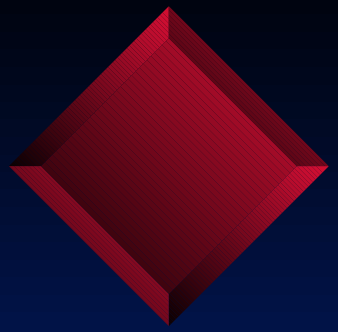


CD

Advanced Topics

PlayStation™

What can I do with the CD during my game?

- Play music
  - ADPCM
  - DA
- Stream
  - Movies
  - Data

Using the CD to play music during gameplay

Playing soundtracks is the most obvious use for the CD

- ADPCM
  - Compressed 8X to 32X
  - Easy to do: See \PSX\SAMPLE\CD\TUTO\TUTO5.C
- DA / Red Book
  - Lossless
  - Easy to do: See \PSX\SAMPLE\CD\TUTO\TUTO4.C

BUT, with the powerful MIDI support for the PlayStation, you may want to go a step further with the CD...

Streaming data from the CD

- Linear streaming
  - Movie playback interleaved with data
  - ADPCM audio interleaved with data
- Non-linear streaming
  - World streaming

PlayStation™

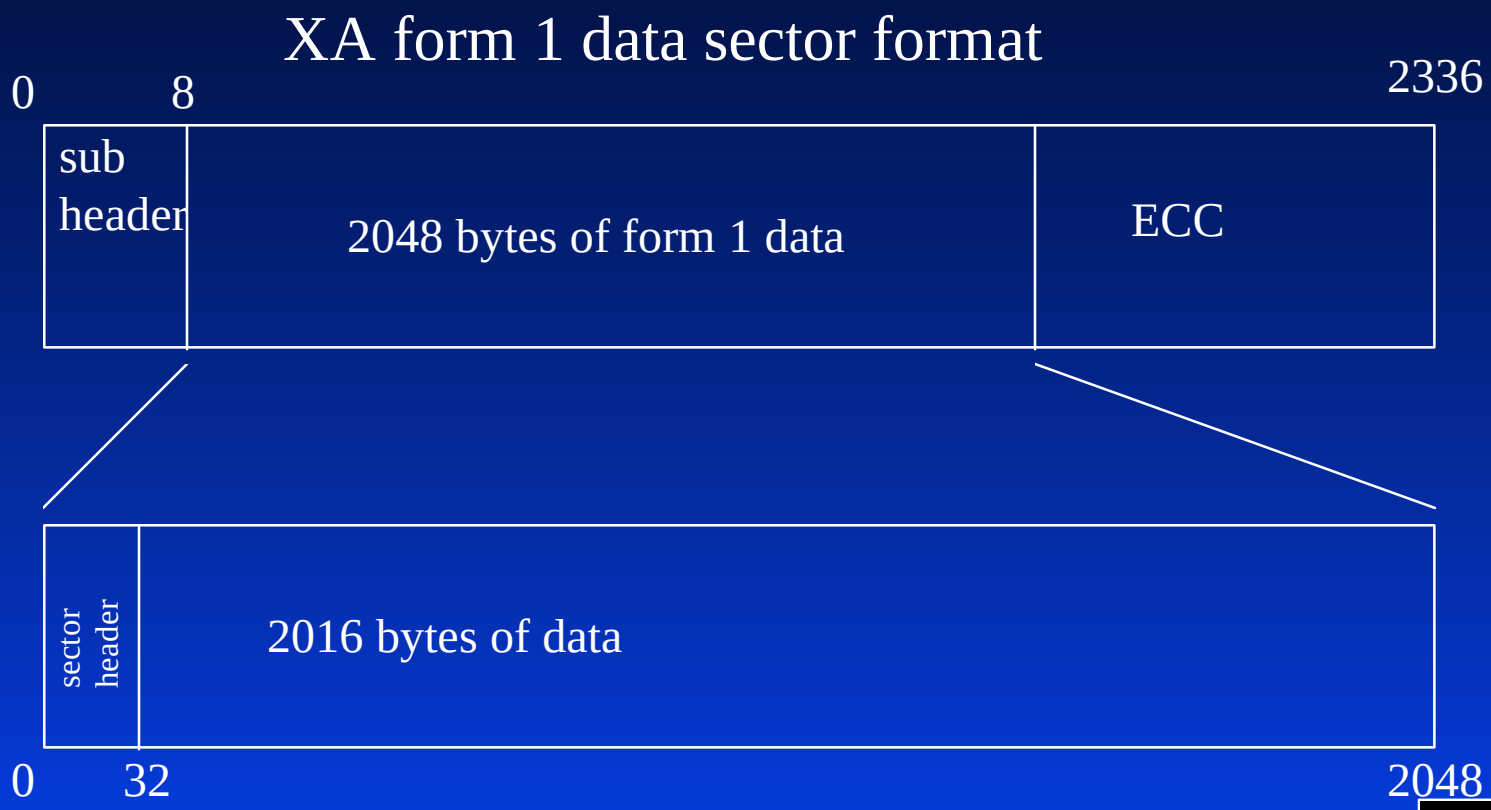
STR format

- The STR format is a sector by sector stream of data and (optionally) interleaved XA audio
- Audio sectors are stripped off in the CD-ROM subsystem and sent to the SPU, leaving only data sectors in the CD buffer
- The most common use of the STR format is MDEC movie streaming
- Understanding the format makes it possible to do much more with the STR format

STR format (cont.)

- Each data sector has a 32 byte header describing the data
- There is a preset data format for MDEC movies, other formats can be created by you
- Data sectors are grouped into frames
- The frame size, in sectors, is fixed for entire file
- Each frame can contain any type of user data
- Frames are streamed off the CD into a frame ring buffer for program access

Interleaving movies and data



PlayStation™

Sector header - general libstr sectors

32 byte sector header

WORD ID

bits 0-3 STR format version

Currently, there is only one STR format version, 0x1

bits 4-11 System reserved

bits 12-15 format ID, always 0x6

WORD Frame format

The high Bit means Sony reserved format.

If you set this bit, you must use a Sony pre-defined frame format.

Currently, the only Sony format is 0x8001, MDEC movie

If you do not set this bit, you may define your own frame format.

WORD current sector number in current frame (ex: 1, 2, 3, 4, 5, 1, 2, 3, ...)

WORD total sector count for current frame (ex: 5, 5, 5, 5, 5, 4, 4, 4, 4, 5, ...)

DWORD current frame number (ex: 1, 1, 1, 1, 1 , 2, 2, 2, 2 , 3...)

DWORD size of frame data in bytes (note: six bytes smaller than real data size)

Last 16 bytes are user defined (for non-MDEC movies)

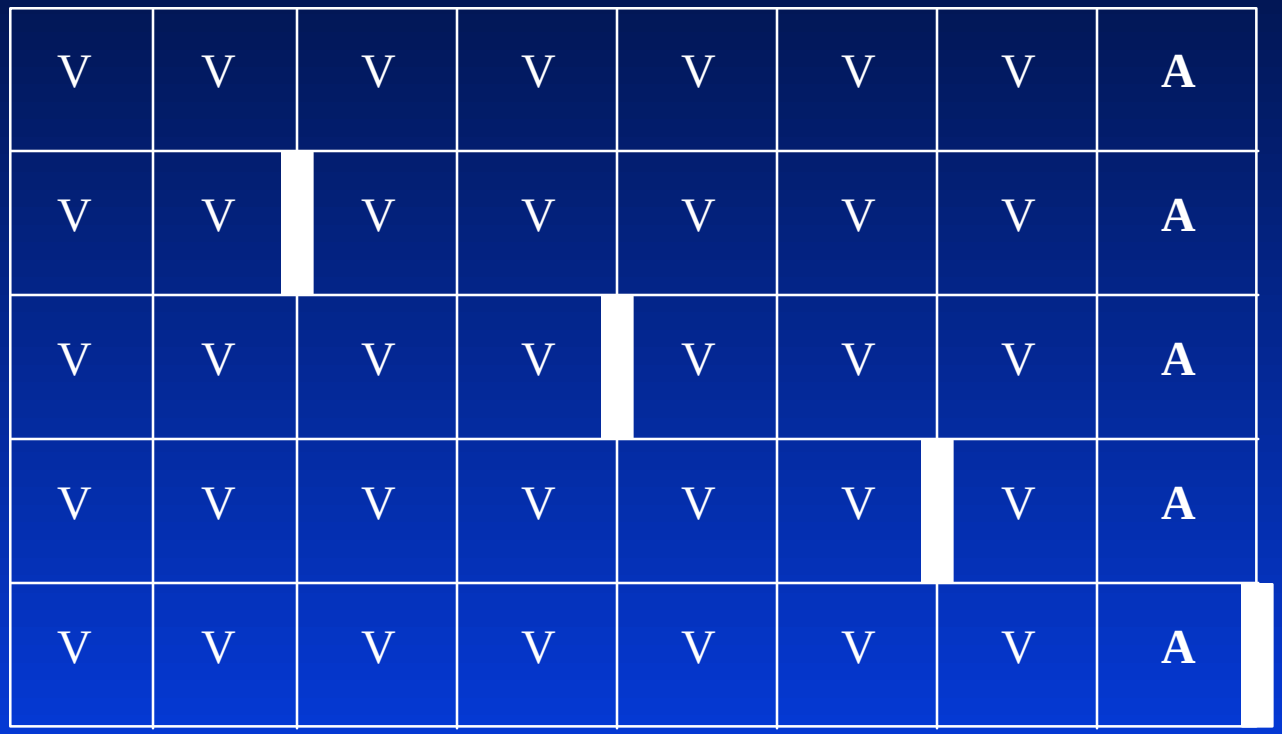
Sector header - Movie frames

32 byte sector header	
Same as general format	WORD ID 0x6001 WORD frame format 0x8001 (bits 10-14 are channel number) WORD current sector number in current frame WORD total sector count for current frame DWORD current frame number DWORD size of .BS in bytes (note: six bytes smaller than entire .BS data)
Added MDEC fields	WORD frame width in pixels WORD frame height in pixels DWORD headm, first DWORD of .BS file DWORD headv, second DWORD of .BS file DWORD 0 (unused)

Movie interleaving

Sector by sector - No additional data

V = Video, **A** = Audio



15 fps, 37.8kHz stereo

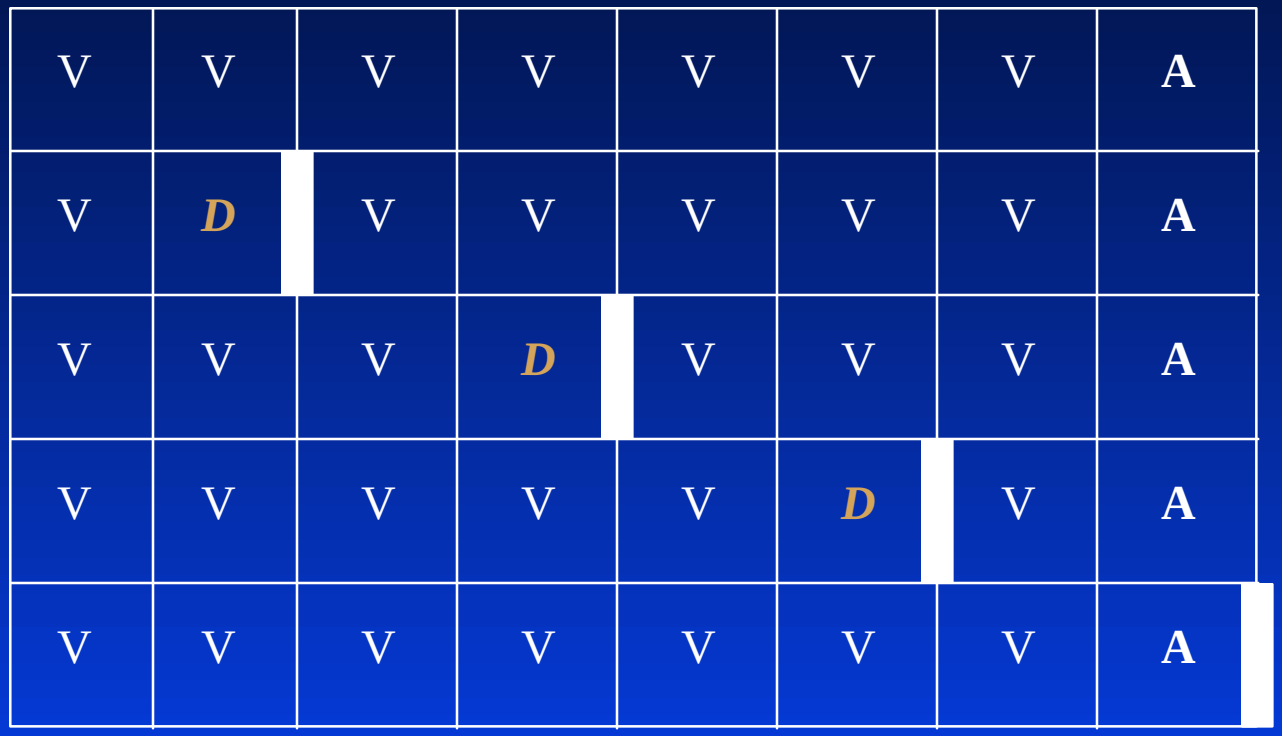
= End of frame

PlayStation™

Movie interleaving w/data

Using sector blocks - Use custom setting in MOVCONV

V = Video, **A** = Audio, ***D*** = Data



15 fps, 37.8kHz stereo

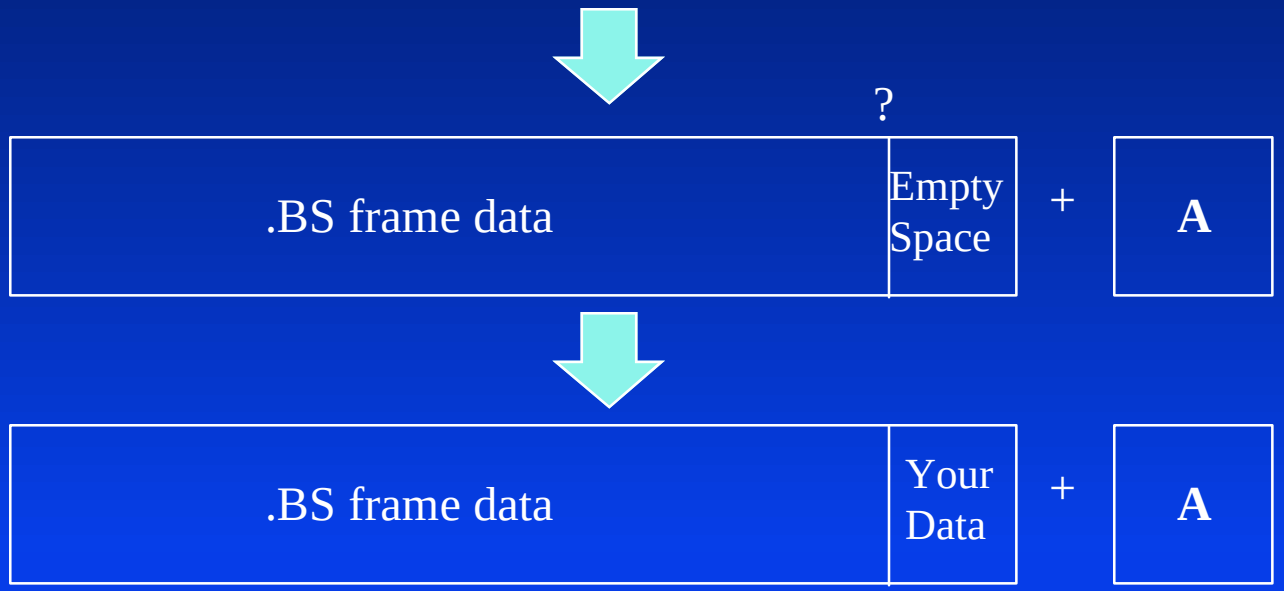
= End of frame

PlayStation™

Movie interleaving w/data

Filling in unused space - not on sector boundaries

Frames are compressed iteratively for best quality, but some frames will have extra space at the end. This space can be filled with additional data.



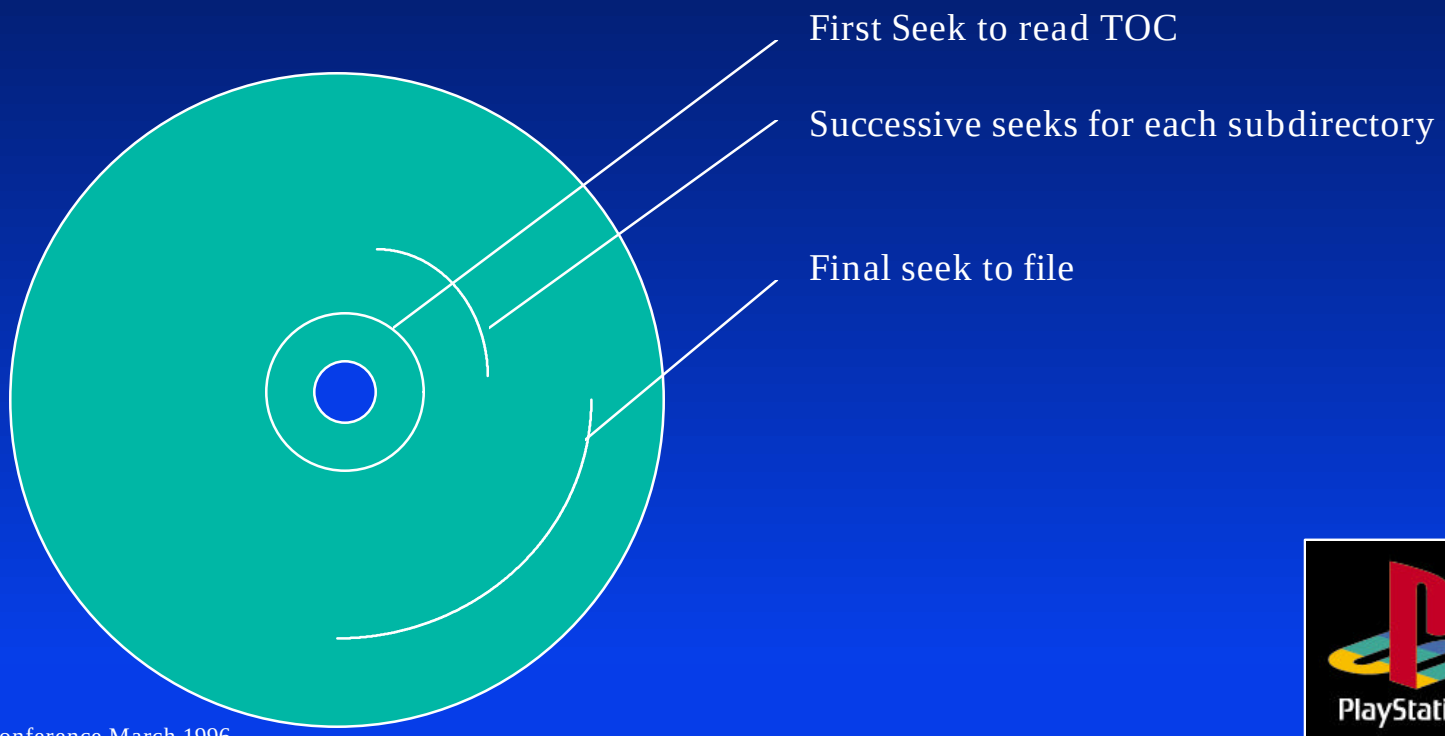
PlayStation™

Speed Issues

- Minimize seeks
  - CdSearchFile() vs. Direct seek
  - Short seeks are OK
  - Load child processes with data simultaneously
  - Set data up in contiguous blocks
- Read asynchronously
- Avoid speed changes
- Do not stop CD

CdSearchFile vs. Direct Seek

CdSearchFile() has a single directory buffer, which causes it to seek multiple times to go to any file.



PlayStation™

Hard-code file locations for speed

Use CDGEN to do a layout

- Put MAIN.EXE last in track 1 (so size can vary)

Use a utility to create POS.H with position info from CCS file

- Absolute file location appears on the line following each sourcefile path

Compile with POS.H

Burn CD / Build emulation image

Short seeks are fast

+ / - 100 sector seeks require rotation of CD read head, but not linear motion, so they are much faster than longer seeks

Optimize data layout to take advantage of this

PlayStation™

Read asynchronously

Main loop:

```
...
CdReadyCallback(cbdataready);
CdControl(CdlReadN, (unsigned char *)&fp.pos, 0);
...
while (((padd = PadRead(1))&PADk) == 0) {
        balls();
        FntPrint("Intr count = %d\n", hit_intr);
        for (i = 0; i < RNGSIZE; i++) {
                for (j = 0; j < 3; j++) FntPrint("%08x", sector[i][j]);
                FntPrint("\n");
        }
        FntFlush(-1);
        opadd = padd;
}
```

Callback:

```
static void cbdataready(int intr, u_char *result) {
        if (intr == CdlDataReady) {
                CdGetSector(sector[rid], 2048/4);
                rid = ((rid+1)&RNGMASK);
                hit_intr++;
        }
}
```

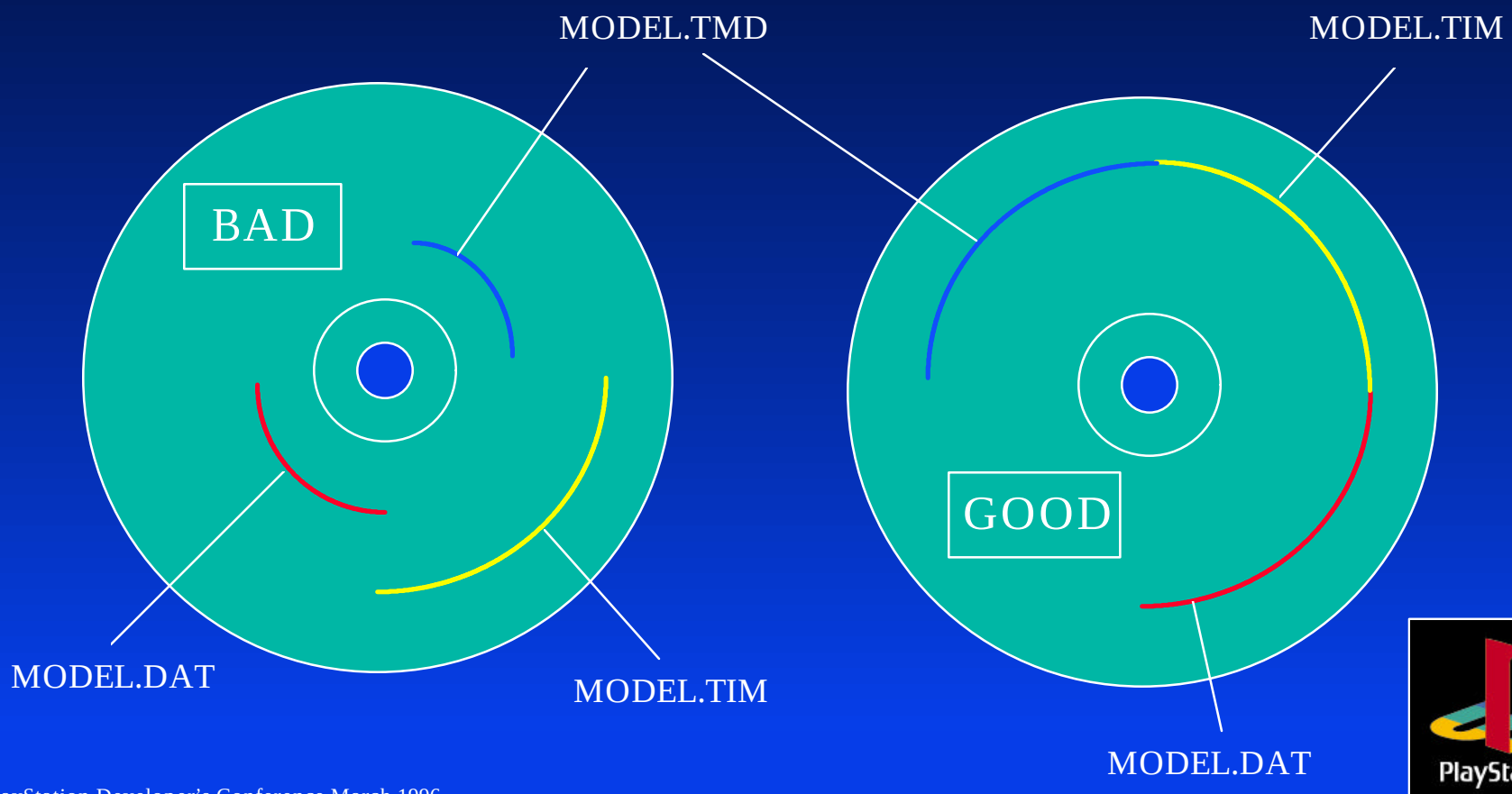
Important Note: You must add your choice of error correction

Avoid speed changes

- Spin up and spin down takes a lot of time
  - Avoid using CD-DA with a lot of intermittant CD access
  - Use XA audio at double speed instead
- Do not use CdlStop, use CdlPause instead

Set data up in contiguous blocks

MODEL.TMD, MODEL.TIM, MODEL.DAT
to be read at same time



PlayStation™

CD Error issues

- Read errors
- Seek errors
  - Retry
  - Overshoot

Read errors

CdControl() only returns successful acceptance of CD command, not successful completion of CD command

Bad Strategy:

```
if (CdControl(CdlReadN, pos, result) != 1) goto error;
```

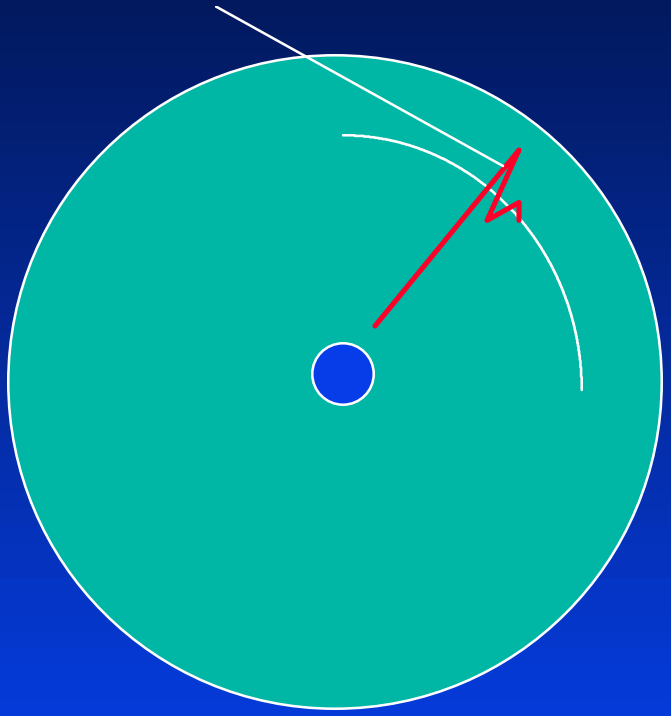
Good strategy

Set up a watchdog timer in VSyncCallback for retry

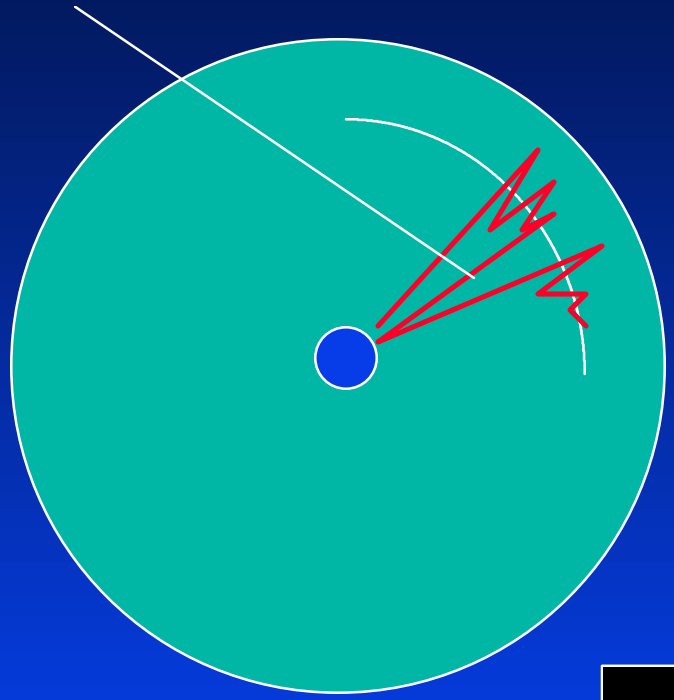
Example: CdRead waits 8 seconds

Seek errors

CD read head can overshoot a few times before success



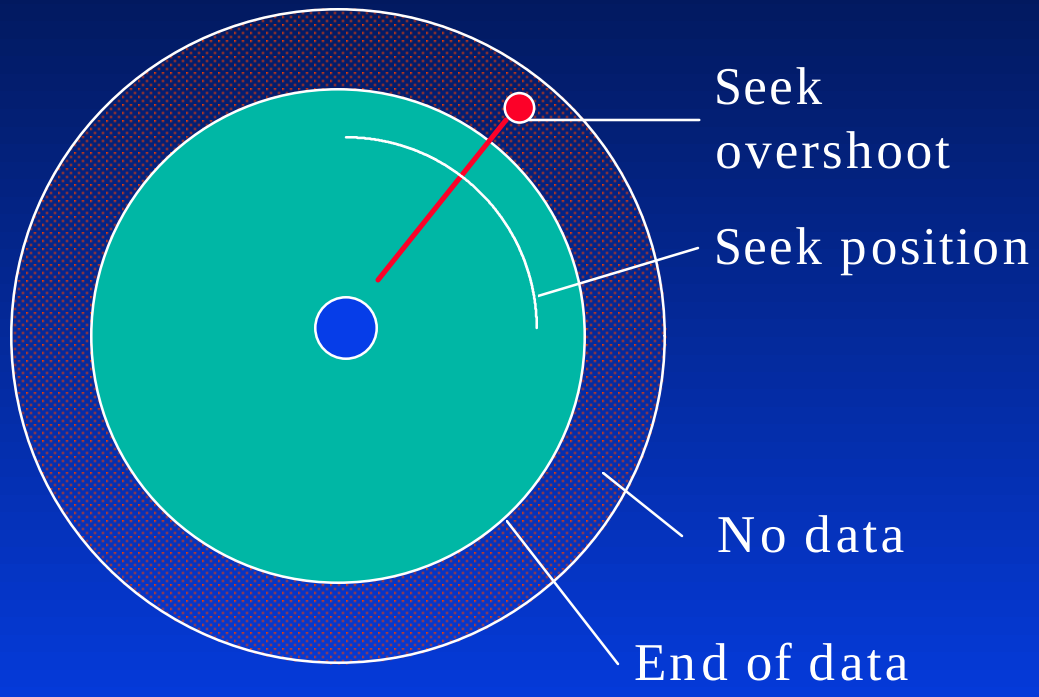
If the seek does not settle within a limited time, the head returns to the center and retries



PlayStation™

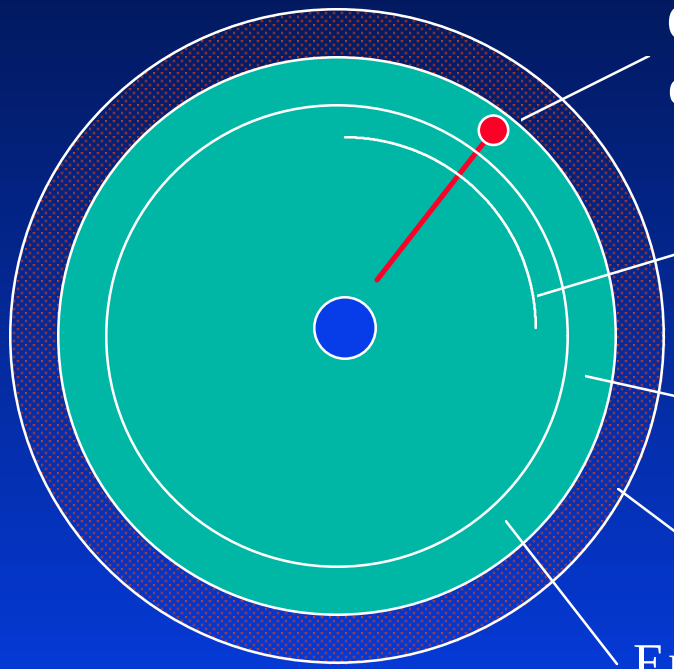
Seek errors (cont.)

A seek past the end of data is fatal



PlayStation™

Seek errors (cont.)



Solution: Put 3 minutes of dummy data at end of CD (may be real data that you do not seek into, but only read)

PlayStation™

Multi CD games

When the CD cover is opened, the CD subsystem is put into an indeterminate state

Two ways to tell when CD subsystem state is restored

1) Polling shell open flag then timing

2) Seeking until a non-error is returned

Polling shell open flag and timing

```
/* Status Contents */
#define CdlStatPlay          0x80    /* playing CD-DA */
#define CdlStatSeek          0x40    /* seeking */
#define CdlStatRead          0x20    /* reading data sectors */
#define CdlStatShellOpen     0x10    /* once shell open */
#define CdlStatSeekError     0x04    /* seek error detected */
#define CdlStatStandby       0x02    /* spindle motor rotating */
#define CdlStatError         0x01    /* command error detected */
```

Psuedocode:

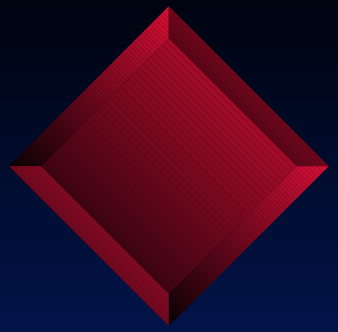
```
// Show "Put in Disk #2..." screen
while (!(status & 0x10));   // wait for lid to open
while (status & 0x10);      // wait for lid to close
// Wait 10 seconds
// Continue ...
```

Seeking until non-error returned

Use logical seek, because a physical seek will
be successful for non-PlayStation CDs
If DA CD is anticipated, use physical seek

Pseudocode:

```
// Show "Put in Disk #2..." screen
while (!(status & 0x10)); // wait for lid to open
while (CdControlB(CdlSeekL, pos, 0)) == error); // wait for success
// Continue ...
```



The End

PlayStation™