

Applied Data Sciences / VMEHSD® Software Specifications

2.1 General Information

The VMEHSD driver is installed as a standard UNIX(TM) device driver. It can handle up to eight (8) VMEHSD's configured for either HSD or IBL mode. The following system calls are supported:

OPEN

CLOSE

IOCTL

Arguments to the IOCTL call provide for reading and/or setting the device configuration and operating mode, for issuing commands to and reading status from an HSD-connected device, and for starting and controlling I/O operations using an IOCB list.

Within the following descriptions, the term configuration is taken to include the relatively static aspects of the VMEHSD configuration: such things as interrupt level, HSD/IBL mode, etc. which would normally be associated with board configuration jumpers, but which are programmable on the VMEHSD. The term (operating) mode refers to more dynamic parameters, mostly concerned with driver software functions.

2.2 Major/Minor Device Numbers

All VMEHSD's have the same major device number which may be set to any convenient value for a particular system. All VMEHSD's in a system are accessed via character special devices with the eight-bit minor device number interpreted as follows:

x u u u c c c c

--- Device number -- ----- Configuration -----

Configuration options follow on next page.

Configuration options include:

0 0 0 Use any VMEHSD configuration (HSD or IBL)

0 0 1 Require HSD configuration only

0 1 0 Use any IBL mode configuration

0 1 1 Access configuration data

1 c p Use only IBL mode with specific connector configuration (C) and priority jumper setting.

1 1 x Require normal (HSD) connector configuration.

1 0 x Require swapped (IBL) connector configuration.

1 x 0 Require Low Priority

1 x 1 Require High Priority

Note that access with a particular minor device does not change the device's configuration, but insures that the configuration is set to what the accessor expects. Configuration data is set to a default condition at system boot-up and may be set only when accessing the "configuration" device (minor device 0xU3 where 'U' is the VMEHSD unit number (0-7). Read and write access to any VMEHSD device are equivalent, and are governed by the file access permissions on the associated device file.

2.3 IBL Mode Support

The VMEHSD driver provides two methods of initiating a link for data transfers in InterBus Link (IBL) configuration. The default mode emulates that used by the ADS PCHSD device and software, using bits in the first IOCB of an IBL data transfer to indicate whether the link request should be

initiated or should be awaited and acknowledged before beginning the transfer.

The alternate method emulates the Encore H.IBLG handler and initiates the link request when the first operation is a write or waits for and acknowledges a link request when the operation is a read. This method is selected by a flag in the hsdmode structure passed to an HSDSETMOD call.

2.4 Symbol Definitions

The header file "hsdio.h" contains definitions of symbols used with the "ioctl(2)" system call to perform various HSD functions, as well as declarations for data structures used with those calls.

2.5 OPEN Function

The standard "open"(2) function must be called to associate a file descriptor with the desired device file. No specific meaning is attached to the "flag" or "mode" arguments. Only one process may open any one VMEHSD device at a time. In addition to errors which may normally be returned from the "open" call, a value of EBUSY (in errno) may be returned if the addressed device is already in use.

2.6 CLOSE Function

The standard "close"(2) function must be called to terminate access to the desired device file. Any pending operations for the open device are cancelled and the associated memory areas released when the device is closed.

2.7 IOCTL Function

The ioctl(2) call is used to perform most operations with the VMEHSD. The ioctl call requires the user to pass the file descriptor of an open device file, a function code and an argument. All ioctl calls except HSDSTRTIO and HSDTESTIO are performed synchronously. That is, the operation is completed and any status information to be returned to the caller is stored before control returns from the ioctl call.

Any of these functions may return a value of -1, with the variable `errno` set to one of the following error codes:

EFAULT If some part of a data structure passed as an argument is not accessible to the user.

EINVAL If some field in a data structure passed as an argument is invalid or inappropriate.

ENXIO If the requested operation is illogical or impossible.

EIO If some I/O error occurred on the VMEHSD.

EBUSY If active I/O requests preclude performing the current request.

EPERM If the caller does not have the requisite privilege for the requested operation.

EINTR If a signal was caught during the course of the current function call.

Each subsection below discusses one of the `ioctl` functions and describes the required argument(s). Detailed information about the various data structures may be found in section 3.0.

2.7.1 HSDGETCFG Get VMEHSD Configuration

Argument: pointer to `hsdcfg` structure

This call returns the current configuration information to the referenced `hsdconfig` structure. This call may be performed on any device, provided no I/O activity is in progress.

2.7.2 HSDSETCFG Set VMEHSD Configuration

Argument: pointer to `hsdconfig` structure

This call sets the configuration of the addressed VMEHSD according to the contents of the referenced `hsdconfig` structure. This call may be performed only on the configuration device (see section 2.2), and only when no I/O

activity is in progress. Note that the busadrs field of the hsdconfig corresponds to the VMEHSD jumper setting and may not be changed. Nor may the ivector field be changed by HSDSETCFG. Other fields may be changed, but extreme care should be used. To change configurations, use HSDGETCFG to obtain the current settings, then issue HSDSETCFG after altering the desired items.

2.7.3 HSDGETMOD Get VMEHSD Operating Mode

Argument: pointer to hsdmode structure

This call returns the current operating mode information to the referenced hsdmode structure. This call may be performed on any device, provided no I/O activity is in progress.

2.7.4 HSDSETMOD Set VMEHSD Operating Mode

Argument: pointer to hsdmode structure

This call sets the current operating mode of the addressed VMEHSD from information in the referenced hsdmode structure. This call may be performed on any device, provided no I/O activity is in progress. To change modes, use HSDGETMOD to obtain the current settings, then issue HSDSETMOD after altering the desired items. Issuing an HSDSETMOD call to the configuration device will cause the mode settings to become the defaults for all users until the system is next re-booted.

2.7.5 HSDGETID Get VMEHSD Device ID

Argument: pointer to character string at least HSDMAXID

This call gets the identification string (including firmware revision level) from the addressed device and returns it to the caller's character string. The identification is an ASCII string terminated by a null byte and is less than HSDMAXID (defined in hsdio.h) bytes long, including the null terminator.

2.7.6 HSDRESET Perform VMEHSD Reset Operations

Argument: reset mode selection

This call performs the requested reset operation(s) on the addressed VMEHSD. The argument to this function is a reset mode selection which may be defined in one of two ways:

a) Constants selecting individual reset operations:

HSD_TERM Issue "terminate device" function

HSD_MCLR Issue master clear (I/O Reset) function

HSD_RESET Issue board reset to VMEHSD

Two or more selections may be or-ed together; they will be performed in the order listed above.

b) The constant HSD_RSTCOD may be or-ed with the desired VMEHSD hardware reset code. The requested code will be issued to the board directly.

This function can be issued to any device with no currently active I/O operations.

2.7.7 HSDGETSTS Read VMEHSD Board Status

Argument: pointer to 32-bit word to receive status

This call issues a board status request to the addressed VMEHSD and stores the status in a 32-bit word addressed by the argument.

2.7.8 HSDDEVSTS Read Attached Device Status

Argument: pointer to hsdctl structure

This call issues a device status request IOCB to the addressed VMEHSD and stores the VMEHSD and device status in the addressed hsdctl structure. The iocb_dsr field of the hsdctl is used as the source of the device-dependent portion of the device status request IOCB. This function can be issued to any VMEHSD operating in an HSD configuration which has no active I/O requests.

2.7.9 HSDDEVCMD Issue Command to Attached Device

Argument: pointer to hsdctl structure

This call issues a device command, optionally followed by a device status request IOCB to the addressed VMEHSD and stores the VMEHSD and device status in the addressed hsdctl structure. The `iocb_cmd1` and `iocb_cmd2` fields of the addressed hsdctl provide the source of the device-dependent portion of the device command IOCB word. If the `SACMD` flag is set in the hsdctl, a device status request IOCB will be command chained to the device command IOCB. The `iocb_dsr` field of the hsdctl provides the device-dependent portion of the device status request IOCB. This function can be issued to any VMEHSD operating in an HSD configuration which has no active I/O requests.

2.7.10 HSDSTRTIO Initiate I/O Operation with IOCB List

Argument: pointer to hsdctl structure

This call initiates an I/O operation using an IOCB list (IOCL). The list address is contained in the referenced hsdctl, as are the various options requested for the operation. The request will be validated, user (virtual) memory addresses in logical IOCB's will be translated to real memory addresses and the referenced memory pages will be faulted in and locked down. The request will then be initiated or queued for processing if the device is currently busy. Any HSDSTRTIO call, with `NWAIT` specified, must be paired with an HSDTESTIO call to complete the driver's processing and release locked memory areas upon completion of the request. No status is stored asynchronously to the hsdctl during request processing; status will be updated only as the result of a HSDTESTIO call. This function may be issued to any VMEHSD device provided the hsdctl structure used does not have its `BUSY` flag set and has not been used for any still-active request.

Options on the HSDSTRTIO call are specified by flag bits set in the `flags` field of the hsdctl:

NWAIT Do not wait for I/O completion; return to caller immediately after queuing request.

PIOCL IOCB list is physical rather than logical. All addresses (IOCB and buffer) are taken to be real. The user is responsible for insuring that the appropriate memory areas are in fact present and locked against paging and/or swapping for the duration of the I/O operations. This operation requires superuser privilege.

NTIMO Suppress timeout checking for this request.

NOTIFY Issue signals to the caller upon a) asynchronous status posting or b) IOCL completion, if the corresponding signal number isignal or csignal in the hsdctl structure is non-zero.

EXTERN Allow the VMEHSD to function in external mode. The caller must provide the 32-bit (real) base address of the buffer to be accessed by the external device in the embase field of the hsdctl structure. If PIOCL is also set and the IOCL address contained in the hsdctl structure is zero, no operation will be initiated after the addressed VMEHSD is placed in external mode.

The BUSY bit in the flags field of the hsdctl structure will be set for the duration of the I/O request. This bit may be tested after an HSDTESTIO call to determine when a no-wait operation has completed. Additional status information is contained in the xstatus field of the hsdctl structure, further explaining the cause of an EIO, EINVAL or EFAULT return.

2.7.11 HSDTESTIO Test Status of I/O Operation

Argument: pointer to hsdctl structure

This call tests the status of an I/O operation begun by an HSDSTRTIO call. The hsdctl address passed as the argument must have previously been used with an HSDSTRTIO call. The only option flag effective with the HSDTESTIO call is NWAIT. If the NWAIT flag is set, the HSDTESTIO call will immediately return a value of zero if the request is complete or will return a value of -1 with errno set to EBUSY if the request is still active. In the case of completion, the BUSY flag will be cleared and the xstatus and hsd_stat fields of the hsdctl structure will be updated.

2.7.12 HSDHALTIO Terminate I/O Operation

Argument: pointer to hsdctl structure

This call terminates an I/O operation begun by an HSDSTRTIO call. The hsdctl address passed as the argument must have previously been used with an HSDSTRTIO call. The operation will be terminated, and status will be returned to the hsdctl structure. This operation is always performed synchronously, regardless of the state of the NWAIT flag. The only flag which is effective in the HSDHALTIO call is the NOABORT flag which will prevent aborting a transfer which may be in progress. If the HSDHALTIO call is made with the NOABORT flag set and the specified request is already complete or is an IBL-mode transfer which is awaiting a link request, the operation will be cancelled. If the operation has actually begun, the HSDHALTIO call will return -1 with errno set to EBUSY.

2.7.13 HSDPRUSTS Get Status from Previous Operation

Argument: pointer to hsdctl structure

This call returns the VMEHSD status from the previous operation in the bd_sts field of the hsdctl structure.

3.1 Introduction

This section describes the data structures used with the "ioctl" function calls to perform I/O operations with the VMEHSD. These data structures and the associated constants are defined in the file "hadio.h". Definitions for the standard HSD IOCB structure are found in the file "hsddef.h".

3.2 Reference Documents

Refer to the respective Encore MPXÄ32 Operating System Reference and Technical software manuals for a detailed description of the software requirements for the IOCB structure and the HSDII board operation. For a comprehensive hardware description of the Encore HSDII compatible card refer to

Encore HSDII Technical Manual, Document # 303-329131-000.

Encore MPX-32 Volume 2, Reference Manual, Document #323-001011-300

Refer to the following documents for assistance programming the VMEHSD card.

ADS VMEHSD Technical Manual 0900052

3.3 Device Command

The Device Command Block (DCB) is a block of eight (8) 32-bit words which provides the VMEHSD card with configuration information, a description of the operation to be performed and a place to return status information to the user. The first two 32-bit words of the DCB contain basic configuration information required for the VMEHSD to access the VMEbus and the address of the DCB. The VMEHSD is activated by writing these two 32-bit words to its command and pointer registers. Writing to the pointer register causes the VMEHSD to execute the operation specified by the contents of the command register. Therefore, the first 32-bit word of the DCB must be written to the command register before the second word is written to the pointer register.

Jumpers JP1-16 on the VMEHSD board determine the physical address of the VMEHSD command and pointer registers. The jumpers determine (1) the upper 16 bits (bits 31-16) of the address and (2) the lower 16 bits, which are FEFC (hex) for the command register or FF00 (hex) for the pointer register. Jumper JP1 corresponds to bit 31 (most significant) of the physical address; JP16 corresponds to bit 16. Jumpers are installed corresponding to ones in the desired address. The factory setting is 0100FEFC (hex) for command register and 0100FF00 (hex) for pointer register.

The required DCB structure is managed by the VMEHSD driver so the user need never be concerned with its contents. Certain fields of the DCB are returned to the caller and may be modified by the HSDGETCFG and HSDSETCFG calls. For the user concerned with this level of detail, the DCB is described in detail in the ADS VMEHSD Technical Manual, part number 0900052.

3.4 hsdconfig Structure

The hsdconfig structure is used with the HSDGETCFG and HSDSETCFG calls to retrieve or change the VMEHSD's configuration. This structure contains the following information:

busadr VMEbus address assigned to addressed VMEHSD

cfg_reg Current contents of VMEHSD configuration register. Significant bit sub-fields of this field are accessed by symbols defining appropriate masks:

HSD_MODE HSD/IBL selection: has value

HSD_MHSD if operating as HSD

HSD_MIBL if operating as IBL

HSD_CCFG IBL connector configuration: value

HSD_CHSD if configured with HSD (normal) connector.

HSD_CIBL if configured with IBL (swapped) connector.

HSD_HPRI IBL link high priority selection (set if high)

HSD_SWP Byte/word swap selection

HSD_EXISTS Bit is set in configuration if a VMEHSD at the configured address actually exists. Clearing this bit and issuing an HSDSETCFG call effectively takes the addressed device off-line.

irqlvl Configured VMEbus interrupt request level

ivector Configured VMEbus interrupt vector

brlmode VMEbus release mode selection

brqlvl VMEbus request level

am_dcb VMEbus address modifier for accessing DCB.

am_iocl VMEbus address modifier for accessing IOCB lists.

am_bfr VMEbus address modifier for accessing data buffers.

3.4.1 Address Modifiers

The VMEbus address modifier (AM) lines allow a bus master to pass additional information about the slave during a data transfer. This information may be used to partition the system to increase reliability or to enhance memory/device protection and privileged access functions. Because system requirements vary, the VMEHSD allows the user to specify the address modifier bits to be used in accessing 1) the DCB for control and status information, 2) the IOCB list and 3) the user's data buffer. The 64 possible AM codes are grouped into three categories: 1) defined by VMEbus specification, 2) defined by user, 3) reserved.

The recommended AM codes for basic I/O operations are:

Hex Value Interpretation

09 Extended non-privileged Data Access

0D Extended supervisory Data Access

0B Extended non-privileged Ascending Access

3.5 hsdmode Structure

The hsdmode structure is used with the HSDGETMOD and HSDSETMOD calls to retrieve or change the VMEHSD's operating mode. This structure contains the following information:

flags Flag bits specifying options

HM_SWW Swap 16-bit words in data transfers

HM_SWB Swap bytes in data transfer

HM_EEM "Encore Emulation Mode": Issue or acknowledge IBL link requests according to Encore H.IBLG handler protocol. The default is to operate according the ADS PCHSD protocol (using bits in the IOCB).

timeout Default timeout to be applied to any operation not having a timeout specified in the associated hsdctl structure.

3.6 Input/Output Control Block

The IOCB list is an array of structures which the calling routine must fill in. The normal HSD bit definitions are adhered to strictly. The following HSD functions, which are described by IOCB Word 1, opcode bits 00 - 07, are supported (bit 00 is most significant):

Bit 00 - HSD I/O Transfer (1 = Input).

Bit 01 - HSD Command Transfer.

Bit 02 - HSD Status Request.

Bit 04 - HSD Interrupt on End-of-Block (IEOB).

Bit 05 - HSD Transfer-In-Channel (TIC).

Bit 06 - HSD Command Chain.

Bit 07 - HSD Data Chain.

If the default method for IBL link request/acknowledgment determination is selected (PCHSD Emulation), bit 14 of the first IOCB word is used to indicate that the link request should be made before the data transfer is begun. Bit 15 is not used as the VMEHSD will always insure that a link has been established before transferring data in an IBL configuration.

The VMEHSD also supports in firmware the SOBNZ (subtract one and branch non-zero) variant of the TIC IOCB as defined by the Encore generic HSD handler, H.HSDG. For this function, the third word of a TIC IOCB

must contain two 16-bit counters. The least-significant one (bits 16-31) is decremented each time the TIC is processed and a branch to the target address is taken. When the count in bits 16-31 reaches zero, it is reset to the value in bits 0-15 and the branch is not taken, causing the IOCB at the next sequential address to be fetched and executed.

Asynchronous status posting and/or notification is controlled by the IEOB bit in the first IOCB word. If this bit is set in the caller's logical IOCB (PIOCL not set), the driver will copy the status posted by the VMEHSD in the fourth word of the physical IOCB into the fourth word of the caller's (logical) IOCB and increment the third word of the logical IOCB. If the NOTIFY flag is set the signal specified by the isignal field of the associated hsdctl, if non-zero, will be sent to the calling process.

3.7 hsdctl Structure

The hsdctl structure is used with the all ioctl calls which initiate an I/O operation or retrieve external device status. This structure contains the following information:

flags Flag bits specifying options

NWAIT No-wait: return immediately without waiting for completion.

PIOCL Physical IOCB List (user has responsibility for memory management, virtual-to-real address translation, etc.)

NTIMO Suppress timeout on this request

NOTIFY Issue signals to caller upon a) asynchronous status posting or b) operation complete, according as isignal or csignal fields are non-zero

EXTERN Enable VMEHSD for external mode operation before issuing IOCB list. If PIOCL is also set and IOCL address is zero, no operation is started.

SACMD Issue device status request after performing command on an HSDDEVSTS call.

NOABORT Do not abort an active I/O transfer on an HSDHALTIO call; return EBUSY status instead.

BUSY Operation specified by this hsdctl structure is currently active.

timeout Time limit in seconds to allow for completion of this operation. If zero, the most recent value set by an HSDSETMOD call will be used. If no value previously set by HSDSETMOD, default is 30 seconds.

xstatus Extended status information. The value returned in this field on an HSDSTRTIO, HSDTESTIO, or HSDHALTIO call gives additional information about the reason for termination. See section 4.0 for a detailed list of status codes.

isignal Signal number to be issued on asynchronous status posting. This field must be non-zero and NOTIFY flag must be set for signal to be issued.

csignal Signal number to be issued on operation completion. This field must be non-zero and NOTIFY flag must be set for signal to be issued.

iocb_dsr Low-order 24 bits of this word are placed in the first word of the device status request IOCB issued by an HSDDEVSTS or HSDDEVCMD (if SACMD flag set) call.

iocb_cmd1 Low-order 24 bits of this word are placed in the first word of the IOCB issued by an HSDDEVCMD call.

iocb_cmd2 This word is placed in the second word of the IOCB issued by an HSDDEVCMD call.

hsd_sts Most recent status stored by the VMEHSD is returned to this field at completion of any operation.

dev_sts Status returned by the external device is stored here by an HSDDEVSTS or HSDDEVCMD (with SACMD set) call.

iocl Pointer to IOCB list to be executed

embase Base VMEbus address to be used for external mode memory accesses. This must be a 32-bit absolute (real) address.