

## 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 divided as follows:

000-0FF – Register space to control and monitor the module (64 long words)

100-1FF – ADC processing registers (64 long words)

200-2FF – Reserved (64 long words)

300-3FF – Reserved (64 long words)

400-4FF – SYSTEM TEST registers (64 long words)

500-5FF – DEBUG (64 long words)

600-6FF – MOLLER (64 long words)

700-FFF – Reserved (640 long words)

In addition to registers that are directly mapped to a VME address (Primary Address), the module supports Secondary Addressing in the A24 address space. These registers are accessed through an address mapping register (Secondary Address Register). Each secondary address is associated with a primary address. A Primary Address may have up to 64 K secondary addresses associated with it. A VME cycle loads the mapping register with data which is the internal (secondary) address of the target register. A VME cycle with the associated primary address accesses (read/write) the chosen internal register. Important registers are assigned primary addresses, allowing them to be directly accessible in a single VME cycle. Setup tables are assigned secondary addresses. This allows for a large internal address space, while maintaining a small VME footprint.

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 FADC data word from the module. Even though the module is logically 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 FADC modules in the crate is used to implement the Multiblock protocol. By means of token passing FADC data may be read out from multiple FADC 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 FADC 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: set  $ADR\_MB[0] = 1$ .

### **1.3 Module Registers**

VERSION – board/firmware revision (0x0)

[7...0] – (R) – firmware revision

[15...8] – (R) – board revision

[31...16] – (R) – board type (“FADC”)

CSR – Control/Status (0x4)

0 – (R) – Event Accepted

1 – (R) – Block of Events Accepted

2 – (R) – Block of Events ready for readout

3 – (R) – BERR Status (1 = BERR asserted)

4 – (R) – Token Status (1 = module has token)

[5...9] – (reserved)

10 – (R) – DAC serialization status (1 = active)

11 – (R) – Data FIFO Empty Flag Asserted

12 – (R) – Data FIFO Almost Empty Flag Asserted

- 13 – (R) – Data FIFO Half Full Flag Asserted
- 14 – (R) – Data FIFO Almost Full Flag Asserted
- 15 – (R) – Data FIFO Full Flag Asserted
- 16 – (R) – ADC Processing FPGA high temperature alarm flag
- 17 – (R) – CTRL FPGA high temperature alarm flag
- [18...19] – (reserved)
- 20 – (W) – Pulse Soft **Trigger 2** (if CTRL[7] = 1 and CTRL[6..4] = 5)  
           (delayed **Trigger 1** follows; delay in TRIG21\_DELAY register)  
       (R) – Trigger 2 -> Trigger 1 sequence active
- 21 – (W) – Pulse Clear Module – soft reset + clear data pipelines  
       (R) – Clear Module process active
- 22 – (W) – ENABLE SCALERS INTO DATA STREAM with FORCED  
           BLOCK TRAILER INSERTION (write ‘1’ to bits 22, 23)
- 23 – (W) – FORCE BLOCK TRAILER INSERTION – will be successful only  
           if there are NO triggers waiting to be processed
- 24 – (R) – Last FORCE BLOCK TRAILER INSERTION Successful
- 25 – (R) – Last FORCE BLOCK TRAILER INSERTION Failed
- 26 – (R) – Local Bus Time Out – target AK or DK timed out (5 us);
- 27 – (R/W) – Local Bus Error – target protocol violation;  
           (write ‘1’ clears latched bits 26, 27)
- 28 – (W) – Pulse Soft **Sync Reset** (if CTRL[11] = 1 and CTRL[10..8] = 6)
- 29 – (W) – Pulse Soft **Trigger 1** (if CTRL[7] = 1 and CTRL[6..4] = 6)
- 30 – (W) – Pulse Soft Reset – initialize counters, state machines, memory
- 31 – (W) – Pulse Hard Reset – initialize module to power-up state

CTRL1 – Control 1 (0x8)

- [1...0] – (R/W) – Sampling Clock Source Select

- 0 = Internal Clock
- 1 = Front Panel connector
- 2 = P0 connector (VXS)
- 3 = P0 connector (VXS)

2 – (not used)

3 – (R/W) – Enable Internal Clock

[6...4] – (R/W) – Trigger Source Select

- 0 = Front Panel Connector (**Trigger 1**)
- 1 = Front Panel Connector (**Trigger 1**; synchronized)
- 2 = P0 Connector (VXS) (**Trigger1, Trigger 2**)
- 3 = P0 Connector (VXS) (**Trigger1, Trigger 2**; synchronized)
- 4 – (not used)
- 5 – Software Generated (**Trigger 2** + delayed **Trigger 1**)
- 6 = Software Generated (**Trigger 1**)
- 7 = Module Internal Logic

7 – (R/W) – Enable Soft Trigger

[10...8] – (R/W) – Sync Reset Source Select

- 0 = Front Panel Connector
- 1 = Front Panel Connector (synchronized)
- 2 = P0 Connector (VXS)
- 3 = P0 Connector (VXS) (synchronized)
- 4 – (not used)
- 5 – (not used)
- 6 = Software Generated
- 7 = no source

11 – (R/W) – Enable Soft Sync Reset

12 – (R/W) – Select Live Internal Trigger to Output (otherwise, Module Trigger).

13 – (R/W) – Enable Front Panel Trigger Output

14 – (R/W) – Enable P0 (VXS) Trigger Output

15 – (R/W) – Insert ADC parameter word into data stream. The data word appears as a block header continuation word and has the following format:

- [31...29] – 0
- [28...18] – PL (see ADC processing FPGA address map below)
- [17...9] – NSB
- [8...0] – NSA

- 16 – (R/W) – Suppress both trigger time words from data stream
- 17 – (R/W) – Suppress trigger time word 2 from data stream  
(most significant bytes)
- 18 – (R/W) – Reserved
- 19 – (reserved)
- 20 – (R/W) – Enable BERR response
- 21 – (R/W) – Enable Multiblock protocol
- 22 – (R/W) – FIRST board in Multiblock system
- 23 – (R/W) – LAST board in Multiblock system
- 24 – (reserved)
- 25 – (R/W) – Enable Debug Mode
- [27...26] – (R/W) – Readout Data Format (see **Appendix 1**)
  - 0 = Standard format (default)
  - 1 = Intermediate compression format
  - 2 = Full compression format
  - 3 = Full compression format
- 28 – (R/W) – Multiblock Token passed on P0
- 29 – (R/W) – Multiblock Token passed on P2
- 30 – (reserved)
- 31 – (R/W) – System Test Mode (0 = normal, 1 = test mode enabled)

CTRL2 – Control 2 (0xC)

- 0 – (R/W) – GO (allow data transfer from external FIFOs to input FIFOs)
- 1 – (R/W) – Enable **Trigger (1 & 2)** to Module (source = CTRL1[6...4])
- 2 – (R/W) – Enable **Sync Reset** to Module (source = CTRL1[10...8])
- 3 – (R/W) – Enable Internal Trigger Logic

4 – (R/W) – Enable Streaming mode (NO event build)

5 – (R/W) – Use pulse derived from leading edge of **Sync Reset** signal as module Sync Reset

[6...7] – (reserved)

8 – (R/W) – Enable Test Event Generation (for debug)

[9...15] – (reserved)

Bits 16 – 31 are functional only in Debug Mode (CTRL1[25] = 1)

16 – (reserved)

17 – (R/W) – Transfer data: build FIFO → output FIFO

[18...31] – (reserved)

#### BLOCK SIZE (0x10)

[15...0] – (R/W) – number of events in a BLOCK.  
Stored Event Count  $\geq$  BLOCK SIZE → CSR[3] = 1.

[31...16] – (reserved)

#### INTERRUPT (0x14)

[7...0] – (R/W) – Interrupt ID (vector)

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

11 - 15 – (reserved)

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

21 – 22 – (reserved)

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

24 – 31 – (reserved)

ADR32 – Address for data access (0x18)

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

1 – 6 – (reserved – 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 – (reserved – read as 0)

[15...7] – (R/W) – Lower Limit address (ADR\_MIN) for Multiblock access

16 – 22 – (reserved – 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:

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

SEC\_ADR – Secondary Address (0x20)

[15...0] – (R/W) – Secondary Address for 24-bit addressing mode

16 – (R/W) – Enable auto-increment mode (secondary address increments by 1 after each access of the associated primary address)

DELAY – Trigger/Sync Reset Delay (0x24) (NOT USED)

[21...16] – (R/W) – Sync reset delay

[5...0] – (R/W) – Trigger delay

INTERNAL TRIGGER CONTROL (0x28)

[23...16] – (R/W) – trigger width (4 ns per count; max = 1024 ns)

[11...0] – (R/W) – trigger hold off delay (4 ns per count; max = 16,384 ns)

### RESET CONTROL (0x2C)

0 – (W) – Hard reset – Control FPGA

1 – (W) – Hard reset – ADC processing FPGA

[2...3] – (reserved)

4 – (W) – Soft reset – Control FPGA

5 – (W) – Soft reset – ADC processing FPGA

[6...7] – (reserved)

8 – (W) – Reset – ADC data FIFO

[9...10] – (reserved)

10 – (W) – Reset – HITSUM FIFO

11 – (W) – Reset – DAC (all channels)

12 – (W) – Reset – EXTERNAL RAM Read & Write Address Pointers

[13...15] – (reserved)

16 – (W) – Take Token – return token to 1<sup>st</sup> board of multiboard set

[17...31] – (reserved)

### TRIGGER COUNT (0x30)

[31...0] – (R) – total trigger count

31 – (W) – reset count

### EVENT COUNT (0x34)

[23...0] – (R) – number of events on board (non-zero → CSR[0] = 1).



[31...24] – (reserved)

BLOCK COUNT – (0x38)

[31...20] – reserved

[19...0] – (R) - number of event BLOCKS on board (non-zero → CSR[2] = 1).

BLOCK FIFO COUNT – (0x3C)

[31...6] – reserved

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

BLOCK WORD COUNT FIFO – (64 deep FIFO) (0x40)

[31...25] – reserved (read as ‘0’)

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

[23...20] – reserved (read as ‘0’)

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

INTERNAL TRIGGER COUNT (0x44)

[31...0] – (R) – internal live trigger count

31 – (W) – reset count

EXTERNAL RAM WORD COUNT (0x48)

[31...22] – reserved (read as ‘0’)

21 – (R) – RAM empty

20 – (R) – RAM full (1,048,576 eight byte words)

[19...0] – (R) – data word count (eight byte words)

DATA FLOW STATUS (0x4C) (for debug)

DAC 1\_2 – DAC channels 1,2 (0x50)

31 – (R) – DAC channel 1 write timeout error

[30...28] – (reserved – read as 0)

[27...16] – (R/W) – DAC value channel 1

15 – (R) – DAC channel 2 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 2

DAC 3\_4 – DAC channels 3,4 (0x54)

31 – (R) – DAC channel 3 write timeout error

[30...28] – (reserved – read as 0)

[27...16] – (R/W) – DAC value channel 3

15 – (R) – DAC channel 4 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 4

DAC 5\_6 – DAC channels 5,6 (0x58)

31 – (R) – DAC channel 5 write timeout error

[30...28] – (reserved – read as 0)

[27...16] – (R/W) – DAC value channel 5

15 – (R) – DAC channel 6 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 6

DAC 7 8 – DAC channels 7,8 (0x5C)

31 – (R) – DAC channel 7 write timeout error

[30...28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 7

15 – (R) – DAC channel 8 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 8

DAC 9 10 – DAC channels 9,10 (0x60)

31 – (R) – DAC channel 9 write timeout error

[30...28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 9

15 – (R) – DAC channel 10 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 10

DAC 11 12 – DAC channels 11,12 (0x64)

31 – (R) – DAC channel 11 write timeout error

[30...28] – (reserved – read as 0)

[27...16]– (R/W) – DAC value channel 11

15 – (R) – DAC channel 12 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 12

DAC 13\_14 – DAC channels 13,14 (0x68)

31 – (R) – DAC channel 13 write timeout error

[30...28] – (reserved – read as 0)

[27...18]– (R/W) – DAC value channel 13

15 – (R) – DAC channel 14 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 14

DAC 15\_16 – DAC channels 15,16 (0x6C)

31 – (R) – DAC channel 15 write timeout error

[30...28] – (reserved – read as 0)

[27...16] – (R/W) – DAC value channel 15

15 – (R) – DAC channel 16 write timeout error

[14...12] – (reserved – read as 0)

[11...0] – (R/W) – DAC value channel 16

STATUS 1 – Input Buffer Status (0x70)

31 – (R) – data buffer ready for input

30 – (R) – data buffer input paused

29 – (R) – reserved (read as ‘0’)

28 – (R) – data buffer empty

27 – (R) – data buffer full

[26...16] – (R) – data buffer word count

[15...0] – (reserved)

STATUS 2 – Build Buffer Status (0x74)

- [31...29] – reserved (read as ‘0’)
- 28 – (R) – data buffer ‘A’ empty
- 27 – (R) – data buffer ‘A’ full
- [26...16] – (R) – data buffer ‘A’ word count
- [15...13] – reserved (read as ‘0’)
- 12 – (R) – data buffer ‘B’ empty
- 11 – (R) – data buffer ‘B’ full
- [10...0] – (R) – data buffer ‘B’ word count

STATUS 3 – Output Buffer Status (0x78)

- [31...30] – reserved (read as ‘0’)
- 29 – (R) – data buffer ‘A’ empty
- 28 – (R) – data buffer ‘A’ full
- [27...16] – (R) – data buffer ‘A’ word count
- [15...14] – reserved (read as ‘0’)
- 13 – (R) – data buffer ‘B’ empty
- 12 – (R) – data buffer ‘B’ full
- [11...0] – (R) – data buffer ‘B’ word count

STATUS 4 – (spare) (0x7C)

- [31...0] – reserved

AUXILIARY 1 – (spare) (0x80)

[31...0] – reserved

TRIGGER CONTROL (0x84) – Under normal conditions, a trigger received by the module is counted and sent to the ADC Processing FPGA. A trigger is considered acknowledged when data associated with it has been transferred to the Control FPGA. To ensure that data buffers in the ADC Processing FPGA are not overrun, the number of unacknowledged triggers is continuously compared to levels (MAX1, MAX2) set by the user. These levels are based on the parameters (processing mode, # samples in window) loaded into the ADC Processing FPGA. Two module based methods to avert buffer overflow and the resulting data corruption are available to the user.

In the first method, module BUSY is asserted when the number of unacknowledged triggers  $\geq$  MAX1. BUSY propagates back to the Trigger Supervisor. The Trigger Supervisor does not deliver triggers to the system while BUSY is asserted. Level MAX1 is based on the BUSY propagation delay, ADC Processing parameters, and the minimum trigger interval parameter of the Trigger Supervisor.

In the second method, triggers received by the module are not sent to the ADC Processing FPGA when the number of unacknowledged triggers  $\geq$  MAX2. Level MAX2 is based on the ADC Processing parameters. Not sending triggers to the FPGA Processing FPGA can cause a loss of synchronization among system components, but will prevent data corruption within the module. (See Auxiliary Scaler 2 for a count of lost triggers.)

Both methods can be applied together, with  $\text{MAX2} \geq \text{MAX1}$ .

31 – (R/W) – enable trigger stop when number of unacknowledged triggers  $\geq$  MAX2

[23...16] – (R/W) – level MAX2

15 – (R/W) – enable module busy assertion when number of unacknowledged triggers  $\geq$  MAX1

[7...0] – (R/W) – level MAX1

#### TRIG21 DELAY (0x88)

[31...12] – reserved

[11...0] – (R/W) – Delay from soft TRIG2 to generated TRIG1 (4 ns/count)

RAM Address Register (0x8C) – The RAM is organized as two 36-bit words with a common address. Auxiliary VME access (R/W) to the RAM is provided through a pair of 32 bit data registers (RAM 1, RAM 2). Note that bits 35 – 32 of each RAM word are not

accessible through VME. During data flow operations, these bits carry event marker tags (header, trailer).

31 – increment address after access (R/W) of RAM 1 Data Register

30 – increment address after access (R/W) of RAM 2 Data Register

[29...21] – reserved (read as 0)

[19...0] – RAM address

RAM 1 Data Register (0x90)

[31...0] – RAM data word bits 67 – 36 (32 bits)

RAM 2 Data Register (0x94)

[31...0] – RAM data word bits 31 – 0 (32 bits)

(PROM Registers 1 and 2 are used for FPGA configuration over VME.)

PROM Register 1 (0x98)

31 – READY – (R) – configuration state machine is available to accept command (i.e. no configuration process is currently executing).

[30...8] – reserved (read as 0)

[7...0] – configuration OPCODE

PROM Register 2 (0x9C)

[31...0] – PROM ID – (R) response to specific OPCODE write to PROM reg 1.

BERR Module Count (0xA0)

[31...0] – BERR count (driven by module to terminate data transmission)

BERR Total Count (0xA4)

[31...0] – BERR count (as detected on bus)

Auxiliary Scaler 1 (0xA8)

[31...0] – Total word count from ADC Processing FPGA

Auxiliary Scaler 2 (0xAC)

[31...0] – Count of triggers lost because number of unacknowledged triggers  $\geq$  MAX2. (See TRIGGER CONTROL register.)

Auxiliary Scaler 3 (0xB0)

[31...0] – Event header word count from ADC Processing FPGA

TRIGGER 2 SCALER (0xB4)

[31...0] – (R) – **Trigger 2** count

31 – (W) – write ‘1’ to reset count

Auxiliary Scaler 5 (0xB8)

[31...0] – Event trailer word count from ADC Processing FPGA

SYNC RESET SCALER (0xBC)

[31...0] – (R) – **Sync Reset** count

31 – (W) – write ‘1’ to reset count

Module Busy Level (0xC0)

[31] – Force module busy

[30...20] – reserved

[19...0] – Busy Level (eight-byte words)

(External RAM word count > Busy Level  $\rightarrow$  module busy = 1)

NOTE: At the system level, an asserted module busy signal should be used to prevent further triggers from being sent to the modules. By setting the Busy Level



well below the memory capacity (1,048,576 eight-byte words), triggers already in the distribution pipeline can still be accepted by the module.

To prevent data corruption in the module if this global trigger control is NOT in place, the module itself will BLOCK input triggers when the number of 8-byte words in memory is within 12K (1.2%) of memory capacity (i.e. 1,036,288).

When global trigger control IS in place, it is important to set Busy Level significantly LESS than 1,036,288 so that local trigger blocking will NEVER occur. (Local trigger blocking will result in a loss of system synchronization.) The safe maximum Busy Level depends on data size per trigger (i.e. mode of ADC processing), but a Busy Level equal to ~87.5% of memory capacity (i.e. 917,504 = 0xE0000) should be adequate for all circumstances.

Generate Event Header Word (0xC4) (for debug)

[31...0] – (W) – Event Header Word

Generate Event Data Word (0xC8) (for debug)

[31...0] – (W) – Event Data Word

Generate Event Trailer Word (0xCC) (for debug)

[31...0] – (W) – Event Trailer Word

MGT STATUS (0xD0)

0 – (R) – lane 1 up (GTX1)

1 – (R) – lane 2 up (GTX1)

2 – (R) – channel up (GTX1)

3 – (R) – hard error (GTX1)

4 – (R) – soft error (GTX1)

5 – (R) – lane 1 up (GTX2)

6 – (R) – lane 2 up (GTX2)

7 – (R) – channel up (GTX2)

8 – (R) – hard error (GTX2)

- 9 – (R) – soft error (GTX2)
- 10 – (R) – SUM DATA VALID
- 11 – (R) – MGT RESET ASSERTED
- [31...12] – (R) - Reserved

MGT CONTROL (0xD4)

- 0 – **RELEASE MGT RESET** (0 = reset MGT, 1 = release reset)
- 1 – Data Type to CTP (0 = counting sequence, 1 = front-end data)
- 2 – Enable Data Alignment on Sync Reset occurrence
- [31...3] – Reserved

RESERVED (2 registers) (0xD8 – 0xDC)

SCALER CONTROL (0xE0) – See SCALERS (0x300 – 0x340)

- 0 – (R/W) – Enable all scalers to count (1 = enable, 0 = disable)
- 1 – (W) – Latch all scalers. Write ‘1’ to simultaneously transfer all 17 scaler counts to registers for readout.
- 2 – (W) – Reset all scalers. Write ‘1’ to simultaneously reset all 17 scaler counts to zero.
- [3 – 31] – (reserved)

BOARD SERIAL NUMBER 0 (0xE4)

- [31...24] – (R) – board serial number byte 0
- [23...16] – (R) – board serial number byte 1
- [15...8] – (R) – board serial number byte 2
- [7...0] – (R) – board serial number byte 3

BOARD SERIAL NUMBER 1 (0xE8)

[31...24] – (R) – board serial number byte 4

[23...16] – (R) – board serial number byte 5

[15...8] – (R) – board serial number byte 6

[7...0] – (R) – board serial number byte 7

#### BOARD SERIAL NUMBER 2 (0xEC)

[31...24] – (R) – board serial number byte 8

[23...16] – (R) – board serial number byte 9

[15...8] – (R) – board serial number byte 10

[7...0] – (R) – board serial number byte 11

SCALER INSERTION INTERVAL (0xF0) - Data from the SCALERS defined below (0x300 – 0x340) may be inserted into the readout data stream at regular event count intervals. The interval is specified in multiples of the event BLOCK SIZE. When the interval is ZERO (the default condition), there is NO insertion of scaler data into the data stream. When programmed for a non-zero interval, the current scaler values are appended to the last event of the appropriate BLOCK of events. The current Trigger 1 count is also inserted as the 18<sup>th</sup> scaler. Note that the scalers are **NOT** reset after their values are captured.

Example: Interval = 10 means that every 10<sup>th</sup> block of events will have the integrated scaler data appended to it.

(See the document FADC V2 Data Format for information on identifying scaler data words in an event.)

The scalers may ALSO be inserted into the data stream when a FORCE BLOCK TRAILER is done by the user. A simultaneous write of ‘1’ to bit 22 and bit 23 of the CSR (0x4) accomplishes this. The scaler values are those at the time of the last trigger’s occurrence.

[15...0] - (R/W) – N (in BLOCKS of events); every N<sup>th</sup> block of events has integrated scaler data appended to the last event in the block.

[31...16] – (reserved)

#### SUM THRESHOLD (History buffer) (0xF4)

31 – (R) – sum data READY for readout if value = ‘1’

[30...16] – (R/W) – reserved (read as ‘0’)

[15...0] – (R/W) – sum threshold value for data capture

#### SUM DATA (History buffer) (0xF8)

31 – (W) – writing a ‘1’ ARMs History buffer for data capture

[30...16] – (R) – reserved (read as ‘0’)

[15...0] – (R) – sum data sample

#### SYSTEM MONITOR (0xFC)

[31...22] – (R) – FPGA auxiliary voltage (2.5V) (common)

$vaux = (((float)((reg\_value \gg 22) \& 0x3FF))/1024.0) * 3.0;$

21 – (R) – reserved (read as ‘0’)

[20...11] – (R) – FPGA core voltage (1.0V) (common)

$vint = (((float)((reg\_value \gg 11) \& 0x3FF))/1024.0) * 3.0;$

10 – (R) – reserved (read as ‘0’)

[9...0] – (R) – CTRL\_FPGA temperature (°C)

$temp\_ctrl = (((float)(reg\_value \& 0x3FF)) * 503.975/1024.0) - 273.15;$

---

SCALER Registers (0x300 – 0x340) (R)

SCALER[0] – (0x300) - input channel 0 count

SCALER[1] – (0x304) - input channel 1 count

SCALER[2] – (0x308) - input channel 2 count

SCALER[3] – (0x30C) - input channel 3 count

SCALER[4] – (0x310) - input channel 4 count

SCALER[5] – (0x314) - input channel 5 count

SCALER[6] – (0x318) - input channel 6 count

SCALER[7] – (0x31C) - input channel 7 count

SCALER[8] – (0x320) - input channel 8 count

SCALER[9] – (0x324) - input channel 9 count

SCALER[10] – (0x328) - input channel 10 count

SCALER[11] – (0x32C) - input channel 11 count

SCALER[12] – (0x330) - input channel 12 count

SCALER[13] – (0x334) - input channel 13 count

SCALER[14] – (0x338) - input channel 14 count

SCALER[15] – (0x33C) - input channel 15 count

TIME COUNT – (0x340) - timer (each count represents 2048 ns)

---

---

SYSTEM TEST Registers (0x400 – 0x410)

TEST BIT REGISTER (0x400)

- 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) – clock\_250 counter status (1 = counting, 0 = not counting)
- [16 - 31] – (R) – reserved (read as ‘0’)

CLOCK\_250 COUNT REGISTER (0x404)

- 0 – (W) – Write ‘0’ resets the counter. Write ‘1’ initiates 20us counting interval.
- [31 - 0] – (R) – CLK\_250 counter value. (Should be 5000 after count interval.)

SYNC\_IN\_P0 COUNT REGISTER (0x408)

- 0 – (W) – Write ‘0’ resets the counter.
- [31 - 0] – (R) – SYNC\_IN\_P0 counter value.

TRIG1\_IN\_P0 COUNT REGISTER (0x40C)

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

[31 - 0] – (R) – TRIG1\_IN\_P0 counter value.

TRIG2\_IN\_P0 COUNT REGISTER (0x410)

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

[31 - 0] – (R) – TRIG2\_IN\_P0 counter value.

---

---

## DEBUG Registers (0x500 – 0x508)

Three registers enable the tracking of the state machine used for 2eSST VME data transmission from the module. State values are saved in a FIFO buffer when recording is enabled and a change in the state value occurs. While enabled the buffer is continuously updated so that the last ‘N’ state values are present in the buffer; ‘N’ is programmed by the user. Recording must be disabled before user readout of the state value buffer (FIFO) can begin. The actual number of valid saved state values is available to the user.

### STATE LEVEL (0x500)

[31...16] – (R) – reserved (read as ‘0’)

[15...9] – (R/W) – reserved

[8...0] – (R/W) – number of state values to capture (up to 511). Default value is 500 (loaded on power up or hard reset of module).

### STATE CSR (0x504)

31 – (R/W) – ‘1’ - buffer armed for state value capture; ‘0’ – buffer not armed

[30...28] – (R) – reserved (read as ‘0’)

[27] – (R) – state buffer full

[26] – (R) – state buffer empty

[25...9] – (R) – reserved (read as ‘0’)

[8...0] – (R) – number of valid state values stored in buffer

### STATE VALUE (0x508)

[31...17] – (R) – reserved (read as ‘0’)

[16...0] – (R) – next 2eSST state value



## MOLLER Registers (0x600 – 0x64C)

### Moller Configuration Registers (16)

Name [VME ADDRESS]	Width (Bits)	Access	Power Up Values (hex)	Function
CONFIG1 [0x600]	32	R/W	0000	4-0 SL1 Pulse Delay 7-5 SL1 1 Shot Width 12-8 SL2 Pulse Delay 15-13 SL2 1 Shot Width 20-16 SL3 Pulse Delay 23-21 SL3 1 Shot Width 28-24 SL4 Pulse Delay 31-29 SL4 1 Shot Width
CONFIG2 [0x604]	32	R/W	0000	4-0 SR1 Pulse Delay 7-5 SR1 1 Shot Width 12-8 SR2 Pulse Delay 15-13 SR2 1 Shot Width 20-16 SR3 Pulse Delay 23-21 SR3 1 Shot Width 28-24 SR4 Pulse Delay 31-29 SR4 1 Shot Width
CONFIG3 [0x608]	32	R/W	0000	4-0 CL1 Pulse Delay 7-5 CL1 1 Shot Width 12-8 CL2 Pulse Delay 15-13 CL2 1 Shot Width 20-16 CL3 Pulse Delay 23-21 CL3 1 Shot Width 28-24 CL4 Pulse Delay 31-29 CL4 1 Shot Width
CONFIG4 [0x60C]	32	R/W	0000	4-0 CR1 Pulse Delay 7-5 CR1 1 Shot Width 12-8 CR2 Pulse Delay 15-13 CR2 1 Shot Width 20-16 CR3 Pulse Delay 23-21 CR3 1 Shot Width 28-24 CR4 Pulse Delay 31-29 CR4 1 Shot Width

CONFIG5 [0x610]	32	R/W	0000	4-0 CL Pulse Delay 7-5 CL 1 Shot Width 12-8 CR Pulse Delay 15-13 CR 1 Shot Width
CONFIG6 [0x614]	32	R/W	0000	10-0 CL PreScaler 21-11 CR PreScaler  30 — 1 set CL PreScaler. 0 to run 31 — 1 set CR PreScaler. 0 to run
CONFIG7 [0x618]	32	R/W	0000	10-0 CL and CR PreScaler 18-11 Coincident Window Width 23-19  30 ---- 1 Reset Trigger Counter 31— 1 set CL and CR PreScaler. 0 to run
CONFIG8 [0x61C]	32	R/W	0000	Select History Buffer Trigger Bits 0 CR 1 CL 2 CR1 3 CR2 4 CR3 5 CR4 6 SR1 7 SR2 8 SR3 9 SR4 10 CL1 11 CL2 12 CL3 13 CL4 14 SL1 15 SL2 16 SL3 17 SL4  <b>18 Falling edge Arm History Buffer</b>
				Config 9-16 Bit Mapping  0 Invert Select CR1 1 Invert Select CR2 2 Invert Select CR3 3 Invert Select CR4

				4 Invert Select CL1 5 Invert Select CL2 6 Invert Select CL3 7 Invert Select CL4  8 Invert Select CR1 9 Invert Select CR2 ----- 10 AND Select CR1 11 AND Select CR2 12 AND Select CR3 13 AND Select CR4  14 AND Select CL1 15 AND Select CL2 16 AND Select CL3 17 AND Select CL4  18 AND Select CR1 19 AND Select CR2 ----- 20 OR Select CR1 21 OR Select CR2 22 OR Select CR3 23 OR Select CR4  24 OR Select CL1 25 OR Select CL2 26 OR Select CL3 27 OR Select CL4  28 OR Select CR1 29 OR Select CR2
CONFIG9 [0x620]	32	R/W	0000	SCM 0 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select
CONFIG10 [0x624]	32	R/W	0000	SCM 1 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select
CONFIG11 [0x628]	32	R/W	0000	SCM 2 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select

CONFIG12 [0x62C]	32	R/W	0000	SCM 3 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select
CONFIG13 [0x630]	32	R/W	0000	SCM 4 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select
CONFIG14 [0x634]	32	R/W	0000	SCM 5 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select
CONFIG15 [0x638]	32	R/W	0000	SCM 6 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select
CONFIG16 [0x63C]	32	R/W	0000	SCM 7 Config 9-0 Invert Select 19-10 AND Select 29-20 OR Select

MOLLER SCALER FIFO (0x640) – For each Moller scaler trigger, a header word and 18 scaler values are stored in a 512 deep FIFO. The SET AVAILABLE flag is asserted indicating to the user (by polling or interrupt) that the scaler data is ready to be read out. When the 19 data words of the set are read out, the SET AVAILABLE flag is de-asserted if no other complete sets have been written to the FIFO.

Header word:

31 – (R) – ‘1’ when helicity signal *high*, ‘0’ when helicity signal *low*

30 – (R) – ‘1’ when helicity signal *low*, ‘0’ when helicity signal *high*

[29...0] – (R) – helicity interval count

Scaler words (18):

[31...0] – (R) – scaler count

### MOLLER CONTROL (0x644)

0 – (R/W) – enable interrupt generation when scaler SET AVAILABLE flag is asserted

1 – (R/W) – enable interrupt generation when data BLOCK READY flag is asserted

[31...2] – (R/W) – reserved

### MOLLER STATUS (0x648)

0 – (R) – scaler SET AVAILABLE flag

1 – (R) – data BLOCK READY flag

2 – (R) – trigger FIFO empty flag

3 – (R) – trigger FIFO full flag

4 – (R) – history data READY flag

[7...5] – (R) – reserved (read as ‘0’)

[12...8] – (R) – number of complete Moller scaler sets stored

[15...13] – (R) – reserved (read as ‘0’)

[25...16] – (R) – number of words stored in Moller scaler FIFO

[26] – (R) – scaler FIFO empty flag

[27] – (R) – scaler FIFO full flag

[29...28] – (R) – reserved (read as ‘0’)

30 – (R) – Moller data acquisition ACTIVE

31 – (R/W) – GO MOLLER – Write ‘1’ to request start of data acquisition, ‘0’ to request stop of data acquisition. Actual acquisition will always start or stop on the falling edge of the helicity flip signal. This assures that data will always be acquired for an integral number of helicity intervals.

MOLLER TRIGGER FIFO (0x64C) – For each Moller data trigger the trigger information is stored as a single 32-bit word in a 1024 deep FIFO. A read of this address will return a

single trigger data word for an event. If the event block level is BLOCK\_SIZE, that number of trigger data words should be read from the FIFO in conjunction with the block event data readout.

[31...4] – (R) – event number

3 – (R) – helicity state

2 – (R) – CL trigger

1 – (R) – CR trigger

0 – (R) – CL and CR trigger

MOLLER HISTORY FIFO (0x650) – On the rising edge of any CL, CR, SL, SR signals selected by the user (see History Buffer Bit Select – 0x61C), all 18 logic signal states are recorded every 4 ns for a period of 128 ns (32 samples). This assists the user in adjusting the timing of signals used in the coincidence logic.

[31...18] – reserved (read as 0)

17 – state of logic signal SL4

16 – state of logic signal SL3

15 – state of logic signal SL2

14 – state of logic signal SL1

13 – state of logic signal CL4

12 – state of logic signal CL3

11 – state of logic signal CL2

10 – state of logic signal CL1

9 – state of logic signal SR4

8 – state of logic signal SR3

7 – state of logic signal SR2

6 – state of logic signal SR1

5 – state of logic signal CR4

4 – state of logic signal CR3

3 – state of logic signal CR2

2 – state of logic signal CR1

1 – state of logic signal CL

0 – state of logic signal CR

## ADC PROCESSING FPGA ADDRESS MAP:

Control Bus Memory Map for FADC FPGA

**Failure to adhere to Min values can result in unpredictable results**

Name [VME ADDRESS]	Width (Bits)	Quantity	Access	Primary Address (Secondary Address)	Power Up Values (hex)	Function
STATUS0 [0x100]	16	1	R	0x0000 (---)		Bits 14 to 0: Code Version Bit 15: 1= Command can be sent to AD9230
STATUS1 [0x104]	16	1	R	0x0001 (---)		15 : 1 Done all Trig received 11-0 : TRIGGER NUMBER
STATUS2 [0x108]	16	1	R	0x0002 (---)		Monitored Pedestal 15 → Sum is valid 14 → 0 Sum OK. 1 One or more is out of bound 13-0 → Sum
CONFIG1 [0x10C]	16	1	R/W	0x0003 (---)	0000	Bit 0-2 (old code process mode):  Bit 3: 1:Run Bit 5-4 : Max Number of Pulses in Mode 10 and 9  Bit 7: Test Mode (play Back).  Bit 8: 0 → mode 9 1 → mode 10  11-10 NSAT  15→ Request Sum of Pedestal for monitoring purpose
CONFIG2 [0x110]			R/W	0x0004 (---)	0000	When 1 ADC values = 0 Bit 0 → ADC 0 Bit 1 → ADC 1 Bit 2 → ADC 2 Bit 3 → ADC 3 Bit 4 → ADC 4



						Bit 5 → ADC 5 Bit 6 → ADC 6 Bit 7 → ADC 7 Bit 8 → ADC 8 Bit 9 → ADC 9 Bit 10 → ADC 10 Bit 11 → ADC 11 Bit 12 → ADC 12 Bit 13 → ADC 13 Bit 14 → ADC 14 Bit 15 → ADC 15
CONFIG4 [0x114]	16	1	R/W	0x0005	0000	15 : <b>Sync Disable. 1 not allow SYNC signal to Reset ADC</b>  7 => rising edge write to AD9230 ADC 6 => 1 write to all ADC  5 => 0 write to AD9230 1 read from AD9230 . Data is at Stat 4 => 1 Reset ADC 3..0 => Select ADC to write to
CONFIG5 [0x118]	16	1		0x0006	0000	15..8 => Registers inside AD9230 7..0 => Data to write to register.
PTW [0x11C]	9	1	R/W	0x0007 (---)	0000	<b>PTW + 1</b> number of ADC sample to include in trigger window. PTW = Trigger Window (ns) * 250 MHz. <b>Minimum is 6.</b>
PL [0x120]	11	1	R/W	0x0008 (---)	0000	Number of sample back from trigger point. PL = Trigger Window(ns) * 250MHz
NSB [0x124]	4	1	R/W	0x0009 (---)	0000	3..0: Read Back Path NSB Number of sample before trigger point to include in data processing. This include the trigger Point. When NSB bit 3 is 1: <b>NSA has to be &gt; NSB bits 1,0 by at least 4 =&gt; NSA – (NSB bits 1,0) ≥ 3</b>

NSA [0x128]	15	1	R/W	0x000A (---)	0000	8..0: Read Back Path NSA Number of sample after trigger point to include in data processing. <b>Minimum is 2</b>
TET [0x12C - 0x148]	12	16	R/W	0x000B - 0x001A	0000	Read Out Energy Threshold.
CONFIG6 (Monitored Pedestal Sum) [0x14C]	16	1	R/W	0x001B	0000	13-10 MNPED : The number of ADC sample to sum up is MNPED + 1. <b>Min is 4</b> 9-0 PMaxPed : When an ADC Samples is greater than this, bit 14 of the 14 bits (13-0) Sum will be set.
CONFIG7 (Read Back Pedestal Sum) [0x150]	16	1	R/W	0x001C	0000	13-10 NPED : The number of ADC sample is NPED + 1 9-0 MaxPed
Test Wave Form [0x154]	16	1	R/W	0x001D	0000	Write to PPG. Read should immediately follow write.
Trigger Path Pedestal Subtract Value [0x158 - 0x194]	12	16	R/W	0x001E- 0x002D	0000	Subtract from ADC(0-15) Count before processing for trigger path
Config 3 Trigger Path Threshold 1 [0x198]	16	1	R/W	0x002E	0000	CL1-4, CR1-4 Trigger Threshold
Config 8 Trigger Path Threshold 2 [0x19C]	13	1	R/W	0x002F	0000	15-13 CL, CR Added Length. Added Length is 2 more than bits15- 13. 0=2, 1=3, 2=4  0-12 SL1-4, SR1-4 Trigger Threshold
Config 9 Trigger Path Threshold 3 [0x1A0]	16	1	R/W	0x0030	0000	CL, CR Trigger Threshold

STATUS 3 [0x1A4]	16	1	R	0x0031		FPGA core temp (DieTemp)
STATUS 4 [0x1A8]	16	1	R	0x0032		7-0 Result of ADC register readback

## Appendix 1 - Hall D FADC250 Data Format (9/16)

### Data Type List

- 0 – block header
- 1 – block trailer
- 2 – event header\*
- 3 – trigger time
- 4 – window raw data
- 5 – 8 – (reserved)
- 9 – pulse parameters\*\*
- 10 – 11 – (reserved)
- 12 – scaler data
- 13 – (reserved)
- 14 – data not valid (empty module)
- 15 – filler (non-data) word

\* reformatted data type

\*\* new data type – information from previous data types 7, 8, 10  
(data types not marked with \* or \*\* remain unchanged)

### 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 raw ADC samples and pulse parameters. Any number of Data Type Continuation words may follow a Data Type Defining word. The scaler data type is an exception. It specifies the number of 32-bit data words that follow.

### Data Types

**Block Header** (0) – Word 1 indicates the beginning of a block of events. Optional continuation Word 2 contains ADC processing parameters.

#### Word 1:

- (31) = 1
- (30 – 27) = 0
- (26 – 22) = slot number (set by VME64x backplane)
- (21 – 18) = module ID ('1' for FADC250)
- (17 – 8) = event block number
- (7 – 0) = number of events in block

Word 2:

- (31) = 0
- (30 – 29) = 0
- (28 – 18) = PL (# samples before trigger point for processing to begin)
- (17 – 9) = NSB (# samples before threshold crossing to include in processing)  
(pulse modes)
- (8 – 0) = NSA (# samples after threshold crossing to include in processing)  
(pulse modes)

**Block Trailer** (1) – indicates the end of a block of events.

- (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.

- (31) = 1
- (30 – 27) = 2
- (26 – 22) = slot number (set by VME64x backplane)
- (21 – 12) = trigger time (bits 9 – 0 (see below))
- (11 – 0) = trigger number

**Trigger Time** (3) – time of trigger occurrence relative to the most recent global reset. Time in the ADC data processing chip is measured by a 48-bit counter that is clocked by the 250 MHz system clock. The six bytes of the trigger time

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

are reported in two words (Type Defining + Type Continuation). (Both Words or Word 2 alone may be suppressed from readout by the user.)

Word 1:

- (31) = 1
- (30 – 27) = 3
- (26 – 24) =  $T_C$  bits 2 – 0 (duplicated in Word 2)
- (23 – 16) =  $T_D$
- (15 – 8) =  $T_E$
- (7 – 0) =  $T_F$

Word 2:

- (31) = 0
- (30 – 24) = reserved (read as 0)
- (23 – 16) =  $T_A$

$$\begin{aligned}(15 - 8) &= T_B \\ (7 - 0) &= T_C\end{aligned}$$

**Window Raw Data** (4) – raw ADC data samples for the trigger window. The first word identifies the channel number and window width. Multiple continuation words contain two samples each. The earlier sample is stored in the most significant half of the continuation word. Strict time ordering of the samples is maintained in the order of the continuation words. A *sample not valid* flag may be set for any sample; e.g. the last reported sample is not valid when the window consists of an odd number of samples.

Word 1:

$$\begin{aligned}(31) &= 1 \\ (30 - 27) &= 4 \\ (26 - 23) &= \text{channel number (0 - 15)} \\ (22 - 12) &= \text{reserved (read as 0)} \\ (11 - 0) &= \text{window width (in number of samples)}\end{aligned}$$

Words 2 - N:

$$\begin{aligned}(31) &= 0 \\ (30) &= \text{reserved (read as 0)} \\ (29) &= \text{sample } x \text{ not valid} \\ (28 - 16) &= \text{ADC sample } x \text{ (includes overflow bit)} \\ (15 - 14) &= \text{reserved (read as 0)} \\ (13) &= \text{sample } x + 1 \text{ not valid} \\ (12 - 0) &= \text{ADC sample } x + 1 \text{ (includes overflow bit)}\end{aligned}$$

**Pulse Parameters** (9) – computed pulse parameters for detected pulses in a channel. The first word identifies the channel number, event number within the block, and pedestal information for the window. Multiple continuation word *pairs* contain information about the pulses detected. For a channel with hits detected:

Word 1: Channel ID and Pedestal information (reported *once* for a channel with hits)

$$\begin{aligned}(31) &= 1 \\ (30 - 27) &= 9 \\ (26 - 19) &= \text{event number within block (1 - 255)} \\ (18 - 15) &= \text{channel number (0 - 15)} \\ (14) &= \text{pedestal quality} \\ (13 - 0) &= \text{pedestal sum}\end{aligned}$$

Word 2: Integral of first pulse in window

$$\begin{aligned}(31) &= 0 \\ (30) &= 1 \\ (29 - 12) &= \text{18-bit sum of raw samples that constitute the pulse data set} \\ (11 - 9) &= \text{integral quality}\end{aligned}$$

(8 – 0) = number of samples within NSA that the pulse is above threshold

Word 3: Time of first pulse in window

(31) = 0

(30) = 0

(29 – 21) = coarse time (4 ns/count)

(20 – 15) = fine time (0.0625 ns/count)

(14 – 3) = pulse peak

(2 – 0) = time quality

Words 2 and 3 are repeated for *each additional pulse* found in the window for the channel.

**Scaler Header** (12) – indicates the beginning of a block of scaler data words. The number of scaler data words that will immediately follow it is provided in the header. The scaler data words are 32 bits wide and so have no bits available to identify them. Currently there are 18 scaler words reported: 16 from individual channels, a timer, and a trigger count. The scalers and time represent values recorded at the indicated trigger count. Scaler data must be enabled into the data stream by the user.

(31) = 1

(30 – 27) = 12

(26 – 6) = reserved (read as 0)

(5 – 0) = number of scaler data words to follow (18 = current)

**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

## Processing Modes

Two processing modes are supported.

Mode 9: reports pulse parameters (type 9) for hits

Mode 10: reports raw window samples (type 4) and pulse parameters (type 9) for hits

## Readout Format

Standard: For a block of  $N$  events

```
Block Header
  Event 1 Header
    Trigger Time 1
    Trigger Time 2
    Data words (event 1)
  Event 2 Header
    Trigger Time 1
    Trigger Time 2
    Data words (event 2)
  -----
  -----
  Event  $N$  Header
    Trigger Time 1
    Trigger Time 2
    Data words (event  $N$ )
Block Trailer
```

Intermediate compression: For a block of  $N$  events and data in events  $I, J, K$

```
Block Header
  Event 1 Header (always)
  Event  $I$  Header
    Data words (event  $I$ )
  Event  $J$  Header
    Data words (event  $J$ )
  Event  $K$  Header
    Data words (event  $K$ )
Block Trailer
```



Full compression: For a block of  $N$  events and data in events  $I, J, K$

Block Header  
    Event 1 Header  
        Data words (event  $I$ )  
        Data words (event  $J$ )  
        Data words (event  $K$ )  
Block Trailer

## Notes about the Readout Format

For the *Full compression* readout format the trigger time in the Event 1 Header enables the detection of synchronization loss among modules. If synchronization is lost (e.g. module misses a trigger), it will likely not be discovered until the next *Sync Event* (TS pauses triggers and test for synchronization). Synchronization loss thus requires that multiple blocks of events be discarded. Since the *Standard* and *Intermediate compression* readout formats have the trigger time available for each event (within the Event Header for *Intermediate compression* format), the exact point of synchronization loss can be discovered in off-line analysis. Events can be re-aligned, with a loss of only one event for each trigger missed by a module.

For simplicity of design the ADC Processing FPGA always transmits data to the Control FPGA in *Standard* format (less Block Header or Trailer words). Based on the readout format selected by the user, the Control FPGA suppresses data words that are not required to be read out.

When using processing mode 10, raw window samples (type 4 words) will precede pulse parameters (type 9 words) for a given channel. This is done for performance reasons. Because type 4 words do not have an event number in their definition, it is recommended that the *Standard or Intermediate compression* readout formats be used for this mode.