IEEE CAMAC routines for the Hytec[1] VSD 2992 CAMAC Serial Branch Highway Driver

J. Criz, D. Slimmer, J. Streets

Online Support Department
Fermilab Computing Division

ABSTRACT

A description of the supported IEEE CAMAC Routine interface and Fermilab extensions for the Hytec VSD2992 Serial Branch Highway Driver.

1. Hytec Electronics Ltd., Ladbroke Close, Woodley, Reading , Berkshire, RG5 4DX, ENGLAND.

1 Introduction

This document is the supplement to a library of routines for the Hytec VSD2992 (VSD), a 6u single width VME module which functions as a CAMAC serial highway driver. The VSD can send and receive CAMAC messages in bit serial or byte serial modes while being controlled over the VME bus. The library consists of implementations of ANSI/IEEE standard 758-1979 CAMAC subroutines written for IRIX and VxWorks. This library is contained within the VSD2992 product and is described below. In addition, some points concerning VSD operation and installation are clarified.

Older tests on the Hytec were done to test speed in comparison to the CES[1] CBD8210, and to test the reliability of the VSD2992. These test results, recorded in Appendix A, show the speed of the C library, which has been written according to standard IEEE formats, and implements many of the functions previously listed in document PN 219 - "IEEE Standard CAMAC routines".

1. Creative Electronic Systems S.A., Route du Pont-Butin 70, CH-1213 Petit-Lancy 1 / GENEVA SWITZERLAND.

2 Installation

2.1 Hytec VSD2992

The VSD2992 is a VME module. When installing the VSD, there are several important hardware settings which need to match the settings of this software package in order to function properly. As detailed in the VSD manual, switches SW3, SW4, and SW5 are used to set the clock speed, and SW2 is used to set the byte spacing. It is recommended that the user sets the clock speed to 5MHz, although it is not absolutely necessary. SW2 is the number of space bytes for reads and writes. All tests were run with both values set to 16 bytes, but the user is free to leave these at the Hytec recommended values.

2.2 VME Address

SW1 sets the location of the beginning of the VSD's VME memory location. The IEEE library assumes that branch 0 corresponds to 0XC000 (or SW1 set to all 0's) so these switches must be set accordingly. Higher addresses are accessed by the routines with the formula

```
VME Address = 0xC000 + BRANCH*0x40.
```

Note: for switches SW1 and SW2, ON=0, OFF=1. The software also uses byte-serial data transfers, therefore LK1 must be set to pins 1 and 2.

2.3 Suggested Switch Settings

The following switch settings were used for our tests.

TABLE 1.

Switch	Setting	Meaning
SW1	all on	VME Address = 0xC000
SW2	all off	Maximum byte space padding
SW3	2 - North	5Mhz
SW4	3 - West	5Mhz
SW5	3 - West	5Mhz Crystal
LK1	1-2	Byte Serial
LK2	2-3	undocumented
LK3-LK9	LK8	Interrupt level 6
LK10-LK16	LK11	Interrupt Acknowledge 6
LK17	2-3	Delay IN

2.4 Software Library

The software library supports software on machines running IRIX5 (`$VSD2992_DIR/lib/libvsd.`*`IPNN`*`.a`) or VxWorks (`$VSD2992_DIR/lib/libvsd.m68040.o`). The routines have been used on IRIX4 version but requires a VME driver. Details for this are included in Appendix A. VxWorks maps the registers directly and the IRIX5 driver uses the standard VME device file */dev/vme/vme0a16a*.

The IRIX version of the VSD IEEE routines depend on the SGI VME interfaces. The VME interface depend on the type of CPU. CPUs are labelled *IPNN,* and can be found with the command `hinv` or `uname -a`. The following installation procedures will create a library suitable for the CPU on which the installation procedure is run. If the disks are mounted via nfs on several types of machine, then the user needs to be aware of which library to use in `$VSD2992_DIR/lib`.

2.5 SGI with IRIX 5

The VSD library needs write access to the VME device files. The default IRIX installation sets these as read only. To change this:

- Login as root
- Change write preferences of files under /dev/vme

```
% cd /dev/vme
% chmod 666 *
```

2.6 Multi-User support

The VSD IEEE routines use a single semaphore based on the file-name */dev/vme/vme0a16a* to allow more than one user to access the device at the same time. This single semaphore is reserved in the system on the very first access to the device after the computer is rebooted. The IEEE routines can be used by two processes accessing two different modules without data corruption.There is currently no method for reserving crates for one particular user.

Reserving and releasing semaphores in IRIX take a significantly longer time than the CAMAC transfer. We therefore include a dummy set of routines which do not take and replace semaphores for situations where speed is essential, in single-user applications. The dummy routines can be used by linking against the file `$VSD2992_DIR/lib/vsd_semnoops.o` before the library `$VSD2992_DIR/lib/libvsd.IP`*`NN`*`.a`. The CAMAC rate increases from 9kHz to 60kHz with the dummy semaphore library, however users should be aware that the driver is no longer multi-user.VxWorks semaphores are fast compared to the CAMAC transfer time, and so the dummy routine is not needed.

2.7 Multi-Branch Support

The VSD IEEE routines currently support multiple VSD boards from the same program. The branch number is used to define the VME base address of the module.

```
VME Address = 0xC000 + BRANCH*0x40.
```

2.8 SGI Installation of Software

The library and test programs have been compiled on a Crimson and a Challenge running IRIX 5.2. The software needs to be recompiled for different CPU types and OS releases. The following commands will create the library.

```
% setup vsd2992
% cd $VSD2992_DIR/src
% rm *.o
% premake -f IRIX
```

The library will be made in `$VSD2992_DIR/lib/libvsd.`*`IPNN`*`.a`. Where *IPNN* is the type of processor, which can be seen from `hinv`, or `uname -a`. Examples programs are located in `$VSD2992_DIR/example`. If the string

```
fndaul 125% premake -f IRIX
No VME interface. Wrong type of SGI.
```

gets printed during the make, this means that the SGI on which you are building does not have a VME interface.

2.9 Examples

The VSD2992 product contains examples in the `$VSD2992_DIR/example directory`. To compile these, type the following:

```
% cd $VSD2992_DIR/example
% rm *.o
% ../src/premake -f IRIX
```

The programs included are as follows.

- pdt - writes (f16) and reads (f0) incrementing PDTs to a module, checking data integrity. Can be used on a regular dataway display module. Accepts slot number as first argument.
- dma - reads (f2) incrementing DMAs from a Fermilab Benchmark module, checks data integrity.Accepts slot number as first argument.
- qxtab - builds a table for a module showing the Q and X response for any (f0-f24, a0-a15) CAMAC command.

3 VSD Software Library

This software library was written to control the VSD from IRIX or VxWorks, using the ANSI/IEEE CAMAC standard. The routines read or write data to memory mapped registers on the VSD, where the commands are sent to the CAMAC serial crate controller. Most commands poll for controller responses. The completion check is made by checking the XMR and SHR bits on the Hytec (bits 8 and 9 of register 04) - and then an error is checked for by examining the status of the SQ bit (bit 6 of register 04).

All functions have been written in C; all argument types are considered pass-by-value unless they are listed with a "*", in which case they are pass-by-reference. FORTRAN programming stubs have also been written in order to allow each procedure to be called from FORTRAN code, although these were also written in C. Each of these FORTRAN routines have names followed by an underscore (_), and have all arguments passed by reference. The C routines return Murmur Error Codes, additional information can sometimes be found in the *errno* variable which is set by system calls. The FORTRAN routines are "void" and do not return values, unless they are functions. A separate function (`camerr`) has been given to save the error from the last FORTRAN command.

Some of the CAMAC IEEE routines have not been written - most of these concern branch operations, LAM interrupts, and booking. These functions are not currently supported by the VSD2992.

3.1 Argument names used in IEEE Standard Routines

The following names are used as arguments in the VSD implementation of the IEEE CAMAC routines. Note that integers are declared as C `int`, and are 32 bits long on VxWorks and IRIX 5. For the short word transfers, we use `unsigned short`, which are 16 bits long.

3.1.1 branch

An integer (0-255), indicating the VME address of the VSD2992. The VME address of the module should be set to 0xC000+0x40*branch.

3.1.2 cb

A control block, a 4 integer array containing the

1. Requested Repeat Count.
2. Tally (Result repeat count).
3. LAM Identification (not used).
4. Channel Identification (not used).

3.1.3 crate

3.1.4 An integer (1-62), indicating the setting of the crate controller.

3.1.5 *data16

A pointer to an array of 16 bit integers.

3.1.6 *data24

A pointer to an array of 32 bit integers.

3.1.7 ext

External Address, a 32 bit integer which contains the branch, crate, slot and sub-address of a CAMAC module.

3.1.8 function

An integer (0-24) to indicate the CAMAC function.

3.1.9 k

Q and X response.

TABLE 2. k: Q and X response

Value of k	Value of Q	Value of X
0	1	1
1	0	1
2	1	0
3	0	0

3.1.10 lflag

An integer containing 1 for true and 0 for false.

3.1.11 q

An integer containing 1 if there was a Q response, and 0 if there was not a Q response.

3.1.12 slot

An integer (1-30) to indicate the slot address of the CAMAC module.

3.1.13 sub-address

An integer (0-15) to indicate the sub address of the function for the CAMAC module.

3.2 IEEE Standard Routines

3.2.1 cdreg(*ext, branch, crate, slot, sub-address)

Encodes branch, crate, slot and sub-address into ext.

3.2.2 cgreg(ext, *branch, *crate, *slot, *sub-address)

Decodes ext into branch, crate, slot and sub-address.

3.2.3 cccc(ext)

Clear the CAMAC dataway pointed to by ext.

3.2.4 ccci(ext, lflag)

Set (lflag=1) or clear (lflag=0) CAMAC crate inhibit.

3.2.5 cccz(ext)

Initialize CAMAC branch.

3.2.6 cfsa(function, ext, *data24, *q)

Executes a single 24 bit CAMAC function.

3.2.7 cfmad(function, ext, *data24, *cb)

Execute a number of 24 bit CAMAC functions, while incrementing the sub-address when there is a Q response or incrementing the slot when there is no Q response.

3.2.8 cfubc(function, ext, *data24, *cb)

Execute a number of 24 bit CAMAC functions at the same address, until there is no Q response.

3.3 IEEE System-Dependent Subroutines

These subroutines are not part of the ANSI/IEEE Std 758-1979, but are listed in its appendix.

3.3.1 ctstat(k)

Return status code of last CAMAC operation.

3.3.2 cdchn(branch, lflag, *route)

Opens up a branch, maps to the VME address. Note that we have followed the PN219 definitions of the arguments, rather than the appendix in ANSI/IEEE Std 758-1979, to conform to existing code. Branch=0 corresponds to base address C000 of the VSD2992, branch=1 corresponds to C040, *etc*. Currently, multi-branch systems are not supported.

3.3.3 cssa(function, ext, *data16, *q)

Execute a 16 bit mode CAMAC function.

3.3.4 csmad(function, ext, *data16, *cb)

Execute a number of 16 bit CAMAC functions, while incrementing the sub-address when there is a Q response or incrementing the slot when there is no Q response.

3.3.5 csubc(function, ext, *data16, *cb)

Execute a number of 16 bit CAMAC functions at the same address, until there is no Q response.

3.4 FNAL Extensions to CAMAC Standard Routines

These routines are not in the ANSI/IEEE Std 758-1979, or in its appendix, however they are listed in PN219, and have been found useful in the past.

3.4.1 camerr

Returns murmur status from last FORTRAN call.

3.4.2 cccbyp(ext,lflag)

Set (lflag=1) or clear (lflag=0) SCC bypass.

Returns:

- VSD_S_INVLD_CR - if crate is not in the range 1-64 inclusive.
- VSD_S_INVLD_ARG - if option argument is not 0(clear) or 1(set).
- VSD_S_SUCCESS - if above errors do not exist.

3.4.3 cccoff(ext, lflag)

Set SCC on-line (lflag=0) or off-line (lflag=1).

Returns:

- VSD_S_INVLD_CR - if crate is not in the range 1-64 inclusive.
- VSD_S_INVLD_ARG - if option argument is not 0(online) or 1(offline).
- VSD_S_SUCCESS - if above errors do not exist.

3.4.4 qrespn

A short integer function. Get Q response from last CAMAC operation.

3.4.5 xrespn

A short integer function. Get X response from last CAMAC operation.

3.4.6 camerr

An integer function, with no arguments. Returns the last FORTRAN error.

3.4.7 cmsg(char *subroutine, int data, int branch)

Prints the status registers 0x04 and 0x0E to the screen for a particular branch. The first 2 arguments are also printed, and can be used to indicate trace information in the output, to help debug a program.

3.4.8 ccctype

This routine has no affect, it is included for compatibility with existing IEEE FORTRAN programs.

3.4.9 cderr

This routine has no affect, it is included for compatibility with existing IEEE FORTRAN programs.

3.4.10 csuns

This routine has no affect, it is included for compatibility with existing IEEE FORTRAN programs.

3.5 VSD Controller Routines

These routines are included for backwards compatibility.

3.5.1 vsd_csr_rd

DESCRIPTION

Read CSR register.

C PROGRAM CALL FORMAT

```
Input:
csr_offset - CSR to read, offset defined in terms of vme_ptr from vsd_config.h
type:        int
access:      read only
mechanism:   by value

Output:
vsd_csr - 16 bit value read from CSR
type:        unsigned short *
access:      write only
mechanism:   by reference

Returns:
VSD_S_SUCCESS - by default
```

3.5.2 vsd_csr_wr

DESCRIPTION

Write to CSR register setting one bit.

C PROGRAM CALL FORMAT

```
Input:
```

```
csr_offset - CSR to write, offset defined in terms of vme_ptr from
vsd_config.h
type:       int
access:     read only
mechanism:  by value

bitnum - bit number to set/clear, starts at bit 0
type:       int
access:     read only
mechanism:  by value

bitval - bit value, 1=set, 0=clear
type:       int
access:     read only
mechanism:  by value

Returns:
VSD_S_SUCCESS - by default
```

3.5.3 vsd_decode_csr

DESCRIPTION

Decode csr register into its (16) bit form.

C PROGRAM CALL FORMAT

```
Input:
csr_offset - CSR to decode, offset defined in terms of vme_ptr from
vsd_config.h
type:       int
access:     read only
mechanism:  by value

read_csr - csr value to decode
type:       long int
access:     read only
mechanism:  by value

bit_dec_csr - scratch area for subroutine
type:       long int [16]
access:     read only
mechanism:  by value

Returns:
VSD_S_SUCCESS - by default
```

3.5.4 vsd_gl_rd

DESCRIPTION

Read SCC GL register. It should be possible to read this information from the VSD2992 directly, but technical assistance from HYTEC was not available to address this problem before this writing.

C PROGRAM CALL FORMAT

```
Output:
vsd_gl - VSD graded LAM register value
type:       unsigned short *
access:     write only
mechanism:  by reference

crate - CAMAC crate number containing demands
type:       long int *
access:     write only
mechanism:  by reference

Returns:
VSD_S_NO_DMD - no LAM/s present
VSD_S_INVLD_DMD - bit M2 not set (not a TRUE demand message)
VSD_S_SUCCESS - errors above do not exist
```

4 IEEE routines not supported

The following routines are not yet in the IEEE function library for the VSD2992.

- CTCI: Test Dataway Inhibit
- CCCD: Enable or Disable Crate Demand
- CTCD: Test Crate Demand Enabled
- CTGL: Test Crate Demand Present
- CDLAM: Declare LAM
- CCLM: Enable or Disable LAM
- CCLC: Clear LAM
- CTLM: Test LAM
- CCLNK: Link LAM to Service Procedure
- CFGA: General Multiple Action
- CFUBL: LAM-Synchronized Block Transfer
- CFUBR: Repeat Mode Block Transfer
- CGLAM: Analyze LAM Identifier
- CGREG: Analyze Register Identifier

4.1 Unsupported Routines in the ANSI/IEEE Appendix

- CCINIT: Branch Initialize
- CGCHN: Analyze Channel Declaration
- CSGA: General Multiple Action
- CSUBL: LAM-Synchronized Block Transfer
- CSUBR: Repeat Mode Block Transfer
- CDCRT: Define Crate Identifier

5 Release Notes

5.1 Version 1.0

First release of IRIX support. Result of Josh Criz's summer student project.

5.2 Version 1.3

Added MURMUR codes, minor bug fixes, documentation update. Release for testing at E781 by Alexandre Atamantchouk

5.3 Version 1.4

Add `cmsg()` and `camerr()`. Include `qxtab()`. Bug fixes from E781:

- fix `ctstat`.
- `cccbyp` now waits longer than for C, Z or I before returning. BYPASS is hardware relays and is delayed by the crate controller for 100ms (see the ANSI/IEEE CAMAC standard).
- Major change from `long int` to `int` for function arguments. IRIX 6 will have 64 bit `long ints`.

5.4 Version 1.5

Minor bug fixes.

5.5 Version 1.6

Support for multiple branches added.

5.6 Version 1.7

Fixed bug in `cmsg()`. Added routine `$VSD2992_DIR/lib/vsd_semnoops.o` to avoid delay with semaphore for single-user systems.

5.7 Version 1.8

Changed library names to conform to DART standards - use "." and not "_" to separate names from the CPU type, and use "m68040" and not "VXWORKS".

Appendix A: SGI with IRIX 4

The VSD software includes the IEEE CAMAC interface on an SGI with an internal VME bus. Tests were made on an SGI 4D/35 running IRIX 4.0.5F, which has a single VME slot. The VSD software requires the physical memory of the SGI to be mapped to the VME address if the VSD2992. To do this, one needs to link the operating system to the new kernel driver, called rvme.

The steps necessary to link and install the new kernel on IRIX 4 are as follows:

- Login as root
- Compile $VSD2992_DIR/driver/rvme.c

  ```
  % cd $VSD2992_DIR/driver
  % cc -c rvme.c
  ```
- Copy the file rvme.o to master.d

  ```
  % cp rvme.o /usr/sysgen/boot/rvme.o
  ```
- Append the following line to /usr/sysgen/system

  ```
  INCLUDE: rvme
  ```
- Add in the rvme driver to the /usr/sysgen/master.d/mem

  ```
  #ifdef IP19
  {0x300000,0xC0014000,}, /* rvme dummy device driver */
  #endif
  ```
- Create the file /usr/sysgen/master.d/rvme

  ```
  *FLAG PREFIX SOFT #DEV DEPENDENCIES
  sc vme 60
  $$$
  ```

  (Note this assumes that the major device number 60 is not already in use).
- Create the device file

  ```
  % mknod /dev/rvme c 60 0
  ```

  (Note this assumes that the major device number 60 is not already in use).
- Relink /unix

  ```
  % lboot -v -u /unix.install
  ```
- Reboot with new kernel

  ```
  % reboot
  ```

A message should appear on the console, informing the user that the new rvme driver has been loaded.

Appendix B: VSD Operation

B.1 cam_test

Its use is primarily for testing library routines and taking timings with the VMETRO. It is not very flexible in regard to taking timings - re-compilation is required. This was purposely done to avoid writing complex code to make the test user friendly which might affect timing measurements.

The one part of cam_test that should be a stand alone routine is the code executed by the menu command that enables or disables demands. LAMs were the last item tested however, and this was not done.

The menu command to read graded LAMs calls vsd_gl_rd. This routine is still in a development state, and prints values to terminal for debugging purposes.

B.2 Testing

The board and software have been tested for speed and reliability. The VSD was tested in both 16 bit and 24 bit modes; it should be noted that the VSD **always** sends or reads 24 bits of data. It is up to the user to make sure that they don't send a single 16-bit word to a module that is reading 24 bits, because whatever was written or read into the upper 8 bits during the last 24-bit operation will still be in the register. The VSD was also tested in both normal and repeat modes. Although repeat mode is faster than single reads and writes, it does not check whether an error has occurred during operations, so care should be taken to make sure that proper reads and writes are being done before use.

All tests were run on the VSD while set to 5Mhz - no errors occurred. However, the preset number of space bytes was left at the maximum, 16 for both reads and writes. The Unix-based computer used was a Silicon Graphics (SGI) 4D/35. Tests were done by a simple looping time testing program, which checked both the calendar-clock and the processor-ticks clock, before and after an operations-calling loop.

test conditions:

- read from SGI 4D/35 with KS 3952 Type L-2 driven by Hytec VSD2992
- VSD2992 clock speed set to 5MHz
- VSD2992 configured for byte serial transfers with 16 space bytes, 25' heavily shielded cables.

test conditions of original results (row 1):

- read from FERMI Benchmark Module with KS 3952 Type L-2 driven by Hytec VSD2992
- VSD2992 clock speed set to 2.5MHz (data corruption at 5MHz)

- VSD2992 configured for repeat mode, byte serial, 6 foot flat ribbon cable

TABLE 3. Timings

	16R	**16W**	**24R**	**24W**
VSD/MVME 167 33MHz	7.25µs		8.55µs	
Normal (CSSA/CFSA)	14.11µs	14.25µs	15.78µs	15.75µs
Repeat (CSSA_FAST)	7.06µs	9.77µs	8.78µs	11.25µs

Note that the IRIX tests are in single-user mode. We have used a single semaphore to implement multiple users accessing the VSD2992 without data corruption. Taking and replacing a semaphore in IRIX can take approximately 90 microseconds on a 4d/35 running IRIX 5.2. In order to use the single-user mode, refer to Chapter 2. Semaphore operations in VxWorks are take a negligible amount of time when compared to the CAMAC transfer time.

Appendix C: CES8210 Testing

The CES CBD8210 is a VME driver that is used as a parallel CAMAC controller. The VSD2992 was originally tested in comparison to the CBD, because many crates connected in a parallel configuration were driven by CBD's, and thus the CBD was used as a standard.

From a user's point of view, the difference between the Hytec and the CBD8210 is in the programming model and in data moving performance. The CBD is controlled with memory mapped commands, while the VSD is controlled through registers. Timing measurements show the CBD to be about twice as fast in 16 bit and 24 bit operations.

There are a couple basic differences in application that apply in general to serial and parallel highway drivers. Serial highway cables can be extended up to [1]1000 feet as compared to [2]100 feet for parallel branch cables. Serial highways can control up to 64 serial CAMAC crate controllers (SCCs), where the parallel highway can control only 7 parallel CAMAC crate controllers.

CBD timing summary extracted from $CBD8210/doc/cbd_timing.txt for comparison:

TABLE 4.

	16R	**16W**	**24R**	**24W**
CBD/MVME 167	2.24µs	2.24µs	3.04µs	2.40µs
CBD/SPARC	3.30µs	3.24µs	5.28µ	4.84µs

1. Using special cable that contains shielded, twisted pairs.

2. May be up to 300 feet, but operation is only guaranteed to 100 feet.