

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

**Description and Instructions**

**For the**

**FADC Pulse Compression**

**May 2018**

**Hai Dong**

# Introduction

 This document specifies the pulse compression for FLASH250 firmware. In addition to implement the pulse compression algorithm as described in FADC Compression Algorithm document by Gaggik Gavalianit also manages other functions such as either passing Event Header Data or suppressing Pulses Parameter Data. The data output format will be slightly different than the document description to accommodate the header and ADC channel information to.

1. **Pulse Compressing Algorithm**
	1. Description

The algorithm takes in N (in multiple of 16) 12-bits samples of ADC, compresses in group of 16 samples, and write out results in 32 bits format to be transfer to the VME bus. The steps are as follow:

* + 1. Find the minimum values of the pulses (Raw Data Word 2-N) Only bits 11-0 are used 🡪 Min Value
		2. Subtract the minimum value of the pulse from the sample 🡪 Sub Values
		3. Isolate lower 4 bits and upper 8 bits of Sub Values into two arrays.
		4. Create an array of 32 bits words contain lower 4 bits 🡪 Low Array
		5. Create an array of 32-bits words contain non-zero upper 8 bits of Diff Value 🡪 Up array
		6. Write Min Value, Low Array, Up Array, the number of zero before 1st non-zero upper 8 bits, and the number of non-zero upper 8-bits in the format shown below
	1. Compressing Algorithm Output for FLASH250

Table 1 shows the data output of the compressing algorithm for FLASH250

Table 1: Data Output Format 36-bits **(BITS 35-32 ARE EITHER PASSED THROUGH or Zero)**

|  |  |
| --- | --- |
| Word Number | Function |
| 0 | Event Header  |
| 1 | Trigger Time Word 1 |
| 2 | Trigger Time Word 2 |
| 3 | ADC Channel A Header (of 1st pulse) |
| 4 | ADC Channel A Low Array #7-0. Low nibble is in bit 3-0. |
| 5 | ADC Channel A Low Array #15-8. |
| 6 | ADC Channel A Up Array #x-0. Low byte is in bit 7-0. If number of byte < 4, unused bytes are zero. x is the last byte |
| **:** |  |
| **:** |  |
| N/4 + 7 (N=number of non-zero 8-bits)  | ADC Channel A Up array of last 4 byte (or byte). Unused bytes are zeroes |
| N/4 + 8 | ADC Channel A Header (of 2nd pulse if existed) |
| N/4 + 9 | ADC Channel A Low Array #7-0. Low nibble is in bit 3-0. |
| **:** |  |
| **:** |  |
| K = Depend on Number of Pulses in Channel A | ADC Channel B Header (of 1st pulse) |
| K +1 | ADC Channel B Low Array #7-0. Low nibble is in bit 3-0. |
| K + 2 | ADC Channel B Low Array #15-8. |
| K + 3 | ADC Channel B Up Array #x-0. Low byte is in bit 7-0. If number of byte < 4, unused bytes are zero. x is the last byte |
| **:** |  |
| **:** |  |
| N/4 + 6 + K | ADC Channel B Header (of 2nd pulse if existed) |
| **:** |  |
| **:** |  |
| Depend on number of channel having pulse and number of pulses in each channel | Last ADC Channel Up Array of last 4 bytes |
| Last Word | Event Trailer |
|  |  |

* 1. ADC Channel X Header

 Bits 35-32 => 0000

 Bits 31 => 1 (beginning of compress data of group of 16 samples)

 Bits 30-27 => 1000 (Compress Data)

 Bits 26-23 => ADC Channel Number

 Bit22 => ‘0’

 Bit21 => => Number of leading zero bytes in Up Array(bit 4)

 Bit 20 => Minimum Value bit 12 (overflow)

 Bits 19-8 => Minimum Value bits 11-0

 Bits 7-4 => Number of leading zero bytes in Up Array(bits 3-0)

 Bits 3-0 => Number of non-zero bytes in Up Array

1. **Example Showing Data Output of each stage of compression**

The following example shows raw data from ADC channel 1 and ADC channel 16 and the outputs from each compression step

* 1. Chan one 32 Samples Inputs in Decimal (1 short and 1 long pulses)
		+ 1. Sample Number 4 3 2 1 0
		1. 100 101 104 99 900 1000 2000 102 100 98 100 100 100 100 100 100
		2. 99 106 88 500 850 900 1000 900 800 700 600 500 450 400 350 300
	2. Chan 1 Minimum Value =
		1. **98** for 1st 16 samples
		2. **88** for 2nd 16 samples
	3. Chan 1 Subtract Min Value from sample in decimal
		1. For 1st 16 samples

2 3 6 1 802 902 1902 4 2 0 2 2 2 2 2 2 in decimal

 2 3 6 1 322 386 76E 4 2 0 2 2 2 2 2 2 in hex

* + 1. For 2nd 16 samples

11 18 0 412 762 812 912 812 712 612 512 412 362 312 262 212 in decimal

B 12 0 19C 2FA 32C 390 32C 2C8 264 200 19C 16A 138 106 D4

* 1. Chan one Low (4 bits) Array
		1. For 1st 16 samples

**2 0 2 2 2 2 2 2**

**2 3 6 1 2 6 E 4**

* + 1. For 2nd 16 samples

**8 4 0 C A 8 6 4**

**B 2 0 C A C 0 C**

* 1. Chan one Up (4 bits) Array
		1. For 1st 16 samples

**00 80 90 19**

* + 1. For 2nd 16 samples

 **16 13 10 0D**

 **2C 26 20 19**

 **2F 32 39 32**

 **00 01 00 19**

* 1. Number of leading zero and non-zero element of Up Array
		1. 1st 16 samples = **4 ; 3**
		2. 2nd 16 samples= **1 ; 15**
	2. Compression efficiency
		1. Input = 16 Sample \* 12 bits/sample = 192 bits
		2. Over Head output:
			1. Low Array = 64 bits
			2. Min Value = 12 bits
			3. NumOfZero = 8 bits
			4. Total = 84 bits
		3. 1st 16 samples
			1. Up Array = 24
			2. Total bits = 24 + 84 = 108
			3. Efficiency = (1- 108/192) \* 100 = 43.75%
		4. 2nd 16 samples
			1. Up Array = 120 bits
			2. Total bits = 204 bits
			3. Efficiency = -6.25%
		5. Total Efficiency for Channel 1 = 37.5%
	3. Channel sixteen 32 Samples Input in Decimal (1 long pulse)
		1. Sample number 3 2 1 0
		2. 213 212 213 213 213 213 212 212 500 600 700 800 800 800 800 800
		3. 700 600 500 400 213 212 213 213 213 213 212 212 213 212 213 200
	4. Channel 16 minimum value
		1. 1st 16 sample = **212**
		2. 2nd 16 sample = **200**
	5. Chan 1 Subtract Min Value from sample in decimal
		1. 1st 16 sample

1 0 1 1 1 1 0 0 288 388 488 588 588 588 588 588 in decimal

1 0 1 1 1 1 0 0 120 184 1E8 24C 24C 24C 24C 24C in hex

* + 1. 2nd 16 sample

500 400 300 200 13 12 13 13 13 13 12 12 13 12 13 0 in decimal

1F4 190 12C C8 D C D D D D C C D C D 0 in hex

* 1. Chan one Up (4 bits) Array
		1. 1st 16 samples

 **0 4 8 C C C C C**

 **1 0 1 1 1 1 0 0**

* + 1. 2nd 16 samples

**D D C C D C D 0**

**4 0 C 8 D C D D**

* 1. Chan one Up (4 bits) Array
		1. 1st 16 samples

**58 58 58 58**

**28 38 48 58**

* + 1. 2nd 16 samples

**1F 19 12 C**

* 1. Number of leading zero and non-zero element of Up Array
		1. 1st 16 samples = **8 ; 8**
		2. 2nd 16 samples = **0 ; 4**
	2. Compression efficiency
		1. 1st 16 samples = [1- (148/192)]\*100 = %22.91
		2. 2nd 16 samples = [1- (116/192)]\*100 = %39.58
		3. Total = %62.45
	3. 32-bits Data Output
		1. Name Value
		2. Event Header Pass through
		3. Trigger Time #1 Pass through
		4. Trigger Time #2 Pass through
		5. Compress Data Header 00000020
		6. Ch one 1st 16-sample header 00206240
		7. Ch one 1st 16 sample low array 20222222
		8. Ch one 1st 16 sample low array 23512664
		9. Ch one 1st 16 sample up array 00800019
		10. Ch one 2nd 16-sample header 0020581F
		11. Ch one 2nd 16 sample low array 840CA864
		12. Ch one 2nd 16 sample low array 120CAC0C
		13. Ch one 2nd 16 sample up array 1613100D
		14. Ch one 2nd 16 sample up array 2C262019
		15. Ch one 2nd 16 sample up array 2F323932
		16. Ch one 2nd 16 sample up array 00010019
		17. Ch sixteen 1st 16-sample header 01E05884
		18. Ch sixteen 1st 16 sample low array 048CCCCC
		19. Ch sixteen 1st 16 sample low array 10111100
		20. Ch sixteen 1st 16 sample up array 58585858
		21. Ch sixteen 1st 16 sample up array 28384858
		22. Ch sixteen 2nd 16-sample header 01E0c804
		23. Ch sixteen 2nd 16 sample low array DDCCDCD0
		24. Ch sixteen 2nd 16 sample low array 40C8DCDD
		25. Ch sixteen 2nd 16 sample up array 1F19120C
		26. Event Trailer Pass through
1. **Firmware Architecture**
	1. Top Level Firmware Architecture

Figure 1 shows the top level firmware architecture.

There are 3 Modes of operation:

* Mode = “00” 🡺 Data In is Pass through to Data Out.
* Mode = “01” 🡪 Data In is output to Data Out and is followed by Compress Data. In this mode, the number of active channel \* number of ADC sample (PTW + 1) has to be less than 1601. This mode only run when PTW + 1 is multiple of 16
* Mode = “10” 🡪 Only Compress Data is output to Data Out. This mode only run when PTW + 1 is multiple of 16

When DataValid is high DataIn is accepted.

When PTW + 1 (number of sample) is is NOT a multiple of 16 and Enable is high, CE Error is high. PTW is the user programmable parameter.

Figure1 : Top Level Firmware Architecture

CA Fifo WrEn

CA Data

32

9

Mode

36

Compressing Architecture

Data

In

PTW

(Number of sample)

FIFO 36x1K

Data

Out

0

1

Data Valid

CA Error

* 1. Compressing Architecture

Figure2 shows the Compressing Architecture (CA). The behavior of the CA depends on bits Comp Enable, PTW, and DataIn 35-27 as shown in Table 1 and Table 2.

Figure2 : Compressing Architecture

3

CA Error

PTW

(Number of sample)

9

Enable

36

Data

In

CA Data

Decode

Bit 35-27

Compressing Engine

0

1

CA Fifo WrEn

Mode

 Table 2: Behavior of CA when Data Valid is 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mode | PTW | DataIn 31-27 | CA Error | CA Fifo WrEn | CA Data |
| 00  | x | x | 0 | 1 | DataIn |
| 01 | Not multiple of 16 | x | 1 | 1 | Only DataIn |
| 01 | Multiple of 16 | See Table 3 | 0 | 1 for Datain to CA dataSee Table 3 for compress data | DataIn. Compress Data |
|  |  |  |  |  |  |
| 10 | Not multiple of 16 | x | 1 | 0 | No output |
| 10 | Multiple of 16 | See Table 3 | 0 | See Table 3 | See Table 3 |
|  |  |  |  |  |  |

 Table 3: Behavior of CA when Data Valid is 1, Mode is “10” and PTW is even

|  |  |  |  |
| --- | --- | --- | --- |
| Data In 35-27 | CA Fifo WrEn | CA Data  | Comment |
| 0000 1 0010 | 1 | DataIn | Event Header |
| 0000 1 0011 | 1 | DataIn | Trigger Time Word 1 |
| 0000 0 0000 | 1 | DataIn | Trigger Time Word 2 (only when immediately follow 1 0011 |
| 0000 1 0100 | 1 | DataIn | Window Raw Data Word 1. Contains (PTW+1) and channel number. |
| 0000 0 0000 | 1 | Compress Engine | Window Raw Data Word 2 - N |
| 0000 1 1001 | 0 | DataIn | Pulse Parameter Word 1 |
| 0000 1 1xxx | 0 | DataIn | Pulse Parameter Word 2(only when immediately follow 1 1001 |
| 0000 0 0000 | 0 | DataIn | Pulse Parameter Word 2 (only when follow 1 1001 by 1 word) |
| 0010 1 1101 | 1 | DataIn | Event Trailer |

**C.** Compressing Engine

The Compressing Engine executes the **Pulse Compressing Algorithm** as described in SECTION 2 above

**D.** Steps to input data into Firmware

* Select Mode
1. = pass through;
2. = pass through and follow by compress data
3. = compressing data only
* Set PTW
* When Data Valid is one, DataIn will be acted upon. In other word Data Valid is used as Data strobe signal
* The first 3 words HAS to be header, trigger time 1, trigger time 2
* The last word has to be End of Event