Technical Information Manual

Revision 2.1
Juin 2007

VF48 DAQ System

VF-48 ADC Channels

How To Contact Us

For technical support on this product, you can contact:

Martin, Jean-Pierre, Ph. D.
Université de Montreal,
2905, Chemin des Services, Local 106
Montreal (Quebec)
H3T 1J4, Canada
(514) 343-7340
jpmartin@lps.umontreal.ca

Mercier, Christian, ing. jr.
Université de Montreal,
2905, Chemin des Services, Local 106
Montreal (Quebec)
H3T 1J4, Canada
(514) 343-6111 #4199
christian.mercier@polymtl.ca

Warning

This file is a sketch of specifications. It is not absolutely exhaustive, but should allow to use the card VF48 (Kopio). We decline all responsibility for damages or injuries caused by an improper use of the modules due to negligence on behalf of the user. For any question, you can contact us.

Contents

1. General Description

1.1. Overview

The VF48 is a one-unit wide VME 6U housing 48 channels digitized by six fast 10-bit ADCs. For each channel, the input signal (50 Ω impedance) is amplified by a high speed differential amplifier (*AD8132*) and them converted by the ADCs at a maximum rate of 60 Msps.

The digitized signal goes to a FPGA (*Cyclone EPC12*) which will calculate the charge of the signal, evaluate the exact trigger time, build an event which will contain all this information and send the event constructed to a local collector. Each FPGA manages eight channels. There are six FPGA connected to the ADCs and one FPGA which collects the data.

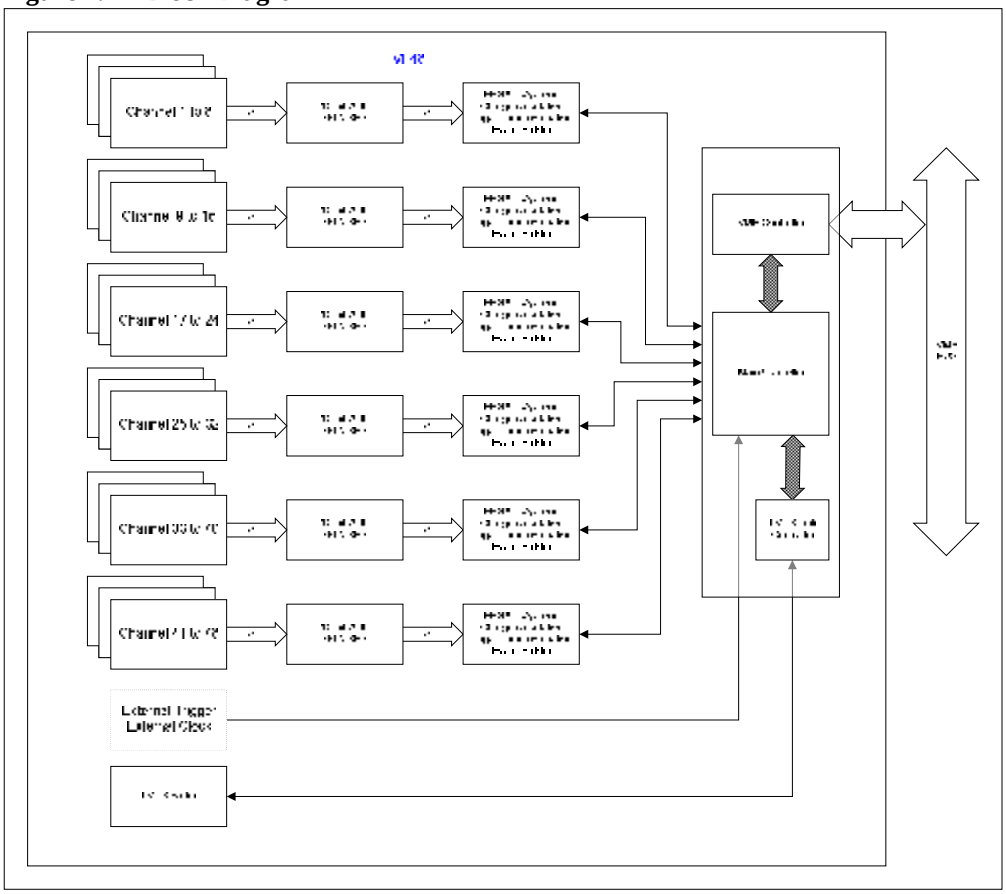
The board can be accessed in A24/A32/A40 addressing mode, D16, D32 and MBLT64 data transfer mode.

It has also a board available to access the VF48 throw a LVDS link. This link can transfer the data at a speed up to 100 Mbits/s. This link can afford all the control necessary to run the VF48. It can accept a clock which permits the synchronization of many VF48 units.

The board contains a trigger structure which permits to trigger the data in three different manners. The trigger can come from an external NIM signal, from a software command or from an internal mechanism. The internal mechanism manages multiple channels hit, deadtime, threshold and multiplicity and is totally programmable by the user.

1.2. Block Diagram

Figure 1.2 – Block Diagram



2. VME Interface

2.1. Transfer Capabilities

2.1.1. Addressing Capabilities

The VF48 board can access the VME bus in VME A24. See Table 2.1.1 to have more information about the Address Modifier accepted.

The Table 2.1.1 summarizes all the supported address Modifer.

Table 2.1.1 – Address Modifier Recognized

AM	Description	Available in version earlier to 2.0	Available in version 2.0 and more
3F	A24 supervisory block transfer (BLT)	Ö	Ö
3E	A24 supervisory program access	Ö	Ö
3D	A24 supervisory data access	Ö	Ö
3C	A24 supervisory 64-bit block transfer (MBLT)		Ö
3B	A24 non privileged block transfer (BLT)	Ö	Ö
3A	A24 non privileged program access	Ö	Ö
39	A24 non privileged data access	Ö	Ö
38	A24 non privileged 64-bit block transfer (MBLT)		Ö
37	A40 BLT	Ö	Ö
34	A40 Access	Ö	Ö
0F	A32 supervisory block transfer (BLT)	Ö	Ö
0E	A32 supervisory program access	Ö	Ö
0D	A32 supervisory data access	Ö	Ö
0C	A32 supervisory 64-bit block transfer (MBLT)		Ö
0B	A32 non privileged block transfer (BLT)	Ö	Ö
0A	A32 non privileged program access	Ö	Ö
09	A32 non privileged data access	Ö	Ö
08	A32 non privileged 64-bit block transfer (MBLT)		Ö
03	A64 block transfer		
01	A64 single transfer access		
00	A64 – 64-bit block transfer (MBLT)		

2.1.2. Base Address

The base address of the board is 0xA00000 + n · 0x10000. The number n is defined by the alignment of the switch on the board. For example, if the alignment on the switch is : ID[3..0] = 1011, then the base address will be 0xAB0000.

Every board will manage a window going until 0x00FFFF above the base address. In the previous example, the VME window was from 0xAB0000 to 0xABFFFF.

2.1.3. Data Transfer Capabilities

The board can access the VME BUS in VME_D16, VME_D32 and VME_D64 mode. The VME_D8 is not compatible. It is not each address that can be access in all modes. See VME Mapping to have more details on this.

2.2. VME Mapping

The Table 2.2.1 describes the VME Mapping. In the current version, we don't care of the 4 LSB of the address. In the Table 2.2.1, the W column indicates if the register is available for a Write. The R column indicates if the register is available for a Read. The last three columns indicate respectively if the register can be accessed in VME_D16, VME_D32 and VME_D64.

Table 2.2.1 – VME Mapping

Offset	VME Mapping	W	R	D16	D32	D64
0x000	CSR	X	X	X	X	X
0x010						
0x020						
0x030	Firmware ID		X	X	X	X
0x040						
0x050	Param DAT	X		X	X	X
0x060	Param ID	X		X	X	X
0x070	Soft Trigger	X		X	X	X
0x080						
0x090	Group Enable	X	X	X	X	X
0x0A0	NbrFrames		X	X	X	X
0x0B0	Global Reset	X				
0x1XX	Event Data		X		X	X

2.2.1. CSR Description

Only six bits of the CSR (Control & Status Registry) are used. They are indicated in the Table 2.2.2.

Table 2.2.2 – CSR Description

Bit	CSR Description	W	R
0	Run	x	X
1	Parameter ID Ready	-	X
2	Parameter DATA Ready	-	X
3	Event Fifo Empty	-	X
4		-	X
5	CRC Error detected	-	X
6		-	X
7	External Trigger	x	X
8		-	X

The **RUN bit** allows the system to accept triggers and start the acquisition.

The **Parameter ID Ready** bit and **Parameter DAT Ready** bit indicate if parameter is ready to be read. The parameter section will describe more precisely how these bits work.

The **Event Fifo Empty** bit indicates if the event fifo is empty. When this signal is low, there's data ready to be read.

The **CRC Error Detected** bit indicates if a CRC Error has been detected in the serial communication on the board. This bit should never go on.

The **External trigger** bit selects the external trigger. If this bit is set to 1, the trigger that will start the acquisition must come from the connector on the front panel. If this bit is zero, then a real signal must be detected on one of the 48 channels if you want to have event built.

2.2.2. Firmware ID (0x000030)

It returns the firmware ID

2.2.3. Parameter DAT (0x000050)

Register used to write the parameter. See the section Writing/Reading Parameter for more details.

2.2.4. Parameter ID (0x000060)

Register used to write the parameter. See the section Writing/Reading Parameter for more details.

2.2.5. Soft Trigger (0x000070)

It generates a trigger on the front end to force the system to acquire data.

2.2.6. Group Enable (0x000090)

It allows enabling a group of channels. Bit 0 enables channel 1 to 8, bit 1 enables channel 9 to 16 until bit 5 which enable channels 41 to 48. By default, they are all enabled.

2.2.7. Nbr Frames (0x0000A0)

It gives the number of data present in the event fifo. See section 4. Triggers & Events for more details.

2.2.8. Global Reset (0x0000B0)

It resets all the system.

2.2.9. Event Data (0x000100-0x00FFFF)

It reads all the data in the fifo. See section 4. Triggers & Events for more details.

3. Writing/Reading Parameter

To write a parameter, you must write the parameter ID (See Table 3.1.1) followed by the parameter Data. You write the parameter ID by doing a VME Write at the address Param ID and you write the parameter Data by doing a VME Write at the address Param DAT.

During a parameter read, it is necessary to be aware that parameter is not immediately in the register ParamDAT. The bit 2 of the CSR rises up when the parameter is ready and falls down when you read it. So, the procedure to read a parameter is this. First, you send the request by writing the parameter ID (VME Write at the address ParamID) with bit 7 set to 1. You must also send a fake data, because if you don't send a fake data, the request will never be sent. After, you read bit 2 of the CSR until it has been set. When the bit is set, you do a VME Read at the address ParamDAT[1].

[1] Do not forget to put a protection because it already happened that bit has never been set. If the bit is not set 2 us after the request, you should send a new request.

3.1. Parameter Frame Header

The first 16 bits of a frame contain information describing the contents of the parameter.

Table 3.1.1 – Parameter Frame Header

15				11				7	6						
C	C	C	C	D	D	D	D	R	V	P	P	P	P	P	P

P : Parameter ID
R : ReadBit
D : Destination Channel (0-5)
V : Version (0- No extension; 1- Param ID 32 bit)
C : Destination Card or Port or Cyclone

Description of the fields *Parameter Frame Header*:

Destination Card

Bits 11 - 8 contain the number of the card where parameter must be sent. The following table describes the direction which has to take parameter according to the number of destination of card, the number of port or the number of cyclone on the board.

Table 3.1.2 – Destination Cyclone

11	10	9	8	Destination Cyclone	Port
0	0	0	0	Channel 1 to 8	0
0	0	0	1	Channel 9 to 16	1
0	0	1	0	Channel 17 to 24	2
0	0	1	1	Channel 25 to 32	3
0	1	0	0	Channel 33 to 40	4
0	1	0	1	Channel 41 to 48	5
0	1	1	0	Not used	Not used
1	1	1	1	Not used	Not used

Destination Channel

This field is not used for the current version. We don't care of these bits.

ReadBit

Bit 7 indicates if command is a demand of writing or reading of the parameter. If ReadBit is HIGH, then it is a read and parameter should be sent back to the collector. If ReadBit is LOW, then it is a write and parameter should be recorded.

During a parameter read, it is necessary to be aware that parameter is not immediately in the register ParamDAT. The bit 2 of the CSR rises up when the parameter is ready and falls down when you read it.

Version Bit

This bit must be zero in this version

Parameter ID

The last six bits contains the parameter ID. Each parameter available is displayed in the Table 3.1.3

Table 3.1.3 – Parameter List

ID#	Parameter List	Default Value	W	R
0				
1	PED	0x0000	X	X
2	Hit Det Threshold	0x000A	X	X
3	Clip Delay	0x0028	X	X
4	PreTrigger	0x0020	X	X
5	Segment Size	0x0100	X	X
6	K	0x0190	X	X
7	L	0x0200	X	X
8	M	0x1000	X	X
9	Channel Enable	0x00FF	X	X
10	Mbits 1 - Feature Enable	0x0000	X	X
11	Mbits 2 - Feature Enable	0x0000	X	X
12	Latency	0x0005	X	X
13	Firmware ID	0x0207		X
14	Attenuator	0x0190	X	X
15	Trigger Threshold	0x000A	X	X

3.2. Parameter Description

This section is there to describe each parameter.

1 - PED

This parameter is not used yet.

2 – Hit Detection Threshold

This parameter defines the threshold to detect a hit. If the signal given by the ADC is above this threshold value, the system will start to calculate the charge, but if the signal doesn't reach the trigger threshold, the charge won't be used.

3 – Clip Delay

This parameter is not used yet.

4 – Pre-Trigger

This parameter defines the number of clock where the data must be recorded before the trigger.

5 – Segment Size

This parameter defines the number of raw data that must be recorded.

6 - K

This parameter defines the peaking time. See the signal processing section for more details.

7 - L

This parameter represents the duration of the peaking time and the flat top together. See the signal processing section for more details.

8 - M

This parameter defines a multiplication factor of the convoluted signal. See the signal processing section for more details.

9 – Channel Enable

This parameter indicates the channels that must be included in the event. Bit 0 enables channel 0, bit 1 enables channel 1, so in succession until channel 7. By default, all channels are enabled.

Note: In version previous to version 2.0.7, the parameter 11 was FeatureDelay_B. This parameter doesn't exist anymore.

10 – M_Bits

This parameter is a parameter where each bit has a different purpose. The Table 3.2.1 gives information about these bits.

Table 3.2.1 – Description of the MBits parameter

Bit	Version 2.0 and more	Version before 2.0
0	Data Simulation	Data Simulation
1	Supress Raw Data	Supress Raw Data
2	Select Corrected Data	Select Corrected Data
3	PolPlus	PolPlus
4	BLR Speed (bit 0)	Disable ADC
5	BLR Speed (bit 1)	Fake One Data
6	Hold BLR	-
7		-
8	Disable ADC	-
9	Offset 1	-
10	Offset 2	-
11	Low Gain Selection	-
12	Card Revision Number (bit 0)	-
13	Card Revision Number (bit 1)	-
14	Card Revision Number (bit 2)	-
15	Card Revision Number (bit 3)	-

The bit 0**, Data Simulation**, enables the simulator. If this bit is set, the simulated data will be continually sent.

The bit 1, **Suppress Raw Data**, suppresses the raw data independently of the segment size parameter.

The bit 2, **Select Corrected Data**, is not used in the current version.

The bit 3, **PolPlus**, is used to invert numerically the input. If the input is 0b0100010001, and the bit PolPlus set then the input will be 0b1011101110.

The bit 8, **Disable ADC**, disable the ADC.

The bit 5, **Fake One Data**, has the same use as the bit 0, Data Simulation, but it will be activated for only one valid signal.

All the other bits are reserved for future use. The names indicated represent the function that is reserved for the bit to be compatible with other project.

11 – M_Bits 2

This parameter is a parameter where each bit has a different purpose. The Table 3.2.2 gives information about these bits.

Table 3.2.2 – Description of the MBits 2 parameter

Bit	Version 2.0 and more	Version before 2.0
0	Enable Channel Suppression	-
1	Doesn't send time evaluation	-
2	Doesn't send charge calculated	-
3		-
4		-
5		-
6		-
7		-
15..8	Sampling Rate Divider	-

The bit 0**, Enable Channel Suppression**, disable an entire channel within a group if none of the sample values of that channel is above the **hit threshold**.

The bit 1**, Doesn’t send charge calculated**, disables the calculation of the charge. If this bit is set, the charge won’t be calculated.

The bit 2**, Doesn’t send time evaluation**, disables the evaluation of the time when the signals passes above the threshold. If this bit is set, the time won’t be evaluated.

Note: In version previous to version 2.0.7, the parameter 11 was FeatureDelay_B. This parameter doesn’t exist anymore.

12 – Latency

This parameter defines the number of clock to be added to the pre-trigger. This parameter should represent the time between the hit detection and the trigger accepted received by the front end.

13 – Firmware ID

This parameter is firmware ID. For example, if the version is 1.0.0, it will return 0x0100. If the version is 11.12.13, it will return 0x0BCD.

14 – Attenuator

This parameter defines the attenuation of the integration. See the signal processing section for more details.

15 – Trigger Threshold

This parameter defines the threshold to accept a trigger. If the signal given by the ADC is above the threshold value, a trigger request will be sent to the collector.

4. Triggers & Events

When a trigger is accepted, the front end starts to build an event which will be transferred to the collector. Each event has the same format. This section defines this format, but before, let us defines some terms usually used in this section.

A **Packet** is a word of 32 bits which contains information related with an event.

An **Event** is a series of packet beginning with a header and ending by a trailer.

4.1. Trigger Detection

Each channel goes through a comparator in order to trigger the feature evaluation. The hit threshold is a **positive** difference of 2 ADC values separated by 2 samples (ex: 2,3,4,5,6,7 **à** hit Thr= 6-3).

In the case the input signal has the opposite polarity, you can either reverse the input signal (the VF48 input is bipolar) or reverse the polarity by software. The polarity switch is applied to the digitized ADC values. Currrently no dedicated reverse polarity function is available. Use the example found under midas/examples/Triumf/c/fevmemodules.c/BOR function.

The acceptance of the next trigger is garanteed when there is space for one event or more. A corresponding deadtime is generated for the duration of the capture of the raw data in the frontend buffer. The buffer size is fixed at 1000 samples. Therefore the pipeline advantage starts when the event size is less than 500 samples (this limits is set by the hardware type used on the board).

There is an output signal (busy out signal) available reflecting the non acceptance of trigger by the VF48. This correspond to the "deadtime" from the raw data capture **ored** with the condition of the frontend buffer having no more room for a complete event.

4.2. Event Description

An event is broken in many packets which follow a precise order. It always has to begin with a **header** and end with a **trailer**. The header and the trailer contain the trigger number and they must be identical. The header is always followed by a **timestamp**. Then, for each channel enabled, it could contain the raw data, the CFD time evaluated and the charge calculated. The Table 4.2.1 shows the content of an event.

Table 4.2.1 – Event's content

Packet's Type	MSB
Event Header	0x8
Time Stamp 1	0xA
Time Stamp 2	0xA
Channel ID	0xC
Raw Data (of current channel)	0x0
CFD Time (of current channel)	0x4
Charge (of current channel)	0x5
Channel ID	0xC
Raw Data (of current channel)	0x0
CFD Time (of current channel)	0x4
Charge (of current channel)	0x5
...	
Channel ID	0xC
Raw Data (of current channel)	0x0
CFD Time (of current channel)	0x4
Charge (of current channel)	0x5
Event Trailer	0xE

The number of raw data is configurable by the parameter *Segment Size* (*Chan Param ID:5*). The next section will describe more in detail each of these packets.

4.3. Event Packet Format

This section describes in detail the contents of each of the packets. Firstly, inside a packet, the 4 MSB indicates the type of packet. The next table makes the association between the MSB and the packets. See Table 4.3.1.

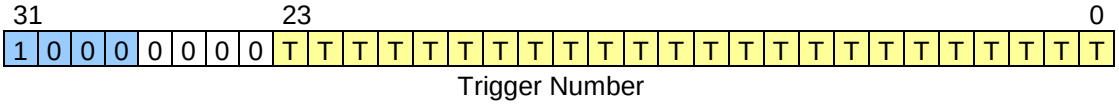
Table 4.3.1 – Event Packet MSB Association

Packet's Type	MSB
Header	0x8
Time Stamp 1	0xA
Time Stamp 2	0xA
Channel ID	0xC
Raw Data	0x0
CFD Time	0x4
Charge	0x5
Trailer	0xE
Header Error	0x9
Error	0xF

Header - 0x8xxxxxxx

The packet Header is always the first packet of an event. It contains the trigger number in the bits 23 to 0.

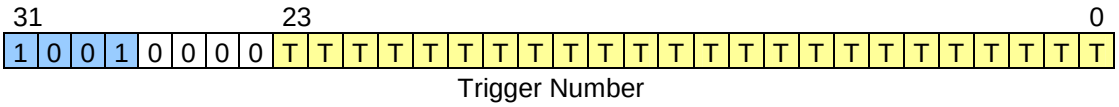
Table 4.3.2 – Header 0x8xxxxxxx



Header Error - 0x9xxxxxxx

The Header Error packet occurs when the first packet of a group was not a Header packet. This could be caused by an internal error. If this packet is detected, the current event must be rejected. However, this packet must not happen and if it does, you should contact your supplier.

Table 4.3.3 – Header Error 0x9xxxxxxx



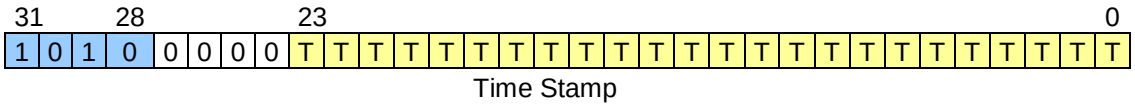
Time Stamp - 0xAxxxxxxx

The Time Stamp packet is the second packet and the third packet of en event. It contains a 48 bits time stamps. The time between two steps is of 25 ns[2]. Then, with a 48 bits

[2] This step is based on a sampling frequency of 40 MHz. In the case where the sampling frequency is 60 MHz or 64 MHz, the step will be respectively of 16,666 ns and 15,625 ns

timestamp, the maximum time measured is 7 036 874,418 seconds. Then, system can count time without completing a buckle during more than 81 days.

Table 4.3.4 – Time Stamp – 0xAxxxxxxx

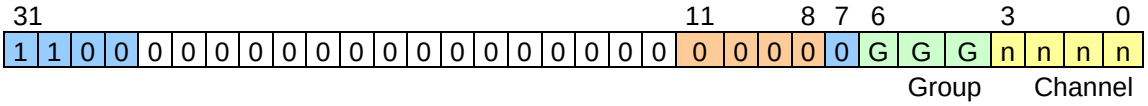


The first time stamp packet contains bit 47 to 24 and the second time stamp packet contains bits 23 to 0.

Channel ID - 0xCxxxxxxx

The channel ID packet is the first packet of a sub-event. It contains all the information to be able to identify the channel. Bit 3 to 0 indicates the channel number. This number can go from 0 to 7. It corresponds to the channel of a particular group. Then, bits 6 to 3 identify the group. They can so take a value going from 0 to 5. Bit 7 is not used in this version.

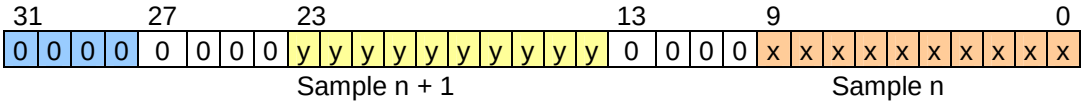
Table 4.3.5 – Channel ID – 0xC0xxxxxx



Data - 0x0xxxxxxx

The Data packet comes after the channel ID packet. It simply contains two data of 10 bits each. The number of data packet within an event is related to the segment size parameter (*Chan Param ID: 5*). It could arise problems to have a null value or an odd value of the segment size.

Table 4.3.6 – Data – 0x0xxxxxxx

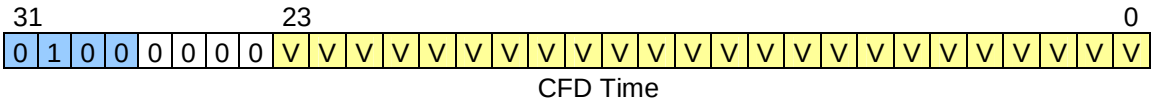


Be careful about the position of the second data. It doesn't start exactly to bit 16. It starts to bit 14. This is done to keep the compatibility with an ADC that would a 14 bits data.

CFD Time - 0x4xxxxxxx

The CFD Time packet is the second packet before the last packet. It contains the CFD Time.

Table 4.3.7 – CFD Time – 0x40xxxxxx

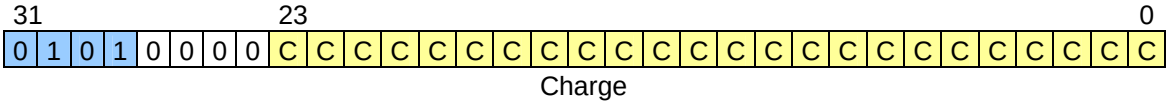


Time algorithm: CDF Equivalent. The input is first clipped with a delay of 3 clocks. If the clipped signal is above the *hit threshold* then the maximum value of the clipped waveform is evaluated. The CFD time is then evaluated by interpolating the time at 50% charge using the two samples on either side of the 50% fraction.

Charge - 0x5xxxxxxx

The Charge Packet is the first packet before the trailer packet. It contains the charge internally calculated.

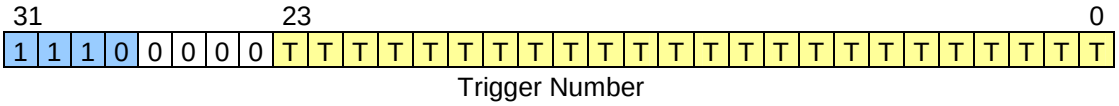
Table 4.3.8 – Charge – 0x50xxxxxx



Trailer - 0xExxxxxxx

The trailer is always the last packet of an event. It contains the trigger number in the bits 23 to 0. The bit 28 indicates an out of sequence flag, then, if this bit is 1, that means that an internal error took place and the packet should be rejected. In theory, system is stable and this type of error should not occur.

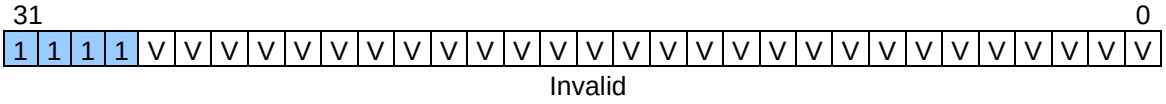
Table 4.3.9 – Trailer – 0xExxxxxxx



Error - 0xFxxxxxxx

The Error packet occurs when a particular packet is not at his right position. This could occur if a timestamp is missing or if a timestamp is not followed by channel. This could be caused by an internal error. If this packet is detected, the entire event must be rejected. However, this packet must not happened and if it does, you should contact your supplier.

Table 4.3.10 – Error 0x9xxxxxx



4.4. Event Acquisition

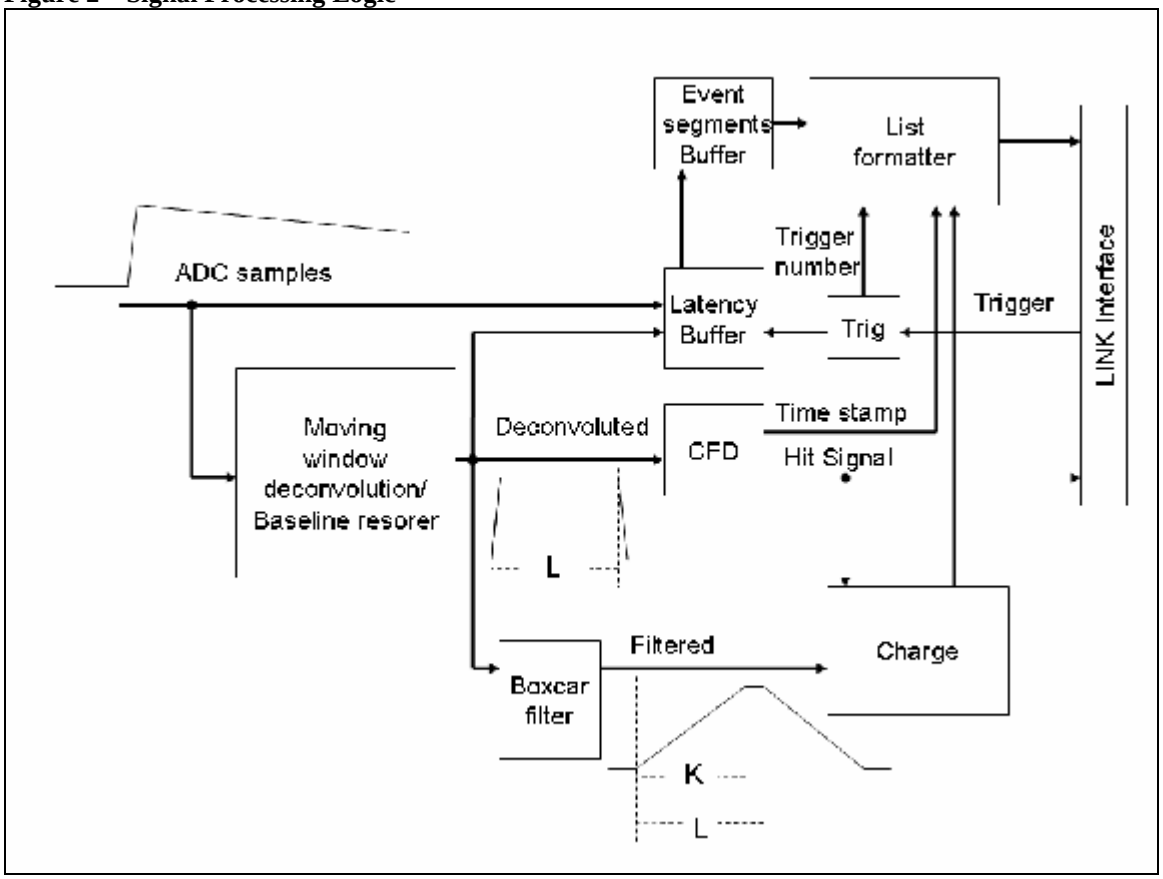
Procedure to read events is very simple. It is a question of reading the number of packet presently available by making a reading on the bus VME at the address *NbrFrames*. Then, a reading in burst of the number of packet available can be made. The reading of events is done in fifo mode only. This means that the reading is done always at the same address, the *Event Data* address.

To start the acquisition, you must set the bit 0 of the CSR to 1. To stop the acquisition, you set the bit 0 of the CSR to 0.

5. Signal Processing

The system integrates a signal processing logic which calculates the charge and the precise time where the hit has been detected. In the FE FPGA the continuous flow of digitized signal samples from the ADC is directed to the signal processing logic. The signal processor runs continuously. It performs tasks equivalent to a spectroscopy amplifier connected to an analog multi-channel analyzer.

Figure 2 – Signal Processing Logic



5.1. Real-Time Digital Filter and Charge Evaluation

The filtering involves many steps. The first step is the deconvolution of the exponential tail of the pulse shape within an arbitrary time window. The moving window deconvolution method is well documented in the litterature. It is implemented as a finite impulse response filter (FIR). If D_0, D_1, … D_n, represent the digitized signal samples, and L the span of the moving window, the nth point of the transformed sequence F_n is given by the relation:

$$F_n = D_n - D_{(n-L)} + \frac{1}{t}\sum_{i=1}^{L} D_{(n-i)} \quad (1)$$

where τ is the exponential time constant of the signal in units of data samples. The deconvolution is useful in the case of a germanium detector to compensate for the differences in ballistic deficit as a function of the rise time of the signal. Depending on the position of the photon interaction, the charge collection time may vary from event to event by as much as 150 nanoseconds. With a preamplifier decay time constant of 50 microseconds, this translates into a fluctuation of the raw signal peak amplitude corresponding to 3 Kev for a 1 Mev gamma. With an analog system, this fluctuation is reduced significantly by using a long filter peaking time, but it never vanishes. With the moving window deconvolution method, the tail pulse shape is transformed into a quasi-rectangular pulse shape, with a duration determined by the span of the moving window. The transformed signal has a leading edge reflecting the shape of the original signal (with a slight correction for the exponential decay), and then reaches a constant maximum value that is exactly the same for a given total charge deposited in the detector whatever the rise-time. The processed signal also returns to zero with no tail.

It can be seen from equation (1) that the first two terms will cancel the DC baseline. However, the sum term that cancels exactly the exponential decay will unfortunately respond to a baseline level with a gain of L/τ. The sum term also limits the effectiveness of the low frequency filtering effect of the first two terms. To alleviate this problem, one can resort to double sampling, or use a baseline restoring scheme. We have chosen this later approach. The baseline restoring process is active only when no signal is present.

At this stage, the filtering is not yet complete. Only the low frequency part of the noise spectrum has been attenuated. The high frequency filtering is performed by applying another FIR transformation to the corrected deconvoluted signal. The most efficient filter for the evaluation of the trend of a noisy straight line (the flat part of the rectangular pulse shape) is the simple floating average transformation, commonly called a *boxcar* filter:

$$G_n = \frac{1}{K}\sum_{i=0}^{K-1} F_{(n-i)} \quad (2)$$

K is chosen to be as close as possible to the size of the flat portion of the rectangular pulse F_n. For a theoretical signal having a zero rise time and an exponential decay with time constant equal to τ, the sequence of points G_n represents a trapeze with a rise time of K, a flat portion of L-K, and a fall time of K. Every point of the flat top part of the pulse is a proper evaluation of the charge. Selecting the maximum value is simple, but it generates a small average bias proportional to the noise. We rather select the measurement point at a pre-determined time after the beginning of the pulse, based on the timing information produced by the CFD discriminator. This amounts to selecting the point at random (as far as the amplitude is concerned) in the flat top region, thereby reducing the bias.

Due to constraint design (the division by τ would involve excessive FPGA resources), we multiply both sides of equation (1) by τ (parameter M), and we do not normalize by the number of points. The result appearing in the event list is given by

$$\text{Charge calculated} = \frac{\text{K} \times \text{M} \times \text{Convoluted Signals}}{128}$$

It can be scaled down by software when the histograms are constructed.

5.2. Digital Constant Fraction Discriminator

Analog constant fraction discriminators (CFD) are widely used to minimize the time walk associated with the detection of signals featuring widely varying amplitudes. The same principle can be applied to digital signal samples, with some differences in the physical implementation. In the VF48 firmware, the implementation reproduces exactly the definition of a CFD: The amplitude of the signal is evaluated, and a threshold is calculated with a predefined fraction. The same signal, delayed through a digital delay line (dual port RAM), is then compared with this calculated threshold until the point immediately below, and the point immediately above the threshold are found. Then, a linear interpolation is performed to evaluate the time corresponding to the threshold crossing. We use time units of 1/16 of the ADC sampling clock period for the interpolation. Often, the rise time is not constant from event to event, as it depends on the position where the electron-holes pairs were created in the crystal. So, it is important to clip the incoming signal to a width equal or shorter than the fastest signal rise time envisioned before the CFD logic. This is achieved by subtracting two samples separated by the proper number of sampling clocks. This has the same effect as the clipping delay line of an analog CFD. The digital CFD produces two outputs: a logic signal synchronized with the ADC clock, when the constant fraction threshold is crossed, and a higher precision time stamp word. The logic signal is used by the trigger logic, whilst the time stamp is part of the event data stream.

5.3. Latency and Event Segment Buffers

In order to accommodate for the latency of the trigger system, a circular latency buffer is used to keep the past values of the signal samples available for a time equal or longer than the trigger decision latency. The input of the buffer is either the raw data signal, or the output of the deconvolution logic. This can be selected by a run parameter. The maximum capacity of the latency buffer is presently 512 elements. When a trigger is accepted, this delay allows the recovery of the associated signal since its very beginning. We usually also include a few samples that have occurred before the signal. The latency buffer is implemented in a dual port random access memory running continuously. The read address is equal to the write address minus the number of clock cycles we want to have the data delayed. The signal data is continuously written in the memory, and the

delayed data continuously available on the readout port. When a trigger is accepted, a transfer gate is generated for a duration corresponding to the size of the data segment requested, and the data read out is transferred to the next buffer stage: the segment event buffer.

The segment buffer is a simple 1024 word FIFO that stores the waveform data segments, plus two extra bits indicating the beginning and end of these segments. The FIFO is read out asynchronously by the list formatter. The function of the Segment Buffer is to store a few events in order to deramdomize the data flow. The segment buffer also generates an almost full flag that is transmitted to the master trigger system. This is the mechanism that throttles the trigger rate when the data acquisition becomes throughput limited. When the throughput is significantly lower than the system bandwidth, the data acquisition is dead-timeless.

6. Firmware Update

To do an update, you will need the *ByteBlaster II* connector connected on the parallel port of your PC and to the connector J3 on the VF48 board. (Or an *USB Blaster* module)

Then, you must download the programmer on the Altera Website:
https://www.altera.com/support/software/download/programming/quartus2/dnl-quartus2_programmer.jsp

Then, you start the program. You change the mode from *JTAG* to *Active Serial Programming*. You push the *Hardware Setup* button and you select *ByteBlaster II*. Then, you push the *Add File* button and you select the pof file that we supplied you. After, you click in the *Program/Configure* case and finally you push the *Start* button.

When the firmware has been programmed, you must unplug the *ByteBlaster II* connector from the VF48 board, shut down the power and put it back.

If the green light is on, programming was completed successfully. If the green light is off, did you forget to unplug the *ByteBlaster II* connector? If no, restart all the procedure.

6.1. Label Convention

A label convention is done to name each version that is created. It starts by the firmware main revision. Then, we add respectively the sub revision, the sampling frequency, the system clock frequency and the special features if they exist.

For example, the name VF48_V207_X6_40_20_Alpha.pof is associated to the firmware version 2.0.7 with a sampling frequency of 40 MHz and a system frequency of 20 MHz. The special feature Alpha signifies that this version is done specifically for Alpha project.

6.2. Firmware History

Version	Orig. Ver.	Front End New Feature & Modification	Collector New Feature & Modification	Hardware Compatibility	By	date
1.0.0	TigCol v2.1.1		LVDS Link adapted to 6 Ports Parameter adapted to 6 Ports VME Interface adapted to Kopio Mapping New Event Builder	1.0	CM	
1.0.1	1.0.0	Run bug resolved Reset bug resolved Event Builder bugs resolved	Run bug resolved Reset bug resolved Event Builder bugs resolved CRC error in CSR	1.0	CM	2005-12-13
1.0.2	1.0.1		CSR Assignment correction Parameter bug resolved	1.0	CM	2005-01-05
2.0.0	1.0.2		Assignment changed VME removed	1.1	JPM	2006-05-05
2.0.1	2.0.0		New VME Interface VME64 corrected	1.1	JPM	2006-10-10
2.0.2	2.0.1		Phase adjustment	1.1	CM	2006-10-13
2.0.3	2.0.2	Charge & CFD integrated in Event Builder	48 channels integrated in one event	1.1	CM	2006-10-19
2.0.7	2.0.3	1. Pipelined events (The acceptance of the next trigger is garanteed when there is space for one event or more) 2. Output signal available reflecting the non acceptance of trigger by the VF48 3. Channel Enable **within** a group 4. Implement Hit threshold 5. CDF Time algorithm 6. Raw data suppression by group 7. Channel Suppression		1.1	JPM	2007-02-14
2.0.8	2.0.7		Parameter ID can now be higher than 0x0F Bug during parameter write resolved	1.1	CM	2007-2-19
2.1.0	2.0.8	Minor modifications	Minor modifications	1.1	CM	2007-5-11