floorcod	Masking Floor Code
floortab	Masking Floor Table
frmsizecod	Frame Size Code
fscod	Sampling Frequency Code
fsnroffst	Channel Fine SNR Offset
gainrng	Channel Gain Range Code
langcod	Language Code
langcod2	Language Code, Ch2
langcode	Language Code Exists
langcode2	Language Code Exists, Ch2
lfe	Low Frequency Effects Channel
lfeexps	Low Frequency Effects Channel Exponents
lfeexpstr	Low Frequency Effects Channel Exponent Strategy
lfefgaincod	Low Frequency Effects Channel Fast Gain Code
lfefsnroffst	Low Frequency Effects Channel Fine SNR Offset
lfemant	Low Frequency Effects Channel Mantissas
lfeon	Low Frequency Effects Channel On
mixlevel	Mixing Level
mixlevel2	Mixing Level, Ch2
mstrcplco	Master Coupling Coordinate
nauxbits	Number of Auxiliary Bits
nchans	Number of Channels
nchgrps	Number of Full-Bandwidth Channel Exponent Groups
nchmant	Number of Full-Bandwidth Channel Mantissas
ncplbnd	Number of Coupled Bands
ncplgrps	Number of Coupled Exponent Groups
ncplmant	Number of Coupled Mantissas
nfchans	Number of Full-Bandwidth Channels
nlfegrps	Number of Low Frequency Effects Channel Exponent Groups
nlfemant	Number of Low Frequency Effects Channel Mantissas
nrematbnd	Number of Rematrixed Bands
origbs	Original Bitstream
phsflg	Phase Flag
phsflginu	Phase Flags In Use
rematflg	Rematrix Flag
rematstr	Rematrixing Strategy
roomtyp	Room Type
roomtyp2	Room Type, Ch2
sdcycod	Slow Decay Code
sgaincod	Slow Gain Code
skipl	Skip Length
skiple	Skip Length Exists
snroffste	SNR Offset Exists
surmixlev	Surround Mix Level
syncframe	Synchronization Frame

syncinfo	Synchronization Information
syncword	Synchronization Word
timecod1	Time Code First Half
timecod2	Time Code Second Half
timecod1e	Time Code First Half Exists
timecod2e	Time Code Second Half Exists

2.1.2. Definitions

2.1.2.1. Coupled channel

A full bandwidth channel whose high frequency information is combined into the coupling channel.

2.1.2.2. Coupled set

The exponent set for the coupling channel.

2.1.2.3. Coupling channel

The combination of the high frequency information from the coupled channels.

2.1.2.4. Exponent set

The group of exponents for an independent channel, for the coupling channel, or for the low frequency portion of a coupled channel.

2.1.2.5. Independent channel

A channel whose high frequency information is not combined into the coupling channel. (The subwoofer channel is always independent.)

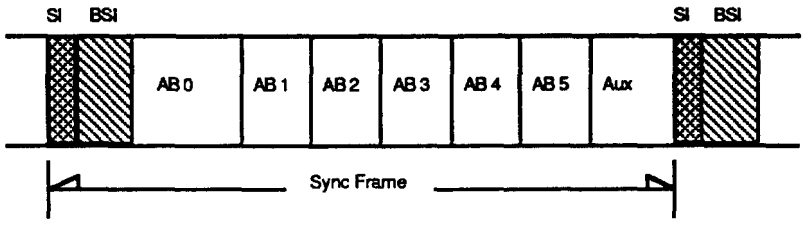
2.1.2.6. Non-coupled set or uncoupled set

The exponent set for the low frequency portion of a coupled channel.

2.2. AC-3 Syntax

2.2.1. Synchronization Frame

An AC-3 serial coded audio bitstream is made up of synchronization frames containing 6 coded Audio Blocks (AB) each representing 256 audio samples. A Synchronization Information (SI) header at the beginning of each frame contains information needed to acquire and maintain synchronization. A Bit Stream Information (BSI) header follows SI, and contains parameters describing the mode of the coded audio service(s). The Audio Blocks may be followed by an Auxiliary Data (Aux) field. At the end of each frame is an error check field that includes a CRC word for error detection, and there is an additional CRC word located in the SI header.



2.2.2. Syntax

The following pseudo code mnemonics describe the order of arrival of information within the bitstream. Some of the control and informative data which need not be sent as often as every syncframe is conditionally present based on preceding flags in the stream. This method is used in order to more efficiently use the available data rate for audio information. This pseudo code is roughly based on C language syntax, but simplified for ease of reading. For example, statements are not followed by semicolons, but are kept one to a line.

A continuous audio bitstream would consist of repeating the following sequence of data.

```
syncframe()
    syncinfo()
    bsi()
    for(i = 0; i < 6; i++)
    {
        audblk[i]()
    }
    auxdata()
    errorcheck()
```

Each of the syncframe elements is defined in the following pseudo code. Note that all bitstream elements arrive most significant bit first in time unless noted otherwise.

mnemonic	word size

```
syncinfo()
{
    syncword ..................................... 16
    crc1 ......................................... 16
    fscod ........................................ 2
    frmsizecod ................................... 6
}

bsi()
{
    bsid ......................................... 5
    bsmod ........................................ 3
```

```
    acmod ..................................................................3

    if((acmod & 0x1) && (acmod != 0x1)) {cmixlev}  /* if 3 front channels */.....2
    if(acmod & 0x4) {surmixlev}  /* if surround channel(s) exists */.....2
    if(acmod == 0x2) {dsurmod}  /* if in 2/0 mode */.....2
    lfeon ..................................................................1
    dialnorm ...............................................................5
    compre .................................................................1
    if(compre) {compr} .....................................................8
    langcode ...............................................................1
    if(langcode) {langcod} .................................................8
    audprodie ..............................................................1
    if(audprodie)
    {
        mixlevel ...........................................................5
        roomtyp ............................................................2
    }
    if(acmod == 0)  /* if 1+1 mode (dual mono) */
    {
        dialnorm2 ..........................................................5
        compr2e ............................................................1
        if(compr2e)  {compr2} ..............................................8
        langcod2e ..........................................................1
        if(langcod2e) {langcode2} ..........................................8
        audprodi2e ........................................................1
        if(audprodi2e)
        {
            mixlevel2 ......................................................5
            roomtyp2 .......................................................2
        }
    }
    copyrightb .............................................................1
    origbs ................................................................1
    timecod1e ..............................................................1
    if(timecod1e) {timecod1} ..............................................14
    timecod2e ..............................................................1
    if(timecod2e) {timecod2} ..............................................14
    addbsie ................................................................1
    if(addbsie)
    {
        addbsil ............................................................6
        addbsi ........................................................(addbsil+1)×8
    }
}   /* end of bsi */
```

```
audblk()
{
    for(n = 0; n < nfchans; n++) {blksw[n]}.....1
    for(n = 0; n < nfchans; n++) {dithflag[n]}.....1

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

/*These fields for coupling strategy*/
    cplstre.....1
    if(cplstre)
    {
        cplinu.....1
        if(cplinu)
        {
            for(n = 0; n < nfchans; n++) {chincpl[n]}.....1
            if(acmod == 0x2) {phsflginu} /* if in 2/0 mode */.....1
            cplbegf.....4
            cplendf.....4
            for(i = 0; i < (cplendf - cplbegf + 2); i++) {cplbndstrc[i]}.....1
        }
    }

/*These fields for coupling coordinates*/
    if(cplinu)
    {
        for(n = 0; n < nfchans; n++)
        {
            if(chincpl[n])
            {
                cplcoe[n].....1
                if(cplcoe[n])
                {
                    mstrcplco[n].....2
                    for(j = 0; j < ncplbnd; j++) /*ncplbnd derived from cplendf, cplbegf, and cplbndstrc*/
                    {
                        cplcoexp[n][j].....4
                        cplco[n][j].....4
                    }
```

```
                }
            }
        }
        if((acmod == 0x2) && phsflginu && (cplcoe[0] || cplcoe[1]))
        {
            for(j = 0; j < ncplbnd; j++) {phsflg[j]} ..........................................1
        }
    }

/*These fields for rematrixing operation in the 2/0 mode*/
    if(acmod == 0x2)  /* if in 2/0 mode */
    {
        rematstr ..........................................................................1
        if(rematstr != reuse)
        {
            if((cplbegf > 2) || (cplinu == 0))
            {
                for(n = 0; n < 4; n++) {rematflg[n]} ..................................1
            }
            if((2 ≥ cplbegf > 0) && cplinu)
            {
                for(n = 0; n < 3; n++) {rematflg[n]} ..................................1
            }
            if((cplbegf == 0) && cplinu)
            {
                for(n = 0; n < 2; n++) {rematflg[n]} ..................................1
            }
        }
    }

/*These fields for exponent strategy*/
    if(cplinu)  {cplexpstr} ...........................................................2
    for(n = 0; n < nfchans; n++)  {chexpstr[n]} .......................................2
    if(lfeon)  {lfeexpstr} ............................................................1
    for(n = 0; n < nfchans; n++)
    {
        if(chexpstr[n] != reuse)
        {
            if(!chincpl[n])  {chbwcod[n]} .............................................6
        }
    }

/*These fields for exponents*/
    if(cplinu)
    {
```

```
        if(cplexpstr != reuse)
        {
            cplabsexp ..........................................4
            for(i = 0; i < ncplgrps; i++) {cplexps[i]} .........7
        }
    }
    for(n = 0; n < nfchans; n++)
    {
        if(chexpstr[n] != reuse)
        {
            exps[n][0] ..........................................4
            for(i = 1; i < nchgrps; i++) {exps[n][i]} ...........7
            gainrng[n] ..........................................2
        }
    }
    if(lfeon)
    {
        if(lfeexpstr != reuse)
        {
            lfeexps[0] ..........................................4
            for(i = 1 ;i < nlfegrps; i++) {lfeexps[i]} ..........7
                                                    /* nlfegrps = 2 */
        }
    }

/*These fields for bit-allocation strategy information*/
    baie ....................................................1
    if(baie)
    {
        sdcycod ..............................................2
        fdcycod ..............................................2
        sgaincod .............................................2
        dbpbcod ..............................................2
        floorcod .............................................3
    }
    snroffste ...............................................1
    if(snroffste)
    {
        csnroffst ............................................6
        if(cplinu)
        {
            cplfsnroffst .....................................4
            cplfgaincod ......................................3
        }
        for(n = 0; n < nfchans; n++)
```

```
    {
        fsnroffst[n] ..... 4
        fgaincod[n] ..... 3
    }
    if(lfeon)
    {
        lfefsnroffst ..... 4
        lfefgaincod ..... 3
    }
}
if(cplinu)
{
    cplleake ..... 1
    if(cplleake)
    {
        cplfleak ..... 3
        cplsleak ..... 3
    }
}
deltbaie ..... 1
if(deltbaie)
{
    if(cplinu) {cpldeltbae} ..... 2
    for(n = 0; n < nfchans; n++) {deltbae[n]} ..... 2
    if(cplinu)
    {
        if(cpldeltbae)
        {
            cpldeltnseg ..... 3
            for(j = 0; j < cpldeltnseg+1; j++)
            {
                cpldeltoffst[j] ..... 5
                cpldeltlen[j] ..... 4
                cpldeltba[j] ..... 3
            }
        }
    }
    for(n = 0; n < nfchans; n++)
    {
        if(deltbae[n])
        {
            deltnseg[n] ..... 3
            for(j = 0; j < deltnseg[n]+1; j++)
            {
                deltoffst[n][j] ..... 5
```

```
                    deltlen[n][j] ..... 4
                    deltba[n][j] ..... 3
                }
            }
        }
    }
    skiple ..... 1
    if(skiple) {skipl} ..... 9

/*These fields for quantized mantissa values*/
    if(cplinu)
    {
        for(i = 0; i < ncplmant;i++) {cplmant[i]} ..... (0-16)
    }
    for(n = 0; n <nfchans; n++)
    {
        for(i = 0; i < nchmant[n]; i++) {chmant[n][i]} ..... (0-16)
    }
    if(lfeon)
    {
        for(i = 0; i < nlfemant; i++) {lfemant[i]} ..... (0-16)
    }
}   /* end of audblk */

auxdata()
{
    for(i = 0; i < nauxbits; i++) {auxbits} ..... nauxbits
}

errorcheck()
{
    crcrsv ..... 1
    crc2 ..... 16
}
```

2.3. Description Of Bitstream Elements

2.3.1. syncinfo - Synchronization Information

2.3.1.1. syncword - Synchronization Word - 16 bits

The syncword is always 0x0B77 or 0000 1011 0111 0111.

2.3.1.2. crc1 - Cyclic Redundancy Check 1 - 16 bits

The 16 bit CRC applies to the first 5/8 of the frame.

2.3.1.3. fscod - Sample Rate Code - 2 bits

This is a 2 bit code indicating sample rate according to the following table.

fscod	sampling rate, kHz
00	48
01	44.1
10	32
11	reserved

2.3.1.4. frmsizecod - Frame Size Code - 6 bits

The frame size code is used along with the sample rate code to determine the number of bytes before the next syncword. See Table 2.1.

2.3.2. bsi - Bitstream Information

2.3.2.1. bsid - Bitstream Identification - 5 bits

This bit field has a value of 00011 (=3) in this version of the coder syntax. The decoder should mute if the value in this bit field is greater than 3.

2.3.2.2. bsmod - Bitstream Mode - 3 bits

This 3-bit code indicates the type of service that the bitstream conveys as defined by the following table.

bsmod	type of service
000	main audio service
001	main audio service minus dialog
010	associated service: visually impaired
011	associated service: hearing impaired
100	associated service: dialog
101	associated service: commentary
110	associated service: emergency - flash
111	reserved

2.3.2.3. acmod - Audio Coding Mode - 3 bits

This 3 bit code indicates which of the main service channels are in use, ranging from 3/2 to 1/0. If the msb of acmod is a 1, surround channels are in use and surmixlev follows in the bitstream. If the msb of acmod is a 0, the surround channels are not in use and surmixlev does not follow in the bitstream. If the lsb of acmod is a 0, the center channel is not in use. If the lsb of acmod is a 1, the center channel is in use. Note: The state of acmod sets the number of full-bandwidth channels parameter, nfchans, (e.g., for 3/2 mode, nfchans = 5; for 2/1 mode, nfchans = 3; etc.). If acmod is 0, then two completely independent program channels (dual mono) are encoded into the bit stream, and are referenced as Ch1, Ch2. In this case, a number of additional items are present in BSI or audblk to fully describe Ch2. Acmod is coded as follows. The table also indicates the channel ordering (the order in which the channels are processed) for each of the modes.

acmod	audio coding mode	channel array ordering
000	1+1	[Ch1, Ch2]
001	1/0	[C]
010	2/0	[L, R]
011	3/0	[L, C, R]
100	2/1	[L, R, S]
101	3/1	[L, C, R, S]
110	2/2	[L, R, SL, SR]
111	3/2	[L, C, R, SL, SR]

2.3.2.4. cmixlev - Center Mix Level - 2 bits

When three front channels are in use, this 2 bit code indicates the nominal down mix level of the center channel with respect to the left and right channels.

cmixlev	mix level
00	0.707 (-3.0 dB)
01	0.596 (-4.5 dB)
10	0.500 (-6.0 dB)
11	reserved

2.3.2.5. surmixlev - Surround Mix Level - 2 bits

If surround channels are in use, this 2 bit code indicates the nominal down mix level of the surrounds (consistent with the ITU BR specification).

surmixlev	mix level
00	0.707 (-3 dB)
01	0.500 (-6 dB)
10	0
11	reserved

2.3.2.6. dsurmod - Dolby Surround Mode - 2 bits

When operating in the two channel mode, this 2 bit code indicates whether or not the program has been encoded in Dolby Surround. This information is not used by the AC-3 decoder, but may be used by other portions of the audio reproduction equipment.

dsurmod	indication
00	no indication
01	NOT Dolby Surround encoded
10	Dolby Surround encoded
11	reserved

2.3.2.7. lfeon - Low Frequency Effects Channel On - 1 bit

This bit has a value of 1 if the lfe (sub woofer) channel is on, and a value of 0 if the lfe channel is off.

2.3.2.8. dialnorm - Dialogue Normalization - 5 bits

This 5 bit code indicates how far the average dialogue level is below digital 100%. Valid values are 0-31, which are interpreted as 0 dB to -31 dB with respect to digital 100%. The value of dialnorm is not used by the AC-3 decoder, but may be used by other parts of the audio reproduction equipment.

2.3.2.9. compre - Compression Code Exists - 1 bit

If this bit is a 1, the following 8 bits represent a compression control value.

2.3.2.10. compr - Compression Gain Word - 8 bits

This encoder generated gain word may be present in the bit stream. If so, it may used to scale the reproduced audio level in order to reproduce a very narrow dynamic range, with an assured limit in instantaneous peak reproduced signal level in the monophonic downmix. The meaning and use of compr is described further in section 8.

2.3.2.11. langcode - Language Code Exists - 1 bit

If this bit is a 1, the following 8 bits represent a language code. If this bit is a 0, the language of the main audio service is not indicated.

2.3.2.12. langcod - Language Code - 7 bits

This is an 7 bit code representing the language of the main audio service. See Table 2.2 for the mapping of langcod into language.

2.3.2.13. audprodie - Audio Production Information Exists - 1 bit

If this bit is a 1, the following two data fields exist indicating information about the audio production.

2.3.2.14. mixlevel - Mixing Level - 5 bits

This 5 bit code indicates the acoustic sound pressure level of the dialogue level during the final audio mixing session. The 5 bit code represents decibels from 0 dB to 31 dB. The reference mixing level is 60 dB SPL plus the value of mixlevel, or 60 dB to 91 dB SPL. The value of mixlevel is not used by the AC-3 decoder, but may be used by other parts of the audio reproduction equipment.

2.3.2.15. roomtyp - Room Type - 2 bits

This 2 bit code indicates the type and calibration of the mixing room used for the final audio mixing session. The value of roomtyp is not used by the AC-3 decoder, but may be used by other parts of the audio reproduction equipment.

roomtyp	type of mixing room
00	not indicated
01	large room, X curve monitor
10	small room, flat monitor
11	reserved

2.3.2.16. dialnorm2 - Dialogue Normalization, Ch2 - 5 bits

This 5 bit code has the same meaning as dialnorm, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono, 1+1 mode).

2.3.2.17. compr2e - Compression Code Exists, Ch2 - 1 bit

If this bit is a 1, the following 8 bits represent a compression control value for Ch2.

2.3.2.18. compr2 - Compression Control Code, Ch2 - 8 bits

This 8 bit code has the same meaning as compr, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono, 1+1 mode).

2.3.2.19. langcode - Language Code Exists, Ch2 - 1 bit

If this bit is a 1, the following 8 bits represent a language code for Ch2. If this bit is a 0, the language of the Ch2 is not indicated.

2.3.2.20. langcode - Language Code, Ch2 - 8 bits

This 8 bit code has the same meaning as langcod, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono, 1+1 mode).

2.3.2.21. audprodie - Audio Production Information Exists, Ch2 - 1 bit

If this bit is a 1, the following two data fields exist indicating information about the audio production for Ch2.

2.3.2.22. mixlevel2 - Mixing Level, Ch2 - 5 bits

This 5 bit code has the same meaning as mixlevl, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono, 1+1 mode).

2.3.2.23. roomtyp2 - Room Type, Ch2 - 2 bits

This 2 bit code has the same meaning as roomtyp, except that it applies to the second audio channel when acmod indicates two independent channels (dual mono, 1+1 mode).

2.3.2.24. copyrightb - Copyright bit - 1 bit

If this bit has a value of 1, the information in the bitstream is indicated as protected by copyright. It has a value of 0 if the information is not indicated as protected.

2.3.2.25. origbs - Original Bitstream - 1 bit

This bit has a value of 1 if this is an original bitstream. This bit has a value of 0 if this is a copy of an original bitstream.

2.3.2.26. timecod1e, timcode2e - Time Code (First and Second) Halves Exists - 2 bits

These fields indicate whether a time code follows in the bitstream. The time code can have a resolution of 1/64th of a frame (one frame = 1/30th of a second). Since only the high resolution portion of the time code is needed for fine

synchronization, the 28 bit time code is broken into two 14 bit halves. The low resolution first half represents the code in 8 second increments up to 24 hrs. The high resolution second half represents the code in 1/64th frame increments up to 8 seconds.

timecod2e,timecod1e	time code present
0,0	not present
0,1	first half (14 bits) present
1,0	second half (14 bits) present
1,1	both halves (28 bits) present

2.3.2.27. timecod1 - Time Code First Half - 14 bits

The first 5 bits of this 14 bit field represent the time in hours, with valid values of 0-23. The next 6 bits represent the time in minutes, with valid values of 0-59. The final 3 bits represents the time in 8 second segments, with valid values of 0-7 (representing 0, 8, 16, ... 56 seconds).

2.3.2.28. timecod2 - Time Code Second Half - 14 bits

The first 3 bits of this 14 bit field represent the time in seconds, with valid values from 0-7 (representing 0-7 seconds). The next 5 bits represents the time in frames, with valid values from 0-29. The final 6 bits represents 1/64 fractions of a frame, with valid values from 0-63.

2.3.2.29. addbside - Additional Bitstream Information Exists - 1 bit

If this bit has a value of 1 there is additional bitstream information the length of which is indicated by the next field. If this bit has a value of 0, there is no additional bitstream information (except for the optional bsicrc).

2.3.2.30. addbsil - Additional Bitstream Information Length - 6 bits

This 6 bit code indicates the length in bytes of the additional bitstream information. Valid range is 0-63, indicating 1-64 additional bytes, respectively. The decoder may skip over this number of bytes in the data stream.

2.3.2.31. addbsi - Additional Bitstream Information - (addbsil+1)×8 bits

This field contains 1 to 64 bytes of any additional information to be included with the bitstream information structure.

2.3.3. audblk - Audio Block

2.3.3.1. blksw[n] - Block Switch Flag - 1-bit wide array

This is a 1-bit wide array of length equal to nfchans (valid lengths are 1 to 5). Each array element indicates the block size of the corresponding channel. A 0 or 1 value indicates a block size of 512 or 256, respectively.

2.3.3.2. dithflag[n] - Dither Flag - 1-bit wide array

This is a 1-bit wide array of length equal to nfchans (valid lengths are 1 to 5). A value of 1 indicates that the decoder should apply dither in the corresponding channel.

2.3.3.3. dynrnge - Dynamic Range Gain Word Exists - 1 bit

If this bit is a 1, the Dynamic Range Gain Word follows in the bitstream. If it is 0, the gain word is not present, and the previous value is reused, except for block 0 of a frame where if the control word is not present the current value is set to 0.

2.3.3.4. dynrng - Dynamic Range Gain Word - 8 bits (present only if dynrnge = 1)

This encoder-generated gain word is applied to scale the reproduced audio as described in section 8.

2.3.3.5. dynrng2e - Dynamic Range Gain Word Exists, Ch2 - 1 bit

If this bit is a 1, the Dynamic Range Gain Word for Ch2 follows in the bitstream. If it is 0, the gain word is not present.

2.3.3.6. dynrng2 - Dynamic Range Gain Word, Ch2 - 8 bits (present only if dynrng2e = 1)

This encoder-generated gain word is applied to scale the reproduced audio of Ch2, as described in section 8.

2.3.3.7. cplstre - Coupling Strategy Exists - 1 bit

If this bit is a 1, the Coupling In Use bit follows in the bitstream. If it is 0, coupling infomation is not present, and coupling parameters hold to current state.

2.3.3.8. cplinu - Coupling In Use - 1 bit

If this bit is a 1, coupling is currently being utilized, and coupling parameters follow. If it is 0, coupling is not being utilized (all channels are independent).

2.3.3.9. chincpl[n] - Channel In Coupling - 1-bit wide array

This is a 1-bit wide array of length equal to nfchans (valid lengths are 2 to 5, since there is no coupling in the 1/0 audio coding mode). An array element equaling 1 indicates that the corresponding channel is a coupled channel.

2.3.3.10. phsflginu - Phase Flags In Use - 1 bit

If this bit is a 1 (defined for 2/0 mode only), the phase flag array is included with coupling coordinate information.

2.3.3.11. cplbegf - Coupling Begin Frequency Code - 4 bits

This 4-bit binary code indicates the lower band edge of the coupling channel by subband (see subband structure in the channel coupling section).

2.3.3.12. cplendf - Coupling End Frequency Code - 4 bits

This 4-bit binary code indicates the upper band edge of the coupling channel. Inclusive upper subband number equals cplendf + 2.

2.3.3.13. cplbndstrc[i] - Coupling Band Structure - 1-bit wide array

Each bit in the array corresponds to a specific coupling subband in ascending frequency, where the first bit corresponds to the subband at cplbegf+1. The length of this array may be computed as the number of subbands in the coupling channel minus one, given by the formula, (cplendf+2-cplbegf). Valid array lengths are 0 to 16 (if there is only one coupling subband, the length of this array is zero and no bits are sent).

This array is used to determine whether a new coupling coordinate is sent for each coupling subband, or whether the previous subband's coordinate is reused. If the bit is a 0, a new coupling coordinate is sent in the data stream for the indicated subband. If the bit is a 1, the coupling coordinate for the next lower frequency subband in the same channel is reused. Coupling coordinates for the first subband in each channel are always sent.

The number of distinct coupling subbands, ncplbnd, may be computed based on this array as ((cplendf+3-cplbegf) - (cplbndstrc[0] + ... + cplbndstrc[cplendf+1-cplbegf])).

2.3.3.14. cplcoe[n] - Coupling Coordinates Exist - 1-bit wide array

This array's length equals the number of coupled channels, (valid lengths are 2 to 5). If a bit is 1, the coupling coordinates for the corresponding channel exist and follow in the bitstream. If a bit is 0, the previous coupling coordinates are reused.

2.3.3.15. mstrcplco[n] - Master Coupling Coordinate - 2-bit wide array

This array's length equals the number of coupled channels, (valid lengths are 2 to 5). This parameter establishes a gain offset for the coupling coordinates for the given channel. Each 2-bit word is coded as follows:

mstrcplco[n]	cplco[n][i] gain multiplier
00	1
01	1/8
10	$1/8^2$
11	$1/8^3$

2.3.3.16. cplcoexp[n][i] - Coupling Coordinate Exponent - 4-bit wide array (one for each coupled channel)

This array's length equals ncplbnd. Each coupled channel has a cplcoexp[n][i] array, which contains the exponents of the gain multiplier to be used by the decoder for restoring the amplitudes of each coupled channel's individual high frequency coefficients on coupled subband group boundaries.

2.3.3.17. cplco[n][i] - Coupling Coordinate - 4-bit wide array (one for each coupled channel)

This array's length equals ncplbnd. Each coupled channel has a cplco[n][i] array, which contains the mantissas of the gain multiplier to be used by the decoder for restoring the amplitudes of each coupled channel's individual high frequency coefficients on coupled subband group boundaries.

2.3.3.18. phsflg[i] - Phase Flag - 1-bit wide array

This array's length equals ncplbnd. Each bit in this array indicates whether the corresponding coupled band in the right channel is phase inverted with respect to the left channel in the 2/0 audio coding mode. The flags correspond to the coupled bands in ascending frequency.

2.3.3.19. rematstr - Rematrixing Strategy - 1 bit

If this bit is a 1, then new rematrix flags are present in the bitstream. If it is 0, rematrix flags are not present, and the previous values should be reused. The rematstr parameter is present only in the 2/0 audio coding mode.

2.3.3.20. rematflg[n] - Rematrix Flag - 1-bit wide array

Each bit in this array indicates whether the transform coefficients in each matrixing band is coded as sums and differences. If the bit is a 1, then (Left + Right) and (Left - Right) are coded, if the bit is 0, Left and Right are coded separately. This operation is performed independently on separate frequency bands (see rematrixing section for band definitions). This array's length equals the number of rematrixing bands, and depends on cplinu and cplbegf as follows:

condition	# of rematrixing bands
cplinu == 0	4
(cplinu == 1) && (cplbegf > 2)	4
(cplinu == 1) && (2 ≥ cplbegf > 0)	3
(cplinu == 1) && (cplbegf == 0)	2

2.3.3.21. cplexpstr - Coupling Exponent Strategy - 2 bits

This code indicates the method of exponent coding that is used for the coupling channel as defined in the following table: (See exponent coding section for explanation of each strategy.)

cplexpstr	exponent strategy
00	reuse prior exponents
01	D15
10	D25
11	D45

2.3.3.22. chexpstr[n] - Channel Exponent Strategy - 2-bit wide array

This array's length equals nfchans. Each element indicates the method of exponent coding used for the corresponding full-bandwidth channel. The 2-bit code states are the same as those for cplexpstr, defined above.

2.3.3.23. lfeexpstr - Low Frequency Effects Channel Exponent Strategy - 1 bit

If this bit is a 1, the exponent coding strategy for the lfe channel is D15. A value of 0 indicates reuse of the prior exponents.

2.3.3.24. chbwcod[n] - Channel Bandwidth Code - 3 to 6 bits

The length of this code depends on the exponent coding strategy for a given channel and whether or not the channel is in coupling. The chbwcod[n] parameter defines the upper band edge of full-bandwidth channels which are not coupled channels. (See exponent coding section for definition of each code.)

2.3.3.25. cplabsexp - Coupling Absolute Exponent - 4 bits

This code indicates a reference point to begin decoding the differential exponents for the coupling channel.

2.3.3.26. cplexps[i] - Coupling Exponents - 7-bit wide array

This array's length equals ncplgrps. Each code indicates the value of 3, 6, or, 12 differentially-coded exponents for the case of D15, D25, or D45 coding, respectively.

2.3.3.27. exps[n][i] - Channel Exponents - 7-bit wide array (one for each full-bandwidth channel)

This array's length equals nchgrps. The first element in each array is a 4-bit absolute starting value, and all subsequent elements are the 7-bit differentially coded exponents.

2.3.3.28. gainrng[n] - Channel Gain Range Code - 2-bit wide array

This array's length equals nfchans. Each 2-bit code is used to determine a block floating-point shift count for the inverse TDAC transform.

2.3.3.29. lfeexps[i] - Low Frequency Effects Channel Exponents - 7-bit wide array

This array's length equals nlfegrps. The first element in each array is a 4-bit absolute starting value, and all subsequent elements are the 7-bit differentially coded exponents.

2.3.3.30. baie - Bit Allocation Information Exists - 1 bit

If this bit is a 1, then five separate fields (totaling 13 bits) follow in the bitstream. Each field gives additional instuctions to the decoder for the bit allocation process. If this bit is a 0, these fields do not exist.

2.3.3.31. sdcycod - Slow Decay Code - 2 bits

This is a 2-bit table pointer generated by the encoder used for adjusting the slow decay parameter in the decode bit allocation process.

2.3.3.32. fdcycod - Fast Decay Code - 2 bits

This is a 2-bit table pointer generated by the encoder used for adjusting the fast decay parameter in the decode bit allocation process.

2.3.3.33. sgaincod - Slow Gain Code - 2 bits

This is a 2-bit table pointer generated by the encoder used for adjusting the gain parameter in the decode bit allocation process.

2.3.3.34. dbpbcod - dB Per Bit Code - 2 bits

This 2-bit code indicates the effective value used in the masking curve calculation in the decoder's bit allocation process.

dbpbcod	knee value
00	off
01	-60 dB
10	-36 dB
11	-12 dB

2.3.3.35. floorcod - Masking Floor Code - 3 bits

This 3-bit code is used in the decoder's bit allocation process.

floorcod	value
000	0 dB

001	-3 dB
010	-6 dB
011	-9 dB
100	-12 dB
101	-15 dB
110	-24 dB
111	off

2.3.3.36. snroffste - SNR Offset Exists - 1 bit

If this bit has a value of 1, SNR offset information follows in the bit stream. If this bit has a value of 0, SNR offset information does not follow, and the previously transmitted values should be used for this block.

2.3.3.37. csnroffst - Coarse SNR Offset - 6 bits

This 6-bit code is used by the bit allocation process in conjunction with the fine SNR offset parameters to determine the absolute offset for each exponent set. The code is mapped to +72 to -24 dB in 1.5 dB steps.

2.3.3.38. cplfsnroffst - Coupling Fine SNR Offset - 4 bits

4-bit code mapped to 0 to 6 dB in 0.375 dB steps.

2.3.3.39. cplfgaincod - Coupling Fast Gain Code - 3 bits

This is a 3-bit table pointer generated by the encoder.

2.3.3.40. fsnroffst[n] - Channel Fine SNR Offset - 4-bit wide array

This array's length equals nfchans. Each element is a 4-bit code mapped to 0 to 6 dB in 0.375 dB steps.

2.3.3.41. fgaincod[n] - Channel Fast Gain Code - 3-bit wide array

This array's length equals nfchans. Each element is a 3-bit table pointer generated by the encoder.

2.3.3.42. lfefsnroffst - Low Frequency Effects Channel Fine SNR Offset - 4 bits

4-bit code mapped to 0 to 6 dB in 0.375 dB steps.

2.3.3.43. lfefgaincod - Low Frequency Effects Channel Fast Gain Code - 3 bits

This is a 3-bit table pointer generated by the encoder.

2.3.3.44. cplleake - Coupling Leak Initialization Exists - 1 bit

If this bit is a 1, leak initialization parameters follow in the bit stream. If this bit is a 0, the previously transmitted values still apply.

2.3.3.45. cplfleak - Coupling Fast Leak Initialization - 3 bits

This 3-bit code gives the fast leak initialization value for the coupling channel's excitation function calculation.

2.3.3.46. cplsleak - Coupling Slow Leak Initialization - 3 bits

This 3-bit code gives the slow leak initialization value for the coupling channel's excitation function calculation.

2.3.3.47. deltbaie - Delta Bit Allocation Information Exists - 1 bit

If this bit is a 1, some delta bit allocation information follows in the bit stream. If this bit is a 0, the previously transmitted delta bit allocation information still applies.

2.3.3.48. cpldeltbae - Coupling Delta Bit Allocation Exists - 2 bits

This 2-bit code indicates the delta bit allocation strategy for the coupling channel, defined as follows:

cpldeltbae, deltbae	code
00	reuse previous state
01	new info follows
10	perform no delta alloc
11	reserved

2.3.3.49. deltbae[n] - Coupling Delta Bit Allocation Exists - 2-bit wide array

Each 2-bit code indicates the delta bit allocation strategy for the corresponding full-bandwidth channel, as defined above.

2.3.3.50. cpldeltnseg - Coupling Delta Bit Allocation Number of Segments - 3 bits

This 3-bit code indicates the number of delta bit allocation segments that exist for the coupling channel.

2.3.3.51. cpldeltoffst[i] - Coupling Delta Bit Allocation Offset - 5-bit wide array

The first 5-bit code indicates the offset in the number of bit allocation bands from the start of the coupling channel to the beginning of the first delta segment. Subsequent codes indicate the offset from the previous delta segment end point.

2.3.3.52. cpldeltlen[i] - Coupling Delta Bit Allocation Length - 4-bit wide array

Each 4-bit code indicates the number of bit allocation bands that the corresponding segment spans.

2.3.3.53. cpldeltba[i] - Coupling Delta Bit Allocation - 3-bit wide array

Each 3-bit code indicates an adjustment to the default masking curve computed in the decoder. The deltas are coded as follows:

cpldeltba, deltba	code
000	-24 dB
001	-18 dB
010	-12 dB
011	-6 dB
100	+6 dB
101	+12 dB

110	+18 dB
111	+24 dB

2.3.3.54. deltnseg[n] - Channel Delta Bit Allocation Number of Segments - 3-bit wide array (each full bw chan)

Each 3-bit code indicates the number of delta bit allocation segments that exist for the corresponding channel.

2.3.3.55. deltoffst[n][j] - Channel Delta Bit Allocation Offset - 5-bit wide array (each full bw chan)

The first 5-bit code indicates the offset in the number of bit allocation bands from the start of the corresponding channel to the beginning of the first delta segment. Subsequent codes indicate the offset from the previous delta segment end point.

2.3.3.56. deltlen[n][j] - Channel Delta Bit Allocation Length - 4-bit wide array (each full bw chan)

Each 4-bit code indicates the number of bit allocation bands that the corresponding segment spans.

2.3.3.57. deltba[n][j] - Channel Delta Bit Allocation - 3-bit wide array (each full bw chan)

Each 3-bit delta code indicates an adjustment to the default masking curve computed in the decoder. The delta coding is defined above.

2.3.3.58. skiple - Skip Length Exists - 1 bit

If this bit is a 1, then the skipl parameter follows in the bit stream. If this bit is a 0, skipl does not exist.

2.3.3.59. skipl - Skip Length - 9 bits

This 9-bit code indicates the number of bytes to skip (ignore) after the mantissas of the current audio block.

2.3.3.60. cplmant[i] - Coupling Mantissas - 0-16-bit wide array

This array's length equals ncplmant, and it contains the quantized transform coefficient mantissas for the coupling channel.

2.3.3.61. chmant[n][i] - Channel Mantissas - 0-16-bit wide array (for each full bw chan)

This array's length equals nchmant, and it contains the quantized transform coefficient mantissas for channel n.

2.3.3.62. lfemant[i] - Low Frequency Effects Channel Mantissas - 0-16-bit wide array

This array's length equals nlfemant, and it contains the quantized transform coefficient mantissas for the lfe channel.

2.3.4. auxdata - Auxiliary Data Field

2.3.4.1. auxbits - Auxiliary Data Bits - nauxbits

This field contains auxiliary data.

2.3.5. errorcheck - Frame Error Detection Field

2.3.5.1. crcrsv - CRC Reserved Bit - 1 bit

Reserved for use in specific applications to ensure CRC2 will not be equal to the sync word. Use of this bit is optional by encoders.

2.3.5.2. crc2 - Cyclic Redundancy Check 2 - 16 bits

The 16 bit CRC applies to the last 3/8 of the frame, or optionally the entire frame.

Table 2.1: Frame Size Code Table

frmsizecod	nominal bitrate	fs = 32 kHz words/syncframe	fs = 44.1 kHz words/syncframe	fs = 48 kHz words/syncframe
000000 (0)	32 kb/s	96	69	64
000001 (0)	32 kb/s	96	70	64
000010 (1)	40 kb/s	120	87	80
000011 (1)	40 kb/s	120	88	80
000100 (2)	48 kb/s	144	104	96
000101 (2)	48 kb/s	144	105	96
000110 (3)	56 kb/s	118	121	112
000111 (3)	56 kb/s	118	122	112
001000 (4)	64 kb/s	192	139	128
001001 (4)	64 kb/s	192	140	128
001010 (5)	80 kb/s	240	174	160
001011 (5)	80 kb/s	240	175	160
001100 (6)	96 kb/s	288	208	192
001101 (6)	96 kb/s	288	209	192
001110 (7)	112 kb/s	336	243	224
001111 (7)	112 kb/s	336	244	224
010000 (8)	128 kb/s	384	278	256
010001 (8)	128 kb/s	384	279	256
010010 (9)	160 kb/s	480	348	320
010011 (9)	160 kb/s	480	349	320
010100 (10)	192 kb/s	576	417	384
010101 (10)	192 kb/s	576	418	384
010110 (11)	224 kb/s	672	487	448
010111 (11)	224 kb/s	672	488	448
011000 (12)	256 kb/s	768	557	512
011001 (12)	256 kb/s	768	558	512
011010 (13)	320 kb/s	960	696	640
011011 (13)	320 kb/s	960	697	640

011100 (14)	384 kb/s	1152	835	768
011101 (14)	384 kb/s	1152	836	768
011110 (15)	448 kb/s	1344	975	896
011111 (15)	448 kb/s	1344	976	896
100000 (16)	512 kb/s	1535	1114	1024
100001 (16)	512 kb/s	1536	1115	1024
100010 (17)	576 kb/s	1728	1253	1152
100011 (17)	576 kb/s	1728	1254	1152
100100 (18)	640 kb/s	1920	1393	1280
100101 (18)	640 kb/s	1920	1394	1280

Table 2.2: Language Code Table

langcod	language	langcod	language	langcod	language	langcod	language
00	unknown/not applicable	20	Polish	40	background sound/clean feed	60	Moldavian
01	Albanian	21	Portuguese	41		61	Malaysian
02	Breton	22	Romanian	42		62	Malagasay
03	Catalan	23	Romansh	43		63	Macedonian
04	Croatian	24	Serbian	44		64	Laotian
05	Welsh	25	Slovak	45	Zulu	65	Korean
06	Czech	26	Slovene	46	Vietnamese	66	Khmer
07	Danish	27	Finnish	47	Uzbek	67	Kazakh
08	German	28	Swedish	48	Urdu	68	Kannada
09	English	29	Turkish	49	Ukrainian	69	Japanese
0A	Spanish	2A	Flemish	4A	Thai	6A	Indonesian
0B	Esperanto	2B	Walloon	4B	Telugu	6B	Hindi
0C	Estonian	2C		4C	Tatar	6C	Hebrew
0D	Basque	2D		4D	Tamil	6D	Hausa
0E	Faroese	2E		4E	Tadzhik	6E	Gurani
0F	French	2F		4F	Swahili	6F	Gujurati
10	Frisian	30	reserved for natl assignment	50	Sranan Tongo	70	Greek
11	Irish	31	"	51	Somali	71	Georgian
12	Gaelic	32	"	52	Sinhalese	72	Fulani
13	Galician	33	"	53	Shona	73	Dari
14	Icelandic	34	"	54	Serbo-Croat	74	Churash
15	Italian	35	"	55	Ruthenian	75	Chinese
16	Lappish	36	"	56	Russian	76	Burmese
17	Latin	37	"	57	Ruthenian	77	Bulgarian
18	Latvian	38	"	58	Russian	78	Bengali
19	Luxembourgian	39	"	59	Punjabi	79	Belorussian

1A	Lithuanian	3A	"	5A	Persian	7A	Bambora
1B	Hungarian	3B	"	5B	Papamiento	7B	Azerbijani
1C	Maltese	3C	"	5C	Oriya	7C	Assamese
1D	Dutch	3D	"	5D	Nepali	7D	Armenian
1E	Norwegian	3E	"	5E	Ndebele	7E	Arabic
1F	Occitan	3F	"	5F	Marathi	7F	Amharic

3. EXPONENT CODING

3.1. Overview

AC-3 bit streams contain coded exponents for all independent channels, all coupled channels, and for the coupling and low frequency effects channels, when they are enabled. Since audio information is not shared across frames, block 0 of every frame will include new exponents for every channel. Exponent information may be shared across blocks within a frame, so blocks 1 through 5 may reuse exponents from previous blocks.

AC-3 uses a differential coding strategy in which the exponents for a channel are differentially coded across frequency. The differential exponents are combined into groups in the audio block. The grouping is done by one of three methods, D15, D25, or D45, referred to as the exponent strategy. The number of grouped differential exponents placed in the audio block for a particular channel depends on the exponent strategy along with the frequency bandwidth information for that channel.

An AC-3 audio block contains two sets of fields with exponent information. The first set defines the exponent coding strategy for each channel, and the second set contains the coded exponents for channels requiring new exponents. For independent channels frequency bandwidth information is included along with the exponent strategy fields. For coupled channels and the coupling channel the frequency information is found in the set of coupling strategy fields.

3.2. Exponent Strategy

Exponent strategy information is included in every AC-3 audio block. Audio information is never shared across frames so block 0 will always contain a strategy indication (D15, D25, or D45) for each channel. Blocks 1 through 5 may indicate reuse prior exponents.

The three exponent coding strategies provide a tradeoff between data cost and frequency resolution, the D15 mode providing the most frequency resolution, and the D45 mode providing the most data savings. In all three modes differential exponents are combined into 7 bit words when coded in an audio block. The main difference between the modes is how many differential exponents are combined together.

The differential exponents are generated by the encoder by converting absolute exponents to values representing 6 dB steps from the previous exponent. Absolute exponents are 5 bit values with 0 indicating the maximum value and each count up representing one half of the previous value. Each differential exponent can take on one of five values: -2, -1, 0, +1, +2. This allows deltas of up to +/- 12 dB between exponents. These five values are mapped into the values 0, 1, 2, 3, 4 before being grouped, as shown in the following table and formulas:

diff exp	mapped value
+ 2	4
+ 1	3
0	2
- 1	1
- 2	0