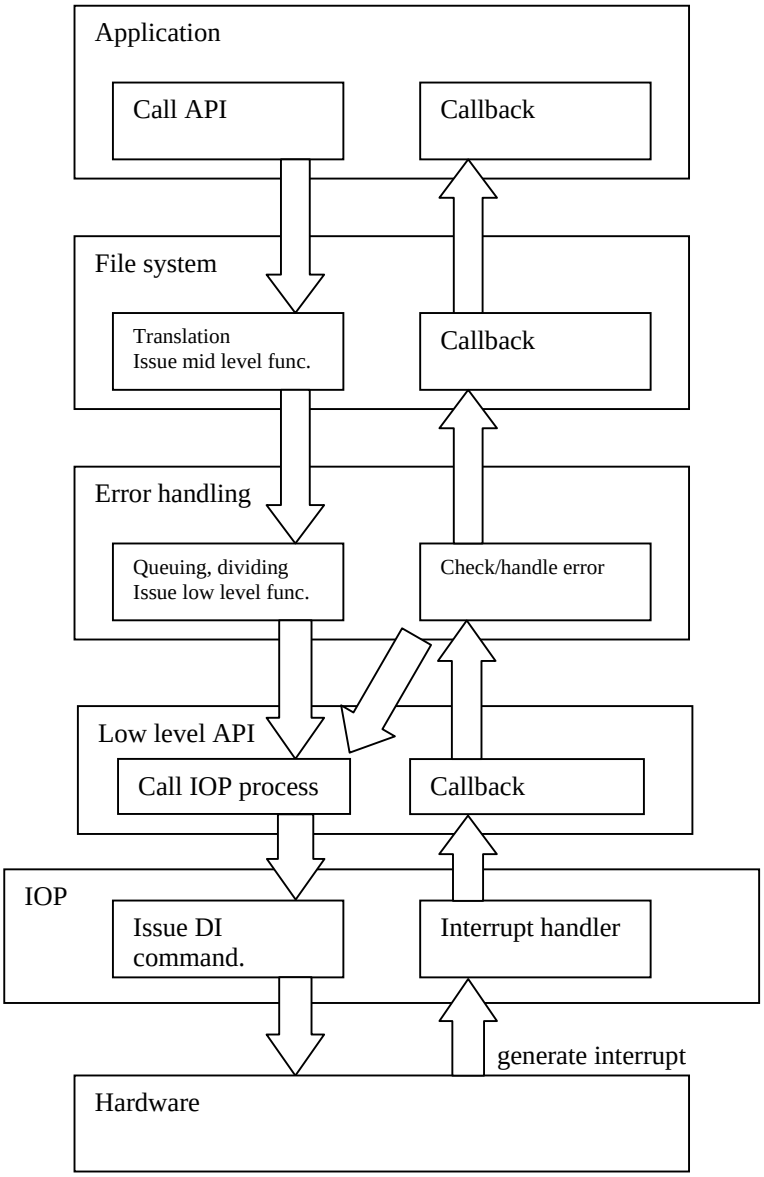
1. Software structure

•

Each layer has several functions. Since most functions are executed asynchronously, those kind of functions have callbacks to notify the result to the caller. The types of arguments and return values depend on the layer.



The reason why there's an arrow from "Check/handle error" part of the error handling layer towards the low level layer is because the error handling layer might issue a command according as the result of the low level layer. One example is when an error occurred. The result of the low level interrupt handler is simply "DEINT occurred", so the error handling layer has to issue low level "Request Error" command to know what happened.

2. Low-level layer(raw)

The main role of this layer is to provide a simple way to set DI registers. This layer consists of three parts.

- Interrupt handler
- Virtual disk image
- API

These are explained in later sections.

Its concept is being a thin layer above hardware. Although all interrupts should be disabled while setting registers (if an interrupt occurs after setting half of the registers and the interrupt handler or its callback tries to issue another drive command, it crushes after the interrupted program resumes), the API doesn't do disable-restore-interrupts stuffs. This is because this function is always called with interrupts disabled, so disabling interrupts in API is redundant. In other words, one who calls this function directly (NOTE: low-level APIs should not be called directly, though. They are too low-level!) has to disable interrupts before he/she calls API.

The part of raw layer is implemented in IOP. If you call these low-level APIs, IOP will execute corresponding processes.

2.1 Interrupt handler

Interrupt handler is called from Hollywood's interrupt handler. It does the following:

- Checks which interrupts occurred by checking DI Status register and DI Cover register.
- Clear the interrupts that fired.
- Call callback if specified passing interrupts information.

Please note that the corresponding INT bit is set even though the interrupt is masked. So interrupt handler should check both INT bit and MASK bit and masked interrupt should not be reported to callback nor should not be cleared. For example, when CVRINT and TCINT fired but CVRINT is masked, interrupt handler will see that both CVRINT and TCINT are set. However, interrupt handler should check that CVRINT is masked so it should not clear CVRINT.

2.2 Virtual disk image

The data on Revolution disk is recorded with encrypted special format for security. Raw layer must make it easy to access user data from Broadway. When the data is accessed, IOP have to decrypt the data and store the original image into specified buffer.

It appears from Broadway, as it were, the original data is still there. It is called "virtual disk image." This image is decrypted and excluded the security system area, such as hash area.

Only DVDLow[Read|Seek]Rvl functions can access "virtual disk image." Any other function can access only the data with encrypted special format.

2.3 APIs

This section describes low level function APIs. Note that none of these should be used from application. These are for internal device driver use.

All the low-level command issuing DI command is asynchronous.

2.3.1 DVDLowRead

```
#include<secure/dvdlow.h>

BOOL DVDLowRead( void* addr, u32 length, u32 offset_4_byte, DVDLowCallback callback);
```

`DVDLowRead` issues a command to the drive to read the file into the buffer specified by *addr*. *Length* and *offset_4_byte* specify which part to read, where *offset_4_byte 0* means the top of the disk. *Offset_4_byte* specifies address per 4-byte. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.2 DVDLowReadDiskID

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDiskID(DVDDiskID* diskID, DVDLowCallback callback);
```

DVDLowReadDiskID issues a command to the drive to read disk id of the disk into the buffer specified by *diskID*. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.3 DVDLowSeek

```
#include<secure/dvdlow.h>

BOOL DVDLowSeek(u32 offset_4_byte, DVDLowCallback callback);
```

DVDLowSeek issues a command to seek. *Offset_4_byte* specifies which part to seek per 4-byte, where *offset_4_byte 0* means the top of the disk. The function specified by *callback* is invoked when the seek finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.4 DVDLowStopMotor

```
#include<secure/dvdlow.h>

BOOL DVDLowStopMotor(BOOL eject, BOOL saving, DVDLowCallback callback);
```

DVDLowStopMotor issues a command to stop the motor or to eject the disc. *Eject* specifies whether to eject or not. *Saving* specifies whether to shift to power saving mode or not. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.5 DVDLowWaitForCoverClose

```
#include<secure/dvdlow.h>

BOOL DVDLowWaitCoverClose(DVDLowCallback callback);
```

DVDLowWaitCoverClose simply enables CVRINT (CVRINT is basically disabled). The function specified by *callback* is invoked when the cover is closed; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.6 DVDLowInquiry

```
#include<secure/dvdlow.h>

BOOL DVDLowInquiry(DVDDriveInfo* info, DVDLowCallback callback);
```

DVDLowInquiry issues a command to inquiry the command. Drive information is stored in the buffer specified by *info*. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

Although inquiry command is acceptable in both DMA and IMM mode, DMA mode is used for this API.

2.3.7 DVDLowRequestError

```
#include<secure/dvdlow.h>

BOOL DVDLowRequestError(DVDLowCallback callback);
```

DVDLowRequestError issues a command to get error code. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The error code retrieved from the drive is stored in DI Immediate Buffer register.

2.3.8 DVDLowReset

```
#include<secure/dvdlow.h>

void DVDLowReset(BOOL spinup);
```

DVDLowReset resets the drive. *Spinup* switches to spin up the drive or not. It doesn't tell the caller when reset finishes; however, host can issue another low-level function whether or not reset finishes. Flipper checks the reset signal of DI and issues the next command after the reset finishes.

This function is not asynchronous.

2.3.9 DVDLowAudioStream

```
#include<secure/dvdlow.h>

BOOL DVDLowAudioStream(u32 subcmd, u32 length, u32 offset_4_byte, DVDLowCallback callback);
```

DVDLowAudioStream issues a command to control audio streaming. *Subcmd* specifies entry or cancel. *Length* and *offset_4_byte* specify which part to stream, where *offset_4_byte 0* means the top of the disk. *Offset_4_byte* specifies address per 4-byte. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.10 DVDLowRequestAudioStatus

```
#include<secure/dvdlow.h>

BOOL DVDLowRequestAudioStatus(u32 subcmd, DVDLowCallback callback);
```

DVDLowRequestAudioStatus issues a command to get audio status. *Subcmd* specifies types of audio streaming information. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.11 DVDLowAudioBufferConfig

```
#include<secure/dvdlow.h>

BOOL DVDLowAudioBufferConfig(BOOL enable, u32 size, DVDLowCallback callback);
```

DVDLowAudioBufferConfig issues a command to set audiio buffer configuration. *Enable* switches audio buffer to enable or disable. *Size* specifies the number of pages in audio buffer. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.12 DVDLowSetResetCoverCallback

```
#include<secure/dvdlow.h>

DVDLowCallback DVDLowSetResetCoverCallback(DVDLowCallback callback);
```

DVDLowSetResetCoverCallback sets the callback for when the cover is closed after reset is deasserted.

The return value is old callback routine.

This function is not asynchronous.

2.3.13 DVDLowBreak

```
#include<secure/dvdlow.h>

BOOL DVDLowBreak(void);
```

DVDLowBreak breaks a running command. This function does not use DI break transaction.

The return value is always TRUE.

2.3.14 DVDLowClearCallback

```
#include<secure/dvdlow.h>

DVDLowCallback DVDLowClearCallback(void);
```

DVDLowClearCallback clears callback routine.

The return value is old callback routine.

This function is not asynchronous.

2.3.15 DVDLowGetCoverStatus

```
#include<secure/dvdlow.h>

u32 DVDLowGetCoverStatus(void);
```

DVDLowGetCoverStatus returns cover status.

This function is not asynchronous.

2.3.16 DVDLowReadDvd

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDVD(void* addr, u32 length, u32 lsn, DVDLowCallback callback);
```

DVDLowReadDvd issues a command to the drive to read the file into the buffer specified by *addr*. *Length* and *lsn* specify which part to read, where *lsn 0* means the top of the disk. *Lsn* specifies the logical sector number of the disk. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.17 DVDLowReadDvdConfig

```
#include<secure/dvdlow.h>

u32 DVDLowReadDVDConfig(BOOL set, u32 type, u32 config, DVDLowCallback callback);
```

DVDLowReadDVDConfig issues a command to set up the configuration for “Read DVD”. *Set* specifies read or write. *Type* specifies items to configure. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.18 DVDLowReadDvdPhysical

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDvdPhysical(DVDVideoPhysical* physical, u32 layer, DVDLowCallback callback);
```

DVDLowReadDvdPhysical issues a command to the drive to read physical information of the disk into the buffer specified by *physical*. *Layer* specifies a number of layer. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.19 DVDLowReadDvdCopyright

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDvdCopyright(DVDVideoCopyright* copyright, u32 layer, DVDLowCallback callback);
```

DVDLowReadDvdCopyright issues a command to the drive to read copyright information of the disk into the buffer specified by *copyright*. *Layer* specifies a number of layer. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.20 DVDLowReadDvdDiscKey

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDvdDiscKey(DVDVideoDiscKey* discKey, u32 layer, DVDLowCallback callback);
```

DVDLowReadDvdDiscKey issues a command to the drive to read disc key information of the disk into the buffer specified by *discKey*. *Layer* specifies a number of layer. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.21 DVDLowReadDvdBca

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDvdBca(DVDVideoBca* bca, u32 layer, DVDLowCallback callback);
```

DVDLowReadDvdBca issues a command to the drive to read bca information of the disk into the buffer specified by *bca*. *Layer* specifies a number of layer. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.22 DVDLowReportKey

```
#include<secure/dvdlow.h>

BOOL DVDLowReportKey(DVDVideoReportKey* reportKey,
                     u32 format, u32 lsn, DVDLowCallback callback);
```

DVDLowReportKey issues a command to the drive to read “Copyright Management Information” and “Title key” into the buffer specified by *reportKey*. *Key* specifies key format. *Lsn* specifies the logical sector number of the disk. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.23 DVDLowOffset

```
#include<secure/dvdlow.h>

BOOL DVDLowSetOffset(u32 subcmd, u32 offset_4_byte, DVDLowCallback callback);
```

DVDLowSetOffset sets offset address for “Read”. *Subcmd* specifies set or get. *Offset_4_byte* specifies offset address per 4-byte, where *offset_4_byte 0* means the top of the disk. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.24 DVDLowStopLaser

```
#include<secure/dvdlow.h>

BOOL DVDLowStopLaser(DVDLowCallback callback);
```

DVDLowReadStopLaser issues a command to the drive to stop laser. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

The response packet retrieved from the drive is stored in DI Immediate Buffer register.

2.3.25 DVDLowReadDiskBca

```
#include<secure/dvdlow.h>

BOOL DVDLowReadDiskBca(DVDDiskBca* diskBca, DVDLowCallback callback);
```

DVDLowReadDiskBca issues a command to the drive to read disk information of the disk into the buffer specified by *diskBca*. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.26 DVDLowMeasSerControl

```
#include<secure/dvdlow.h>

BOOL DVDLowMeasSerControl(DVDLowDriveSer* ser,
                          BOOL clear, BOOL enable, DVDLowCallback callback);
```

DVDLowMerSerControl control SER measure and read measured SER into the buffer specified by *ser*. *Clear* specifies clear or not. *Enable* specifies enable or not. The function specified by *callback* is invoked when the command finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.27 DVDLowGetCoverReg

```
#include<secure/dvdlow.h>

u32 DVDLowGetCoverReg(void);
```

DVDLowGetCoverReg returns cover registry value.

This function is not asynchronous.

2.3.28 DVDLowGetLengthReg

```
#include<secure/dvdlow.h>

u32 DVDLowGetLengthReg(void);
```

DVDLowGetLengthReg returns length registry value.

This function is not asynchronous.

2.3.29 DVDLowGetImmBufferReg

```
#include<secure/dvdlow.h>

u32 DVDLowGetImmBufferReg(void);
```

DVDLowGetImmBufferReg returns Immediate buffer registry value.

This function is not asynchronous.

2.3.30 DVDLowEnableInterrupts

```
#include<secure/dvdlow.h>

BOOL DVDLowEnableInterrupts(void);
```

DVDLowEnableInterrupts enables DI interrupts except CVRINT.

The return value is always TRUE.

This function is not asynchronous.

2.3.31 DVDLowMaskCoverInterrupt

```
#include<secure/dvdlow.h>

BOOL DVDLowMaskCoverInterrupt(void);
```

DVDLowMaskCoverInterrupt masks cover interrupt.

The return value is always TRUE.

This function is not asynchronous.

2.3.32 DVDLowClearCoverInterrupt

```
#include<secure/dvdlow.h>

BOOL DVDLowClearCoverInterrupt(void);
```

DVDLowClearCoverInterrupt clears cover interrupt.

The return value is always TRUE.

This function is not asynchronous.

2.3.33 DVDLowReadRvl

```
#include<secure/dvdlow.h>

BOOL DVDLowReadRvl(void* addr, u32 length, u32 offset, DVDLowCallback callback);
```

`DVDLowReadRvl` issues a command to the drive to read the file on the virtual disk image into the buffer specified by *addr*. *Length* and *offset* specify which part to read, where *offset 0* means the top of the virtual disk image. The function specified by *callback* is invoked when the read finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.34 DVDLowSeekRvl

```
#include<secure/dvdlow.h>

BOOL DVDLowSeekRvl(u32 offset, DVDLowCallback callback);
```

DVDLowSeek issues a command to seek on the virtual disk image. *Offset* specifies which part to seek, where *offset 0* means the top of the virtual disk image. The function specified by *callback* is invoked when the seek finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.35 __DVDLowWrite

```
#include<secure/dvdlow.h>

BOOL __DVDLowWrite(void* addr, u32 length, u32 offset_4_byte, DVDLowCallback callback);
```

`__DVDLowWrite` issues a command to the drive to write the data from the buffer specified by *addr*. *Length* and *offset_4_byte* specify which part to write, where *offset_4_byte 0* means the top of the disk. *Offset_4_byte* specifies address per 4-byte. The function specified by *callback* is invoked when the write finishes; however, you can set *callback* to null if you do not want any callback to be invoked.

The return value is always TRUE.

2.3.36 __DVDLowTestAlarm

```
#include<secure/dvdlow.h>

BOOL __DVDLowTestAlarm(const OSAlarm* alarm);
```

`__DVDLowTestAlarm` checks if the alarm specified by *alarm* is used in this Low Level API.

This function is not asynchronous.

2.3.41 __DVDLowCancelForcedTimeour

```
#include<secure/dvdlow.h>

void __DVDLowSetForcedTimeout(OSTime timeout);
```

`__DVDLowWrite` cancels forced timeout.

This function is not asynchronous.

3. Question for BO

- Is it impossible to store H4 and H3 on hashing area?
- How long does it takes in seconds or minutes to calculate all hash values of Revolution disc contents?
- Is it possible to store "raw" data in top of disk image?
  - To understand whether Revolution disk or GameCube disk, we use 32Byte data stored in top of disk image.
- We want to play multi GameCube/Revolution games with one disc.
  - Is it possible to include more than one GameCube/Revolution disc image on one disc?
  - It is necessary to store "raw" disc header data at the head of each image.