5 Using the Module

5.1 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 divided as follows:

000-7FF – Register space to control and monitor the module

800-FFF – F1 chip configuration space

Detailed information about the F1 chip configuration space is found in Appendix C. To summarize, the F1 chip configuration space is divided as follows:

800-9FF – Configuration register file. The 4-byte registers are ordered as:

Chip 0 register 0, 1, 2, …, 15,
Chip 1 register 0, 1, 2, …, 15,
…
Chip 7 register 0, 1, 2, …, 15.

Direct writes to an F1 chip register may be made to the address:

Address = 800 + (40 * chip #) + (4 * register #).

The order in which the registers are written should be the following:

register 15, 10, 0, 1, 6, 8, 9, 11, 12, 13, 14, 7, 2, 3, 4, 5.

Sample values of the F1 chip registers under several operating conditions are found in Appendix B.

A00-BFF – Setup RAM (see Appendix C).

C00-DFF – Configuration control registers (see Appendix C).

E00-FFF – Spare

A32 - The base address of this range is programmed into register ADR32. It occupies 4 Mbytes of VME address space, organized in 1 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 EVENT LEVEL). The address range must be enabled by setting ADR_MB[0] = 1.

5.2 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 the A24 address space (800-9FF). The source for the control signals is set to Front (CTRL[0] = 0) and the internal clock is selected and turned on (CTRL[2] = 0, CTRL[1] = 1). The EVENT LEVEL 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), and the F1 chips that are to participate in the readout are selected (CTRL[23…16]). The event level interrupt (CTRL[3] = 1) is enabled if interrupt generation is desired. The BERR response is enabled (CTRL[6] = 1) to allow the module to indicate when the complete *block* of data has been read out.

When the programmed number of triggers has been received, the *event level flag* (CSR[3]) 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. The source of the control signals for the modules is set to Front or Back depending on the distribution system used. The internal clocks need not be turned on, as a clock source is provided as part of the distribution system. The EVENT LEVEL register and the chips selected for readout are loaded into 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[7] = 1). The *left-most* TDC module in the system is programmed as the *first* module (CTRL[8] = 1), and the *right-most* TDC module is designated the *last* module (CTRL[9] = 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 backplane mounted signal distribution systems, the *token* passing lines are included in the system. Front signal distribution systems need additional connections to the backplane for token passing lines (see Table 2).

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. Upon detecting BERR, the *first* module takes possession of the *token*. Interrupt generation must be re-enabled by writing 0x80000000 to the address in ADR32 of the *first* module.

5.3 Module Registers

CSR – Control/Status (0x0)

0 – (R/W) – spare control bit

1 – (R/W) – TDC chip configuration error (writing 1 clears status)

2 – (R) – TDC chip configuration active

3 – (R) – event level flag asserted

4 – (R) – zero events in buffer

5 – (R) – BERR status

6 – (R) – Token status

7 – (R) – ERROR condition exists for an enabled TDC chip

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

15 – (R) – TDC chip 7 error

16 – (R) – Internal Buffer # for next data word (0 = Buffer #0, 1 = Buffer #1)

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

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

19 – (R) – End-of-Block flag for next data word

20 – (R) – Filler Word flag for next data word (valid only for Buffer #1)

21 – (R) – External FIFO empty flag

22 – 23 – Spare (read as zero)

24 – 27 – (not used)

28 – (W) – Pulse Spare Out (if CTRL[12] = 1)

29 – (W) – Pulse Soft Trigger (if CTRL[5] = 1)

30 – (W) – Pulse Soft Reset

31 – (W) – Pulse Hard Reset

CTRL – Control (0x4)

0 – (R/W) – Front/Back Control Signals select (0 = Front, 1 = Back)

1 – (R/W) – Internal Clock Control (0 = Clock OFF, 1 = Clock ON)

2 – (R/W) – Clock Select (0 = Internal (if CTRL[0] = 0), 1 = External)

3 – (R/W) – Enable Event Level Interrupt

4 – (R/W) – Enable Error Interrupt

5 – (R/W) – Enable Soft Trigger

6 – (R/W) – Enable BERR response

7 – (R/W) – Enable Multiblock protocol

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

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

10 – (R/W) – Enable readout of all F1 chip headers/trailers

11 – (R/W) – Enable BUSY output to front panel

12 – (R/W) – Enable Spare output

13 – 15 – (not used – read as 0)

16 - 23 (R/W) – Enable Readout from TDC 0 - 7

24 – 31 – (not used – read as 1)

EVENT COUNT (0x8)

[15…0] - (R) – Event Count[15…0]. Event Count = 0 → CSR[4] = 1.

[31…16] – (not used)

EVENT LEVEL (0xC)

[15…0] - (R/W) – Event Level[15…0]. Event Count ≥ Event Level → CSR[3] = 1.

[31…16] – (not used)

INTERRUPT (0x10)

[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)

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

24 – 31 – (not used)

ADR32 – Address for data access (0x14)

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

1 – 5 – (not used – read as 0)

[15…6] – (R/W) – Base Address for 32-bit addressing mode (4 Mbyte total)

ADR_MB – Multiblock Address for data access (0x18)

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

1 – 5 – (not used – read as 0)

[15…6] – (R/W) – Lower Limit address (ADR_MIN) for Multiblock access

16 – 21 – (not used – read as 0)

[31…22] – (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:

$$ADR\_MIN \leq Address < ADR\_MAX.$$

TDC STATUS (4 registers)

(0xXX – TDC chips ‘A’ & ‘B’): (0x1C – TDC chips 0 & 1)
(0x20 – TDC chips 2 & 3)

(0x24 – TDC chips 4 & 5)
(0x28 – 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)

FIRM – Firmware revision (0x2C)

[7…0] – (R) – firmware revision

[15…8] – (R) – board type (“F1”)

SCALER 1 – Event Count (0x30)

[31…0] – (R) – total event count as determined by the 1st enabled TDC chip

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

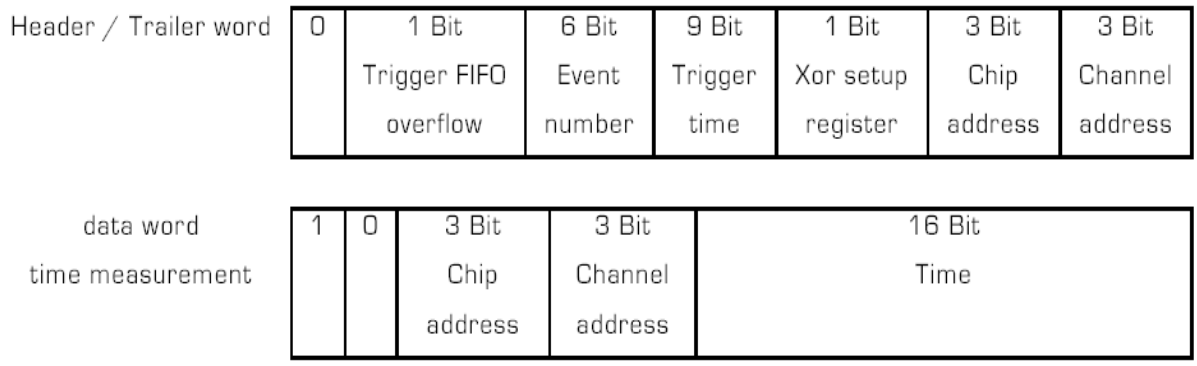
[23…20] – (R) – low 4 bits of event counter for TDC chip #6

[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

5.4 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. In this case the chip data stream for a trigger appears as follows:

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

In this case, the headers/trailers take up space in the F1TDC chip's output buffer while providing redundant information. A better solution is to enable only the header for channel 0, and the trailer for channel 7. 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

In our application of the F1TDC chip, we require that *at least* the header for channel 0 and the trailer for channel 7 be enabled for each chip that is to be read out. (The chips to be read out are specified by CTRL[23…16]). This minimal header/trailer information is used to assemble event fragments from different chips into a single event fragment that is associated with a given trigger for the board.

During module readout, the default mode is to suppress all headers/trailers *except* the *header* for channel 0 of the *first* chip enabled for readout (**Event Header**) and the *trailer* for channel 7 of the *last* chip enabled for readout (**Event Trailer**). If all chips 0-7 are enabled, the module data stream associated with a trigger will appear as follows:

header – chip 0, channel 0 (**Event Header**)
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
…
…
…
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
trailer – chip 7, channel 7 (**Event Trailer**)

Whenever an intermediate chip header indicates an *error condition* (Trigger FIFO Overflow) it will be *forced* into the data stream so that this condition will not be missed.

The normally suppressed intermediate chip headers do 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 are monitored for *changes* between chips. Intermediate chip headers are forced into the data stream only if their Event Number or Trigger Time differs from the values of the preceding chip. Since the **Event Header** is always inserted into the data stream, the Event Number and Trigger Time information for each chip can be completely reconstructed. As an example, suppose chip 4 of the module records a different Trigger Time for an event than the other 7 chips. The module data stream for this event will appear as follows:

header – chip 0, channel 0 (**Event Header**)
data – chip 0, channel 0, earliest hit
…
data – chip 3, channel 7, latest hit
header – chip 4, channel 0 (intermediate header)
data – chip 4, channel 0, earliest hit
…
data – chip 4, channel 7, latest hit
header – chip 5, channel 0 (intermediate header)
data – chip 5, channel 0, earliest hit
…
data – chip 7, channel 7, latest hit
trailer – chip 7, channel 7 (**Event Trailer**)

This indicates that chips 0-3 have the same Event Number and Trigger Time, while chip 4 has recorded different values. The intermediate header from chip 5 re-establishes values that are the same for chips 5-7.
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.

For diagnostic purposes, all intermediate chip headers/trailers can be forced to be read out by setting register bit CTRL[10] = 1. In this mode, each trigger will produce 16 header/trailer words from a module in addition to the time measurement data words.

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

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

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

24 – F1 chip Hit FIFO Overflow

25 – F1 chip Output FIFO Overflow

26 – F1 chip Resolution Locked

[31…27] – Slot ID

Slot ID = 1-21 indicates valid data. The module will return Slot ID = 30 when the data is not valid (e.g. an empty module is read, or data is not yet ready to be extracted). Slot ID = 0 is the tag for a 'filler' word. This is a non-valid data word that is inserted to make the block of data output from the module consist of an *even* number of words. (An even number of words simplifies 64-bit readout mode.) Note that a block of data may consist of many events (register EVENT LEVEL).

Data should be ignored if Slot ID = 0 or 30. **For 64-bit readout, if EITHER of the words of the pair have Slot ID = 30, BOTH words should be ignored**. (If one of the words is valid, it will appear again in the next successful read).