

# STAR BEMC TDC VME Interface & Programming Reference

## Preliminary 5/21/01

### Overview

The STAR Barrel EMC Tower Data Collector is comprised of six 6U VME cards residing in a crate on the STAR platform. The six cards are 5 TDC Input Cards and 1 TDC Output Cards. Each Input Card has six fiber inputs from the BEMC FEE (HOTLink @ 2xRHIC) crates making a total of 30 input channels. The Output Card has 2 Glink fiber outputs, one to DAQ (Glink @ 30 MHz) and the other to the L2 (future upgrade: Glink @ 60 MHz). The Output card also has trigger input from a TCD (STAR standard 20 pin ribbon cable). As well the Output card has a TTL busy input from DAQ (coaxial Lemo). All cards communicate via the TDC bus, which connects in parallel all the 64 VME user pins on P2 on the back side of the VME crate. TDC Overall block diagram is shown online as figure TDC0v1:  
<http://www.cs.indiana.edu/hyplan/willie/star.html>

### Theory of Operation

#### Global Memory Map

Each TDC Input and Output Card have a 256 byte memory map to the VME bus in A16 (D16 D8EO) space. Each card has an 8 position DIP switch which sets the base address on any even 256 byte boundary. Although each card can be placed anywhere in the A16 space independent of the other cards, it is the intension to have the Output Card sit at the TDC base address and each input card sit at  $base+256*(N+1)$ , where N is the input card number from 0 to 5. For example, the resulting overall map would be (below) for a base address of 1000 hex:

VME A16 address (hex) Base = 1000 hex Set by switches	Card	Card base DIP Switches S2	Card number DIP switches S1
1000 – 10FF	TDC Output Card	00010000	N/A
1100 – 11FF	TDC Input Card 0	00010001	0000
1200 – 12FF	TDC Input Card 1	00010010	0001
1300 – 13FF	TDC Input Card 2	00010011	0010
1400 – 14FF	TDC Input Card 3	00010100	0011
1500 – 15FF	TDC Input Card 4	00010101	0100

Note: DIP switch bit = 0 is ON (closed), DIP switch bit = 1 is OFF (open)

Lowest switch number is LSB and highest MSB

All TDC cards support VME D16 (16 bit data transfer) and D8EO (8 bit data transfer on even or odd bytes), but the TDC input cards only have an internal 12 bit data bus, so the most significant 4 bits of the 16 bit word are ignored on write, and will be invalid on read. The output card has a 16 bit data bus so all bit are always valid on read. As per VME standard accessing the D8E (even byte address) is the most significant byte, and accessing the D8O (odd byte address) is the least significant byte. Accessing the VME 16 bit word on the even address will write/read a 16 or 12 bit register within the TDC. TDC cards will not respond to D32 VME accesses. All register can be read or written to one byte at a time to leave a 16 or 12 bit value in the register, but writing to some registers cause actions (other than storing data), and these actions will be repeated for each write, which may have undesirable side effects.

## Output Card Memory Map

TDC OUTPUT **PRODUCTION** CARD REGISTERS B = base address

VME address	Register (binary)	Register Write Action	Register Read Action
B+0 to B+1E			“.V.M.E.I.D.I.U.C.F.T.D.C.1.0.....” VME card ID in ASCII
B+0 to B+1E	00 ----	Maps to Input Card 0 Channel 0	Maps to Input Card 0 Channel 0
B+40	10 0000		Trigger word = Trigger FIFO status / Trigger command / DAQ command
B+42	10 0001		Token number “TT”
B+44	10 0010	FIFO L2 next	FIFO token number write
B+46	10 0011	FIFO DAQ next	FIFO trigger word write
B+48	10 0100		FIFO L2 token number
B+4A	10 0101		FIFO DAQ token number
B+4C	10 0110		FIFO L2 trigger word
B+4E	10 0111		FIFO DAQ trigger word
B+FC	111 1110	Program FPGA mask reg (lower/odd byte)	FPGA DONE status (lower/odd byte)
B+FE	111 1111	Program FPGA data reg (lower/odd byte) Set PROGRAM pins high/low (bit 10)	FPGA INIT status (lower/odd byte)

TDC OUTPUT **PROTOTYPE** CARD REGISTERS B = base address

VME address	Register (binary)	Register Write Action	Register Read Action
B+0 to B+1E	00 ----	Maps to Input Card 0 Channel 0	Maps to Input Card 0 Channel 0
B+20	01 0000	Global SEND DATA TEST	GBUS 11-0
B+22	01 0001		GBUS 23-12
B+24	01 0010		GBUS 35-24
B+26	01 0011	L2 Score ADDR => Write to RB reg	RB (read back) register
B+28	01 0100	Glink DFA data send	Glink RX data
B+2A	01 0101	Glink CFA data send	
B+2C	01 0110		Glink RX CAV count
B+2E	01 0111		Glink RX DAV count
B+30	01 1000	Glink RX FIFO increment read count	Glink RX FIFO data
B+32	01 1001	Glink RX FIFO clear counters	Glink RX FIFO status
B+34	01 1010		Glink RX FIFO write count
B+36	01 1011		Glink RX FIFO read count
B+40	10 0000		Trigger word = Trigger FIFO status / Trigger command / DAQ command
B+42	10 0001		Token number "TT"
B+44	10 0010	FIFO L2 next	FIFO token number write
B+46	10 0011	FIFO DAQ next	FIFO trigger word write
B+48	10 0100		FIFO L2 token number
B+4A	10 0101		FIFO DAQ token number
B+4C	10 0110		FIFO L2 trigger word
B+4E	10 0111		FIFO DAQ trigger word
B+50	10 1---		L2 SC Score Board read
B+60	11 0000		FIFO L2 write counter
	11 0001		FIFO DAQ write counter
	11 0010		FIFO L2 read counter
	11 0011		FIFO DAQ read counter

## Input Card Memory Map

TDC INPUT CARD CHANNELS B = base address (card DIP switches S2)

VME address	Example VME address range With base = 1100 hex	Input Channel Registers in each INRX FPGA (see table below)
B+00	1100 – 111F	Channel 0 registers
B+20	1120 – 113F	Channel 1 registers
B+40	1140 – 115F	Channel 2 registers
B+60	1160 – 117F	Channel 3 registers
B+80	1180 – 119F	Channel 4 registers
B+A0	11A0 – 11BF	Channel 5 registers
B+C0	11C0 – 11DF	Not Used
B+E0	11E0 – 11FF	Not Used

ONE CHANNEL TDC INPUT CARD REGISTERS BC = base address + channel \* 0x20

VME address	Reg	Register Write Action	Register Read Action
BC+00	0000	Write NEW TEST token number	Read TEST token number
BC+02	0001	Write NEW DAQ token number	Read DAQ token number
BC+04	0010	Write NEW L2 token number	Read L2 token number
BC+06	0011	Write NEW RXWRITE token number	Read RXWRITE token number
BC+08	0100	Read RAM @ TEST into TEST buffer	Read TEST word address counter
BC+0A	0101	Read RAM @ DAQ into HOLD buffer	Read DAQ word address counter
BC+0C	0110	Read RAM @ L2 into HOLD buffer	Read L2 word address counter
BC+0E	0111	Write TEST buffer into RAM @ TEST	Read RXWRITE word address counter
BC+10	1000	Write TEST buffer	Read TEST buffer
BC+1C	1110	FPGA MASK (lower/odd byte)	FPGA DONE status (lower/odd byte)
BC+1E	1111	FPGA DATA (lower/odd byte) Set PROGRAM pins high/low (bit 10)	FPGA INIT status (lower/odd byte)

## Input Card Register Description

Figure TDC11v1 shows where the VME accessible registers for each channel of each input card fit into the TDC Input Card data path. Each input card has 6 sets of these registers starting at the base address and repeating every 0x20 hex for each subsequent channel number. The TOKEN NUMBER registers (TEST, DAQ, L2, RXWRITE) behave like normal read/write registers, however writes cause the matching WORD ADDRESS COUNTER to reset to zero. The WORD ADDRESS COUNTER registers (TEST, DAQ, L2, RXWRITE) can be read normally, but writes do not set the register. Instead a write causes a RAM read or write cycle as specified above. This action will transfer data addressed by the specified TOKEN NUMBER and WORD ADDRESS COUNTER registers to or from the target or source, and the specified WORD ADDRESS COUNTER register will be incremented. The TEST BUFFER register behaves like a normal read/write register and has no direct effect on the RAM, but instead is temporary storage and holds values to be written or values read from the RAM.

Since the RAM on an Input Cards is not direct memory mapped, a protocol has to be used to access the memory. Each Input Card has six (one for each channel) 1 Meg x 12 bit memories that hold the 160 ADC data values and 4 header words (stored in 164 out of 256 memory locations) for each of 4096 tokens, which is a total a 9 Megabytes of memory for each Input Card (45 MB total for the BEMC TDC). Memory for each channel is accessed by pointers using the TEST TOKEN NUMBER, TEST WORD COUNTER, and TEST BUFFER registers specific to that Input Card channel. Below are algorithms for memory access.

### **Reading values from a TDC Input Card channel**

The following reads 164 values from the Input Card memory at token\_number into Data(). The write to READ\_RAM\_at\_TEST causes the memory to be read into the TEST BUFFER and the TEST WORD COUNTER to be incremented.

```
Main()
  VME_write_word( TEST_TOKEN, token_number) set TEST TOKEN and reset word counter
  For I=0 to 163
    VME_write_byte( READ_RAM_at_TEST, 0 ) trigger RAM read into test buffer
    Data(I)= 0xff & VME_read_word( TEST_BUFFER ) get data from card
```

### **Writing values to a TDC Input Card channel**

The following writes 164 values from Data() into the Input Card memory at token\_number. The write to WRITE\_RAM\_at\_TEST causes the TEST BUFFER memory to be written into RAM and the TEST WORD COUNTER to be incremented.

```
Main()
  VME_write_word( TEST_TOKEN, token_number) set TEST TOKEN and reset word counter
  For I=0 to 163
    VME_read_word( TEST_BUFFER, Data(I) ) send data to card
    VME_write_byte( WRITE_RAM_at_TEST, 0 ) trigger test buffer write to RAM
```

## FPGA Programming for Input & Output Cards

The FPGA Programming registers allow the VME host to program and re-program all the Xilinx FPGA's on the input card at any time. Since Xilinx FPGA's are RAM based they must be programmed each time the crate power is applied. Although in the future the TDC will self-program on power up, currently (5/20/01) it must be programmed via the VME host before it will function. Each input card has six INRX FPGA's (Xilinx 4010XL) and one IMUX FPGA (Xilinx 4010XL). The output card has one SCORE FPGA and one GLMUX FPGA. The six INRX FPGA's are all programmed with the exactly the same bit stream from one file, and the IMUX FPGA from another file.

Programming is accomplished by the use of 2 write registers and 2 read registers on each input and output card. The FPGA MASK register allows the VME host to selectively access one or more FPGA's for programming operations. The FPGA DATA register sends DIN and CCLK's to all FPGA's selected by the MASK register. The Enable PGM pins bit and MASK register allow the host to drive the PROGRAM pins lows on selected FPGA's. The FGPA INIT status register and FPGA DONE status register allow the host to verify that the operation was successful. A detailed description of the programming procedure and signal timing for each of the 5 pins (CCLK, DIN, PROGRAM, INIT, DONE) is given in the Xilinx Data book (also online at [www.xilinx.com](http://www.xilinx.com)), which may be helpful in understanding the TDC programming interface to these 4010XL Xilinx parts.

### Input Card Register Bitmap

Bit Number VME bus (odd byte)	FPGA MASK Program Register Address = B+1D	FPGA DATA Program Register Address = B+1F	FPGA DONE Status Register Address = B+1D	FPGA INIT Status Register Address = B+1F
0 LSB	INRX 0	INRX 0	INRX 0	INRX 0
1	INRX 1	INRX 1	INRX 1	INRX 1
2	INRX 2	INRX 2	INRX 2	INRX 2
3	INRX 3	INRX 3	INRX 3	INRX 3
4	INRX 4	INRX 4	INRX 4	INRX 4
5	INRX 5	INRX 5	INRX 5	INRX 5
6	IMUX	IMUX	IMUX	IMUX
7 MSB	Alternate IMUX	Alternate IMUX	Alternate IMUX	Alternate IMUX
Bit Number VME bus (even byte)	FPGA PGMB Program Register Address = B+1C			
8 LSB				
9				
10	Enable PGM pins			
11				

NOTE: the memory map duplicates these 2 read and 2 write registers for all 8 possible channel locations, so it is only necessary to access these registers at one channel to program all channels.

### Output Card Register Bitmap

Bit Number VME bus (odd byte)	FPGA MASK Program Register Address = B+FD	FPGA DATA Program Register Address = B+FF	FPGA DONE Status Register Address = B+FD	FPGA INIT Status Register Address = B+FF
0 LSB	SCORE	SCORE	SCORE	SCORE
1	GLMUX	GLMUX	GLMUX	GLMUX
Bit Number VME bus (even byte)	FPGA PGMB Program Register Address = B+FC			
8 LSB				
9				
10	Enable PGM pins			
11				

## FPGA Programming Algorithm for TDC Input Cards

After a reset of all FPGA's, the host needs to read 2 ASCII hex files, convert these files into bit streams and send these streams to the FPGA DATA register. This algorithm must be repeated for each Input Card to be programmed with values for FPGA\_MASK, FPGA\_PGMB, FPGA\_DONE, and FPGA\_INIT that map to the registers of a specific Input Card. The "inrx.hex" and "imux.hex" files are the same for all Input Cards. Below is the algorithm:

```
Main()
  VME_write_byte( FPGA_MASK, 0x00)      set MASK to all active
  VME_write_byte( FPGA_PGMB, 0x04)     turn on all PROGRAM pins
  Wait( 1 uS )                          wait 1 microsecond
  VME_write_byte( FPGA_PGMB, 0x00)     turn off all PROGRAM pins
  Wait( 5 mS )                          wait 5 milliseconds for FPGA to respond
  If ( 0 == 0x7F & VME_read_byte( FPGA_DONE ) ) &&
    ( 0x7F == 0x7f & VME_read_byte( FPGA_INIT ) )
    Then We are OK
    Else ERROR: FPGA's didn't all respond to PROGRAM pulse

  VME_write_byte( FPGA_MASK, 0xC0)     set MASK to all INRX's active
  While not at end of file "inrx.hex" (file1)      loop for all the characters in the input file1
    Token1 = get_char( file1 )
    Token2 = get_char( file1 )
    Nibble1 = convert_hex_digit_to_integer( Token1 )
    Nibble2 = convert_hex_digit_to_integer( Token2 )
    Byte = 0x10 * Nibble1 + Nibble2
    Send_byte_to_xilinx( Byte )        send data bits to all INRX's

  VME_write_byte( FPGA_MASK, 0x3F)     set MASK to IMUX active
  While not at end of file "imux.hex" (file2)      loop for all the characters in the input file2
    Token1 = get_char( file2 )
    Token2 = get_char( file2 )
    Nibble1 = convert_hex_digit_to_integer( Token1 )
    Nibble2 = convert_hex_digit_to_integer( Token2 )
    Byte = 0x10 * Nibble1 + Nibble2
    Send_byte_to_xilinx( Byte )        send data bits to IMUX

  If ( 0x7F == 0x7F & VME_read_byte( FPGA_DONE ) ) &&
    ( 0x7F == 0x7f & VME_read_byte( FPGA_INIT ) )
    Then We are OK
    Else ERROR: FPGA's didn't all program correctly

Send_byte_to_xilinx( Byte )            function definition
For I=0 to 7
  If ( 1 == bit_of_integer( Byte, I ) )      send data bit to all selected FPGA's
    Then VME_write_byte( FPGA_DATA, 0xFF )
    Else VME_write_byte( FPGA_DATA, 0x00 )
```

## FPGA Programming Algorithm for TDC Output Cards

After a reset of both FPGA's, the host needs to read 2 ASCII hex files, convert these files into bit streams and send these streams to the FPGA DATA register. FPGA\_MASK, FPGA\_PGMB, FPGA\_DONE, and FPGA\_INIT must be set to Output Card registers. Note that this algorithm is almost identical to the one for the input cards with the exception of a few constants and the file names. Below is the algorithm:

```
Main()  
  VME_write_byte( FPGA_MASK, 0x00)      set MASK to all active  
  VME_write_byte( FPGA_PGMB, 0x04)      turn on all PROGRAM pins  
  Wait( 1 uS )                          wait 1 microsecond  
  VME_write_byte( FPGA_PGMB, 0x00)      turn off all PROGRAM pins  
  Wait( 5 mS )                          wait 5 milliseconds for FPGA to respond  
  If ( 0 == 0x03 & VME_read_byte( FPGA_DONE ) ) &&  
    ( 0x03 == 0x03 & VME_read_byte( FPGA_INIT ) )  
    Then We are OK  
    Else ERROR: FPGA's didn't all respond to PROGRAM pulse  
  
  VME_write_byte( FPGA_MASK, 0xFE)      set MASK to SCORE active  
  While not at end of file "score.hex" (file1)      loop for all the characters in the input file1  
    Token1 = get_char( file1 )  
    Token2 = get_char( file1 )  
    Nibble1 = convert_hex_digit_to_integer( Token1 )  
    Nibble2 = convert_hex_digit_to_integer( Token2 )  
    Byte = 0x10 * Nibble1 + Nibble2  
    Send_byte_to_xilinx( Byte )          send data bits to SCORE  
  
  VME_write_byte( FPGA_MASK, 0xFD)      set MASK to GLMUX active  
  While not at end of file "glmux.hex" (file2)      loop for all the characters in the input file2  
    Token1 = get_char( file2 )  
    Token2 = get_char( file2 )  
    Nibble1 = convert_hex_digit_to_integer( Token1 )  
    Nibble2 = convert_hex_digit_to_integer( Token2 )  
    Byte = 0x10 * Nibble1 + Nibble2  
    Send_byte_to_xilinx( Byte )          send data bits to GLMUX  
  
  If ( 0x03 == 0x03 & VME_read_byte( FPGA_DONE ) ) &&  
    ( 0x03 == 0x03 & VME_read_byte( FPGA_INIT ) )  
    Then We are OK  
    Else ERROR: FPGA's didn't all program correctly  
  
Send_byte_to_xilinx( Byte )              function definition  
  For I=0 to 7  
    If ( 1 == bit_of_integer( Byte, I ) )          send data bit to all selected FPGA's  
      Then VME_write_byte( FPGA_DATA, 0xFF )  
      Else VME_write_byte( FPGA_DATA, 0x00 )
```