr[bin] * amp[ch][bin] * sin(pi * angle[ch][bin]);

chmant[ch][bin] = -2 * ( y[bin] * Zr[ch][bin]  +  y[N/2-1-bin] * Zi[ch][bin] );
```

Where:

Zr[bin] = real(Z[k]);

Zi[bin] = imag(Z[k]);

and y[bin] = cos(2pi * (N/4 + 0.5) / N * (k + 0.5));

for bin=k=0,1,…,N/2-1

## 3.5 Spectral Extension Processing

Enhanced AC-3 decoders support a new coding technique, based on high frequency regeneration, called spectral extension. A detailed description of the spectral extension process follows.

### 3.5.1 Overview

When spectral extension is in use, high frequency transform coefficients of the channels that are participating in spectral extension are synthesized. Transform coefficient synthesis involves copying low frequency transform coefficients, inserting them as high frequency transform coefficients, blending the inserted transform coefficients with pseudo-random noise, and scaling the blended transform coefficients to match the coarse (banded) spectral envelope of the original signal. To enable the decoder to scale the blended transform coefficients to match the spectral envelope of the original signal, scale factors are computed by the encoder and transmitted to the decoder on a banded basis for all channels participating in the spectral extension process. For a given channel and spectral extension band, the blended transform coefficients for that channel and band are multiplied by the scale factor associated with that channel and band.

The spectral extension process is performed beginning at the spectral extension begin frequency, and ending at the spectral extension end frequency. The spectral extension begin frequency is derived from the spxbegf bit stream parameter. The spectral extension end frequency is derived from the spxendf bit stream parameter.

In some cases, it may be desirable to use channel coupling for a mid-range portion of the frequency spectrum and spectral extension for the higher-range portion of the frequency spectrum. In this configuration, the highest coupled transform coefficient number must be 1 less than the lowest transform coefficient number generated by spectral extension.

### 3.5.2 Sub-Band Structure for Spectral Extension

Transform coefficients #25 through #228 are grouped into 17 sub-bands of 12 coefficients each, as shown in **Table 3.13**. The final table entry does not represent an actual sub-band, but is included for the case when the spxendf parameter is 17. The spectral extension sub-bands containing transform coefficients #37 through #228 coincide with coupling sub-bands. The parameter spxbegf, derived from the bit stream parameter of the same name, indicates the number of the first spectral extension sub-band. The parameter spxendf, derived from the bit stream

parameter of the same name, indicates a number one greater than the last spectral extension sub-band. From the sub-band indicated by spxbegf to the sub-band indicated by spxendf, transform coefficients are synthesized for all channels participating in the spectral extension process (chinspx[ch] == 1). Below the sub-band indicated by spxbegf, channels may be independently coded. Alternatively, channels may be coded independently below the coupling begin frequency, and coupled from the coupling begin frequency to the spectral extension begin frequency.

Spectral extension sub-bands are combined into spectral extension bands for which spectral extension coordinates are generated (and included in the bit stream). Like channel coupling, each spectral extension band is made up of one or more consecutive spectral extension sub-bands. The number of spectral extension bands and the size of each band are determined from the spectral extension band structure array (spxbndstrc[]). Upon frame initialization, the default spectral extension banding structure is copied into the spxbndstrc[] array. If (spxbndstrce == 1), the spxbndstrc[sbnd] bit stream parameters are present in the bit stream and are used to fill the spxbndstrc[] array. If (spxbndstrce == 0), the existing values in the spxbndstrc[] array are used to compute the number of spectral extension bands and the size of each band.

The following pseudo code indicates how to determine the number of spectral extension bands and the size of each band.

**Pseudo Code**

```
nspxbnds = 1;
spxbndsztab[spxbegf] = 12;

for (bnd = spxbegf+1; bnd < spxendf; bnd ++)
{
    if (spxbndstrc[bnd] == 0)
    {
        spxbndsztab[nspxbnds] = 12;
        nspxbnds++;
    }
    else
    {
        spxbndsztab[nspxbnds – 1] += 12;
    }
}
```

**Table 3.13** Spectral Extension Band Table

| spx sub-band # | low tc # | high tc # | spxbegf | spxendf |
|---|---|---|---|---|
| 0 | 25 | 36 | | |
| 1 | 37 | 48 | | |
| 2 | 49 | 60 | 0 | |
| 3 | 61 | 72 | 1 | |
| 4 | 73 | 84 | 2 | |
| 5 | 85 | 96 | 3 | 0 |
| 6 | 97 | 108 | 4 | 1 |
| 7 | 109 | 120 | 5 | 2 |
| 8 | 121 | 132 | | |
| 9 | 133 | 144 | 6 | 3 |
| 10 | 145 | 156 | | |
| 11 | 157 | 168 | 7 | 4 |
| 12 | 169 | 180 | | |
| 13 | 181 | 192 | | 5 |
| 14 | 193 | 204 | | |
| 15 | 205 | 216 | | 6 |
| 16 | 217 | 228 | | |
| 17 | 229 | | | 7 |

### 3.5.3 Spectral Extension Coordinate Format

Spectral extension coordinates exist for each spectral extension band [bnd] of each channel [ch] that is using spectral extension (chinspx[ch] ==1). Spectral extension coordinates must be sent at least once per frame, and may be sent as often as once per block. The spxcoe[ch] bit stream parameter informs the decoder when spectral extension coordinates are present in the bit stream. If (spxcoe[ch] == 0), no spectral extension coordinates for channel [ch] are present in the bit stream, and the previous spectral extension coordinates should be reused. If (spxcoe[ch] == 1), spectral extension coordinates are present in the bit stream for channel [ch].

When present in the bit stream, spectral extension coordinates are transmitted in a floating point format. The exponent is sent as a 4-bit value (spxcoexp[ch][bnd]) indicating the number of right shifts which should be applied to the fractional mantissa value. The mantissas are sent as 2-bit values (spxcomant[ch][bnd]) which must be properly scaled before use. Mantissas are unsigned values so a sign bit is not used. Except for the limiting case where the exponent value = 15, the mantissa value is known to be between 0.5 and 1.0. Therefore, when the exponent value < 15, the msb of the mantissa is always equal to '1' and is not transmitted; the next 2 bits of the mantissa are transmitted. This provides one additional bit of resolution. When the exponent value = 15 the mantissa value is generated by dividing the 2-bit value of spxcomant by 4. When the exponent value is < 15 the mantissa value is generated by adding 4 to the 2-bit value of spxcomant and then dividing the sum by 8.

Spectral extension coordinate dynamic range is increased beyond what the 4-bit exponent can provide by the use of a per channel 2-bit master spectral extension coordinate (mstrspxco[ch]) which is used to scale all of the spectral extension coordinates within that channel. The exponent values for each channel are increased by 3 times the value of mstrspxco which applies to that

channel. This increases the dynamic range of the spectral extension coordinates by an additional 54 dB.

The following pseudo code indicates how to generate the spectral extension coordinate (spxco) for each spectral extension band [bnd] in each channel [ch].

**Pseudo code**

```
if (spxcoexp[ch][bnd] == 15)
{
    spxco_temp[ch][bnd] = spxcomant[ch][bnd] / 4;
}
else
{
    spxco_temp[ch][bnd] = (spxcomant[ch][bnd] + 4) / 8;
}
spxco[ch][bnd] = spxco_temp[ch][bnd] >> (spxcoexp[ch][bnd] + 3*mstrspxco[ch]);
```

### 3.5.4 High Frequency Transform Coefficient Synthesis

This process synthesizes transform coefficients above the spectral extension begin frequency. The synthesis process consists of a number of different steps, described in the following sections.

#### 3.5.4.1 Transform Coefficient Translation

The first step of the high frequency transform coefficient synthesis process is transform coefficient translation. Transform coefficient translation consists of making copies of a channel's low frequency transform coefficients and inserting them as the channel's high frequency transform coefficients. The parameter spxstrtf, derived from the bit stream parameter of the same name, is used as the index into a table to determine the first transform coefficient to be copied. The parameter spxbegf, derived from the bit stream parameter of the same name, is used as the index into a table to determine the first transform coefficient to be inserted. The parameter spxendf, derived from the bit stream parameter of the same name, is used as the index into a table to determine the last transform coefficient to be inserted.

Transform coefficient translation is performed on a banded basis. For each spectral extension band, coefficients are copied sequentially starting with the transform coefficient at copyindex and ending with the transform coefficient at (copyindex + bandsize – 1). Transform coefficients are inserted sequentially starting with the transform coefficient at insertindex and ending with the transform coefficient at (insertindex + bandsize – 1).

Prior to beginning the translation process for each band, the value of (copyindex + bandsize – 1) is compared to the copyendmant parameter. If (copyindex + bandsize – 1) is greater than or equal to the copyendmant parameter, the copyindex parameter is reset to the copystartmant parameter.

The following pseudo code indicates how the spectral component translation process is carried out for channel [ch].

**Pseudo Code**

```
copystartmant = spxbandtable[spxstrtf];
copyendmant = spxbandtable[spxbegf];

copyindex = copystartmant;
insertindex = copyendmant;

for (bnd = 0; bnd < nspxbnds; band++)
{
    bandsize = spxbndsztab[bnd];
    if ((copyindex + bandsize) > copyendmant)
    {
        copyindex = copystartmant;
    }

    for (bin = 0; bin < bandsize; bin++)
    {
        if (copyindex == copyendmant)
        {
            copyindex = copystartmant;
        }
        tc[ch][insertindex] = tc[ch][copyindex];
        insertindex++;
        copyindex++;
    }
}
```

#### 3.5.4.2 Transform Coefficient Noise Blending

The next step of the high frequency transform coefficient synthesis process is transform coefficient noise blending. In this step, the translated transform coefficients are blended with pseudo-random noise in order to create a more natural sounding signal.

##### 3.5.4.2.1 Blending Factor Calculation

The first step of the transform coefficient noise blending process is to determine blending factors for the pseudo-random noise and the translated transform coefficients. The blending factor calculation for each band is based on both the spxblend bit stream parameter and the frequency mid-point of the band. This enables unique blending factors to be computed for each band from a single bit stream parameter. Because the spxblnd parameter exists in the bit stream only when new spectral extension coordinates exist in the bit stream, the blending factors can be reused for all blocks in which spectral extension coordinates are reused.

The following pseudo code indicates how the blending factors for a channel [ch] are determined.

**Pseudo Code**

```
noffset[ch] = spxblend[ch] / 32.0;
spxmant = spxbandtable[spxbegf];

if  (spxcoe[ch])
{
      for (bnd = 0; bnd < nspxbnds; bnd++)
      {
            bandsize = spxbndsztab[bnd];
            nratio = ((spxmant + 0.5*bandsize) / spxbandtable[spxendf]) – noffset[ch];

            if (nratio < 0.0)
            {
                  nratio = 0.0;
            }
            else if (nratio > 1.0)
            {
                  nratio = 1.0;
            }
            nblendfact[ch][bnd] = squareroot(nratio);
            sblendfact[ch][bnd] = squareroot(1 – nratio);
            spxmant += bandsize;
      }
}
```

#### 3.5.4.2.2 Banded RMS Energy Calculation

The next step is to compute the banded RMS energy of the translated transform coefficients. The banded RMS energy measures are needed to properly scale the pseudo-random noise samples prior to blending.

The following pseudo code indicates how to compute the banded RMS energy of the translated transform coefficients for channel [ch].

**Pseudo Code**

```
spxmant = spxbandtab[spxbegf];

for (bnd = 0; bnd < nspxbnds; bnd++)
{
      bandsize = spxbndsztab[bnd];
      accum = 0;
      for (bin = 0; bin < bandsize; bin++)
      {
            accum = accum + (tc[ch][spxmant] * tc[ch][spxmant]);
            spxmant++;
      }
      rmsenergy[ch][band] = squareroot(accum / bandsize);
}
```

#### 3.5.4.2.3 Noise Scaling and Transform Coefficient Blending Calculation

In order to properly blend the translated transform coefficients with pseudo-random noise, the noise components for each band must be scaled to match the energy of the translated transform coefficients in the band. The energy matching can be achieved by scaling all the noise components in a given band by the RMS energy of the translated transform coefficients in that band, provided the noise components are generated by a zero-mean, unity-variance noise generator. Once the zero-mean, unity-variance noise components for each band have been scaled

by the RMS energy for that band, the scaled noise components can be blended with the translated transform coefficients.

The following pseudo code indicates how the translated transform coefficients and pseudo-random noise for a channel [ch] are blended. The function noise() returns a pseudo-random number generated from a zero-mean, unity-variance noise generator.

**Pseudo Code**

```
spxmant = spxbandtable[spxbegf];

for (bnd = 0; bnd < nspxbnds; bnd++)
{
    bandsize = spxbndsztab[bnd];
    nscale = rmsenergy[ch][bnd] * nblendfact[ch][bnd];
    sscale = sblendfact[ch][bnd];
    for (bin = 0; bin < bandsize; bin++)
    {
        tctemp = tc[ch][spxmant];
        ntemp = noise();
        tc[ch][spxmant] = tctemp * sscale + ntemp * nscale;
        spxmant++;
    }
}
```

#### 3.5.4.3 Blended Transform Coefficient Scaling

The final step of the high frequency transform coefficient synthesis process is blended transform coefficient scaling. In this step, blended transform coefficients are scaled by the spectral extension coordinates to form the final synthesized high frequency transform coefficients. After this step, the banded energy of the synthesized high frequency transform coefficients should match the banded energy of the high frequency transform coefficients of the original signal.

The blended transform coefficient scaling process for channel [ch] is shown in the following pseudo code.

**Pseudo Code**

```
spxmant = spxbandtable[spxbegf];

for (bnd = 0; bnd < nspxbnds; bnd++)
{
    bandsize = spxbndsztab[bnd];
    spxcotemp = spxco[ch][bnd];
    for (bin = 0; bin < bandsize; bin++)
    {
        tctemp = tc[ch][spxmant];
        tc[ch][spxmant] = tctemp * spxcotemp * 32;
        spxmant++;
    }
}
```

## 3.6 Transient Pre-Noise Processing

Transient pre-noise processing is a new audio coding improvement technique, using time scaling synthesis, which reduces the duration of pre-noise introduced by low-bit rate audio coding of transient material. A detailed description of the time scaling synthesis process follows.

### 3.6.1 Overview

When transient pre-noise reduction processing is used, decoded PCM audio located prior to transient material is processed in the decoder using time scaling synthesis. The synthesized PCM audio is used to remove the transient pre-noise, thereby improving the perceived quality of low-bit rate audio coded transient material. To enable the decoder to efficiently perform transient pre-noise processing with no impact on decoding latency, transient location detection and time scaling synthesis analysis is performed by the encoder and the information transmitted to the decoder. The encoder performs transient pre-noise processing for each full bandwidth audio channel and transmits the information once per frame. The transmitted transient location and time scaling synthesis information are relative to the first decoded PCM sample contained in the audio frame containing the bit stream information. It should be noted that it is possible for the time scaling synthesis parameters contained in audio frame N, to reference PCM samples and transients located in audio frame N+1, but this does not create a requirement for multi-frame decoding.

### 3.6.2 Application of Transient Pre-Noise Processing Data

The bit stream syntax and high level description of the transient pre-noise parameters contained in the audio frame field are outlined in Sections 2.3.3 and 2.4.2, respectively. The parameter transproce indicates whether any of the full bandwidth channels in the current audio frame have associated transient pre-noise time scaling synthesis processing information. If transproce is set to a value of '1', then the parameter chintransproc[ch] can be set for each full bandwidth channel. For each full bandwidth channel where chintransproc[ch] is set to a value of '1', the transient location parameter transprocloc[ch] and time scaling length parameter transproclen[ch] are each set to values that have been calculated by the encoder.

**Figure 3.2** provides an overview of how the transient pre-noise parameters that are computed and transmitted by the encoder are applied in the decoder. As shown in **Figure 3.2***a*, the parameter transprocloc[ch] identifies the location of the transient relative to the first sample of decoded PCM channel data in the audio frame that contains the transient pre-noise processing parameters. As defined, transprocloc[ch] has four sample resolution to reduce the data rate required to transmit the transient location and must be multiplied by 4 to get the location of the transient in samples. As also shown in **Figure 3.2***a*, the parameter transproclen[ch] provides the time scaling length, in samples, relative to the leading edge of the audio coding block in which the transient is located. As shown in **Figure 3.2***b*, the location of the leading edge of the audio coding block indicates the start of the transient pre-noise. The start of the audio coding block and location of the transient provide the total length of the transient pre-noise in samples, PN. As part of the normal decoding operation, the decoder inherently knows the starting location of the audio coding block that contains the transient and this does not need to be transmitted.

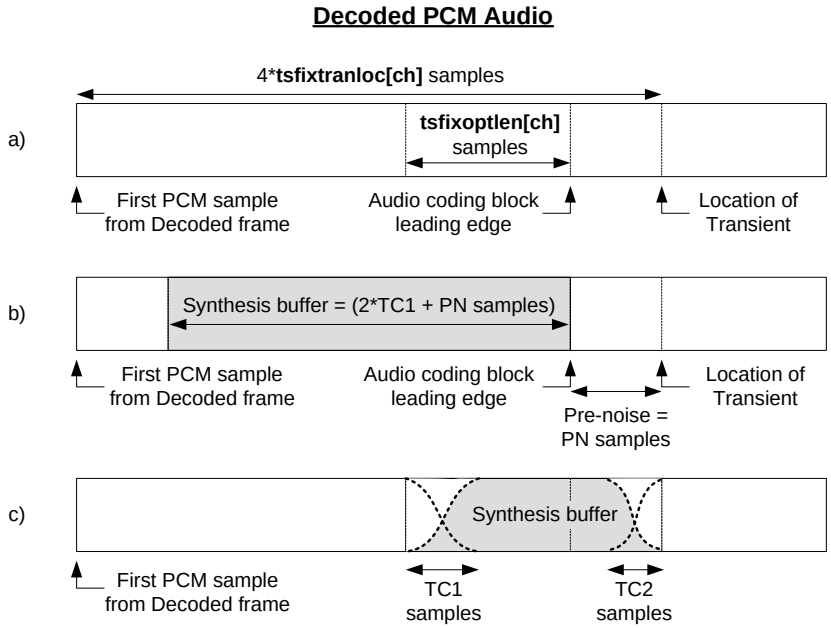


**Figure 3.2** Transient pre-noise time scaling synthesis summary.

Also shown in **Figure 3.2***b* is how the time scaling synthesis audio buffer, which is used to modify the transient pre-noise, is defined relative to the decoded audio frame. The time scaling synthesis buffer is (2*TC1 + PN) PCM samples in length, where TC1 is a time scaling synthesis system parameter equal to 256 samples. The first sample of the time scaling synthesis buffer is located (2*TC1 + 2*PN) samples before the location of the transient.

**Figure 3.2***c* outlines how the time scaling synthesis buffer is used along with the transproclen[ch] parameter to remove the transient pre-noise. As shown in **Figure 3.2***c* the original decoded audio data is cross-faded with the time scaling synthesis buffer starting at the sample located (PN + transproclen[ch]) samples before the location of the transient. The length of the cross-fade is TC1 or 256 samples. Nearly any pair of constant amplitude cross-fade windows may be used to perform the overlap-add between the original data and the synthesis buffer, although standard Hanning windows have been shown to provide good results. The time scaling synthesis buffer is then used to overwrite the decoded PCM audio data that is located before the transient, including the transient pre-noise. This overwriting continues until TC2 samples before the transient where TC2 is another time scaling synthesis system parameter equal to 128 samples. At TC2 samples before the transient, the time scaling synthesis audio buffer is cross-faded with the original decoded PCM data using a set of constant amplitude cross-fade windows.

The following pseudo code outlines how to implement the transient pre-noise time scaling synthesis functionality in the decoder for a single full bandwidth channel, [ch].

Where:

win_fade_out1 = TC1 sample length cross-fade out window (unity to zero in value)
win_fade_in1 = TC1 sample length cross-fade in window (zero to unity in value)
win_fade_out2 = TC2 sample length cross-fade out window (unity to zero in value)
win_fade_in2 = TC2 sample length cross-fade in window (zero to unity in value)

**Pseudo Code**

```
/* unpack the transient location relative to first decoded pcm sample. */
transloc = transprocloc[ch];
/* unpack time scaling length relative to first decoded pcm sample. */
translen = transproclen[ch];
/* compute the transient pre-noise length using audio coding block first sample, aud_blk_samp_loc. */
pnlen = (transloc – aud_blk_samp_loc);
/* compute the total number of samples corrected in the output buffer. */
tot_corr_len = (pnlen + translen);

/* create time scaling synthesis buffer from decoded output pcm buffer, pcm_out[ ]. */
for (samp = 0;  samp < (2*TC1 + pnlen); samp++)
      synth_buf[samp] = pcm_out[(transloc – (2*TC + 2*pnlen) + samp)];
end

/* use time scaling synthesis buffer to overwrite and correct pre-noise in output pcm buffer. */
start_samp = (transloc – pnlen – translen);
for (samp = 0; samp < TC1; samp++)
{
      pcm_out[start_samp + samp] =    (pcm_out[start_samp + samp] * win_fade_out1[samp]) +
                                      (synth_buf[samp] * win_fade_in1[samp]);
}
for (samp = TC1; samp < (tot_corr_len – TC2); samp++)
{
      pcm_out[start_samp + samp] = synth_buf[samp];
}
for (samp = (tot_corr_len – TC2); samp < tot_corr_len; samp++)
{
      pcm_out[start_samp + samp] =    (pcm_out[start_samp + samp] * win_fade_in2[samp]) +
                                      (synth_buf[samp] * win_fade_out2[samp]);
}
```

## 3.7 Channel and Program Extensions

The Enhanced AC-3 bit stream syntax allows for time-multiplexed substreams to be present in a single bit stream. By allowing time-multiplexed substreams, the Enhanced AC-3 bit stream syntax enables a single program with greater than 5.1 channels, multiple programs of up to 5.1 channels, or a mixture of programs with up to 5.1 channels and programs with greater than 5.1 channels, to be carried in a single bit stream.

### 3.7.1 Overview

An Enhanced AC-3 bit stream must consist of at least one independently decodable stream (type 0 or 2). Optionally, Enhanced AC-3 bit streams may consist of multiple independent substreams (type 0 or 2) or a combination of multiple independent (type 0 and 2) and multiple dependent (type 1) substreams.

All Enhanced AC-3 decoders must be able to decode independent substream 0, and skip over any additional independent and dependent substreams present in the bit stream.

Optionally, Enhanced AC-3 decoders may use the information present in the acmod, lfeon, strmtyp, substreamid, chanmape, and chanmap bit stream parameters to decode bit streams with a single program with greater than 5.1 channels, multiple programs of up to 5.1 channels, or a mixture of programs with up to 5.1 channels and programs with greater than 5.1 channels.

### 3.7.2 Decoding a Single Program with Greater than 5.1 Channels

When a bit stream contains a single program with greater than 5.1 channels, independent substream 0 contains a 5.1 channel downmix of the program for compatibility with playback systems containing 5.1 speakers. The audio in independent substream 0 can also be downmixed for compatibility with playback systems containing less than 5.1 speakers. Decoders reproducing 5.1 or fewer channels from a program containing greater than 5.1 channels shall decode only independent substream 0 and skip all associated dependent substreams.

In order to accommodate playback by systems with greater than 5.1 speakers, the Enhanced AC-3 bit stream will carry one or more dependent substreams that contain channels that either replace or supplement the 5.1 channel data carried in independent substream 0.

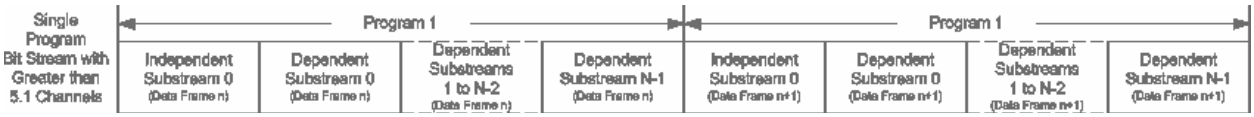


**Figure 3.3** Bitstream with a single program of greater than 5.1 channels.

If the chanmape parameter of a dependent substream is set to 0, then the acmod and lfeon parameters of the dependent substream are used to identify the channels present in the dependent substream, and the corresponding audio channels in the independent substream are overwritten with the dependent audio channel data. For example, if the dependent substream uses acmod 1/0 (center channel only) and has lfeon set to 1, then the center channel audio data carried in the dependent stream will replace the center channel audio data carried in the independent stream, and the LFE audio data carried in the dependent stream will replace the LFE data carried in the independent stream.

If the chanmape parameter of a dependent substream is set to 1, then the chanmap parameter is used to determine the channel mapping for all channels contained in the dependent stream. Each bit of the chanmap parameter corresponds to a particular channel location. Audio data is contained in the dependent substream for each chanmap bit that is set to 1. The order of the coded channels in the dependent substream is the same as the order of the bits set to 1 in the chanmap parameter. For example, if the Left channel bit is set to 1 in the channel map field, then Left channel audio data will be contained in the first coded channel of data in the dependent substream. If channels are present in the dependent substream that correspond to channels in the associated independent substream, then the dependent substream data for those channels replaces the independent substream data for the corresponding channels. All channels present in the dependent substream that do not correspond to channels in the independent substream are used to enable output for speaker configurations with greater than 5.1 channels.

The maximum number of channels rendered for a single program is 14.

### 3.7.3 Decoding Multiple Programs with up to 5.1 Channels

When an Enhanced AC-3 bit stream contains multiple independent substreams, each independent substream corresponds to an independent audio program. The application interface may inform the decoder which independent audio program should be decoded by selecting a specific independent substream ID. The decoder should then only decode substreams with the desired independent substream ID, and skip over any other programs present in the bit stream with different substream ID's. The default program selection should always be Program 1.

In some cases, it may be desirable to decode multiple independent audio programs. In these cases, the application interface should inform the decoder which independent audio programs to

decode by selecting specific independent substream ID's. The decoder should then decode all substreams with the desired independent substream ID's, and skip over any other programs present in the bit stream with different substream ID's.

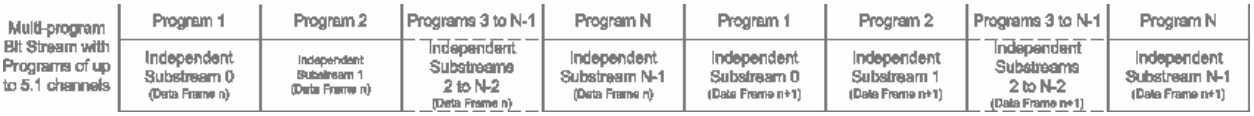


**Figure 3.4** Bitstream with multiple programs of up to 5.1 channels.

### 3.7.4 Decoding a Mixture of Programs with up to 5.1 Channels and Programs with Greater than 5.1 Channels

When an Enhanced AC-3 bit stream contains multiple independent and dependent substreams, each independent substream and its associated dependent substreams correspond to an independent audio program. The application interface may inform the decoder which independent audio program should be decoded by selecting a specific independent substream ID. The decoder should then only decode the desired independent substream and all its associated dependent substreams, and skip over all other independent substreams and their associated dependent substreams. If the selected independent audio program contains greater than 5.1 channels, the decoder should decode the selected independent audio program as explained in Section 3.7.2. The default program selection should always be Program 1.

In some cases, it may be desirable to decode multiple independent audio programs. In these cases, the application interface should inform the decoder which independent audio programs to decode by selecting specific independent substream ID's. The decoder should then decode the desired independent substreams and their associated dependent substreams, and skip over all other independent substreams and associated dependent substreams present in the bit stream.

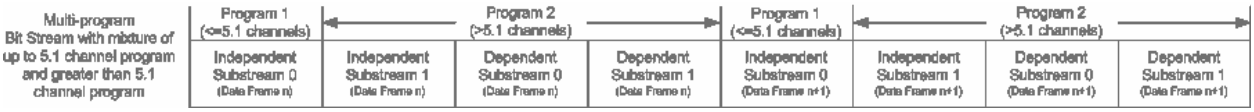


**Figure 3.5** Bitstream with mixture of programs of up to 5.1 channels and programs of greater than 5.1 channels.

### 3.7.5 Dynamic Range Compression for Programs Containing Greater than 5.1 Channels

A program using channel extensions to convey greater than 5.1 channels may require two different sets of compr and dynrng metadata words: one set for the 5.1 channel downmix carried by independent substream 0 and a separate set for the complete (greater than 5.1 channel) mix. If a decoder is reproducing the complete mix, the compr and dynrng metadata words carried in independent substream 0 shall be ignored. The decoder shall instead use the compr and dynrng metadata words carried by the associated dependent substream. If multiple associated dependent substreams are present, only the last dependent substream may carry compr and dynrng metadata words, and these metadata words shall apply to all substreams in the program, including the independent substream.

The compre bit is used by the decoder to determine which dependent substream in a program is the last dependent substream of the program. Therefore, the compre bit in the last dependent substream of a program must be set to 1, and the compre bit in all other dependent substreams of

the program must be set to 0. Additionally, the compr2e, dynrnge, and dynrng2e bits for all but the last dependent substream of a program must be set to 0. The compr2e, dynrnge, and dynrng2e bits for the last dependent substream shall be set as required to transmit the proper compr2, dynrng, and dynrng2 words for the program.

Note that the compr2e, compr2, dynrng2e, and dynrng2 metadata words are only present in the bit stream when acmod = 0.

## 4. AHT VECTOR QUANTIZATION TABLES

**Table 4.1** VQ Table for Hebap 1 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0x1bff | 0x1283 | 0x0452 | 0x10ad | 0x28ac | 0x12d4 |
| 1 | 0xe9ba | 0xf38d | 0xc76d | 0xfa90 | 0xf815 | 0x0351 |
| 2 | 0x0279 | 0x1837 | 0x1b61 | 0xce15 | 0xf6fe | 0xf5b4 |
| 3 | 0xfa44 | 0xe489 | 0x1da8 | 0x2979 | 0xe8c6 | 0xf40a |

**Table 4.2** VQ Table for Hebap 2 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0xd0d7 | 0x0260 | 0xe495 | 0x024e | 0x0fa0 | 0x0365 |
| 1 | 0x1a24 | 0x3d49 | 0xe7de | 0xdbe9 | 0xffb6 | 0x0085 |
| 2 | 0x073f | 0xfc23 | 0x5074 | 0xf498 | 0xee85 | 0x00e1 |
| 3 | 0xfb56 | 0xf0c3 | 0xfccb | 0xe65a | 0xfc95 | 0xb0b6 |
| 4 | 0xf536 | 0xf393 | 0xf002 | 0xea09 | 0xbdcf | 0x2625 |
| 5 | 0x060b | 0x1ab7 | 0x07bc | 0x4f09 | 0xfbd1 | 0xec86 |
| 6 | 0x184d | 0xba05 | 0xea74 | 0x187a | 0x0166 | 0x048a |
| 7 | 0x0ea9 | 0xfbd6 | 0x10bb | 0xf365 | 0x3e38 | 0x27ca |

**Table 4.3** VQ Table for Hebap 3 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0xd8d4 | 0x512b | 0x2ae6 | 0xee30 | 0x031e | 0xffbc |
| 1 | 0x2b2a | 0x500a | 0xe627 | 0xeb22 | 0xf8fb | 0xf9a1 |
| 2 | 0x0f89 | 0xfde2 | 0x1bce | 0xfb72 | 0x499c | 0x3956 |
| 3 | 0xef20 | 0xffa0 | 0xe381 | 0xfe14 | 0xa9de | 0xef4b |
| 4 | 0x0a84 | 0x16e0 | 0x159a | 0x5566 | 0xe3d4 | 0xeb33 |
| 5 | 0xff79 | 0xa4a1 | 0x03c2 | 0x1fb3 | 0xfd7c | 0x017e |
| 6 | 0xf9e5 | 0x0d48 | 0xf31d | 0x1255 | 0xe514 | 0x577e |
| 7 | 0x0dcf | 0x0bd6 | 0x1c80 | 0x1846 | 0x4ffc | 0xd0bd |
| 8 | 0x0039 | 0xe559 | 0x0738 | 0xa8b3 | 0xe8e1 | 0x1aa7 |
| 9 | 0xfccb | 0xf1b9 | 0xfe7d | 0xe793 | 0xf939 | 0xa89b |
| 10 | 0xe862 | 0x0632 | 0xb636 | 0xc7c8 | 0x23fe | 0x02c1 |
| 11 | 0xe9ac | 0x0108 | 0xb9d4 | 0x391a | 0x1ef1 | 0xfeaf |
| 12 | 0xff92 | 0x006c | 0x0008 | 0x004a | 0xffa7 | 0xffce |
| 13 | 0x19d4 | 0xfa13 | 0x54b7 | 0xf986 | 0xe0f3 | 0xff0a |
| 14 | 0x54a3 | 0xe741 | 0xdf9e | 0xff9b | 0xfabb | 0xffea |
| 15 | 0xaa0d | 0xe6b4 | 0x1f26 | 0x0288 | 0x0806 | 0xfeb5 |

**Table 4.4** VQ Table for Hebap 4 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0x5903 | 0x15c0 | 0xe9e6 | 0xff64 | 0xfe06 | 0xffdf |
| 1 | 0x19ec | 0xee0f | 0x375d | 0xbc6f | 0xbf75 | 0x0360 |
| 2 | 0x0e4a | 0x580c | 0x0068 | 0xf91d | 0xffac | 0x0006 |
| 3 | 0x544c | 0xba69 | 0xe38e | 0xf9d9 | 0xf7e2 | 0xfec0 |
| 4 | 0xf747 | 0x2721 | 0xf558 | 0x3a5a | 0xcab8 | 0xbb05 |
| 5 | 0xf9e4 | 0xbab6 | 0xb527 | 0x35a7 | 0x0ac5 | 0x0b87 |
| 6 | 0x11a8 | 0x1586 | 0x1ce1 | 0x2a2f | 0x4b1f | 0xca36 |
| 7 | 0x018f | 0x0ba0 | 0xfbb5 | 0x1395 | 0xfb79 | 0x564f |
| 8 | 0x0e28 | 0xf6c9 | 0x1248 | 0xf742 | 0x58ae | 0x0eb5 |
| 9 | 0xef97 | 0xdfa3 | 0xe566 | 0xcf9a | 0xb812 | 0x3c16 |
| 10 | 0xebb2 | 0xe52b | 0xd8c1 | 0xdf54 | 0xc16a | 0xafae |
| 11 | 0xff72 | 0xa771 | 0xfe90 | 0x1127 | 0xfe30 | 0xfff3 |
| 12 | 0x032e | 0xfba2 | 0xfbbf | 0xa9fd | 0x004a | 0x0611 |
| 13 | 0xf9ae | 0x4b16 | 0xbb16 | 0xcb4e | 0x034a | 0xf6fb |
| 14 | 0x1251 | 0x406a | 0x514d | 0xc3e5 | 0xefbc | 0xf080 |
| 15 | 0xf314 | 0x2bce | 0xcb1a | 0x351f | 0xb3ef | 0x35ca |
| 16 | 0x0719 | 0x0356 | 0x52e9 | 0xfc3a | 0xf995 | 0xfef4 |
| 17 | 0xf5e5 | 0xff95 | 0xb146 | 0x0178 | 0x0496 | 0xfed0 |
| 18 | 0xf499 | 0x01c5 | 0xeaf2 | 0x02ee | 0xa9ee | 0xfc2e |
| 19 | 0xb5bc | 0x41c7 | 0x2710 | 0xf204 | 0x08a3 | 0x05b3 |
| 20 | 0x0553 | 0xf59e | 0xffdf | 0xf01d | 0x048d | 0xaa1f |
| 21 | 0xde70 | 0xf538 | 0xbb90 | 0xc18f | 0x3a31 | 0x052b |
| 22 | 0x028c | 0xdb8d | 0x0cb5 | 0xc6e2 | 0x2f95 | 0x4cec |
| 23 | 0xe727 | 0x168d | 0xc3dd | 0x438b | 0x40ce | 0xf496 |
| 24 | 0xfd6b | 0xfda7 | 0x0649 | 0x5852 | 0x03e0 | 0xfbeb |
| 25 | 0x1361 | 0x2393 | 0x2bd9 | 0x1e95 | 0x3fc0 | 0x48c3 |
| 26 | 0xaa90 | 0xfa67 | 0x008a | 0x05be | 0xf89d | 0xff3c |
| 27 | 0xb3d5 | 0xb8e5 | 0x2b30 | 0xfdfc | 0x09ef | 0xf737 |
| 28 | 0xfb54 | 0xbb5a | 0x4eb6 | 0x2cc6 | 0xfe6f | 0x0a3b |
| 29 | 0x121e | 0xe026 | 0x2e73 | 0xc271 | 0x44cf | 0xc595 |
| 30 | 0xffad | 0x0116 | 0x0143 | 0x0037 | 0xff66 | 0x00e8 |
| 31 | 0x1e6c | 0x05b6 | 0x47db | 0x3bc0 | 0xc26d | 0xfb95 |

**Table 4.5** VQ Table for Hebap 5 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0xf2be | 0xb2ee | 0x0b93 | 0x2576 | 0x1234 | 0x4cd9 |
| 1 | 0xc2cf | 0xe6fb | 0x4507 | 0x0f14 | 0xdfd8 | 0xb41b |
| 2 | 0x1173 | 0x019c | 0xba2f | 0xe09b | 0x02b3 | 0xbc65 |
| 3 | 0x0dfc | 0x093b | 0x1ae6 | 0x0eb3 | 0x18eb | 0xafd6 |
| 4 | 0xbcb2 | 0xc8cb | 0xfa8c | 0xa27d | 0x20b5 | 0xcf07 |
| 5 | 0xe077 | 0xac23 | 0xc2ea | 0x0c8e | 0x1fa9 | 0xe8b3 |
| 6 | 0x10f7 | 0x1431 | 0x0a7b | 0xbe4a | 0xebe6 | 0xbf52 |
| 7 | 0x18cc | 0xd654 | 0x32c3 | 0x9c64 | 0xa9b6 | 0x0ffb |
| 8 | 0xf4c0 | 0xdf52 | 0xe8b0 | 0xbcfa | 0xf5b2 | 0x5a5c |
| 9 | 0xec19 | 0xc837 | 0xa89d | 0x54ed | 0x0e69 | 0x0b91 |
| 10 | 0xf675 | 0xbab5 | 0x6243 | 0x0a93 | 0x063a | 0x0007 |
| 11 | 0xb835 | 0x2332 | 0x10ae | 0x02db | 0xfe56 | 0xfd80 |
| 12 | 0xa371 | 0x609c | 0x160a | 0x0264 | 0xfecc | 0xfc3c |
| 13 | 0xfd01 | 0x04f4 | 0x00e1 | 0x0663 | 0x00ad | 0x0394 |
| 14 | 0x154f | 0x195d | 0x1326 | 0x2940 | 0x5a01 | 0xbd0c |
| 15 | 0x4343 | 0xadc2 | 0xb6e4 | 0x1348 | 0xf2a4 | 0x0d1d |
| 16 | 0xfa92 | 0x3c80 | 0xaa46 | 0xc6ed | 0x053b | 0x021e |
| 17 | 0xe52e | 0xf732 | 0xd0da | 0xf3fd | 0xb1f3 | 0xaf72 |
| 18 | 0xf8f5 | 0x2dff | 0x053f | 0x22d5 | 0x02b5 | 0x5fb1 |
| 19 | 0xab96 | 0x24f6 | 0x1249 | 0x2426 | 0xe179 | 0x3e20 |
| 20 | 0xea49 | 0xf436 | 0xdc2f | 0xfabd | 0xa7ed | 0x3244 |
| 21 | 0xfe92 | 0x13d4 | 0xf941 | 0x4fcb | 0xfee5 | 0xf495 |
| 22 | 0xf8a2 | 0xe757 | 0xfc55 | 0xf7df | 0xfa89 | 0x0db9 |
| 23 | 0xf3a7 | 0xfde7 | 0xec2d | 0x2c04 | 0x4bc4 | 0x03dd |
| 24 | 0x0929 | 0x1039 | 0x1689 | 0xef4f | 0x00e9 | 0xfe71 |
| 25 | 0xaa7a | 0xfb8e | 0xbfa6 | 0x170e | 0x1570 | 0xf375 |
| 26 | 0x2717 | 0xcf0e | 0x498d | 0x51c4 | 0xfb7a | 0x06fe |
| 27 | 0xfb73 | 0x1396 | 0xfb51 | 0x190f | 0xdf1e | 0xadd2 |
| 28 | 0x0764 | 0xf232 | 0x0ee7 | 0xe92a | 0x402b | 0x4f40 |
| 29 | 0xf598 | 0xd295 | 0xefcd | 0xb879 | 0xa74a | 0x3a00 |
| 30 | 0x4368 | 0x28b3 | 0x1e54 | 0x2f08 | 0x4a0c | 0x09dd |
| 31 | 0xac55 | 0xb703 | 0xd56f | 0x1110 | 0xe475 | 0x11bb |
| 32 | 0xf9da | 0x0802 | 0x1680 | 0x60b4 | 0x3e6f | 0x450e |
| 33 | 0xfde6 | 0xa6ad | 0x2b3b | 0x283d | 0x0181 | 0x0210 |
| 34 | 0xdeef | 0xf42f | 0xc01b | 0xa53b | 0x406b | 0x0e46 |
| 35 | 0x16d0 | 0x023f | 0x2e72 | 0x079b | 0x6245 | 0x19fd |
| 36 | 0x19e1 | 0xf244 | 0xf854 | 0x0f0a | 0xfe7a | 0xff8c |
| 37 | 0x4655 | 0x51a4 | 0x37f3 | 0xe23b | 0xd556 | 0x2e1a |
| 38 | 0xed07 | 0xf48c | 0xcbea | 0xe179 | 0x5476 | 0x08db |
| 39 | 0xfdbd | 0xdb29 | 0xfd14 | 0xacb7 | 0x304f | 0x2049 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 40 | 0xdf83 | 0x055f | 0xba49 | 0x0b69 | 0x2366 | 0x561e |
| 41 | 0x47de | 0x21bb | 0xfa21 | 0xf68e | 0xb889 | 0xc672 |
| 42 | 0xf455 | 0x3b19 | 0xf2fd | 0x571c | 0x3636 | 0xcab9 |
| 43 | 0x16f2 | 0xb5ae | 0x3ce3 | 0x2c56 | 0xaefe | 0x07b3 |
| 44 | 0x062d | 0xe4d5 | 0xac40 | 0x0997 | 0x0041 | 0x019e |
| 45 | 0x0203 | 0xee8c | 0xfd67 | 0xedc0 | 0x007d | 0xb4ea |
| 46 | 0x53f7 | 0xb0b3 | 0xf8b0 | 0xf87a | 0xff2d | 0xfc02 |
| 47 | 0x1445 | 0xd026 | 0xf911 | 0xa402 | 0xee3e | 0x16b5 |
| 48 | 0x0141 | 0xe745 | 0x3936 | 0x1b3f | 0xf913 | 0x0363 |
| 49 | 0xca0a | 0x0c6c | 0x1ef7 | 0x01bc | 0x4c60 | 0x0c4a |
| 50 | 0xe5fc | 0x2fdc | 0xf84c | 0x4400 | 0xa128 | 0xcd64 |
| 51 | 0xfd17 | 0x3814 | 0xfbad | 0x5cbe | 0xda61 | 0xb858 |
| 52 | 0x476c | 0xe11b | 0xe295 | 0x4aae | 0x1e29 | 0xce8d |
| 53 | 0x0786 | 0x3afd | 0xcdd0 | 0x0869 | 0x547f | 0x0748 |
| 54 | 0xf7ae | 0x5b78 | 0x42a0 | 0xc313 | 0xf9f8 | 0x0057 |
| 55 | 0x207a | 0xd1d0 | 0x38f5 | 0xaf91 | 0x1ed3 | 0xf7cd |
| 56 | 0x4c90 | 0x591e | 0xbc68 | 0xf808 | 0x011d | 0xf0e9 |
| 57 | 0xdfea | 0xb86e | 0x29e4 | 0xca50 | 0xcb63 | 0xf97e |
| 58 | 0x380f | 0x1310 | 0xb1be | 0x03c4 | 0xef83 | 0xff4c |
| 59 | 0x9fea | 0xbf05 | 0x4d0c | 0x1725 | 0x12a9 | 0x113e |
| 60 | 0xf641 | 0x5bc5 | 0xc0f3 | 0x0b66 | 0xfbf2 | 0xf826 |
| 61 | 0x4a1e | 0xf614 | 0x341f | 0x057d | 0xe7ce | 0xfb90 |
| 62 | 0x09b9 | 0x3566 | 0x586e | 0xe371 | 0xff7f | 0xf518 |
| 63 | 0xc976 | 0x4187 | 0x59e4 | 0x02d8 | 0x0d45 | 0x00a2 |
| 64 | 0x0bde | 0x03e1 | 0x2246 | 0xaa2f | 0xe62f | 0x038e |
| 65 | 0xcf64 | 0xa88e | 0xf5be | 0xeb51 | 0x4c40 | 0x2690 |
| 66 | 0xf889 | 0xb89e | 0xb7b6 | 0xc58e | 0x1298 | 0x1bcf |
| 67 | 0x206a | 0xf45e | 0x651e | 0x1dec | 0xe127 | 0x03fc |
| 68 | 0x17f4 | 0x3b17 | 0x4945 | 0xa0ce | 0xe67f | 0xe61d |
| 69 | 0x1ef4 | 0x2f5d | 0xdb0d | 0xa266 | 0x157e | 0x03a9 |
| 70 | 0xbd60 | 0xeb03 | 0x09da | 0x0147 | 0x0469 | 0xfe7a |
| 71 | 0x3d9e | 0x4df3 | 0xd774 | 0x2ba4 | 0xf3dd | 0x3a05 |
| 72 | 0xd180 | 0xe065 | 0xb9f8 | 0xa8f1 | 0xbcab | 0xe56d |
| 73 | 0xcdc2 | 0xf784 | 0xe693 | 0x1727 | 0x30aa | 0xa8ad |
| 74 | 0xfe0f | 0x0142 | 0x040e | 0xe60d | 0xeae4 | 0x4f57 |
| 75 | 0x043b | 0xa638 | 0xded2 | 0x2f62 | 0xfd06 | 0x0a3f |
| 76 | 0x13cb | 0x4d00 | 0xf893 | 0xffe2 | 0xfebb | 0x0055 |
| 77 | 0x03db | 0xe93a | 0x1074 | 0xdcba | 0x23a1 | 0x9e32 |
| 78 | 0xe144 | 0x1c74 | 0xcffc | 0x3272 | 0xaba8 | 0x51cd |
| 79 | 0xf9a2 | 0xe1f2 | 0xf775 | 0xdeb3 | 0xea1c | 0x9d94 |
| 80 | 0xe5f4 | 0x0184 | 0xa7f9 | 0x05f6 | 0x237a | 0x00c1 |
| 81 | 0xe145 | 0xa8dc | 0x142b | 0x016a | 0x03b0 | 0xfefd |
| 82 | 0x0ef0 | 0xd1b6 | 0x1da7 | 0xa578 | 0x62fe | 0x5cdb |
| 83 | 0xd6f8 | 0x101b | 0xad89 | 0x52b5 | 0x57a7 | 0xfcba |

| | | | | | | |
|---|---|---|---|---|---|---|
| 84 | 0xed8d | 0x5523 | 0x1828 | 0xff86 | 0x066a | 0xfd33 |
| 85 | 0x5fb8 | 0x4daf | 0xf805 | 0x03da | 0x0007 | 0xffc9 |
| 86 | 0x954f | 0xff79 | 0x0985 | 0x0103 | 0x0059 | 0x0133 |
| 87 | 0x5f7e | 0xf0df | 0xeaf1 | 0xfccc | 0xf6ad | 0x0169 |
| 88 | 0x1599 | 0x1698 | 0x48fa | 0x00f2 | 0xaa78 | 0xf05d |
| 89 | 0x5720 | 0x1183 | 0x02d2 | 0xd02e | 0x1d92 | 0x3c58 |
| 90 | 0x21e1 | 0x0bc1 | 0x4fd5 | 0x5274 | 0xad94 | 0xf3f8 |
| 91 | 0xfb94 | 0x0a91 | 0xf8df | 0x152c | 0xfcef | 0x4864 |
| 92 | 0x4224 | 0xcb33 | 0xbf83 | 0xc5f6 | 0xb099 | 0xc873 |
| 93 | 0x08ab | 0x0564 | 0x53e2 | 0xfb98 | 0x0147 | 0x0053 |
| 94 | 0xf77f | 0x540d | 0xf0f0 | 0xc89c | 0xff34 | 0xf771 |
| 95 | 0x03b9 | 0xdb2e | 0x3e02 | 0xd62a | 0xf361 | 0x5226 |
| 96 | 0xfe5b | 0xfa9f | 0x0280 | 0xdfd1 | 0xae10 | 0x087e |
| 97 | 0x10d5 | 0x4852 | 0xdc74 | 0xb871 | 0xc362 | 0x0e78 |
| 98 | 0xe8c9 | 0x01c1 | 0xdf3d | 0x0433 | 0xa93e | 0xec80 |
| 99 | 0x0b89 | 0x31f4 | 0x476d | 0x0596 | 0x3a59 | 0x54e3 |
| 100 | 0xf49f | 0x0191 | 0xed7d | 0xb177 | 0x06a3 | 0xfb85 |
| 101 | 0x0d79 | 0x1479 | 0x2295 | 0x5676 | 0xe285 | 0x05ab |
| 102 | 0xf796 | 0x2188 | 0x46c8 | 0xc302 | 0x4b77 | 0xe899 |
| 103 | 0x0dad | 0x0b19 | 0x1709 | 0x18fd | 0x21b6 | 0x59ea |
| 104 | 0x09a3 | 0x0b8c | 0x017b | 0x1647 | 0xa9e1 | 0xf773 |
| 105 | 0xbdbd | 0xfdae | 0x4986 | 0xeb51 | 0x0668 | 0x0306 |
| 106 | 0x0b50 | 0xfa70 | 0x0e02 | 0xf70c | 0x4dc6 | 0xf8e2 |
| 107 | 0xb771 | 0x52e3 | 0xc94f | 0xcee3 | 0x4052 | 0x027b |
| 108 | 0xf832 | 0xb48e | 0xbf71 | 0x2fb0 | 0xbf40 | 0xe152 |
| 109 | 0xda36 | 0x03f4 | 0xab73 | 0x0b43 | 0xce58 | 0x0911 |
| 110 | 0xfc13 | 0x01d7 | 0xf1d3 | 0x1f6d | 0xd4b1 | 0x63bd |
| 111 | 0x102d | 0xac20 | 0xf58f | 0x02f4 | 0xfd69 | 0xfdf5 |
| 112 | 0x195a | 0x2153 | 0x4b59 | 0x4a05 | 0x17cc | 0xdb7d |
| 113 | 0x4245 | 0x6017 | 0x36c8 | 0x2758 | 0xfde8 | 0xd73a |
| 114 | 0xe02d | 0x0861 | 0xa60c | 0xbd4f | 0x154b | 0xeecf |
| 115 | 0xc5e7 | 0x5028 | 0xb881 | 0xda0b | 0xd193 | 0xba59 |
| 116 | 0xf70e | 0xc8d8 | 0x0816 | 0x57c3 | 0x0687 | 0x02d5 |
| 117 | 0xdea6 | 0x3925 | 0x0dc1 | 0xaf9f | 0x1a11 | 0x2008 |
| 118 | 0x4f18 | 0x113a | 0xfaaa | 0xfdb7 | 0x04cd | 0xf66f |
| 119 | 0x1d2b | 0xe414 | 0x3563 | 0xdfca | 0x5778 | 0xbc58 |
| 120 | 0xf874 | 0x0f23 | 0xdc98 | 0xf11c | 0x03be | 0x0109 |
| 121 | 0xeed1 | 0x0b8f | 0xc1d9 | 0x4c8e | 0x135a | 0xfbaf |
| 122 | 0x4659 | 0xd93d | 0xb927 | 0xf0ea | 0x2baa | 0x16bd |
| 123 | 0xc6fc | 0xfb35 | 0x25bc | 0x5473 | 0x2bdc | 0xd237 |
| 124 | 0xfd2f | 0xf95c | 0x006d | 0xf7a2 | 0x003d | 0xe58c |
| 125 | 0x9fd5 | 0xa808 | 0x15e8 | 0xf85b | 0xf91f | 0xfc0c |
| 126 | 0xa350 | 0xee9d | 0xf580 | 0xc6a9 | 0xef56 | 0x26bf |
| 127 | 0x212f | 0xfc82 | 0x4fd6 | 0xca04 | 0xbc8d | 0x008b |

**Table 4.6** VQ Table for Hebap 6 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0x27aa | 0x1cc5 | 0x41dd | 0x48f9 | 0xa693 | 0xf1cc |
| 1 | 0xf5c5 | 0xf134 | 0xea67 | 0xebb8 | 0xdccf | 0xb0b6 |
| 2 | 0xea31 | 0xa6f0 | 0x5331 | 0x1b64 | 0x02e9 | 0x02d0 |
| 3 | 0x01ac | 0xfa4d | 0x006d | 0xf3f6 | 0x0169 | 0xdf2d |
| 4 | 0x1fe1 | 0x5781 | 0x00f1 | 0x06db | 0xfc96 | 0xf4f8 |
| 5 | 0x0474 | 0x3163 | 0x0902 | 0x56f7 | 0x9dc6 | 0xbb6b |
| 6 | 0xf5cf | 0x0d33 | 0x2861 | 0xb2ee | 0xc394 | 0xa278 |
| 7 | 0xf038 | 0xce04 | 0x9b54 | 0x3328 | 0x0f41 | 0x0523 |
| 8 | 0x1210 | 0xa309 | 0x3528 | 0x62e5 | 0xfb69 | 0x087d |
| 9 | 0xff9f | 0x35b3 | 0xebfe | 0x5ad7 | 0x1076 | 0xa97f |
| 10 | 0x1ade | 0xfebe | 0x4758 | 0xfcaa | 0xd174 | 0xfd23 |
| 11 | 0x4380 | 0xce83 | 0xda23 | 0x5c0b | 0xc090 | 0xfae3 |
| 12 | 0x16aa | 0xec98 | 0x4c46 | 0xad36 | 0x9fd2 | 0xff49 |
| 13 | 0x16db | 0xc0f7 | 0x3b7d | 0xdae8 | 0xf9fe | 0x0179 |
| 14 | 0x3710 | 0x61e1 | 0x346b | 0x2062 | 0x5b18 | 0x41b2 |
| 15 | 0xe3a3 | 0x0076 | 0xc205 | 0x4a99 | 0x2635 | 0xfeeb |
| 16 | 0xef40 | 0x5455 | 0xcc18 | 0xc07d | 0x40f9 | 0xed02 |
| 17 | 0x132d | 0xb4ef | 0x5b73 | 0x3971 | 0xfd2e | 0x007d |
| 18 | 0x4c06 | 0xed84 | 0xf878 | 0xd2f9 | 0x5122 | 0x1531 |
| 19 | 0x9456 | 0xf4bf | 0xef15 | 0x0180 | 0xf7c9 | 0x0557 |
| 20 | 0xfef6 | 0xdc29 | 0x1541 | 0x66dd | 0xf87c | 0x107d |
| 21 | 0xf466 | 0xb136 | 0xaac8 | 0x154a | 0xe2fe | 0x14e0 |
| 22 | 0xff23 | 0xe5d8 | 0x025b | 0xdc4c | 0x051c | 0x948e |
| 23 | 0x2595 | 0xdf44 | 0xf851 | 0x24bb | 0xf98d | 0x5921 |
| 24 | 0x1d8e | 0xeb7e | 0xefbb | 0x0569 | 0xfc22 | 0x0230 |
| 25 | 0xfb12 | 0x60a2 | 0xb58f | 0x29f5 | 0x1da1 | 0xe446 |
| 26 | 0x01c3 | 0x4ea2 | 0xd923 | 0xe881 | 0xf774 | 0xfa4e |
| 27 | 0x56e9 | 0x24a4 | 0x2388 | 0x2acf | 0xf6c3 | 0xf174 |
| 28 | 0x48ec | 0xfd76 | 0xfb2e | 0x2b54 | 0x1dfe | 0x1751 |
| 29 | 0x4b07 | 0xfa33 | 0xfbcc | 0xfd25 | 0xfd54 | 0x002b |
| 30 | 0xec93 | 0x3476 | 0x4eab | 0x003c | 0x01dc | 0xfc59 |
| 31 | 0xb1c3 | 0x2206 | 0x09c3 | 0x03f8 | 0xfb7a | 0x014f |
| 32 | 0x98d3 | 0x48a6 | 0xf767 | 0xfd63 | 0x0d51 | 0x0319 |
| 33 | 0xed8a | 0x22ab | 0x9fe1 | 0xda52 | 0x0e3b | 0xfee5 |
| 34 | 0x33f7 | 0xac64 | 0xf195 | 0xfb60 | 0xf84e | 0x064c |
| 35 | 0x00ad | 0x003c | 0x0397 | 0x04cd | 0x1b1e | 0xfd67 |
| 36 | 0x3ff9 | 0x425f | 0x14dd | 0xc941 | 0xf700 | 0xb05a |
| 37 | 0x62f6 | 0xd68f | 0x2eab | 0xe21b | 0xe725 | 0x36ea |
| 38 | 0x5d79 | 0xcc35 | 0xe3c6 | 0xfc57 | 0x00ea | 0xff45 |
| 39 | 0x18a7 | 0xf8ab | 0x30da | 0xf8a9 | 0x493f | 0xa4d3 |
| 40 | 0x026d | 0x192d | 0x0d1a | 0xa12e | 0x20d6 | 0x14c3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 41 | 0xf31f | 0xec56 | 0xeda0 | 0xec28 | 0x9b7e | 0x14e3 |
| 42 | 0xfb05 | 0xcc11 | 0xfc3b | 0xa4ea | 0x04be | 0x6693 |
| 43 | 0xe794 | 0x2733 | 0xb177 | 0x3bc5 | 0xc137 | 0x1410 |
| 44 | 0x255a | 0xf0b9 | 0xb3ca | 0x1289 | 0x56fe | 0xefb5 |
| 45 | 0x1f2a | 0xb370 | 0x36c8 | 0xe98f | 0xae89 | 0x22eb |
| 46 | 0x0007 | 0xf039 | 0x03df | 0xe84f | 0x0034 | 0xb421 |
| 47 | 0x0d9d | 0x0b99 | 0x1e34 | 0x1e6a | 0x62e0 | 0x183e |
| 48 | 0xfc41 | 0xcdf4 | 0xf8d0 | 0xa729 | 0x1c9c | 0x2a4e |
| 49 | 0xedb2 | 0x068e | 0xd844 | 0xebab | 0x10c6 | 0xfb09 |
| 50 | 0x0f31 | 0x0516 | 0x1d1a | 0x027e | 0x4f96 | 0xf3c3 |
| 51 | 0xcf30 | 0xdc5d | 0x481f | 0xcfc9 | 0xe3ba | 0x4878 |
| 52 | 0xe7d7 | 0x21c9 | 0xe509 | 0xfc81 | 0x42d5 | 0x40dc |
| 53 | 0xd958 | 0x6fa3 | 0x0b1d | 0x0668 | 0x0b6d | 0xfed6 |
| 54 | 0x3a78 | 0x9a7c | 0x3a1e | 0xa234 | 0x0717 | 0xe6b6 |
| 55 | 0x65fb | 0x142e | 0x52e9 | 0x3e01 | 0x5471 | 0x39e9 |
| 56 | 0xab4c | 0x4036 | 0x5018 | 0xc7f6 | 0xe436 | 0xefbe |
| 57 | 0x6fe7 | 0x005a | 0xf9dc | 0x0315 | 0xfc7a | 0xffb5 |
| 58 | 0xfa39 | 0x09a7 | 0xf023 | 0x0e1c | 0xf740 | 0x2aa2 |
| 59 | 0x21a8 | 0x4453 | 0x4367 | 0xbbd0 | 0x427e | 0xc01b |
| 60 | 0xaf0e | 0xb75b | 0x62ba | 0x4538 | 0xf20b | 0x069f |
| 61 | 0xfc1b | 0x17f1 | 0xe761 | 0x2bf2 | 0xd3a1 | 0xb2e5 |
| 62 | 0xffb6 | 0xf05f | 0xf9d0 | 0x3448 | 0x00a2 | 0xff70 |
| 63 | 0xfdef | 0x524c | 0x1ef3 | 0xd37c | 0x01a6 | 0xffe6 |
| 64 | 0x1bbe | 0xcb25 | 0xb1a9 | 0x0a45 | 0xff4e | 0xfe53 |
| 65 | 0x23f1 | 0x0558 | 0xa922 | 0x0a3f | 0xafed | 0x6139 |
| 66 | 0xfe50 | 0x1a13 | 0xfef6 | 0x2213 | 0x0050 | 0x6d78 |
| 67 | 0x4c25 | 0xf3dc | 0xdbd3 | 0x0776 | 0xaaef | 0x14e1 |
| 68 | 0x36ff | 0xd31f | 0x313c | 0x17bf | 0x4da5 | 0x0523 |
| 69 | 0x2ac3 | 0x266d | 0xb74c | 0x3d7e | 0x12b8 | 0x025d |
| 70 | 0xf90f | 0x0eae | 0xf009 | 0x54c0 | 0x1788 | 0xfdc0 |
| 71 | 0x0def | 0xf206 | 0x3ffb | 0x0a78 | 0xf928 | 0x02cc |
| 72 | 0xec47 | 0xfa89 | 0xee3a | 0xfd74 | 0xbac7 | 0xf2da |
| 73 | 0xf1cd | 0xeeec | 0xe686 | 0xa978 | 0x1cd6 | 0x05b2 |
| 74 | 0x2fd2 | 0x4af6 | 0x160e | 0xe179 | 0xb0bf | 0x5360 |
| 75 | 0xe2ac | 0x4df0 | 0x5bf6 | 0xd9e7 | 0x1625 | 0xf83a |
| 76 | 0xf71d | 0x3c4e | 0x2a9b | 0xba29 | 0x1961 | 0x350e |
| 77 | 0xc1ea | 0xc2e2 | 0xed94 | 0x1783 | 0x5eba | 0xe7dd |
| 78 | 0xf7ff | 0xe538 | 0xfb48 | 0x0396 | 0x4547 | 0xffbb |
| 79 | 0xf177 | 0x238b | 0xc13f | 0xa3bb | 0x175d | 0xf6d8 |
| 80 | 0x1eb6 | 0xdd2a | 0x5de1 | 0x63a4 | 0xd4e7 | 0xfd1b |
| 81 | 0xced4 | 0x4c0c | 0x3939 | 0x3c5b | 0xad16 | 0x0493 |
| 82 | 0x0836 | 0x047b | 0x0ae5 | 0x1000 | 0x0883 | 0x222e |
| 83 | 0xb8da | 0xbaa2 | 0xd782 | 0xebad | 0xfbd6 | 0xf22b |
| 84 | 0xf4fd | 0xb20a | 0xd16f | 0x1790 | 0x207b | 0x2886 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 85 | 0xdc8a | 0xf7cc | 0x4be7 | 0xffef | 0x02dc | 0xfd4f |
| 86 | 0xc750 | 0xb4e8 | 0xe449 | 0x4927 | 0x074e | 0x597a |
| 87 | 0x0f48 | 0x0293 | 0x63fd | 0xf05a | 0x2593 | 0x036d |
| 88 | 0x0a38 | 0x58a7 | 0xe976 | 0x4600 | 0x0ee4 | 0x4efc |
| 89 | 0x0a01 | 0x68df | 0xeb83 | 0xd564 | 0x08d0 | 0xfdfb |
| 90 | 0xec92 | 0x00c6 | 0xaa21 | 0xf1e8 | 0x569e | 0xb614 |
| 91 | 0x533c | 0xfb45 | 0x4ac8 | 0x4133 | 0xf9cc | 0x2c7e |
| 92 | 0xf902 | 0x0f77 | 0xf260 | 0x1b5b | 0xe43d | 0x518d |
| 93 | 0xe824 | 0xb9dd | 0xb6de | 0x60bb | 0x407c | 0x0c8b |
| 94 | 0x4fee | 0x9e65 | 0xb077 | 0x1184 | 0xec09 | 0xfe22 |
| 95 | 0xe716 | 0xf832 | 0xd80b | 0xfdcf | 0xa9e9 | 0xa8bd |
| 96 | 0x0be7 | 0xb65e | 0x1da2 | 0x3997 | 0xb26a | 0x18cf |
| 97 | 0xec49 | 0x057d | 0xda38 | 0x041f | 0xaa87 | 0x2ba2 |
| 98 | 0x0d99 | 0xda1d | 0x197e | 0xbef1 | 0x591d | 0x5593 |
| 99 | 0xb776 | 0x445d | 0x3948 | 0x050b | 0x13a2 | 0x4cdc |
| 100 | 0x3f06 | 0xb29e | 0xbdc4 | 0xb9ed | 0xbddb | 0x16a8 |
| 101 | 0xdfe0 | 0x132c | 0x22e7 | 0x08e0 | 0xfb8c | 0xa54f |
| 102 | 0x0624 | 0x0ac1 | 0xf9c2 | 0x085f | 0xf2ee | 0xaa5a |
| 103 | 0xd998 | 0xfbdc | 0x9356 | 0x04be | 0x1c79 | 0x0096 |
| 104 | 0x0062 | 0x0602 | 0x0217 | 0x4415 | 0xa562 | 0xfc7b |
| 105 | 0x535c | 0xb14e | 0x0ce1 | 0xf930 | 0xdff1 | 0xac2a |
| 106 | 0xefba | 0xede7 | 0xba12 | 0x1566 | 0x0505 | 0x0088 |
| 107 | 0x4919 | 0x520b | 0x60f2 | 0x2c9d | 0x0502 | 0xedf6 |
| 108 | 0xf231 | 0x1dd4 | 0xfef7 | 0x085d | 0xfcc3 | 0xf80d |
| 109 | 0xf390 | 0x4d01 | 0x0ad7 | 0xfffe | 0x0442 | 0x0068 |
| 110 | 0xe58d | 0xb127 | 0x0b7a | 0xf7b3 | 0xffdc | 0x04f4 |
| 111 | 0x2558 | 0x24d6 | 0x2572 | 0x5654 | 0x3603 | 0x1898 |
| 112 | 0xfde9 | 0xb1ce | 0x10b4 | 0xf8b4 | 0xfe40 | 0xbce1 |
| 113 | 0xa0e0 | 0x37a4 | 0xcab1 | 0xadd0 | 0x08df | 0x2d23 |
| 114 | 0xf5aa | 0x3c4d | 0xee13 | 0x48ce | 0xef35 | 0xfd92 |
| 115 | 0xb1a0 | 0x1049 | 0x46c3 | 0xfa84 | 0x359a | 0xf8df |
| 116 | 0xc019 | 0x2378 | 0x02e8 | 0x5605 | 0x007d | 0x2a2a |
| 117 | 0x25ac | 0xc6f1 | 0xb7d1 | 0xc686 | 0x2ba6 | 0xaeee |
| 118 | 0xfeba | 0xa32e | 0x1800 | 0x1ee5 | 0x025a | 0x0604 |
| 119 | 0xe606 | 0x19ea | 0xce75 | 0x5394 | 0x5131 | 0xe549 |
| 120 | 0x109c | 0xadcd | 0x15fc | 0x48ff | 0x5d34 | 0x2088 |
| 121 | 0x4642 | 0x1648 | 0xeb83 | 0xb953 | 0xfdd5 | 0x0c93 |
| 122 | 0x17cb | 0x3798 | 0xec03 | 0xbbd0 | 0xb404 | 0xd27f |
| 123 | 0xabae | 0x2c26 | 0x3c4a | 0x63f6 | 0x1a79 | 0x97c5 |
| 124 | 0x536b | 0xdfcc | 0x16f5 | 0xf22c | 0x17bf | 0xf5f9 |
| 125 | 0x0a2b | 0xf669 | 0x152d | 0xd002 | 0xb564 | 0x15c6 |
| 126 | 0xf947 | 0x98e7 | 0xa390 | 0x5978 | 0xfea3 | 0x0ecb |
| 127 | 0x088d | 0xfb4d | 0x14dc | 0x0cb1 | 0xa7a7 | 0x0068 |
| 128 | 0xf980 | 0xd4f4 | 0xf4d7 | 0xaf0d | 0xa20f | 0x4dbc |

| | | | | | | |
|---|---|---|---|---|---|---|
| 129 | 0x5959 | 0xe34f | 0xb7cf | 0xc6e8 | 0xdf30 | 0xcd5b |
| 130 | 0x0ec1 | 0x0f76 | 0x202f | 0x500e | 0xe4b1 | 0xfb4f |
| 131 | 0xff60 | 0xf9b3 | 0xfce7 | 0xde17 | 0x023d | 0x0308 |
| 132 | 0x10c9 | 0xf136 | 0x4f95 | 0x17c2 | 0xeb37 | 0xb820 |
| 133 | 0x4939 | 0x099f | 0x3102 | 0xe1bb | 0xe1ca | 0xf779 |
| 134 | 0x2b42 | 0xed90 | 0x5667 | 0x0721 | 0xa0e1 | 0x0ff0 |
| 135 | 0x05df | 0xb516 | 0xf9df | 0x000d | 0xfec7 | 0x0177 |
| 136 | 0x013e | 0xfdc1 | 0x09f0 | 0x00b2 | 0x0066 | 0x0028 |
| 137 | 0xc184 | 0x96ef | 0x1390 | 0x0cf8 | 0x02ae | 0x0487 |
| 138 | 0x649b | 0x6906 | 0x023e | 0xe8d6 | 0xf0b4 | 0x057f |
| 139 | 0xdc44 | 0xe20f | 0xf4c5 | 0xdf40 | 0xb719 | 0x6720 |
| 140 | 0xe2eb | 0xb988 | 0xb824 | 0x2262 | 0xf734 | 0xaa82 |
| 141 | 0x1eab | 0x2dfd | 0x6b5d | 0xcdd1 | 0xfa7e | 0x4c86 |
| 142 | 0x08c0 | 0x173b | 0x2bef | 0x3e6c | 0xe69d | 0x5ed8 |
| 143 | 0x54a9 | 0xb7ad | 0x262b | 0x1996 | 0xf556 | 0x014e |
| 144 | 0xefcb | 0x0628 | 0xd4fe | 0x0059 | 0xa093 | 0xe9b2 |
| 145 | 0x1e28 | 0x05c6 | 0x53a4 | 0x9e3f | 0xdf3f | 0x0009 |
| 146 | 0xf670 | 0x27ea | 0xce2c | 0xc131 | 0x0489 | 0xacdc |
| 147 | 0xddcb | 0xc7a3 | 0xa67a | 0xc624 | 0x0a45 | 0x3614 |
| 148 | 0xe3ac | 0x0b1b | 0xda59 | 0x0b42 | 0xc6df | 0x5fb1 |
| 149 | 0xfd5e | 0xe67e | 0x019e | 0xa4db | 0xaca1 | 0x01c6 |
| 150 | 0x0838 | 0xe758 | 0x2a87 | 0x46a7 | 0xfb51 | 0x00af |
| 151 | 0xfe13 | 0xfdce | 0xf54d | 0x0076 | 0xfbce | 0x005d |
| 152 | 0xd8e5 | 0xf015 | 0x9259 | 0x56a4 | 0x3ae5 | 0xfd84 |
| 153 | 0xede3 | 0xbfe8 | 0xdcd5 | 0xb03e | 0xd2a8 | 0xae3c |
| 154 | 0x12cf | 0x3e14 | 0x5eae | 0xcabe | 0xf3fe | 0xfbdd |
| 155 | 0xe5bc | 0x1202 | 0xb6ac | 0xc44d | 0xbed3 | 0x5db4 |
| 156 | 0x3bf5 | 0xfd5e | 0xf19e | 0x54af | 0x117b | 0xd0c8 |
| 157 | 0x1294 | 0x0a21 | 0x14ea | 0x1771 | 0x3ad7 | 0x677a |
| 158 | 0xa2f9 | 0xbc9d | 0x1b20 | 0x017a | 0x02b6 | 0x029e |
| 159 | 0x5b60 | 0xdd79 | 0xc687 | 0x1d78 | 0xfc94 | 0x2b50 |
| 160 | 0x0e38 | 0x0d08 | 0x5841 | 0xf259 | 0xf6e8 | 0xff8f |
| 161 | 0x011c | 0x1b02 | 0x0c19 | 0x27bb | 0x19ee | 0xb743 |
| 162 | 0x09a8 | 0x1758 | 0x2b2e | 0xd160 | 0xfda5 | 0xfd69 |
| 163 | 0x3f2f | 0x4039 | 0x336c | 0xf035 | 0x123b | 0x1d07 |
| 164 | 0x4b8a | 0x3cae | 0xe67b | 0x0691 | 0xed07 | 0x4298 |
| 165 | 0x4283 | 0x0214 | 0xb588 | 0xfa5f | 0xebf6 | 0x043d |
| 166 | 0xceb7 | 0xbb37 | 0x080e | 0x9d0c | 0x4a41 | 0xc107 |
| 167 | 0x2748 | 0xadf8 | 0xcabe | 0xf47b | 0x3c07 | 0x4dde |
| 168 | 0xfd78 | 0xf9bb | 0x273e | 0xf9c8 | 0x33f0 | 0x4d60 |
| 169 | 0xfbe2 | 0x29f8 | 0x021a | 0x616a | 0x259e | 0xdca4 |
| 170 | 0xd88d | 0x0be2 | 0x9e0c | 0xa20c | 0x3693 | 0x0064 |
| 171 | 0x1993 | 0x1afb | 0x1b77 | 0x286c | 0x5cdf | 0xba22 |
| 172 | 0xa6f7 | 0xf840 | 0xfa8f | 0xf2fe | 0x2433 | 0x37ed |

| | | | | | | |
|---|---|---|---|---|---|---|
| 173 | 0xc7f6 | 0xf081 | 0x0be2 | 0x3f7e | 0xbc69 | 0x25ae |
| 174 | 0xac6f | 0x5c4c | 0x4185 | 0x02cc | 0x0a67 | 0x0072 |
| 175 | 0xb5b8 | 0xf422 | 0x0626 | 0xff0b | 0x05b7 | 0xfce7 |
| 176 | 0x578a | 0x5b91 | 0xc6d3 | 0xfdee | 0x439e | 0x3531 |
| 177 | 0xd2c2 | 0x1eff | 0xc97e | 0x5ba9 | 0x9fcc | 0x67b6 |
| 178 | 0xfbeb | 0x0e5f | 0xf756 | 0x294c | 0x5207 | 0xf18a |
| 179 | 0xc367 | 0x00c5 | 0x414e | 0x9fe5 | 0x1351 | 0x0005 |
| 180 | 0x2a1d | 0x10ef | 0x68a6 | 0xdc9d | 0xc0e8 | 0xf4e8 |
| 181 | 0x3ecb | 0xa1dc | 0xf0a3 | 0xe54f | 0x3165 | 0xe48b |
| 182 | 0x0830 | 0x9c1c | 0xdf4e | 0x1a9e | 0x000b | 0x049a |
| 183 | 0xd1b8 | 0xfdb9 | 0xdd47 | 0xafc1 | 0xd719 | 0xfe84 |
| 184 | 0xf649 | 0x60c9 | 0xab79 | 0xb473 | 0x067c | 0xfd24 |
| 185 | 0x0909 | 0x356f | 0x0ff5 | 0x5fe5 | 0x6073 | 0xad45 |
| 186 | 0xf6c2 | 0xfe08 | 0xefde | 0xd6b6 | 0x5c74 | 0x07a9 |
| 187 | 0x4f9b | 0x4591 | 0xdade | 0x0e95 | 0xb5f6 | 0xe76c |
| 188 | 0xf0f0 | 0x41a2 | 0xfc5f | 0xb0aa | 0xbab5 | 0x1a8d |
| 189 | 0x308f | 0x17be | 0xd3f8 | 0xc78e | 0x1b01 | 0x5bb4 |
| 190 | 0x1dd4 | 0xf989 | 0x59e9 | 0x29df | 0xdf9c | 0x0346 |
| 191 | 0xde91 | 0xfb2d | 0xb950 | 0x0f39 | 0x3edd | 0x05d2 |
| 192 | 0xf1fe | 0x2054 | 0x3b3d | 0xf131 | 0xad63 | 0x06cd |
| 193 | 0xee6f | 0x54eb | 0x093e | 0xfeea | 0xed48 | 0x3cbd |
| 194 | 0xa5ae | 0xca74 | 0x1df4 | 0x3f68 | 0x5e38 | 0x3ab1 |
| 195 | 0xb1b5 | 0x3215 | 0xb140 | 0x4133 | 0xd279 | 0xc12f |
| 196 | 0xcec7 | 0x4f0f | 0x0da8 | 0xf60b | 0xe5a7 | 0xd1b6 |
| 197 | 0x1159 | 0x1e84 | 0x512f | 0x42b8 | 0x2d03 | 0xda55 |
| 198 | 0x60be | 0x212e | 0xa4fe | 0xf342 | 0x2b5d | 0xe430 |
| 199 | 0xd885 | 0xe239 | 0xa978 | 0xb881 | 0x6815 | 0x254e |
| 200 | 0x9c33 | 0x01dd | 0x1ec2 | 0xf9fe | 0x0463 | 0xff58 |
| 201 | 0x01d6 | 0x266a | 0xfea5 | 0x5d89 | 0xd773 | 0xdb05 |
| 202 | 0xf000 | 0xda1a | 0xe538 | 0xabd8 | 0x516d | 0x1c06 |
| 203 | 0x14fa | 0x2614 | 0xa32b | 0xfb5a | 0x0200 | 0xf9fe |
| 204 | 0xfc12 | 0xd8c2 | 0xce97 | 0x4b22 | 0xf902 | 0xfc86 |
| 205 | 0x3b04 | 0x5c44 | 0xc2e2 | 0xf626 | 0xfb4d | 0xfad3 |
| 206 | 0xe312 | 0xf5d3 | 0x0447 | 0xff09 | 0xfe27 | 0x00b1 |
| 207 | 0x1f99 | 0x0004 | 0x3088 | 0xa8f4 | 0x28a5 | 0xe1d0 |
| 208 | 0x56b4 | 0x2a17 | 0xec4d | 0x02b2 | 0x0216 | 0xff2c |
| 209 | 0xf3af | 0xfa76 | 0xbe3d | 0x47fa | 0x3dcd | 0x59ac |
| 210 | 0x1631 | 0xf74b | 0x0c7c | 0xf2aa | 0xaac7 | 0xc629 |
| 211 | 0x0013 | 0x0313 | 0x0408 | 0x00aa | 0xdf99 | 0xfd7b |
| 212 | 0xfc8e | 0xf6f1 | 0x961f | 0x01b0 | 0xeed8 | 0x05db |
| 213 | 0xfab6 | 0xd1d5 | 0xffb4 | 0xb064 | 0xd7cb | 0x2c40 |
| 214 | 0x00d3 | 0xed6f | 0xedbd | 0xe4eb | 0xcb1e | 0x388f |
| 215 | 0x179b | 0x148c | 0xfe35 | 0xfe32 | 0x008f | 0xffbf |
| 216 | 0xf5f4 | 0x1c58 | 0xf30b | 0x23fc | 0xa570 | 0xd8fa |

| | | | | | | |
|---|---|---|---|---|---|---|
| 217 | 0x9ece | 0xdac4 | 0x49ba | 0x17d5 | 0x097d | 0xc76e |
| 218 | 0x207a | 0x08e5 | 0x3770 | 0x0db8 | 0x6519 | 0x55f0 |
| 219 | 0x00d0 | 0x4efa | 0xfee7 | 0x9f36 | 0xffc1 | 0xfb61 |
| 220 | 0x0447 | 0xe86e | 0x0a92 | 0xaa51 | 0xf5a1 | 0x0233 |
| 221 | 0x0017 | 0xe8d6 | 0x00f3 | 0xdce3 | 0x14e1 | 0x504e |
| 222 | 0xc396 | 0x319b | 0x1040 | 0x2b4f | 0x508d | 0xd750 |
| 223 | 0x5203 | 0xffab | 0xdeec | 0x00c2 | 0x03eb | 0xdad5 |
| 224 | 0xb34b | 0xf2f9 | 0xc8ff | 0x0df6 | 0xa4ab | 0xfd65 |
| 225 | 0xf7e4 | 0x0da1 | 0xf388 | 0xb459 | 0x021b | 0xfa06 |
| 226 | 0x1cb8 | 0xc493 | 0x5844 | 0x4ba9 | 0x0413 | 0x40f3 |
| 227 | 0xf8b0 | 0xfe63 | 0x04d3 | 0xeb64 | 0xf222 | 0x558f |
| 228 | 0x1efb | 0xf828 | 0x4248 | 0xe571 | 0x72d1 | 0xf655 |
| 229 | 0xcaeb | 0x20c5 | 0xa3ac | 0xa9b5 | 0xc89e | 0xc827 |
| 230 | 0xd2c9 | 0xb186 | 0x3eb8 | 0xf8c8 | 0x3d69 | 0x1194 |
| 231 | 0x0f09 | 0xbf3b | 0x4ec1 | 0xad5d | 0x1e62 | 0x2e58 |
| 232 | 0xe66d | 0xfb07 | 0xb66b | 0xd42e | 0x2d74 | 0x0414 |
| 233 | 0x09e0 | 0xe5dd | 0xba03 | 0xd39e | 0xece2 | 0xfc10 |
| 234 | 0x04d9 | 0x10a4 | 0x090f | 0x17df | 0x0d9d | 0x4ef1 |
| 235 | 0x0bc6 | 0xf418 | 0x14c4 | 0xee45 | 0x515f | 0x21fe |
| 236 | 0xf902 | 0xc6a5 | 0x0116 | 0x3684 | 0xd8af | 0xd6cd |
| 237 | 0xa734 | 0xe0eb | 0xfb7e | 0x35fd | 0xfa34 | 0xfb21 |
| 238 | 0xe36b | 0xfd99 | 0x3326 | 0x49ef | 0x26a9 | 0x05ac |
| 239 | 0x09f8 | 0xf6de | 0x0d60 | 0xedea | 0x2b74 | 0xb380 |
| 240 | 0xd48b | 0xafb7 | 0xd599 | 0xd5e1 | 0xaee1 | 0x1ab1 |
| 241 | 0x03d8 | 0xc509 | 0x168f | 0x6225 | 0x1501 | 0xb2a9 |
| 242 | 0x0205 | 0x33d8 | 0xe2de | 0xf951 | 0x5084 | 0xe883 |
| 243 | 0xac57 | 0x33c3 | 0xaec5 | 0x3489 | 0x4381 | 0x3330 |
| 244 | 0xc23d | 0xc088 | 0x5a35 | 0xf03b | 0xdffd | 0x0367 |
| 245 | 0x0246 | 0x311b | 0xad77 | 0xc652 | 0xdc1d | 0x1635 |
| 246 | 0x10de | 0xf910 | 0x2ca1 | 0xba9d | 0xd93f | 0x0241 |
| 247 | 0x177d | 0x41be | 0x44f7 | 0x9b5a | 0xeed0 | 0xf222 |
| 248 | 0xca50 | 0xbf63 | 0x0e34 | 0xf2fe | 0xad9d | 0xc1f2 |
| 249 | 0x19a5 | 0xd475 | 0x21c9 | 0xccc6 | 0x5b31 | 0xcb03 |
| 250 | 0xf612 | 0xdcaa | 0xe27a | 0x7238 | 0x0e75 | 0xfe81 |
| 251 | 0xd68c | 0x61a3 | 0x0765 | 0xdfee | 0x51b8 | 0xc0ae |
| 252 | 0x149c | 0x4156 | 0x29a3 | 0x4de4 | 0xed41 | 0xb484 |
| 253 | 0xfdec | 0xdbac | 0x6cd0 | 0x1365 | 0xff0f | 0x0218 |
| 254 | 0xfd03 | 0xaf1e | 0xf2ac | 0x49b6 | 0x0acd | 0x058c |
| 255 | 0xf40d | 0x0a94 | 0xb5b2 | 0xfeb5 | 0x0dd1 | 0x0074 |

**Table 4.7** VQ Table for Hebap 7 (16-bit two's complement)

| index | val[index][0] (16-bit two's complement) | val[index][1] (16-bit two's complement) | val[index][2] (16-bit two's complement) | val[index][3] (16-bit two's complement) | val[index][4] (16-bit two's complement) | val[index][5] (16-bit two's complement) |
|---|---|---|---|---|---|---|
| 0 | 0xad4b | 0x5585 | 0x2896 | 0x354e | 0x29de | 0xdc27 |
| 1 | 0xa809 | 0xdfff | 0x4798 | 0xe61b | 0x63ae | 0xd5a0 |
| 2 | 0x1a90 | 0xca42 | 0xcc22 | 0x5792 | 0x394b | 0xae36 |
| 3 | 0x092b | 0x2914 | 0x0465 | 0xf281 | 0x15c1 | 0x6a00 |
| 4 | 0xe627 | 0x2e4b | 0xa034 | 0x5999 | 0x4f8a | 0xe87d |
| 5 | 0xaaf8 | 0x29b9 | 0xb361 | 0xc553 | 0xdee2 | 0xf7df |
| 6 | 0xf547 | 0x21ec | 0xece1 | 0x6c5b | 0x1d82 | 0xd147 |
| 7 | 0xfc04 | 0x099c | 0xfc46 | 0x1292 | 0xfd8d | 0xc010 |
| 8 | 0xb30a | 0x5a39 | 0x004b | 0xca8c | 0xf5ac | 0x083c |
| 9 | 0x0fd1 | 0xf4c8 | 0x16db | 0xee95 | 0x5686 | 0x3110 |
| 10 | 0xacc8 | 0xbd17 | 0xfd26 | 0x1cfb | 0xd276 | 0xd6e5 |
| 11 | 0x2c44 | 0x0700 | 0x682a | 0x5bde | 0xb397 | 0xfe15 |
| 12 | 0xba5d | 0xbe77 | 0xcada | 0xc7cb | 0xa9f3 | 0xf660 |
| 13 | 0x0443 | 0xe8b1 | 0xe0d9 | 0xbdaf | 0xb030 | 0xaa55 |
| 14 | 0x47d4 | 0xfbb1 | 0x078d | 0x341e | 0xbbc9 | 0x46c2 |
| 15 | 0x5876 | 0x43c1 | 0xd912 | 0x45ff | 0x4762 | 0x02ba |
| 16 | 0x05cc | 0x4f49 | 0xe986 | 0x986d | 0x134d | 0xa909 |
| 17 | 0xf5d5 | 0x11eb | 0xe92e | 0x4820 | 0x223f | 0xf5f8 |
| 18 | 0xf513 | 0xf9be | 0x54d1 | 0x0c1b | 0x9bad | 0x0c98 |
| 19 | 0xb5ad | 0x1255 | 0xec71 | 0x17ac | 0x07b4 | 0xc509 |
| 20 | 0xf773 | 0x252c | 0xfdee | 0x50bd | 0xedca | 0xdf93 |
| 21 | 0xa8cb | 0xdd49 | 0x09e1 | 0xd3a8 | 0x1564 | 0x03e6 |
| 22 | 0x5654 | 0xec44 | 0x0673 | 0xf59f | 0x1207 | 0x090f |
| 23 | 0x5177 | 0xf3fa | 0xf2fe | 0x1009 | 0x349e | 0x0bfd |
| 24 | 0x0055 | 0x4389 | 0x2818 | 0xc660 | 0x00d6 | 0x005a |
| 25 | 0x9903 | 0xb65f | 0xb468 | 0x4b2c | 0xd816 | 0x26b5 |
| 26 | 0xd9f5 | 0x5011 | 0xe64d | 0xe4b9 | 0x0b4b | 0xfd1e |
| 27 | 0x505f | 0xc20c | 0xa67f | 0x1ad6 | 0x004c | 0x0147 |
| 28 | 0x2228 | 0xcdb3 | 0xa65f | 0xf6bc | 0xb420 | 0xd9d5 |
| 29 | 0xcdaa | 0x3f37 | 0x525c | 0x0eed | 0x02ed | 0xca20 |
| 30 | 0xc185 | 0x47df | 0x0957 | 0xbb03 | 0x4c1c | 0xe87e |
| 31 | 0x058f | 0x2dd6 | 0x0fd3 | 0x4b5a | 0x1ac9 | 0xb31f |
| 32 | 0xebb0 | 0x2626 | 0x4746 | 0x099f | 0x494c | 0xed0c |
| 33 | 0xfdab | 0x4c2a | 0x052b | 0xdc78 | 0xfecc | 0xfbb0 |
| 34 | 0xf3e5 | 0x9b7d | 0xc2cf | 0x62f4 | 0x121a | 0x0a4b |
| 35 | 0x4ca7 | 0xf6b0 | 0xe117 | 0x2e14 | 0xdb8b | 0xc8fc |
| 36 | 0x0a52 | 0x6755 | 0xad9d | 0xd78e | 0xf963 | 0xf951 |
| 37 | 0x560f | 0x5479 | 0x2d3c | 0xa67d | 0xefd3 | 0x0081 |
| 38 | 0xe816 | 0x0dd6 | 0x0393 | 0xfefb | 0xffef | 0xfe81 |
| 39 | 0x06a0 | 0x1a30 | 0xfa6f | 0x5166 | 0x0359 | 0xeec0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 40 | 0x058f | 0xc450 | 0xde9a | 0xda3d | 0x145a | 0x1637 |
| 41 | 0xee58 | 0xfd9b | 0xd25d | 0x15f2 | 0x1086 | 0x026b |
| 42 | 0x03a9 | 0xec9d | 0xc8ea | 0xbd30 | 0xe506 | 0xe8c0 |
| 43 | 0xc524 | 0xfe1f | 0xe3bb | 0xc5d2 | 0x49bc | 0x54a9 |
| 44 | 0x9bc6 | 0x0b5e | 0x0477 | 0xfeb9 | 0xfe36 | 0xfc1d |
| 45 | 0xda48 | 0xfccd | 0x9ebc | 0x0af4 | 0x4f01 | 0x043b |
| 46 | 0xfba9 | 0xf19e | 0xf904 | 0xb3dc | 0x03c6 | 0x0335 |
| 47 | 0x1c7d | 0xab01 | 0x2a26 | 0xe46d | 0xa503 | 0xf945 |
| 48 | 0xfee6 | 0xd4ab | 0x00aa | 0xae2a | 0x8f02 | 0x3147 |
| 49 | 0x4612 | 0x0e81 | 0xf9e5 | 0x0375 | 0x0005 | 0x0234 |
| 50 | 0x17e4 | 0x58a8 | 0x08c2 | 0xe4d9 | 0x26f7 | 0xe80c |
| 51 | 0x10f2 | 0x68b8 | 0xf187 | 0x07b8 | 0xfbc9 | 0xf5f6 |
| 52 | 0xfd6b | 0xe123 | 0xf594 | 0xc4a6 | 0x453a | 0x1117 |
| 53 | 0xefb2 | 0xd4d3 | 0x02cd | 0xa816 | 0x061a | 0x2fdc |
| 54 | 0xe6fb | 0x479e | 0x17d7 | 0x1b47 | 0x1744 | 0x4713 |
| 55 | 0x267b | 0x14fa | 0x5c34 | 0xe533 | 0xe657 | 0xffc2 |
| 56 | 0x55cc | 0x342f | 0xfd55 | 0x0ec9 | 0x0878 | 0x00d1 |
| 57 | 0xf20f | 0xfb99 | 0xb2f4 | 0xf9f8 | 0x051c | 0xfcdd |
| 58 | 0xf3f5 | 0x3eb1 | 0xca21 | 0xf3fb | 0x10c6 | 0x5ca1 |
| 59 | 0xd8f1 | 0x26d7 | 0xc200 | 0x3286 | 0xa3b1 | 0x54c3 |
| 60 | 0x25fa | 0x5935 | 0x2fa0 | 0x3af3 | 0x159d | 0x12e5 |
| 61 | 0x08c3 | 0x0833 | 0x04db | 0x0ff9 | 0x128c | 0x329c |
| 62 | 0x0fa7 | 0xf65c | 0x0d19 | 0xf3ec | 0x228b | 0x4280 |
| 63 | 0x10ea | 0x17ef | 0x15ad | 0x2421 | 0x2bda | 0x6fb0 |
| 64 | 0xda8e | 0xdd87 | 0x00ec | 0x03f1 | 0x01c7 | 0xfc3c |
| 65 | 0x1aad | 0x4b5a | 0xfc06 | 0x00c8 | 0x071d | 0x0242 |
| 66 | 0x144c | 0x03bd | 0x2884 | 0x0d02 | 0xce00 | 0xff81 |
| 67 | 0xf432 | 0xdfff | 0xc723 | 0x562d | 0x1720 | 0x041d |
| 68 | 0x2ae6 | 0x6556 | 0xa01e | 0xa512 | 0xd17f | 0xe57b |
| 69 | 0x588b | 0xd4fe | 0x1668 | 0x0a07 | 0x5c99 | 0xd7f3 |
| 70 | 0xf2f1 | 0xef77 | 0xeaae | 0x50bb | 0xd5a5 | 0xf1eb |
| 71 | 0xefdd | 0xf1e4 | 0x11df | 0xfcc3 | 0xfea2 | 0xfcb1 |
| 72 | 0xf319 | 0x0d7b | 0xe31a | 0xd2ac | 0x0bcf | 0x01c7 |
| 73 | 0x0c80 | 0xdab5 | 0x0c82 | 0xa693 | 0x2bb0 | 0x989e |
| 74 | 0xc8f3 | 0xefeb | 0x3c16 | 0x37d7 | 0xd55d | 0xb067 |
| 75 | 0x0edf | 0xd4f8 | 0x5624 | 0x3822 | 0xc420 | 0xe1cb |
| 76 | 0xe76d | 0xbac9 | 0xf9e8 | 0x2f10 | 0xb2a3 | 0xfe45 |
| 77 | 0xe7fd | 0xef76 | 0xff60 | 0x20ab | 0x586e | 0x2e87 |
| 78 | 0x4afd | 0x0497 | 0x1cfe | 0xd96d | 0xefd8 | 0x1260 |
| 79 | 0xffb8 | 0xe21c | 0xff90 | 0xd14d | 0xf362 | 0x6a27 |
| 80 | 0x0cca | 0x174b | 0x1d4d | 0xbd85 | 0x0362 | 0x9c94 |
| 81 | 0x02e1 | 0x0745 | 0x0729 | 0x07e6 | 0x0950 | 0x1293 |
| 82 | 0xeb9f | 0x1d58 | 0x0cfb | 0x0a9b | 0x0bf9 | 0xf9ba |
| 83 | 0x1097 | 0x0235 | 0x15fd | 0x09c1 | 0x4663 | 0xecc8 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 84 | 0xf4ef | 0xba00 | 0xe082 | 0x3d7a | 0xfc06 | 0x0858 |
| 85 | 0x0bea | 0xb3e5 | 0x4222 | 0x748b | 0xd812 | 0x3b31 |
| 86 | 0xd3ae | 0x0076 | 0x9b3c | 0xca3f | 0x3bd8 | 0xfe2c |
| 87 | 0xed28 | 0x1360 | 0xef59 | 0x0627 | 0xd69f | 0x4c69 |
| 88 | 0xdff5 | 0xfa7f | 0xfd05 | 0xfb8d | 0xfda1 | 0x0580 |
| 89 | 0xf765 | 0xd369 | 0x07e5 | 0xe70c | 0xf5d8 | 0x02c7 |
| 90 | 0xfe63 | 0xf631 | 0xff28 | 0xf241 | 0x9195 | 0x06b7 |
| 91 | 0xc792 | 0x429a | 0x365d | 0x34bb | 0x9b5e | 0xc107 |
| 92 | 0x4b1c | 0x1cad | 0xcff3 | 0x02aa | 0xf131 | 0xff39 |
| 93 | 0xef9f | 0x510a | 0xc2dd | 0x2c55 | 0x16e4 | 0xfcc8 |
| 94 | 0xac0e | 0xf226 | 0xfffd | 0xf957 | 0xf089 | 0x23fd |
| 95 | 0x3c55 | 0xf8ac | 0x07dc | 0xb355 | 0x3f64 | 0xed13 |
| 96 | 0xf4cd | 0xf16b | 0xe346 | 0xff51 | 0xb169 | 0x2ba6 |
| 97 | 0xf20d | 0x9ff5 | 0x4cf4 | 0x19fe | 0x03d3 | 0xfd72 |
| 98 | 0x553c | 0xe2fa | 0xe611 | 0xd5f1 | 0xdf56 | 0x3cb7 |
| 99 | 0x39eb | 0x4639 | 0xe3dc | 0xf2af | 0x0772 | 0xbc78 |
| 100 | 0x0dc5 | 0xf095 | 0xfa79 | 0xf512 | 0x44f0 | 0x0822 |
| 101 | 0xe64c | 0xc469 | 0xba07 | 0x0539 | 0x3bea | 0x52a6 |
| 102 | 0x1842 | 0x25e2 | 0x3b33 | 0x9fa6 | 0xa815 | 0xf061 |
| 103 | 0xf934 | 0xfdaf | 0x0447 | 0xe19d | 0x61e2 | 0x15e1 |
| 104 | 0x53a7 | 0xfe50 | 0xf986 | 0xe50e | 0xfa62 | 0xc78a |
| 105 | 0xe4e1 | 0x02bc | 0xd095 | 0xfd17 | 0xa185 | 0x57c2 |
| 106 | 0x188f | 0x0cd3 | 0x2afe | 0x0f04 | 0x4af0 | 0x39bd |
| 107 | 0xa81a | 0x3baa | 0x1543 | 0xf508 | 0xfc36 | 0xf2f1 |
| 108 | 0x0cb9 | 0xf184 | 0x1288 | 0xdf93 | 0x591e | 0xd820 |
| 109 | 0x5f1a | 0xae16 | 0x4d86 | 0x03db | 0xd14a | 0xe77b |
| 110 | 0x0f42 | 0xb30b | 0x3304 | 0xf9b7 | 0x48d1 | 0x1d2a |
| 111 | 0x98d7 | 0xa7eb | 0x3fb1 | 0x07de | 0x2adf | 0x4670 |
| 112 | 0xe481 | 0x122f | 0xc61e | 0x4933 | 0x3dad | 0x0510 |
| 113 | 0x245e | 0xf96f | 0x394b | 0xf302 | 0x67a7 | 0xd1b3 |
| 114 | 0x1660 | 0x171d | 0x3458 | 0x2724 | 0xf744 | 0x9f80 |
| 115 | 0x06cd | 0xe5b9 | 0x3197 | 0xaa07 | 0x0ff0 | 0x154a |
| 116 | 0xf5c3 | 0x24b1 | 0x5297 | 0x9aae | 0xf3a6 | 0xf61f |
| 117 | 0x50c0 | 0x49ce | 0xc98d | 0x1b4e | 0xdfbc | 0x3dc3 |
| 118 | 0xa2f6 | 0x2baf | 0xcab9 | 0x2e5c | 0x3ead | 0x0a46 |
| 119 | 0x47b9 | 0xd814 | 0x033d | 0x0358 | 0xfc0e | 0x009d |
| 120 | 0x3840 | 0xedba | 0x1421 | 0xcc16 | 0x94d6 | 0xd4ec |
| 121 | 0x546d | 0x2bf8 | 0x442d | 0x1db4 | 0x334a | 0xfe1c |
| 122 | 0x0007 | 0x04d4 | 0x023d | 0x1076 | 0x15c8 | 0xf3f7 |
| 123 | 0x0394 | 0xdc7c | 0x0505 | 0xdd02 | 0x04a1 | 0x8fe5 |
| 124 | 0x5453 | 0x5c8f | 0x4aac | 0xf4bb | 0xc836 | 0xdf0a |
| 125 | 0x5b76 | 0xe7ef | 0x32b2 | 0x0bf5 | 0xdb79 | 0x08bc |
| 126 | 0xf402 | 0xe350 | 0xb154 | 0x169c | 0x0246 | 0xfdd9 |
| 127 | 0xf067 | 0x013b | 0xe1a3 | 0x2020 | 0x924e | 0xcf4f |

| | | | | | | |
|---|---|---|---|---|---|---|
| 128 | 0x35c6 | 0xc403 | 0x4b05 | 0xaf70 | 0x32f3 | 0xb4d1 |
| 129 | 0x0ec1 | 0xff4f | 0x1f5d | 0xfc17 | 0x4594 | 0x142a |
| 130 | 0xe374 | 0xef19 | 0xb950 | 0xfd94 | 0xfaba | 0x3a54 |
| 131 | 0x39a4 | 0xfb3b | 0xcded | 0xc5b6 | 0xfddd | 0x69f5 |
| 132 | 0x08ba | 0x06ac | 0x0acc | 0x1528 | 0x1f32 | 0x9db5 |
| 133 | 0x0b39 | 0x0e34 | 0x0f98 | 0x14e0 | 0x279e | 0x530b |
| 134 | 0x0486 | 0x1503 | 0x01fc | 0xd6ee | 0x0122 | 0xf9b1 |
| 135 | 0x045a | 0x60d5 | 0x40bf | 0x9db0 | 0xfed6 | 0xf4f0 |
| 136 | 0xfbad | 0xe800 | 0xf882 | 0xe191 | 0xf465 | 0xa514 |
| 137 | 0x0fb0 | 0x2a29 | 0x43a5 | 0xef0a | 0xae0a | 0xf2c9 |
| 138 | 0xee72 | 0xff31 | 0xd921 | 0xf209 | 0x1f0b | 0x0482 |
| 139 | 0xe268 | 0x1fb5 | 0xc921 | 0x4256 | 0x98a7 | 0x946c |
| 140 | 0xc4c4 | 0x3ee0 | 0xbe34 | 0xdd4a | 0xa358 | 0x3e22 |
| 141 | 0x615a | 0x1630 | 0xf8ae | 0x01a4 | 0x0084 | 0x0075 |
| 142 | 0xfe06 | 0xb492 | 0xff3a | 0x019c | 0xfec9 | 0x02f0 |
| 143 | 0xf88e | 0x0f8d | 0xe1f8 | 0x40b6 | 0xb4a5 | 0xc67e |
| 144 | 0xfe71 | 0xfd27 | 0xf121 | 0xef9c | 0xcf95 | 0x1dd7 |
| 145 | 0x0d28 | 0x091a | 0x2384 | 0x5c86 | 0xd7ce | 0xf957 |
| 146 | 0xf3b4 | 0x0a24 | 0xe0ce | 0x390a | 0xed39 | 0x40f3 |
| 147 | 0x1f79 | 0x05c9 | 0x0031 | 0x4335 | 0x6125 | 0x1d32 |
| 148 | 0xb498 | 0xfdff | 0x2e81 | 0x092a | 0x15d4 | 0x0d25 |
| 149 | 0xec39 | 0xaacc | 0x2c6a | 0x2a90 | 0x1311 | 0x0105 |
| 150 | 0x12ba | 0x5061 | 0x13f5 | 0xe877 | 0xe08f | 0xfa0f |
| 151 | 0x1fbd | 0xc65c | 0x509f | 0xc5ba | 0x5d85 | 0xf1be |
| 152 | 0x30a3 | 0x0565 | 0x0e1d | 0x21ef | 0xa23e | 0x12f0 |
| 153 | 0x1a46 | 0x2993 | 0x2766 | 0x6281 | 0x9db9 | 0xfbd7 |
| 154 | 0x19a1 | 0x3699 | 0x0b5f | 0x54e9 | 0x4051 | 0x9a3e |
| 155 | 0xf910 | 0x0a0f | 0xb36a | 0xbe60 | 0x0bd8 | 0x1a17 |
| 156 | 0x3aa4 | 0xba0a | 0xdf0a | 0xabce | 0x9619 | 0x2e20 |
| 157 | 0x0d78 | 0xfc64 | 0xc1d7 | 0xfb91 | 0x1406 | 0xaf7b |
| 158 | 0x1e28 | 0x08b2 | 0x4437 | 0x153a | 0x710e | 0x4490 |
| 159 | 0x04de | 0x3cfe | 0xd221 | 0x602a | 0xbb7d | 0x0cc8 |
| 160 | 0x0c8f | 0x461e | 0x0adf | 0xfd2e | 0xa770 | 0x175b |
| 161 | 0xe9d2 | 0xf390 | 0x9a19 | 0x65b2 | 0x19b7 | 0x0ce6 |
| 162 | 0x4f56 | 0xf21d | 0xf565 | 0xfe44 | 0xfa31 | 0x05f6 |
| 163 | 0xaf60 | 0xaa2e | 0xd051 | 0x9b3f | 0x229f | 0xfbf4 |
| 164 | 0x45e0 | 0x023a | 0xc11a | 0x2089 | 0xf607 | 0x3bab |
| 165 | 0xf58b | 0x26de | 0xf8a9 | 0x405d | 0xce26 | 0x8eb1 |
| 166 | 0xff88 | 0xf753 | 0x00db | 0x0061 | 0x016d | 0x0023 |
| 167 | 0x04f6 | 0xfd32 | 0x05c8 | 0xf57f | 0x078a | 0xe299 |
| 168 | 0x0768 | 0x222e | 0x0772 | 0x473b | 0xce6c | 0xe7e2 |
| 169 | 0xf16b | 0x3591 | 0xd966 | 0xc1a8 | 0xfaa0 | 0xe416 |
| 170 | 0xd698 | 0x2130 | 0x3e5f | 0xdda8 | 0x1d6c | 0x4fd7 |
| 171 | 0x0be1 | 0xcb6f | 0x0408 | 0x96b8 | 0x169b | 0x6198 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 172 | 0xee12 | 0xdfe4 | 0xdb96 | 0xe820 | 0xbca5 | 0x6491 |
| 173 | 0xba70 | 0x1b3a | 0x0ea8 | 0x0272 | 0xff8e | 0x0882 |
| 174 | 0x1161 | 0xed02 | 0x1b8e | 0xeae4 | 0x1282 | 0xf4f5 |
| 175 | 0x133a | 0xfd75 | 0x49fb | 0xd976 | 0x0350 | 0x075e |
| 176 | 0xfeb0 | 0xeade | 0x1c42 | 0x4fdc | 0xda91 | 0xfda8 |
| 177 | 0x030d | 0xb3ee | 0xce98 | 0x19ea | 0x0586 | 0x01c2 |
| 178 | 0xf2b9 | 0xbe7e | 0x2b63 | 0x3390 | 0xea86 | 0x549f |
| 179 | 0xf33f | 0x12fb | 0xe8b7 | 0x1d6a | 0xd5ab | 0x6db6 |
| 180 | 0x286e | 0xcd9b | 0x6463 | 0x6428 | 0xfd81 | 0x015f |
| 181 | 0x048b | 0x494b | 0xeaa6 | 0xc511 | 0xff6f | 0xfa9f |
| 182 | 0xc773 | 0x6a5d | 0x8569 | 0x8073 | 0x53bf | 0xf4b2 |
| 183 | 0x3c3c | 0x4987 | 0x5670 | 0x4bc6 | 0x5837 | 0x3513 |
| 184 | 0xd64e | 0x29d6 | 0x13e1 | 0xed6c | 0x038d | 0xaee8 |
| 185 | 0xcd6c | 0xaf4c | 0x1cf2 | 0x0aa2 | 0x0d63 | 0x2d41 |
| 186 | 0xfbaf | 0x47c6 | 0x4d13 | 0xda4e | 0x57aa | 0x4cb2 |
| 187 | 0xfed8 | 0xe572 | 0xc6ab | 0x5463 | 0x4d54 | 0x5397 |
| 188 | 0xb46a | 0xe2b3 | 0x6366 | 0x3358 | 0x218c | 0x9d2e |
| 189 | 0x0c14 | 0xd686 | 0x51a0 | 0x2421 | 0xf302 | 0x0704 |
| 190 | 0xfcd5 | 0x05a9 | 0x0c22 | 0x128c | 0x2f29 | 0xc84a |
| 191 | 0xaf10 | 0x37c3 | 0xef14 | 0x9b12 | 0xe96b | 0xad63 |
| 192 | 0xebf4 | 0x293a | 0xc93c | 0xa97a | 0x0b18 | 0xfdd6 |
| 193 | 0x63bd | 0x44f0 | 0x3a26 | 0xadae | 0x099b | 0x6236 |
| 194 | 0xdb66 | 0xf904 | 0xcdc2 | 0xe912 | 0x9b2d | 0xd4f1 |
| 195 | 0x1a2a | 0x0333 | 0x2849 | 0x00a6 | 0x6bbd | 0x020b |
| 196 | 0x0065 | 0xb444 | 0x0d55 | 0x25a6 | 0x0040 | 0x0326 |
| 197 | 0xf54a | 0xb9f5 | 0xf5f0 | 0x5922 | 0x2169 | 0x0466 |
| 198 | 0x0b9c | 0x3b63 | 0x0700 | 0x635a | 0xe9a0 | 0xbc8f |
| 199 | 0xfa75 | 0x0644 | 0x112e | 0x2cbc | 0x06c3 | 0x5ceb |
| 200 | 0xebf0 | 0x1211 | 0xd663 | 0x6d4d | 0x26a9 | 0xf632 |
| 201 | 0xd6e0 | 0x927f | 0x0bb7 | 0xfa06 | 0xfcc0 | 0xfcc2 |
| 202 | 0xd483 | 0xcf21 | 0x56be | 0xe3b5 | 0xa3e6 | 0xab3e |
| 203 | 0x4227 | 0xaa7c | 0x0745 | 0xda7a | 0x24d8 | 0x4a52 |
| 204 | 0x2825 | 0x252c | 0x68bf | 0x07da | 0xecb1 | 0xdc88 |
| 205 | 0x15ab | 0xf75e | 0x37be | 0xc43c | 0xb48c | 0x071e |
| 206 | 0xed0e | 0xfcf1 | 0xdd01 | 0xf3fc | 0xb1a8 | 0xf383 |
| 207 | 0x2028 | 0xf516 | 0xbaa8 | 0x33fc | 0x0c9d | 0xfc21 |
| 208 | 0xd033 | 0xe64b | 0x284b | 0xdab0 | 0x08d4 | 0xaf58 |
| 209 | 0xe4a8 | 0x1599 | 0xe27f | 0xe2be | 0xd79a | 0xd7e6 |
| 210 | 0x0e39 | 0x4c17 | 0xe8ac | 0xb567 | 0xb776 | 0x3205 |
| 211 | 0x0503 | 0xefbc | 0x1066 | 0x90c7 | 0xf63e | 0x074a |
| 212 | 0x3eaf | 0x68ca | 0xcd03 | 0xe754 | 0x03d9 | 0xf9c3 |
| 213 | 0xfe6d | 0x3570 | 0x1939 | 0x61ee | 0x69f4 | 0xaf1a |
| 214 | 0xb96a | 0xf902 | 0x9e66 | 0x1741 | 0xfc46 | 0x67e8 |
| 215 | 0xa160 | 0xc3e9 | 0x60d4 | 0x07a1 | 0xfb90 | 0x00bb |

| | | | | | | |
|---|---|---|---|---|---|---|
| 216 | 0xf70f | 0x30d9 | 0xaefe | 0xfc78 | 0x4794 | 0x530a |
| 217 | 0x0a62 | 0xe804 | 0x3f33 | 0x5704 | 0xfdd4 | 0x086a |
| 218 | 0xe839 | 0x367e | 0x9bae | 0x93bf | 0x0fd1 | 0xed45 |
| 219 | 0xac34 | 0x6769 | 0x4beb | 0xdc5c | 0x037f | 0x012f |
| 220 | 0xa948 | 0x99bf | 0xe876 | 0x6099 | 0xa672 | 0xdcba |
| 221 | 0xc83c | 0xc192 | 0x5cb4 | 0xa6bd | 0x2434 | 0xf0ff |
| 222 | 0x732a | 0x55a3 | 0xe7bb | 0x068f | 0xf7f5 | 0xfba0 |
| 223 | 0xfe4d | 0x264a | 0xf0cd | 0x3047 | 0xef40 | 0xb5e5 |
| 224 | 0x4d38 | 0xffaa | 0x09a3 | 0x07c6 | 0xfc03 | 0xeb16 |
| 225 | 0x51fa | 0xddb1 | 0xeb2f | 0xa3f6 | 0xed86 | 0x0a71 |
| 226 | 0xec19 | 0x15e5 | 0xedeb | 0x4ace | 0x65b5 | 0xd01d |
| 227 | 0x03cc | 0x1aca | 0x11c7 | 0x6d2d | 0xf047 | 0xf720 |
| 228 | 0x17bb | 0xf344 | 0xec83 | 0xfe8b | 0xf9dd | 0xf16e |
| 229 | 0xe3a8 | 0xcd40 | 0xdd8c | 0xec0b | 0x5a0e | 0x13be |
| 230 | 0x0398 | 0x0a37 | 0x1ee8 | 0xe347 | 0xecd7 | 0x4eda |
| 231 | 0xff06 | 0x154e | 0x0c44 | 0x1b10 | 0xb6dd | 0xf7fd |
| 232 | 0xd7c5 | 0xeeec | 0x4c98 | 0x130f | 0xfd6b | 0xf8a3 |
| 233 | 0x39e0 | 0xde65 | 0xb299 | 0x17f7 | 0xad26 | 0x371c |
| 234 | 0xd157 | 0xf751 | 0x139a | 0x2e74 | 0x58d5 | 0x0196 |
| 235 | 0xcc80 | 0xf5cb | 0xcc38 | 0xa85f | 0xcf3e | 0xdf44 |
| 236 | 0x42aa | 0x62b3 | 0xf71f | 0x13c0 | 0xfeaa | 0x0091 |
| 237 | 0x20d1 | 0xbaed | 0x4aa8 | 0x2977 | 0xb403 | 0x42bb |
| 238 | 0x5155 | 0xe9bc | 0x300a | 0x9c02 | 0x2897 | 0x1e0c |
| 239 | 0x11c6 | 0x3da3 | 0x43ba | 0xb44d | 0xed60 | 0x04b6 |
| 240 | 0xe1d5 | 0x2a54 | 0x95e4 | 0xd351 | 0x1ab3 | 0xf910 |
| 241 | 0x09ee | 0x0c7f | 0x115a | 0x4469 | 0xf181 | 0xfc6e |
| 242 | 0x51e0 | 0xbe7a | 0xe94a | 0x2b4f | 0xffba | 0x59b1 |
| 243 | 0x0ce9 | 0x0b67 | 0x1870 | 0xed40 | 0xae1a | 0xf362 |
| 244 | 0x1724 | 0xbf5d | 0x0887 | 0x0aad | 0x0d76 | 0xa4f6 |
| 245 | 0xe853 | 0xff3e | 0xc9e4 | 0xd525 | 0x4c20 | 0x0405 |
| 246 | 0x1173 | 0xe8b4 | 0xb5c4 | 0x05ef | 0xfe99 | 0x0357 |
| 247 | 0xf9d3 | 0xe249 | 0x5636 | 0xd2c4 | 0xd8d0 | 0x42ce |
| 248 | 0xcf84 | 0x09f9 | 0x10e4 | 0x57e4 | 0x1677 | 0x2f8a |
| 249 | 0x9dd9 | 0x464c | 0xe710 | 0x049c | 0x049e | 0x2596 |
| 250 | 0x5ba6 | 0xdee9 | 0xeed8 | 0xf593 | 0x1dd6 | 0xbe3d |
| 251 | 0xea79 | 0xf4b9 | 0xd5fb | 0xae6d | 0x1c4e | 0x041d |
| 252 | 0x0a8f | 0xaf86 | 0xe27e | 0x1d5c | 0xe1c4 | 0x16ec |
| 253 | 0x50be | 0x558d | 0x01c9 | 0x3a79 | 0xbb07 | 0xd16f |
| 254 | 0x0e13 | 0xf9c5 | 0xf77f | 0xff63 | 0xffd5 | 0x025d |
| 255 | 0x09d1 | 0x22fa | 0x291f | 0x581f | 0xc11c | 0xc157 |
| 256 | 0x1772 | 0x1357 | 0x1a8b | 0xed02 | 0xa880 | 0x49a1 |
| 257 | 0x1da6 | 0xf963 | 0x9f90 | 0xf2b4 | 0x3759 | 0x04be |
| 258 | 0xeed2 | 0xe5f9 | 0xe52a | 0xd89d | 0x9fec | 0x2425 |
| 259 | 0x28e4 | 0x4557 | 0xe1bc | 0x0093 | 0xe756 | 0x1143 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 260 | 0x3f3b | 0xbf53 | 0xefe9 | 0x10ce | 0x1dc9 | 0x1521 |
| 261 | 0x0ce7 | 0x0aaf | 0x1d22 | 0xb242 | 0xf732 | 0xf18a |
| 262 | 0xf7e3 | 0x5469 | 0x3a16 | 0x3101 | 0xe83f | 0xf91c |
| 263 | 0x1246 | 0x2ddc | 0x0b2b | 0x1b29 | 0x077f | 0xf0e1 |
| 264 | 0x0dc2 | 0xaaa3 | 0xf65b | 0xd72b | 0x49cd | 0xd60a |
| 265 | 0x0eaf | 0xd831 | 0xecfe | 0xf59d | 0xba59 | 0xfb26 |
| 266 | 0x3a8f | 0x2487 | 0x2e5e | 0xf9db | 0xed10 | 0x5815 |
| 267 | 0x2525 | 0x95f0 | 0x29ee | 0x5173 | 0x99b7 | 0xba2a |
| 268 | 0xe3fe | 0xfa90 | 0xb3d9 | 0x31ca | 0x2006 | 0xf83c |
| 269 | 0x075b | 0x6dfe | 0xfcb2 | 0xe3bd | 0x00f9 | 0x00e9 |
| 270 | 0xe3e0 | 0x029d | 0xfe8d | 0xf47c | 0x5ac2 | 0xe9fd |
| 271 | 0x0c45 | 0x0120 | 0x0c97 | 0xfb16 | 0xff9e | 0x9429 |
| 272 | 0x43dd | 0xa53d | 0x13f6 | 0xd441 | 0xf5f2 | 0xd321 |
| 273 | 0xecc0 | 0x05ee | 0xeab0 | 0x029e | 0xb89a | 0x079f |
| 274 | 0x285e | 0xb267 | 0xedd7 | 0x0169 | 0xff60 | 0xfc65 |
| 275 | 0x492c | 0x37b8 | 0xf3ad | 0xe2c3 | 0xf300 | 0x1747 |
| 276 | 0xf1e2 | 0x5255 | 0x1c6c | 0x0dd0 | 0x1fb9 | 0xfa08 |
| 277 | 0xdf1a | 0x01f4 | 0xb512 | 0x49f1 | 0x6718 | 0xfbf1 |
| 278 | 0x3d17 | 0x6444 | 0x20b7 | 0x076f | 0x0799 | 0xd135 |
| 279 | 0xf564 | 0x0d3d | 0x68e2 | 0xee15 | 0x070b | 0x0016 |
| 280 | 0x0499 | 0xfd71 | 0x04d1 | 0xf7b0 | 0x1ea4 | 0x06e7 |
| 281 | 0xfd07 | 0x2011 | 0xb4a6 | 0xee0f | 0x0783 | 0xfea9 |
| 282 | 0xfd4f | 0xf236 | 0xf33d | 0xf124 | 0xf53f | 0x4886 |
| 283 | 0xf7c2 | 0x07aa | 0xfab7 | 0x4103 | 0x0acd | 0xa5c2 |
| 284 | 0xfe4f | 0x1329 | 0x012e | 0x32d8 | 0x3e3d | 0xe8ef |
| 285 | 0x0c83 | 0x101e | 0x2bad | 0xea88 | 0xf61f | 0xfb78 |
| 286 | 0xfbbd | 0xe6bb | 0xfa79 | 0x1632 | 0xfef4 | 0x0247 |
| 287 | 0xdb43 | 0xb38c | 0x1848 | 0x067a | 0x03e1 | 0xffb5 |
| 288 | 0xf961 | 0xee68 | 0xf70f | 0xf008 | 0xe664 | 0xbf3f |
| 289 | 0x1298 | 0xfc84 | 0xd56a | 0x1974 | 0x5e87 | 0xe885 |
| 290 | 0xff03 | 0x03e8 | 0x003f | 0xffaf | 0xff8d | 0xfe82 |
| 291 | 0xfacb | 0x5ea0 | 0xfd46 | 0xedc5 | 0xf50f | 0xb538 |
| 292 | 0xfc94 | 0x8f3e | 0xaa8f | 0x3185 | 0xe738 | 0x0ca3 |
| 293 | 0x41cf | 0x5299 | 0x99c4 | 0xf391 | 0xfe74 | 0x00e6 |
| 294 | 0x4778 | 0xe192 | 0xcdc7 | 0xfd59 | 0xfa3f | 0x0005 |
| 295 | 0xd708 | 0x2ca5 | 0x64cd | 0xfb9e | 0x0579 | 0xfe4a |
| 296 | 0x0ec6 | 0xe2fb | 0x6860 | 0x449f | 0x4b39 | 0x30fe |
| 297 | 0x18bc | 0xfd16 | 0x31f5 | 0x2464 | 0xa7f2 | 0xeb16 |
| 298 | 0x0d5a | 0xa738 | 0x6962 | 0x477f | 0x0434 | 0x03bc |
| 299 | 0x954d | 0xf454 | 0x0398 | 0x00eb | 0x08b9 | 0x0051 |
| 300 | 0x1837 | 0x14b0 | 0x3edd | 0x39b0 | 0xdf13 | 0xfba8 |
| 301 | 0xe6e0 | 0x4b2c | 0x26c1 | 0xf34b | 0x04fe | 0xfc46 |
| 302 | 0x5e95 | 0x0801 | 0xa66d | 0x0a19 | 0xf696 | 0xef88 |
| 303 | 0x2446 | 0x37ca | 0xb2e9 | 0xf06f | 0xf6d8 | 0x0404 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 304 | 0xb160 | 0x4649 | 0xdb0e | 0x59e4 | 0xbda9 | 0x21b1 |
| 305 | 0xe510 | 0xaf06 | 0x0eb2 | 0x4407 | 0x5745 | 0x4a50 |
| 306 | 0x034a | 0x5e75 | 0x61e6 | 0xe931 | 0xffb2 | 0x03a9 |
| 307 | 0xfd93 | 0x4d0a | 0xa174 | 0xf856 | 0xc5fa | 0xffc8 |
| 308 | 0x58ee | 0xec01 | 0x43d5 | 0x5d3c | 0xb3e8 | 0xe662 |
| 309 | 0xf792 | 0x4452 | 0xac45 | 0x0d0c | 0xcded | 0xb0b9 |
| 310 | 0xda6b | 0x43ad | 0x02cb | 0x08d9 | 0xefe5 | 0xfe14 |
| 311 | 0x23c4 | 0x3293 | 0x6aa7 | 0xad49 | 0xe848 | 0xdb0f |
| 312 | 0xcc94 | 0xa51b | 0xc94a | 0xefa8 | 0x1b42 | 0x0002 |
| 313 | 0x03aa | 0xcbbb | 0x0dc0 | 0xa117 | 0x5976 | 0x4c85 |
| 314 | 0xecd1 | 0xb2c2 | 0x4d34 | 0xdba2 | 0xce96 | 0x0eeb |
| 315 | 0xeaaa | 0xef67 | 0xe4b5 | 0xe78c | 0xc989 | 0x9dc2 |
| 316 | 0x247d | 0x2881 | 0xc9da | 0xe5d0 | 0x581c | 0xfdf9 |
| 317 | 0x19fb | 0x4950 | 0xed09 | 0x311a | 0x398a | 0xd81f |
| 318 | 0xfcc9 | 0x46c7 | 0x018e | 0xf9d2 | 0xff8c | 0xfe95 |
| 319 | 0xe4e9 | 0xce6a | 0x9118 | 0x2168 | 0x1b31 | 0xff11 |
| 320 | 0xf5d6 | 0xeda0 | 0xfc03 | 0x07df | 0x1409 | 0x5c76 |
| 321 | 0xcef1 | 0xe002 | 0x9e3c | 0x4870 | 0x3763 | 0x067f |
| 322 | 0x0ee5 | 0x522c | 0xda6c | 0xec45 | 0xf8f8 | 0xfbc1 |
| 323 | 0xa9d7 | 0x4123 | 0x3a70 | 0x24f3 | 0x0ae2 | 0x425f |
| 324 | 0x9a48 | 0xb48a | 0xe6f2 | 0x0450 | 0x16a6 | 0xb989 |
| 325 | 0xf937 | 0x60f9 | 0x28b1 | 0xd4b1 | 0x0380 | 0xeb67 |
| 326 | 0xf8c1 | 0x2d8d | 0xf50d | 0x60e9 | 0xac45 | 0xb2b0 |
| 327 | 0xa44f | 0xbea7 | 0xe96a | 0x160b | 0x0a4c | 0x134c |
| 328 | 0xf944 | 0x1124 | 0x97cf | 0xca81 | 0x294a | 0x9ad9 |
| 329 | 0x3bfe | 0xb3d8 | 0x6682 | 0xb7c3 | 0x06c8 | 0x1f76 |
| 330 | 0x1634 | 0x519a | 0x0ffb | 0xb564 | 0xc704 | 0xd71c |
| 331 | 0x436c | 0xc05d | 0x3a0b | 0xbad1 | 0xb51a | 0x3093 |
| 332 | 0x95cf | 0xcee3 | 0x1a57 | 0xfdce | 0x03d0 | 0xfeff |
| 333 | 0x306b | 0xde56 | 0xa918 | 0xb27d | 0x2b05 | 0x1e52 |
| 334 | 0x0ed7 | 0x2e4d | 0x941a | 0xdee7 | 0x0441 | 0xfa29 |
| 335 | 0x102d | 0xf77a | 0x97a0 | 0xfd21 | 0xfcfa | 0x05bd |
| 336 | 0x0c35 | 0x35c2 | 0x11fe | 0x7249 | 0x4953 | 0xd91a |
| 337 | 0xbbc7 | 0xdb1b | 0xbb66 | 0xf61e | 0xe6dd | 0xf172 |
| 338 | 0xf932 | 0x10ff | 0xe547 | 0xb2c3 | 0x259b | 0xd662 |
| 339 | 0x1c53 | 0x0dc5 | 0x2a53 | 0x15e1 | 0x626e | 0xa4cc |
| 340 | 0xd7c4 | 0xba5a | 0x0277 | 0x2d78 | 0x07fc | 0xae72 |
| 341 | 0xfc97 | 0xdeca | 0xfbd9 | 0xc2c6 | 0xd63b | 0x3a56 |
| 342 | 0xc1ab | 0x6de9 | 0x1494 | 0x01dd | 0xfbe3 | 0x0486 |
| 343 | 0xfa29 | 0xdd92 | 0xe97c | 0x9e7b | 0x6584 | 0x1ee3 |
| 344 | 0xfbf2 | 0xff8e | 0xf6fc | 0xfad9 | 0xe6b0 | 0x05c0 |
| 345 | 0x131f | 0xba17 | 0x9b06 | 0x14b5 | 0xff44 | 0x062d |
| 346 | 0x0c80 | 0x4349 | 0x10fa | 0x5655 | 0xb791 | 0x560c |
| 347 | 0xd7f6 | 0x0221 | 0xd54c | 0x08e4 | 0x925a | 0x1fb6 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 348 | 0x3bef | 0x0919 | 0x2464 | 0x5039 | 0x3a3c | 0x521d |
| 349 | 0x18b9 | 0x17f2 | 0xa044 | 0x0345 | 0xde43 | 0xe92c |
| 350 | 0x1cda | 0xfe0b | 0x2907 | 0x4ea3 | 0x2cab | 0xed6d |
| 351 | 0xf547 | 0x5e6e | 0xdbc6 | 0x3ba9 | 0xdf3b | 0xe935 |
| 352 | 0x0bb0 | 0xf4d0 | 0x17a0 | 0xe2cf | 0x2da7 | 0xb1e4 |
| 353 | 0xfc8d | 0xd14e | 0xd908 | 0xaabb | 0xeeac | 0x95d6 |
| 354 | 0x0d82 | 0x4caa | 0x0500 | 0x0a25 | 0x4d89 | 0x1487 |
| 355 | 0xeb3d | 0x4abd | 0xc74a | 0xdd0e | 0x35b5 | 0xfab8 |
| 356 | 0x48d2 | 0x44f7 | 0x2af9 | 0x1aa1 | 0xb80e | 0x18c0 |
| 357 | 0xf95f | 0x08c4 | 0xede0 | 0x0f6c | 0xcda6 | 0xeb67 |
| 358 | 0x4fcc | 0x292e | 0x104a | 0xfc0c | 0x4bef | 0x54bb |
| 359 | 0xf481 | 0xb2e9 | 0xef90 | 0x0528 | 0x038d | 0xdd3f |
| 360 | 0x2487 | 0xe07e | 0xf5c6 | 0xcd7b | 0x67d6 | 0x0db3 |
| 361 | 0x25e9 | 0xa79c | 0x2077 | 0x1fe7 | 0xcc13 | 0x15e8 |
| 362 | 0x0c96 | 0x0ea5 | 0xfa1c | 0x00a5 | 0xffcc | 0xff3c |
| 363 | 0x0066 | 0xa728 | 0xdd80 | 0x0387 | 0xd363 | 0xc6ba |
| 364 | 0xff88 | 0x176e | 0x4d35 | 0x3459 | 0x0e2c | 0x144d |
| 365 | 0x2150 | 0x16c3 | 0xfbd6 | 0x0306 | 0xffd9 | 0xff5a |
| 366 | 0x24c3 | 0xdafc | 0x256d | 0xcd34 | 0x5f88 | 0x6144 |
| 367 | 0x45d6 | 0x08bb | 0xab79 | 0x4ffe | 0x126c | 0xe3ea |
| 368 | 0xf64e | 0x2527 | 0x064b | 0xaa49 | 0x3796 | 0xfaf7 |
| 369 | 0x2448 | 0xf70d | 0x5aaf | 0xf284 | 0xd5a6 | 0x000b |
| 370 | 0x2518 | 0x0be1 | 0x140a | 0xf0ce | 0xad1d | 0xa7c3 |
| 371 | 0x37b6 | 0xd992 | 0x4ee3 | 0x36c3 | 0x005b | 0xbcd0 |
| 372 | 0xb761 | 0x03d4 | 0x0011 | 0x0335 | 0x0078 | 0xfdc2 |
| 373 | 0x2ffd | 0xb4bb | 0x35ae | 0x3ff5 | 0xff5f | 0x1789 |
| 374 | 0xf2dc | 0x05fa | 0xf05b | 0x0996 | 0xd588 | 0xa2e1 |
| 375 | 0x0069 | 0x13dd | 0xfefc | 0x169e | 0xfdb4 | 0x4ae2 |
| 376 | 0x1019 | 0x1049 | 0x347f | 0x3934 | 0x51a3 | 0x1d0a |
| 377 | 0xff51 | 0x332d | 0xf188 | 0x5ac1 | 0x0f43 | 0x277a |
| 378 | 0xe82b | 0x5bab | 0x1454 | 0xfac3 | 0x063f | 0x3376 |
| 379 | 0xf36f | 0xf25a | 0x3b0d | 0xdf3d | 0xd20e | 0xed72 |
| 380 | 0x047a | 0x1243 | 0xb44e | 0x3a45 | 0xec1d | 0x00f9 |
| 381 | 0xabfe | 0x2798 | 0xbfa7 | 0xcc07 | 0x47ce | 0xde67 |
| 382 | 0x0274 | 0x098f | 0x0d10 | 0x0c3a | 0xec05 | 0x0077 |
| 383 | 0x45ec | 0xa86a | 0xbb1f | 0x55cf | 0xc05b | 0xe204 |
| 384 | 0x41df | 0x5e96 | 0x15ec | 0xf0ee | 0xfcd7 | 0x0eee |
| 385 | 0xf70d | 0x276b | 0xf6c8 | 0x9deb | 0xfb36 | 0x0138 |
| 386 | 0x0b8d | 0x2bf8 | 0x6879 | 0xcc2e | 0xf281 | 0xfb98 |
| 387 | 0xb2ce | 0xf56c | 0x11fc | 0x18d3 | 0x0666 | 0x639d |
| 388 | 0xb377 | 0xe1b7 | 0x0c57 | 0xffab | 0xfe17 | 0xf8c1 |
| 389 | 0x032e | 0x30de | 0x4a85 | 0xedb7 | 0xf5ce | 0xfa3e |
| 390 | 0xa490 | 0xb5ad | 0x1fc9 | 0x4da6 | 0x1ee8 | 0xfee6 |
| 391 | 0x0347 | 0xb33c | 0x2e97 | 0x6a8e | 0xf375 | 0x08da |

| | | | | | | |
|---|---|---|---|---|---|---|
| 392 | 0x0fb4 | 0xfbaa | 0x2022 | 0xfb06 | 0x51ba | 0x61e4 |
| 393 | 0x67d0 | 0x0145 | 0xde0b | 0xff18 | 0xf756 | 0xfd45 |
| 394 | 0xd3e3 | 0xef98 | 0x070d | 0xe5ef | 0xa664 | 0xfac5 |
| 395 | 0xf82b | 0xc1f2 | 0xfbe9 | 0x93d9 | 0xcc4d | 0x3822 |
| 396 | 0xa9c7 | 0x079d | 0x3377 | 0xc2d8 | 0xf8ca | 0x1f77 |
| 397 | 0x0bdf | 0x2ef9 | 0x1bdc | 0x9fc8 | 0x019d | 0xf6d5 |
| 398 | 0xa210 | 0xff32 | 0x30ab | 0xe602 | 0xfe5f | 0xd895 |
| 399 | 0x4703 | 0xa378 | 0xafdd | 0xbff4 | 0x1c3e | 0x02fb |
| 400 | 0x161b | 0xec23 | 0x3636 | 0xa34f | 0xd4bb | 0xb37d |
| 401 | 0x2c4c | 0x01f5 | 0x61d0 | 0x1dc0 | 0xb336 | 0x0645 |
| 402 | 0x97e6 | 0x22ae | 0x2930 | 0x01a1 | 0x0513 | 0x0105 |
| 403 | 0x387c | 0x2c69 | 0xf341 | 0x2706 | 0x2002 | 0x46bf |
| 404 | 0x054b | 0xae9a | 0xdc14 | 0xc144 | 0xde91 | 0xfd26 |
| 405 | 0xf882 | 0xae37 | 0xb88b | 0xf663 | 0xf5a5 | 0x10dc |
| 406 | 0xf506 | 0x5fc9 | 0xd50c | 0x9b87 | 0x0134 | 0xfb2e |
| 407 | 0xdc8d | 0xbc80 | 0xf8d7 | 0x8d62 | 0xa16b | 0xbf09 |
| 408 | 0xf4e5 | 0x27b5 | 0xeb25 | 0xf4b8 | 0x5562 | 0xaca4 |
| 409 | 0xc228 | 0x3a01 | 0xa31c | 0x1440 | 0x2781 | 0xaf61 |
| 410 | 0xb3b1 | 0xd39f | 0x20dd | 0x05ce | 0xa396 | 0xe981 |
| 411 | 0xe2a8 | 0x0403 | 0xaec6 | 0x35a4 | 0x4db4 | 0xaa52 |
| 412 | 0xd09c | 0xe492 | 0xb519 | 0xdd78 | 0x566d | 0xbf96 |
| 413 | 0x0791 | 0x145a | 0xe752 | 0xa314 | 0x3355 | 0x2b4a |
| 414 | 0xff33 | 0x1794 | 0xfe84 | 0x21d2 | 0xff17 | 0x6d74 |
| 415 | 0xea6d | 0x1d35 | 0x1dd3 | 0x5c2b | 0x2623 | 0xf5e2 |
| 416 | 0x549a | 0x9167 | 0xf3f2 | 0xfed4 | 0xfbf8 | 0x06d0 |
| 417 | 0xa8b0 | 0x4106 | 0x00d0 | 0x1a09 | 0xbc08 | 0xf42c |
| 418 | 0x4832 | 0x2478 | 0xf54f | 0xb454 | 0x0197 | 0xeedb |
| 419 | 0xeccf | 0xbc26 | 0x4983 | 0xbb0a | 0x3468 | 0x3b80 |
| 420 | 0x1e45 | 0x18e0 | 0x5a5f | 0xb902 | 0x1da0 | 0xef68 |
| 421 | 0xfa2f | 0xe685 | 0x024a | 0xd853 | 0x3a74 | 0x63e0 |
| 422 | 0x0f04 | 0xe7f4 | 0x1321 | 0xcd0b | 0xa802 | 0x160f |
| 423 | 0xded5 | 0xf7c7 | 0x9f3a | 0x0389 | 0xdb92 | 0x05b0 |
| 424 | 0xf420 | 0xfa3c | 0x048e | 0xeeb4 | 0x2be4 | 0x23f4 |
| 425 | 0x0d45 | 0xfa55 | 0x351e | 0xc21f | 0x5fdc | 0x16bb |
| 426 | 0x2123 | 0xf44f | 0x542b | 0xbdec | 0x1e3d | 0x5dd2 |
| 427 | 0xc5ac | 0xa332 | 0xeb2c | 0xe5f8 | 0xee6f | 0x33d3 |
| 428 | 0x4bb3 | 0x3274 | 0xf7a2 | 0xfd1f | 0x526c | 0xa9ab |
| 429 | 0x0d41 | 0xedeb | 0x1667 | 0xb61f | 0xe4c7 | 0x0a7f |
| 430 | 0x047c | 0xc0ed | 0xac47 | 0x9241 | 0xfd8a | 0xc78f |
| 431 | 0x1c84 | 0x02a0 | 0x4862 | 0xbbd4 | 0xd85b | 0x015f |
| 432 | 0x2c5c | 0xd522 | 0x433c | 0x1210 | 0x0091 | 0x457f |
| 433 | 0xfd39 | 0xf269 | 0xf742 | 0x3e0f | 0x07eb | 0x0000 |
| 434 | 0x9270 | 0x0702 | 0xfdaf | 0xf53a | 0xaaa4 | 0x2d0f |
| 435 | 0xb31d | 0x1349 | 0x55f4 | 0x5413 | 0xf3b4 | 0x06fe |

| | | | | | | |
|---|---|---|---|---|---|---|
| 436 | 0x032d | 0x2027 | 0x0a49 | 0x2ecd | 0xf41d | 0x56b9 |
| 437 | 0x22f8 | 0x9f48 | 0xfd4e | 0x3a19 | 0xf6c2 | 0xeb04 |
| 438 | 0x20d6 | 0xeac1 | 0xfeee | 0xfd7e | 0xff6f | 0x030a |
| 439 | 0xe633 | 0x1c5a | 0x512c | 0xa42d | 0xb73f | 0x58fe |
| 440 | 0xa690 | 0x9c70 | 0x2724 | 0xf9b2 | 0x05e4 | 0xfa8f |
| 441 | 0x1db7 | 0x0197 | 0x9f9a | 0xbfff | 0xf8f4 | 0xeda5 |
| 442 | 0xd6a0 | 0xb53d | 0x28de | 0xf15d | 0x2211 | 0xe4f9 |
| 443 | 0x32d2 | 0x14ac | 0xe7aa | 0xecec | 0xae58 | 0xf8fb |
| 444 | 0x41fb | 0xca36 | 0xfe2f | 0x4b8f | 0xd60b | 0xcd61 |
| 445 | 0x6269 | 0xc631 | 0xe9cf | 0xfdf7 | 0xfebf | 0xfb45 |
| 446 | 0x1b05 | 0xf3eb | 0x4ed7 | 0x96e9 | 0xd106 | 0x050f |
| 447 | 0x0131 | 0x07c8 | 0x4c01 | 0xfc27 | 0x0019 | 0xfdf7 |
| 448 | 0x1a33 | 0xf18e | 0x20ad | 0xde11 | 0x55a1 | 0x95e2 |
| 449 | 0x123c | 0x176d | 0x1bcd | 0x2db0 | 0x5f51 | 0xd5d6 |
| 450 | 0x02e8 | 0xdb38 | 0x4db5 | 0x07ab | 0x1ef2 | 0xd9a0 |
| 451 | 0x0d66 | 0x5322 | 0xf938 | 0x2a5c | 0x2275 | 0x6987 |
| 452 | 0xdd93 | 0x05f1 | 0xa21a | 0x0673 | 0x1e9e | 0xfb48 |
| 453 | 0x0f47 | 0xd42b | 0x0cc9 | 0xcf03 | 0x1c00 | 0x47e2 |
| 454 | 0x548a | 0x239d | 0xd2f0 | 0xeb78 | 0x1ba5 | 0x094e |
| 455 | 0x0064 | 0x0ee9 | 0xe5c7 | 0x04dc | 0x05ee | 0xfebf |
| 456 | 0x1f0a | 0xb712 | 0x29ab | 0xecfe | 0x02d7 | 0x0308 |
| 457 | 0xc1f5 | 0xe02a | 0xf7d9 | 0x58d3 | 0x061f | 0xf266 |
| 458 | 0x111c | 0xf551 | 0x2115 | 0xe48f | 0xd360 | 0x0525 |
| 459 | 0x695a | 0x1129 | 0x1df1 | 0x4499 | 0xfd36 | 0x028a |
| 460 | 0xc0c1 | 0xfcbd | 0x20ad | 0x0703 | 0xc816 | 0x3fa9 |
| 461 | 0x1198 | 0xd8c0 | 0x1dee | 0x97be | 0xbbec | 0x0a14 |
| 462 | 0x0030 | 0xf070 | 0x0234 | 0xe911 | 0x0a62 | 0xb71e |
| 463 | 0x3123 | 0x9a60 | 0xc2e6 | 0x0a70 | 0xfabd | 0xfc89 |
| 464 | 0xecaa | 0x1070 | 0xe565 | 0x0a09 | 0xaf73 | 0xde2e |
| 465 | 0xf8d4 | 0xc61e | 0xea3d | 0xa4e6 | 0xc630 | 0x650b |
| 466 | 0x153a | 0x9215 | 0xf6cb | 0xf4bd | 0xfdc6 | 0x097f |
| 467 | 0x3328 | 0xf52d | 0x61a2 | 0xcf30 | 0x9f6d | 0xfbff |
| 468 | 0xe9d4 | 0xef0d | 0x0774 | 0x48c4 | 0xacb5 | 0x43d6 |
| 469 | 0x6c0c | 0x9307 | 0xc3cf | 0x059c | 0xe438 | 0xf73f |
| 470 | 0x1f53 | 0x0f07 | 0x5ff8 | 0xfe2b | 0x25ca | 0x29bb |
| 471 | 0xfc79 | 0xd85b | 0x0709 | 0xacf4 | 0x12bb | 0xddd1 |
| 472 | 0x0462 | 0xda92 | 0x0a41 | 0x5907 | 0x03bc | 0x0372 |
| 473 | 0x1ec4 | 0x4a83 | 0xd954 | 0xa136 | 0x1d48 | 0x243d |
| 474 | 0x03d4 | 0x9774 | 0xeaf6 | 0x1514 | 0x043e | 0x0670 |
| 475 | 0x70a6 | 0xfb0a | 0xfe41 | 0x0005 | 0xfe53 | 0xffec |
| 476 | 0xc44d | 0x17f4 | 0x591c | 0x04e4 | 0xd915 | 0x01ff |
| 477 | 0x0353 | 0x1ef5 | 0xfe37 | 0xd04e | 0x10a5 | 0x1d9b |
| 478 | 0xee4e | 0x2104 | 0xfb22 | 0x38a5 | 0x9e89 | 0xe980 |
| 479 | 0xba6a | 0xd619 | 0x269f | 0xa287 | 0xcb88 | 0x0756 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 480 | 0xc537 | 0x27b5 | 0x406b | 0xc6f5 | 0xd240 | 0xad5c |
| 481 | 0xf30b | 0x0348 | 0xe9cd | 0x578d | 0x07ca | 0x024a |
| 482 | 0x5a76 | 0xe964 | 0xc53d | 0xd77c | 0xdbc9 | 0xcb2d |
| 483 | 0xfcfb | 0xdadb | 0xf067 | 0xa138 | 0x210f | 0x16ac |
| 484 | 0xde9f | 0xfd41 | 0xcf68 | 0xf06f | 0x9dde | 0x920d |
| 485 | 0xbeed | 0x3e81 | 0x0aba | 0x064b | 0x13e9 | 0xfbed |
| 486 | 0x0029 | 0xe3f3 | 0x4dbf | 0x7b43 | 0x8213 | 0x3667 |
| 487 | 0xe9e6 | 0x034d | 0xce1a | 0x1649 | 0x4137 | 0xffaa |
| 488 | 0x14a2 | 0x3a1b | 0x6992 | 0x5284 | 0x3da0 | 0xd713 |
| 489 | 0x3978 | 0x4cc0 | 0xd321 | 0xcbcf | 0xb11c | 0xc483 |
| 490 | 0x2195 | 0xdc4e | 0xfd8e | 0x2a8b | 0xe881 | 0x18ca |
| 491 | 0xfa30 | 0xfb08 | 0xfa39 | 0xfae9 | 0xf188 | 0xea93 |
| 492 | 0xf2d6 | 0x45cf | 0xe634 | 0x6162 | 0x651e | 0xf3c9 |
| 493 | 0x20e0 | 0x6c87 | 0xfa97 | 0x14e6 | 0xef5c | 0x4e19 |
| 494 | 0x1638 | 0x016a | 0x435e | 0x0ee1 | 0xf352 | 0x0440 |
| 495 | 0xff97 | 0x8c59 | 0x0abb | 0x3b77 | 0xff59 | 0x0e8a |
| 496 | 0x0dae | 0xf385 | 0x219a | 0x1e5c | 0xf9e2 | 0xfc6d |
| 497 | 0xfe15 | 0x0cb9 | 0xf689 | 0x1592 | 0x507e | 0xff9c |
| 498 | 0xc984 | 0xd398 | 0xc3f1 | 0xaa96 | 0xdeb8 | 0x2fbd |
| 499 | 0xfd98 | 0x0978 | 0xf819 | 0x112e | 0xf123 | 0x1fac |
| 500 | 0xe3dc | 0x5233 | 0x52db | 0xdb4d | 0xb441 | 0x0380 |
| 501 | 0xe997 | 0xc4c8 | 0xacce | 0x42aa | 0xfc12 | 0xfe92 |
| 502 | 0x1875 | 0x0ca8 | 0xd15f | 0xc0ab | 0xc234 | 0x19b5 |
| 503 | 0xf3ad | 0x60dc | 0x0aad | 0xfb17 | 0xfc95 | 0xf9c3 |
| 504 | 0xb00b | 0x2b56 | 0x5e07 | 0xdce5 | 0x3738 | 0x08ac |
| 505 | 0xc8e6 | 0x2eb7 | 0xa821 | 0x1027 | 0xfbe1 | 0xead4 |
| 506 | 0x0321 | 0xf5a1 | 0x003c | 0xeb34 | 0xfcea | 0x1731 |
| 507 | 0xe334 | 0xf91c | 0xa85f | 0x9a34 | 0x54cb | 0x1052 |
| 508 | 0xe9ad | 0xe608 | 0xc5c4 | 0x052d | 0xa214 | 0x05d5 |
| 509 | 0xe878 | 0xcf38 | 0x5d7a | 0x0b86 | 0x0641 | 0x0495 |
| 510 | 0x4a7b | 0x44de | 0x4609 | 0xd662 | 0x2ab0 | 0xeca2 |
| 511 | 0x0c9f | 0xf32c | 0x6ac8 | 0x104e | 0xf96d | 0x01f1 |