#

**Physics Division -- *Fast* Electronics Group**

**Description and Instructions**

**For Compton Polarimeter Firmware Version 0x3900**

**Mar 2021**

**Hai Dong**

# 1.0 Introduction

 This document describes the firmware of the EFADC250\_DAQ. It has 7 unique functions as follows:

1. Read Out Processing
2. Host Interface
3. Play Back
4. IC Configuration
5. Miscellaneous

The IC Configuration and Host Interface functions are running at 125 MHz to support the 1GBits Ethernet.

The remaining functions are running nominally at 250MHz clock rate and all the time references listed in this document are Number of Samples \* 4 nS.

 **Read Out Processing**

# 2.0 Read Out Processing

**Figure 1: Signal Processing**

BCM

PMT

Pockets Cell

+h

-h

+v

TETPMT

TETBCM

1uS

12.8 uS

-v

Helicity

T-Settle

**Triggering Scheme:**

1. When NSAT BCM sample are greater than TETBCM, a trigger pulse (TrigBcm) is generated.
2. When NSAT PMT sample are greater than TETPMT, a trigger pulse (TrigPmt) is generated.
3. TrigSel selects either TrigBcm or TrigPmt to start data processing.
4. MODE selects either Sample or Semi-Int data processing

**Data Processing:**

**In Sample Mode**

* 1. Write out S1 samples of BCM signal starting at NSB sample before TriggerSample.
	2. Write out S2 samples of PMT signal starting at NSB sample before TriggerSample.
	3. Write out NSB sample before TriggerSample and every 10th samples up to S3 samples of Pockels Cell signal.
	4. Write out NSB sample before TriggerSample and every 10th samples up to S4 samples of Helicity signal.
	5. Write out NSB sample before TriggerSample and every 10th samples up to S5 samples of T-Settle signal.

**Semi-Int Mode**

* Add S1 12-bits samples of BCM signal starting at NSB sample before TriggerSample.
* Add S2 12-bits samples of PMT signal starting at NSB sample before TriggerSample.
* Add NSB sample before TriggerSample and every 10th samples up to S3 12-bits samples of Pockels Cell signal.
* Add NSB sample beforeTriggerSample and every 10th samples up to S4 12-bits samples of Helicity signal.
* Add NSB sample before TriggerSample and every 10th samples up to S5 12-bits samples of T-Settle signal.
* Write out all five 21-bits sums.
* Overflow bit of the ADC sample does not include in the sums. If any sample is overflow (1 1111 1111 1111) or underflow (1 0000 0000 0000), the corresponded flag bit will be set.

NSAT => Number of Sample above Threshold (user program 1,2,3,4)

TriggerSample is the sample that causes trigger. NSB is number of sample before TriggerSample. For example if NSB = 0, it is the TriggerSample. If NSB = 1, it is the sample before TriggerSample.

**Register Specification:**

MODE : 1 bit (0= select Semi-Int; 1=select Sample Mode)

TETBCM : 12 bits (1- 4095)

TETPMT : 12 bits (1- 4095)

NSAT : 2 bits (0=1 NSAT, 1=2 NSAT, 2= 3 NSAT, 3=4 NSAT)

TrigSel : 1 bit (0= select TrigBcm; 1= select TrigPmt)

S1 : 9 bits (2-510) must be even

S2 : 9 bits (2-510) must be even

S3 : 6 bits (2-52) must be even

S4 : 6 bits (2-52) must be even

S5 : 6 bits (2-52) must be even

PreScale : 8 bits.

NSBBCM : 5 bits

NSBPMT : 5 bits

NSBPocketCell : 5 bits

NSBHelicity : 5 bits

NSBTSettle : 5 bits

**User is required to follow the requirement in setting the run parameters. For example S1, S2, S3, S4 and S5 must be even number.**

**STOP Collection before changing Mode.**

**Failure to do so might cause incorrect result.**

A counter (Trigger Number) will increment for each trigger. When the connection to the host is back up and no new data can be sent, triggers will not be processed (no packet will be created for these triggers) but the counter (Trigger Number) still increment. The trigger number is part of every data packet sent to host. The user can use this feature to verify that no trigger is missed.

PreScale feature allows the user to reduce data rate. The parameter specifies the number of trigger NOT to be processed. For example if PreScale is set to 10, only every 10th triggers (10th, 20th, 30th …) will be processed. Triggers (1,2,3,4,5,6,7,8,9, , 11, 12,13,14,15,16,17,18,19, , …) are not going to be processed. Effectively the data rate is reduced by 10.

ADC Input Channel Mapping:

 ADC0 🡪 BCM

 ADC1 🡪 PMT

 ADC2 🡪 Pocket Cell

 ADC3 🡪 Helicity

 ADC4 🡪 T Settle

 **Host Interface**

# 3.0 Host Interface

The firmware uses UDP Ethernet protocol for communication with the host computer for register configuration as shown in Appendix 1 Table 2. UDP Ethernet protocol transactions consists of data packets. Each UDP packet starts with 0x5A 0x5A following by Opcode as shown in Table1 and Table 2 and finally data for Opcode that consists of data. The data is for setting the registers (shown in Appendix 3), the LCD display, Play Back, and Ethernet Parameters. Appendix 1: “UDP Data Packet from Host to EFADC” and Appendix 2: “UDP Data Packet from EFADC to Host” specify the format and show an example for each Opcode. For every UDP Packet from the host there will be an “acknowledge good” or an “acknowledge bad” response from EFADC. It is advised that the user read this acknowledgement immediately sending command. If a command is requesting data such as “read register”, the EFADC will also send back the registers content. The packet is being clocked out at 8 nS per byte.

The firmware uses TCP Ethernet protocol to send processed data to host computer. TCP packet size is 1470 bytes and the time out is 200 uS. The number of bytes for each trigger depends on the mode. Since TCP protocol will send the packet when the number of byte is equaled to 1470 bytes or at time out, data of two or more triggers might be sent in same TCP packet. A packet contains less than 1460 bytes will be sent if a time out of 200 uS occurs. The user parse the packet using packet format shown in Appendix 4: TCPIP Data Format from EFADC. The TCPIP data packet for each trigger from EFADC depends on the mode of operation. The packet is being clocked out at 8 nS per byte.

**In Sample Mode**

* Header Word (one 32 bits)
* Trigger Time (two 32 bits)
* ADC0 (BCM) samples (S1 / 2 32 bits)
* ADC1 (PMT) samples (S2/ 2 32 bits)
* ADC2 (Pocket Cell) samples ((S3 / 2) + 1 32 bits)
* ADC3 (Helicity) samples ((S3 / 2) + 1 32 bits)
* ADC4 (TSettle) samples ((S3 / 2) + 1 32 bits)
* Event Trailer (one 32 bits)

**Semi-Int Mode**

* Header Word (one 32 bits)
* Trigger Time (two 32 bits)
* ADC0 (BCM) Sum (one 32 bits)
* ADC1 (PMT) Sum (one 32 bits)
* ADC2 (Pocket Cell) Sum (one 32 bits)
* ADC3 (Helicity) Sum (one 32 bits)
* ADC4 (TSettle) Sum (one 32 bits)
* Event Trailer (one 32 bits)

 **Play Back**

# 4.0 Play Back

User defines pulses maybe injected into the processing pipeline using a playback feature. Play Back stores 32, 13-bit ADC values in RAM and cycles through 32 ADC values on the rising edge of Trigger signal. There are 16 Play Back, one per ADC Channel. All 512 ADC values are written into memory using set playback command. Set Play Back Data in Appendix 1 shows an example of UDP packet to store 512 ADC values into RAM. When **bit 7 of Config 1** (test mode) is set, Play Back outputs (instead of ADC IC outputs) are applied to all ADC processing functions. The data from Play Back is shown below (See Set Play Back Data in Appendix 1 for the relation between byte location in the packet and ADC Chan Play Back Word # . Trigger rate is .5 second.

Play Back Word #

ADC Chan

C

0

C

2

3

3

1

30

31

32

1

1

C

2

3

3

1

30

31

32

1

2

C

2

3

3

1

30

31

32

1

3

C

2

3

3

1

30

31

32

1

4

C

2

3

3

1

30

31

32

1

Trigger

C

 **IC Configuration**

# 5.0 IC Configuration

1. **ADC IC AD9230**

The ADC AD9230 ICs are needed to be configured after power up.

* 1. To configure all ADC ICs at one time
		1. Poll bit 15 of Status 0 for a one. This indicates firmware is ready to accept command.
		2. Write 0 to bit 7 of Config 4. Rising edge firmware sends data to AD9230
		3. Select register of AD9230 to write to by writing to bits 15-8 of Config 5. Write data to be written to register of AD9230 by writing bits 7-0 of Config 5.
		4. Set bits 7,6 and reset Bit 5 of Config 4. Bit 6 tells firmware to write to AD230. Bit 5 tells firmware to write to all AD9230
	2. For Example to configure all ADC to convert negative going signal:
		1. Configure AD9230 delay clock
			1. Poll bit 15 of Status 0 for a one.
			2. Reset bit 7 of Config 4
			3. 0x17 to bits 15-8 of Config 5 to select AD9230 ADC\_CLK\_OUT\_DELAY\_REG
			4. 0x9E to bits 7-0 of Config 5. Data to write to ADC\_CLK\_OUT\_DELAY\_REG 0x9E. Delay clock b
			5. Set bit 7 and 6, reset bit 5 of Config 4. This tells firmware to write to AD9230
			6. Poll bit 15 of Status 0 for a one
			7. Reset bit 7 of Config 4
			8. 0xFF to bit 15-8 of Config 5 to select ADC\_MASTER\_TO\_SLAVE\_REG
			9. 0x01 to bit 7-0 of Config 5. Data to write to ADC\_MASTER\_TO\_SLAVE\_REG. Tell AD9230 to execute delay clock setting.
			10. Set bit 7 and 6, reset bit 5 of Config 4
		2. Configure AD9230 to run in CML mode
			1. Poll bit 15 of Status 0 for a one.
			2. Reset bit 7 of Config 4
			3. 0x0F to bits 15-8 of Config 5 to select AD9230 ADC\_AIN\_CONFIG\_REG
			4. 0x02 to bits 7-0 of Config 5. Data to write to ADC\_AIN\_CONFIG\_REG. Run in CML mode
			5. Set bit 7 and 6, reset bit 5 of Config 4. This tells firmware to write to AD9230
			6. Poll bit 15 of Status 0 for a one
			7. Reset bit 7 of Config 4
			8. 0xFF to bit 15-8 of Config 5 to select ADC\_MASTER\_TO\_SLAVE\_REG
			9. 0x01 to bit 7-0 of Config 5. Data to write to ADC\_MASTER\_TO\_SLAVE\_REG. Tell AD9230 to execute delay clock setting.
			10. Set bit 7 and 6, reset bit 5 of Config 4
		3. Tell AD9230 to turn off test mode
			1. Poll bit 15 of Status 0 for a one.
			2. Reset bit 7 of Config 4
			3. 0x0D to bits 15-8 of Config 5 to select AD9230 ADC\_TEST\_REG
			4. 0x00 to bits 7-0 of Config 5. Data to write to ADC\_TEST\_REG. Turn off test mode
			5. Set bit 7 and 6, reset bit 5 of Config 4. This tells firmware to write to AD9230
			6. Poll bit 15 of Status 0 for a one
			7. Reset bit 7 of Config 4
			8. 0xFF to bit 15-8 of Config 5 to select ADC\_MASTER\_TO\_SLAVE\_REG
			9. 0x01 to bit 7-0 of Config 5. Data to write to ADC\_MASTER\_TO\_SLAVE\_REG. Tell AD9230 to execute delay clock setting.
			10. Set bits 7 and 6, reset bit 5 of Config 4
1. **BIAS DAC AD5516ABC-1**

This function allows configuring the 16 Bias DAC on the EFADC-16 board. The bias DAC provide the pedestal (base line) values. Pedestal is the ADC sample values when there is no input signal. To configure the Bias DAC:

* 1. Read Bit 14 of Status 2 for a one to indicate this function is ready
	2. Write Config. 5
		1. Bits 15..12 🡪 Select which Bias DAC to write to.
		2. Bits 11.0 🡪 Value to be written to Bias DAC.
	3. Write 1 to bit 15 of Config 1
	4. Write 0 to bit 15 of Config 1
1. **LCD**

This function allows user to write to “NHD-C0216CZ\_FSW\_FBW-3V3” LCD. This LCD is 2 lines of 16 characters for each line. Set LCD Data 1St Line and Set LCD Data 2nd Line in Appendix 1 show the UDP data packet to write to LCD

 **Miscellaneous**

# 6.0 Miscellaneous

1. **FPGA Die Temperature**

The temperature of the FPGA die can be read at register STATUS3 (Die Temp). The Celsius temperature is calculated as follow:

DieTemp\_C = ((float)(STATUS3) \* 503.975/1024) - 273.15;

1. **Firmware Version Number**

Firmware version number can be read at register STATUS1 bits 14 to 0. Bits 15-8 is 0x38 and bits 7-0 indicates the code revision

1. **Board Serial Number**

Board serial number can be read at register STATUS2 bits 8 to 0.

1. **Time Stamp**

 Time Stamp keep track of time while Collecting Data (after receiving Collect On command). It freezes when not collecting data (after receiving Collect Off command). It is reset at power up or when bit 6 of Config 1 is set.

1. **Trigger Number**

 Trigger number is the number of trigger that has processed. If triggers come faster than processing time, these triggers are not processed and won’t be counted. This condition can happen in Verifying and Streaming modes when the time to send NumSampToWrOut samples is faster than the Ethernet link or the host can receive. The firmware checks that there is sufficient buffer before accepting another trigger.

 **APPENDIX 1**

 **UDP Data Packet from Host to EFADC**

# 7.0 Appendix 1 UDP Data Packet from Host to EFADC

**The format for sending data to the EFADC250 is as follow:**

1. 5A
2. 5A
3. Op-Opcode as shown in Table 1
4. Data

**The format for “acknowledge good” response from EFADC250 is as follow:**

1. 5A
2. 5A
3. 00
4. 03
5. FA

**The format for “acknowledge bad” response from EFADC250 is as follow:**

1. 5A
2. 5A
3. 00
4. 03
5. FE

Table 1: OpCode from Host

|  |  |  |
| --- | --- | --- |
| OP-CODE | Function | Type Of Data |
| 01 | Set Registers, LCD, Play Back | 0x0000, the bytes followed are register data (see register files chart). 0x0002, the bytes followed are to be displayed on LCD (must be 64 bytes).0x0003, the bytes followed are to be stored in Play Back Memory |
| 02 | ActivateCommand | 00: Collect Off. Collect is OFF after power up.01 : Collect Data On02: Reserved03: Read Back Registers (only one time).04: Read Play Back Memory05: Play Back all 512 samples once. Set bit 7 of Config1 to be ADC samples.**06**: Write new IPv4\_ADDR, SUBNET\_MASK, GATEWAY\_IP, MAC\_ADDR to config ROM**07**: Read new IPv4\_ADDR, SUBNET\_MASK, GATEWAY\_IP, MAC\_ADDR to config ROM |

 **Examples of data from Host to EFADC-15:**

Turn Collect On (See Appendix A for Registers’ Definition)

5A --- header

5A

02 -- Opcode to turn Collect on, Regs Read back request, Reset ADC

01 -- 01 indicated turn Collect on.

Processed Data from ADC is sent to PC until Turn Collect off command is received.

Turn Collect Off (See Appendix A for Registers’ Definition)

5A --- header

5A

02 -- Opcode to turn Collect on, Regs Read back request, Reset ADC

00 -- 00 indicated turn Collect off.

Request Register and Status Read Back (See Appendix A for Registers’ Definition)

5A --- header

5A

02 -- Opcode to turn Collect on, Regs Read back request, Reset ADC

03 -- 03 indicated request all Registers to be sent back

Register and Status are sent to PC once. After PC sent this command, it should wait for registers and status data to arrive before sending another command. See example of register and status below.

Set Registers (See Appendix A for Registers’ Definition)

5A --- header

5A

01 -- Opcode to set Register, PLayBack, LCD

00 -- 0000 indicates data is for registers.

00

00 -- Config 1 Hi Byte

01 -- Config 1 Lo Byte

00 -- Config 2 Hi Byte

02 -- Config 2 Lo Byte

00 -- Config 3 Hi Byte

03 -- Config 3 Lo Byte

00 -- Config 4 Hi Byte

04 -- Config 4 Lo Byte

00 -- Config 5 Hi Byte

05 -- Config 5 Lo Byte

00 -- Config 6 Hi Byte

06 -- Config 6 Lo Byte

00 -- Config 7 Hi Byte

07 -- Config 7 Lo Byte

00 -- Config 8 Hi Byte

08 -- Config 8 Lo Byte

00 -- Config 9 Hi Byte

09 -- Config 9 Lo Byte

00 -- Config 10 Hi Byte

0A -- Config 10 Lo Byte

00 -- Config 11 Hi Byte

0A -- Config 11 Lo Byte

00 -- Config 12 Hi Byte

0B -- Config 12 Lo Byte

Set LCD Data 1St Line

5A --- header

5A

01 -- Opcode to set Register, PLayBack, LCD

00 -- 0002 indicated data is for LCD data

02

02 -- LCD CMD

01 -- Return Home, 1st CHAR

03 -- LCD Char

\*\* -- LCD Char 1 ASCII

03 -- LCD Char

\*\* -- LCD Char 2 ASCII

03 -- LCD Char

\*\* -- LCD Char 3 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 4 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 5 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 6 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 7 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 8 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 9 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 10 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 11 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 12 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 13 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 14 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 15 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 16 Lo Byte

Set LCD Data 2nd Line

5A --- header

5A

01 -- Opcode to set Register, PLayBack, LCD

00 -- 0002 indicated data is for LCD data

02

02 -- LCD CMD

01 -- Go to 2nd line 1st CHAR

03 -- LCD Char

\*\* -- LCD Char 1 ASCII

03 -- LCD Char

\*\* -- LCD Char 2 ASCII

03 -- LCD Char

\*\* -- LCD Char 3 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 4 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 5 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 6 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 7 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 8 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 9 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 10 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 11 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 12 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 13 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 14 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 15 Lo Byte

03 -- LCD Char

\*\* -- LCD Char 16 Lo Byte

Set Play Back Data

5A --- header

5A

01 -- Opcode to set Register, PLayBack, LCD

00 -- 0001 indicates data is for Play Back Data.

03 --

01 -- bit 12-8 of ADC 1 Play Back Wd 1

02 -- bit 7-0 of ADC 1 Play Back Wd 1

01 -- bit 12-8 of ADC 1 Play Back Wd 2

02 -- bit 7-0 of ADC 1 Play Back Wd 2

 :

 :

01 -- bit 12-8 of ADC 1 Play Back Wd 32

02 -- bit 7-0 of ADC 1 Play Back Wd 32

01 -- bit 12-8 of ADC 2 Play Back Wd 1

02 -- bit 7-0 of ADC 2 Play Back Wd 1

01 -- bit 12-8 of ADC 2 Play Back Wd 2

02 -- bit 7-0 of ADC 2 Play Back Wd 2

 :

 :

01 -- bit 12-8 of ADC 16 Play Back Wd 31

02 -- bit 7-0 of ADC 16 Play Back Wd 31

01 -- bit 12-8 of ADC 16 Play Back Wd 32

02 -- bit 7-0 of ADC 16 Play Back Wd 32

 **APPENDIX 2**

 **UDP Data Packet from EFADC to Host**

# 8.0 Appendix 2 UDP Data Packet from EFAD to Host

**The format of data from EFADC250 is as follows:**

1. 5A5A
2. Op-Opcode as shown in Table 2
3. Data

Table 2: OpCode from EFADC250

|  |  |  |
| --- | --- | --- |
| OP-CODE | Function | Type Of Data |
| 03 | Data to PC | 01: Reserved03: Register and Normal (Running) Status values.04: Play Back Memory data05: Reserved07: Register and (Ethernet Parameters) Status valuesFA: CMD Received is goodFE: CMD Received is bad  |
|  |  |  |

**Examples of data from EFADC-15 to Host:**

Registers and Status (See Appendix A for Registers’ and Status Definition) when COnfig 1 Bit 11 is zero

5A --- header

5A

03 -- Opcode to read Register, PLayBack, LCD, etc

03 -- 03 indicates data is for registers and Status is running (normal) condition

00 -- Config 1 Hi Byte

01 -- Config 1 Lo Byte

00 -- Config 2 Hi Byte

02 -- Config 2 Lo Byte

00 -- Config 3 Hi Byte

03 -- Config 3 Lo Byte

:

:

00 -- Config 10 Hi Byte

0A -- Config 10 Lo Byte

00 -- Config 11 Hi Byte

0A -- Config 11 Lo Byte

00 -- Config 12 Hi Byte

0A -- Config 12 Lo Byte

 :

 :

00 -- Status 0 Hi Byte

00 -- Status 0 Lo Byte

01 -- Status 1 Hi Byte

01 -- Status 1 Lo Byte

02 -- Status 2 Hi Byte

02 -- Status 2 Lo Byte

03 -- Status 3 Hi Byte

03 -- Status 3 Lo Byte

04 -- Status 4 Hi Byte

04 -- Status 4 Lo Byte

 :

 :

09 -- Status 9 Hi Byte

09 -- Status 9 Lo Byte

10 -- Status 10 Hi Byte

10 -- Status 10 Lo Byte

Read Play Back Data

5A --- header

5A

03 -- Opcode to read Register, PLayBack, LCD

04 -- 04 indicates data is for Pedestal Sums

01 -- bit 12-8 of ADC ch 0 Play Back Wd 0

02 -- bit 7-0 of ADC ch 0 Play Back Wd 0

01 -- bit 12-8 of ADC ch 0 Play Back Wd 1

02 -- bit 7-0 of ADC ch 0 Play Back Wd 1

 :

 :

01 -- bit 12-8 of ADC ch 0 Play Back Wd 31

02 -- bit 7-0 of ADC ch 0 Play Back Wd 31

01 -- bit 12-8 of ADC ch 1 Play Back Wd 0

02 -- bit 7-0 of ADC ch 1 Play Back Wd 0

01 -- bit 12-8 of ADC ch 1 Play Back Wd 1

02 -- bit 7-0 of ADC ch 1 Play Back Wd 1

 :

 :

01 -- bit 12-8 of ADC ch 15 Play Back Wd 30

02 -- bit 7-0 of ADC ch 15 Play Back Wd 30

01 -- bit 12-8 of ADC ch 15 Play Back Wd 31

02 -- bit 7-0 of ADC ch 15 Play Back Wd 31

 **APPENDIX 3**

 **Register Definition**

# 9.0 Registers Definition

**Register Definitions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Access** | **Runs**  | **Set Ethernet** |
| STATUS 0 | R | 15..0 🡪 FirmwareVersion (0x38\_ \_) | IPv4\_addr (31..16) |
| STATUS 1 | R | 15..0 🡪BdSerialNum (0x0001) | IPv4 addr (15..0) |
| STATUS 2 | R | 15 🡪 AD9230\_READY 14 🡪 AD5516 READY 13..0 🡪 zeroes | Subnet mask (31..16)  |
| STATUS 3 | R | 9..0 🡪 DieTemp | Subnet mask (15..0) |
| STATUS 4 | R | 15..0 🡪PortNumber | MAC addr (47..32) |
| STATUS 5 | R | 15..0 🡪MAC addr (31..16) | MAC addr (15..0) |
| STATUS 6 | R | 12. 0 🡪ADC 0 Pedestal |  |
| STATUS 7 | R | 12. 0 🡪ADC 1 Pedestal |  |
| STATUS 8 | R | 12. 0 🡪ADC 2 Pedestal |  |
| STATUS 9 | R | 12. 0 🡪ADC 3 Pedestal |  |
| STATUS 10 | R | 12. 0 🡪ADC 4 Pedestal |  |
|  |  |  |  |
| CONFIG 1 | R/W | 15 🡪 Rising edge write AD5516 reg to DAC14..13 🡪 NSAT  0=1 NSAT; 1=2 NSAT; 2=3 NSAT; 3=4 NSAT,12..10 🡪 9,8 🡪 Mode 00: **Semi-Int** mode  01: **Verifying** mode  10; 11 : **Reserve**7 🡪 Test Mode 1 = Sample from Play Back6 🡪 1 Reset Trigger Number and Time Stamp 5-2🡪1 -> Trigger Select (0 BCM, 1 PMT) 0 🡪 Sel Ethernet Para for Status |  |
| CONFIG 2 | R/W | 11..0 🡪 TET BCM  | IPv4\_ADDR\_to\_ROM(31.. 16) |
| CONFIG 3 | R/W | 11..0 🡪 TET PMT | IPv4\_ADDR\_to\_ROM(15.. 0) |
| CONFIG 4IDLAY Set | R/W | 15..12 🡪 Select which ADC receive IDELAY control bits and read back IDELAY comparator error11 🡪 Idelay comparator reset10 🡪 Increment IDELAY N delay value9 🡪 Decrement IDELAY P delay value8 🡪 Reset IDELAY7 🡪 rising edge write to AD9230 ADC6 🡪 1 write to all ADC5 🡪 0 write to AD9230 1 read from AD9230 . Data is at Stat4 🡪 1 Reset ADC3..0 🡪 Select ADC to write to | SUBNET\_MASK\_to\_ROM(31.. 16) |
| CONFIG 5 | R/W | 15..8 🡪 Registers inside AD92307..0 🡪Data to write to register. | SUBNET\_MASK\_to\_ROM(15..0) |
| CONFIG 6 | R/W | 15..12 🡪 select DAC to write DAC value bits \11..0 🡪 DAC value | BdSerialNum to be save to ROM(15..0) |
| CONFIG 7 |  | 13-9 🡪 NSB BCM8-0 🡪 S1 BCM |  |
| CONFIG 8 | R/W | 13-9 🡪 NSB PMT8-0 🡪 S1 PMT | PortNumber\_to\_ROM(15..0) |
| CONFIG 9 | R/W | 13-9 🡪 NSB Pockets Cell5-0 🡪 S1 Pockets Cell  | MAC\_ADDR\_to\_ROM(47..32) |
| CONFIG 10 | R/W | 13-9 🡪 Helcity5-0 🡪 S1 Helicity  | MAC\_ADDR\_to\_ROM(31..16) |
| CONFIG 11 | R/W | 13-9 🡪 NSB T Settle5-0 🡪 S1 T Settle | MAC\_ADDR\_to\_ROM(15..0) |
| CONFIG 12 | R/W | 7-0 🡪 PreScale | When = x"ABCD" save BdSerialNum to ROM  |

 **APPENDIX 4**

 **TCPIP Data format from EFADC**

# 10.0 TCPIP Data format from EFADC

**Appendix A: Data Format of FADC Processing**

**Event Header** (2) – indicates the start an event.

 (31) = 1

 (30 – 27) = 2

 (26 – 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

 Time = TA TB TC TD TE TF

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

Word 1:

 (31) = 1

 (30 – 27) = 3

 (26 – 24) = “000”

 (23 – 16) = TD

 (15 – 8) = TE

 (7 – 0) = TF

Word 2:

 (31) = 0

 (30 – 24) = reserved (read as 0)

 (23 – 16) = TA

 (15 – 8) = TB

 (7 – 0) = TC

**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 bit 13 will be set when PTW+1 is odd.

Word 1:

 (31) = 1

 (30 – 27) = 4

 (26 – 23) = channel number (0 – 15)

 (22 – 12) = reserved (read as 0)

 (8 – 0) = number of samples

Words 2 - N:

 (31) = 0

 (30) = reserved (read as 0)

 (29) = sample x not valid

 (28 – 16) = ADC sample x (includes overflow bit)

 (15 – 14) = reserved (read as 0)

 (13) = sample x + 1 not valid

 (12 – 0) = ADC sample x + 1 (includes overflow bit)

**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 Integral of pulse

 (31) = 1

 (30 – 27) = 9

 (26 – 23) = channel number (0 – 15)

 (22) = 0

 (21) = One or more samples is overflow = 0x1FFF

 (20) = One or more sample is underflow = 0x1000

 (19 – 0) = 20-bit sum of raw samples that constitute the pulse data set

**Event Trailer:** Indicate the end of an event.

 EVENT\_TRAILER = X"E8000000";

 **APPENDIX 5**

 **Read Out Data Format**

# 11.0 Read out Data Format

Read Out Data Format:

Sample mode

* Header Word (one 32 bits)
* Trigger Time (two 32 bits)
* BCM samples (S1/2 32 bits)
* PMT samples (S2/2 32 bits)
* Pockels Cell samples (S3/2 + 1 32 bits)
* Helicity samples (S4/2 + 1 32 bits)
* T-Settle samples (S5/2 + 1 32 bits)
* Event Trailer (one 32 bits)
* ------------------------------------------------------------

Total 32-bits Words = [(S1+S2+S3+S4+S5)/2] + 4 + 3 32-bits words every 12.8 uS

Total Bytes = Total 32-bits Words \* 4

Semi-Int mode

* Header Word (one 32 bits)
* Trigger Time (two 32 bits)
* BCM sum (1 32 bits)
* PMT sum (1 32 bits)
* Pockels Cell sum (1 32 bits)
* Helicity sum (1 32 bits)
* T-Settle sum (1 32 bits)
* Event Trailer (one 32 bits)
* ------------------------------------------------------------

Total 32-bits Words = 9 32-bits words every 12.8 uS

Total Bytes = 36

 **APPENDIX 6**

 **DAC to ADC Counts with No Inputs**

# 12.0 DAC Setting to ADC Count

**ADC COUNT in Dec (Bit 13 is Overflow) with No Inputs**

Board Serial Number 0001

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DAC Count | ADC 0 | ADC 1 | ADC 2 | ADC 3 | ADC 4 |
| 4095 | 4096 | 4096 | 4096 | 4096 | 4096 |
| 4000 | 4096 | 4096 | 4096 | 4096 | 4096 |
| 3500 | 4096 | 4096 | 4096 | 4096 | 4096 |
| 3100 | 353 | 380 | 391 | 412 | 374 |
| 3000 | 489 | 532 | 526 | 553 | 509 |
| 2900 | 626 | 669 | 659 | 694 | 643 |
| 2800 | 766 | 802 | 791 | 820 | 776 |
| 2700 | 899 | 940 | 927 | 952 | 910 |
| 2600 | 1036 | 1074 | 1059 | 1084 | 1045 |
| 2500 | 1175 | 1207 | 1194 | 1217 | 1179 |
| 2400 | 1305 | 1344 | 1326 | 1352 | 1314 |
| 2300 | 1445 | 1481 | 1462 | 1484 | 1448 |
| 2200 | 1583 | 1614 | 1596 | 1619 | 1589 |
| 2100 | 1720 | 1751 | 1727 | 1751 | 1720 |
| 2000 | 1853 | 1889 | 1863 | 1884 | 1852 |
| 1900 | 1990 | 2023 | 1998 | 2017 | 1986 |
| 1800 | 2124 | 2159 | 2132 | 2151 | 2120 |
| 1700 | 2265 | 2293 | 2265 | 2278 | 2253 |
| 1600 | 2396 | 2430 | 2400 | 2415 | 2390 |
| 1500 | 2536 | 2567 | 2532 | 2548 | 2526 |
| 1400 | 2672 | 2565 | 2532 | 2552 | 2524 |
| 1300 | 2810 | 2839 | 2797 | 2816 | 2792 |
| 1200 | 2947 | 2972 | 2933 | 2948 | 2928 |
| 1100 | 3083 | 3108 | 3068 | 3080 | 3061 |
| 1000 | 3216 | 3242 | 3202 | 3214 | 3200 |
| 900 | 3365 | 3369 | 3335 | 3352 | 3332 |
| 800 | 3489 | 3512 | 3471 | 3480 | 3466 |
| 700 | 3629 | 3651 | 3601 | 3615 | 3600 |
| 600 | 3770 | 3783 | 3735 | 3748 | 3733 |
| 500 | 3902 | 3915 | 3867 | 3882 | 3868 |
| 400 | 4033 | 4054 | 4005 | 4010 | 4003 |
| 300 | 8191 | 8191 | 8191 | 8191 | 8191 |
| 200 | 8191 | 8191 | 8191 | 8191 | 8191 |
| 100 | 8191 | 8191 | 8191 | 8191 | 8191 |