CORE driver API modifications

from version 3.2.1 to 3.4.0

Introduction

The intent of this document is to explain to the user the differences between the API of CORE driver version 3.2.1 and CORE driver version 3.4.0 and thus making it possible to easily port IAL based on CORE driver version 3.2.1 to IAL based on 3.4.0 version.

The following sections are the modifications. Note that in the end of each section a note if the change is backward compatible and thus no need to modify IAL, or a modifications is required to the IAL.

1. FUA support for FPDMA commands (NCQ)

`MV_BOOLEAN` FUA field added to the structure `MV_UDMA_COMMAND_PARAMS`.

CORE driver version 3.2.1 issues FPDMA command with FUA unset (force unit access, see S2Ext_1_1 SATA spec). The modification in CORE driver version 3.4.0 is that the driver queues FPDMA read/write command according to the FUA field passed to the driver via mvSataQueueCommand function.

This parameter is effective only when the EDMA is configured in MV_EDMA_MODE_NATIVE_QUEUING mode (NCQ).

The parameter FUA type is MV_BOOLEAN. FUA=MV_TRUE activates force unit access.

This modification is backward compatible since the IAL has to zero out the MV_UDMA_COMMAND_PARAMS data structure before setting its parameters, and thus setting FUA to 0 which is MV_FALSE.

2. Asynchronous calls upon recoverable / unrecoverable errors

The CORE driver calls the asynchronous notification function `mvSataEventNotify` upon detecting SATA / ATA errors. These errors are classified into three categories - (the error category indicated by param2):

- Recoverable sata error
- Unrecoverable sata error
- Device error

The CORE Driver calls the asynchronous notification function `mvSataEventNotify` upon detecting SATA/ATA errors, those errors classified into 3 categories, (the error category indicated by param2):

When unrecoverable sata error detected, the CORE Driver disables the channel and flushes the outstanding command with ABORT status, so it's the IAL's responsibility to do the error recovery (HW reset) as specified by the Adapter spec, then restarting the channel again.

This modifications requires adding a new type to the CORE driver - `MV_EVENT_TYPE_SATA_ERROR. The addition is found in the MV_EVENT_TYPE enum.`

`The second modifications is MV_SATA_ERROR_EVENT enum which describes the type of the event.`

Note that this change is backward compatible

3. Added New function: mvSataGetNumOfPortQueuedCommands

Prototype:

MV_U8 mvSataGetNumOfPortQueuedCommands (MV_SATA_ADAPTER *pAdapter,
MV_U8 channelIndex,
MV_U8 PMPort,
MV_U8 *pCommandsPerChannel)

DESCRIPTION:

Returns the number of posted ATA commands on an EDMA engine for the given SATA port and PMPort.

INPUT:

PAdapter	- Pointer to the adapter's data structure.
channelIndex	- Index of the required channel
PMPort	- Port multiplier port number
PCommandsPerChannel	- If not null, gets the total number of outstanding command for the given channel.

RETURN:

Number of queued commands, 0xFF if error detected.

This change is backward compatible

4. Removed the Auto Flush feature

`mvEnableAutoFlush/mvDisableAutoFlush` functions are removed in CORE driver version 3.4.0. Instead, upon device error (regardless of the EDMA mode DMA.TCQ, NCQ) the CORE driver detects the erring command, completes it with error indication (MV_COMPLETION_TYPE_ERROR), and if there are commands that are aborted by the drive or EDMA (due to the device error) the CORE driver detects those commands and re-issues them back.

The CORE driver in this case also keeps the channel ready for accepting new commands from the IAL.

This change is not backward compatible. The IAL must remove any reference to `mvEnableAutoFlush or mvDisableAutoFlush.`

5. Changed mvLog module API functions and Types

- Added new message type "MV_DEBUG_FATAL_ERROR". This message is used for tracking critical errors. This message type is added in order to separate between normal errors (Device error , connect /disconnect interrupts ...) and between errors that may occur only due to HW/SW bugs (e.g. receiving completion in the response queue with none valid tag ...)
- Changed the filterMask type to MV_U32 (instead of MV_U8). The reason for the is that the modification mentioned in the bullet above requires 9 debug levels; and since the filterMask previously used can be used to up to 8 debug levels, this modification is introduced to enhance the debug levels up to 32.
- Change the log messages debug level to be bitwise Ored. In the former implementation, the mvLog module assumed that each message belongs only to one level, but the correct behavior should be the capability of having bitwise OR between any debug level.

This change is not backward compatible