



Fermi National Accelerator Laboratory

TM-1539

VII
VME/IORFI-II Interface Routines

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

July 7, 1988



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

VII
VME/IORFI-II Interface Routines

-Version-
Software:1.0
Document:1.0

July 7, 1988

Dean Alleva
Development and Evaluation Group
RD/Computing
Fermilab

TABLE OF CONTENTS

1	INTRODUCTION	3
2	PILS AND VALET-PLUS.....	4
3	VII FILES	5
4	VII PARAMETERS	6
5	STATUS CODES	7
6	THE TRANSACTION ROUTINES	8
6.1	Initialization and Control Parameters	8
6.2	Read and Write Data	12
6.2.1	Single Word Transfers	12
6.2.3	Block Transfers	14
6.3	Read and Write CSR	15
6.3.1	Single Word Transfers	15
6.3.2	Block Transfers	17
6.4	Single Cycle Operations	18
6.4.1	Arbitration Cycle	18
6.4.2	Bus Release	18
6.4.3	Bus Disconnect	18
6.4.4	Primary Address Cycle	19
6.4.5	Secondary Address Cycle	20
6.4.6	Data Cycle	21
7	STATUS REPORTING ROUTINES	22
7.1	Status messages	23
8	REFERENCES	24

1 INTRODUCTION

This document describes the VME/IORFI-II Interface routines (VII). These routines were designed to meet two needs. First, the routines enable programs written in PILS running on a MVME 101 under Valet-Plus to control an IORFI-II interface from VME [1], [2]. Secondly, the routines provide a high level language version of the FASTBUS standard routines for the IORFI-II which can easily be translated into other high level languages (like C). The routines fall into two general types, control and transaction. The control routines work directly with the interface registers. These routines set up and monitor operations between VME and the IORFI-II. The control routines are usually used indirectly by the programmer through the transaction routines. The transaction routines, such as VII_WRITE_DAT, use the control routines to carry out complete functions on FASTBUS. Most FASTBUS operations have been implemented except for the compound routines and some low level routines.

To facilitate access to the IORFI-II registers from VME, a Super-VIOR DMA board was used as a set of I/O registers tied to the IORFI-II's front panel connectors [3]. The DMA controller on the Super-VIOR was not used and a much simpler board (only containing a set of four registers accessible from VME) could replace the Super-VIOR.

The routines are written in PILS, a high-level language similar to BASIC and Pascal which is powerful and fast enough for most applications [1]. However, PILS has proven to be too slow for efficient control of the IORFI-II. This is discussed further in the next section along with a discussion of possible future implementations of the routines.

This document is divided into eight sections, the first being the introduction. The remaining sections detail the PILS and Valet-Plus system, the VII files, the VII parameters, error code design, FASTBUS transaction routines, error reporting routines and finally a reference section.

It is assumed that the reader is familiar with VME, FASTBUS, and has some knowledge of the IORFI-II [2]. A copy of the VII software is available on BitNet at Fermilab as

```
"FNAL::USR$ROOT:[ALLEVA.PUBLIC]VII.DEF"
"FNAL::USR$ROOT:[ALLEVA.PUBLIC]VII.CNT"
"FNAL::USR$ROOT:[ALLEVA.PUBLIC]VII.LBR"
```

These files are detailed in section three (3).

2 PILS AND VALET-PLUS

The VII routines were written in PILS and executed on a 8 MHz Motorola 68000 processor [4]. The Valet/PILS system has proven to be adequate in performance for several other projects but falls short when controlling the IORFI-II. The IORFI-II interface requires a high level of program control and monitoring and its performance is directly dependent on program performance. Transfer time per 32-bit word on FASTBUS during block transfers was greater than 100 microseconds. This is about 4 times faster than already existing software on other machines.

To improve speed, some of the VII code was rewritten in machine code and executed on a 16 MHz Motorola 68020 processor [5]. Performance increased to 8 microseconds per 32-bit transfer. Since this software has not exceeded the system bandwidth, further speed increases could be gained with faster processors. Future implementations of the VII routines will be coded in C giving much higher performance than PILS and allowing the code to be easily transferred to faster processors. Transfer rates of 1 megabyte per second seem to be possible without any changes to the IORFI-II.

3 VII FILES

The VII code is divided into three files. The first, VII.DEF, contains definitions for constants used by the VII routines. These constants include the IORFI-II register bitmaps and function definitions as well as the VII status codes. All of the constants are accessible to user programs.

The second file, VII.CNT, contains the control routines which set up and maintain operations on FASTBUS. These routines supply the necessary "intelligence" for the control of the IORFI-II during FASTBUS arbitration cycles, address cycles, single word transfers, block transfers and when releasing the bus. The control routines are accessed by higher level library routines and are not designed to be accessed directly by user programs.

The final file, VII.LIB, contains the FASTBUS standard routine library. These routines are supplied in two forms, full name and short name. Most of the standard routines are defined except for the compound routines and FASTBUS signal line access routines. Most of the unimplemented routines, however, can be built from the existing routines.

All of the above files must be loaded into the PILS system in the order shown below.

```
> load getput.lib ! File that defines get's and put's
                  ! for VME memory access.
> load vii.def
> load vii.cnt
> load vii.lib
> compile
```

4 VII PARAMETERS

A user program may set VII parameters during execution to enable or disable features or set control values. Below is a list of the VII parameters, the routines which set them and what each parameter does. Note that each parameter has its own associated routine and that some of them can also be set by VII_INITIALIZE.

Parameter	Routine	Default	Description
GK timer	VII SET GK TIMER or VII_INITIALIZE	0 (disabled)	Set the GK timeout value
AK timer	VII SET AK TIMER or VII_INITIALIZE	0 (disabled)	Set the AK timeout value
DK timer	VII SET DK TIMER or VII_INITIALIZE	0 (disabled)	Set the DK timeout value
retry count	VII SET RETRY COUNT or VII_INITIALIZE	0	Set the retry count for failed operations.
arbitration vector	VII SET ARB VECTOR or VII_INITIALIZE	0	Set arbitration vector for FASTBUS arbitration cycles.
enable AI	VII_ENABLE_AI	disabled	Causes the IORFI-II to follow the assured access protocol.
enable EG	VII_ENABLE_EG	enabled	Causes the IORFI-II to assert EG during address cycles.
enable parity	VII_ENABLE_PARITY	disabled	Enable parity generation and compare.
enable warnings	VII_ENABLE_WARN	enabled	Enable warning status messages to be treated as errors.
release bus	VII_SET_REL_BUS	enabled	Release bus, lower GK, after complete transaction.
disconnect	VII_SET_DISC	disabled	Release AS/AK lock after complete transaction. Leave GK high.
release on error	VII_SET_REL_ERROR	enabled	Release bus, lower GK, on error (after retries).

All of the above mentioned routines are described in the section detailing with the VII routines.

5 STATUS CODES

All VII transaction routines are functions which return status codes. These codes indicate to the user program the completion status of the given transaction routine. To facilitate greater flexibility, the status codes have been implemented to allow for different severities. Four severity levels are supported: fatal, error, warning and informational. Status codes of fatal severity terminate the transaction routine even if further retries are possible. Status codes of error severity terminate the transaction routine only after the retry count is exhausted. Warning status codes can be treated in two ways. If warnings are enabled (see VII_ENABLE_WARN above), warning codes are treated as errors. If warnings are disabled, warnings are treated as informational messages. Information messages are reported but have no effect on program execution. The format of a status code is shown below. The high nibble (bits 28-31) contain the severity level, the remaining part of the word contains the status code.

```

Status word: [ severity |      status code      ]
              bits: 31 - 28|27                -      0

```

```

Severity codes:   HEX 8xxxxxxx   - Fatal
                  HEX 4xxxxxxx   - Error
                  HEX 2xxxxxxx   - Warning
                  HEX 1xxxxxxx   - Informational

```

Where xxxxxx is the status code unique to each message.

Status codes may be processed either by the user program or by passing the code to VII_STATUS. This routine first displays a corresponding status message to the display device. The routine then terminates execution of the program if the code is other than an informational status code or a warning status code with warnings turned off. If error codes are to be processed by the user program, VII_ERROR MESSAGE may be called with the status code instead of VII_STATUS. This routine will display the status message but, unlike VII_STATUS, will not terminate execution of the program. For a list of the status codes and messages see section 7.1 .

6 THE ROUTINES

There are two types of routines detailed here. The first section details various VII initialization routines. The remaining sections detail routines that do complete operations on FASTBUS from VME. These routines are the basic building block from which more complex routines can be built. The FASTBUS transaction routines are all functions which return status codes.

6.1 Initialization And Control Parameters

These routines initialize the operational parameters (see section 4) and reset the IORFI-II interface. These routines are subroutines and do not return any status codes.

1) VII_INITIALIZE (svior_add, gk_to, ak_to, dk_to, retry, arb)

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

Parameters:

svior_add	(INT32, input):	VME address of Super-VIOR interface.
gk_to	(INT32, input):	GK timeout value. For values greater than one (1), the GK timer is enabled. For each increment of gk_to, VII will look for GK an additional 50 microseconds. If no response takes place in the allotted time, a GK timeout error is posted. If gk_to is set to a value less than one (1), GK timeouts are disabled.
ak_to	(INT32, input):	Same as gk_to but for AK responses.
dk_to	(INT32, input):	Same as gk_to but for DK responses.
retry	(INT32, input):	This is the retry count. When set to a value greater than zero (0) the VII routines will retry a failed operation retry times.
arb	(INT32, input):	This is the arbitration vector level to be used by the IORFI-II during arbitration on FASTBUS.

2) VII_RESET

Description: Resets the IORFI-II interface and returns the VII operational parameters to their default state.

3) VII_SET_GK_TIMER (gk_timer_value)

Description: Sets the internal GK timeout value. If gk_timer_value less than one (1), GK timeouts are disabled. See VII_INITIALIZE above for a description of the GK timer values.

Parameters:

gk_timer_value (INT32, input): GK timer value, see VII_INITIALIZE.

4) VII_SET_AK_TIMER (ak_timer_value)

Description: Sets the internal AK timeout value. If ak_timer_value less than one (1), AK timeouts are disabled. See VII_INITIALIZE above for a description of the AK timer values.

Parameters:

ak_timer_value (INT32, input): AK timer value, see VII_INITIALIZE.

5) VII_SET_DK_TIMER (dk_timer_value)

Description: Sets the internal DK timeout value. If dk_timer_value less than one (1), DK timeouts are disabled. See VII_INITIALIZE above for a description of the DK timer values.

Parameters:

dk_timer_value (INT32, input): DK timer value, see VII_INITIALIZE.

6) VII_SET_RETRY_COUNT (retry_value)

Description: Sets the retry count for failed operations.

Parameters:

retry_count (INT32, input): Retry count value.

7) VII_SET_ARB_VECTOR (arb_vector_value)

Description: Set the arbitration vector level to be used by the IORFI-II during arbitration cycles on FASTBUS.

Parameters:

arb_vector_value (INT32, input): arbitration vector level value.

The following routines set or clear various VII parameters (see section 4). The routines are each passed a parameter called val. The routines corresponding parameter is set if val is greater than zero (0), cleared if val is less than or equal to zero (0).

8) VII_ENABLE_AI (val)

Description: Enable or disable assured access protocol during arbitration cycles on FASTBUS.

Parameters:

val (INT32, input): sets/clears the AI enable flag.

9) VII_ENABLE_EG (val)

Description: Enable or disable the EG line assertion by the IORFI-II during address cycles on FASTBUS.

Parameters:

val (INT32, input): sets/clears the EG enable flag.

10) VII_ENABLE_PARITY (val)

Description: Enable or disable parity generation (during FASTBUS writes) and parity compare (during FASTBUS reads).

Parameters:

val (INT32, input): sets/clears the parity enable flag.

11) VII_ENABLE_WARN (val)

Description: Enable or disables warning messages to be treated as error messages. If disabled, warning messages are displayed but do not cause a program to abort. See section 7.2 for details on status codes.

Parameters:

val (INT32, input): sets/clears the warning enable flag.

12) VII_SET_REL_BUS (val)

Description: Sets or clears the release bus flag. GK will be lowered on exiting a transaction routine, not a primitive routine (such as VII_CYCLE_PA).

Parameters:

val (INT32, input): sets/clears the release bus flag.

13) VII_SET_DISC (val)

Description: Sets or clears the disconnect flag. AS/AK lock is released on exiting a transaction routine, not a primitive routine (such as VII_CYCLE_PA).

Parameters:

val (INT32, input): sets/clears the disconnect flag.

14) VII_SET_REL_ERROR (val)

Description: Sets/clears the release bus on error flag. GK is lowered on exiting a transaction routine, not a primitive routine (such as VII_CYCLE_PA), if the routine exits with an error.

Parameters:

val (INT32, input): sets/clears the release bus on error flag.

6.2 Read And Write Data

3.2.1 Single Word Transfers -

- 1) VII READ DAT (rword, pradd, scadd)
VIIRD (rword, pradd, scadd)

Description: Does a single word transfer from FASTBUS data space.

Parameters:

rword (INT32, output): returned 32-bit value read from FASTBUS.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 2) VII WRITE DAT (wword, pradd, scadd)
VIIWD (wword, pradd, scadd)

Description: Does a single word transfer into FASTBUS data space.

Parameters:

wword (INT32, output): the 32-bit word to be written.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 3) VII READ DAT MULT (rword, pradd, scadd)
VIIRD \bar{M} (rword, pradd, scadd)

Description: Does a broadcast single from FASTBUS data space.

Parameters:

rword (INT32, output): returned 32-bit value read from FASTBUS.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 4) VII WRITE DAT MULT (wword, pradd, scadd)
VIIWD \bar{M} (wword, pradd, scadd)

Description: Does a broadcast single word transfer into FASTBUS data space.

Parameters:

wword (INT32, output): the 32-bit word to be written.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 5) VII READ DAT SA (rword, pradd)
VIIRD $\overline{\text{SA}}$ (rword, pradd)

Description: Does a single word transfer from the NTA register in FASTBUS data space.

Parameters:

rword (INT32, output): returned 32-bit value read from FASTBUS.
pradd (INT32, input): Primary address.

- 6) VII WRITE DAT SA (wword, pradd)
VIIW $\overline{\text{SA}}$ (wword, pradd)

Description: Does a single word transfer to the NTA register in FASTBUS data space.

Parameters:

wword (INT32, output): the 32-bit word to be written.
pradd (INT32, input): Primary address.

6.2.2 Block Transfers -

- 1) VII READ DAT BLOCK (bfadd, wct, pradd, scadd)
 VIIRDB (bfadd, wct, pradd, scadd)

Description: Does a block transfer to VME memory from FASTBUS data space.

Parameters:

pradd (INT32, input): Primary address.
 scadd (INT32, input): Secondary address.
 bfadd (INT32, input): VME memory address where block is
 transferred.
 wct (INT32, input): Number of 32-bit words to transfer.

- 2) VII WRITE DAT BLOCK (bfadd, wct, pradd, scadd)
 VIIWDB (bfadd, wct, pradd, scadd)

Description: Does a block transfer from VME memory into FASTBUS data space.

Parameters:

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

- 3) VII READ DAT BLOCK MULT (bfadd, wct, pradd, scadd)
 VIIRDBM (bfadd, wct, pradd, scadd)

Description: Does a broadcast block transfer to the VME memory from
 FASTBUS data space.

Parameters:

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

- 4) VII WRITE DAT BLOCK MULT (bfadd, wct, pradd, scadd)
 VIIWDBM (bfadd, wct, pradd, scadd)

Description: Does a broadcast block transfer from VME memory into
 FASTBUS data space.

Parameters:

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

6.3 Read And Write CSR

6.3.1 Single Word Transfers -

- 1) VII READ_CSR (rword, pradd, scadd)
VIIRC (rword, pradd, scadd)

Description: Does a single word transfer from FASTBUS control space.

Parameters:

rword (INT32, output): returned 32-bit value read from FASTBUS.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 2) VII WRITE_CSR (wword, pradd, scadd)
VIIWC (wword, pradd, scadd)

Description: Does a single word transfer into FASTBUS control space.

Parameters:

wword (INT32, output): the 32-bit word to be written.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 3) VII READ_CSR_MULT (rword, pradd, scadd)
VIIRCM (rword, pradd, scadd)

Description: Does a broadcast single from FASTBUS control space.

Parameters:

rword (INT32, output): returned 32-bit value read from FASTBUS.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 4) VII WRITE_CSR_MULT (wword, pradd, scadd)
VIIWCM (wword, pradd, scadd)

Description: Does a broadcast single word transfer into FASTBUS control space.

Parameters:

wword (INT32, output): the 32-bit word to be written.
pradd (INT32, input): Primary address.
scadd (INT32, input): Secondary address.

- 5) VII READ CSR SA (rword, pradd)
VII \overline{R} CSA (\overline{rword} , pradd)

Description: Does a single word transfer from the NTA register in FASTBUS control space.

Parameters:

rword (INT32, output): returned 32-bit value read from FASTBUS.
pradd (INT32, input): Primary address.

- 6) VII WRITE CSR SA (wword, pradd)
VII \overline{W} CSA (\overline{wword} , pradd)

Description: Does a single word transfer to the NTA register in FASTBUS control space.

Parameters:

wword (INT32, output): the 32-bit word to be written.
pradd (INT32, input): Primary address.

6.3.2 Block Transfers -

- 1) VII READ CSR BLOCK (bfadd, wct, pradd, scadd)
 VII \overline{R} CB (\overline{b} fadd, wct, pradd, scadd)

Description: Does a block transfer to VME memory from FASTBUS control space.

Parameters:

pradd (INT32, input): Primary address.
 scadd (INT32, input): Secondary address.
 bfadd (INT32, input): VME memory address where block is transferred.
 wct (INT32, input): Number of 32-bit words to transfer.

- 2) VII WRITE CSR BLOCK (bfadd, wct, pradd, scadd)
 VII \overline{W} CB (\overline{b} fadd, wct, pradd, scadd)

Description: Does a block transfer from VME memory into FASTBUS control space.

Parameters:

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

- 3) VII READ CSR BLOCK MULT (bfadd, wct, pradd, scadd)
 VII \overline{R} CBM (\overline{b} fadd, wct, pradd, scadd)

Description: Does a broadcast block transfer to the VME memory from FASTBUS control space.

Parameters:

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

- 4) VII WRITE CSR BLOCK MULT (bfadd, wct, pradd, scadd)
 VII \overline{W} CBM (\overline{b} fadd, wct, pradd, scadd)

Description: Does a broadcast block transfer from VME memory into FASTBUS control space.

Parameters:

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

6.4 Single Cycle Operations

6.4.1 Arbitration Cycle -

- 1) VII_CYCLE_ARBITRATE
VIIARB

Description: Does an arbitration cycle, holding onto the bus when control is gained.

6.4.2 Bus Release -

- 1) VII_CYCLE_RELEASE_BUS
VIIREL

Description: Releases the bus by lowering GK and AS.

6.4.3 Bus Disconnect -

- 1) VII_CYCLE_DISCONNECT
VIIDISC

Description: Releases any AS/AK lock but leaves GK high.

6.4.4 Primary Address Cycle -

- 1) VII_CYCLE_PA_DAT (pradd)
VIIPD (pradd)

Description: Does a primary address cycle to data space.

Parameters:

pradd (INT32, input): The primary address.

- 2) VII_CYCLE_PA_CSR (pradd)
VIIPC, (pradd)

Description: Does a primary address cycle to CSR space.

Parameters:

pradd (INT32, input): The primary address.

- 3) VII_CYCLE_PA_DAT_MULT (pradd)
VIIPDM (pradd)

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

Parameters:

pradd (INT32, input): The primary address.

- 4) VII_CYCLE_PA_CSR_MULT (pradd)
VIIPCM (pradd)

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

Parameters:

pradd (INT32, input): The primary address.

6.4.5 Secondary Address Cycle -

- 1) VII_CYCLE_READ_SA (rword)
VIIRSA (rword)

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

Parameters:

rword (INT32, output): Returned word from read.

- 2) VII_CYCLE_WRITE_SA (wword)
VIIWSA (wword)

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

Parameters:

wword (INT32, input): Word to be written to NTA.

6.4.6 Data Cycle -

- 1) VII_CYCLE_READ_WORD (rword)
VIIRW (rword)

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

Parameters:

rword (INT32, output): word transferred.

- 2) VII_CYCLE_WRITE_WORD (wword)
VIIWW (wword)

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

Parameters:

wword (INT32, input): word to transfer.

- 3) VII_CYCLE_READ_BLOCK (bfadd, cnt)
VIIRB (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): VME buffer address of block buffer.

cnt (INT32, input): size of block in 32-bit words.

- 4) VII_CYCLE_WRITE_BLOCK (bfadd, cnt)
VIIRW (bfadd, cnt)

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

Parameters:

bfadd (INT32, input): VME buffer address of block to transfer.

cnt (INT32, input): size of block in 32-bit words.

7 STATUS REPORTING ROUTINES

1) VII_STATUS (status_code)

Description: This routine is called with the status code returned from one of the transaction or primitive routines. It displays a message to the display device indicating the type of status code. The routine also terminates program execution if the status code severity level was other than informational. Warning status codes are treated as informational messages if warnings are disabled.

Parameters:

status_code (INT32, input): status code to be processed.

2) VII_ERROR_MESSAGE (status_code)

Description: Displays the a message to the display device corressponding to status_code.

Parameters:

status_coce (INT32, input): status code to be displayed.

7.1 Status Messages

Status Constant	Severity	Code(hex)	Description
-----	-----	-----	-----
C_VIIS_SUCCESS	-	1	Operation success
C_VIIS_GK_TIMEOUT	error	40000011	GK timeout
C_VIIS_AK_TIMEOUT	error	40000012	AK timeout
C_VIIS_DK_TIMEOUT	error	40000013	DK timeout
C_VIIS_NO_AK_RELEASE	fatal	80000014	No AK(D) with AS(D)
C_VIIS_NO_DK_RELEASE	fatal	80000015	No DK(D) with AS(D)
C_VIIS_NET_BUSY	error	40000021	Network busy (address SS=1)
C_VIIS_NET_FAIL	fatal	80000022	Network failure (address SS=2)
C_VIIS_NET_ABORT	error	40000023	Network abort (address SS=3)
C_VIIS_SS_4A	fatal	80000024	Reserved address SS response 4
C_VIIS_SS_5A	fatal	80000025	Reserved address SS response 5
C_VIIS_SS_6A	fatal	80000026	Reserved address SS response 6
C_VIIS_INVALID_AI	error	40000027	AI invalid (SS=7)
C_VIIS_BUSY	error	40000031	Device Busy (data SS=1)
C_VIIS_EOB	error	40000032	End of block (data SS=2)
C_VIIS_UDEF	fatal	80000033	User defined data SS response 3
C_VIIS_SS_4D	fatal	80000034	Reserved data SS response 4
C_VIIS_SS_5D	fatal	80000035	Reserved data SS response 5
C_VIIS_DERR_REJECT	error	40000036	Data error reject (data SS=6)
C_VIIS_DERR_ACCEPT	error	40000037	Data error accept (data SS=7)
C_VIIS_PARITY_ERROR	warning	2000FFFF	Parity error

8 REFERENCES

- [1] Berners-Lee, T. et al. The VALET-PLUS, a VMEbus Microcomputer for Physics Applications. Fifth conference on Real Time Computer Applications in Nuclear, Particle and Plasma Physics- San Francisco, May 1987.
- [2] C. Rotolo and S. Chappa, I/O Register To FASTBUS Interface II, Fermilab Note FBN01, January 1983.
- [3] SUPER-VIOR, VMEbus Dual 16-bit Input/Output Register with Full DMA, hardware description, Opifex AB publication (Version 1.1).
- [4] MVME 101 MC68000 Monoboard Computer, user's manual, Motorola Microsystems, MVME101/D2, 1983.
- [5] FIC 8230 Fast Intelligent Control, user's manual, Creative Electronic Systems, 1987.