ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Applied Data Sciences VMEHSD* at our website: ***Click HERE***

VMEHSD

Software Device Driver
for Silicon Graphics IRIX Version 6.5
User Manual

Document Number: 0900115
Rev. 1.0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPLIED DATA SCIENCES, INC. has contained in this subject matter proprietary material. All manufacturing, reproduction, use, and sales rights pertaining to this material are expressly reserved. This material is submitted in confidence for a specified purpose and the user, by accepting this material, agrees that it will not be used, copied or reproduced in whole or in part without the expressed written permission of Applied Data Sciences, Inc.

APPLIED DATA SCIENCES, INC. reserves the right to make design changes or modification to any product to improve performance or incorporate new functions.

The material in this document is for informational purposes and is subject to change without notice.

APPLIED DATA SCIENCES, INC. assumes no responsibility for any errors which may appear in this document.

TO CONTACT US

Our mailing address is:

Applied Data Sciences, Inc.
P.O. Box 814209
Dallas, TX 75381-4209
USA

Our shipping address is:

Applied Data Sciences, Inc.
1300 North I-35E, Suite 100
Carrollton, TX 75006
USA

Our telephone numbers are:

972-242-7944 (USA & international)

Our facsimile number is:

972-242-8874 (USA & international)

Our email address is:

support@appdatsci.com

Copyright (c) 1998
APPLIED DATA SCIENCES, INC.
All Rights Reserved.
Printed in the U.S.A.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

REVISION STATUS

REV	PAGES CHANGED	DESCRIPTION	APPR	DATE
1.0	----	Initial Release	JWG	12/09/98

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

LIST OF RELATED DOCUMENTS

TITLE	DOCUMENT NUMBER
Encore CSD	
HSDII Technical Manual	303-000270-200
HSDII Hardware Reference Manual	301-320050-000
MPX-32 Reference Manual, Volume I	323-001011-300
Applied Data Sciences	
VMEHSD Interface Board-Technical Manual	0900052
VMEHSD IOCB Test Program -- User Manual	0900055

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TRADEMARK ACKNOWLEDGMENTS

VMEHSD is a registered trademark of Applied Data Sciences.
Silicon Graphics and IRIS are registered trademarks of Silicon Graphics, Inc.
IRIX is a trademark of Silicon Graphics, Inc.
UNIX is a registered trademark of UNIX Laboratories, Inc.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

TABLE OF CONTENTS

SECTION	TITLE	PAGE
1.0	INTRODUCTION	
1.1	VMEHSD Board Description	1-1
1.2	Encore HSDII Board Description/Emulation	1-2
1.3	Software Overview	1-2
1.3.1	Porting Considerations	1-3
1.4	Requirements	1-3
2.0	DRIVER FUNCTIONS	
2.1	General Information	2-1
2.2	Major/Minor Device Numbers	2-1
2.3	IBL Mode Support	2-2
2.4	Symbol Definitions	2-2
2.5	OPEN Function	2-3
2.6	CLOSE Function	2-3
2.7	READ Function	2-3
2.8	WRITE Function	2-4
2.9	IOCTL Function	2-5
2.9.1	HSDGETCFG [Get VMEHSD Configuration]	2-6
2.9.2	HSDSETCFG [Set VMEHSD Configuration]	2-6
2.9.3	HSDGETMOD [Get VMEHSD Operating Mode]	2-6
2.9.4	HSDSETMOD [Set VMEHSD Operating Mode]	2-6
2.9.5	HSDGETID [Get VMEHSD Device ID]	2-7
2.9.6	HSDRESET [Perform VMEHSD Reset Operations]	2-7
2.9.7	HSDGETSTS [Read VMEHSD Board Status]	2-7
2.9.8	HSDDEVSTS [Read Attached Device Status]	2-8
2.9.9	HSDDEVCMD [Issue Command to Attached Device]	2-8
2.9.10	HSDPRVSTS [Read Status of Previous Operation]	2-8
2.9.11	HSDAUXCMD [Prefix Auxiliary IOCL to Read/Write]	2-9
3.0	DATA STRUCTURES	
3.1	Introduction	3-1
3.2	Reference Documents	3-1
3.3	Device Command	3-1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

3.4 hsdconfig Structure 3-2
3.4.1 Address Modifiers 3-3
3.5 hsdmode Structure 3-3
3.6 Input/Output Control Block 3-4
3.7 hsdctl Structure 3-4

4.0 STATUS RETURNS

4.1 Status Returns 4-1
4.2 hsdctl Status Returns 4-1
4.3 VMEHSD Board Status Returns 4-2

5.0 PROGRAM INSTALLATION

5.1 Distribution Media 5-1
5.2 Installation 5-1
5.2.1 Customizing the SYSTEM File 5-2
5.2.2 Customizing the MASTER File 5-3
5.2.3 Completing the Installation 5-3

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDICES

APPENDIX	TITLE	PAGE
A	Example Programs	A-1
B	HSDBUG Terminal Operation	B-1
C	SETHSD Utility	C-1

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION 1.0
INTRODUCTION

1.1 VMEHSD Board Description

The VMEHSD(TM) board provides a high speed, bi-directional link for transferring control, status, and data between a VMEbus System and the Encore High Speed Digital Interface (HSDII) or any 32-bit external device using the Encore High Speed Digital Interface (HSDII) protocol. In the IBL mode the VMEHSD board can transfer data only to another VMEHSD board or to an Encore Computer HSDII board configured for the IBL mode. This board will also emulate the HSD's External Mode, providing a "dumb" device with random read/write capabilities.

The VMEHSD is a 6U VME board residing in a VME chassis. The board has bus master (slot 0) capability. There are two (2) 50 pin IDC male headers on the front edge of the VMEHSD board which provide connections to the external device. Connection between the VMEHSD and the HSDII compatible device is via two (2) 50 pin flat ribbon cables or equivalent. All power for the VMEHSD board is supplied by the internal VME power supply.

This high-speed data link provides a 16/32 bit data bus transfer between the VME memory and the VMEHSD board. The data may then be word and/or byte swapped for transfer between the VMEHSD board and/or HSDII compatible external device. The relationship between the most significant and least significant bits for the VME and the HSD words is pictorially shown in the following sketch.

******** One VME 32-Bit Word *********

(MSB)		(LSB)
VMED31	- -	VMED00

******** is equivalent to ********

(MSB)		(LSB)
HSD00	- -	HSD31

******** One HSD 32 Bit Word ********

For additional technical specifications of this board refer to the VMEHSD Technical Manual, Document number 0900052.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

1.2 Encore HSDII Board Description/Emulation

This information is presented to help you understand the purpose of the VMEHSD board -- that is to emulate the functions of the Encore HSDII board. The HSDII board is a high-speed, general purpose interface board which is installed in a Encore computer chassis. This board is manufactured by Encore, Inc. Computer Systems Division, Fort Lauderdale, Florida.

The Encore HSDII board provides a full 32-bit parallel interface to an external device. The VMEHSD also provides a 32-bit parallel interface to an external device. A detailed description of the HSDII board is found in the HSDII Technical Manual and HSDII Hardware Reference Manual. The document part numbers of these two manuals are contained in the index under the section LIST OF RELATED DOCUMENTS.

The VME host computer board ("host") initiates all commands via software I/O operations. Once the host has initiated the respective operation (assuming it is a data transfer), the external device executes the data transfers between the VMEHSD's onboard FIFO and the external device independent of the host's operation. The handshake seen by the external device appears functionally identical to the Encore HSDII card.

1.3 Software Overview

This document provides all the information necessary to integrate the user's software with the firmware resident on the VMEHSD.

A VMEHSD Device Command consists of eight (8) 32-bit words which make up a Device Command Block (DCB). Basic board configuration (i.e., data setup and command) to the VMEHSD is contained in the first word of the DCB. The second word is the 32-bit VME address of the Device Command Block (DCB) itself. The remainder of the DCB contains additional information describing the VMEHSD operation to be performed and provides a place for the VMEHSD to return status information.

The VMEHSD board is controlled by writing data to onboard registers. The first two words of the DCB are written to its Command and Pointer Registers, respectively. The VMEHSD is activated when a host writes to its Pointer Register, which causes it to fetch any other required information and execute the requested function. At completion of the requested operation, the VMEHSD may store status information in the DCB.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The VMEHSD driver, "vhsd", is intended for operation on Silicon Graphics Onyx and Challenge systems running IRIX(TM) version 6.5. *It will not work on Onyx2 or Origin 200 systems.* The driver handles all communication with the VMEHSD on behalf of the calling program. It allows for reading and setting the configuration and operating mode of the VMEHSD. It provides synchronous (wait mode) data transfers via read() and write() calls.

Details of the driver installation are in section 5.0.

1.3.1 Porting Considerations

The version of vhsd for IRIX 6.5 is distinct from, but functionally equivalent to vhsd for IRIX 5.x, which is Part Number 0950075. All the same functions are supported with the following changes:

1. Asynchronous I/O operations are now supported in IRIX, so the AIO library is not supplied.

2. Declarations contained in data structures used with **ioctl(2)** calls have been changed as required to maintain the sizes of all elements when using 32- or 64-bit execution models. In other words, elements which were 32 bits in the 5.3 version are still 32 bits, *except* that the IOCL pointer in the *hsdctl* structure used with the HSDAUXCMD function may be 32 or 64 bits, as required by the execution model.

3. Similarly, the declaration of the IOCB as a structure (in file **hsddef.h**) has been changed to accommodate 32- and 64-bit programs. The data buffer address, usually contained in the second 32-bit IOCB word may now be a 64-bit pointer if using the 64-bit model. All programs constructing IOCB lists for the HSDAUXCMD function should use the element *w2a*, which is a model-specific pointer to the data buffer, instead of *w2.a*, which is a 32-bit pointer in the second IOCB word. The latter element has been removed from the definition of the IOCB structure. Also, programs compiled to the 64-bit model should not clear the third word of IOCB's constructed in the fashion, as the third IOCB word holds the other half of the 64-bit data pointer in this case.

1.4 Requirements

Software: IRIX version 6.5, including "C" compiler.

Hardware: Silicon Graphics Challenge or Onyx System with VMEbus adapter and DMA mapping hardware.
VMEHSD Card.

Knowledge of the Encore HSDII IOCB structure will aid in using the VMEHSD board and software.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION 2.0
DRIVER FUNCTIONS

2.1 General Information

The VMEHSD driver is installed as a standard IRIX(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
READ
WRITE
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 identifying one of the eight possible VMEHSD's and possibly dictating its operating configuration. Access to a device with a particular minor device number does not change its configuration, but merely insures that the configuration is set as the user requires. For example, attempting to open a device whose minor device number requires IBL configuration while the device is set to HSD configuration will cause the open to fail with the return "no such device". The "configuration" device is used only to change the configuration of the VMEHSD and may not be addressed for any I/O operation.

The minor device number is encoded as follows:

x	u	u	u	c	c	c	c

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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

CONFIGURATION:

Configuration options include:

0 0 0 0	ANY configuration (HSD or IBL)
0 0 0 1	HSD configuration only
0 0 1 0	IBL only (normal or swapped)
0 0 1 1	CONFIGURATION device
0 1 x x	IBL mode with specific connector configuration and priority jumper setting.
0 1 1 x	IBL, normal (HSD) connector configuration.
0 1 0 x	IBL, swapped (IBL) connector configuration.
0 1 x 0	IBL, Low Priority
0 1 x 1	IBL, High Priority

DEVICE NUMBER:

Configuration data is set to a default condition at system boot-up and may be changed only when accessing the "configuration" device (minor device where '$U3_{16}$' 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 the InterBus Link (IBL) configuration. The default mode does not initiate any link request, but expects to wait for, and acknowledge a link request before each data transfer. A flag in the hsdmode structure passed to an HSDSETMOD call may be set to indicate that the VMEHSD should initiate the link request for all transfers.

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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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 canceled and the associated memory areas released when the device is closed.

2.7 READ Function

The standard read(2) function is used to transfer data from the VMEHSD to the user's buffer. All data transfers are performed directly to the user's buffer. Therefore, the user's buffer address must be aligned on a 32-bit boundary and the transfer byte count must be a multiple of four. The maximum length of a single transfer is 262,140 bytes (65535 or hex FFFF 32-bit words). This is the maximum that can be transferred by a single IOCB. If the requested transfer size exceeds the maximum, the driver will return an error (EINVAL). The xstatus value, which may be retrieved via an HSDPRVSTS call, will be EXLONG (transfer count too long).

Options on the read call are specified by flag bits set in the flags field of the hsdmode structure passed to an ioctl (HSDSETMOD) call:

HM_CBR — Issue Device Command before reading. A Device Command IOCB is command chained before the list performing the requested data transfer. The device-dependent portion of the command is taken from the *cmd_rd* field of the *hsdmode* structure last used in an ioctl (HSDSETMOD) call. See Section 3.5 for information on how to specify the desired device command.

HM_SAR — Request Device Status after reading. A Device Status Request IOCB is command chained to the end of the list performing the requested data transfer. The ioctl (HSDPRVSTS) call may be used to retrieve the status returned by the device.

The HM_CBR and HM_SAR options are valid only for a VMEHSD operating in an HSD configuration. The HSDAUXCMD call may be used to specify an arbitrary IOCB list to be chained preceding the data transfer IOCL (see Section 2.9.14). The HSDAUXCMD option takes precedence over the HM_CBW flag if both are given.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The returned value will be the number of bytes actually received from the external device or will be -1 in case of an error. In addition to the errors which may normally be returned from the "**read**" call, the following additional conditions may be reported by *errno* values:

EINVAL	Requested transfer length is not positive, is greater than the maximum allowed or is not a multiple of four bytes (1 longword).
EIO	An error occurred in the data transfer.
ENOMEM	A DMA mapping failure occurred, preventing the transfer from taking place. This error should not happen.

Further explanation may be obtained by issuing the ioctl (HSDPRVSTS) call and examining the returned *xstatus* field. (See Section 2.9.13.)

2.8 WRITE Function

The standard **write**(2) function is used to transfer data from the user's buffer to the VMEHSD. All data transfers are performed directly from the user's buffer and the **write** call will not return until the data transfer to the external device is complete. The user's buffer address must be aligned on a 32-bit boundary and the transfer byte count must be a multiple of four. The maximum length of a single transfer is 262,140 bytes (65535 or hex FFFF 32-bit words). This is the maximum that can be transferred by a single IOCB. If the requested transfer size exceeds the maximum, the driver will return an error (EINVAL). The *xstatus* value, which may be retrieved via an HSDPRVSTS call, will be EXLONG (transfer count too long).

Options on the **write** call are specified by flag bits set in the *flags* field of the *hsdmode* structure passed to an ioctl (HSDSETMOD) call:

HM_CBW	Issue Device Command before writing. A Device Command IOCB is command chained before the list performing the requested data transfer. The device-dependent portion of the command is taken from the *cmd_wt* field of the *hsdmode* structure last used in an ioctl (HSDSETMOD) call. See Section 3.5 for information on how to specify the desired device command.
HM_SAW	Request Device Status after writing. A Device Status Request IOCB is command chained to the end of the list performing the requested data transfer. The ioctl (HSDPRVSTS) call may be used to retrieve the status returned by the device.

The HM_CBW and HM_SAW options are valid only for a VMEHSD operating in an HSD configuration. The HSDAUXCMD call may be used to specify an arbitrary IOCB list to be chained preceding the data transfer IOCL (see Section 2.9.14). The HSDAUXCMD option takes precedence over the HM_CBW flag if both are given.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

The returned value will be the number of bytes actually transferred to the external device, or -1 in case an error occurred. In addition to the errors which may normally be returned from the "**write**" call, the following additional conditions may be reported by *errno* values:

EINVAL	Requested transfer length is not positive, is greater than the maximum allowable or is not a multiple of four bytes (1 longword).
EIO	An error occurred in the data transfer.
ENOMEM	A DMA mapping failure occurred, preventing the transfer from taking place. This error should not happen.

Further explanation may be obtained by issuing the ioctl (HSDPRVSTS) call and examining the returned *xstatus* field. (See Section 2.9.13.)

2.9 IOCTL Function

The **ioctl**(2) call is used to perform all VMEHSD operations other than simple data transfers. 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 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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.9.1 HSDGETCFG [Get VMEHSD Configuration]

Argument: pointer to *hsdconfig* 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.

2.9.2 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.

2.9.3 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.

2.9.4 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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.9.5 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 not more than HSDMAXID (defined in **hsdio.h**) bytes long, including the null terminator.

2.9.6 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 three 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.

c) The constant HSD_HALT for the reset mode will result in termination of any currently active I/O operation. The driver in effect, simulates an immediate time-out and terminates the operation appropriately.

This function can be issued to any device. Unless the argument is HSD_HALT, there must be no currently active I/O operation on the device.

2.9.7 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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

2.9.8 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 which is operating as an HSD and which has no active I/O requests.

2.9.9 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 device-dependent portion of the device command IOCB. 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 as an HSD and which has no active I/O requests.

2.9.10 HSDPRVSTS [Read Status of Previous Operation]

Argument: pointer to *hsdctl* structure

This call returns status from the last operation performed on the addressed device to the referenced *hsdctl* structure. The most recent board and external device status are stored in fields *hsd_sts* and *dev_sts*, respectively. The extended status code from the last operation is stored in the *xstatus* field and the value returned to the system *errno* variable from the last **ioctl** call is stored in the *perrno* field. This call is useful in obtaining more detailed information about a failed data transfer operation. It may be issued to any device at any time. The return value from this function is always zero, unless problems are encountered in accessing the user's *hsdctl* structure, in which case -1 will be returned and *errno* will be set to EFAULT.

2.9.11 HSDAUXCMD [Prefix Auxiliary IOCL to Read/Write]

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Argument: pointer to *hsdctl* structure

This call allows the user to specify auxiliary device commands, status requests, and data outputs to be prefixed to the next **read** or **write** call. It also allows the user to specify a User Device-Dependent command for the Read or Write IOCB. The *iocl* field of the referenced *hsdctl* should point to the desired IOCB list (IOCL). The IOCL may contain Device Command, Device Status Request or Write Data IOCB's, only. The IOCL and any data to be written to the device, are moved to a buffer internal to the driver. At the next **read** or **write** call, the list copied from the HSDAUXCMD call will be prefixed to the IOCL required for the data transfer. When the I/O operation is complete, status stored by the VMEHSD in the driver's copy of the auxiliary IOCL will be posted back to the user's IOCL before return from the **read** or **write** is made. An HSDAUXCMD call with the *iocl* address set to 0 will cancel any pending HSDAUXCMD. This call may be issued to any device with no currently active I/O operation.

On any HSDAUXCMD call (regardless of whether the *iocl* address is 0) the contents of the *uddcmd* field of the referenced *hsdctl* will be saved. The saved value will be inserted into the User Device-Dependent Command (UDDCMD) field of the IOCB's used for the next READ or WRITE call. The UDDCMD field occupies bits 08-15 of the first word of a data transfer IOCB.

In addition to the errors returned by other **ioctl** functions, HSDAUXCMD may return an *errno* value of EINVAL with the *hsdctl xstatus* field set to one of the following:

EXAUX	an invalid IOCB (not Device Command, Status Request or Write) is in the list.
EXLONG	The IOCB list and all write data would not fit in the auxiliary kernel buffer defined for the device.

Note: The Device Command and Status Request IOCB's are legal in an HSDAUXCMD list, but are invalid for a VMEHSD operating in IBL configuration; it is the user's responsibility to insure this restriction is met.

The size of the driver's auxiliary buffer is determined at the time the kernel is built. See Section 5.2.2 for details of configuring the auxiliary buffer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION 3.0
DATA STRUCTURES

3.1 Introduction

This section describes the data structures used with the **ioctl**(2) function calls to perform I/O operations with the VMEHSD. These data structures and the associated constants are defined in the file **hsdio.h**. Definitions for the standard HSD IOCB structure are found in the file **hsddef.h**.

3.2 Reference Documents

Refer to the respective Encore MPX-32 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 in programming the VMEHSD card:

ADS VMEHSD Technical Manual, Document # 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.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Jumpers JP1-16 on the VMEHSD board determine the physical address of the VMEHSD command and pointer registers. The jumpers determine the upper 16 bits (bits 31-16) of the addresses. The lower 16 bits of the addresses are fixed at FEFC (hex) for the command register and 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 0100 (hex), giving 0100FEFC (hex) for command register and 0100FF00 (hex) for pointer register.

The required DCB structure is managed by the VMEHSD driver so the user never needs to 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, document 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:

busadrs — 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 has 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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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_CBR	Issue device command before read
HM_CBW	Issue device command before write
HM_SAR	Request device status after read
HM_SAW	Request device status after write
HM_LIN	Initiate IBL link request on each data transfer
HM_EEM	"Encore Emulation Mode": Issue or acknowledge IBL link requests according to Encore H.IBLG handler protocol.

IBL link operations are summarized in the following table:

Operation	HM_LIN	HM_EEM	Link
Any	0	0	Ack
Any	1	x	Issue
Write	0	1	Issue
Read	0	1	Ack

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

timeout — Default timeout to be applied to any operation not using an *hsdctl* structure or not having a timeout specified in the associated *hsdctl* structure.

cmd_rd — Two-longword array containing the device-dependent portion of the device command used if HM_CBR is set. The lower 24 bits of the first word and the whole second word are placed in a device command IOCB which is command chained to the required data transfer IOCBs.

cmd_wt — Two-longword array containing the device-dependent portion of the device command used if HM_CBW is set. The lower 24 bits of the first word and the whole second word are placed in a device command IOCB which is command chained to the required data transfer IOCBs.

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 and all bits are Hi True):

Bit 00 - HSD I/O Transfer (1 = Input & 0 = Output).
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.

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.

3.7 *hsdctl* Structure

The *hsdctl* structure is used with 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:

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

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

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 HSDPRVSTS call gives additional information about the reason for termination. See Section 4.0 for a detailed list of status codes.
perrno	Previous value of system variable *errno*, returned on HSDPRVSTS call only.
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.

*Note: The header file **hsdio.h** defines additional fields in the **hsdctl** structure for compatibility with previous versions of the driver. Only those elements listed here are valid for use with the IRIX 6.5 driver.*

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION 4.0
STATUS RETURNS

4.1 Status Returns

Status returns from the system calls used to access the VMEHSD driver are consistent with other UNIX device drivers. In addition, status information may be returned to the *hsdctl* structure associated with the call, and may be stored in the IOCB list used.

In case of an error, the system function called will return -1 and the variable *errno* will be set to an error code. Some status returns are specific to the VMEHSD driver and may differ from the meaning associated with the same return from another device.

EPERM Caller is attempting an HSDSETCFG function on a non-configuration device (see Section 2.2).

EBUSY May be returned on an OPEN call if the addressed device is already in use. Also returned by **astat** when an **aread** or **awrite** call is still in progress.

4.2 *hsdctl* Status Returns

The *xstatus* field of the *hsdctl* structure contains an expanded error code in the case of an EINVAL, EFAULT, or EIO return. The expanded values are as defined in **hsdio.h**.

Expansion of EFAULT code:

EXIOLA Unable to access some part of IOCB list.

EXBUFA Caller unable or unauthorized to access some part of a data buffer addressed by the IOCB list.

Expansion of EINVAL code:

EXBUFB IOCB specifies buffer address not on a 32-bit boundary.

EXCNT IOCB specifies I/O transfer count of 0.

EXCHN Chaining error on IOCB (DC and CC).

EXCMD Command or Status request IOCB is in error.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

EXLONG If operation is **read** or **write**: data transfer request is too long. If operation is ioctl (HSDAUXCMD): list and data will not fit in device's auxiliary kernel buffer.

EXAUX Invalid IOCB type (not Device Command, Status Request, or Write Data transfer) found in the list passed to a HSDAUXCMD call.

Expansion of EIO code

EXTIMO Request not completed before timeout occurred.

EXKILL Request aborted by HSDHALTIO or HSDRESET, HSD_RESET call.

EXIOER Actual VMEHSD I/O error (see *hsd_stat* field for bad status value).

4.3 VMEHSD Board Status Returns

The *hsd_sts* field of the *hsdctl* structure contains the status returned by the addressed VMEHSD on the last operation. This status is also stored in the fourth word of any IOCB other than a TIC or Device Status Request.

Format of the VMEHSD status word is:

Bit	31	Current IOCB specifies a read operation
Bit	30	Current IOCB specifies command transfer
Bit	29	Current IOCB specifies status request
Bit	28	Current IOCB specifies data chaining
Bit	27	Current IOCB specifies command chaining
Bit	26	Link request was received (IBL mode only)
Bit	25	External terminate received from device
Bit	24	Device End-of-Block signal received
Bit	23	Operation complete
Bit	22	External device present
Bit	21	Output FIFO empty
Bit	20	Output FIFO half full
Bit	19	Input FIFO empty
Bit	18	Input FIFO half full
Bit	17	Not external mode
Bit	16	Transfer counter non-zero

Bits 15-0 Residual count. Number of 32-bit words requested but not transferred.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

SECTION 5.0
PROGRAM INSTALLATION

5.1 Distribution Media

The VMEHSD device driver is distributed on 4mm DAT tape in **tar** format and contains the following components:

Device driver code:	Part Number 0950111
SETHSD Utility:	Part Number 0950045

5.2 Installation

Copy the installation files from the distribution tape:

tar xvf /dev/tape

The following files should be placed in a convenient directory for compiling the driver:

hsd_dt.h	VMEHSD device table structure declarations
vhsd.c	VMEHSD driver source
vhsd.master	Description of VMEHSD driver for IRIX
vhsd.sm	File describing VMEHSD hardware
Makefile.vhsd	Instructions for building driver, sethsd
bld.kern	Shell script to compile driver and build new IRIX kernel
mkhsdfiles	Shell script for creating HSD device nodes in **/dev** directory

The following files should be placed where they are accessible to programs compiled to use the VMEHSD driver.

hsddef.h	Structure declarations for IOCB contents
hsdio.h	Structure and symbol definitions for users of the VMEHSD driver

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.1 Customizing the SYSTEM File

Edit the file **vhsd.sm** to describe the target configuration. Each VMEHSD is described by a VECTOR directive such as the following. The entire directive is a single line although it is broken up here for illustration.

```
VECTOR bustype=VME module=vhsd ipl=4 ctlr=0 adapter=0
        iospace=(A32S,0x20000000,0x10000)
        exprobe_space(r,A32S,0x2000FF04,4,0x564D4548,0xFFFFFFFF)
```

The following items may be changed in each VECTOR line:

ipl VMEbus Interrupt Priority Level for the board (may be a value from 1 through 7). Interrupt vectors will be allocated by the driver when the system is booted.

ctlr VMEHSD "unit" number. These should be assigned sequentially, starting from 0, and may not exceed 7.

adapter The number of the VMEbus adapter to which this VMEHSD is attached. Should be zero except for systems with more than one VMEbus adapter.

iospace The second argument specifies the VMEbus address of the device and must match the VMEHSD address jumper setting (JP1) exactly. The other arguments must not be changed.

exprobe The third argument specifies the VMEHSD register to probe to see if the board is installed. This value must be equal to the bus address specified by **iospace** plus (hex) FF04. The other **exprobe** arguments must not be changed.

Note that when a probe address is given, *lboot* will include the VMEHSD driver and tables only for devices that are physically present. To include the driver in the kernel unconditionally, delete the part of the VECTOR directive(s), from **exprobe** to the end of the line. Refer to the comments in the file **/var/sysgen/system/irix.sm** to select appropriate VMEbus addresses which are reserved for customer devices.

A typical configuration for a VMEHSD on an IP19 processor is:

VMEbus address (jumper):	20000000_{16}
Base (iospace) address:	20000000_{16}
Interrupt Priority Level:	4
Bus Request Level:	3

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

5.2.2 Customizing the MASTER File

The **vhsd.master** file may be edited to set the major device number used for VMEHSD devices and the sizes of buffers allocated for HSDAUXCMD operations. The major device number is given in the "SOFT" field of the first non-comment line of the file. It should be chosen from the range of values reserved for customer devices so as not to conflict with other devices in the system. If this number is changed, the same value must be given to the **mkhsdfiles** script to create the VMEHSD device files.

The symbols SIZE_0 through SIZE_7, defined in this file, specify the length, in (32-bit) longwords, of the auxiliary buffers used for VMEHSD units 0 through 7, respectively. The default value is 1024 longwords (4096 bytes). An auxiliary buffer must be large enough to accommodate the user's auxiliary list (4 longwords per IOCB) as well as all data to be written with the list. However, this buffer is allocated by the driver at boot time and reduces total available memory, so its size should be kept as small as practical. Auxiliary buffers are not defined for devices which are not configured. DMA mapping hardware for the auxiliary buffer, as well as for the largest permissible user buffer are allocated when the device is opened and freed when it is closed.

To alter the size of an auxiliary buffer, set the corresponding SIZE_**n** symbol to the desired length, in longwords. To suppress use of the auxiliary buffer, set the length to 0. Note that if no auxiliary buffer is defined for a particular VMEHSD, an HSDAUXCMD call addressed to that unit will always return EINVAL, with *xstatus* set to EXLONG (IOCB list too long for auxiliary buffer).

5.2.3 Completing the Installation

Compile the driver by invoking the **bld.kern** script or by performing the following steps manually:

1. Determine the processor model (IP5, IP12, IP19, etc.) being used. The **hinv** command may be used to determine this value.

2. Set the environment variable CPUBOARD to the appropriate processor model number. For example:

   setenv CPUBOARD IP19

3. **smake -f Makefile.vhsd vhsd.o**

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

To install vhsd, copy the driver object file and the configuration files to the proper **var/sysgen** directories and invoke **autoconfig**. For example, execute the following commands while logged in as **root**:

```
cp      vhsd.o          /var/sysgen/boot
cp      vhsd.sm         /var/sysgen/system
cp      vhsd.master     /var/sysgen/master.d/vhsd
cp      /unix           /unix.orig          (save running kernel)
/etc/autoconfig -f                          (generate /unix.install)
/etc/reboot
```

Invoking the **mkhsdfiles** script will create device files for the requested number of VMEHSD's. The appropriate command is:

mkhsdfiles num_devices [major_number]

The first argument specifies the number of VMEHSD's for which device files are to be created. The second, optional parameter gives the major device number to be used. The default major device number is 60, selected from the range (60-79) of major device numbers reserved for customer use (see file **/usr/include/sys/major.h**). If the major number is changed, file **vhsd.master** must be changed correspondingly. The VMEHSD device file names and corresponding minor device numbers are as follows:

File	Minor Device (hex)	Description
/dev/hsd0	00	Generic HSD unit 0
/dev/hsd0h	01	Unit 0, HSD configuration
/dev/hsd0i	02	Unit 0, IBL configuration
/dev/hsd0c	03	Unit 0, configuration device
/dev/hsd1	10	Generic HSD, unit 1
/dev/hsd1h	11	Unit 1, HSD configuration
/dev/hsd1i	12	Unit 1, IBL configuration
/dev/hsd1c	13	Unit 1, configuration device
... etc ...		

Note: User-level programs calling the VMEHSD driver should be aware of possible interaction with the CPU data cache. Cache operations are disabled on VMEHSD I/O buffer areas (as indicted by IOCB address and count). To avoid

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

unwanted interference with other data areas, these buffers should be allocated at addresses which are aligned to the processor's secondary cache line size. This size may be between 4 and 32 words, depending on the processor in use.

The **sethsd** utility may be built by invoking the command **make -f sethsd.mk** in the vhsd installation directory. This program may be used to examine a VMEHSD's configuration or to alter it from its default (e.g. at boot time). A description of **sethsd** is found in appendix C.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX A
EXAMPLE PROGRAMS

The following is a tested IBL example and is provided as an aid for software development purposes only. This example is for a VMEHSD board connected to a Encore HSDII board or other compatible interface (i.e, an IBL mode configuration). The VMEHSD program will read data and then write data back to the Encore computer, the Encore is the Master (initiator) controller.

```
/*********************************************************************/
/*                                                                   */
/*   filename:            example.c                                  */
/*   date:          Feb 9, 1993                                      */
/*                                                                   */
/*   IBL Read/Write Example Program for VMEHSD with SGI IRIX Driver   */
/*                                                                   */
/*   Compilation command is   "cc example.c -o example"              */
/*                                                                   */
/*   This program is a tested example of IBL mode data transmission  */
/*   between a VMEHSD and another HSD device.  The SGI IRIX driver,  */
/*   vhsd, is used to control the VMEHSD.                            */
/*                                                                   */
/*   Test Environment     VMEHSD card, SGI 4D/35 system, IRIX 4.0.1  */
/*            VMEHSD driver ver 1.4                                  */
/*                                                                   */
/*            Applied Data Sciences VMEHSD, Motorola MVME167       */
/*            host, Motorola UNIX System V/68 R3V7                   */
/*            VMEHSD IOCB test program, v 1.3                        */
/*                                                                   */
/*            IOCB lists used:                                       */
/*            #1:    00024000 00000001 00000000 00000000             */
/*                   (Buffer 1 contains data to write)           */
/*            #2:    80024000 00000002 00000000 00000000             */
/*                   (Buffer 2 receives data from this pgm -         */
/*                   incrementing longword pattern)                  */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/fcntl.h>
#include "hsddef.h"
#include "hsdio.h"

void   xerror();                /* forward declaration     */

long   input_darray[16384];
long   output_darray[16384];

main()
{

    /*  declare variables  */
    int                 fd,          /* File descriptor for HSD */
                        status,      /* status return from I/O calls  */
                        i,     /* general indices         */
                        j;

/*     hsdctl and hsdmode structures are defined in file "hsdio.h" */

    struct hsdctl   hsdctl;
    struct hsdmode  hsdmode;

    for(j=0; j<1600; j++)              /*  fill output data array  */
       output_darray[j] = j;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    printf("Opening the HSD channel...\n");

    /*  open HSD channel; make sure unit is in IBL mode by requesting hsd0i
     *  We will use the following default options:
     *       Always wait for IBL link to be initiated by other end
     *  We will set the following options with the HSDSETMOD call:
     *       500 second timeout waiting for I/O to complete
     */

    fd = open("/dev/hsd0i", O_RDWR);

    /*  check open error status  */

    if ( fd < 0 )  {
       perror("Error opening /dev/hsd0i");
       fprintf(stderr, "BAILING OUT DUE TO OPEN ERROR\n");
       exit(1);
    }

    printf("Setting timeout value with HSDSETMOD\n");

    hsdmode.timeout = 500;
    status = ioctl(fd, HSDSETMOD, &hsdmode);
    if ( status < 0 )  {
       perror("Error on ioctl(HSDSETMOD) call");
       close(fd);
       exit(1);
    }

    printf("Resetting VMEHSD hardware\n");
    status = ioctl(fd, HSDRESET, HSD_RESET);
    if ( status < 0 )  {
       perror("Error on ioctl(HSDRESET) call");
       ioctl(fd, HSDPRVSTS, &hsdctl);
       printf ("Board status returned: %08x\n", hsdctl.hsd_sts);
       close(fd);
       exit(1);
    }


    do  {                  /*  this is the main outer loop   */

/*     start read transfer                                     */

       printf("Start read....\n");

       status = read (fd, input_darray, 65536);

/*     Get additional status info to `hsdctl'                  */

       ioctl(fd, HSDPRVSTS, &hsdctl);

/*     Check status from Read                                  */

       if (status < 0)  {
           perror ("Error on READ");
           xerror (hsdctl.xstatus);
       } else
           printf ("%d words read\n", status/4);

/*     print board status returned from PRVSTS call            */

        printf("FCB HSD status: %08X\n",hsdctl.hsd_sts);


/*     start write transfer                                     */

       printf("Prepare to write...\n\n");
       status = write (fd, output_darray, 65536);
        printf("Write complete!\n");

/*     Get additional status info to `hsdctl'                  */

       ioctl(fd, HSDPRVSTS, &hsdctl);


/*     Check status from Write          */

       if (status < 0)  {
           perror ("Error on WRITE");
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
            xerror (hsdctl.xstatus);
        } else
            printf ("%d words written\n", status/4);

/*      print board status returned from PRVSTS call          */

         printf("FCB HSD status: %08X\n",hsdctl.hsd_sts);

/*      print out read data  */

         for(j=0; j<40; j+=8)  {
            printf ("%.4d  ", j);
            for (i=j; i < j+8 ; i++)
               printf("%.8X ",input_darray[i]);
            printf ("\n");
        }
        printf ("\n");


         printf("%.8X\r",output_darray[0]);
         output_darray[0] = output_darray[0] + 1;

        printf ("...Run again ? [Y/N]");

      } while ( toupper(getchar()) == 'Y');

}  /*  end of program  */
/*      Function  xerror
 *
 *      Print descriptive message corresponding to extended status
 *      value returned in hsdctl.xstatus field.  Note that the meaning
 *      of the code returned depends on the primary system error code
 *      (found in `errno').
 */
void          xerror (xerr)
int           xerr;               /* extended error code from hsdctl */
{
    char      *msg;               /* Pointer to specific error msg   */

    switch (errno)  {                   /* determine system error code    */

    case EFAULT:                  /* Memory access errors          */
       switch (xerr)  {

       case EXIOLA:
           msg = "Unable to access IOCB list";
           break;
       case EXBUFA:
           msg = "Unable to access I/O buffer";
           break;
       }
       break;

    case EINVAL:                  /* Invalid operation errors      */
       switch (xerr)  {

       case EXIOLB:
           msg = "IOCB not longword aligned";
           break;
       case EXBUFB:
           msg = "Buffer not longword aligned";
           break;
       case EXCNT:
           msg = "I/O transfer count = 0";
           break;
       case EXCHN:
           msg = "Chaining error on Read/Write IOCB";
           break;
       case EXCMD:
           msg = "Error on Command or Status IOCB";
           break;
       case EXLONG:
           msg = "Too many physical IOCB's required for transfer";
           break;
       case EXAUX:
           msg = "Auxiliary list/data too long for buffer";
           break;
       }
       break;
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
    case EIO:                       /* I/O operation errors          */
       switch (xerr)  {

       case EXTIMO:
           msg = "Operation timed out";
           break;
       case EXKILL:
           msg = "Operation killed by user (HSDHALTIO)";
           break;
       case EXIOER:
           msg = "Error on I/O transfer";
           break;
       }
       break;
    }

    fprintf (stderr, "Extended error code %d, (%s)\n", xerr, msg);
}
```

The following is an example of the use of the HSDAUXCMD function and is provided as an aid for software development purposes only. This example is derived from the program used to test the original VMEHSD driver for IRIX 4.0.1, using a mass memory box as described in the comments contained in the listing.

```
/*****************************************************************
 *      auxexamp  --    Example read  & write with AUXCMD        *
 *****************************************************************
 *                                                               *
 *      This routine is derived from the program used to test the*
 *      HSDAUXCMD function in the VMEHSD driver ver 1.4 for IRIX.*
 *      Code similar to this was run against a mass memory box   *
 *      connected via an HSD interface to the VMEHSD.            *
 *                                                               *
 *      Operations:                                              *
 *                                                               *
 *      1.  The AUXCMD function is used with a WRITE call to issue*
 *          command IOCB's to the device before the data is written.*
 *      2.  The AUXCMD function is used with a READ call to issue*
 *          command IOCB's to the device and also to over-write the*
 *          previous data with a new pattern.                    *
 *      3.  The data read back is verified to see that it has changed.*
 *                                                               *
 *      Note:  Most error checking has been omitted to clarify the*
 *             program logic.                                    *
 *                                                               *
 *      Modification for IRIX 6.5 (allows program to compile and run*
 *      under either 32-bit or 64-bit execution model:           *
 *                                                               *
 *          In AUXCMD list used for read() call,                 *
 *          change  auxlist[2].w2.a =(char)auxbfr;               *
 *          to      auxlist[2].w2a = (char)auxbfr; (use generic pointer)*
 *                                                               *
 *          delete the following line:                           *
 *                  auxlist[2].w3.w =  0;                        *
 *                                                               *
 *****************************************************************/

#include    <stdio.h>               /* standard I/O functions        */
#include    <errno.h>               /* error number definitions      */
#include    <fcntl.h>               /* file operation definitions    */
#include    <ctype.h>               /* character conversion macros   */
#include    "hsddef.h"              /* IOCB structure definition     */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
#include    "hsdio.h"               /* FCB structure, ioctl functions   */

int                 fd;             /* file descriptor                  */
struct hsdctl       hsdctl;         /* hsdctl structure for ioctl's     */
struct hsdmode      hsdmode;

#define NBUF        0xC000          /* Buffer size in (32-bit) words    */
#define NBUFB       NBUF*4          /* Buffer size in bytes (for rd/wt) */

long        auxbfr[50];
IOCB        auxlist[15];            /* auxiliary IOCB list              */


main()
{
    int     errno_save,             /* save errno on read/write calls   */
            i,                      /* general index                    */
            sts;                    /* save call status                 */
    long    *bfr;                   /* buffer pointer                   */

/*  Allocate memory for the buffer and initialize it                     */

    bfr = (long*)malloc(NBUFB);     /* Get buffer                       */
    for (i=0; i<NBUF; i++)          /* Fill with incrementing pattern   */
        bfr[i] = i;

/*  Open the HSD; issue a message and exit if open fails.                */

    if ( (fd = open("/dev/hsd0", O_RDWR)) < 0 )   {
        perror ("opening /dev/hsd0");
        exit(1);
    }

/*********************************************************************
 *              WRITE, using AUXCMD for device commands              *
 *********************************************************************/

/*      Build IOCB list of 2 device commands in 'auxlist'            */

    auxlist[0].w1.w =  0x42500000;  /* Cmd, cmd chained, dev.depend=50  */
    auxlist[0].w2.w =  1;           /* Second word = 1                  */
    auxlist[0].w3.w =
    auxlist[0].w4.w =  0;           /* Word 3 and 4 = 0                 */

    auxlist[1].w1.w =  0x40000000;  /* Cmd, not chained, dev.depend=00  */
    auxlist[1].w2.w =  0;           /* Second word = 0                  */
    auxlist[1].w3.w =
    auxlist[1].w4.w =  0;           /* Word 3 and 4 = 0                 */

/*      Set up 'auxlist' contents as list to prefix the write        */

    hsdctl.iocl = auxlist;          /* point to list in hsdctl          */
    sts = ioctl (fd, HSDAUXCMD, &hsdctl);

/*      Do the write from 'bfr', NBUFB bytes                         */

    if ( write(fd, bfr, NBUFB) != NBUFB ) {     /* check for write error*/
        perror ("Error on write");
        ioctl(fd, HSDPRVSTS, &hsdctl);
        fprintf (stderr, "Returned: errno = %d, xstatus = %d\n",
                    hsdctl.perrno, hsdctl.xstatus);
    }


/*********************************************************************
 *              READ, using AUXCMD for commands AND data             *
 *********************************************************************/

/*          2 device commands to set up for write                    */

    auxlist[0].w1.w =  0x42500000;  /* Cmd, cmd chained, dev.depend=50  */
    auxlist[0].w2.w =  1;           /* Second word = 1                  */
    auxlist[0].w3.w =
    auxlist[0].w4.w =  0;           /* Word 3 and 4 = 0                 */

    auxlist[1].w1.w =  0x42000000;  /* Cmd, cmd chained, dev.depend=00  */
    auxlist[1].w2.w =  0;           /* Second word = 0                  */
    auxlist[1].w3.w =
    auxlist[1].w4.w =  0;           /* Word 3 and 4 = 0                 */
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

```
/*          Write 5 words from 'auxbfr'                                    */

    auxlist[2].w1.w =  0x02000005;  /* Write 5 words, cmd chained         */
    auxlist[2].w2a =  (char)auxbfr; /* Write from 'auxbfr'                */
    auxlist[2].w4.w =  0;           /* Word 3 and 4 = 0                   */

/*          2 more device commands to set up for read                      */

    auxlist[3].w1.w =  0x42500000;  /* Cmd, cmd chained, dev.depend=50    */
    auxlist[3].w2.w =  1;           /* Second word = 1                    */
    auxlist[3].w3.w =
    auxlist[3].w4.w =  0;           /* Word 3 and 4 = 0                   */

    auxlist[4].w1.w =  0x40000000;  /* Cmd, not chained, dev.depend=00    */
    auxlist[4].w2.w =  0;           /* Second word = 0                    */
    auxlist[4].w3.w =
    auxlist[4].w4.w =  0;           /* Word 3 and 4 = 0                   */

/*          Initialize 'auxbfr'; clear 'bfr'                               */

    auxbfr[0] = 0x11111111;
    auxbfr[1] = 0x22222222;
    auxbfr[2] = 0x33333333;
    auxbfr[3] = 0x44444444;
    auxbfr[4] = 0x55555555;
    for (i=0; i<NBUF; i++)
        bfr[i] = 0;

/*      Set up 'auxlist' contents as list to prefix the read               */

    hsdctl.iocl = auxlist;          /* point to list in hsdctl            */
    sts = ioctl (fd, HSDAUXCMD, &hsdctl);


/*      Do the read to 'bfr', NBUFB bytes                                  */

    if ( read(fd, bfr, NBUFB) != NBUFB ) {      /* check for write error*/
        perror ("Error on read");
        ioctl(fd, HSDPRVSTS, &hsdctl);
        fprintf (stderr, "Returned: errno = %d, xstatus = %d\n",
                    hsdctl.perrno, hsdctl.xstatus);
    }

/*************************************************************************
 *                  VERIFY the data read back                            *
 *************************************************************************/

/*      First 5 words should match the contents of 'auxbfr'                */

    for (i=0; i<5; i++) {
        if ( bfr[i] != auxbfr[i] )
            printf ("Error in word %d\n", i);
    }

/*      After that, it should be the original incrementing pattern         */

    for (; i<NBUF; i++)  {
        if ( bfr[i] != i )
            printf ("Error in word %d\n", i);
    }

/*      Close VMEHSD device and exit                                        */

    close(fd);
}
_
```


Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX B
HSDBUG Terminal Operation

This appendix describes the use of HSDBUG, the built-in debugging program on the VMEHSD. The program may be used by simply attaching a serial cable to the VMEHSD's DB-9 port. Use of HSDBUG may be simultaneous with VME HOST accesses. Note however, that certain data transfers and test options will change the board's setup, and interfere with ongoing operations.

NOTE: Ensure that the 68020 processor on the VMEHSD is set for 16 Megahertz operation. Refer to VMEHSD Technical Manual, Document Number 0900052, Section 2.2.

B.1 Terminal Attachment and Setup

Any data terminal or PC/emulation software that supports the described setup may be used. Line characteristics are:

8 data bits
1 stop bit
No parity
9600 bps

The serial port connector on the VMEHSD is a standard DB-9 Male. The only connections used are RXD, TXD, and GND. The pinout is as follows:

2 - TXD out
3 - RXD in
5 - GND common.

Note that due to the high data rate (9600bps), the serial cable used should be limited to 10 meters.

B.2 Program Appearance

If a terminal is attached when the VMEHSD is reset (or powered-on), board diagnostics will appear. The revision levels and status of several tests will be displayed similar to below:

VMEHSD version X.0
ROM test passed
RAM test passed
VAC revision 1AC0

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Then, a menu similar to that below will be shown:

Function menu - press character to choose:
W - Write to a register or RAM
R - Read from a register or RAM
T - Complete test of HSD/IBL circuitry- DISCONNECT HSD CABLES first!
3 - IBL IOCB, sends 16 lwords of incrementing data
4 - IBL RD, receives up to FFFF lwords from (PC)

Note: For alphabetic choices, HSDBUG accepts only uppercase letters. If an illegal (unsupported) key is pressed, the terminal will beep, and give the following message:

The function you pressed is not supported.

B.3 HSDBUG Commands:

All available commands are described in detail in Sections B.3.1 thru B.3.5.
Available commands are:

1. Read Function
2. Write Function
3. Test Function
4. IBL Write Function
5. IBL Read Function

B.3.1 Read Function

The "R" function will prompt for an address. Enter the address of the desired location, and HSDBUG will perform a LONG read at that location, displaying its contents.

B.3.2 Write Function

The "W" function will prompt for both an address and a data value. The length of the data value supplied will determine the length of the Write executed. For example:

(data = 5B) results in a MOV.B transfer, while

(data = 005B) results in a MOV.W transfer.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

B.3.3 Test Function

The "T" command will send the board into a self-diagnostic test. The board actually generates a data pattern, sends it out to be looped back in and checked. Therefore, disconnecting any external cables is highly recommended to keep from confusing any attached external HSD device.

The board will report whether the self test passed. If the self-test failed, power the system down and attempt it again. If the failure persists, call Applied Data Sciences with a description of the environment and the error reported.

B.3.4 IBL Write Function

The "3" command will cause the VMEHSD to send out 16 longwords in the pattern:

00010203,
04050607, etc.

To use this function, you need to attach an IBL device to the VMEHSD. The IBL device should be attempting an IBL-READ operation, WAITING for a link-request in. For example, the IOCB for this on a PCHSD would be "80010010". The VMEHSD will report its final status on the HSDBUG screen. Note that after either the "3" or "4" operations, a board reset is recommended to ensure that the VMEHSD is in a "clean" state.

B.3.5 IBL Read Function

The "4" command will cause the VMEHSD to input up to FFFF longwords from an attached IBL device. The attached device should be attempting an IBL-WRITE operation, WAITING for a link request in. For example, the equivalent PCHSD IOCB would be "00010010", to send 16 longwords to the VMEHSD. The VMEHSD will display the received data on the HSDBUG screen. Since HSDBUG has no display-throttling (i.e., <CTL-S>) capability, you should keep the transfer size small in order to see all the data transferred.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

APPENDIX C
SETHSD UTILITY

The **sethsd** program is a utility which can be used to examine or alter a VMEHSD's configuration. It is normally installed as "setuser root" to allow it to issue an HSDSETCFG on a VMEHSD configuration device (normally only accessible to **root**). It is invoked with command line arguments specifying the configuration to be set and the device files to be altered. If no changes are given, the current configuration for each addressed device will be displayed. The command line must list the desired options followed by the names of the devices to be affected. **Sethsd** returns zero if successful, non-zero if any requested operation fails.

The configuration of any device may be examined; all configuration or mode changes should be directed to a configuration device.

C.1 Available Options

-D	Disable the VMEHSD (place off-line by clearing bit HSD_EXISTS in its configuration register)
-H	(Re-enable and) set device to HSD configuration
-I	(Re-enable and) set device to IBL configuration
-n	Set connector to normal (HSD) configuration
-f	Set connector to flipped (IBL) configuration
-p	Set high priority for IBL link arbitration (-I resets to low priority)

(Note: -I may be combined with -p and -n or -f)

-B	Set mode to enable byte swapping
-W	Set mode to enable word swapping
-BW	Set mode to enable byte and word swapping
-WB	Set mode to enable byte and word swapping
-N	Set mode to disable byte/word swapping
-R	Perform power-up reset

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

-T Perform device self-test

-S Read and display device status

(Note: Swapping options set on a configuration device become the defaults for all access to that unit, but may be over-ridden by an HSDSETMOD.)

-i <lev> Set VMEbus interrupt level to value <lev> (0-7). THIS OPERATION SHOULD BE USED WITH CAUTION.

-b <lev> Set VMEbus request level to value <lev> (0-3). THIS OPERATION SHOULD BE USED WITH CAUTION.

-b <lev> <mod>

Set VMEbus request level to value <lev> (0-3); set bus release mode to <mod>.
<mod> may be a value 0-3 or one of

- **ror** - release on request
- **rwd** - release when done
- **roc** - release on BCLR*
- **hold** - bus capture and hold

-a <m1>,<m2>,<m3>

Set VMEbus address modifiers to <m1>, <m2>, and <m3> for DCB, IOCB's and data buffers, respectively. Missing values, indicated by adjacent commas will not be altered.

The following options involve I/O operations and, as such, should be directed to a normal VMEHSD device file (not the configuration device).

-R Perform a board re-boot. Hardware and firmware are restored to an initial power-up condition.

-T Perform a self-test on board.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

C.2 EXAMPLES:

sethsd -I /dev/hsd0c

Set VMEHSD unit 0 to IBL mode, other parameters defaulted: normal (HSD) connectors, low priority.

sethsd -If /dev/hsd0c

Same as above, but set connectors to flipped (IBL) configuration.

sethsd -Inp -W -b 2 rwd

Configure VMEHSD unit zero as an IBL with HSD (normal) connectors and high link arbitration priority, performing word swaps on all data by default and accessing the VMEbus at level 2, releasing when each transfer is done.

sethsd -H -BW /dev/hsd1c

Configure VMEHSD unit one as an HSD performing byte and word swaps on all data transfers by default.

sethsd -D /dev/hsd2c

Disable VMEHSD unit two, and place it off-line.

Artisan Technology Group - Quality Instrumentation ... Guaranteed | (888) 88-SOURCE | www.artisantg.com

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP