Using .XA In PlayStation Development

SCEE Technical Note

June 1998

Overview

The PlayStation can be used to access CD-ROM XA file formats. This file format can be very versatile and can be used for numerous techniques. This technical note will attempt to produce a set of examples that could be incorporated into your PlayStation titles.

.XA Formats

The one problem with using .XA files is the numerous ways of presenting the files, the issues surrounding certain formats and the differing ways of accessing and controlling the data from CD. The following is a list of possible combinations that can make up an .XA file; for each item in the list this article will go through a series of steps showing how to create and use the technique.

1. Playing an .XA file. Normally this is a PCM (raw audio data) or .WAV file converted to .XA.
2. Playing Interleaved .XA files. This file type normally consists of a number of separate audio files or channels interleaved together. In normal operation the file is either completely made up of audio samples or alternatively one channel is used as a data channel. This file is of a different format to a PlayStation .STR file.
3. PlayStation .STR with audio, is a type of .XA file, mainly due to the fact that the audio needs to be .XA so that during playback the audio is passed directly through the sound system without the CPU or programmer having to do anything. It is possible to interleave more than one audio track together with a video stream to allow one video sequence to have two different audio tracks.
4. Interleaving streams. Video streams take a little while to initialise and set up, this technique interleaves multiple streams together to allow quick switching between video streams. This could be used to switch quickly to different endings from the same stream. Alternatively more than one stream can use the same .XA track.
5. Linking Streams together

Part 1 - Using .XA files On Their Own

Audio data converted to .XA files can be played without programmer or CPU intervention, this data is similar to CD-DA except that the seek time is much faster than CD-DA. The playing of certain .XA files can be done at double speed, meaning that the CD system does not have to change speed to read data and play audio during game play. Another side effect is that if you use .XA as your background music then users cannot play these audio tracks on their CD players as standard DA tracks.

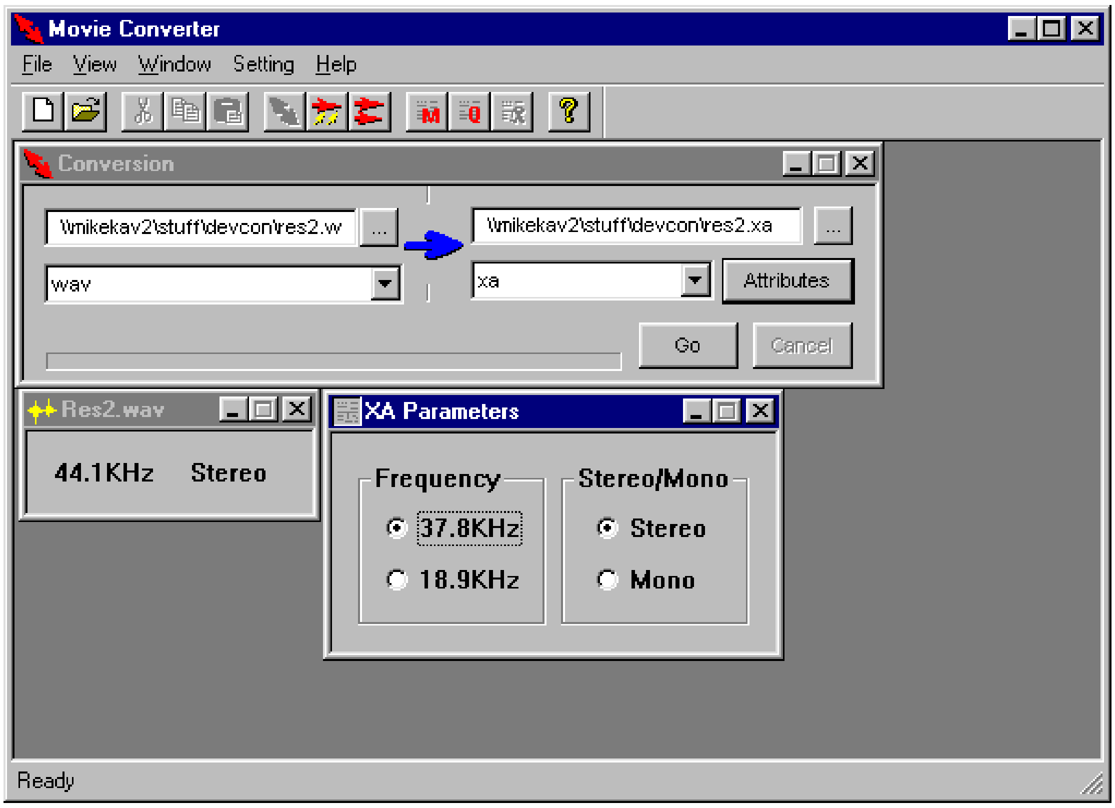
Part 1 - Creation

The following example uses two .XA files. One a short, seven-second-speech sample and the other a two-minute music track.

Initially the sound source is grabbed at 44.1kHz stereo, 16-bit sample. The following are the initial properties of each file.

Filename	Res2.wav	Breath.wav
Media Length	7.00 seconds	2 minutes 0.00 seconds
Audio Format	PCM,44.1 kHz, 16 Bit, Stereo	PCM,441 kHz, 16 bit, Stereo
File Length	1,234,880 bytes	21,168,052 bytes

These samples are downsampled and converted to .XA format by MovConv 3.1, resulting in the following file properties:


Movie Converter
File View Window Setting Help
Conversion
\\mikekav2\stuff\devcon\res2.w
\\mikekav2\stuff\devcon\res2.xa
Attributes
Go
Cancel
Ready
XA Parameters

wav → xa

Res2.wav: 44.1KHz Stereo

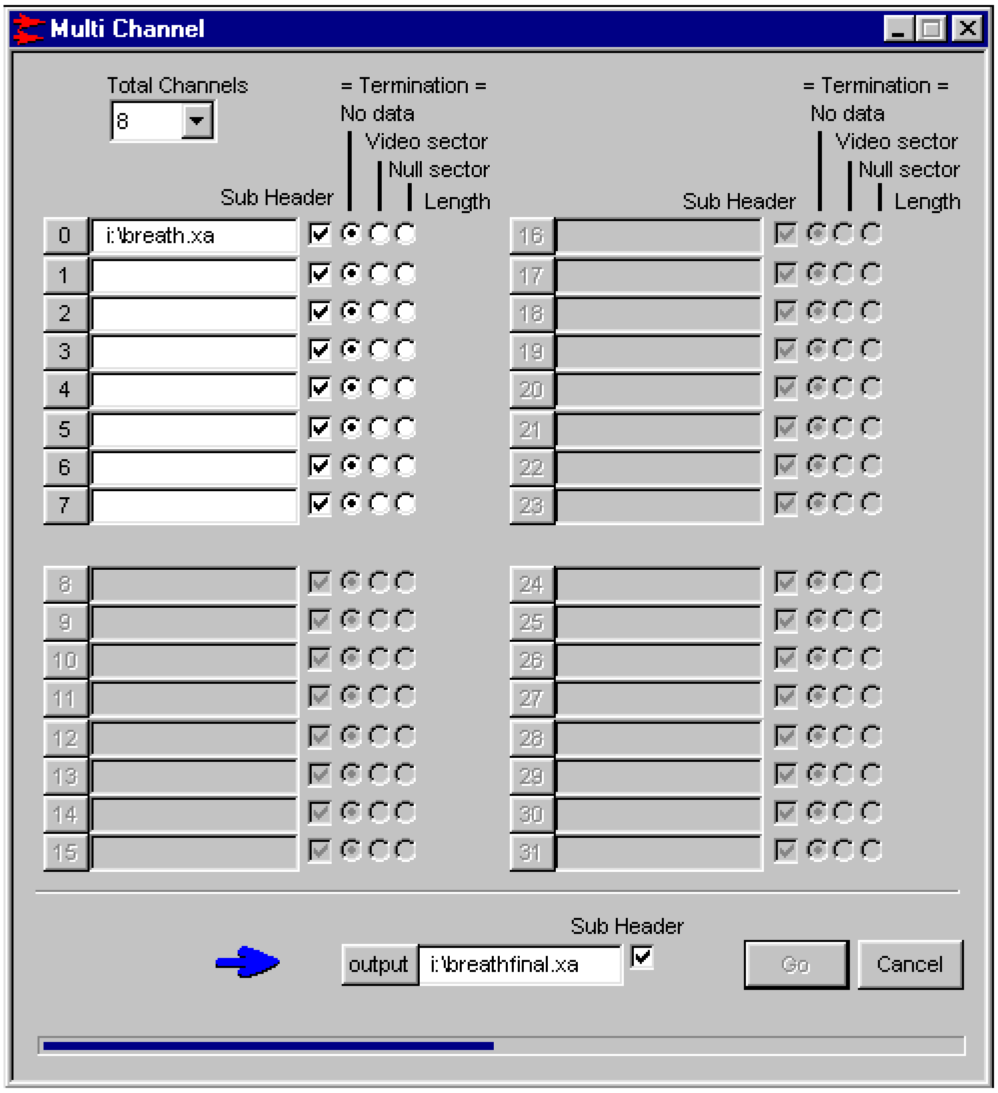
XA Parameters:
Frequency: [✓ 37.8KHz] 18.9KHz
Stereo/Mono: [✓ Stereo] Mono

Filename	Res2.xa	Breath.xa
Media Length	7.00 seconds	2 minutes 0.00 seconds
Audio Format	XA,37.8 kHz, Stereo	XA,37.8 kHz, Stereo
File Length	308,352 bytes	5,256,000 bytes

These files now need to be converted to the correct format based on the playback speed and type. The following table shows the acceptable values for the .XA format.

Playback speed	Sampling frequency	Stereo / monaural	Data/gap ratio	Total number of channels
Double speed	37.8KHz	Stereo	1 sector / 7 sectors	8
Double speed	37.8KHz	Monaural	1 sector / 15 sectors	16
Double speed	18.9KHz	Stereo	1 sector / 15 sectors	16
Double speed	18.9KHz	Monaural	1 sector / 31 sectors	32
Standard speed	37.8KHz	Stereo	1 sector / 3 sectors	4
Standard speed	37.8KHz	Monaural	1 sector / 7 sectors	8
Standard speed	18.9KHz	Stereo	1 sector / 7 sectors	8
Standard speed	18.9KHz	Monaural	1 sector / 15 sectors	16

As we are just intending to use the .XA samples without interleaving the best choice from the table is row one. This allows us to have double speed access using the values we want for the actual sample regarding quality and output type.



The following table shows the final .XA file properties:

Filename	Res2final.xa	Breathfinal.xa
Media Length	7.00 seconds	2 minutes 0.00 seconds
Audio Format	XA,37.8 kHz, Stereo	XA,37.8 kHz, Stereo
File Length	2,466,816 bytes	24,724,224 bytes

Part 1 - Building on to an Emulated Disk Image

The sample code used in the following examples will use both the PlayStation disk and any emulated image as data file storage only. The code will be designed to work on a development kit just by running the .CPE. The full code is available from the developers web site at:
https://www-s.playstation.co.uk/site2/ftp/developer/Sample_Code/samples/xatut.zip

The .CTI file that is used to build the image is very simple. The files are put on as XASource; this definition ensures that the emulator uses the sub-header information contained within the source files when building the image.

```
Disc CDROMXA_PSX
```

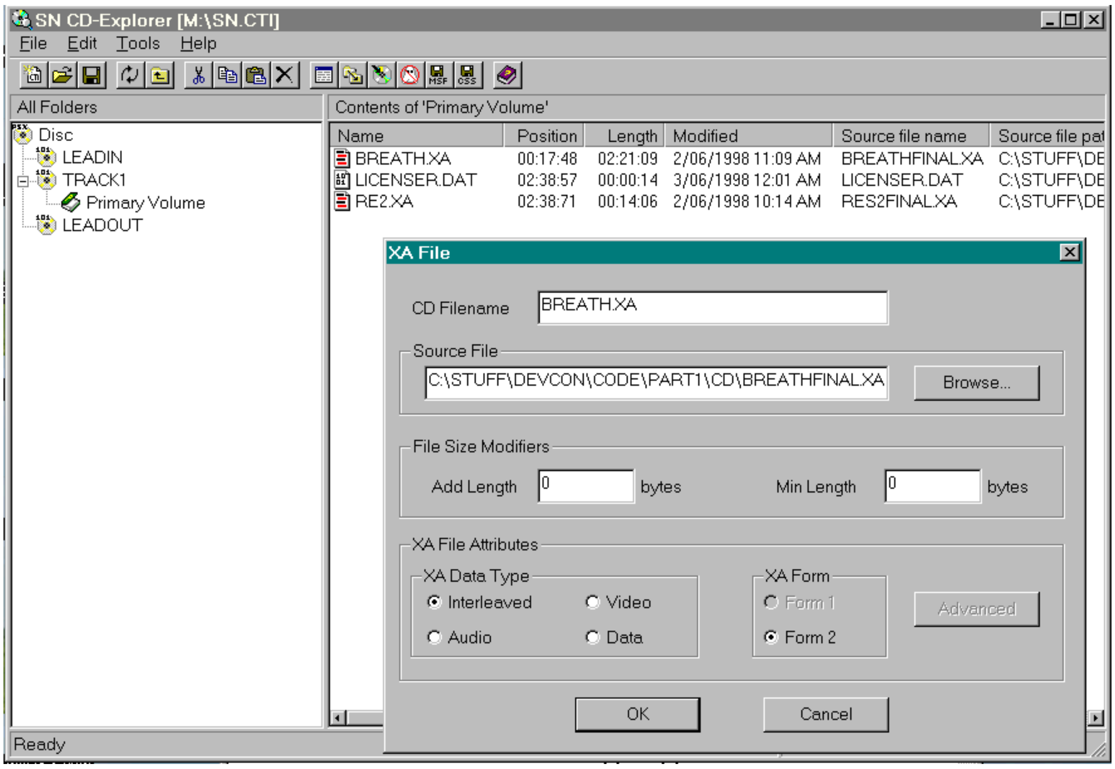
```
; LeadIn with 2 seconds of empty data
LeadIn XA
  Empty 150
  PostGap 150
EndTrack

Track XA
  Pause 150
  Volume ISO9660
    PrimaryVolume
      SystemIdentifier PLAYSTATION
      ApplicationIdentifier PLAYSTATION
      LPath
      OptionalLpath
      Mpath
      OptionalMpath

      Hierarchy

          File RES2.XA
            XASource i:\res2fi~1.xa
          EndFile

          File BREATH.XA
            XASource i:\breath~1.xa
          EndFile

      EndHierarchy
    EndPrimaryVolume
  EndVolume
EndTrack

LeadOut XA
  Empty 150
EndTrack

EndDisc
```

Then the image can be created using:

```
buildcd –s<id>:<partition> <.CTI filename>
```

Part 1 - Building on to a new Emulator Disk Image

To use the .XA files on the new SN emulator system simply drag the files from the source location onto the right hand pane of the SN CD-Explorer, right-click on the files and select the Interleaved option for the XA Data Type in the XA File Attributes section.



Contents of 'Primary Volume'

Name	Position	Length	Modified	Source file name	Source file pat
BREATH.XA	00:17:48	02:21:09	2/06/1998 11:09 AM	BREATHFINAL.XA	C:\STUFF\DE
LICENSER.DAT	02:38:57	00:00:14	3/06/1998 12:01 AM	LICENSER.DAT	C:\STUFF\DE
RE2.XA	02:38:71	00:14:06	2/06/1998 10:14 AM	RES2FINAL.XA	C:\STUFF\DE

XA File

CD Filename: BREATH.XA

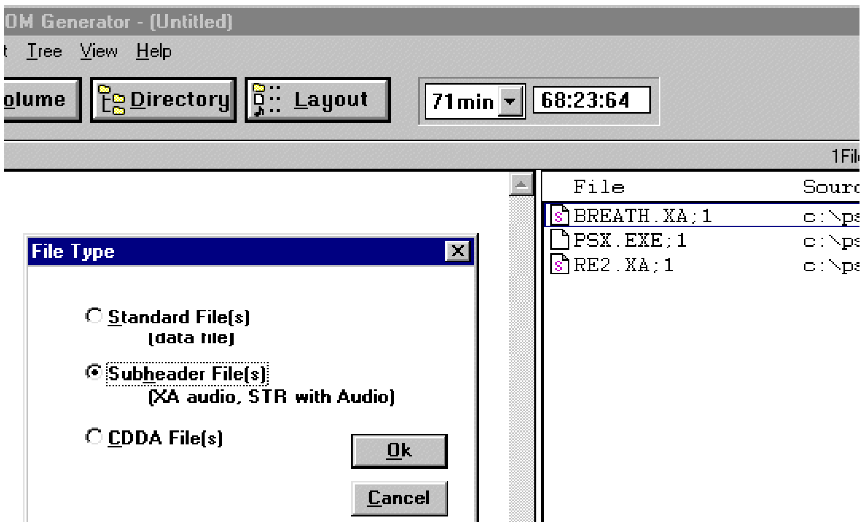
Source File: C:\STUFF\DEVCON\CODE\PART1\CD\BREATHFINAL.XA

File Size Modifiers: Add Length 0 bytes; Min Length 0 bytes

XA File Attributes
- XA Data Type: [✓ Interleaved] Video Audio Data
- XA Form: Form 1 [✓ Form 2]

Part 1 - Building On To A PlayStation Disk

To use the .XA created files on a gold disk use CD-Gen as normal, use the Directory View and the Put Files option to add the .XA files to the CD image. Then select the File Types button and select the <XA> option. This will set a small 's' in the file icon in the directory view, indicating that the file is an .XA sub-headed file and will be burned onto the CD accordingly.



File	Sourc
BREATH.XA;1	c:\ps
PSX.EXE;1	c:\ps
RE2.XA;1	c:\ps

File Type
- Standard File(s) (data file)
- [✓ Subheader File(s)] (XA audio, STR with Audio)
- CDDA File(s)

Code for playing the .XA

As with DA there are a few operations that are needed to make full use of the .XA file. These include the ability to play and stop the audio playback as well as some initialisation functions.
The full source for this example is on the developers web site:
https://www-s.playstation.co.uk/site2/ftp/developer/Sample_Code/samples/xatut.zip

File Initialisation

Before playing the .XA the files on the CD need to be initialised. In this example we only need to (1) check the .XA files are present and (2) calculate a start and end position on the CD. Note that if you are using a CD emulator the SECTOR_SIZE must be changed to 2336 as the CD emulator incorrectly returns the full XA sector size rather than a standard sector size of 2048 bytes.

NOTE: As from buildcd version 2.42 this SECTOR_SIZE bug is fixed so the SECTOR_SIZE can always be referred to as being 2048 bytes

```
#ifdef CD
#define SECTOR_SIZE 2048
#else
#define SECTOR_SIZE 2336
#endif

for(i=0;i<2;i++)
        {
        if(CdSearchFile(&fp, theXAFile[i].filename) == 0)
        {
                printf("%s: not found\n", theXAFile[fileNo].filename);
               return;
        }
        // get CD file start position
                theXAFile[i].startpos = CdPosToInt(&fp.pos);

        // get CD file end position, start pos + number of sectors -1
        theXAFile[i].endpos = theXAFile[i].startpos + (fp.size/SECTOR_SIZE) -1;
```

.XA Initialisation

All that is needed to set up .XA playback is a quick change of the mode. CdlModeSF specifies that sub-header filtering operation is on to allow the controller to play the correct audio channel and CdlModeSize1 is used so that we can track the sector number that is being played. This allows us to detect when the end of the sample is reached. A callback also needs to be hooked onto the CDReadyCallback(). This callback is detailed further on in the article.

```
        param[0] = CdlModeSpeed|CdlModeRT|CdlModeSF|CdlModeSize1;
        CdControlB(CdlSetmode, param, 0);
        CdControlF(CdlPause,0);

        return CdReadyCallback((CdlCB)cbready);
```

Shutdown

Again very simple, all that is needed is to reset the CD mode back to double speed only. As the CD mechanism is already at double speed no slow mechanical operation is required and the CD system just switches to no sub-header filtering.

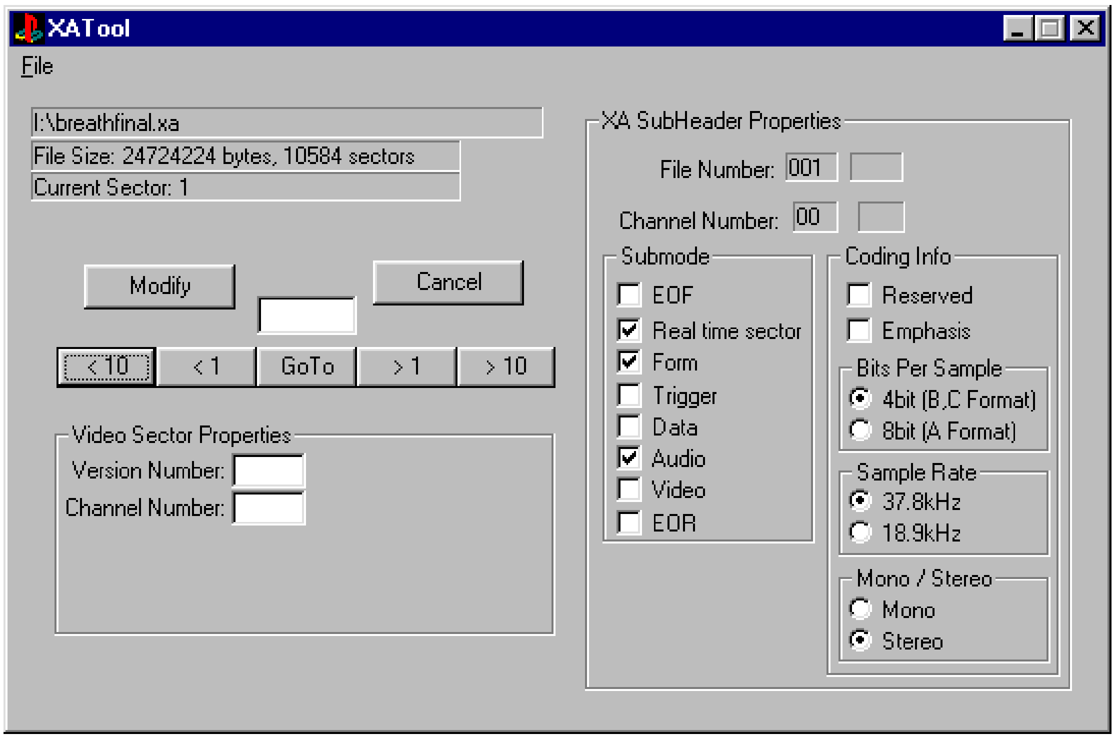
```
        CdControlF(CdlPause,0);
        param[0] = CdlModeSpeed;
        CdControlB(CdlSetmode, param, 0);
```

Playing

The playback of .XA involves the setting up of a filter; this specifies the file and channel number to actually read data from. The file and channel number are contained within the .XA sub-header information, which can be viewed on .XA files you have created that are still on your PC hard disk using XATOOL.EXE. This program can be found on the web site together with the source code for the example.

https://www-s.playstation.co.uk/site2/ftp/developer/Sample_Code/samples/xatut.zip

In this case viewing the .XA file shows that only one sector in every eight has both a channel number and a file number defined, the other sectors are null data sectors.



I:\breathfinal.xa
File Size: 24724224 bytes, 10584 sectors
Current Sector: 1

Video Sector Properties
Version Number:
Channel Number:

XA SubHeader Properties
File Number: 001
Channel Number: 00

Submode
[ ] EOF
[✓] Real time sector
[✓] Form
[ ] Trigger
[ ] Data
[✓] Audio
[ ] Video
[ ] EOR

Coding Info
[ ] Reserved
[ ] Emphasis
Bits Per Sample: [✓ 4bit (B,C Format)] 8bit (A Format)
Sample Rate: [✓ 37.8kHz] 18.9kHz
Mono / Stereo: Mono [✓ Stereo]

We need to set a filter to tell the CD system which sub-header to use, therefore our code needs to set a CdlFilter structure to file number one and channel zero. Once set we can play the .XA using CdlReadS using a start position of the file's location on the CD.

```
        CdlLOC  loc;
        CdlFILTER theFilter;

        theFilter.file=1;
        theFilter.chan=0;
        CdControlF(CdlSetfilter, (u_char *)&theFilter);

        // Starting position on CD
        CdIntToPos(startPos, &loc);
        CdControlF(CdlReadS, (u_char *)&loc);
```

Stopping

In this case the easiest way to stop the CD is to calculate the position of the end of the file and pass this in as the end position when choosing to play the sample. In the initialisation function for the .XA playback a CD callback can be set, which will occur every data sector. In this case seven out of eight sectors are data sectors. In the callback we can obtain the position of the CD as CdlModeSize1 returns the current sector number of the CD. Therefore within the callback all we simply do is retrieve this value and compare it against the calculated end position of the sample.

```
        if (intr == CdlDataReady)
        {
                CdGetSector((u_long*)buffer,585);
                currentPos = CdPosToInt( (CdlLOC *)buffer );
                if(currentPos >= theXAFile[fileNo].endpos)
                {
                      SsSetSerialVol(SS_SERIAL_A,0,0);
                      CdControlF(CdlPause,0);
                }
        }
```

Summary

Utilising .XA files this way is not the most effective use of the format, since for comparable quality, the converted file will be far larger. However the technique does give some benefits over CD-DA if the slight drop in quality is acceptable. In particular it allows you to switch quickly between files, and between playing music and loading. This is due to the fact that the CD motor speedup/slowdown is not used. It is also faster to seek between .XA samples than move from track to track using .DA.

Part 2 - Using Interleaved .XA Files

Various audio samples can be interleaved together to create an .XA file that acts as a sample bank. Basically this involves filling the empty data sectors that were shown in the first example with other useful data such as other audio samples.

There are two general methods of building interleaved .XA which affect the code involved in playback, you can either choose to terminate the sample data with video sectors or make the choice to leave one channel in the sample bank a blank data channel. The following four methods outline some ways of using the interleaved files.

Method 1 – Interleaving using a data channel. This is just a step up from the original example, with more channels taken up with audio data.

Method 2 – Filling all channels with audio, tracking end of sample with either video sectors or data sectors. If you prefer not to have a wasted data channel but fill all the channels with audio then this is the option for you.

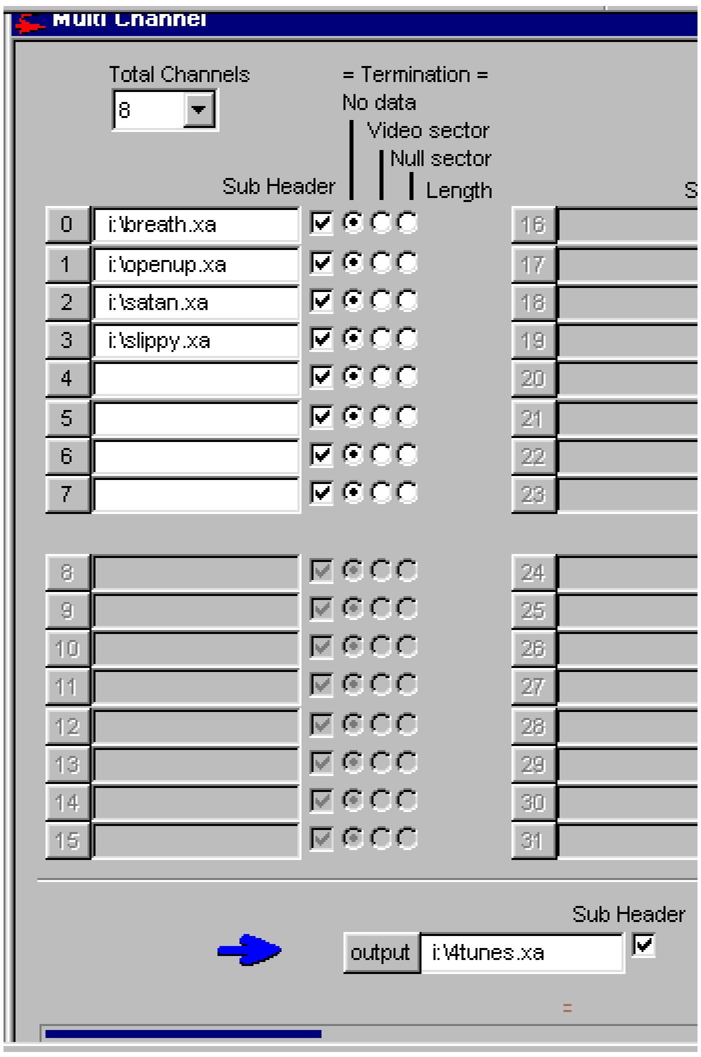
Method 3 – Interleaving using a data channel. This section details a generalised .XA player for use for test playback of .XA data created using the data channel method

Method 4 – Interleaving using a data channel for different sized samples and multiple samples in one channel. From creation of samples to playback of .XA

Method 1 - Creation

The initial samples can be built as in the first example, for this example four PCM,44.1 kHz, 16 bit. Stereo samples are to be converted from .WAV files to 37.8 kHz, Stereo .XA files.

The following interleaved file is then created, with the properties described above we still have four channels spare which we could fill with other audio tracks, as long as at least one channel is left blank for null data.


Multi Channel

Total Channels: 8

= Termination =

		Sub Header	No data	Video sector	Null sector	Length		
0	i:\breath.xa	✓	●	○	○		16	
1	i:\openup.xa	✓	●	○	○		17	
2	i:\satan.xa	✓	●	○	○		18	
3	i:\slippy.xa	✓	●	○	○		19	
4		✓	●	○	○		20	
5		✓	●	○	○		21	
6		✓	●	○	○		22	
7		✓	●	○	○		23	
8		✓	●	○	○		24	
9		✓	●	○	○		25	
10		✓	●	○	○		26	
11		✓	●	○	○		27	
12		✓	●	○	○		28	
13		✓	●	○	○		29	
14		✓	●	○	○		30	
15		✓	●	○	○		31	

output: i:\4tunes.xa — Sub Header ✓

Method 1 - Building on to both Emulators as an Emulated Disk Image

This file can be built onto a CD image in the same way as part one, remembering to specify the file as an .XA source file:

```
        File 4TUNES.XA
           XASource i:\4tunes.xa
        EndFile
```

Method 1 - Building On To A PlayStation Disk

Burn a CD using CD-Gen as with the first example.

Method 1 - Code for Playing The .XA

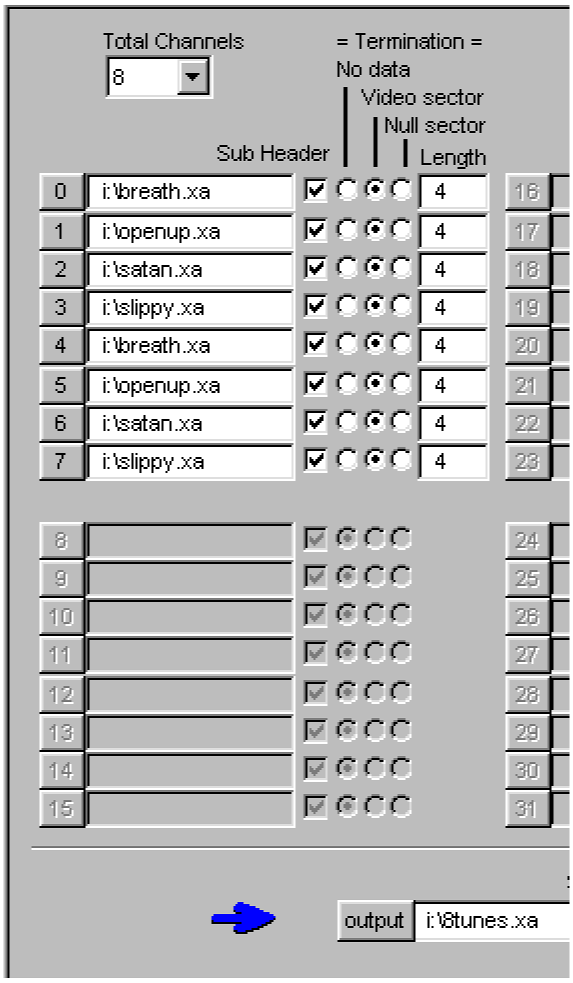
In this example all the tunes finish at the same time so we do not need to change the start and end positions, we only need to choose which channel to play. If the samples were of differing lengths we would need to calculate the lengths for each channel and pass the channel end position to the PlayXA() function.

The only other change is to add the setting of the .XA channel. This is achieved by modifying PlayXA() slightly so that the channel number is passed in rather than hard coded to be zero.

```
void PlayXA(int channel, int startPos, int endpos)
{

        CdlLOC  loc;
        CdlFILTER theFilter;

         // set volume
         SsSetSerialVol(SS_SERIAL_A,127,127);

        theFilter.file=1;
        theFilter.chan=channel;
        CdControlF(CdlSetfilter, (u_char *)&theFilter);

        // Starting position on CD
        CdIntToPos(startPos, &loc);
         currentPos=startPos;
        CdControlF(CdlReadS, (u_char *)&loc);
        return;
}
```

Method 2 - Creation

The initial samples are once again kept the same as the above example, however this time all the channels are filled with audio tracks. This time the channels are terminated by video sectors, rather than null data. See below:


Total Channels: 8

= Termination =

		Sub Header	No data	Video sector	Null sector	Length	
0	i:\breath.xa	✓		●		4	16
1	i:\openup.xa	✓		●		4	17
2	i:\satan.xa	✓		●		4	18
3	i:\slippy.xa	✓		●		4	19
4	i:\breath.xa	✓		●		4	20
5	i:\openup.xa	✓		●		4	21
6	i:\satan.xa	✓		●		4	22
7	i:\slippy.xa	✓		●		4	23
8		✓	●				24
9		✓	●				25
10		✓	●				26
11		✓	●				27
12		✓	●				28
13		✓	●				29
14		✓	●				30
15		✓	●				31

output: i:\8tunes.xa

Method 2 - Building on to both Emulators as an Emulated Disk Image

This file can be built onto a CD image in the same way as part 1 of this sample for both emulators.

Method 2 - Building On To A PlayStation Disk

Burn a CD using CD-Gen in the usual way.

Method 2 - Code for Playing The .XA

As the file no longer contains any data channels the CD system will not return a data sector until the end of the sample, where it will return the video sector that terminates the sample. If the samples are of different sizes then the shorter samples will return their data sectors while the longer samples are playing. Therefore to detect the end of the sample we need to check which video sector channel number is being returned within the data sector. The only code modification required is in the initialisation and the CDCallBack. Also for ease of use I have made the channel number global to allow testing in the callback.

Method 2 - Initialisation

As we no longer need to track the sector number the CD mode needs to be set appropriately:

```
        param[0] = CdlModeSpeed|CdlModeRT|CdlModeSF;
        CdControlB(CdlSetmode, param, 0);
        CdControlF(CdlPause,0);

        return CdReadyCallback((CdlCB)cbready);
```

Method 2 - Stopping

Now that the audio samples are terminated by video sectors we just need to check that the sector in the CD buffer is a video sector and then check to make sure it is the particular channel number we require. For this case we can assign a pointer to the start of the CD buffer like so:

u_long *chanAddress=(u_long *)&buffer[0];

Then all we need to do is check that we have the channel we want by checking the video sector header channel number

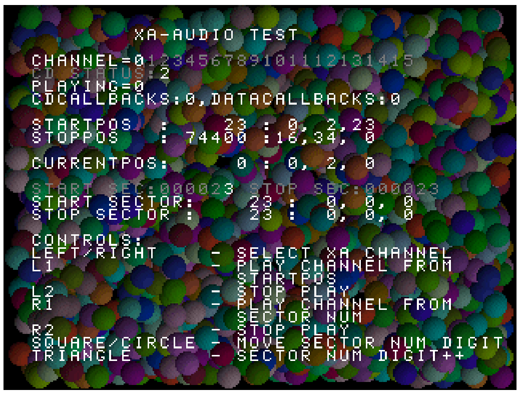
```
void cbready(int intr, u_char *result)
{
        if (intr == CdlDataReady)
        {
                CdGetSector((u_long *)buffer,8);

                ID = *(unsigned short *)(chanAddress+3);
                // video sector channel number format = 1CCCCC0000000001
                currentChannel = *((unsigned short *)(chanAddress+3)+1);
                currentChannel = (currentChannel&31744)>>10;

// video sector channel number format = 1CCCCC0000000001
                currentChannel = *((unsigned short *)(buffer+3)+1);
                currentChannel = (currentChannel&31744)>>10;

                // If this is a video sector then check that this is the channel
                // you want then stop playing the .XA sample
                if( (ID == 352) && (currentChannel == gChannel) )
                {
                        SsSetSerialVol(SS_SERIAL_A,0,0);
                                CdControlF(CdlPause,0);
                }

        }
}
```

Method 2 - Summary

Because this technique allows you to quickly switch from sample to sample, as well as being able to detect when the sample has finished playing, it is possible to use this technique for real-time speech effects ranging from just one comment to combining comments to provide sports style commentary.

Method 3 - Code

It is possible to use any .XA file that contains a data channel with the program in the Code\Part2\Method3 subdirectory. This program shows how to handle playback of .XA in this format. The program can play .XA from the start of the file to the finish by using the L1/L2 to start and stop the .XA playback. Left and right buttons select the channel number for playback and the circle and square buttons can be used together with the triangle and cross button to select a start and stop sector to be played back using R1. The program uses any data channels in an interleaved .XA file to track the current position the CD is within the file.



```
          XA-AUDIO TEST

CHANNEL=0123456789101112131415
CD STATUS:2
PLAYING=0
CDCALLBACKS:0,DATACALLBACKS:0

STARTPOS    :     23 : 0, 2,23
STOPPOS     :  74400 :16,34, 0

CURRENTPOS:        0 : 0, 2, 0

START SEC:000023 STOP SEC:000023
START SECTOR:      23 :   0, 0, 0
STOP SECTOR :      23 :   0, 0, 0

CONTROLS:
LEFT/RIGHT     - SELECT XA CHANNEL
L1             - PLAY CHANNEL FROM
                 STARTPOS
L2             - STOP PLAY
R1             - PLAY CHANNEL FROM
                 SECTOR NUM
R2             - STOP PLAY
SQUARE/CIRCLE  - MOVE SECTOR NUM DIGIT
TRIANGLE       - SECTOR NUM DIGIT++
```

Method 4 - Creation

The object of this section is to outline how to create and play your own interleaved .XA sample bank. In normal use you would require the .XA file to contain numerous samples spread over the various channels used. To track the position the CD is within the channel currently being played one or more channels need to be a blank data channel. This ensures we can use the playback routines detailed in the preceding sample. In this sample I shall use the same code from Method three to playback the .XA in the created sample bank.

The first step is to decide on a sample playback rate, checking the table on Page 2 I have decided to use the following settings:

Playback speed	Sampling frequency	Stereo / monaural	Total number of channels
Double speed	37.8KHz	Stereo	8

This gives the benefit that the CD does not need to change speed when switching from audio playback to data reading which is needed when using .DA. As the speed change is in part a mechanical operation it can be a slow operation. I then need to gather together some samples to build into the sample bank, to add a degree of safety I am going to add two seconds silence after each sample to account for any problem tracking the CD's position such as the jogging of the CD. For this sample I am going to use the following:

Channel 0 – Various Speech Samples

FileName	Media Length	Description
Tanya.wav	1.342 seconds	Tanya from Red Alert
Silence.wav	2.000 seconds	'PAD'
Oneass.wav	5.179 seconds	the geneticist from South Park
Silence.wav	2.000 seconds	'PAD'
Ben.wav	1.586 seconds	'Use the force Luke'
Silence.wav	2.000 seconds	'PAD'
Street.wav	3.600 seconds	Ryu from Streetfighter
Silence.wav	2.000 seconds	'PAD'
Bruce.wav	12.893 seconds	Bruce Lee
Silence.wav	2.000 seconds	'PAD'
Beefcake.wav	1.690 seconds	Cartman from South Park
Silence.wav	2.000 seconds	'PAD'
Kickass.wav	1.424 seconds	Cartman from South Park
Silence.wav	2.000 seconds	'PAD'

Total Time = 41.714

Channel 1 – Gun Shot Sound Effects

FileName	Media Length	Description
Gunshot1.wav	0.939 seconds	Sound Effect
Silence.wav	2.000 seconds	'PAD'
Gunshot2.wav	2.146 seconds	Sound Effect
Silence.wav	2.000 seconds	'PAD'
Gunshot3.wav	0.474 seconds	Sound Effect
Silence.wav	2.000 seconds	'PAD'
Gunshot4.wav	0.642 seconds	Sound Effect
Silence.wav	2.000 seconds	'PAD'
Gunshot5.wav	1.722 seconds	Sound Effect
Silence.wav	2.000 seconds	'PAD'
Silencer.wav	0.375 seconds	Sound Effect
Silence.wav	2.000 seconds	'PAD'

Total Time = 18.298

Channel 2 – Sci-Fi Sound Effects

FileName	Media Length	Description
Lazer.wav	1.732 seconds	Sound Effect
Silence.wav	2.000 seconds	‘PAD’
Klaxon.wav	5.437 seconds	Sound Effect
Silence.wav	2.000 seconds	‘PAD’

Total Time = 11.169

Channel 3 – Destruction Sound Effects

FileName	Media Length	Description
Explo1.wav	3.499 seconds	Sound Effect
Silence.wav	2.000 seconds	‘PAD’
Ric1.wav	0.627 seconds	Sound Effect
Silence.wav	2.000 seconds	‘PAD’
Ric2.wav	0.658 seconds	Sound Effect
Silence.wav	2.000 seconds	‘PAD’
Ric3.wav	2.143 seconds	Sound Effect
Silence.wav	2.000 seconds	‘PAD’

Total Time = 14.927

Channel 4 – Musical Score

FileName	Media Length	Description
Tune.wav	48.663 seconds	Tune
Silence.wav	2.000 seconds	‘PAD’

Total Time = 50.663

Channel 5 – Song

FileName	Media Length	Description
Song.wav	48.421 seconds	Cartman’s song
Silence.wav	2.000 seconds	‘PAD’

Total Time = 50.421

Channel 6 – Speech

FileName	Media Length	Description
Shambo.wav	42.393 seconds	Cartman’s Game
Silence.wav	2.000 seconds	‘PAD’

Total Time = 44.393

In reality you probably would not use .XA in this way as the sound effects could be just as easily played using the SPU hardware. However I didn’t have enough speech around to create some form of commentary style .XA.

The initial samples are all PCM, 44.1kHz, 16 bit, stereo samples and will be converted to 37.8kHz, stereo .XA files. Also for use in splitting the samples up into manageable chunks within the channel itself I have a 2-second silent .WAV converted to .XA. This will be used in between any samples in the same channel.

At this point it is worth mentioning that certain sound editing packages add some custom information at the end of a .WAV file such as author and package used to create the file. This is normally not a problem as the .WAV header contains details of the length of the data. However MovConv and MovPack do not use the header information in this way, and so will convert .WAV files to .XA using all the data after the .WAV header up to the end of the file. This can lead too .XA files with audible noise such as clicks or pops where the custom information has been converted to sound data at the end of the sample. The tools checkwav and cleanwav contained in this archive report on the .WAV details and remove any custom information respectively so enabling the use of any .WAV file safely with MovConv.

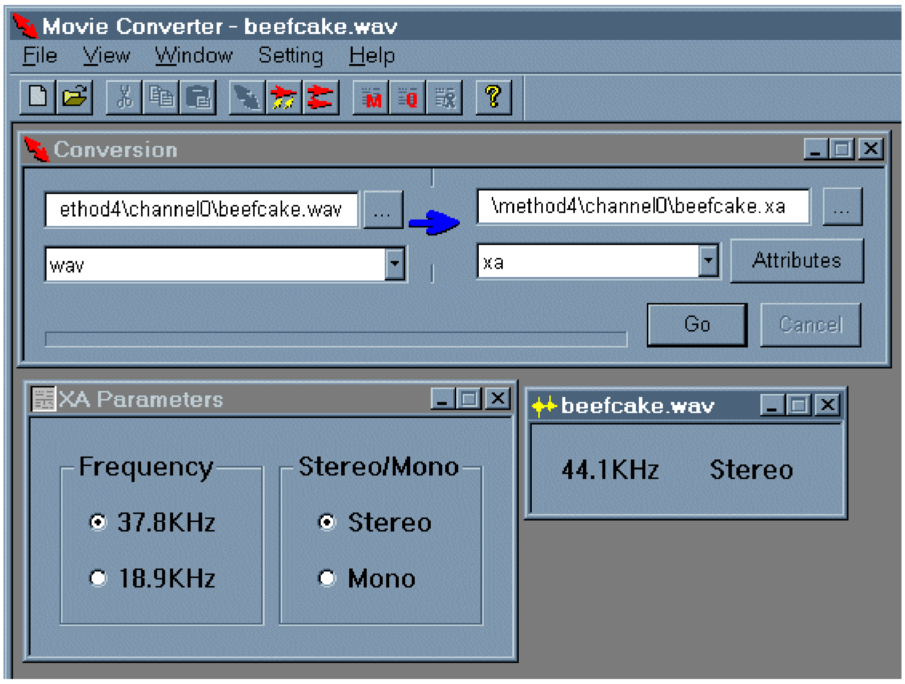

Movie Converter - beefcake.wav
Conversion
ethod4\channel0\beefcake.wav → \method4\channel0\beefcake.xa
wav → xa
XA Parameters
Frequency: (●) 37.8KHz ( ) 18.9KHz
Stereo/Mono: (●) Stereo ( ) Mono
beefcake.wav
44.1KHz Stereo

Figure 1 - Converting .WAV's to .XA, do this for all WAV's

We can now create the six channels that we are going to use, before doing this we should keep a record of the separate sample sizes so we can create a set of offsets for use during playback. For this sample the .XA sample sizes and the calculated start and stop positions can be viewed in the spreadsheet 'XA Calculations.xls' in the docs directory of the archive.

An example of the data from the spreadsheet for channel 0 looks like this:

Channel 0 – Speech

FileName	FileSize	/2336 = Sectors	Start Pos	Stop Pos
Tanya.xa	60736	26	0	200
Silence.xa	88768	38	208	504
Oneass.xa	228928	98	512	1288
Silence.xa	88768	38	1296	1592
Ben.xa	70080	30	1600	1832
Silence.xa	88768	38	1840	2136
Street.xa	158848	68	2144	2680
Silence.xa	88768	38	2688	2984
Bruce.xa	565312	242	2992	4920
Silence.xa	88768	38	4928	5224
Beefcake.xa	74752	32	5232	5480
Silence.xa	88768	38	5488	5784
Kickass.xa	63072	27	5792	6000

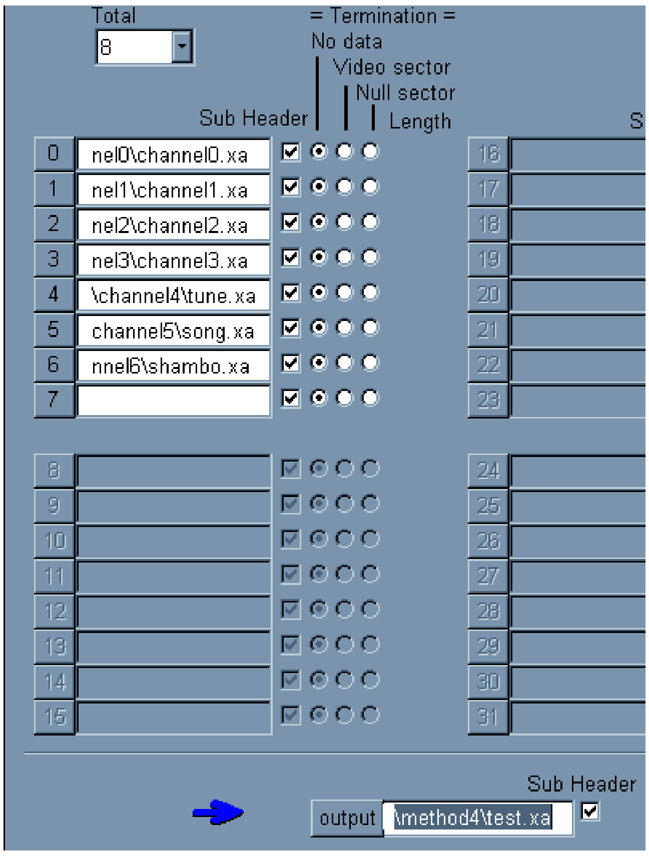
Silence.xa	88768	38	6008	6304

To concatenate all the separate samples together into one channel I just used the DOS command

```
copy tanya.xa+silence.xa+Oneass.xa ……… channel0.xa
```

It would be quite easy to create a program which could take all the separate files, output all the start / stop positions and concatenate the files together, however I couldn't be bothered for this tutorial !

Once all the channels have been created we can interleave them together into the sample bank:



Total: 8

	Sub Header	
0	nel0\channel0.xa	✓
1	nel1\channel1.xa	✓
2	nel2\channel2.xa	✓
3	nel3\channel3.xa	✓
4	\channel4\tune.xa	✓
5	channel5\song.xa	✓
6	nnel6\shambo.xa	✓
7		✓

output: \method4\test.xa Sub Header ✓

The sample bank is now ready for use.

Method 4 - Building on to the Emulator as an Emulated Disk Image

This file can be built onto the old emulator as a CD image in the same way as part one, remembering to specify the file as an .XA source file:

```
        File TEST.XA
           XASource .\test.xa
        EndFile
```

See test.cti in the Method4 subdirectory for the complete .cti file used.

Method 4 - Building On To A PlayStation Disk

Burn a CD using CD-Gen as with Part 1 – Building On To A PlayStation Disk

Method 4 - Code for Playing The .XA

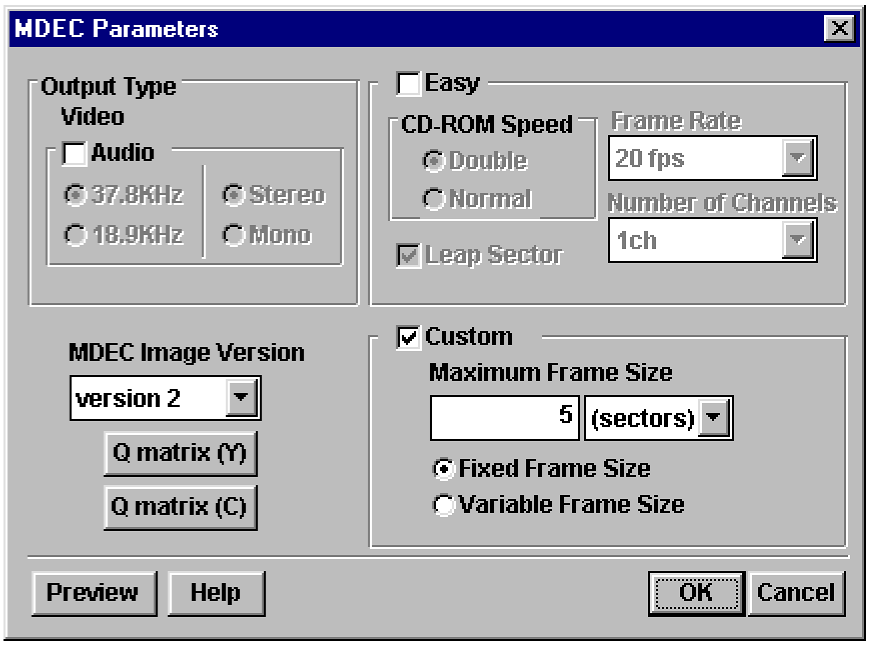
As detailed earlier, the code used to test the .XA playback is in the Method3 sub-directory, just specify the filename in the code and then use square, circle, triangle, cross to select the calculated start and stop sectors and use R1 to playback the relevant sample.

Part 3 - Advanced Interleaving Techniques – Interleaving 2 Audio Tracks Together With One Video Stream

To save space on CD it is possible to interleave multiple audio tracks together within a video stream, this allows the programmer to specify which audio track to play when playing the video stream.

Creation

Create the sound samples as usual, using 37.8kHz for the frequency and stereo output. Then convert the .AVI that you wish to use as the video using the following parameters.



MDEC Parameters

Output Type: Video

Audio [ ]: [✓ 37.8KHz] 18.9KHz; [✓ Stereo] Mono

Easy [ ]

CD-ROM Speed: [✓ Double] Normal

Leap Sector [✓]

Frame Rate: 20 fps

Number of Channels: 1ch

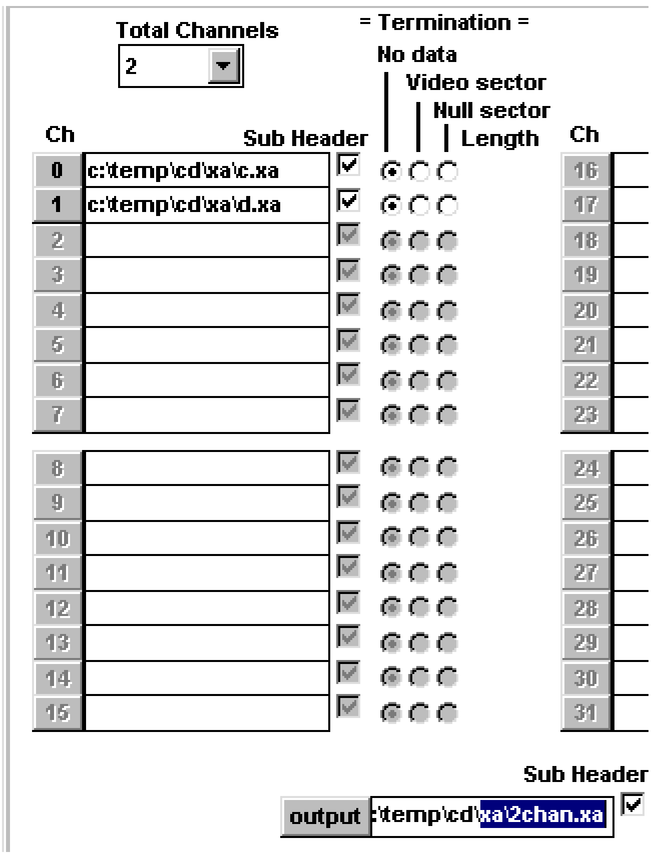
MDEC Image Version: version 2

Custom [✓]

Maximum Frame Size: 5 (sectors)

[✓ Fixed Frame Size] Variable Frame Size

Take the two .XA audio files you have created and use the Multi-Channel option in MovConv or MovPack to interleave these files together using the following settings:



Total Channels: 2

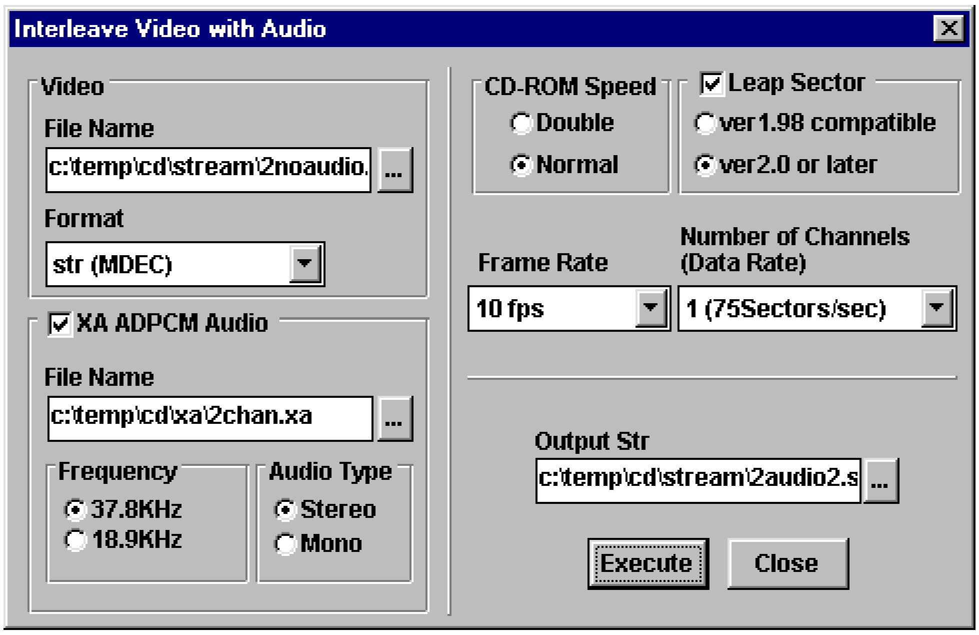
= Termination = No data | Video sector | Null sector | Length

Ch		Sub Header	No data	Video sector	Null sector
0	c:\temp\cd\xa\c.xa	✓	●	○	○
1	c:\temp\cd\xa\d.xa	✓	●	○	○

Sub Header ✓

output :\temp\cd\xa\2chan.xa

The interleaved audio can now be combined with the video footage using the 'Interleave Video With Audio' option from MovConv like so:



Interleave Video with Audio

Video
File Name: c:\temp\cd\stream\2noaudio.
Format: str (MDEC)

✓ XA ADPCM Audio
File Name: c:\temp\cd\xa\2chan.xa
Frequency: [✓ 37.8KHz] 18.9KHz
Audio Type: [✓ Stereo] Mono

CD-ROM Speed: Double [✓ Normal]
✓ Leap Sector: ver1.98 compatible [✓ ver2.0 or later]

Frame Rate: 10 fps
Number of Channels (Data Rate): 1 (75Sectors/sec)

Output Str: c:\temp\cd\stream\2audio2.s

Building on to an Emulated Disk Image And Gold CD

This stream file can be put onto both the CD emulator and a gold disk in the same way as a normal video stream with audio file.

Modification Required For Standard Movie Code

As the file now contains two audio tracks, the audio track must be selected before playing the movie. For normal streams the audio channel filter number is set to one, however when creating streams using the above technique the audio channels are numbered zero and one. The code required to set the filter correctly is detailed below where channel can be zero or one. This can be set at the start of the video stream or alternatively while the stream is actually playing.

```
CdlFILTER theFilter;

theFilter.file=1;
theFilter.chan=channel;
CdControlF(CdlSetfilter, (u_char *)&theFilter);
```

Part 4 – Interleaving Streams

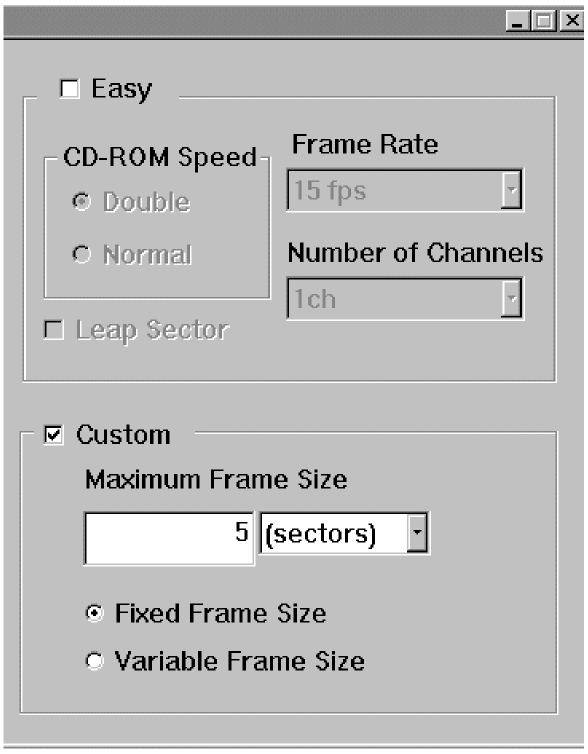
Although using the following techniques it is possible to interleave many streams together it is best to stick to two as the number of streams interleaved directly effects the frame rate of each movie. Interleaving streams is probably best used for background menus

Method 1 – Interleaving two streams together without audio

Method 2 – Interleaving two streams together with audio

Method 1 - Creation

The playback speed for each single movie in this example is 30fps, therefore the AVI's are converted using the settings below



☐ Easy

CD-ROM Speed: [✓ Double] Normal

☐ Leap Sector

Frame Rate: 15 fps

Number of Channels: 1ch

☑ Custom

Maximum Frame Size: 5 (sectors)

[✓ Fixed Frame Size] Variable Frame Size

Use custom settings so that we can define a fixed frame rate which is important is this case.

After converting the AVI's we then need to interleave these two streams together. Before we do however we need to ensure that each stream has its channel value set correctly. This is basically the same technique as used to set channels using MovPack for audio sectors. As I'm not too happy about what MovPack appears to do to video data I use a DOS program called StrSet.exe to set the channel number of the video sector header. This is a DOS based program that takes the filename and the channel number required as input and modifies the file as appropriate. For the first stream I set the channel to zero (the default) and for the second stream I set the channel to one.

We now can interleave the two video streams together, using buildcd allows us to create the interleaved file whilst not actually building the file onto the CD image.

```
; To use this file use the command line buildcd -g makefile.cti.
; This .cti file is just designed to create a video stream with two
; video streams interleaved together
;
Disc CDROMXA_PSX     ;disk format
```

```
  CDGeneratorFile test.ccs

  LeadIn XA             ;lead in track, track 0

    Empty       1000    ;defines lead in size min (150)
    PostGap     150     ;required gap at end of lead in

  EndTrack              ; end of lead in track



  Track XA                      ;start of XA (data) track

    Pause      150              ;required pause in first track after lead in

    Volume          ISO9660     ;define ISO 9660 volume
    systemarea    "system.cnf"
    PrimaryVolume               ;start point of primary volume


    SystemIdentifier        "PLAYSTATION"       ;required indetifier
    VolumeIdentifier        "PSXTEST"           ; app specific identifiers
    VolumeSetIdentifier     "PSXTEST"
    PublisherIdentifier     "SONY"
    DataPreparerIdentifier  "SONY"
    ApplicationIdentifier   "SONY"

    Lpath               ; Path tables as specified for PlayStation
    OptionalLpath
    Mpath
    OptionalMpath

    Hierarchy           ;start point of root directory definition

XAInterleavedFile   2STREAM.STR 2STREAM.STR

       XAChannelInterleave TimeCritical 0-0-0-0-0-1-1-1-1-1

       XAChannel 0
              XAFileAttributes Form1 Video
              Source Channel0.str
       XAEndChannel
       XAChannel 1
              XAFileAttributes Form1 Video
              Source Channel1.str
       XAEndChannel

XAEndInterleavedFile

    EndHierarchy        ;ends root directory definition

    EndPrimaryVolume    ;ends primary volume definition

    EndVolume           ;ends ISO 9660 definition

    PostGap  150        ;required to change track type

  EndTrack              ;ends track definition


  LeadOut  XA        ;required lead out track (must match previous track type)
    Empty  150          ;required minimum lead out
  EndTrack              ;ends track definition


EndDisc ;ends disk definition
```

Method 1 - Building On To An Emulator

Once created the interleaved stream can be burnt on to an emulated image using the following .cti file

```
ShowDefines
Disc    CDROMXA_PSX    template.img
```

```
LeadIn  XA
 Empty  1000
 PostGap       150
EndTrack

Track   XA
 Pause  150
 Volume ISO9660

  PrimaryVolume

        SystemIdentifier              "PLAYSTATION"
        VolumeIdentifier              "TEMPLATE"
        VolumeSetIdentifier           "VOLUME1"
        PublisherIdentifier           "SONY"
        DataPreparerIdentifier "EXAMPLE"
        ApplicationIdentifier  "SONY"

        Lpath
        OptionalLpath
        Mpath
        OptionalMpath

        Hierarchy           ;start point of root directory definition

          File  2STREAM.STR;1
           XASource  2stream.str
          EndFile

        EndHierarchy        ;ends root directory definition

    EndPrimaryVolume    ;ends primary volume definition

        EndVolume           ;ends ISO 9660 definition
        Empty   300
        PostGap  150        ;required to change track type
        EndTrack              ;ends track definition

  LeadOut  XA
   Empty  500
  EndTrack

EndDisc
```

Method 1 - Building On To A Gold Disk

To use this video stream file on a gold disk use CD-Gen with the .CCS file that is created with the emulated image or alternatively add the stream file as an .XA file with audio so that a small s appears in the file icon in the directory view.

Method 1 - Code

Now either video stream can be selected by using StSetChannel() and specifying either 0 or 1. This switching is instantaneous between video streams and can be done at any time during the playback. Note that this function must be used after StSetStream() has been set because StSetStream() sets the channel by default to 1.

Method 2 - Creation

To add sound to an interleaved video stream, obtain the sound track by converting the original .WAV file to a 37.8kHz, stereo .XA file. Unfortunately buildcd does not allow us to use subheadered .XA files within the channel definition so we have to strip the header information from the .XA file. To do this the following data is stripped out.

XA-ADPCM audio sector (2336 bytes)	Final Data
Subheader (8 bytes)	Removed (needs to be defined later)
XA-ADPCM audio (2324 bytes)	Kept

Dummy data (4 bytes)	Removed (added later)

To do this use the EXTRACT tool or the StripXA.exe located in the tools directory with the sample code. This converts an .XA file to the above format, now create the two video streams as described in the above example and we have all the separate files ready for interleaving. Because the .XA has lost it definition, i.e. it is now only data, we need to add some definitions to the audio channel to allow it to be actually recognised as an audio channel. Within our .cti file that we use to create the interleaved file we use the following definitions for the audio channel. The following is the complete .cti file used in the following command to create the interleaved file

```
buildcd –g <.CTI filename>
```

```
; To use this file use the command line buildcd -g makefile.cti.
; This .cti file is just designed to create a video stream with two
; video streams interleaved together
;
Disc CDROMXA_PSX     ;disk format

  CDGeneratorFile test.ccs

  LeadIn XA             ;lead in track, track 0
    Empty       1000    ;defines lead in size min (150)
    PostGap     150     ;required gap at end of lead in
  EndTrack              ; end of lead in track

  Track XA                      ;start of XA (data) track
    Pause      150              ;required pause in first track after lead in
    Volume          ISO9660     ;define ISO 9660 volume
    systemarea              "licensee.dat"
    PrimaryVolume               ;start point of primary volume
    SystemIdentifier        "PLAYSTATION"       ;required indetifier
    VolumeIdentifier        "PSXTEST"           ; app specific identifiers
    VolumeSetIdentifier     "PSXTEST"
    PublisherIdentifier     "SONY"
    DataPreparerIdentifier  "SONY"
    ApplicationIdentifier   "SONY"

    Lpath               ; Path tables as specified for PlayStation
    OptionalLpath
    Mpath
    OptionalMpath

    Hierarchy           ;start point of root directory definition

XAInterleavedFile   2STREAM.STR 2STREAM.STR

        XAChannelInterleave  TimeCritical  0-0-0-0-0-1-1-2-1-1-1-0-0-0-0-2-0-1-1-1-1-
1-0-2-0-0-0-0-1-1-1-2-1-1-0-0-0-0-0-2-1-1-1-1-1-0-0-2-0-0-0-1-1-1-1-2-1-0-0-0-0-0-1-2-
1-1-1-1-0-0-0-2-0-0-1-1-1-1-1-2

       XAChannel 0
              XAFileAttributes Form1 Video
              Source Channel0.str
       XAEndChannel

        XAChannel 1
               XAFileAttributes Form1 Video
               Source Channel1.str
        XAEndChannel

        XAChannel 2
              XAFileAttributes Form2 Audio
              XAAudioAttributes ADPCM_B Stereo
              Source Openup.xxa
       XAEndChannel

XAEndInterleavedFile

    EndHierarchy        ;ends root directory definition
    EndPrimaryVolume    ;ends primary volume definition
    EndVolume           ;ends ISO 9660 definition
    PostGap  150        ;required to change track type
  EndTrack              ;ends track definition
  LeadOut  XA        ;required lead out track (must match previous track type)
    Empty  150          ;required minimum lead out
  EndTrack              ;ends track definition

EndDisc ;ends disk definition
```

The two main differences to the interleaved video stream without audio is that the .XA audio needs the XA subheader to be redefined, this is done in the channel definition. Also the interleave is defined so that the interleave is correct with an audio sector every eighth sector, this results in a massive XAChannelInterleave definition.

Method 2 - Building On To A Gold Disk

To use this video stream file on a gold disk use CD-Gen with the .CCS file that is created with the emulated image or alternatively add the stream file as an .XA file with audio so that a small ‘s’ appears in the file icon in the directory view.

Method 2 - Code

Now either video stream can be selected by using StSetChannel() and specifying either 0 or 1. This switching is instantaneous between video streams and can be done at any time during the playback. Whilst playing either video the audio stays the same. Again make sure that the StSetChannel() function is called after StSetStream()

Part 5 – Linking Streams

It is fairly simple to link together streams, and it may be necessary to do this at some time. Maybe if you have two sources of video footage and you only have them in the .STR format. This section will detail how to link two streams together when the .STR files do not contain any audio.

All that is needed to do is to create a PC program that will take two .STR files and output the final .STR file. The only thing that needs to be modified for each stream is the frame number in the second stream file, this just needs to follow on from the first stream file’s last frame. The code to do this is all fairly explanatory and is in the tools\source\strlink subdirectory of this archive. Note that to successfully link streams together they must be the same resolution and also share the same audio properties.