Fermi National Accelerator Laboratory

TM-1477

VTI
VME/CIPRICO Interface Routines

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

September 4, 1987

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

Note Number XXXX

VTI
VME/CIPRICO Interface Routines

-Version-
Software:1.0
Document:1.0

September 4, 1987

Dean Alleva
Development and Evaluation Group
RD/Computing
Fermilab

TABLE OF CONTENTS

1 INTRODUCTION

This document details the VME/CIPRICO Interface routines (VTI). These routines where designed to allow programs written in PILS running on a MVME 101 under Valet-plus to control a CIPRICO tape controler [1], [2]. The routines fall into two general types. The low level rountines, such as vti_make_pb, create the data structures used by the CIPRICO as well as manipulate the CIPRICO's control registers. The high level routines, such as vti_rewind, use the lower level routines to carry out complete funtions on tape drives. Most tape operations are implemented except for ring buffer record routines. The creation of ring buffer record lists linked to parameter blocks is possible, but no high level routines are implemented to work with these lists.

The routines are written in PILS, a high-level language similar to BASIC and Pascal. This language 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 CIPRICO.

This document is divided into six sections, the first is the introduction. The remaining sections detail memory utilization, parameter block creation and linking, polling and interrupts, status reporting, and the VTI routines.

It is assumed that the reader is familiar with VME and the CIPRICO controller as well as the linked list data structure. A copy of the VTI software is available on BitNet at Fermilab as "FNAL::USR$ROOT:[ALLEVA.PUBLIC]VTI.SRC".

2 MEMORY UTILIZATION

The CIPRICO requires the creation of parameter blocks in VME system memory. When the CIPRICO executes a command, it retrieves the command from a parameter block. Parameter blocks contain a link field which can be used to create long lists of blocks. These lists can be executed by the CIPRICO without intervention from the CPU.

Most VTI routines must be passed an address where they can create a parameter block. Some routines, like vti_config, require more then one address. These addresses are supplied by the programmer since the VTI routines and the VALET system do no memory managment. Until some form of memory management is available in the OS, it is up to the programmer to maintain memory and parameter block list address integrity. In this document, routine parameter names specifying VME system memory addresses end with "add", such as pbadd.

3 PARAMETER BLOCK CREATION AND LINKING

The routine vti_make_pb creates a parameter block at the supplied address. In non-link mode, each parameter block created has its link field cleared. To create a list of parameter blocks, a call to vti_start_pblink begins the link process. Each following call to vti_make_pb still creates a block at the supplied address, but links the last parameter block created to the new one. This is done by setting the last parameter block's link field to the address of the new parameter block. Thus, vti_make_pb adds, when in link mode, the new parameter block to the end of the block list. A call to vti_end_pblink stops the linking process by returning to non-link mode. Starting the process over creates a new list of parameter blocks. The link process is shown in the code fragment below:

```
      ...
100   vti_make_pb (pbadd1, ...) ! unlinked parameter block
      ...
200   vti_start_pblink          ! begin linking of blocks
210   vti_make_pb (pbadd2, ...) ! first record in list
220   vti_make_pb (pbadd3, ...) ! Second record in list
      ...
300   vti_end_pblink            ! end of linking
310   vti_go (pbadd2)           ! go with list
320   vti_make_pb (pbadd4, ...) ! unlinked parameter block
      ...
```

The addresses are specified by the variables pbaddx, were x is a number specifing a unique new address. A second way to make a linked list is to call the high level routines while in linked mode. In this way, the parameter blocks are not executed after the routines create them, but are linked with previous blocks. The created list can then be executed with a call to vti_go. This is shown in the following example:

```
      ...
100   vti_rewind (pbadd1, ...)   !  rewind tape
110   vti_start_pblink           !  begin link
120   vti_search (pbadd1, ...)   !  not executed yet
130   vti_read (pbadd2, ...)     !  linked after search
      ...
200   vti_end_pblink             ! stop link
210   vti_go (pbadd1)            ! starts with search, then read, ...
      ...
```

(NOTE: It is now possible to link parameter blocks, but the interrupt handler is too simple to process the list (that is, to handle multiple interrupts from one command). Because of the primitive nature of the "background" processing under Valet-plus, it is best to wait for a more sophisticated OS to handle full CIPRICO support).

4 POLLING AND INTERRUPTS

Two methods are used to monitor the processing of parameter blocks. In polling mode, the gate field inside the executing parameter block is tested in a loop which is exited if the Command Complete bit is set. Polling should not be used when executing linked parameter blocks.

When interrupt mode is used, a loop is entered until a interrupt flag is set by an interrupt handler. The handler is executed upon an interrupt.There are two handlers present, one for errors and one for success. This type of interrupt handling is not much different then polling, but Valet-plus has no "wait for interrupt" command. In a multitasking environment, a wait for interrupt command would enable the OS to block the waiting processes and execute other processes.

5 STATUS REPORTING

Once a command, or list of commands, has been executed, the status may be fetched in any of two ways. First, a call to vti_display_error will return the status code of the last command executed. For linked parameter blocks, the status returned is either success or the error code for the error which halted list execution. Vti_display_error also prints to the display device a message indicating the error code returned.

Status may also be fetched with a call to vti_get_error, which returns status without displaying any messages. Note that these routines should be called after a parameter block or parameter block list is executed.

Originally, status reporting was to be done in the interrupt handler routines and returned to the main program through a parameter in the command 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 ROUTINES

6.1 VTI Initialize

1) VTI_INITIALIZE (ciprico_add, add_mod, inter)
   VTINIT

   Description: This routine initializes internal VTI data values. It should be called once before any calls to other VTI routines.

   Parameters:

ciprico_add	(INT32, input):	VME address of CIPRICO's Attention Register.
add_mod	(INT32, input):	Address modifier for use by the CIPRICO to access system memory.
inter	(INT32, input):	If set to 1, VTI uses interrupts to monitor the CIPRICO. If set to 0, VTI uses polling.

6.2 Parameter Block Initialization

The vti_set_X routines set the field specified by X in a parameter block. For example, vti_set_control sets the control field. These routines are usually not used by a program, they exsist for use by vti_make_pb.

1) VTI_MAKE_PB (pbadd, com, skp, rev, unt, sp, bc)
   VTMPB

   Desciption: This routine creates a parameter block at the supplied VME system memory address.

   Parameters:

pbadd	(INT32, input):	address where new parameter block is to be created.
com	(INT32, input):	the command code to be placed in the command field.
skp	(INT32, input):	the skip bit value (1 or 0)
rev	(INT32, input):	the reverse bit value (1 or 0)
unt	(INT32, input):	the unit number (0 to 7)
sp	(INT32, input):	source/destination pointer value
bc	(INT32, input):	byte count value

6.3 Parameter Block Linking

1) VTI_START_PBLINK
   VTSPL

   Description: Begin the linking of parameter blocks. Any calls to vti_make_pb after a call to vti_start_pblink will link the new parameter block at the end of the list of parameter blocks.

2) VTI_END_PBLINK
   VTEPL

   Description: Ends the linking of parameter blocks. Any calls to vti_make_pb after a call to vti_end_pblink will not link the parameter block to the list. Once this procedure is called the linked list is closed to all further links. The procedure should be called before another call to vti_start_pblink or a call to vti_go with the list as a parameter..

6.4 Parameter Block Execution

1) VTI_GO (pbadd)
   VTGO

   Description: Executes the parameter block at address pbadd. This procedure does either polling or interrupts when waiting for the CIPRICO to complete the command (or list of commands). Polling or interrupt monitoring is selected with the routine VTI_INITAILIZE.

   Parameters:

   pbadd (INT32, input): address of parameter block (or address of first parameter block in list of parameter blocks) to be executed.

6.5 Ring Buffer Record Initialize

1) VTI_MAKE_RB (rbadd, bcnt, sdptr)
   VTMRB

   Description: Create a ring buffer record at the address rbadd.

   Parameters:

rbadd	(INT32, input):	address where new ring buffer is to be created.
bcnt	(INT32, input):	byte count field value
sdptr	(INT32, input):	source/destination pointer

6.6 Ring Buffer Record Linking

1) VTI_START_RBLINK
   VTSRL

   Description: Starts the linking of ring buffer records. This routine functions the same as vti_start_pblink except that it works with ring buffer records.

2) VTI_END_RBLINK
   VTERL

   Description: Works the same as VTI_END_PBLINK, except the procedure functions with ring buffer records.

6.7 Error Reporting

1) VTI_DISPLAY_ERROR (error)
   VTDERR

   Description: Returns and displays the status code of the last completed operation. In linked parameter blocks the status returned is either success (0) or the code of the error which stoped execution of the list.

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

2) VTI_GET_ERROR (error)
   VTGERR

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

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

6.7.1 Update Status -

1) VTI_READ_STATUS (pbadd, unit)
   VTRST

   Description: Updates the two status bytes in the parameter block (at pbadd) reflecting the current status of the selected drive.

   Parameters:
   pbadd (INT32, input): VME memory address for the returned parameter block.
   unit (INT32, input): The drive number

6.8 Command Routines

6.8.1 CIPRICO Configuration And Firmware Inforamtion -

1) VTI_CONFIGURE (pbadd, cfadd, bcont, tc1, sig, throt, retry, tc2)
   VTCON

   Description: Configures the CIPRICO tape controler.

   Parameters:

pbadd	(INT32, input):	address where a parameter block can be created.
cfadd	(INT32, input):	address where a temporary configuration block can be created.
bcont	(INT32, input):	Bus control
tc1	(INT32, input):	Tape control 1
sig	(INT32, input):	Signals
throt	(INT32, input):	Throttle
retry	(INT32, input):	Retry value
tc2	(INT32, input):	Tape control 2

2) VTI_READ_CONFIG (pbadd, cfadd)
   VTRCON

   Description: Returns the configuration of the controler in a configuration block.

   Parameters:

pbadd	(INT32, input):	VME address where a parameter block is created.
cfadd	(INT32, input):	VME address where configuration block will be created.

3) VTI_READ_ID (pbadd, firmwara_id)
   VTRID

   Description: Returns the controler's firmware ID.

   Parameters:

pbadd	(INT32, input):	VME address where a parameter block is created.
firmware_id	(INT32, output):	Returned ID

4) VTI_READ_DATE (pbadd, firmware_date)
   VTRDATE

   Description: Return the controler's firmware date.

   Parameters:

pbadd	(INT32, input):	VME address where routine creates a parameter block.
firmware_date	(INT32, output):	Returned date

6.8.2 Tape Rewind -

1) VTI_REWIND (pbadd, unit)
   VTREW

   Description: Rewinds the tape drive specified by the given unit number.

   Parameters:
   pbadd (INT32, input): address where a temporary parameter block can be created.
   unit (INT32, input): specifies the tape drive to rewind.

2) VTI_OVER_REWIND (pbadd, unit)
   VTOREW

   Description: Does a overlaped rewind on given unit.

   Parameters:
   pbadd (INT32, input): VME address for the creation of the parameter block.
   unit (INT32, input): Drive number

6.8.3 Tape Read And Write -

1) VTI_READ (pbadd, buffadd, numbytes, unit)
   VTRD

   Description: Reads a number of bytes from a tape to a memory buffer.

   Parameter:

pbadd	(INT32, input):	VME system memory address where a parameter block is created.
buffadd	(INT32, input):	VME address of system memory where data buffer is located.
numbytes	(INT32, input):	Number of bytes to transfer from data buffer.
unit	(INT32, input):	Tape drive number to do read from.

2) VTI_WRITE (pbadd, buffadd, numbytes, unit)
   VTWT

   Description: Write a number of bytes from a buffer to tape.

   Parameters:

pbadd	(INT32, input):	address where a temporary parameter block is created.
buffadd	(INT32, input):	VME address of system memory where data buffer is located.
numbytes	(INT32, input):	Number of bytes to transfer to data buffer.
unit	(INT32, input):	Tape drive number to do write to.

6.8.4 Putting A Drive Offline -

1) VTI_OFF_LINE (pbadd, unit)
   VTOFF

   Description: Rewinds the tape drive specified by unit and sets it offline.

   Parameters:
   pbadd (INT32, input): VME address where a temporary parameter block is created.
   unit (INT32, input): Tape drive number to set offline.

6.8.5 Drive Reset -

1) VTI_RESET_DRIVES (pbadd)
   VTRDRV

   Description: Resets all drives connected to controler.

   Parameters:
   pbadd (INT32, input): VME memory address where a parameter block can created.

6.8.6 Tape Erase -

1) VTI_FIXED_ERASE (pbadd, unit)
   VTFER

   Description: Does a fixed erase on the given unit.

   Parameters:
   pbadd (INT32, input): VME memory address for the creation of the parameter block.
   unit (INT32, input): Drive number

2) VTI_VAR_ERASE (pbadd, unit)
   VTVER

   Description: Does a variable length erase to end of tape on the given unit.

   Parameters:
   pbadd (INT32, input): VME memory address used to create a parameter block.
   unit (INT32, input): Drive number

6.8.7 Marking Files And Searching Files -

1) VTI_MARK_FILE (pbadd, unit)
VTMFL

Description: Marks a end of file on the given unit.

Parameters:
pbadd (INT32, input): VME memory address for parameter block.
unit (INT32, input): Drive number

2) VTI_SEARCH (pbadd, file_marks, unit)
VTSRCH

Description: Does a forward search a given number of file marks on the given unit.
Parameters:
pbadd (INT32, input): VME address for parameter block.
unit (INT32, input): Drive number

3) VTI_SEARCH_MULT (pbadd, file_marks, direction, unit)
VTSRCHM

Description: Does a search, in either direction, a given number of file marks on a given unit.

Parameters:

pbadd	(INT32, input):	VME system memory address for creation of a parameter block.
file_marks	(INT32, input):	Number of file marks to search.
Direction	(INT32, input):	Direction of search. (1 = Change direction)
unit	(INT32, input):	Drive number

6.8.8 Searching Records -

1) VTI_SPACE_REC (pbadd, rec_num, unit)
VTSREC

Description: Spaces a given number of records on the given unit.

Parameters:

pbadd	(INT32, input):	VME memory address for parameter block.
rec_num	(INT32, input):	Number of records to scan over.
unit	(INT32, input):	Drive number

6.8.9 CIPRICO Diagnostics -

1) VTI_DIAG_1 (pbadd) and VTI_DIAG_2 (pbadd)
   VTDIA1 and VTDIA2

   Description: These routines do diagnostic tests on the CIPRICO. For diagnostic test 2, the CIPRICO must be configured with the diagnostic bit in the bus control by high.

   Parameters (INT32, input): VME system memory address for creation of a parameter block.

7 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] Tapemaster 3000 VMEbus tape drive controler, Ciprico Product Specification. Publication Number 21014000, Revision A- March 1987.