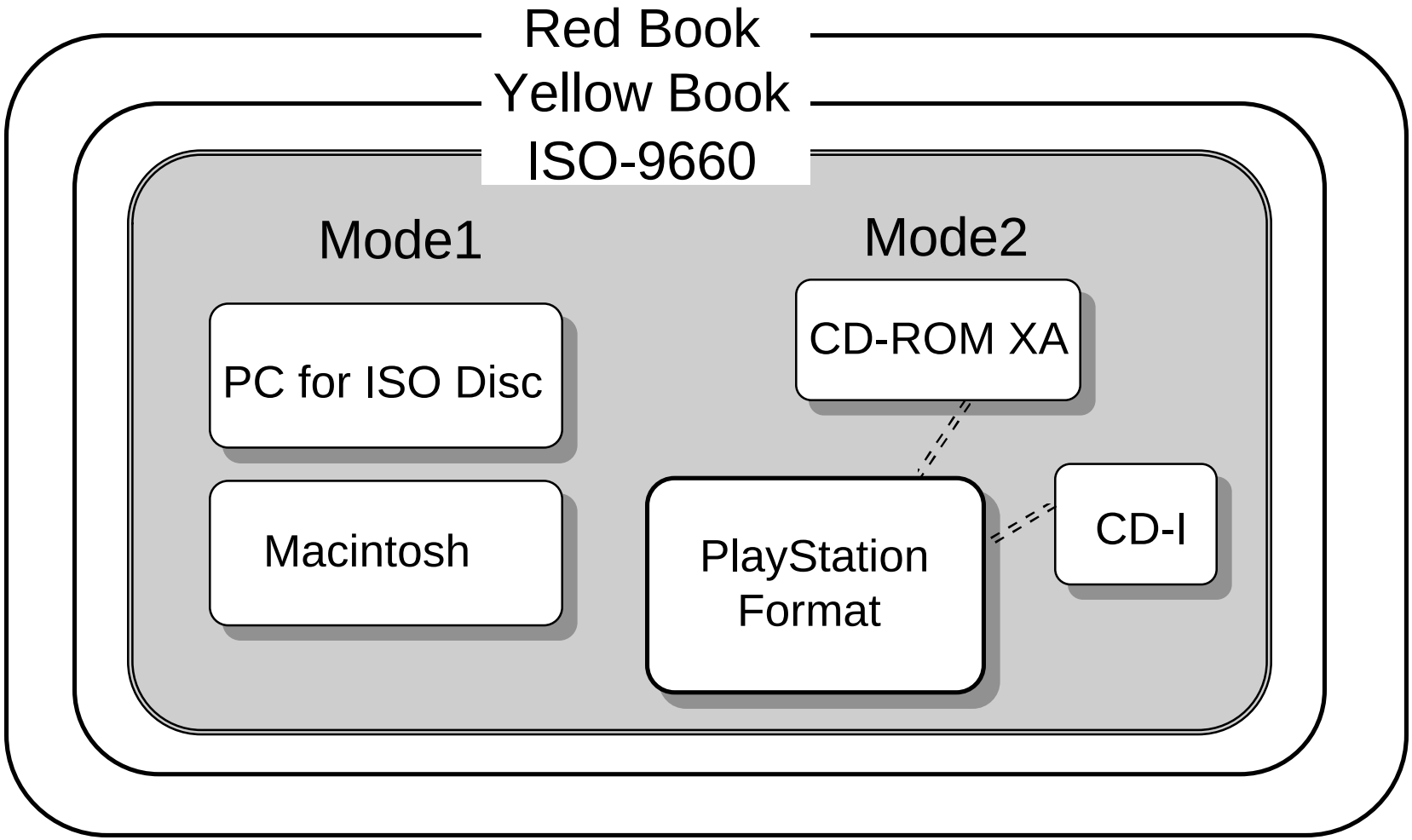
Software Development Seminar

CD-ROM (Basic)

CD-ROM Basic Overview
LibCD Overview

CD-ROM Basic Overview

Relationships between CD-ROM formats



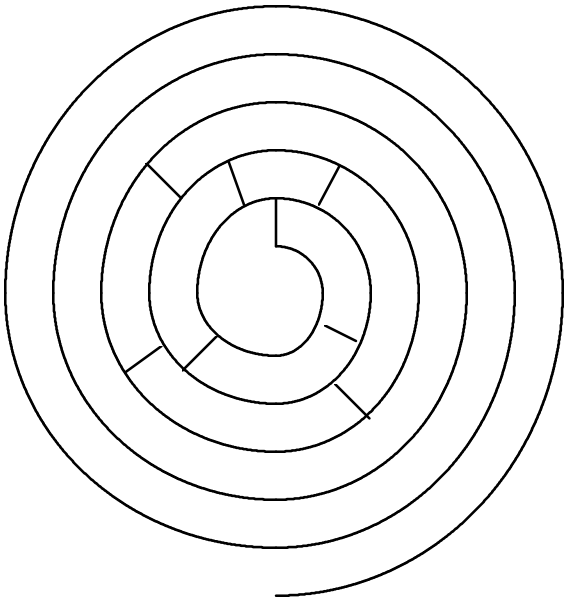
Relationships between CD-ROM formats (cont)

Red Book	Describes the format of an audio CD
Yellow Book	Specification for the storage of Digital Data on a CD (Compact Disc)
ISO-9660	Describes the directory structure for configuring the filesystem
CD-I	Describes the OS including the graphic system for playing data, audio, graphics, etc.
CD-ROM XA	Describes real-time data of audio and graphics

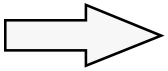
The PlayStation format conforms to ISO-9660

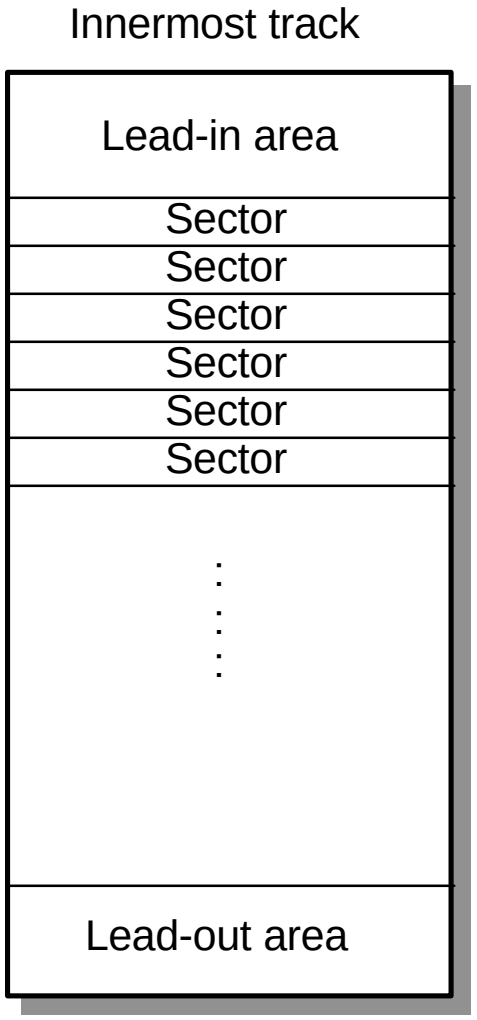
The PlayStation format is not compatible with what is technically in common with CD-I and CD-ROM XA

Data layout



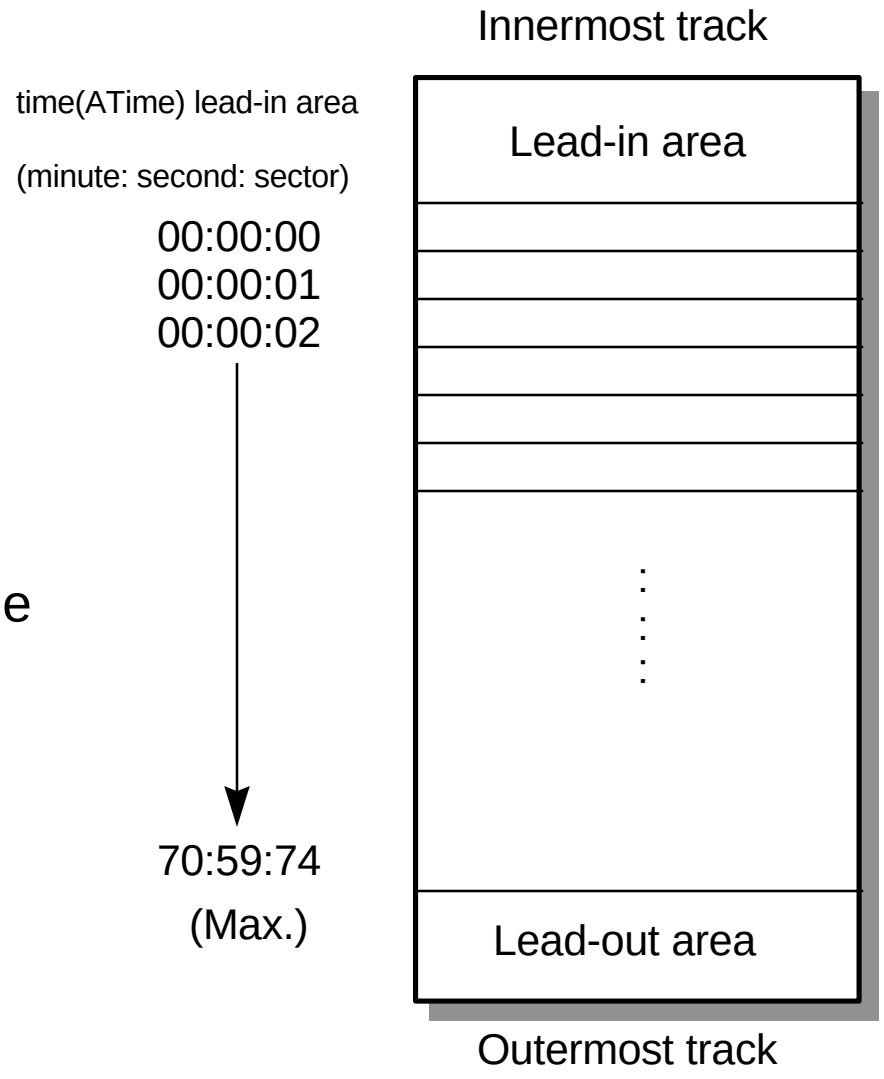
Fixed length sectors are arranged in a spiral





Sector address

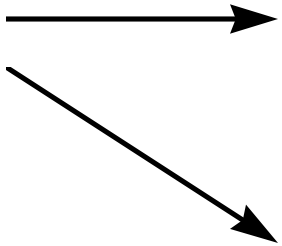
- Minute: second: specified in the sector
- One second = 75 sectors
- Immediately after Lead-in, the first sector is 00:00:00



Sector type

- DATA

  program
  graphics/movie data
  others, general data



Mode2/Form1
data sector

- ADPCM

  compressed audio data

Mode2/Form2
data sector

- CD-DA

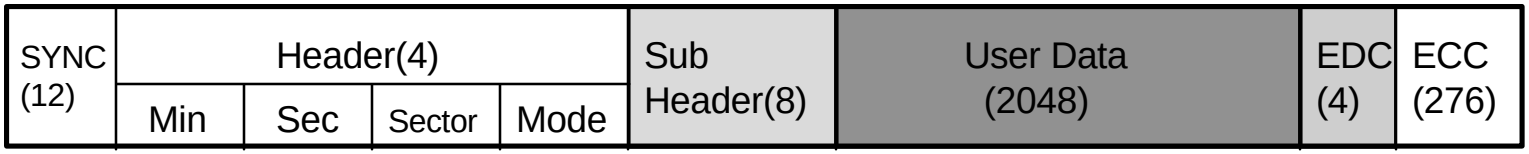
  same as audio CD
  44.1 KHz 16-bit PCM data

CD-DA sector

Structure of separate type sectors

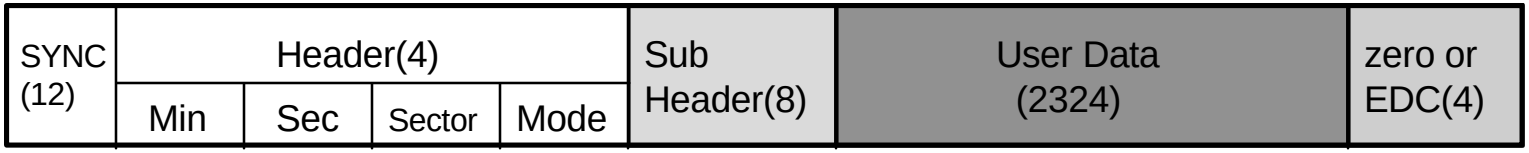
Data :Mode2/Form1

ECC corrected data



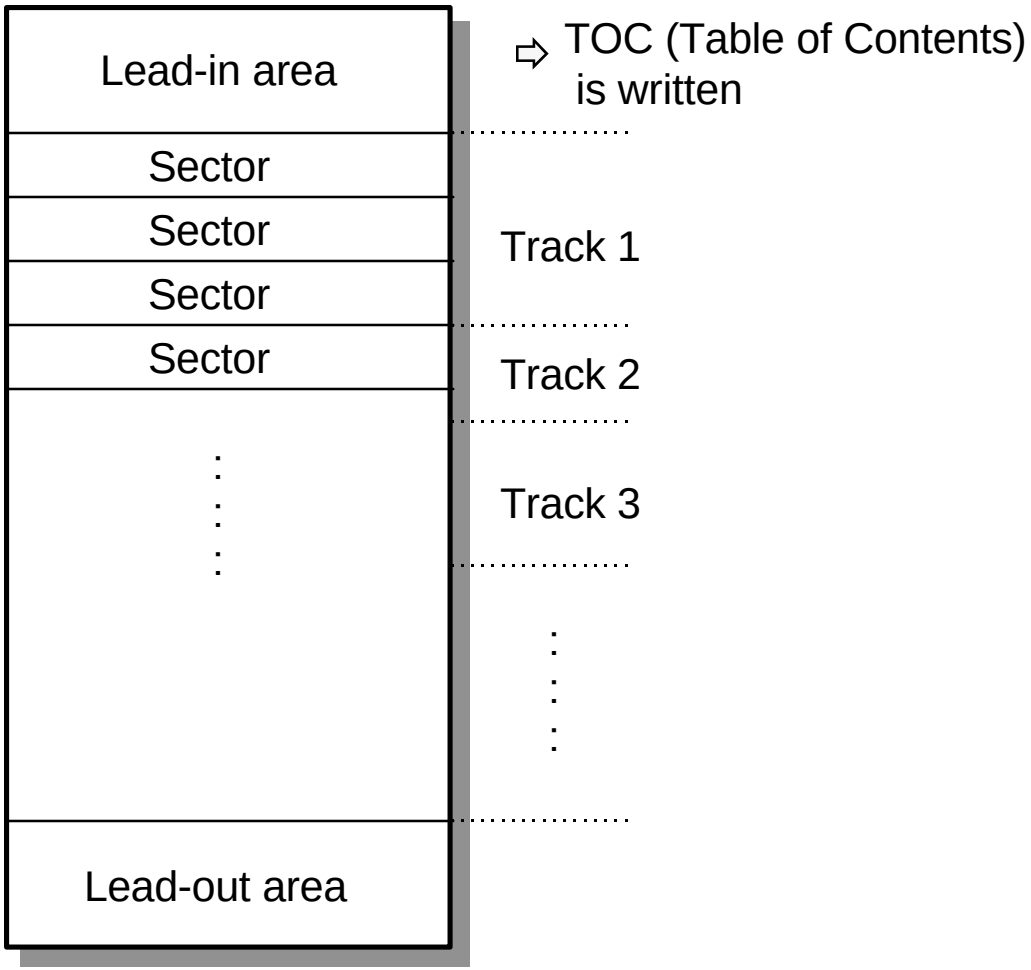
Data :Mode2/Form2

No ECC correction
Audio data, etc. from interpolation
Used in played back data

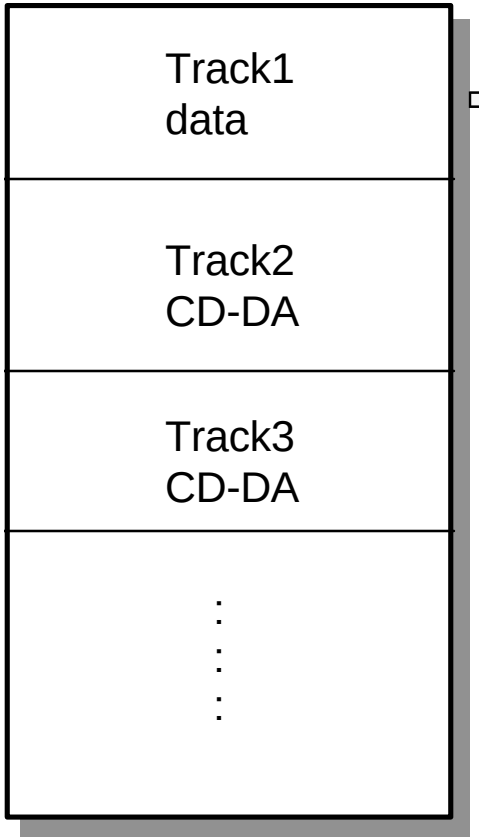


CD-DA

Track structure



Track structure (cont)



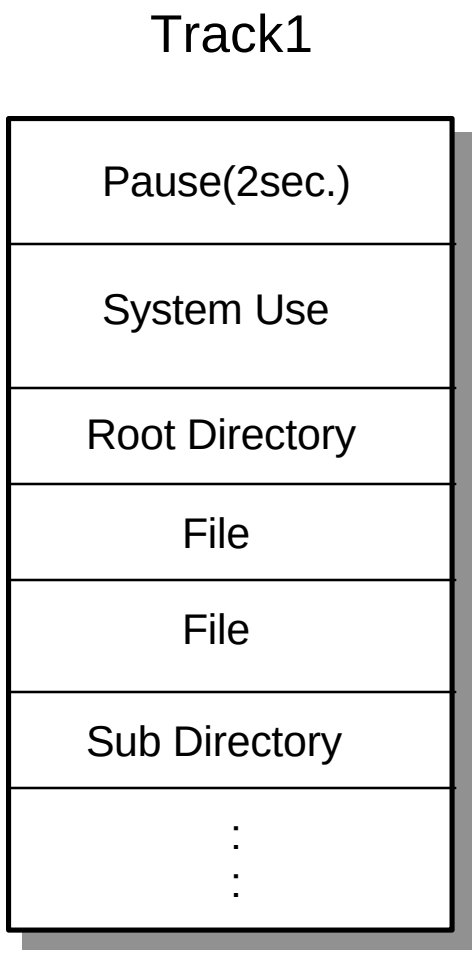
Lead track
data track
ISO-9660 filesystem

Note: The track must be CD-DA starting with track 2

ISO-9660 filesystem

Supports hierarchical directory
30 files / directory in LibCD

Each file is laid out continuously
pointing to the start location only



ISO-9660 filesystem (cont)

- System Area
  Stores license data

- Primary Volume Descriptor
  Maintains various types of information like disk name and author

- Path Table
  Maintains the location of each subdirectory within the disk

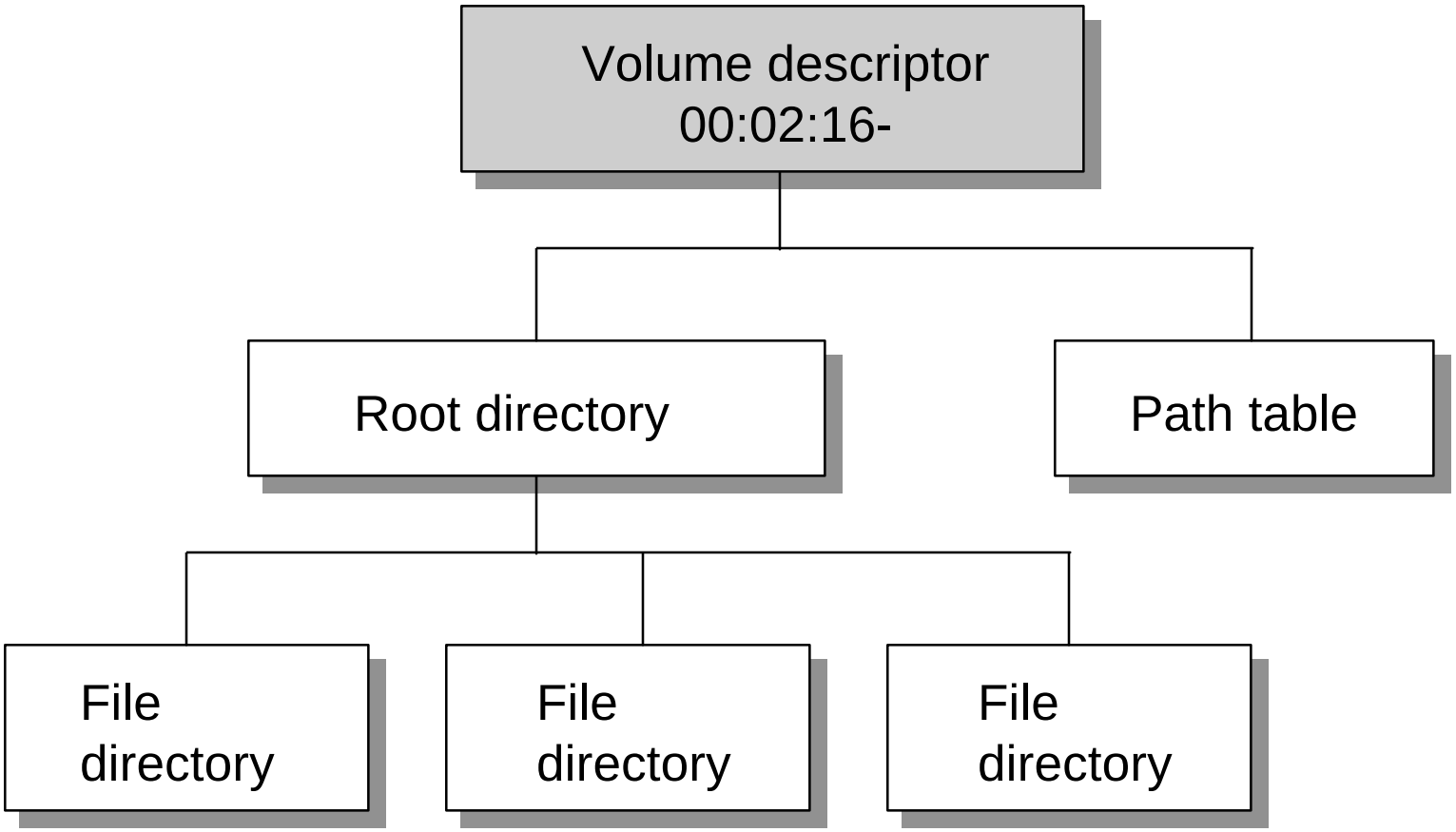
System area

• The system area is a fixed group of 16 sectors from physical sector 00:02:00 to 00:02:15

• **Use**

Boot area
Embeds a control code that limits access to the CD
Stores data used by the interface necessary for this OS
Others

Volume descriptor



Volume descriptor (cont)

System Identifier	**PLAYSTATION**
Volume Identifier	**SLPS_*****
Volume Set Identifier	(Example) **PSXGAME**
Volume Set Size	
Volume Set Seq	
Publisher Identifier	(Example) **SCE**
Data Preparer Identifier	(Example) **SCE**
Application Identifier	**PLAYSTATION**
Copyright File Identifier	
Abstract File Identifier	
Bibliographic File Identifier	
Creation Time	
Modification Time	
Expiry Time	
Effective Time	

Path table

- Has all addresses of subdirectories in the hierarchy
- Improves performance of file access to the CD-ROM
- Contains one additional index of a separate hierarchical directory
- Useful for dedicated media which is read like a CD-ROM

Filenames

- Filenames

  (Name) . (Extension) ; 1

  8 char 3 char

- Directory names

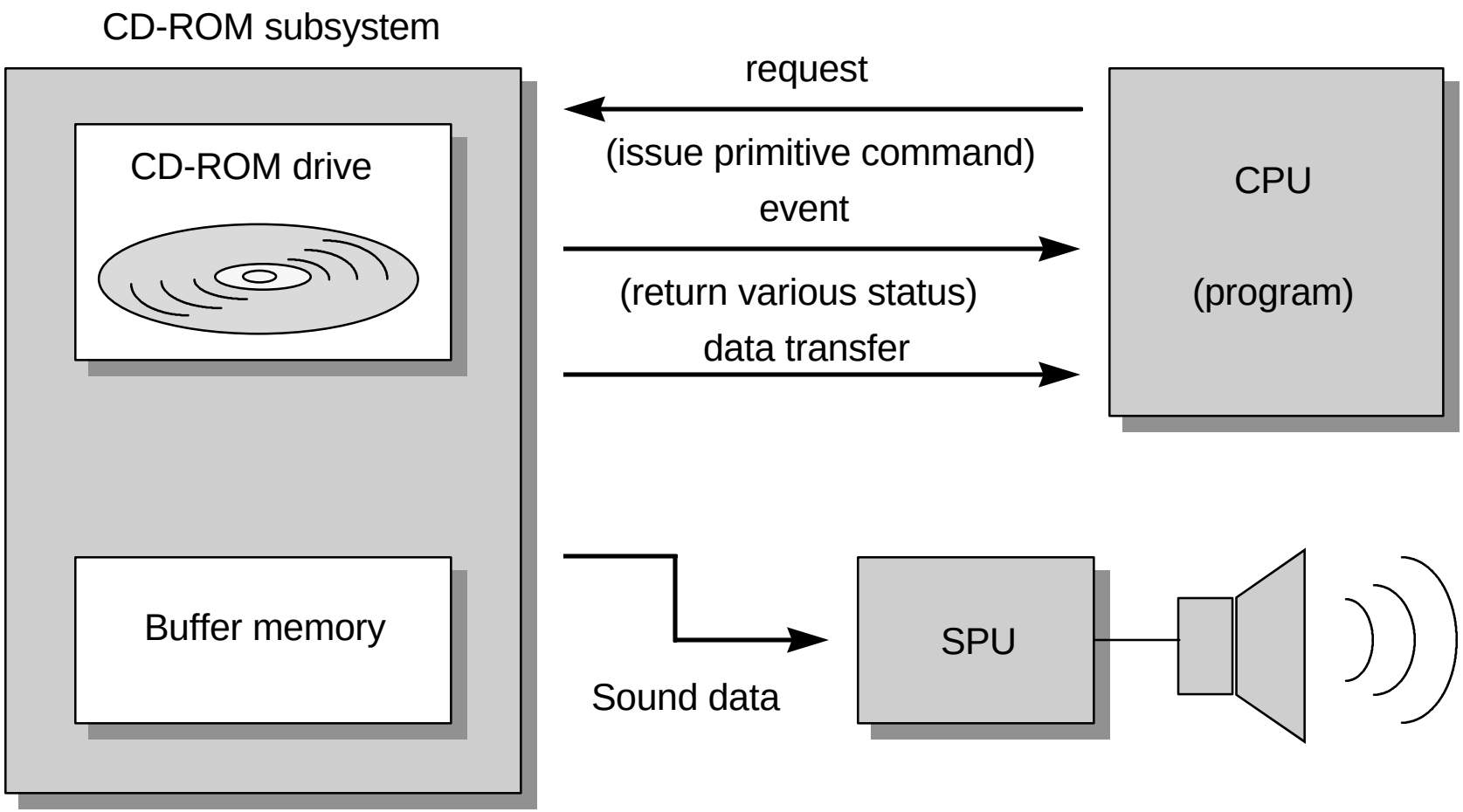
  (Name) [no extension]

  8 char

Comparison of DOS and ISO-9660

	ISO-9660	DOS
Limit on usable characters in filenames	0-9, A-Z	0-9, A-Z _^$~!#%&-{}()@'`
Limit of depth of hierarchy	Max. 8	Unlimited
Limit of subdirectory name	No extension	Extension

CD-ROM playback system



Data read

- Read data goes into the sector buffer of the internal subsystem

- From the sector buffer it is transferred one sector at a time to the host

- 300KByte (double speed)
  150KByte (standard speed)

- Normal Read / streaming

Primitive commands

- Initialize
- CD Player operation (PLAY, STOP, PAUSE, FORWARD, etc...)
- Seek
- Sector Read
- Others (Status Read, etc...)

Running mode

DA playback mode

CD-DA track

No header in any sector

Position set according to subcode

⇩

Physical Seek

Data transfer mode

Data sector (Track1)

Header in every sector

Position set according to address

⇩

Logical Seek

LibCD Overview

LibCD

- LibCD is a library of low level sub-CPU functions which pertain to the CD-ROM

- Includes high level functions like file search and multisector read

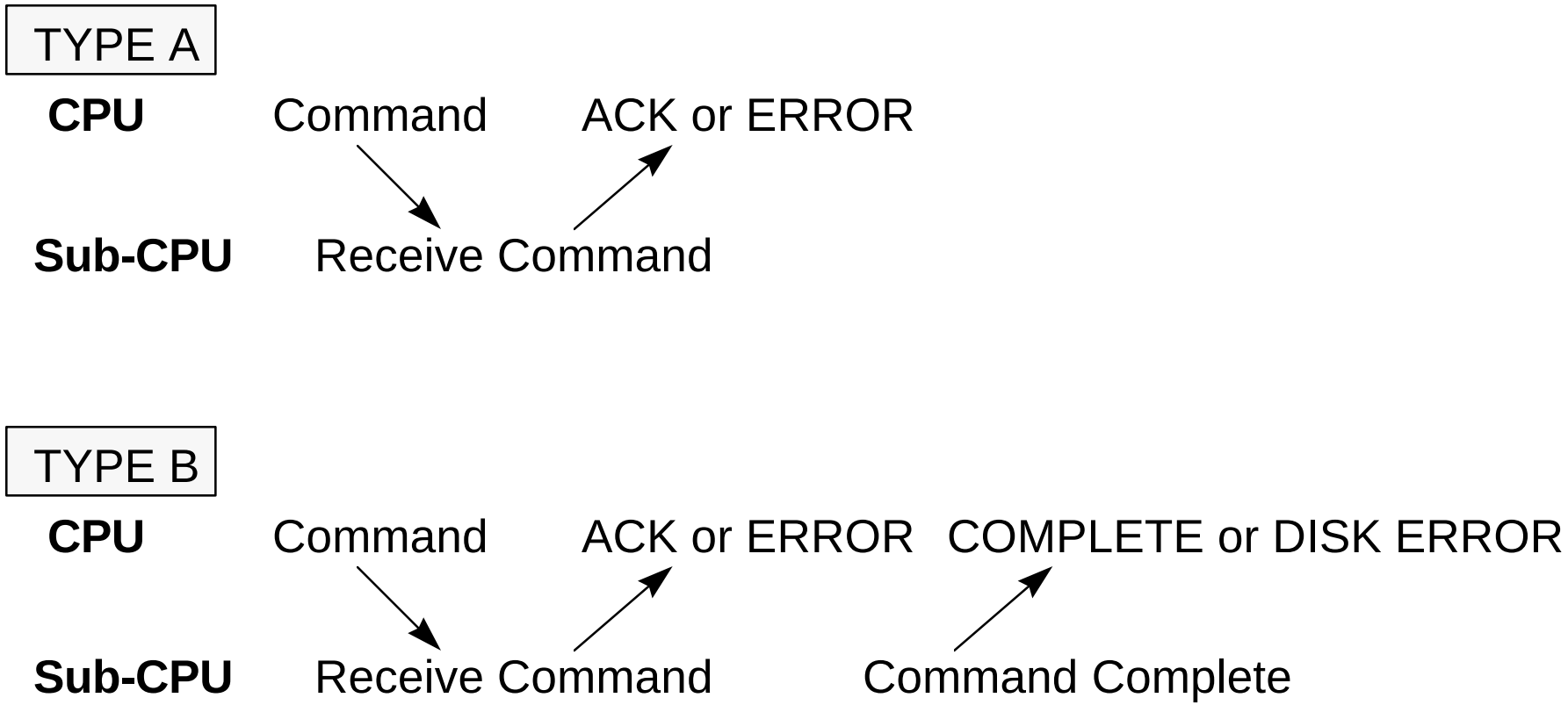
- Includes the streaming library

Using LibCD

```
main{
        ResetCallback();
        CdInit();
                :
                :
                :
```

Initial program initialization

SUB-CPU command interface



Low level functions (1)

- CdControlB — Block until COMPLETE
- CdControl — Block until ACK
- CdControlF — Non-blocking

Parameter 1	Command code	Macro declaration
Parameter 2	Parameter	char[4]
Parameter 3	Result	char[8]
Return value	0:Error 1:ACK	

Low level functions (2)

int CdSync(int mode, u_char *result)

int mode 0:block mode 1:non block

Determine state of Sub-CPU

Return value CdlComplete	Command end
CdlDiskError	Error detected
CdlNoIntr	Command executing

Reading data

- Set location
- Start read
- Wait for completion and detect errors

Set location

(Example) 20 minutes, 2 seconds, sector 60

```
pos.minute      =       0x20;
pos.second      =       0x02;
pos.sector      =       0x60;

while(CdControl(CdlSetloc, &pos, 0) == 0);
```

Start reading

(Example) Read sector 100 using multi-speed

```
while(CdRead(100, buff, CdlModeSpeed) == 0);
```

- CdRead returns immediately after the command is issued
- CdRead is a high level function

Waiting for completion and detecting errors

int CdReadSync(int mode, u_char *result);
int mode 0:block 1:non block

- Return value indicates number of remaining unread sectors
- Error is indicated when -1 is returned on a read
- Monitor as non block [non-blocking] when reading in the background

Error processing

- Retry when CdReadSync return value is -1
- Retry after executing CdSetLoc

Obtaining the location from the filename

CdlFILE *CdSearchFile(CdlFILE *fp, char *name);

- Example name is "\\sce\\sample.dat;1"
- Location is in fp->pos
- Directory no. 40, File no. 30 inside the directory

Increasing read speed

- Reduce number of files read
- Access absolute sector
- Use background reading effectively
- Do not split the program

Description of DA/XA/Streaming data

Sample Demonstration

CD-DA Playback

CD-DA

- Abbreviation for Compact Disc Digital Audio
- Same format as a music CD
- Plays back at standard speed
- Information located from the data track and beyond

CD-DA playback method

- Set volume
- Set mode
- Set playback point and start playback
- Process errors
- Detect termination

Setting volume

- Volume is controlled through the SPU library
- Set serial attribute and serial volume
- Normally, set MAX value to 7f

Setting mode

- Set mode with CdlSetmode command in the sub-CPU
- CD-DA Mode (CdlModeDA)
- Report Mode (CdlModeRept)
  Report status of the sub-CPU 10X/second

Setting playback point and starting playback

- Set with CdlSetloc command in the sub-CPU
- For parameters, give minute, second, sector as members of the CdlLOC structure
- Format is BCD decimal notation
  (Example) 1 minute, 2 seconds, sector 30 -->
  (0x01, 0x02, 0x30)
- Begin playback with CdlPlay command

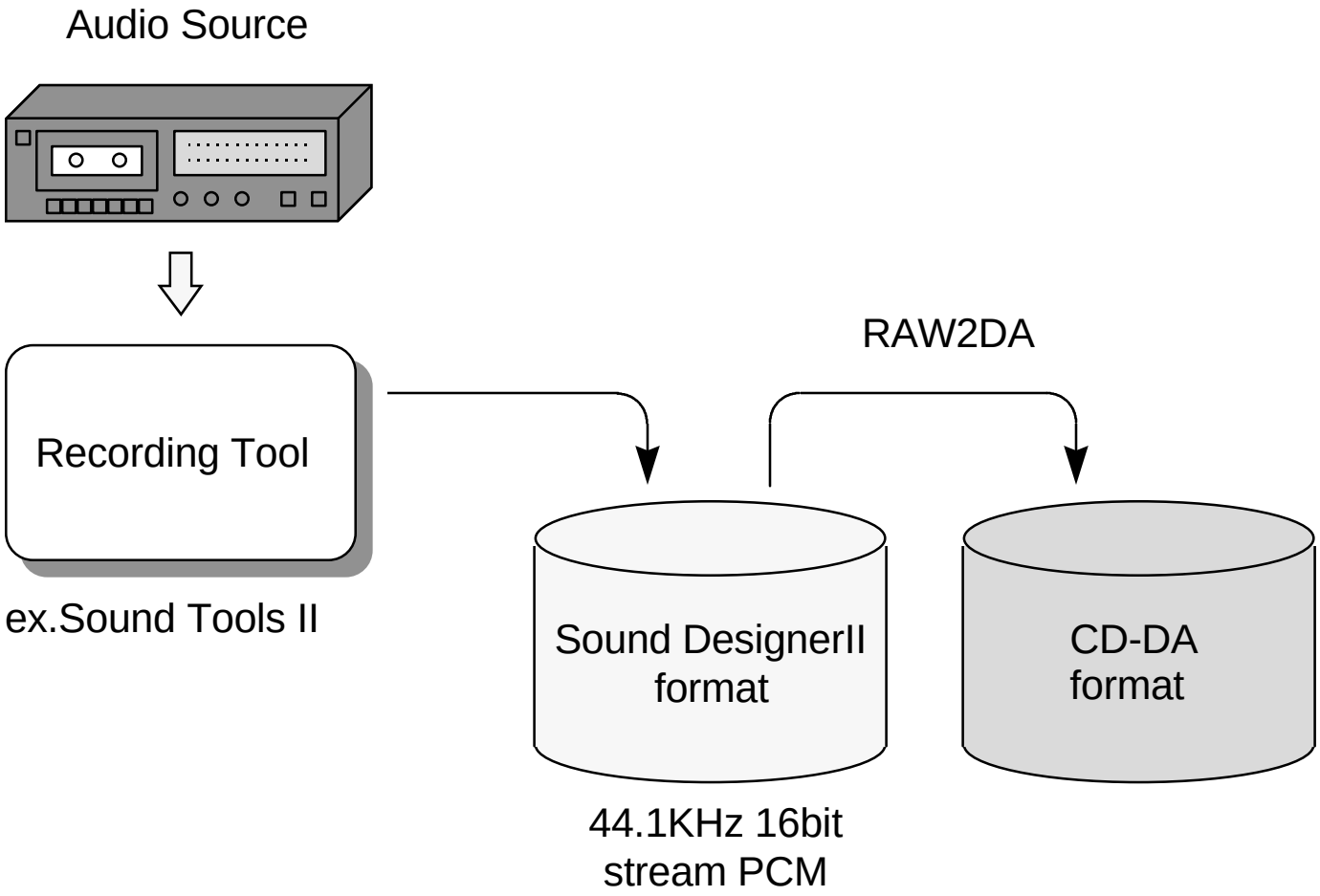
Error processing

- Retry is attempted when command is not received
- Monitor DISK ERROR after start of recording

Detecting termination

- Information obtained in report mode
- Comparison with a previously determined location
- Comparison is easy using logical sectors

Creating a CD-DA file



CD-XA Playback

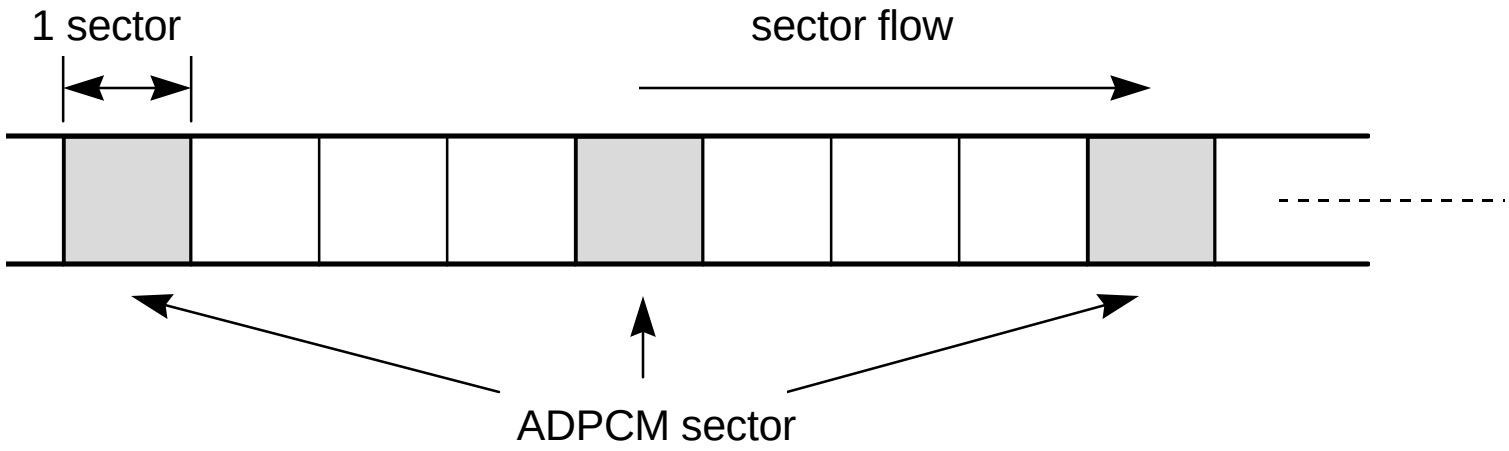
CD-XA

- ADPCM sound compression
- Can interleave compressed sound and data in available sections of the CD
- Two modes, B and C, depending on the rate of compression
- Another sound channel can generate sound in LR stereo

ADPCM (XA audio)

Data quantity is 1/4 - 1/16 of CD-DA

Fs: changes 37.8 KHz / 18.9 KHz according to stereo/monaural



Adjust insertion and transfer of other sectors to come between ADPCM sectors

⇩

Interleave

CD-XA playback method

- Set volume
- Select channel
- Specify playback point
- Set mode and start playback
- Process errors
- End processing

Selecting the channel

- Make the "file" member of the CdlFILTER structure to "1"
- Set the desired channel in the "chan" member of the CdlFILTER structure
- Set these using the CdlSetfilter command

Setting the mode and starting playback

- Mode

  CdlModeSpeed (during multi-speed)
  CdlModeRT ADPCM playback mode
  CdlModeSF Channel set mode

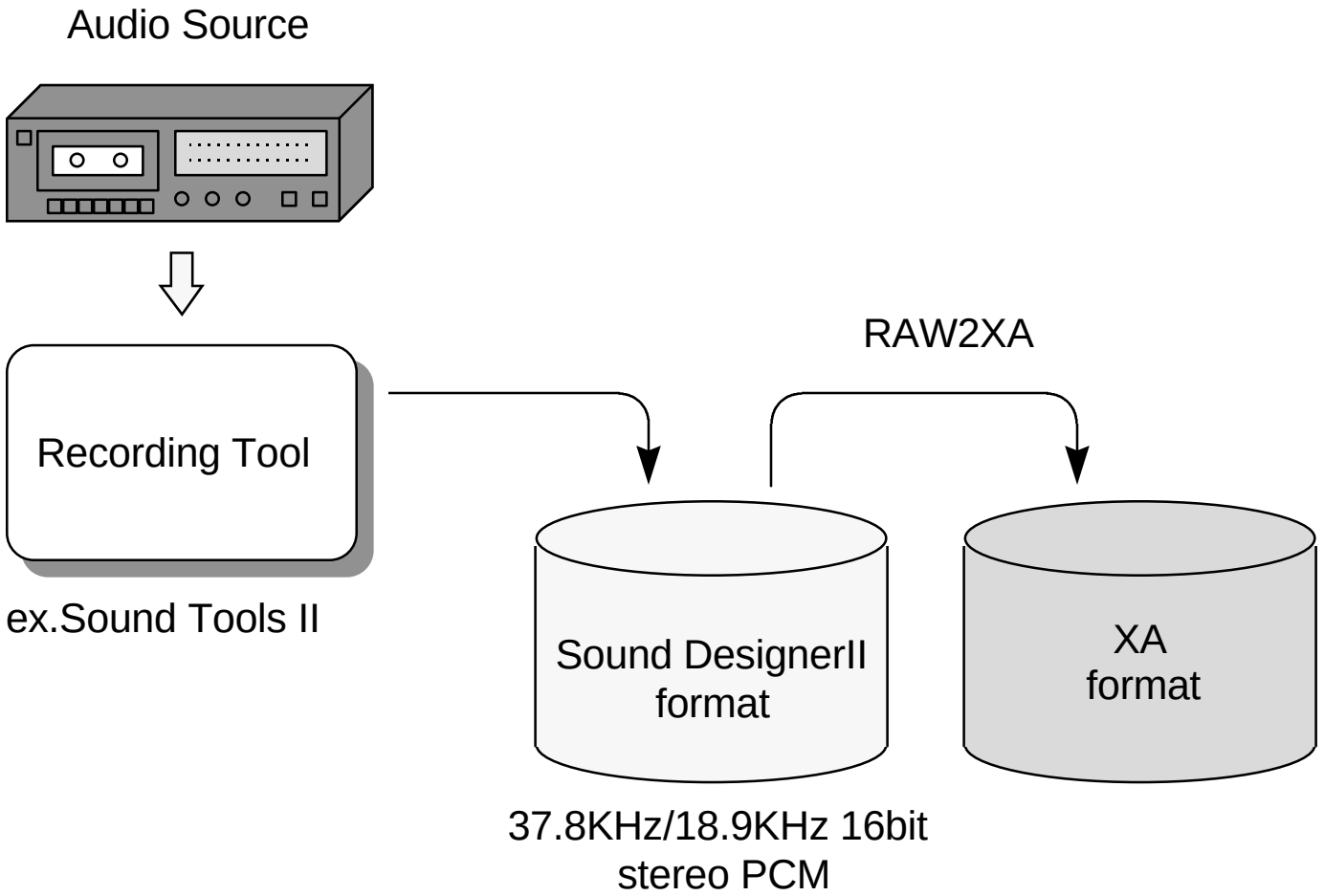
- Start playback and set mode

  CdRead2(mode)

End processing

- No report mode in XA
- Obtain current location with CdlGetlockL command

Creating an ADPCM (XA) file

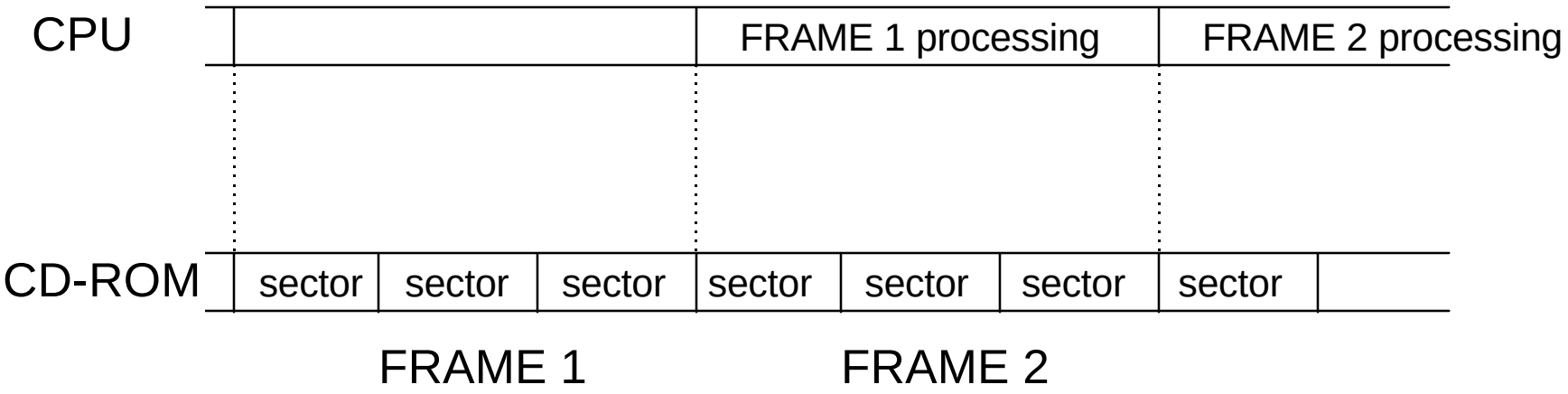


Streaming Library

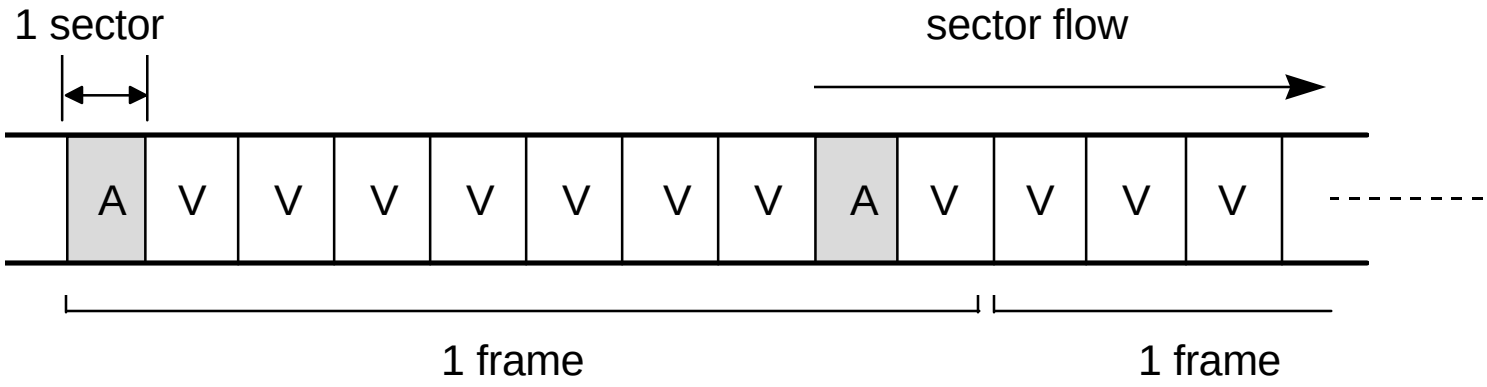
Streaming

- Frames of data are processed one after another without stopping the CD-ROM rotation

- A unit of processing is called a FRAME and is a collection of multiple sectors



Structure of movie data sector



- The audio data (ADPCM) sector is interleaved at the rate of 1:8

- The number of sectors required for one frame's worth of data depends on the movie frame rate

⇩

Automatically generated by the movie converter

Services of the streaming library

- Frame processing
- Ring buffer
- Synchronous processing
- Error processing

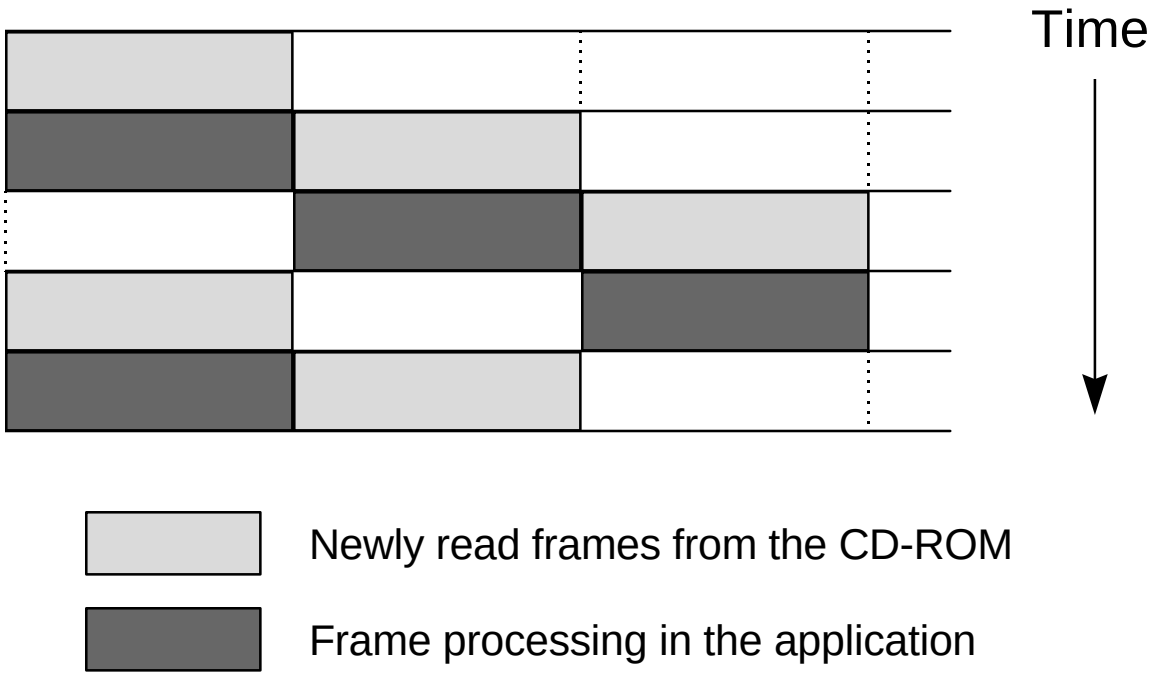
Frames

- A logically related collection of sectors
  (Example) a movie frame
- The streaming library works in units of frames
- The number of sectors which makes up a frame varies per frame
- A 32-byte sector header is included to implement a frame

Ring buffer

- Library data is stored in a ring buffer
- The ring buffer area is maintained by the application
- The boundary of the ring buffer does not cross a frame
- The ring buffer keeps BUSY/FREE status per frame

Ring buffer transitions



Synchronous processing

- Application and CD-ROM reads are asynchronous
- When processing is delayed, the ring buffer will not empty
- If the ring buffer is not empty and a new frame cannot be inserted, the library will not insert the frame (overflow)
- When processing is too fast, the ring buffer will empty (underflow)
- When a new frame cannot be extracted, the previous frame will be used in the application

Error processing

- Errors are detected on a sector basis
- Frames which have sector errors are not inserted in the ring buffer

Streaming operation

- Set up ring buffer (StSetRing)
- Initialize streaming library (StSetStream
- Start CD-ROM (CdRead2)
- Acquire a frame (StGetNext, StFreeRing)
- Detect end
- Postprocess (StUnSetRing)

Setting up the ring buffer

- Allocate ring buffer area on the application side
- Set buffer size to 4-5 times size of frame
- Pass the address and size of the ring buffer area to the library with StSetRing()

Initialize streaming library

```
void StSetStream(u_long mode,
                 u_long start_frame, u_long end_frame,
                 int (*func1)(), int (*func2)());
```

mode	1 if processing 24-bit data for animation
start_frame	Start streaming trigger frame Set to 0 if no trigger
end_frame	Callback of each frame
func2	Normally, 0

Streaming start

- After specifying location with CdlSetloc command

int CdRead2(long mode);

mode	CdlModeStream	Always necessary
	CdlModeSpeed	Multi-speed play
	CdlModeRT	Simultaneously playback ADPCM audio

Frame acquisition

u_long StGetNext(u_long **addr, u_long **header);

u_long *addr address of frame data
u_long *header address of frame header

Return value 0 = acquired frame is available in the ring buffer
1 = acquired frame is not available in the ring buffer

After frame buffer processing in the application, free with StFreeRing()

End detection

- The absolute frame number is part of the streaming data
- It is monitored when end is detected
- The frame header is cast to StHEADER and compared with the FrameCount
- Actual data has 4-5 dummy frames

Postprocessing

- Streaming and CdRead do not coexist
- After streaming, postprocessing occurs
- PAUSE or STOP
- StUnSetRing()

Movies using MDEC

- Long movie playback is possible by combining MDEC and streaming
- Use MDECs libpress library
- A streaming frame is one movie frame

Movie data flow

- Streaming one frame of data
- VLC decoder (lossless) [CPU]
- Expansion of MDEC (lossy) [MDEC]

Compression/decompression method

- MDEC Initialization
- VLC Decoder — DecDCTvlc()
- Input to MDEC — DecDCTin()
- Output from MDEC — DecDCTout()
- Switch frames

MDEC initialization

- DecDCTReset (0)
  Processing is delayed since the coefficient table is transferred

- DecDCTReset (1)
  Processing is sped up as a result of Warm Reset

- Must issue DecDCTReset (0) first

VLC decoder

- Decodes one frame of acquired data through the streaming library

```
int DecDCTvlc(u_long *bs, u_long *buf)
```

u_long *bs	input
u_long *buf	output

- Decompression takes time because it is performed in the CPU
- DecDCTvlc() does not return until decompression completes
- Data can be decompressed separately

Image decode process using MDEC

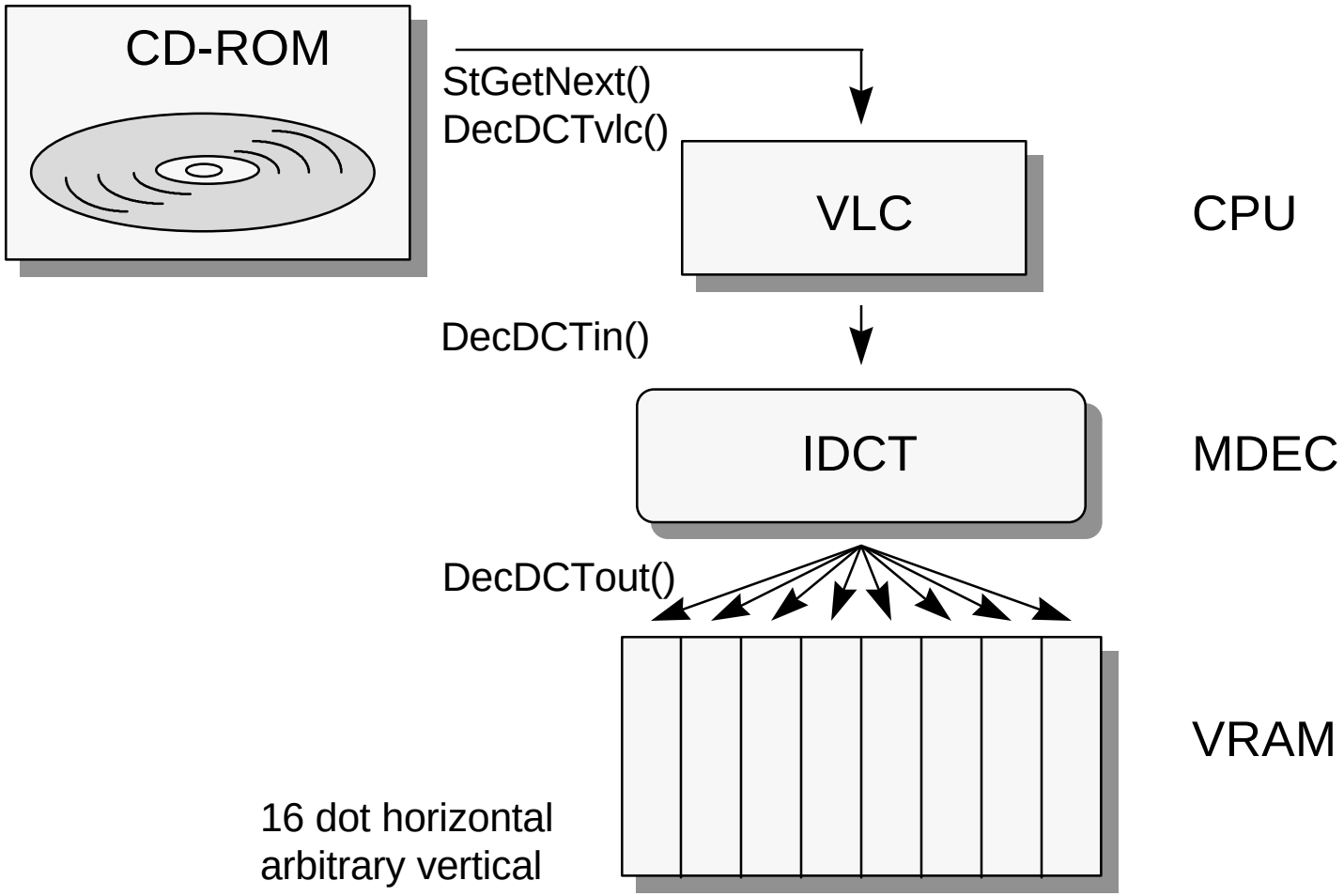


Table buffer

- During VLC decompression, since MDEC also aligns, VLC output is buffered in a table

MDEC input

void DecDCTin(u_long *buf, int mode);

u_long *buf	start address of input region
int mode	RGB24bit/RGB16bit

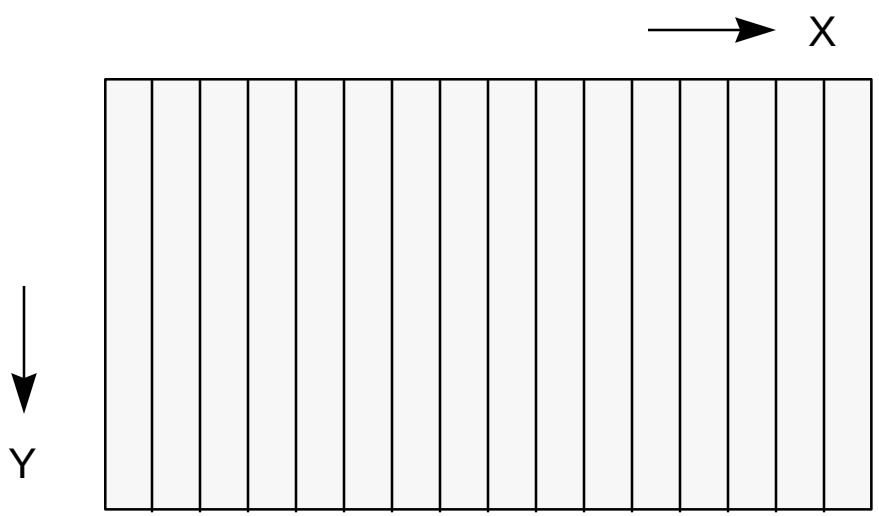
Input the input size to the input data
Since this is only setting, this function returns immediately

MDEC output

void DecDCTout(u_long *buf, int size);

u_long *buf	start address of output region
int size	receive size

Image of the output picture

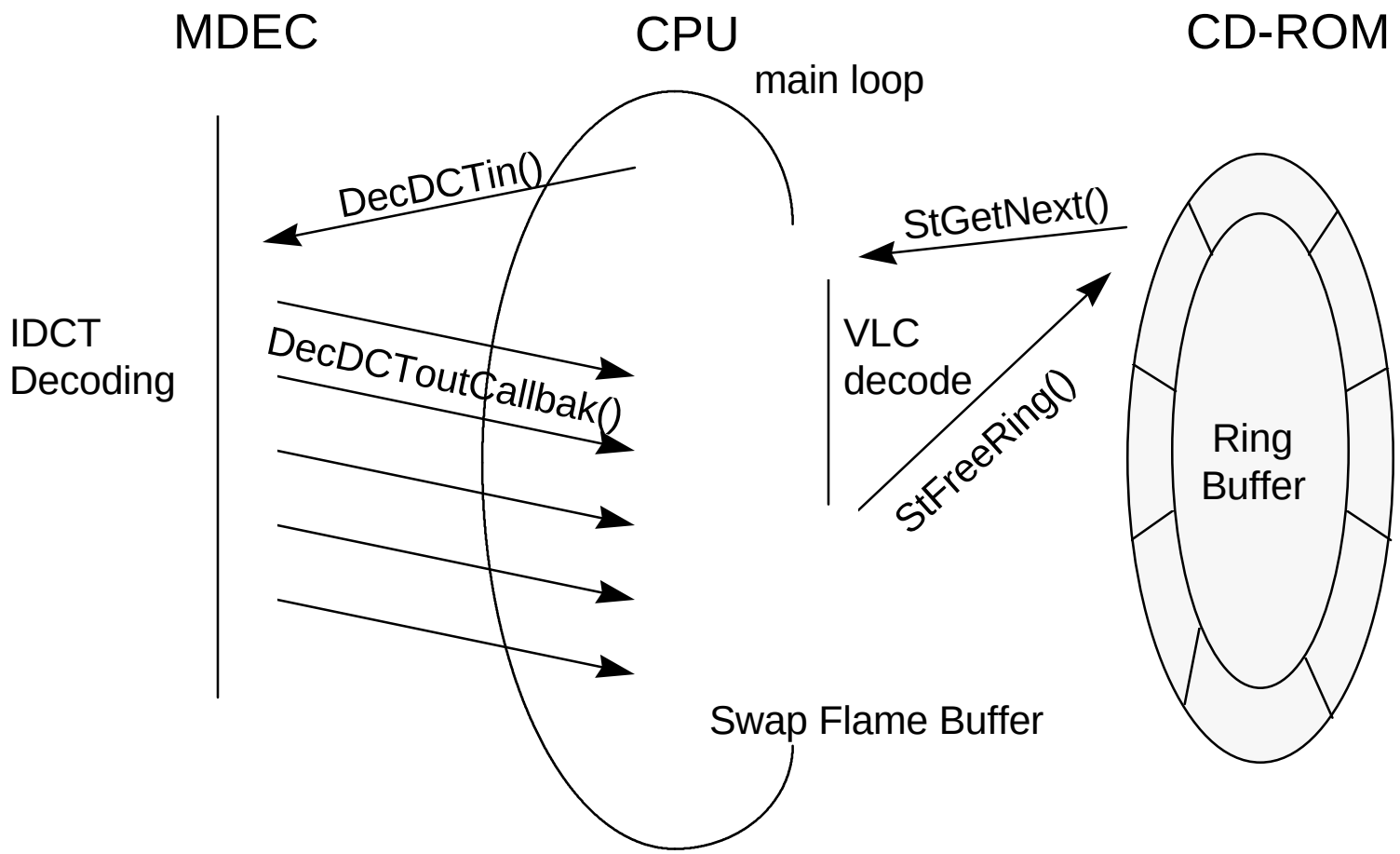


16 dot horizontal
arbitrary vertical

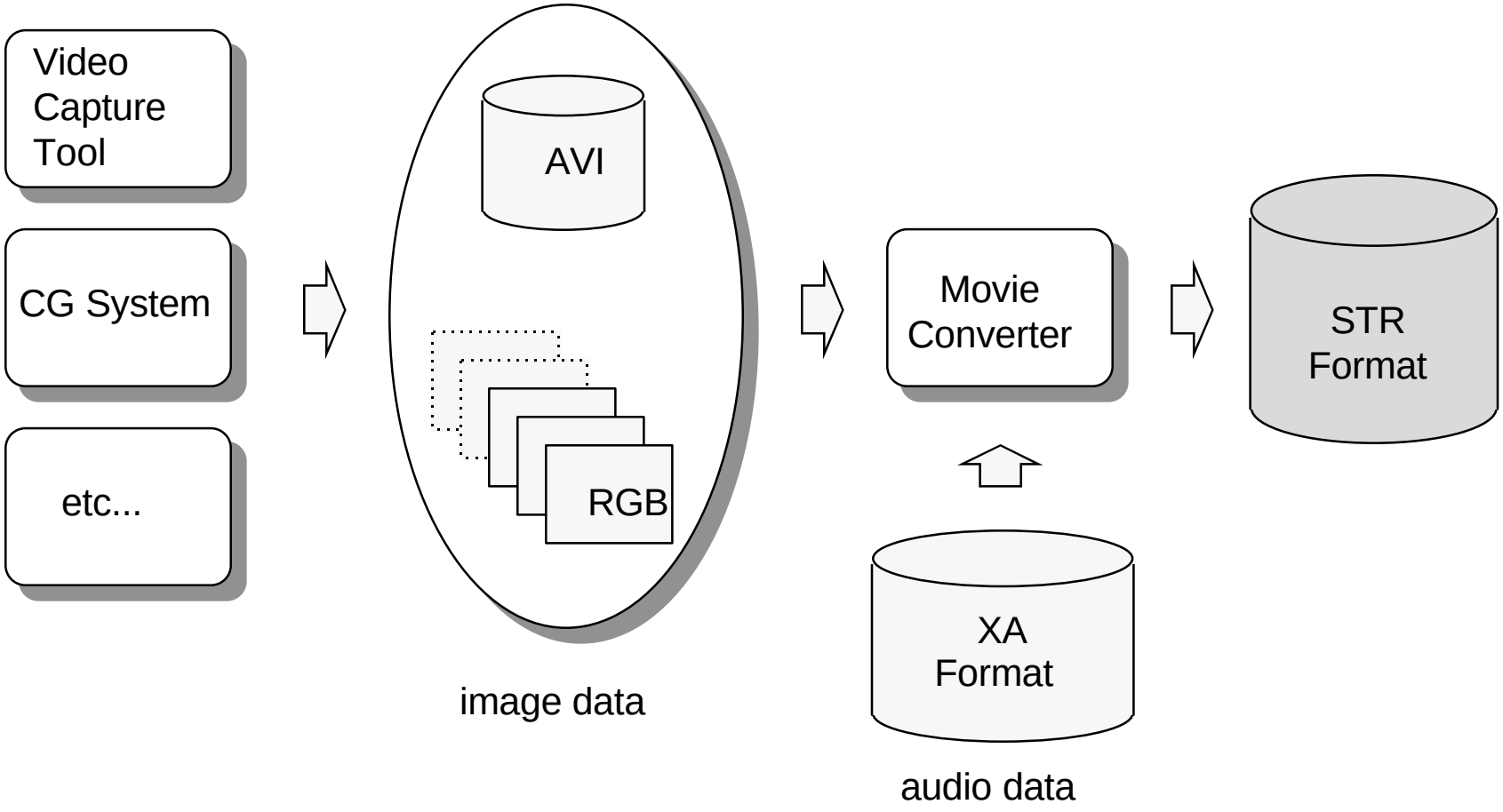
Switching frames

- The image packet that was output is expanded to VRAM
- The screen is switched after expanding all of the image packets on one screen to VRAM

Animation program timing chart



Creating movie data



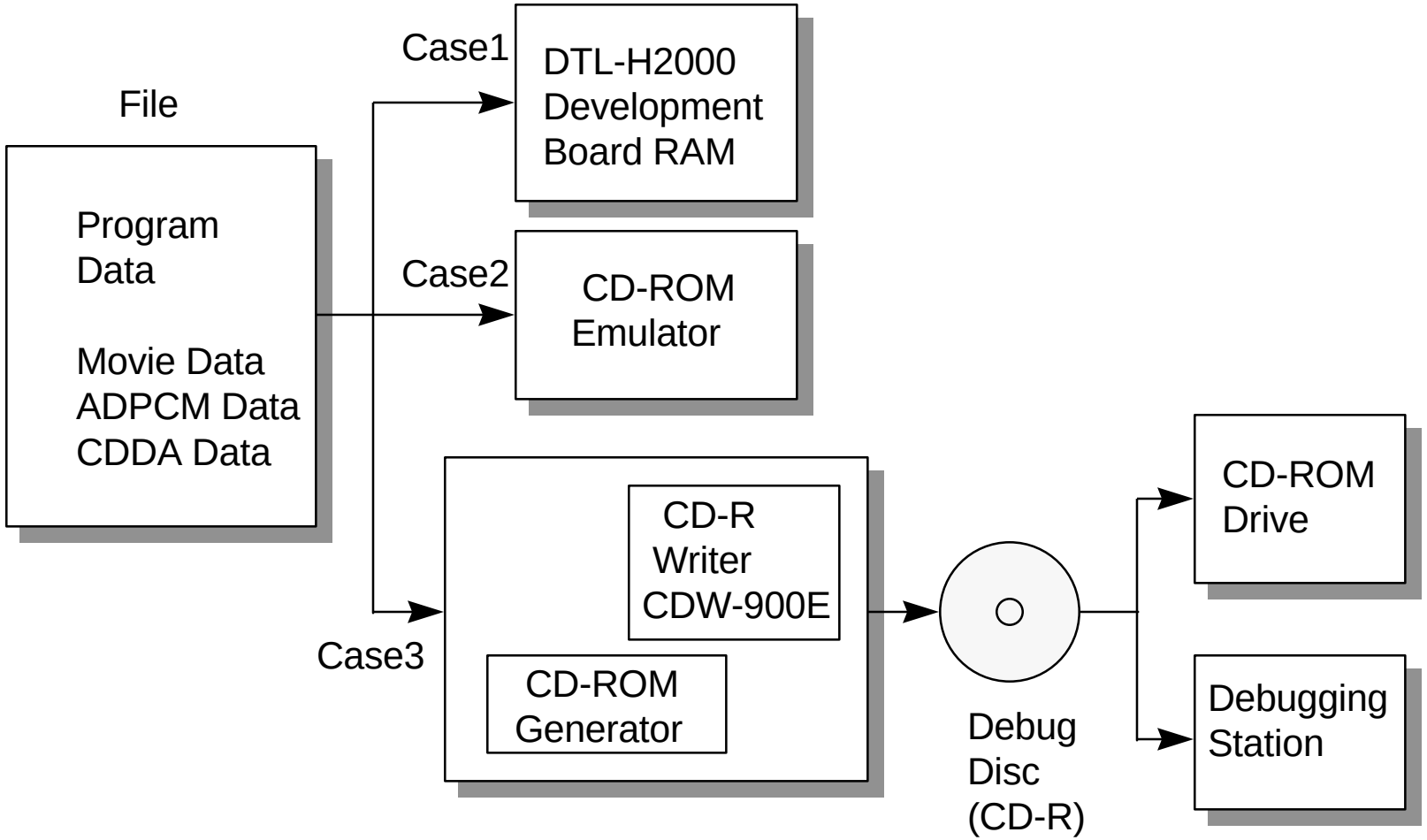
Execution from the CD-ROM/ CD-ROM Emulator

Sample Demonstration

Program operating method

- Execute by downloading to the DTL-H2000
- Execute from the CD-ROM Emulator
- Execute from the CD-ROM

Development method



CD-ROM Emulator

- Build the CD-ROM image to a dedicated hard disk
- Provide access timing of the nearest CD-ROM drive to the PlayStation
- Not necessary to burn the CD-R

CD-ROM Generator

- CD-R Writer (CDW-900E) and creation software (for Windows)

- Necessary to create the CD-R for debug and master disk

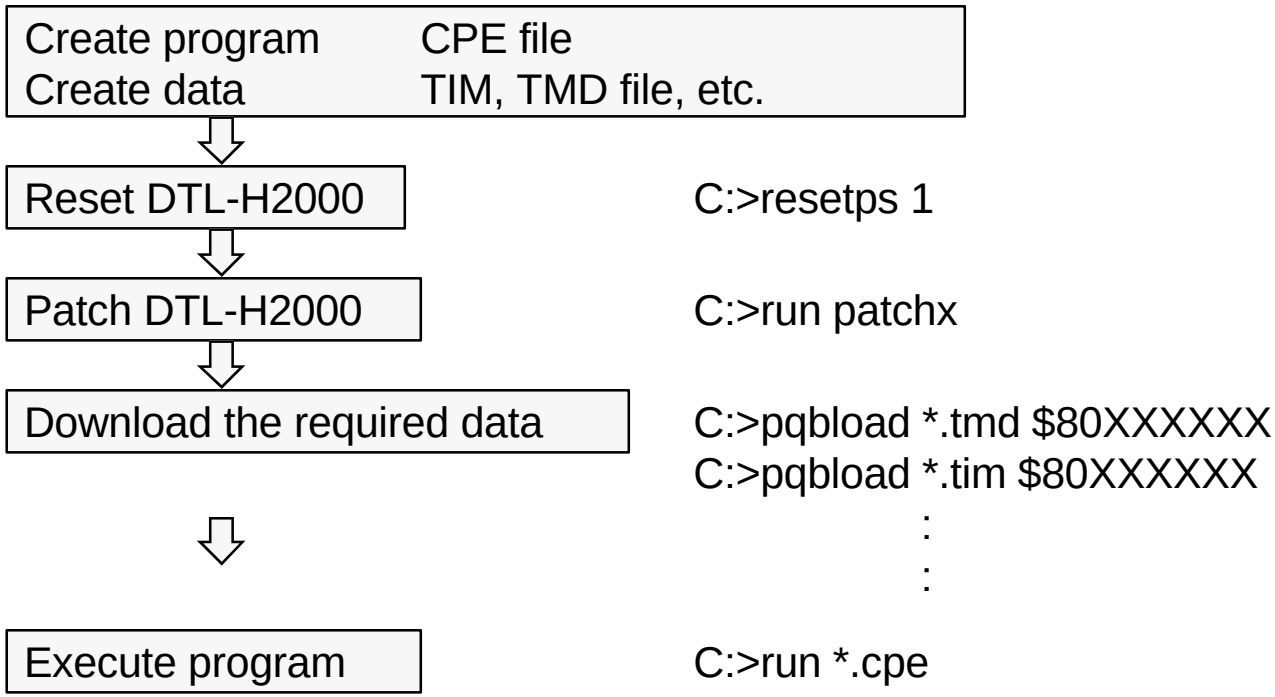
Sample program

```
/* Sample program when loadingand executing the PSX.EXE on the CD-ROM/
        CD-ROM Emulator
        "cdexec.c"
        [steps]     C:>ccpsx -g -Xo$801ff000 cdexec.c -ocdexec.cpe
                    C:>run patchx
                    c:>run cdexec
*/
void main()
{
        _96_remove();
        _96_init();
        LoadExec("cdrom:\\PSX.EXE;1", 0x801fff00, 0);  [1]
                    /* filename   stack   stack size */
                       /* filename can be up to 20 characters */
}
```

[1] Since LoadExec is a routine within ROM there is no problem if this program and the PSX.EXE effective address overlap

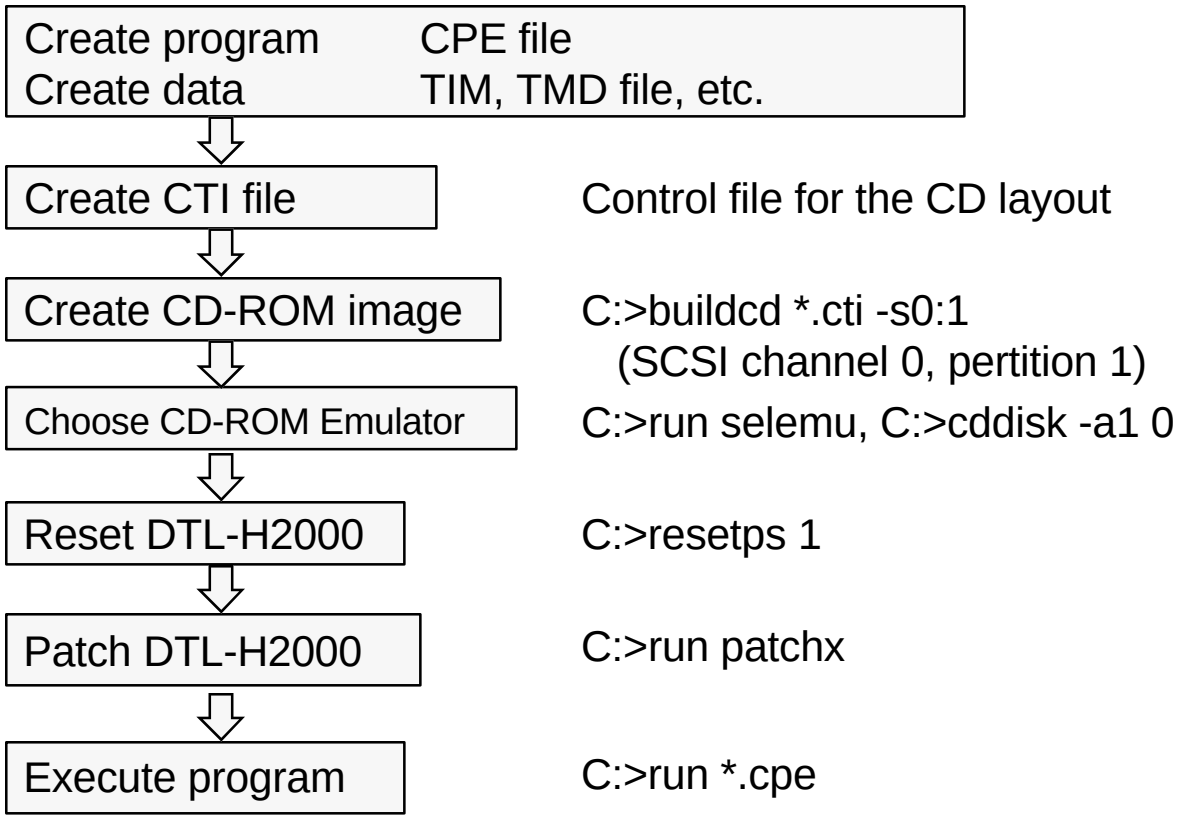
Appendix

Downloading to the DTL-H2000



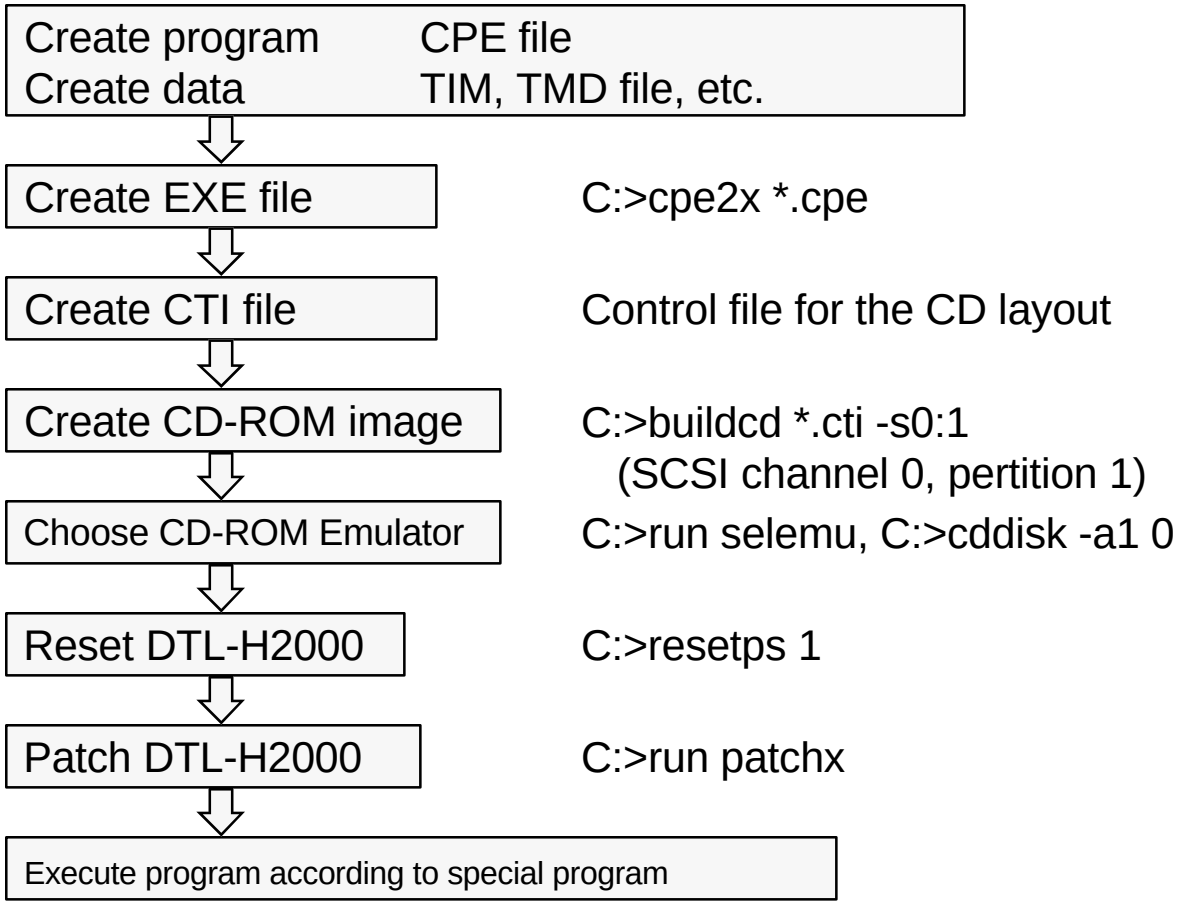
Execution from the CD-ROM Emulator (1)

Imaging only the data file



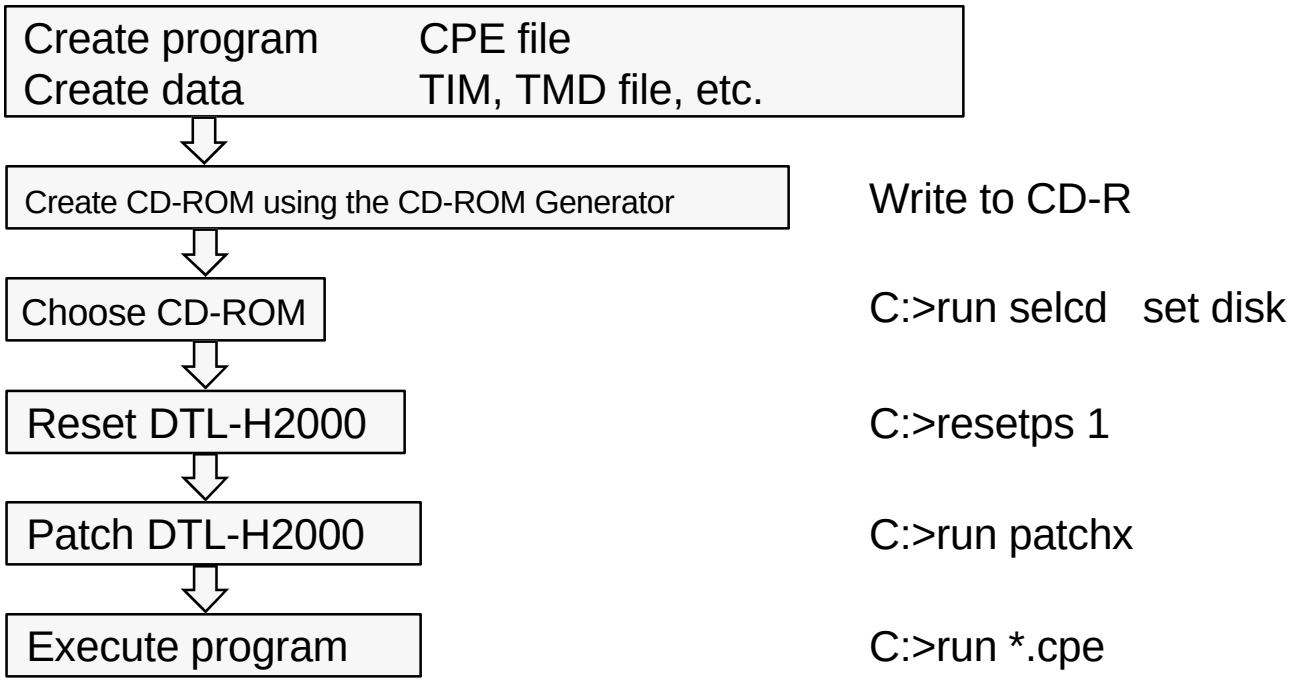
Execution from the CD-ROM Emulator (2)

Imaging all files



Execution from CD-ROM (1)

Make a CD-ROM of the data file only



Execution from CD-ROM (2)

Make a CD-ROM of all files