1 Using the V2/V3 F1TDC Module (V33 - 12/8/16)

1.1 Controlling the Module

Communication with the module is by standard VME bus protocols. All registers and memory locations are defined to be 4-byte entities. The VME slave module has three distinct address ranges.

A24 – The base address of this range is set by a 12-element DIP switch on the board. It occupies 4 Kbytes of VME address space, organized in 1 K 32-bit words. Relative to the base address, this space is utilized as follows:

000-FFF – Register space to control and monitor the module

A32 - The base address of this range is programmed into register ADR32. It occupies 8 Mbytes of VME address space, organized in 2 M 32-bit words. A read of any address in this range will yield the next TDC data word from the module. Even though the module is a FIFO, the expanded address range allows the VME master to increment the address during block transfers. This address range can participate in single cycle, 32-bit block, and 64-bit block reads. The only valid write to this address range is the data value 0x80000000 which re-enables the module to generate interrupts (after one has occurred). The address range must be enabled by setting ADR32[0] = 1.

A32 - The lower and upper limits of this address range are programmed into register ADR_MB. This common address range for a set of TDC modules in the crate is used to implement the Multiblock protocol. By means of token passing TDC data may be read out from multiple TDC modules using a single logical block read. The board possessing the token will respond to a read cycle in this address range with the next TDC data word from that module. The token is passed along a private daisy chain line to the next module when it has transferred all data from a programmed number of events (register BLOCK SIZE). The address range must be enabled by setting ADR_MB[0] = 1.

1.2 Module Operation

The following describes the setup and operation of the TDC in single and multiple module applications.

Single Module – After a reset of the module (CSR[31] = 1), the F1 chip configuration registers may be written directly through an A24 register (3C). The reference clock source is set to INTERNAL (CTRL[2..0] = 3). The sync_reset source is set to SOFT (CTRL[4..3] = 3, CTRL[9] = 1), and a sync_reset signal is generated (CSR[28] = 1). The BLOCK SIZE register is loaded with the number of events (i.e. triggers) that constitute a *block*. The INTERRUPT register may be loaded with the

interrupt ID and level if the module is to initiate an interrupt when the defined *block* of data is available for readout. The address for data access is loaded (ADR32). The event level interrupt (CTRL[24] = 1) is enabled if interrupt generation is desired. The BERR response is enabled (CTRL[25] = 1) to allow the module to indicate when the complete *block* of data has been read out. Permit data flow (CTRL2(0) = 1). Then set the trigger source to FRONT PANEL (CTRL[6..5] = 1).

When the programmed number of triggers has been received, the Block Ready Flag (CSR[4]) will be set and an interrupt will be generated if enabled. The user should initiate a DMA block read (32 or 64-bit) from the address in stored in ADR32. The length of the block read should be programmed to be larger than the expected size of the data *block* (e.g. 4 Mbytes). The module will terminate the DMA transfer by issuing BERR when all data from the *block* has been transferred. Interrupt generation must be re-enabled by writing 0x80000000 to the address in ADR32.

Multiple Modules – All modules should be reset and loaded with the configuration register values as described above for a single module. A P0 (VXS) distribution of reference clock, sync_reset, and trigger signal is assumed here. Set CTRL[2..0] = 0, CTRL[4..3] = 2, CTRL[6..5] = 2. The BLOCK SIZE register is loaded in each module. A unique address for data access is loaded into ADR32 for each module. The common address range for the Multiblock protocol is loaded into ADR_MB for each module. All modules are programmed to participate in the Multiblock protocol (CTRL[26] = 1). The *left-most* TDC module in the system is programmed as the *first* module (CTRL[27] = 1), and the *right-most* TDC module is designated the *last* module (CTRL[28] = 1). For the *first* module, the INTERRUPT register may be loaded and the event level interrupt bit enabled, if desired. The BERR response is enabled for the *last* module only. For VXS P0 signal distribution systems, the *token* passing lines are included in the system. Front signal distribution systems need external connections on the backplane (P2) between slots for token passing lines. Permit data flow in each module (CTRL2(0) = 1).

When the programmed number of triggers has been received, the *event level flag* will be set in each module and an interrupt will be generated by the *first* module (if enabled). The user should initiate a DMA block read (32 or 64-bit) from the address stored in ADR_MB. The length of the block read should be programmed to be larger than the total size of data from all modules (e.g. 4 Mbytes x # modules). Since the *first* module initially has the *token*, it will respond with data to the VME bus cycles. When data from the *first* module's block has been depleted, it passes the *token* to the next module in the chain. This module will respond with data until its data block is exhausted and the *token* is passed to the next module. The *last* module will terminate the DMA transfer by issuing BERR when all data from its *block* has been transferred. The token must be returned to the *first* board by writing '0x1000000' to the CSR register of the *first* board. Interrupt generation must be re-enabled by writing 0x80000000 to the address in ADR32 of the *first* module.

1.3 Module Registers

VERSION – board/firmware revision (0x0)

[7…0] – (R) – firmware revision

[15…8] – (R) – board revision

[31…16] – (R) – board type (“00F1”)

CSR – Control/Status (0x4)

0 – (R) – Reference clock PLL locked status (1 = locked)

1 – (R) – Board clock PLL locked status (1 = locked)

2 – (R) – TDC chip configuration error

3 – (R) – Block of Events Accepted (Event Level Flag)

4 – (R) – Block of Events Ready for readout

5 – (R) – BERR status (1 = module asserted BERR)

6 – (R) – Token status (1 = module has token)

7 – (R) – Events on board empty flag status (1 = no events)

8 – (R) – TDC chip 0 error

9 – (R) – TDC chip 1 error

10 – (R) – TDC chip 2 error

11 – (R) – TDC chip 3 error

12 – (R) – TDC chip 4 error

13 – (R) – TDC chip 5 error

14 – (R) – TDC chip 6 error (V2 only)

15 – (R) – TDC chip 7 error (V2 only)

16 – (R/W) – TRIGGER TIME WORD ERROR – a mismatch of build event number and trigger number from time word occurred. Clear latched status by writing '1' to this bit.

17 – (R) – Internal Buffer #0 empty flag

18 – (R) – Internal Buffer #1 empty flag

19 – 20 – Spare (read as zero)

21 – (W) – FORCE BLOCK TRAILER INSERTION – will be successful only if there are NO triggers waiting to be processed

22 – (R) – Last FORCE BLOCK TRAILER INSERTION Successful

23 – (R) – Last FORCE BLOCK TRAILER INSERTION Failed

24 – (W) – Token Return – return token to 1st module (after readout)

25 – 26 – Spare

27 – (W) – Pulse software generated START (if CTRL[9] = 1)

28 – (W) – Pulse software generated SYNC_RESET (if CTRL[9] = 1)

29 – (W) – Pulse software generated TRIGGER (if CTRL[9] = 1)

30 – (W) – Pulse soft reset

31 – (W) – Pulse hard reset

CTRL – Control (0x8)

0 – (R/W) – Reference clock select (0 = P0, 1 = internal/front panel)

1 – (R/W) – Internal reference clock enable (0 = OFF, 1 = ON)

2 – (R/W) – Internal/front panel reference clock select (only if CTRL[0] = 1) (0 = internal, 1 = front panel)

[4…3] – (R/W) – SYNC_RESET source (0 = none, 1 = FP, 2 = P0, 3 = soft)

[6…5] – (R/W) – TRIGGER source (0 = none, 1 = FP, 2 = P0, 3 = soft)

[8…7] – (R/W) – START source (0 = none, 1 = FP, 2 = P0, 3 = soft)

9 – (R/W) – Enable Soft control signals (SYNC_RESET, TRIGGER, START)

10 – 15 – Spare

[23…16] – (R/W) – Enable data from F1 chips 7...0 (‘1’ = enable, ‘0’ = disable) (DO NOT enable chips 6, 7 for V3 type module.)

24 – (R/W) – Enable Interrupt

25 – (R/W) – Enable BERR response

26 – (R/W) – Enable Multiblock protocol

27 – (R/W) – FIRST board in Multiblock system

28 – (R/W) – LAST board in Multiblock system

29 – (R/W) – Multiblock Token passed on P0

30 – (R/W) – Multiblock Token passed on P2

31 – Spare

EVENT COUNT (0xC) – Number of events currently stored on board

[23…0] - (R) – Event count[23…0]

[31…24] – (not used – read as ‘0’)

BLOCK SIZE (0x10)

[15…0] - (R/W) – Number of events defining a block

[31…16] – (not used = read – as ‘0’)

INTERRUPT (0x14)

[7…0] – (R/W) – Interrupt ID (vector)

[10…8] – (R/W) – Interrupt Level [2..0]. Valid values = 1,..,7.

11 - 15 – (not used)

[20…16] – (R) – Geographic Address (slot number) in VME64x chassis.

21 – 22 – (not used – read as '0')

23 – (R) – Parity Error in Geographic Address.

24 – 31 – (not used – read as '0')

ADR32 – Address for data access (0x18)

0 – (R/W) – Enable 32-bit address decoding

1 – 6 – (not used – read as 0)

[15…7] – (R/W) – Base Address for 32-bit addressing mode (8 Mbyte total)

ADR_MB – Multiblock Address for data access (0x1C)

0 – (R/W) – Enable Multiblock address decoding

1 – 6 – (not used – read as 0)

[15…7] – (R/W) – Lower Limit address (ADR_MIN) for Multiblock access

16 – 21 – (not used – read as 0)

[31…23] – (R/W) – Upper Limit address (ADR_MAX) for Multiblock access

The board that has the TOKEN will respond with data when the VME address satisfies the following condition:

$$\text{ADR\_MIN} \leq \text{Address} < \text{ADR\_MAX}.$$

TDC STATUS (4 registers)

(0xXX – TDC chips 'A' & 'B'): (0x20 – TDC chips 0 & 1)
(0x24 – TDC chips 2 & 3)
(0x28 – TDC chips 4 & 5)
(0x2C – TDC chips 6 & 7)

(bits 0 – 15 for chip 'B', bits 16 – 31 for chip 'A')

0 – Resolution LOCKED (B)

1 – TDC Hit FIFO Overflow (B)

2 – TDC Trigger FIFO Overflow (B)

3 – TDC Output FIFO Overflow (B)

4 – External FIFO Full (B)

5 – External FIFO Almost Full – BUSY asserted if chip enabled (B)

6 – External FIFO Empty (B)

7 – TDC Initialized (B)

8 – Loss of Resolution Lock Occurred (B)

9 – TDC Hit FIFO Overflow Occurred (B)

10 – TDC Trigger FIFO Overflow Occurred (B)

11 – TDC Output FIFO Overflow Occurred (B)

12 – External FIFO Full Occurred (B)

13 – 15 – (not used – read as 0)

16 – Resolution LOCKED (A)

17 – TDC Hit FIFO Overflow (A)

18 – TDC Trigger FIFO Overflow (A)

19 – TDC Output FIFO Overflow (A)

20 – External FIFO Full (A)

21 – External FIFO Almost Full – BUSY asserted if chip enabled (A)

22 – External FIFO Empty (A)

23 – TDC Initialized (A)

24 – Loss of Resolution Lock Occurred (A)

25 – TDC Hit FIFO Overflow Occurred (A)

26 – TDC Trigger FIFO Overflow Occurred (A)

27 – TDC Output FIFO Overflow Occurred (A)

28 – External FIFO Full Occurred (A)

29 – 31 – (not used – read as 0)

SCALER 1 – Event Count (0x30)

[31…0] – (R) – total event count

SCALER 2 – Event Counters (0x34)

[3…0] – (R) – low 4 bits of event counter for TDC chip #1

[7…4] – (R) – low 4 bits of event counter for TDC chip #2

[11…8] – (R) – low 4 bits of event counter for TDC chip #3

[15…12] – (R) – low 4 bits of event counter for TDC chip #4

[19…16] – (R) – low 4 bits of event counter for TDC chip #5

[23…20] – (R) – low 4 bits of event counter for TDC chip #6

[27…24] – (R) – low 4 bits of event counter for TDC chip #7

[31…28] – (R) – low 4 bits of event counter for TDC chip #8

SPARE (0x38) (R/W)

F1TDC CHIP CONFIGURATION (0x3C) (R/W)

[23…21] – F1 chip address

20 – Common bit (1 = broadcast)

[19…16] – register address on F1 chip

[15…0] – data for F1 register selected

CTRL2 (R/W) (0x40)

0 – GO DATA – allow data to be accepted from F1 chips

1 – GO HEADERS – force all F1 chip headers into data stream

15 – Force BUSY output

16 – System Test Mode (overrides Single Board Test Mode)

17 – Single Board Test Mode

18 – Trigger LED on (Single Board Test Mode)

19 – STATUS LED on (Single Board Test Mode)

CONFIGURATION_CSR (0x44) – (Firmware Update)

31 – (R/W) – Write Enable (1 = write mode, 0 = read mode)

30 – (R/W) – Bulk Erase (bit 31 = 1 also required)

29 – (R/W) – Sector Erase (bit 31 = 1 also required)

[28…16] – (R/W) – Reserved

[15…9] – (R) – Reserved

8 – (R) – Busy (operation in progress)

[7…0] – (R) – Last Valid Data Read

CONFIGURATION_ADR/DATA_(R/W) (0x48) – (Firmware Update)

[31…8] – EPROM address

[7…0] – EPROM data to write

SERIAL EPROM (R/W) (0x4C) – (Module Serial Number)

The serial number is 7 ASCII characters long, and extends from EPROM byte address 0 thru 6. Read the serial number character-by-character. To read one character:
WRITE data word **0x3000 | eprom_byte_address** to this register,
READ data word from this register. Character = data_word >> 24

TRIGGER 2 SCALER – (0x50)

[31…0] – (R) – total TRIGGER 2 count

SYNC_RESET SCALER – (0x54)

[31…0] – (R) – total SYNC_RESET count

TEST SCALER – (0x58)

TOKEN TEST (0x5C)

TOKEN TEST SCALER (0x60)

AUXILIARY 'B' (3 registers) (0x64 – 0x6C) – **V2 ONLY**

PULSE DELAY (0x64) – **V3 ONLY**

PULSER DAC (0x68) – **V3 ONLY**

PULSER CONTROL (0x6C) – **V3 ONLY**

BLOCK COUNT – (0x70)

[31…20] – not used

[19…0] – (R) - number of event BLOCKS on board (non-zero → CSR[4] = 1).

BLOCK FIFO COUNT – (0x74)

[31…6] – not used

[5…0] – (R) - number of entries in BLOCK WORD COUNT FIFO

BLOCK WORD COUNT FIFO – (64 deep FIFO) (0x78)

[31…25] – not used (read as ‘0’)

24 – (R) – count not valid (word count FIFO empty)

[23…20] – not used (read as ‘0’)

[19…0] – (R) - number of words in next event BLOCK

STATUS 0 – (0x7C) (R)

[2…0] – reserved

[5…3] – current F1 chip (0-7) data stream selected for build

6 – full flag status, FPGA output FIFO #1 (32k x 32)

7 – full flag status, FPGA output FIFO #2 (32k x 32)

[15…8] – paused status of F1chip FPGA input data streams 7…0

[23…16] – empty flag status, FPGA input FIFOs (64 x 16), for F1 chip data streams 7…0

[31…24] – empty flags status, FPGA intermediate FIFOs (256 x 32), for F1 chip data streams 7…0

STATUS 1 – (0x80) (R)

31 – empty flag status, FPGA output FIFO #1 (32k x 32)

[30…16] – word count, FPGA output FIFO #1 (32k x 32)

15 – empty flag status, FPGA output FIFO #2 (32k x 32)

[14…0] – word count, FPGA output FIFO #2 (32k x 32)

STATUS 2 – (0x84) (R)

[31…24] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 0 data stream

[23…16] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 1 data stream

[15…8] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 2 data stream

[7…0] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 3 data stream

STATUS 3 – (0x88) (R)

[31…24] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 4 data stream

[23…16] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 5 data stream

[15…8] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 6 data stream

[7…0] – word count, FPGA intermediate FIFO (256 x 32), F1 chip 7 data stream

STATUS 4 – (0x8C) (R)

[31…28] – data transfer state machine value, F1 chip 0 input data stream

[27…24] – data transfer state machine value, F1 chip 1 input data stream

[23…20] – data transfer state machine value, F1 chip 2 input data stream

[19…16] – data transfer state machine value, F1 chip 3 input data stream

[15…12] – data transfer state machine value, F1 chip 4 input data stream

[11…8] – data transfer state machine value, F1 chip 5 input data stream

[7…4] – data transfer state machine value, F1 chip 6 input data stream

[3…0] – data transfer state machine value, F1 chip 7 input data stream

STATUS 5 – (Data into intermediate FIFO) – (0x90) (R)

[31…24] – chip 0 header count

[23…16] – chip 0 trailer count

[15…8] – chip 1 header count

[7…0] – chip 1 trailer count

STATUS 6 – (Data into intermediate FIFO) – (0x94) (R)

[31…24] – chip 2 header count

[23…16] – chip 2 trailer count

[15…8] – chip 3 header count

[7…0] – chip 3 trailer count

STATUS 7 – (Data into intermediate FIFO) – (0x98) (R)

[31…24] – chip 4 header count

[23…16] – chip 4 trailer count

[15…8] – chip 5 header count

[7…0] – chip 5 trailer count

STATUS 8 – (Data into intermediate FIFO) – (0x9C) (R)

[31…24] – chip 6 header count

[23…16] – chip 6 trailer count

[15…8] – chip 7 header count

[7…0] – chip 7 trailer count

STATUS 9 – (LATCHED ERRORS - Data into intermediate FIFO) – (0xA0) (R)

0 – chip 0: word assembled incorrectly (high/low)

1 – chip 0: two consecutive headers detected

2 – chip 0: two consecutive trailers detected, or start with trailer

3 – chip 0: trigger number from header not equal to header count

4 – chip 1: word assembled incorrectly (high/low)

5 – chip 1: two consecutive headers detected

6 – chip 1: two consecutive trailers detected, or start with trailer

7 – chip 1: trigger number from header not equal to header count

8 – chip 2: word assembled incorrectly (high/low)

9 – chip 2: two consecutive headers detected

10 – chip 2: two consecutive trailers detected, or start with trailer

11 – chip 2: trigger number from header not equal to header count

12 – chip 3 word assembled incorrectly (high/low)

13 – chip 3: two consecutive headers detected

14 – chip 3: two consecutive trailers detected, or start with trailer

15 – chip 3: trigger number from header not equal to header count

16 – chip 4: word assembled incorrectly (high/low)

17 – chip 4: two consecutive headers detected

18 – chip 4: two consecutive trailers detected, or start with trailer

19 – chip 4: trigger number from header not equal to header count

20 – chip 5: word assembled incorrectly (high/low)

21 – chip 5: two consecutive headers detected

22 – chip 5: two consecutive trailers detected, or start with trailer

23 – chip 5: trigger number from header not equal to header count

24 – chip 6: word assembled incorrectly (high/low)

25 – chip 6: two consecutive headers detected

26 – chip 6: two consecutive trailers detected, or start with trailer

27 – chip 6: trigger number from header not equal to header count

28 – chip 7: word assembled incorrectly (high/low)

29 – chip 7: two consecutive headers detected

30 – chip 7: two consecutive trailers detected, or start with trailer

31 – chip 7: trigger number from header not equal to header count

STATUS 10 – (Data into input FIFO) – (0xA4) (R)

[31…24] – chip 0 header count

[23…16] – chip 0 trailer count

[15…8] – chip 1 header count

[7…0] – chip 1 trailer count

STATUS 11 – (Data into input FIFO) – (0xA8) (R)

[31…24] – chip 2 header count

[23…16] – chip 2 trailer count

[15…8] – chip 3 header count

[7…0] – chip 3 trailer count

STATUS 12 – (Data into input FIFO) – (0xAC) (R)

[31…24] – chip 4 header count

[23…16] – chip 4 trailer count

[15…8] – chip 5 header count

[7…0] – chip 5 trailer count

STATUS 13 – (Data into input FIFO) – (0xB0) (R)

[31…24] – chip 6 header count

[23…16] – chip 6 trailer count

[15…8] – chip 7 header count

[7…0] – chip 7 trailer count

STATUS 14 – (LATCHED ERRORS - Data into input FIFO) – (0xB4) (R)

0 – chip 0: two consecutive high words detected

1 – chip 0: two consecutive low words detected

2 – chip 0: header or trailer detected for channels other than 0 or 7

3 – chip 1: two consecutive high words detected

4 – chip 1: two consecutive low words detected

5 – chip 1: header or trailer detected for channels other than 0 or 7

6 – chip 2: two consecutive high words detected

7 – chip 2: two consecutive low words detected

8 – chip 2: header or trailer detected for channels other than 0 or 7

9 – chip 3: two consecutive high words detected

10 – chip 3: two consecutive low words detected

11 – chip 3: header or trailer detected for channels other than 0 or 7

12 – chip 4: two consecutive high words detected

13 – chip 4: two consecutive low words detected

14 – chip 4: header or trailer detected for channels other than 0 or 7

15 – chip 5: two consecutive high words detected

16 – chip 5: two consecutive low words detected

17 – chip 5: header or trailer detected for channels other than 0 or 7

18 – chip 6: two consecutive high words detected

19 – chip 6: two consecutive low words detected

20 – chip 6: header or trailer detected for channels other than 0 or 7

21 – chip 7: two consecutive high words detected

22 – chip 7: two consecutive low words detected

23 – chip 7: header or trailer detected for channels other than 0 or 7

[24…31] – (reserved)

STATUS 15 – (0xB8) (R)

[31…25] – reserved (read as 0)

[24…8] – 17-bit SST state machine value

[7…0] – 8-bit event build state machine value

STATUS 16 – (spare) – (0xBC)

System Test Registers (0xC0 – 0xFC) (set CTRL(17) = 1 (System Test Mode))

TEST BIT REGISTER (0xC0)

0 – (R/W) – trigger_out_p0 (1 = asserted, 0 = not asserted)

1 – (R/W) – busy_out_p0 (1 = asserted, 0 = not asserted)

2 – (R/W) – sdlink_out_p0 (1 = asserted, 0 = not asserted)

3 – (R/W) – token_out_p0 (1 = asserted, 0 = not asserted)

[4 – 7] – (R/W) – spare out test bits

8 – (R) – status_b_in_p0 state (1 = asserted, 0 = not asserted)

9 – (R) – token_in_p0 state (1 = asserted, 0 = not asserted)

[10 - 14] – (R) – reserved (read as '0')

15 – (R) – 'clk_31.25' counter status (1 = counting, 0 = not counting)

[16 - 31] – (R) – reserved (read as '0')

CLOCK_31.25 COUNT REGISTER (0xC4)

0 – (W) – Write '0' resets the counter. Write '1' initiates 20 us counting interval.

[31 - 0] – (R) – CLK_31.25 counter value. (Should be 625 after count interval.)

SYNC_IN_P0 COUNT REGISTER (0xC8)

0 – (W) – Write '0' resets the counter.

[31 - 0] – (R) – SYNC_IN_P0 counter value.

TRIG1_IN_P0 COUNT REGISTER (0xCC)

0 – (W) – Write '0' resets the counter.

[31 - 0] – (R) – TRIG1_IN_P0 counter value.

TRIG2_IN_P0 COUNT REGISTER (0xD0)

0 – (W) – Write ‘0’ resets the counter.

[31 - 0] – (R) – TRIG2_IN_P0 counter value.

SPARE TEST REGISTERS (11) – (0xD4 – 0xFC)

State Machine History – The capability exists to record the state machine values for several important state machines in the design. This greatly enhances the ability to understand and correct error conditions that may occur.

The user defines the target state machine and the maximum number of state transitions to record in the STATE LEVEL register. Recording begins when bit 31 of the STATE CSR is set to '1'. Recording is continuous – up to *N* state transitions are preserved in the FIFO memory when bit 31 of the STATE CSR is set to '0'. The user determines the actual number of state values *n* recorded by reading STATE CSR bits [8 – 0], and then performs *n* reads of the STATE VALUE FIFO. The order of the state values returned is from earliest to latest.

STATE LEVEL – (0x100)

[31 - 18] – reserved

[17 - 16] – state machine selector (0 = SST, 1 = event build (default))

[15 - 9] – reserved

[8 - 0] – maximum number *N* of consecutive state values to save (default = 500)

STATE CSR – (0x104)

31 – (R/W) – save states

[30 - 28] – reserved

27 – (R) – state storage full

26 – (R) – state storage empty

[25 - 9] – reserved

[8 - 0] – (R) – number of state values currently saved

STATE VALUE FIFO – (0x108) (R)

[31 - 18] – reserved

[17 - 0] – state machine value

SPARE STATE REGISTER – (0x10C)

Event Building Control and Monitoring – For each trigger, event fragments from multiple (up to 8) F1TDC chips must be combined into a single event. In the ideal case, each F1TDC chip produces exactly one event fragment (bracketed with a header and trailer), so the sequential collection of these fragments by the event builder is straightforward. However, at very high trigger and hit rates, one or more of the F1TDC chips may fail to report data (header/trailer) for a trigger. For an event builder that assumes the ideal F1TDC behavior, misalignment of the data or event builder hang up can result.

The current event builder can detect and correct for missing F1TDC event fragments with a timeout mechanism and by comparing the trigger number reported in the F1TDC header word with the actual trigger count in the FPGA. Even though an F1TDC chip fails to report data for a trigger, it increments the trigger number reported in the header, so that subsequent triggers will have the proper trigger number. When an event fragment is determined to be missing (timeout or skipped trigger number), a fake header and companion data word is inserted in its place. Data alignment is maintained and the event builder will not hang up.

The fake header and data words have the appropriate chip number and report the channel number as 7. The fake header has the correct trigger number (as determined by the FPGA), and reports a trigger time of 511 (not realizable by an actual event). The fake data word sets bit 22 to 1 as a tag, and reports the hit time as 0 (not realizable by an actual event).

Control and monitoring of this complex event builder is through the registers below.

EVENT BUILDER TIMER AND CONTROL – (0x110)

[31 - 16] – reserved

15 – disable insertion of fake data on timeout (default = 0)

14 – disable insertion of fake data on skipped trigger number (default = 0)

13 – disable deletion of data arriving after timeout (default = 0)

[12 - 8] – reserved

[7 - 0] – timeout value (default = 128 => 2.56 us)

INSERTED DATA COUNT – (0x114)

[31 - 16] – (R) – inserted data count due to timeouts

[15 - 0] – (R) – inserted data count due to skipped trigger numbers

(Writing '1' to bit 31 clears counters)

DELETED DATA COUNT – (0x118)

[31 - 16] – reserved

[15 - 0] – (R) – deleted data count due to data arriving after timeout and insertion of fake data

(Writing '1' to bit 31 clears counter)

SPARE EVENT BUILDER REGISTER – (0x11C)

EVENT BLOCK MONITOR REGISTER – (0x120) (R)

31 – Block of Events Accepted (Event Level Flag) (identical to **CSR(3)**)

30 – Block of Events Ready for readout (identical to **CSR(4)**)

[29 - 18] – number of events on board waiting to be built

[17 - 9] – number of completely built blocks of events on board

[8 - 0] – number of events built in current block build

EVENT BUILD PROGRESS REGISTER – (0x124)

31 – (W) – writing '1' resets count

[30 - 24] – reserved (read as '0')

[23 - 0] – (R) – Maximum number of events on board waiting to be built since last reset

Busy Assertion Control – The module can be configured to assert ***busy*** when the output FIFO word count reaches a fixed limit (57,344 words of 65,536 storage), or when the number of on-board events *not yet built* exceeds a programmable value. (Both modes can be selected simultaneously.) The Trigger Supervisor module will not accept and distribute triggers while ***busy*** is asserted. This mechanism protects a module from exhausting its on-board storage (resulting in data and synchronization loss). The configuration and status of the ***busy*** signal is through the BUSY CONTROL REGISTER. Note that a forced assertion of ***busy*** is possible through CTRL2, bit 15.

BUSY CONTROL REGISTER – (0x128)

31 – Enable ***busy*** assertion due to output FIFO word count (default = '1')

30 – Enable ***busy*** assertion due to events not built count (default = '0')

[29 - 25] – reserved (read as '0')

24 – (R) – current ***busy*** status ('1' = asserted)

23 – (R) – current word count ***busy*** status

22 – (R) – current event count ***busy*** status

21 – (R) – latched word count ***busy*** status ('1' means at least one occurrence)

20 – (R) – latched event count ***busy*** status ('1' means at least one occurrence)

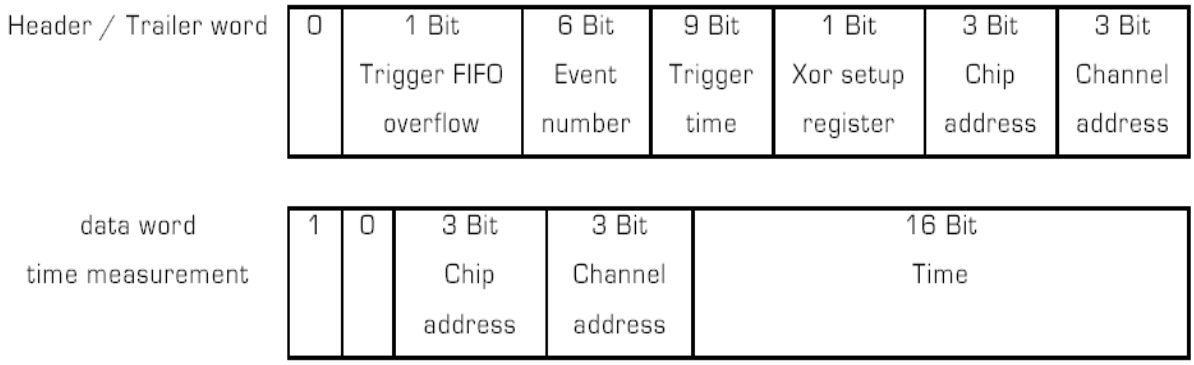
[19 - 16] – reserved (read as '0')

[15 - 0] – minimum number of events waiting to be built for ***busy*** assertion (when bit 30 = 1)

(writing the register clears latched status bits 20, 21)

1.4 Data Format

The F1TDC chip outputs 24-bit words. The words are of 2 types: header/trailer and data. The header/trailer words are used as event or channel separators, and provide information such as the event number and trigger time. A data word contains the time measurement for a hit. The bit assignments are shown below.



The header/trailer words can be enabled for each channel of the chip. The headers/trailers take up space in the F1TDC chip's output buffer while providing redundant information. So we enable only the header for channel 0, and the trailer for channel 7. This minimal header/trailer information is used to assemble event fragments from different chips into a single event fragment that is associated with a trigger for the board. The chip data stream then appears as follows:

header – channel 0
data – channel 0, earliest hit
…
data – channel 0, latest hit
data – channel 1, earliest hit
…
data – channel 1, latest hit
…
…
data – channel 7, earliest hit
…
data – channel 7, latest hit
trailer – channel 7

During event building on the module, all trailers are deleted. In the following discussion the module is assumed to have 8 F1 chips (V2). (The V3 module has 6 F1 chips.) The data stream associated with a trigger will appear as follows:

header – chip 0, channel 0
data – chip 0, channel 0, earliest hit
…
data – chip 0, channel 0, latest hit
…
…
data – chip 0, channel 7, earliest hit
…
data – chip 0, channel 7, latest hit
header – chip 1, channel 0
data – chip 1, channel 0, earliest hit
…
data – chip 1, channel 0, latest hit
…
…
data – chip 1, channel 7, earliest hit
…
data – chip 1, channel 7, latest hit
…
…
…
header – chip 7, channel 0
data – chip 7, channel 0, earliest hit
…
data – chip 7, channel 0, latest hit
…
…
data – chip 7, channel 7, earliest hit
…
data – chip 7, channel 7, latest hit

Chip headers contain independent measurements of the Event Number and Trigger Time. To assure that all chips of the module stay correctly synchronized, the Event Number and Trigger Time should be monitored by the user.

In the current version of the firmware, chip headers after chip 0 are suppressed from the data stream if no error conditions are present on the chip, and their Event Number and Trigger Time are identical to the values of the preceding chip. Since the first chip header is always inserted into the data stream, the Event Number and Trigger Time information for each chip can be completely reconstructed. The user also has the option of ***forcing*** all chip headers into the data stream (see CTRL2 register).

Any difference in the Event Number among the chips indicates a serious error that requires a reset of the board. **Trigger Time differences of up to 1 count among the chips is acceptable.** (Note that for a 9-bit Trigger Time, 0 and 511 differ by 1.) For the Trigger Time, this assumes that an external SYNC_RESET signal has been successfully applied at the start of the run. The user is strongly encouraged to monitor the Event Number and Trigger Time provided in these headers to assure synchronization across the entire system.

The data word transferred across the VME bus includes the 24-bit TDC word AND additional word type and F1TDC chip error information, as follows:

DATA - module data word (4 Mbyte address range, base programmed in register ADR32)

[31…27] – Word type

26 – F1 chip Resolution Locked Status

25 – F1 chip Output FIFO Overflow Status

24 – F1 chip Hit FIFO Overflow Status

[23…0] – F1 chip word, as described above

1.5 Data Word Categories

Data words from the module are divided into two categories: Data Type Defining (bit 31 = 1) and Data Type Continuation (bit 31 = 0). Data Type Defining words contain a 4-bit data type tag (bits 30 - 27) along with a type dependent data payload (bits 26 - 0). Data Type Continuation words provide additional data payload (bits 30 – 0) for the *last defined data type*. Continuation words permit data payloads to span multiple words and allow for efficient packing of some types of data. Any number of Data Type Continuation words may follow a Data Type Defining word.

Current Data Type List for F1TDC

0 – block header
1 – block trailer
2 – event header
3 – trigger time
4 – 6 – reserved
7 – time measurement data
8 – chip header
9 – 13 – reserved
14 – data not valid (empty module)
15 – filler (non-data) word

Data Types

Block Header (0) – indicates the beginning of a block of events. (High-speed readout of a board or set of boards is done in blocks of events.)

(31) = 1
(30 – 27) = 0
(26 – 22) = slot number (set by VME64x backplane)
(21 – 18) = module ID (‘3’ for V2, ‘4’ for V3))
(17 – 8) = event block number (used to align blocks when building events)
(7 – 0) = number of events in block

Block Trailer (1) – indicates the end of a block of events. The data words in a block are bracketed by the block header and trailer.

(31) = 1
(30 – 27) = 1
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = total number of words in block of events

Event Header (2) – indicates the start an event. The included trigger number is useful to ensure proper alignment of event fragments when building events. The 22-bit trigger

number is not a limitation, as it will be used to distinguish events within event blocks, or among events that are concurrently being built or transported.

(31) = 1
(30 – 27) = 2
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = event number (trigger number)

Trigger Time (3) – time of trigger occurrence relative to the most recent global reset. Time is measured by a 40-bit counter that is clocked by the 31.25 MHz system clock. The global reset signal is distributed to every module in the system. The assertion of the global reset clears the counters and inhibits counting. The de-assertion of global reset enables counting and thus sets t = 0 for the component. The trigger time is necessary to ensure system synchronization and is useful in aligning event fragments when building events. With careful clock, trigger, and global reset distribution it may be possible to achieve identical trigger times from all components of the system. However, even if t = 0 is not the same for all components, *changes* in trigger times can be monitored to ensure system synchronization is maintained. The five bytes of the trigger time

$$\text{Time} = T_A\ T_B\ T_C\ T_D\ T_E$$

are reported in two words (Type Defining + Type Continuation):

Word 1:

(31) = 1
(30 – 27) = 3
(26 – 24) = reserved (read as 0)
(23 – 16) = T_C
(15 – 8) = T_D
(7 – 0) = T_E

Word 2:

(31) = 0
(30 – 16) = reserved (read as 0)
(15 – 8) = T_A
(7 – 0) = T_B

Time Measurement Data (7) – bit 22 = 0 indicates an input channel has a hit within the defined time window relative to the trigger signal (bit 22 = 1 tags fake data)

(31) = 1
(30 – 27) = 7
(26) – F1 chip Resolution Status (1 = locked, 0 = unlocked)
(25) – F1 chip Output FIFO Overflow Status (1 = overflow, 0 = normal)
(24) – F1 chip Hit FIFO Overflow Status (1 = overflow, 0 = normal)
(23) = 1
(22) = 0 (1 for fake data)
(21 – 19) – F1 chip number (0-7)

(18 – 16) – F1 channel number on chip (0-7) (7 for fake data)
(15 – 0) – time measurement (0 for fake data)

Chip Header (8) – contains information about the trigger received by the F1 chip
(31) = 1
(30 – 27) = 8
(26) – F1 chip Resolution Status (1 = locked, 0 = unlocked)
(25) – F1 chip Output FIFO Overflow Status (1 = overflow, 0 = normal)
(24) – F1 chip Hit FIFO Overflow Status (1 = overflow, 0 = normal)
(23) = 0
(22) = F1 chip Trigger FIFO Overflow Status (1 = overflow, 0 = normal)
(21 – 16) – F1 chip Trigger Number
(15 – 7) – F1 chip Trigger Time (511 for fake data)
(6) – Setup Register Tag (should NOT change during run) (0 for fake data)
(5 – 3) – F1 chip number (0-7)
(2 – 0) – F1 channel number on chip (0-7) (7 for fake data)

Data Not Valid (14) – module has no valid data available for read out.
(31) = 1
(30 – 27) = 14
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = undefined

Filler Word (15) – non-data word appended to the block of events. Forces the total number of 32-bit words read out of a module to be a multiple of 2 or 4 when 64-bit VME transfers are used. **This word should be ignored**.
(31) = 1
(30 – 27) = 15
(26 – 22) = slot number (set by VME64x backplane)
(21 – 0) = undefined

1.6 Event Blocking

To increase readout performance the module packages events into blocks. The number of events in a block is defined by the BLOCK_SIZE register. Data from the block of events is bracketed by the Block Header and Block Trailer. Event Headers separate the data for events within the block.

For example, suppose BLOCK_SIZE = 3. Readout of the block will yield the following data stream (Note: ALL chip headers are shown here) :

Block Header
- Event Header (event 1 of block)
  - Trigger Time – word 1
  - Trigger Time – word 2
  - Chip 0 header
    - Time measurement data – chip 0
    - . . .
  - Chip 7 header
    - Time measurement data – chip 7
- Event Header (event 2 of block)
  - Trigger Time – word 1
  - Trigger Time – word 2
  - Chip 0 header
    - Time measurement data – chip 0
    - . . .
  - Chip 7 header
    - Time measurement data – chip 7
- Event Header (event 3 of block)
  - Trigger Time – word 1
  - Trigger Time – word 2
  - Chip 0 header
    - Time measurement data – chip 0
    - . . .
  - Chip 7 header
    - Time measurement data – chip 7

Block Trailer

Note: Filler Words may appear before or after the Block Trailer in the readout stream. These force the total number of 32-bit words read out to be a multiple of 2 or 4, depending on the VME readout mode used. Filler Words should be ignored.

1.6a Software Block Trailer

It is possible for the user to **FORCE** the insertion of a Block Trailer into the data stream by software command (see **CSR** bits 21, 22 ,23). This capability is useful in simplifying readout of data when a partial block of events is on-board. This situation can occur if the data run does not end on a block boundary, or if an individual module has missed a valid trigger. In the latter case, after reading out the module, the user should CLEAR it (i.e. soft reset), and then apply a SYNC_RESET to the system before continuing.

1.7 Input Channel Mapping

In the above description, channel number means the chip channel number reported by the F1 chip itself. How this relates to the input channel number (front panel) depends on the module type (V2 or V3). The input channel numbers are numbered consecutively from bottom to top of the front panel: (0…31) for V2, (0… 47) for V3.

Recall (from the F1 chip documentation) that in normal resolution mode (V3), the chip provides 8 measurement channels (0…7), while in high resolution mode (V2), pairs of channels are combined to provide 4 measurement channels. In high resolution mode, reported channel numbers 0 & 1 refer to the first measurement channel; similarly, the pairs 2 & 3, 4 & 5, 6 & 7 refer to the remaining high resolution measurement channels.

The module itself performs the following re-mapping of the F1 chip channel numbers:
7 ->0, 6->1, 5->2, 4->3, 3->4, 2->5, 1->6, 0->7.

The USER should apply the following relationships (C-code) to obtain the front panel input channel number:

V3: input_channel = (chip_number <<3) | chip_channel;

V2: int channel_map[8] = {0, 0, 1, 1, 2, 2, 3, 3};
input_channel = (4 * chip_number) + channel_map[chip_channel];