Fermi National Accelerator Laboratory

TM-1478

VFI
VME/FASTBUS Interface Routines

Dean Alleva
Development and Evaluation Group
RD/Computing
Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

September 10, 1987

Operated by Universities Research Association Inc. under contract with the United States Department of Energy

Note Number XXXX

VFI
VME/FASTBUS Interface Routines

-Version-
Software:1.0
Document:1.0

September 10, 1987

Dean Alleva
Development and Evaluation Group
RD/Computing
Fermilab

TABLE OF CONTENTS

1 INTRODUCTION

This document describes the VME/FASTBUS Interface routines (VFI). These routines where designed to enable programs written in PILS running on a MVME 101 under Valet-plus to control a VME/FASTBUS Interface [1], [2]. The routines fall into two general types, control and transaction. The control rountines, such as vmec_reset, work directly with the interface's registers. These routines set up and monitor operations between VME and FASTBUS. The transaction routines, such as vfi_write_dat, use the control routines to carry out complete functions on FASTBUS. Most FASTBUS operations are implemented except for the compound routines and some low level routines.

The routines are written in PILS, a high-level language similar to BASIC and Pascal which is powerful and fast enough for most applications. One of the most powerful features of the Valet/PILS system is the ability to set up exception vectors and exception handlers directly in a program. This feature is used to handle interrupts from the VME/FB interface.

This document is divided into seven sections, the first is the introduction. The remaining sections detail the interface's registers, the structure of operation codes, polling and interrupts, status reporting, the control routines, and the transaction routines.

It is assumed that the reader is familiar with VME, FASTBUS, and has some knowledge of the interface's design [2]. A copy of the VFI software is available on BitNet at Fermilab as "FNAL::USR$ROOT:[ALLEVA.PUBLIC]VFI.SRC".

2 VFI REGISTERS

The VME/FASTBUS Interface is a register controled system. Parameters for a given operation are loaded into the interface's registers. Execution of an operation begins by setting special bits in the control register. An operation can then be monitored by polling the control register or through interrupts.

A brief description of each register is given below. A detailed description of these registers is given in the document describing the interface's design [2].

VME Control Register: Used to start execution of an operation, select polling or interrupts, reset the interface, and monitor execution (using polling.)

FASTBUS Status Register: Contains an error code if the last operation ended with an error.

VME Interrupt Registers: There are two of these registers, one for errors and one for success. These registers are loaded with interrupt vectors used to signal operation completion.

Primary Address Register: Loaded with the FASTBUS primary address to be used in an operation.

Secondary Address Register: Loaded with the FASTBUS secondary address to be used in an operation.

DMA Word Count, Address, and Control: These registers are used to do DMA transfers between the interface's buffer memory and FASTBUS.

Operation Register: Loaded with a code to control an operation on FASTBUS, see the next section.

3 OPERATION CODES

Each VME/FASTBUS operation carried out by the interface is controled by a special code loaded into the operation register. This code tells the interface which cycles are to be done, what MS codes to use, as well as what special functions to carry out. The operation register is a 32-bit register but only the lower 24 bits are usable, their functions are detailed below. For a full description of the operation register see the document describing the VME/FB interface.

BIT Number*	Function
0	Status clock, set if status is to be returned.
1	Do an arbitration cycle
2	Do a primary address cycle
3	Do a secondary address cycle
4	Do a data cycle
5	Release GK (after operation)
6	Release AS (after operation)
7	Not used
8-10	Primary address cycle MS code
11	Not used
12-14	Data cycle MS code
15	Not used
16	Read Select, set if operation is a read
17	Set EG, set if EG is to be set high by interface
18	Block transfer, set if operation is a block transfer
19-23	Not used

* All bits are high enabled.

4 INTERRUPTS AND POLLING

Two methods are used to monitor an operation. In polling mode, the control register value is fetched in a loop and two bits are checked. The Master Ready bit is set by the interface when an operation is complete. The Error bit is set if an error occured during execution and indicates that the status register contains the error's code. The status register is checked only if the Error bit is set.

When interrupt mode is used, a loop is entered until an interrupt flag is set by an interrupt handler. An interrupt handler is executed upon an interrupt. Using the Valet/PILS interrupt vector routines, two vectors are set up. One vector, loaded into the error interrupt register, activates an error interrupt handler. The normal interrupt register is loaded with a vector which activates a normal interrupt handler. Upon execution, either of these routines will set the interrupt flag. The status register is checked only if an error interrupt has taken place.

This type of interrupt handling is basicly polling for an interrupt. Valet-plus, however, has no "wait for interrupt" command. In a more sophisticated mulitasking environment, a wait for interrupt would enable the OS to block processes and execute others.

5 STATUS REPORTING

Once a command has been executed, the status may be fetched in any of two ways. First, a call to vfi_display_error will return the status code of the last command executed. Vfi_display_error also prints to the display device a message indicating the error code returned.

Status may also be fetched with a call to vfi_get_error, which returns status without displaying any messages. Note that these routines should be called after an operation is executed.

Originally, status reporting was to be done in the interrupt handler routines. Status was to be returned to the main program through a parameter in the transaction routine headers. However, Valet-plus can not handle IO (get, put, print, etc.) correctly while interrupts from VME are pending. When a more sophisticated operating system becomes available status reporting will be changed.

6 THE CONTROL ROUTINES

These routines do low level operations on the interface registers.

6.1 VFI Initialize

1) VMEC_RESET (set_inter)
   VCR

   Description: Resets the interface and initializes internal VFI data values. Should be the first VFI call in a program.

   Parameters:

   set_inter (INT32, input): If set to 1, VFI uses interrupts to monitor an operation. Otherwise, VFI uses polling.

6.2 Loading A Command

These routines are usually not used directly by a program. The load routines are called by the transaction routines to set up the interface registers for a operation.

1) VMEC_RUN_BLOCK (contval, primadd, secadd, dmaadd, dmawc)

Description: Loads operation parameters into the interface registers and executes the command. This command should be a block transfer command.

Parameters:

contval	(INT32, input):	Control register opcode value.
primadd	(INT32, input):	Primary address value.
secadd	(INT32, input):	Secondary address value.
dmaadd	(INT32, input):	Interface buffer address where transfer starts.
dmawc	(INT32, input):	Number of 32-bit words to transfer.

2) VMEC_RUN_NO_BLOCK (contval, primadd, secadd, buffadd)

Description: Similar to VMEC_RUN_BLOCK, but this routine is used for non-block transfers.

Parameters:

contval	(INT32, input):	Control register opcode value.
primadd	(INT32, input):	Primary address value.
secadd	(INT32, input):	Secondary address value.
buffadd	(INT32, input):	Inferface buffer address where transfer takes place (one word).

6.3 Execution Control

These routines a called by VMEC_RUN_BLOCK or VMEC_RUN_NO_BLOCK to execute a command and the wait for completion. In general, these routines are not used directly by a program.

1) VMEC_GO

Description: Starts the interface.

2) VMEC_WAIT

Description: Returns when interface is done with operation.

6.4 Loading CSR-8 Value

1) VMEC_LOAD_CSR8 (csr8_value)
   VCLCSR

   Description: Loads the interface's CSR-8 register. This routine should be used to set the CSR-8 register before any calls to the transaction routines.

   Parameter:
   csr8_value (INT32, input): value for CSR-8 register.

7 THE TRANSACTION ROUTINES

These routines do complete operations on FASTBUS from VME. These routines are the basic building block from which more complex compound routines can be built.

7.1 Read And Write Data

7.1.1 Single Word Transfers -

1) VFI_READ_DAT (pradd, scadd, bfadd)

Description: Does a single word transfer to the interface's internal memory buffer from FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is placed.

2) VFI_WRITE_DAT (pradd, scadd, bfadd)

Description: Does a single word transfer from the interface's internal memory buffer into FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is to be taken.

3) VFI_READ_DAT_MULT (pradd, scadd, bfadd)

Description: Does a broadcast single word transfer into the internal memory buffer from FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is placed.

4) VFI_WRITE_DAT_MULT (pradd, scadd, bfadd)

Description: Does a broadcast single word transfer from the interface's internal memory buffer into FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is to be taken.

5) VFI_READ_DAT_SA (pradd, scadd, bfadd)

Description: Does a single word transfer, using the NTA register, into the interface's internal memory buffer from FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is placed.

6) VFI_WRITE_DAT_SA (pradd, scadd, bfadd)

Description: Does a single word transfer, using the NTA register, from the interface's internal memory buffer into FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is to be taken.

7.1.2 Block Transfers -

1) VFI_READ_DAT_BLOCK (pradd, scadd, bfadd, cnt)

Description: Does a block transfer to the interface's internal memory buffer from FASTBUS data space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address where block is transfered.
cnt	(INT32, input):	Number of 32-bit words to transfer

2) VFI_WRITE_DAT_BLOCK (pradd, scadd, bfadd, cnt)

Description: Does a block transfer from the interface's internal memory buffer into FASTBUS data space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address of transfer block.
cnt	(INT32, input):	Number of 32-bit words to transfer

3) VFI_READ_DAT_BLOCK_MULT (pradd, scadd, bfadd, cnt)

Description: Does a broadcast block transfer to the interface's internal memory buffer from FASTBUS data space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address where block is placed.
cnt	(INT32, input):	Number of 32-bit words to transfer

4) VFI_WRITE_DAT_BLOCK_MULT (pradd, scadd, bfadd, cnt)

Description: Does a broadcast block transfer from the interface's internal memory buffer into FASTBUS data space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address of transfer block.
cnt	(INT32, input):	Number of 32-bit words to transfer

7.2 Read And Write CSR

7.2.1 Single Word Transfers -

1) VFI_READ_CSR (pradd, scadd, bfadd)

Description: Does a single word transfer to the interface's internal memory buffer from FASTBUS CSR space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address where transfer word is placed.

2) VFI_WRITE_CSR (pradd, scadd, bfadd)

Description: Does a single word transfer from the interface's internal memory buffer into FASTBUS CSR space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address where transfer word is to be taken.

3) VFI_READ_CSR_MULT (pradd, scadd, bfadd)

Description: Does a broadcast single word transfer to the interface's internal memory buffer from FASTBUS CSR space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address where transfer word is placed.

4) VFI_WRITE_CSR_MULT (pradd, scadd, bfadd)

Description: Does a broadcast single word transfer from the interface's internal memory buffer into FASTBUS CSR space.

Parameters:

pradd	(INT32, input):	Primary address
scadd	(INT32, input):	Secondary address
bfadd	(INT32, input):	Interface internal address where transfer word is to be taken.

5) VFI_READ_CSR_SA (pradd, scadd, bfadd)

Description: Does a single word transfer, using the NTA register, into the interface's internal memory buffer from FASTBUS CSR space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is placed.

6) VFI_WRITE_CSR_SA (pradd, scadd, bfadd)

Description: Does a single word transfer, using the NTA register, from the interface's internal memory buffer into FASTBUS CSR space.

Parameters:

pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where transfer word is to be taken.

7.2.2 Block Transfers -

1) VFI_READ_CSR_BLOCK (pradd, scadd, bfadd, cnt)

Description: Does a block transfer to the interface's internal memory buffer from FASTBUS CSR space.

Parameters:

```
pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where block
                      is transfered.
cnt   (INT32, input): Number of 32-bit words to transfer
```

2) VFI_WRITE_CSR_BLOCK (pradd, scadd, bfadd, cnt)

Description: Does a block transfer from the interface's internal memory buffer into FASTBUS CSR space.

Parameters:

```
pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address of transfer
                      block.
cnt   (INT32, input): Number of 32-bit words to transfer
```

3) VFI_READ_CSR_BLOCK_MULT (pradd, scadd, bfadd, cnt)

Description: Does a broadcast block transfer to the interface's internal memory buffer from FASTBUS CSR space.

Parameters:

```
pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address where block
                      is placed.
cnt   (INT32, input): Number of 32-bit words to transfer
```

4) VFI_WRITE_CSR_BLOCK_MULT (pradd, scadd, bfadd, cnt)

Description: Does a broadcast block transfer from the interface's internal memory buffer into FASTBUS CSR space.

Parameters:

```
pradd (INT32, input): Primary address
scadd (INT32, input): Secondary address
bfadd (INT32, input): Interface internal address of transfer
                      block.
cnt   (INT32, input): Number of 32-bit words to transfer
```

7.3 Single Cycle Operations

7.3.1 Aribration Cycle -

1) VFI_CYCLE_ARBITRATE

Description: Does an arbitration cycle, holding onto the bus when control is gained. Note that an error code of hex 91 (No PRIM and No AS/AK Lock) indicates success for this routine.

7.3.2 Bus Release -

1) VFI_CYCLE_RELEASE_BUS

Description: Releases the bus by lowering GK and AS.

7.3.3 Primary Address Cycle -

1) VFI_CYCLE_PA_DAT (pradd)

Description: Does a primary address cycle to data space.

Parameters:
pradd (INT32, input): The primary address

2) VFI_CYCLE_PA_CSR (pradd)

Description: Does a primary address cycle to CSR space.

Parameters:
pradd (INT32, input): The primary address

3) VFI_CYCLE_PA_DAT_MULT (pradd)

Description: Does a broadcast primary address cycle to data space.

Parameters:
pradd (INT32, input): The primary address

4) VFI_CYCLE_PA_DAT_MULT (pradd)

Description: Does a broadcast primary address cycle to CSR space.

Parameters:
pradd (INT32, input): The primary address

7.3.4 Secondary Address Cycle -

1) VFI_CYCLE_READ_SA (bfadd)

Description: Does a secondary address cycle read. Bus mastership and primary address cycle must be completed before using this routine.

Parameters:

bfadd (INT32, input): Internal buffer address where secondary address is written.

2) VFI_CYCLE_WRITE_SA (bfadd)

Description: Does a secondary address cycle write. Bus mastership and primary address cycle must be completed before using this routine.

Parameters:

bfadd (INT32, input): Internal buffer address where secondary address is taken.

7.3.5 Data Cycle -

1) VFI_CYCLE_READ_WORD (bfadd)

Description: Does a single word read cycle. Bus mastership and primary address cycles must be completed before using this routine.

Parameters:
bfadd (INT32, input): buffer address where word is transfered

2) VFI_CYCLE_WRITE_WORD (bfadd)

Description: Does a single word write cycle. Bus mastership and primary address cycles must be completed before using this routine.

Parameters:
bfadd (INT32, input): buffer address of word to transfer

3) VFI_CYCLE_READ_BLOCK (bfadd, cnt)

Description: Does a block read cycle. Bus mastership and primary address cycles must be completed before using this routine.

Parameters:
bfadd (INT32, input): buffer address of block to transfer
cnt (INT32, input): size of block in 32-bit words

4) VFI_CYCLE_READ_WORD (bfadd)

Description: Does a single word data read cycle. Bus mastership and primary address cycles must be completed before using this routine.

Parameters:
bfadd (INT32, input): address of word to transfer.

8 ERROR REPORTING ROUTINES

For description of possible error codes, see the document detailing the VME/FB interface. A returned code of hex 80 indicates success.

1) VFI_DISPLAY_ERROR (error)
   VFDERR

   Description: Returns and displays the status code of the last completed operation.

   Parameters:
   error (INT32, output): The returned error code.

2) VFI_GET_ERROR (error)
   VFGERR

   Description: Same as vfi_display_error except that the routine does not display any error message.

   Parameters:
   error (INT32, output): The returned error code.

9 REFERENCES

[1] Berners-Lee, T. et al. The VALET-PLUS, a VMEbus Microcoputer for Physics Applications. Fith conference on Real Time Computer Applications in Nuclear, Particle and Plasma Physics- San Francisco, May 1987

[2] Gustafsson, L. A VME to Fastbus Interface using a Finite State Machine Coprocessor. FNAL internal report- September 1985.