Using the FADC V2 Module **(6/20/12)**

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 – HITSUM processing registers (64 long words)

300-3FF – SCALER registers (64 long words)

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

500-FFF – Reserved (704 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 by setting 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 – (R) – PLL 1 Locked **(bits 5-10 not currently implemented)**

6 – (R) – PLL 2 Locked

7 – (R) – PLL 3 Locked

8 – (R) – PLL 1 Loss of Lock Occurred

9 – (R) – PLL 2 Loss of Lock Occurred

10 – (R) – PLL 3 Loss of Lock Occurred

11 – (R) – Data FIFO 1 (channels 1 – 16) Empty Flag Asserted

12 – (R) – Data FIFO 1 (channels 1 – 16) Almost Empty Flag Asserted

13 – (R) – Data FIFO 1 (channels 1 – 16) Half Full Flag Asserted

14 – (R) – Data FIFO 1 (channels 1 – 16) Almost Full Flag Asserted

15 – (R) – Data FIFO 1 (channels 1 – 16) Full Flag Asserted

(bits 16-25 not implemented)

16 – (R) – Data FIFO 2 (channels 9 – 16) Empty Flag Asserted

17 – (R) – Data FIFO 2 (channels 9 – 16) Almost Empty Flag Asserted

18 – (R) – Data FIFO 2 (channels 9 – 16) Half Full Flag Asserted

19 – (R) – Data FIFO 2 (channels 9 – 16) Almost Full Flag Asserted

20 – (R) – Data FIFO 2 (channels 9 – 16) Full Flag Asserted

21 – (R) – HITSUM FIFO Empty Flag Asserted

22 – (R) – HITSUM FIFO Almost Empty Flag Asserted

23 – (R) – HITSUM FIFO Half Full Flag Asserted

24 – (R) – HITSUM FIFO Almost Full Flag Asserted

25 – (R) – HITSUM FIFO Full Flag Asserted

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)

29 – (W) – Pulse Soft Trigger (if CTRL[7] = 1)

30 – (W) – Pulse Soft Reset

31 – (W) – Pulse Hard Reset

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
- 1 = Front Panel Connector (synchronized internally)
- 2 = P0 Connector (VXS)
- 3 = P0 Connector (VXS) (synchronized internally)
- 4 – (not used)
- 5 – (not used)
- 6 = VME (software generated)
- 7 = Module Internal Logic (synchronized internally)

7 – (R/W) – Enable Soft Trigger

[10…8] – (R/W) – Sync Reset Source Select
- 0 = Front Panel Connector
- 1 = Front Panel Connector (synchronized internally)
- 2 = P0 Connector (VXS)
- 3 = P0 Connector (VXS) (synchronized internally)
- 4 – (not used)
- 5 – (not used)
- 6 = VME (software generated)
- 7 = no source

11 – (R/W) – Enable Soft Sync Reset

12 – (R/W) – Select Live Internal Trigger to Output

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

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

(bits 15-17 not implemented)

15 – (R/W) – Enable P2 Trigger Output

16 – (R/W) – Enable Readout of ADC channels 1 – 8

17 – (R/W) – Enable Readout of ADC channels 9 – 16

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

19 – (R/W) – Enable Error Interrupt **(not implemented)**

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 – (R/W) – Bypass External RAM **(not implemented)**

25 – (R/W) – Enable Debug Mode

26 – 27 – spare

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

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

30 – spare

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 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) – Select data for readout (0 = channels 1 – 8, 1 = channels 9 – 16)

(streaming mode only) **(not implemented)**

6 – 7 – (not used)

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

9 – 15 – (not used)

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

16 – (R/W) – Transfer data: input FIFO → build FIFO **(not implemented)**

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

30 – (R/W) – Disable external FIFO output (channels 1 – 8) **(not implemented)**

31 – (R/W) – Disable external FIFO output (channels 9 – 16) **(not implemented)**

BLOCK SIZE (0x10)

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

[31…16] – (not used)

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)

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

24 – 31 – (not used)

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 – 22 – (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:

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

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)

[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)

[7…0] – (R/W) – trigger hold off delay (4 ns per count)

RESET CONTROL (0x2C)

0 – (W) – Hard reset – Control FPGA

1 – (W) – Hard reset – ADC processing FPGA 1 (channels 1 – 16)

(bits 2, 3, 6, 7, 9, 10 not implemented)

2 – (W) – Hard reset – ADC processing FPGA 2 (channels 9 – 16)

3 – (W) – Hard reset – HITSUM FPGA

4 – (W) – Soft reset – Control FPGA

5 – (W) – Soft reset – ADC processing FPGA 1 (channels 1 – 16)

6 – (W) – Soft reset – ADC processing FPGA 2 (channels 9 – 16)

7 – (W) – Soft reset – HITSUM FPGA

8 – (W) – Reset – ADC data FIFO 1 (channels 1 - 8)

9 – (W) – Reset – ADC data FIFO 2 (channels 9 - 16)

10 – (W) – Reset – HITSUM FIFO

11 – (W) – Reset – DAC (all channels)

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

13 – 31 – (not used)

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] – (not used)

BLOCK COUNT – (0x38)

[31…20] – not used

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

BLOCK FIFO COUNT – (0x3C)

[31…6] – not used

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

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

[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

INTERNAL TRIGGER COUNT (0x44)

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

31 – (W) – reset count

EXTERNAL RAM WORD COUNT (0x48)

[31…22] – not used (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 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 3_4 – DAC channels 3,4 (0x54)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 5_6 – DAC channels 5,6 (0x58)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 7_8 – DAC channels 7,8 (0x5C)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 9_10 – DAC channels 9,10 (0x60)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 11_12 – DAC channels 11,12 (0x64)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 13_14 – DAC channels 13,14 (0x68)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

DAC 15_16 – DAC channels 15,16 (0x6C)

31 – (not used)

[30…28] – (not used – read as 0)

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

15 – (not used)

[14…12] – (not used – read as 0)

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

STATUS 1 – Input Buffer Status (0x70)

31 – (R) – data buffer (channel 1 – 16) ready for input

30 – (R) – data buffer (channel 1 – 16) input paused

29 – (R) – not used (read as '0')

28 – (R) – data buffer (channel 1 – 16) empty

27 – (R) – data buffer (channel 1 – 16) full

[26…16] – (R) – data buffer (channel 1 – 16) word count

(bits 15-0 not implemented)

15 – (R) – data buffer (channel 9 – 16) ready for input

14 – (R) – data buffer (channel 9 – 16) input paused

13 – (R) – not used (read as ‘0’)

12 – (R) – data buffer (channel 9 – 16) empty

11 – (R) – data buffer (channel 9 – 16) full

[10…0] – (R) – data buffer (channel 9 – 16) word count

STATUS 2 – Build Buffer Status (0x74)

[31…29] – not used (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] – not used (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] – not used (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] – not used (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

AUXILIARY 2 – (spare) (0x84)

[31…0] – reserved

AUXILIARY 3 – (spare) (0x88)

[31…0] – reserved

(AUXILIARY 4 – (spare) has been deleted) (7/12/11)

(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).)

RAM Address Register (0x8C) (7/12/11)

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] – not used (read as 0)

[19…0] – RAM address

RAM 1 Data Register (0x90) (7/12/11)

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

RAM 2 Data Register (0x94) (7/12/11)

[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) (7/12/11)

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) (7/12/11)

[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 FPGA 1 (channel 1-16)

Auxiliary Scaler 2 (0xAC)

[31…0] – Total word count FPGA 2 (channel 9-16) **(not implemented)**

Auxiliary Scaler 3 (0xB0)

[31…0] – Event header word count FPGA 1 (channel 1-16)

Auxiliary Scaler 4 (0xB4)

[31…0] – Event header word count FPGA 2 (channel 9-16) **(not implemented)**

Auxiliary Scaler 5 (0xB8)

[31…0] – Event trailer word count FPGA 1 (channel 1-16)

Auxiliary Scaler 6 (0xBC)

[31…0] – Event trailer word count FPGA 2 (channel 9-16)) **(not implemented)**

Module Busy Level (0xC0)

[31] – Force module busy

[30…20] – reserved

[19…0] – Busy level (eight byte words)
(External RAM word count > Busy level → module busy = 1)

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

[31- 0] – Event Header Word

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

[31- 0] – Event Data Word

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

[31- 0] – 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)

[31- 1] – Reserved

RESERVED (2 registers) (0xD8 – 0xDC)

SCALER CONTROL (0xE0)

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)

SPARE (7 registers) (0xE4 – 0xFC)

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.

ADC_PROCESSING_FPGA_ADDRESS_MAP:

Control Bus Memory Map for FADC FPGA

Name **[VME ADDRESS]**	Width (Bits)	Quantity	Access	Primary Address (Secondary Address)	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 (---)	TRIGGER NUMBER BIT 15 to 0
STATUS2 **[0x108]**	16	1	R	0x0002 (---)	Tbd. Read 0
CONFIG 1 **[0x10C]**	16	1	R/W	0x0003 (---)	Bit 0-2 (process mode): 000 → Select option1 001 → Select option2 010 → Select option3 011 → Select option4 111 → Run option1 then option4 for each trigger Bit 3: 1:Run Bit 6-5 : Number of Pulses in Mode 1 and 2 Bit 7: Test Mode (play Back).
CONFIG 2 **[0x110]**			R/W	0x0004 (---)	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
CONFIG 4 **[0x114]**	16	1		0x0005 (---)	7 => rising edge write to AD9230 ADC 6 => 1 write to all ADC. Bits 3..0 are don't care 5 => 0 write to AD9230 1 read from AD9230 3..0 => Select ADC to write to
CONFIG 5 **[0x118]**	16	1		0x0006 (---)	15..8 => Registers inside AD9230 7..0 => Data to write to register.
PTW **[0x11C]**	9	1	R/W	0x0007 (---)	Number of ADC sample to include in trigger window. PTW = Trigger Window (ns) * 250 MHz. **Minimum is 6**. **Always report Even Number. For odd PTW number, discard the last sample reported.**
PL **[0x120]**	11	1		0x0008 (---)	Number of sample back from trigger point. PL = Trigger Window(ns) * 250MHz
NSB **[0x124]**	12	1		0x0009 (---)	Number of sample before trigger point to include in data processing. This include the trigger Point. **Minimum is 2 in all mode.**
NSA **[0x128]**	13	1		0x000A (---)	Number of sample after trigger point to include in data processing. **Minimum is (6 in mode 2)and (3 in mode 0 and 1). Number of**

					sample report is 1 more for odd and 2 more for even NSA number.
TET **[0x12C – 0x148]** (2 channels per word) (see **Note 1** below)	12	16		0x000B - 0x001A (---)	Trigger Energy Threshold.
PTW DAT BUF LAST ADR **[0x14C]**	12	1		0x001B (---)	Last Address of the Secondary Buffer. See calculation below
PTW MAX BUF **[0x150]**	8	1		0x001C (---)	The maximum number of unprocessed PTW blocks that can be stored in Secondary Buffer. See Calculation below.
Test Wave Form **[0x154]**	16	1		0x001D (---)	Write to PPG. Read should immediately follow write.
ADC0 Pedestal Subtract **[0x158]**	16	1	R/W	0x001E	Subtract from ADC0 Count before Summing
ADC1 Pedestal Subtract **[0x15C]**	16	1	R/W	0x001F	Subtract from ADC1 Count before Summing
ADC2 Pedestal Subtract **[0x160]**	16	1	R/W	0x0020	Subtract from ADC2 Count before Summing
ADC3 Pedestal Subtract **[0x164]**	16	1	R/W	0x0021	Subtract from ADC3 Count before Summing
ADC4 Pedestal Subtract **[0x168]**	16	1	R/W	0x0022	Subtract from ADC4 Count before Summing
ADC5 Pedestal Subtract **[0x16C]**	16	1	R/W	0x0023	Subtract from ADC5 Count before Summing
ADC6 Pedestal Subtract **[0x170]**	16	1	R/W	0x0024	Subtract from ADC6 Count before Summing
ADC7 Pedestal Subtract **[0x174]**	16	1	R/W	0x0025	Subtract from ADC7 Count before Summing

ADC8 Pedestal Subtract **[0x178]**	16	1	R/W	0x0026	Subtract from ADC8 Count before Summing
ADC9 Pedestal Subtract **[0x17C]**	16	1	R/W	0x0027	Subtract from ADC9 Count before Summing
ADC10 Pedestal Subtract **[0x180]**	16	1	R/W	0x0028	Subtract from ADC10 Count before Summing
ADC11 Pedestal Subtract **[0x184]**	16	1	R/W	0x0029	Subtract from ADC11 Count before Summing
ADC12 Pedestal Subtract **[0x188]**	16	1	R/W	0x002A	Subtract from ADC12 Count before Summing
ADC13 Pedestal Subtract **[0x18C]**	16	1	R/W	0x002B	Subtract from ADC13 Count before Summing
ADC14 Pedestal Subtract **[0x190]**	16	1	R/W	0x002C	Subtract from ADC14 Count before Summing
ADC15 Pedestal Subtract **[0x194]**	16	1	R/W	0x002D	Subtract from ADC15 Count before Summing

PTW MAX BUF = INT(2016 / (PTW + 8) * 250000000)

Where:

2016 → Number of address of Secondary Buffer

PTW → Trigger Window width in nano-second

PTW DAT BUF LAST ADR = PTW MAX BUF * (PTW + 6)- 1;

Where:

6 → 4 address for Time Stamp and 2 address for Trigger Number

NumberOfBytePerTrigger → PTW * 250 MHz.

NOTE 1: Trigger Energy Threshold (TET)

0x12C – Channel 1 & Channel 2

[31…28] – not used
[27…16] – channel 1 threshold
[15…12] – not used
[27…16] – channel 2 threshold

0x130 – Channel 3 & Channel 4

[31…28] – not used
[27…16] – channel 3 threshold
[15…12] – not used
[27…16] – channel 4 threshold

. . . . . .

0x148 – Channel 15 & Channel 16

[31…28] – not used
[27…16] – channel 15 threshold
[15…12] – not used
[27…16] – channel 16 threshold

Appendix 1 – Register structure for FADC V2 (6/20/12)

```
struct fadc_struct {
                    long version;           /* (0x000 - 0x0FC) - CTRL */
                    long csr;
                    long ctrl1;
                    long ctrl2;
                    long blk_size;          /* 0x010 */
                    long intr;
                    long adr32;
                    long adr_mb;
                    long sec_adr;           /* 0x020 */
                    long delay;
                    long trig_cfg;
                    long reset;
                    long trig_count;        /* 0x030 */
                    long ev_count;
                    long blk_count;
                    long blk_fifo_count;
                    long blk_wrd_count;     /* 0x040 */
                    long trig_live_count;
                    long mem_wrd_count;
                    long flow_status;
                    long dac1_2;            /* 0x050 */
                    long dac3_4;
                    long dac5_6;
                    long dac7_8;
                    long dac9_10;           /* 0x060 */
                    long dac11_12;
                    long dac13_14;
                    long dac15_16;
                    long status1;           /* 0x070 */
                    long status2;
                    long status3;
                    long status_in;
                    long aux1;              /* 0x080 */
                    long aux2;
                    long aux3;
                    long mem_adr;
                    long mem1_data;         /* 0x090 */
                    long mem2_data;
                    long prom_reg1;
                    long prom_reg2;
                    long berr_count;        /* 0x0A0 */
                    long berr_in_count;     /* 0x0A4 */
                    long scaler_aux[6];     /* 0x0A8 - 0x0BC */
                    long busy_level;        /* 0x0C0 */
                    long gen_evt_head;      /* 0x0C4 */
                    long gen_evt_data;
                    long gen_evt_trail;
                    long status_mgt;
                    long control_mgt;
                    long reserved_ctrl[2];
                    long scaler_ctrl;
                    long spare_ctrl[7];     /* 0x0E4 - 0x0FC */
```

```
                                /* ADC processing */
        long adc_status0;       /* (0x100-1FC) */
        long adc_status1;
        long adc_status2;
        long adc_config1;
        long adc_config2;
        long adc_config4;
        long adc_config5;
        long adc_ptw;
        long adc_pl;
        long adc_nsb;
        long adc_nsa;
        long adc_thres[8];
        long adc_ptw_last;
        long adc_ptw_max;
        long adc_test_wave;
        long adc_pedestal[16];  /* 9/1/11 */
        long spare_adc[26];     /* 9/1/11 */

                                /* HITSUM processing */
        long hitsum_status;     /* (0x200 - 0x2FC) */
        long hitsum_cfg;
        long hitsum_width;      /* 2nd addr (0x0 - 0xF) */
        long hitsum_delay;
        long hitsum_trig_width;
        long hitsum_trig_bits;
        long hitsum_window_width;
        long hitsum_overlap_bits;
        long hitsum_pattern;    /* 2nd addr(0x0 - 0x10000) */
        long hitsum_fifo;
        long hitsum_sum_thres;
        long spare_hitsum[53];

        long scaler[16];     /* (0x300 - 0x3FC) - scalers */
        long time_count;
        long spare_scaler[47];

        long testbit;        /* (0x400 - 0x4FC) - test1 */
        long count_250;
        long count_sync;
        long count_trig1;
        long count_trig2;
        long spare_test1[59];
};
```