The VSAT bunch train scheme

Henrik Carling, Göran Jarlskog, Per Jonsson, Björn Lundberg, Ulf Mjörnmark
Lund University, Lund, Sweden

Abstract

This report describes how the VSAT detector runs with the bunch train scheme that is operational from 1995. Minimal modifications is done to the pre-1995 electronics. All new electronics is placed on a FASTBUS card. This card contains a peak finding system that issues the HOLD. FlashADCs and a CPU is used to monitor the performance of the system.

February 20, 1996
Comments to : *jum@quark.lu.se*
This note is available through WWW:
http://www.quark.lu.se/activities/delphi/delphi.info/documentation/minibunches.ps

1 Introduction

Since the minibunches are separated by a few hundred ns, the total time during which a hit in the detector is possible is in the order of 1us. The signals from the detector during a hit are a about 0.5us, which means that it is not sufficient to do a HOLD on the signals a fixed time after a bunch crossing - this has been the case until 1994. Furthermore it is not only desirable to measure the pulse height, but also to correlate the pulse to a specific minibunch.

The very front end electronics is based on a sample and hold circuit. No modifications is done on this electronics. Instead is a peakfinding system added. Minor modifications is done on the transmission of signals between FASTBUS and the MUX card crates.

A gate is opened for each minibunch at the time when we expect a peak. A peak finder will then try to find a peak within this gate. If a peak is found a HOLD is sent to the electronics. A HOLD is awlays sent after the last minibunch.

The VSAT trigger will not look at the minibunchstructure, but consider the whole train as a single bunch. The on/offline software has to use the minibunch number stored for each VSAT module to find out the "true" trigger type.

The peak finding electronics is implemented on a FASTBUS card, that contain FlashADCs and a CPU which will monitor the performance of the peakfinding system.

2 Hardware

This section describes in some detail the signal logic, the FASTBUS card and the changes needed on the existing system. It is possible to bypass the logic, in the sense that the HOLD signals are directly taken from WNG1 (strips) and WNG2 (FADs). This will make it possible to run with pre-1995 conditions. This may be useful for debugging and TESTIN runs.

2.1 Signal timing

The timing of the signals is shown in figure 2. A fixed time (dt1) after a BCO (Beam Cross Over) WNG1 enables the gate generator. During each gate (dt4) the peak sensor is active, and looks for a peak in the incoming sum of the FADs.

If a peak is found and the sum is above the threshold (the or of LOW and HIGH), a peak is indicated. A HOLD to the FADs is sent, followed a fixed time later (dt3) by a HOLD to the strips.

If no peak is found for any of the minibunches in the train, WNG2 disables the gate generator (after the time dt2) and a HOLD is generated.

This system is used individually for the four modules, i.e. we may get data from different minibunches for the different modules.

The delay from the time of the peak until the HOLD is sent, may not exceed 100 ns.

The readout is started when the signal READOUT is received from the LTS.

The logic is implemented on a FASTBUS card, described in section 2.2.

2.2 FASTBUS card

The FASTBUS card is named LURFB (Lund University Risc FastBus). The logic of the LURFB card is shown in figure 3. It is logically divided into three parts:

- Peak finder and HOLD generation.
- Flash ADC system
- CPU part

The peak finder and FADC parts, are separate for each VSAT module.

The peak finder is used to find a peak in the analog sum of the FAD signals, the Flash ADC system gives the shape of the sum as digital values in the data stream. The CPU performs formatting of the data, writing it into the FEB (Front End Buffer), monitors and controls the system.

2.2.1 Peak finder and HOLD generation

When the HOLD is sent, the current minibunch number is stored in the status register 1,as well as the current status of its threshold line. There are three bits for each module, i.e. we can accept at most eight minibunches (numbered 0 to 7), in addition to the single bit for the threshold status.

In order to generate the signals with the proper timing, there are programmable registers for:

- Bunch interspace time
- Number of minibunches
- Delay between HOLD to FAD and HOLD to strips

The number of minibunches and the bunch interspace time must be provided from the central DELPHI Data Acquisition System.

A separate hold is generated for each VSAT module, and sent to the auxilliary backplane. By setting a bit in CSR0 is it possible to bypass the peak finder. In this case are the HOLDs taken directly from WNG1 and WNG2:

- WNG1 to all the strip planes
- WNG2 to all the FADs

2.2.2 Flash ADC system

The 8-bit flash ADC starts sampling the summed signals at WNG1, and continues until ca 400 ns after the leading edge of WNG2. The time after WNG2 is needed to get the complete shape for the last minibunch. The data is stored in a FIFO (figure 5). The sampling rate is 20 Mhz, i.e. one sample each 50 ns. If the maximum sampling time is 750 ns (Bunchtrain length) + 400 ns = 1150 ns, there is in total 24 32 bit words for each VSAT event.

The number samples to do can be programmed and is is taken from CSR17. The number of samples done is stored in SR3. If any FIFO gets full is a bit set in SR3.

2.2.3 CPU part

The CPU part is based on the TMS320C50 Digital Signal Processor (DSP). This system is developed in Lund as a part of a general purpose CPU card, it will e.g. be used on a VME card for the experiment H1 at DESY. The DSP reads the FIFOs, formats the data and stores it in a 32kx16-bit Dual Port Memory, which is interfaced to FASTBUS. The Dual Port Memory is as well used to download the DSP program via FASTBUS.

The DSP is booted by setting bit 0 in CSR19, this halts the processor. The program is downloaded into the Dual Port Memory, and the processor is started by resetting bit 0 in CSR19. The ROM bootcode of the DSP will then load the program from the DPM into the onchip program memory. The boot mode is 16 bit parallel mode, and the source page address should be set to the base page of the DPM.

Boot code format in DPM	
word	*description*
0	Program destination
1	N = Number of words to transfer - 1
2	First word
.	.
N+2	Last word
7FFF	Source page address and boot mode = 82(hex)

At warm boot, i.e. when the program is already loaded into the DSP program memory, the last address is set to the base page of the program address and the boot mode to warm boot (= 3).

2.2.4 CSR space

LURFB is using five CSR registers, CSR0, CSR16, CSR17, CSR18 and CSR19.

register	*bit(s)*	*description*
0 read	16-31	FASTBUS ID = 6911
16 write	0-2	Number of minibunches
	8-15	FADC data samples
	16	Serial input 1
	17	Serial input 2
	18	Serial output 1
	19	Serial output 2
	20	Serial output 3
	21	Bypass mode
16 read	0-2	Number of minibunches
	8-15	FADC data slices
	16	Serial input 1
	17	Serial input 2
	18	Serial output 1
	19	Serial output 2
	20	Serial output 3
	21	Bypass mode
17 write	0-7	Length of gate
	8-15	Interbunch spacing
	16-23	Delay between hold to FADs and hold to strips
	24-31	Offset time from WNG1 to first gate
17 read	0-7	Length of gate
	8-15	Interbunch spacing
	16-23	Delay between hold to FADs and hold to strips
	24-31	Offset time from WNG1 to first gate
18 write	0-2	Threshold DAC address [0-7]
	3-10	Threshold DAC data
19 write	0	1 = DSP halted
	1	Start FIFO write sequence
	2	Reset FIFOs

2.2.5 Status registers

There are three 16 bit status registers (SR1, SR2 and SR3).

SR1	
bit(s)	*description*
0-2	Minubunch # B1
3-5	Minibunch # B2
6-8	Minibunch # F1
9-11	Minibunch # F2
12	Threshold at HOLD for B1
13	Threshold at HOLD for B2
14	Threshold at HOLD for F1
15	Threshold at HOLD for F2

SR2	
bit(s)	*description*
0	Trigger type bit 0
1	Trigger type bit 1
2	Bunch # bit 0
3	Bunch # bit 1
4	Account # bit 0
5	Account # bit 1
6	T2YES
15	READOUT

SR3	
bit(s)	*description*
0-6	FADC samples done
7	Last FADC sample available in FIFO
8	FIFO full

All the above signals must be stable during the DSP readout, except READOUT which is latched at rising edge.

2.2.6 Serial switch

A simple serial switch is included on the card. This switch contains two terminal input who can be routed to any of the three terminal outputs. Only one route can be active at any given time. The routing is done by setting the corresponding bits in CSR16.

2.3 Auxilliary backplane

The auxilliary backplane is used to get trigger and event information from the VSAT Local Trigger Supervisor (LTS). The signals and the pin numbers are given in table, se also figure 4. Those signals with only one pin given are single ended ECL, while the others are differential ECL. Note that LURFB/LTS and PATCHCARD/FDDP has different polarity conventions !

FASTBUS auxilliary backplane *italic* marks signal source - = signal do not exist, NC = signal exist but not connected					
signal	*LURFB*	*LTS*	*FBPC*	*FDDP1*	*FDDP2*
READOUT	A4	*A4*	-	B4	B4
BUSY	NC	A6	-	NC	*B6*
T2YES	A14	*A14*	-	B14	B14
Bunch #bit 0	A16	*A16*	-	B16	B16
Bunch # bit 1	A17	*A17*	-	B17	B17
Trigger type bit 0	A18	*A18*	-	B18	B18
Trigger type bit 1	A19	*A19*	-	B19	B19
Threshold F1LO	A20-B20	A20-B20	*B20-A20*	-	-
Threshold F1H1	A21-B21	A21-B21	*B21-A21*	-	-
Threshold F2LO	A22-B22	A22-B22	*B22-A22*	-	-
Threshold F2H1	A23-B23	A23-B23	*B23-A23*	-	-
Threshold B1LO	A24-B24	A24-B24	*B24-A24*	-	-
Threshold B1H1	A25-B25	A25-B25	*B25-A25*	-	-
Threshold B2LO	A26-B26	A26-B26	*B26-A26*	-	-
Threshold B2H1	A27-B27	A27-B27	*B27-A27*	-	-
FIFORESET	A28-B28	*A28-B28*	B28-A28	-	-
WNG1	A29-B29	*A29-B29*	-	-	-
GND	A30-B30	A30-B30	A30-B30	A30-B30	A30-B30
WNG2	A32-B32	*A32-B32*	-	-	-
Testin forward	-	*A34-B34*	B34-A34	-	-
Testin backward	-	*A35-B35*	B35-A35	-	-
Hold to strip f1	*A36-B36*	-	B29-A29	-	-
Hold to strip b1	*A37-B37*	-	B31-A31	-	-
Hold to fad f1	*A38-B38*	-	B32-A32	-	-
Hold to fad b1	*A39-B39*	-	B33-A33	-	-
Hold to strip f2	*A40-B40*	-	B6-A6	-	-
Hold to strip b2	*A41-B41*	-	B7-A7	-	-
Hold to fad f2	*A42-B42*	-	B8-A8	-	-
Hold to fad b2	*A43-B43*	-	B9-A9	-	-
TX0	-	*A36*	A36	-	-
RV0	-	*A37*	A37	-	-
TX1	-	*A38*	A38	-	-
RV1	-	*A39*	A39	-	-
Account # bit 0	A47	*TR1(A47)*	-	TR1	TR1
Account # bit 1	A48	*TR2(A48)*	-	TR2	TR2
FASTBUS slot	18	8	12	20	23
Front panel	17-19	7-9	12-16	20-22	23-25

3 Data readout

At READOUT are the FIFOs read by the DSP, formatted and stored in memory. SE_HI is stored temporarily and saved only if the next trigger type indicates a false bhabha or a DELPHI T2 event. READOUT is generated by LTS when there is a T1YES and a VSAT trigger, or by T2YES if there is a T1YES with no VSAT trigger (a DELPHI event). The logic is shown in figure 6. The data is stored in a DPM to be read from FASTBUS. The DPM is divided into four front end buffers, the current buffer is choosen from the account number stored in SR2.

The last two addresses in the DPM are used as a communication area between the DSP and readout program. This area is used to inform the DSP program when a new run is starting.

Communication	
address	*description*
7FFE	DSP program version number
7FFF	Command number: 0 : no command else : restart

FIFORESET is issued at T1NO or by T2NO when there is a T2NO with no VSAT trigger. This signal is generated by the LTS and distributed though the auxilliary backplane. On LURFB is this signal ORed with a software generated reset.

The BUSY signal is set by READOUT and reset by the DSP program after the readout has been done. This signal is hardwire ORed on the FASTBUS auxilliary backplane with the BUSY signals from the FDDPs.

4 Data format

The first word in the FEB is always the total length of the event.Then follows for each VSAT event, pointers, summary words, the length of this VSAT event, and the FADC data. The summary words contains information added by the DSP program and the status registers. The status registers are described in section 2.2.5 Pointers are added before the data, to handle a flexible length.

Event Data format	
word	*description*
1	Total word count
2	Identifier code = 1800
3	Data source = 69110040
4	Pointers
5	Summary word 1
6	Summary word 2
7	Summary word 3
8	Data word 1
8+X	......
8+N	Data word N

Pointers	
bit(s)	*description*
0-7	summary words
8-15	FADC data
16-23	0
24-31	0

Summary word 1	
bit(s)	*description*
0-15	Status register 1 (SR1)
16-31	Status register 2 (SR2)

Summary word 2	
bit(s)	*description*
0-15	Status register 3 (SR3)
16-31	VSAT event number since last T2YES

Summary word 3	
bit(s)	*description*
0-15	Total T2YES counter
16-31	Total trigger counter

FADC data word	
bit(s)	*description*
0-7	FADC data B1
8-15	FADC data B2
16-23	FADC data F1
24-31	FADC data F2

T2 Event Data format	
blocklet	*description*
FDDP1 FDDP2 LURFB	VSAT event 1
...	...
FDDP1 FDDP2 LURFB	VSAT event N
FDDP1 FDDP2 LURFB	DELPHI event
LTS	trigger blocklet

Table 1: T2 Event data format

At T2YES is the data transferred to the FIP, where it is encapsulated in the normal DELPHI standard format. The FIP T2 processing formats the data into separate VSAT events, with the DELPHI event as the last event. The LTS trigger blocklet contains trigger information for all the VSAT events in this T2 buffer.

In the T3 formatting are all T2YES events which are not accepted by DELPHI removed. The trigger blocklet for the T2 buffers are collected into one blocklet, which then contains information for all VSAT events in the T3 buffer. The T2 and T3 buffers are shown in tables 1 and 2.

5 Modifications to 1989-1994 setup

A few modifications were needed on the existing setup, redoing of some cabling and crate backplanes.

5.1 FASTBUS auxilliary backplane

The signals that need to be rewired are:

T3 Event Data format	
blocklet	*description*
FDDP1 FDDP2 LURFB	VSAT event 1 - T2 event 1
...	...
FDDP1 FDDP2 LURFB	VSAT event N1 - T2 event 1
...	...
...	...
FDDP1 FDDP2 LURFB	VSAT event 1 - T2 event M
...	...
FDDP1 FDDP2 LURFB	VSAT event NM - T2 event M
FDDP1 FDDP2 LURFB	DELPHI event - T2 event M
LTS	trigger blocklet T2 events 1-M

Table 2: T3 Event data format

- WNG1
- WNG2
- Thresholds

WNG1 and WNG2 is removed from the patchcard and wired to the peakfinder card. From the peakfinder card are four HOLD signals to the FADs (one for each mdoule), and four signals to the strips (one for each module).

5.2 FASTBUS patch card

The patchcard is modified to handle the extra HOLD signals from the peakfinder card.

5.3 Cabling

The cables form the FASTBUS card to the digital patch card in the tunnel crates carries the new extra HOLD signals.

5.4 Digital patch card

The patchcard is modified to handle the new HOLD signals from the peakfinder card, and distributes on the backplane.

5.5 Tunnel crate backplane

The backplane is redone to distribute the HOLD signals. The new HOLDs are not common to the modules, i.e. the backplane is modified for this

5.6 FAD MUX card

A front panel connector for the sum is placed on the card. A cable is pulled from the MUX cards to the peakfinder FASTBUS card.

6 Offline and online software

6.1 Introduction

The main purpose of the software modifications is to

- Accept and decode the new rawdata blocklet.
- Work as a peakfinder and tag the minibunches (until the hardware peakfinder works).
- Find out the "true trigger type".
- Correct for lost events, due to multiple events within the same train.

All these modifications are done in the spirit of backward compatibility.

6.2 Online software

The online software consists of two parts:

- The VSAT local monitor
- The VSAT specific central monitoring code

The local monitor runs on the VSAT node of the online cluster, WSDEVS. It performs a detailed monitoring of the VSAT detector performance, filling histograms of interest, producing error messages etc. and is also responsible for producing pedestal values and calibration constants, used in the offline analysis, to the database. The central monitoring code contains a special routine for the VSAT. This routine is used to fill the VSAT histograms used in the central DELPHI quality checking.

6.2.1 The VSAT local monitor

The VSAT local monitor is modified in order to be able to read the new blocklet and unpack the information about minibunch number and flash ADC samples for each event. The modifications are done in such away that if the monitor finds the new blocklet it reads the information, if it doesn't it works as before with the information from the FDDPs and LTS only.

The minbunch information is used to create online histograms for the flash ADC samples in order to check continuosly the performance and behaviour of the peakfinder. If desired an AHA-routine can be used to automatically monitor any change in the behaviour of the peakfinder, giving an immediate alarm if something goes wrong. The most sensitive histograms for changes in peakfinder performance are the bhabha energy histograms which immediately shows if the hold signal is generated at the wrong time for a small sample of the events. These are also monitored by the central monitor and the shape is checked every five minutes automatically.

6.2.2 The central monitoring code

The central monitoring code is modified in such a way that it just ignores the new LURFB-blocklet. This is done because for the moment we don't want the modifications to disturb the central DELPHI quality checking and all debugging should be performed by the local monitor. For the future we can decide what, if any, histograms we want to include for the minibunches in the central quality checking.

6.3 Offline software

The offline software consists of several parts:

- LUMANA, the VSAT DELANA routines.
- COMPACT, which performs a compacting of the DELANA output files.
- VSDST, where the main analysis is performed.

All these programs are modified in order to cope with the new raw data structure.

6.3.1 LUMANA

LUMANA is modified in a way much similar to the local monitor code, that is it finds the new blocklet and unpacks the information on minibunch number. Then it puts this information in a free place in an already existing dataword written to the output file. If no LURFB blocklet is found the minibunch number 0 is assigned to all events and LUMANA works in the old way with only the FDDP and LTS information. Since the hardware peakfinder is not active yet, a routine performing a peak finder function has to be added. This will work by looking for when the flash ADC samples go above a threshold, since the pulses have to be modified hardware wise before we can see any real peaks, and depending on where this happens assign a minibunch number to the event. The reason why we want to do this at this early stage and not later in VSDST is that we want to get rid of the flash ADC samples which occupy a lot of space for each event. If we want to look at the flash ADC samples offline e.g. for peakfinder analysis, we can run a standalone version of LUMANA which creates an ntuple containing the FADC samples for each event. This standalone version also contains detailed checking of the contents of the LURFB blocklet.

6.3.2 COMPACT

Compact is left without any modifications, since the dataword where we put the minbunch information was already existing before.

6.3.3 VSDST

This is where the major modifications has to be done and much of them will depend on how we want to perform the analysis of the minibunch data. At first stage we have to add a routine for comparing the minibunch numbers for each event and diagonal modules. This because we need to find out the trigger type in the software, since the hardware trigger looks at the complete bunchtrain as a whole when it assigns the triggertype. For example a single electron in minibunch 1 in module B2 and a single electron in minibunch 2 in F1, will be triggered as a bhabha in diagonal 1 and the true identity of the event can only be discovered by looking at the minibunch number assigned by the peakfinder. We also have to add a routine which corrects for the number of lost events due to that the peakfinder triggers on the first peak in time, which means that if we have two events within the same bunchcrossing the last event will not be readout. This would theoretically make us see more events, proportionally, in the first minibunch and gradually less in the later ones. At second stage we have to decide if we want to perform the luminosity analysis separately for the different minibunches. This could be motivated if the conditions regarding energy and beamparameters are different for the different minibunches. This is the most likely scenario, since we expect to have different conditions for the different wagons in the bunchtrains.

7 Performance

The correspondance between the sum of FAD trigger planes from the FDDPs, and the pulseheight form the LURFB FADC at the time of the HOLD, is shown in figure 1.

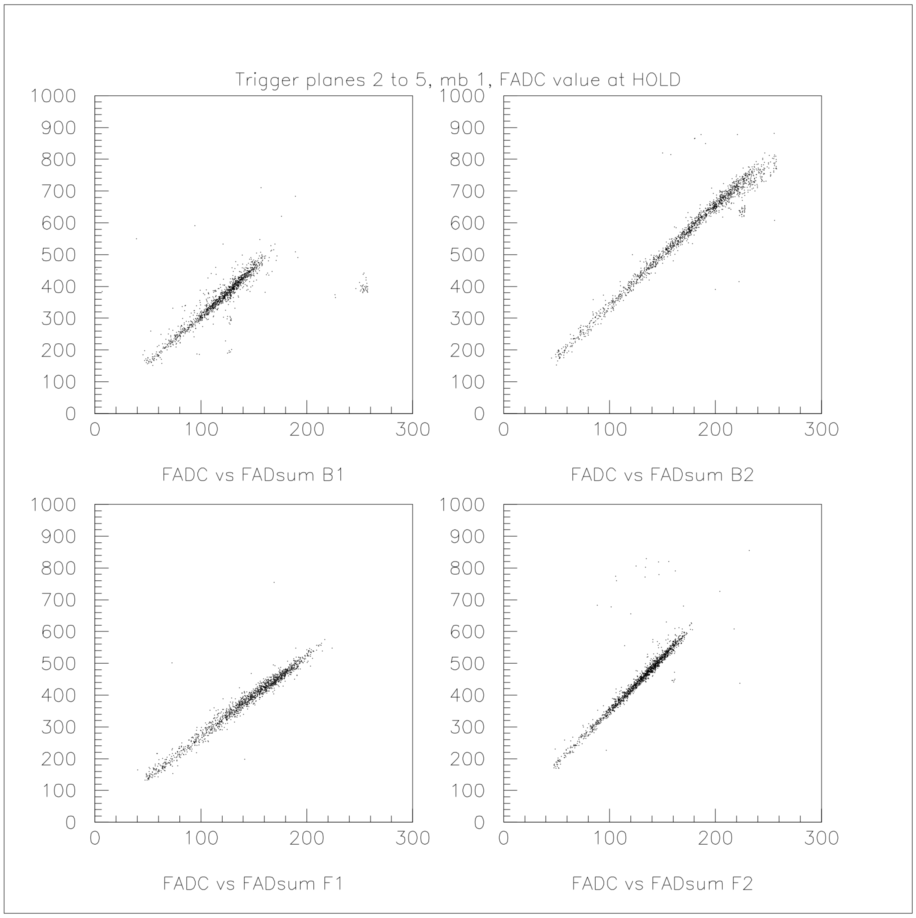


Figure 1: FADC value versus trigger FAD sum

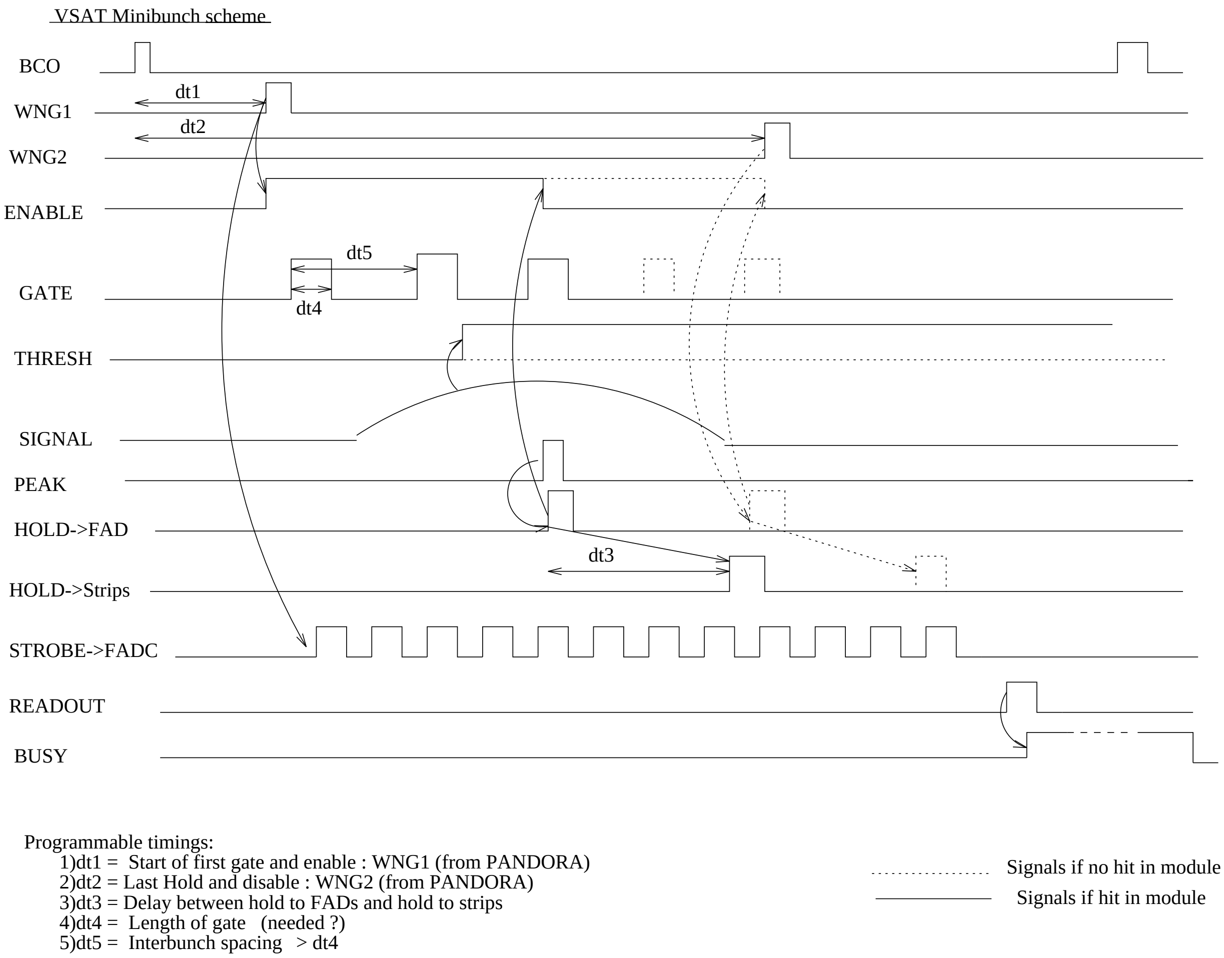


Figure 2: Signal timing

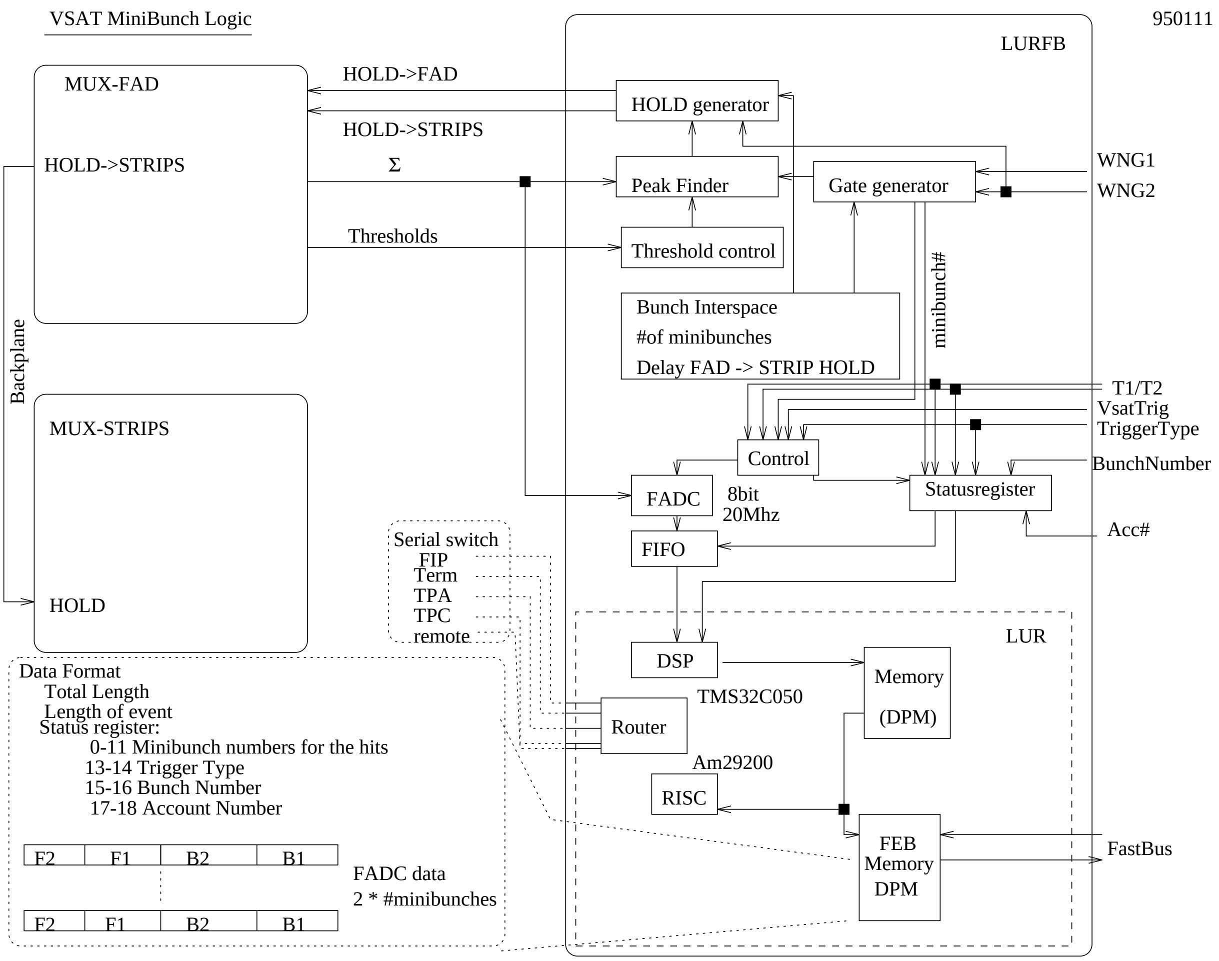
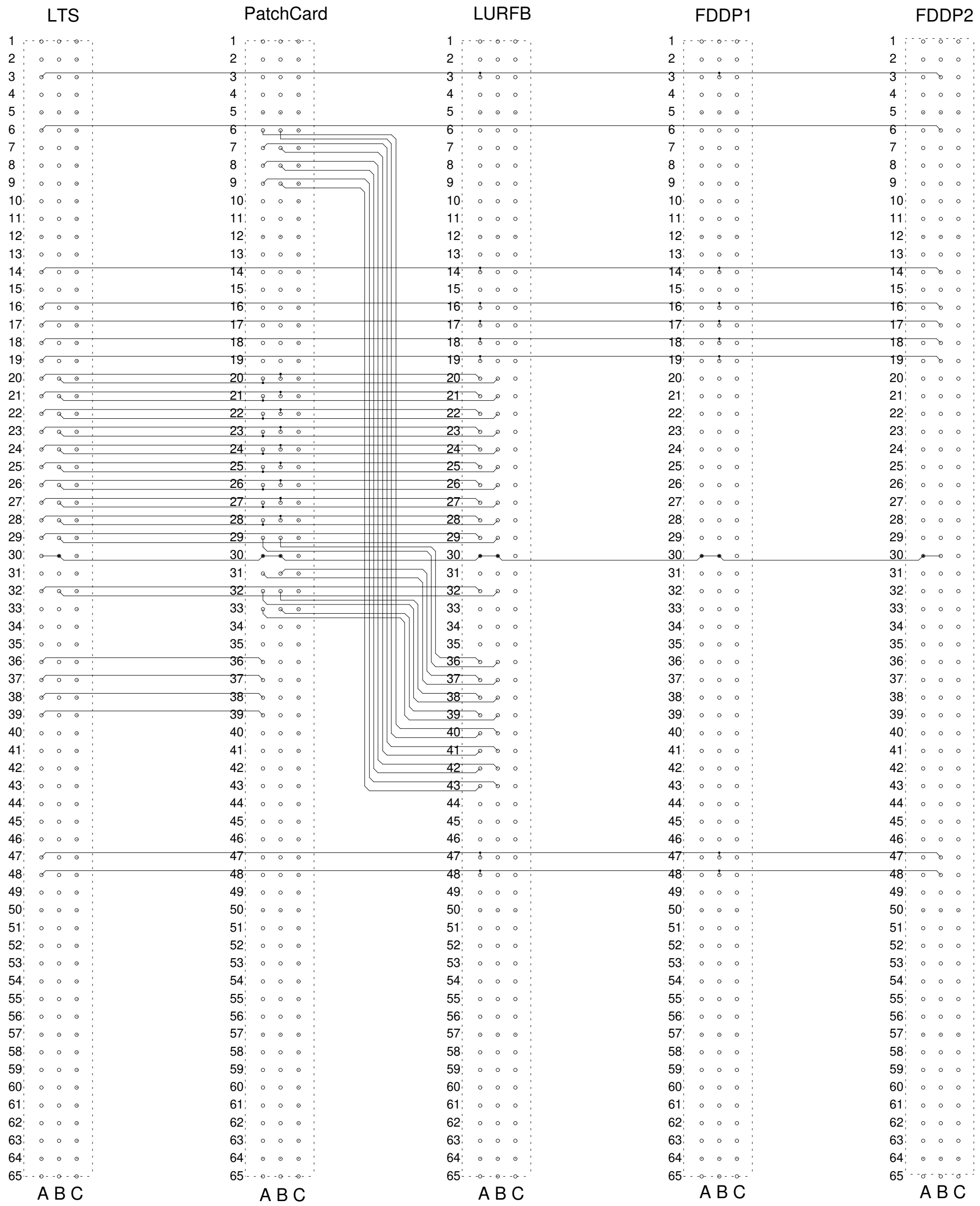


Figure 3: FASTBUS card logic



View of VSAT FASTBUS auxiliary backplane

Figure 4: FASTBUS crate auxilliary backplane

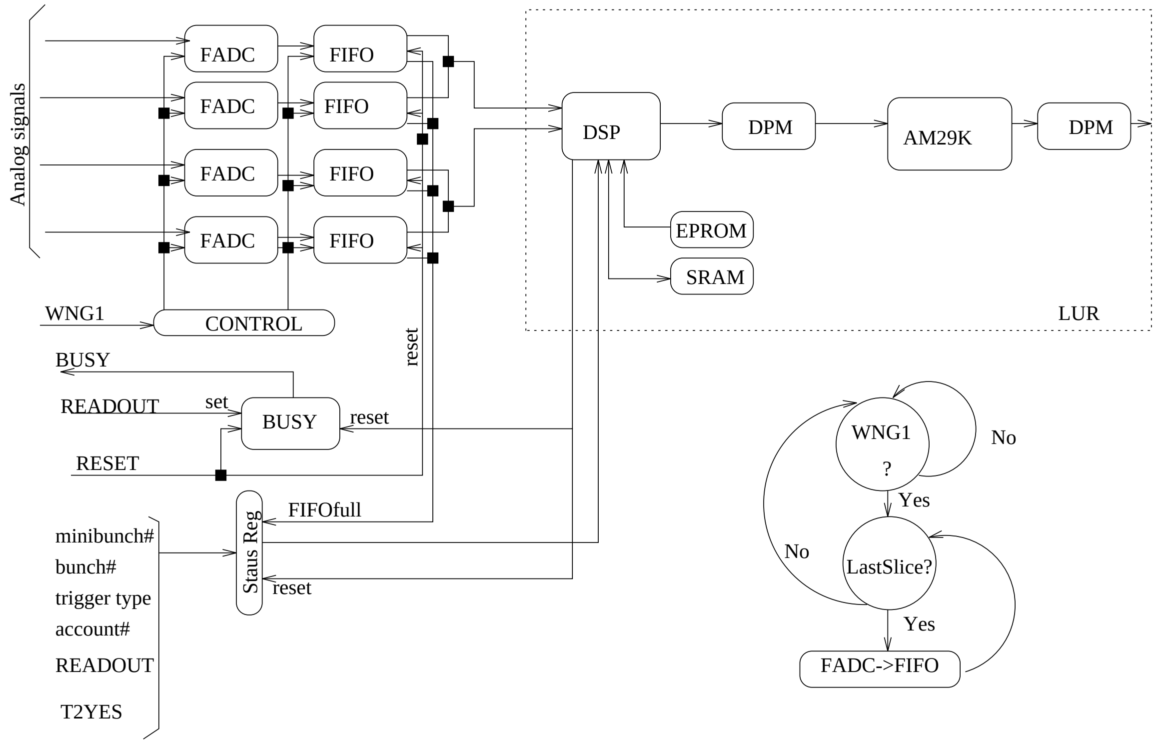


Figure 5: FADC hardware logic

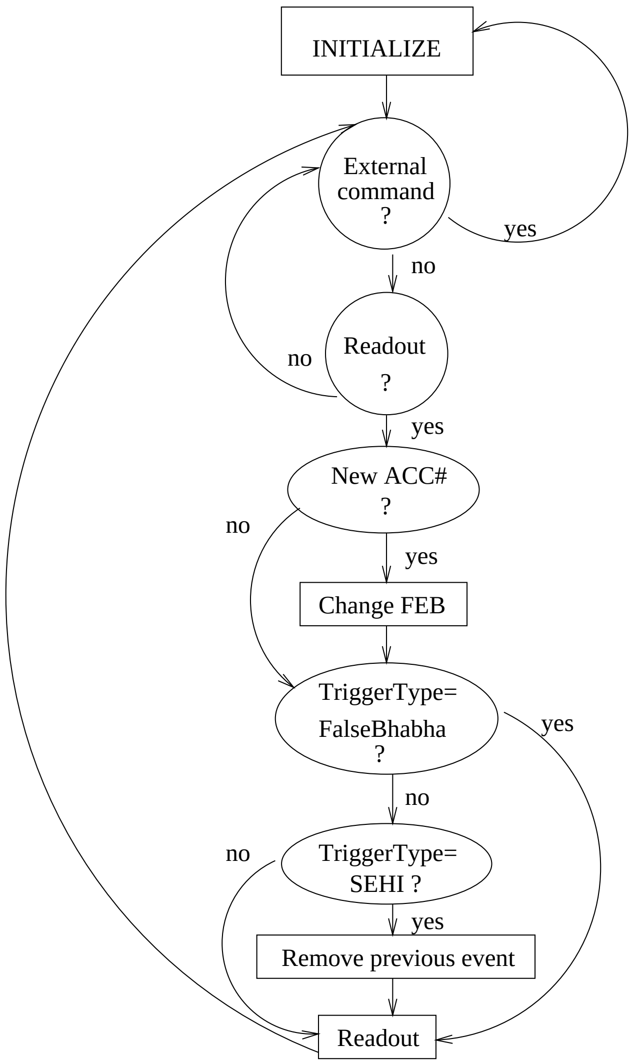


Figure 6: FADC readout logic

Data Format

Total length of VSAT event1

Identifier 1 = EC001800 — EC = Error Code

Identifier 2 = 69110020 — 6911 = FastbusId of module

pointers + summary words

FADC data slice 1

⋮

FADC data slice N

Total length of VSAT event1

•
•
•
•
•

Total length of VSAT eventM

Identifier 1 = EC001800

Identifier 2 = 69110020

pointers + summary words

FADC data slice 1

⋮

FADC data slice N

Total length of VSAT eventM

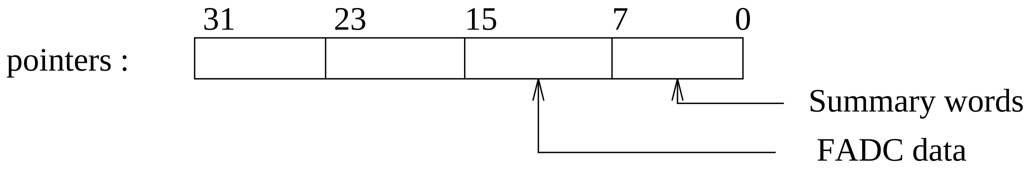


Figure 7: Data format in DELPHI data stream

T2BUFFERs	
VSAT event 1	Header
	FDDP1
	FDDP2
	LUFAC
	LTS
	⋮
VSAT event N	Header
	FDDP1
	FDDP2
	LUFAC
	LTS
DELPHI T2 event	Header
	FDDP1
	FDDP2
	LUFAC
	LTS

T3BUFFER	
VSAT event 1	Header
	FDDP1
	FDDP2
	LUFAC
VSAT event 2	FDDP1
	FDDP2
	LUFAC
	⋮
VSAT event N	FDDP1
	FDDP2
	LUFAC
DELPHI T3 event	FDDP1
	FDDP2
	LUFAC
	LTS

Figure 8: VSAT data format in data stream

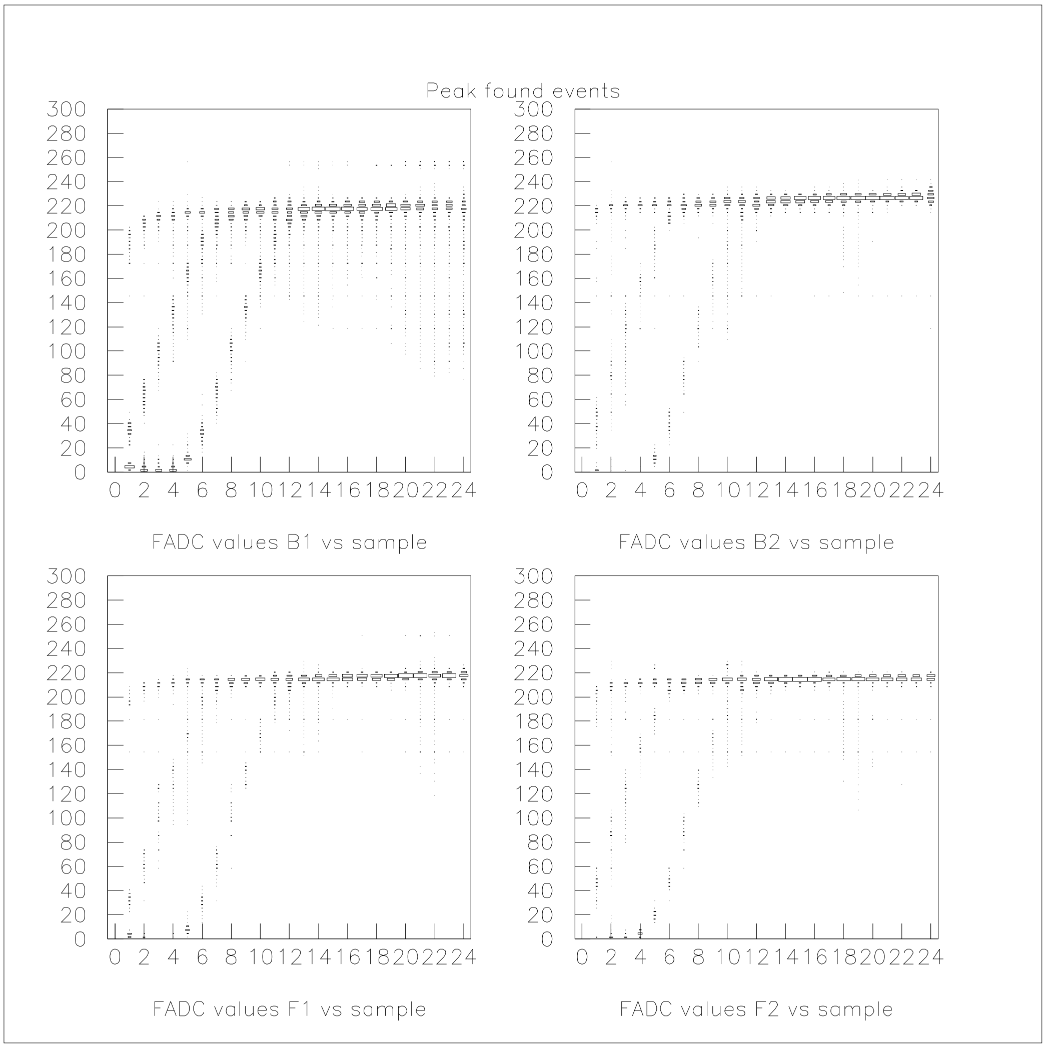


Figure 9: FADC data when a peak is found

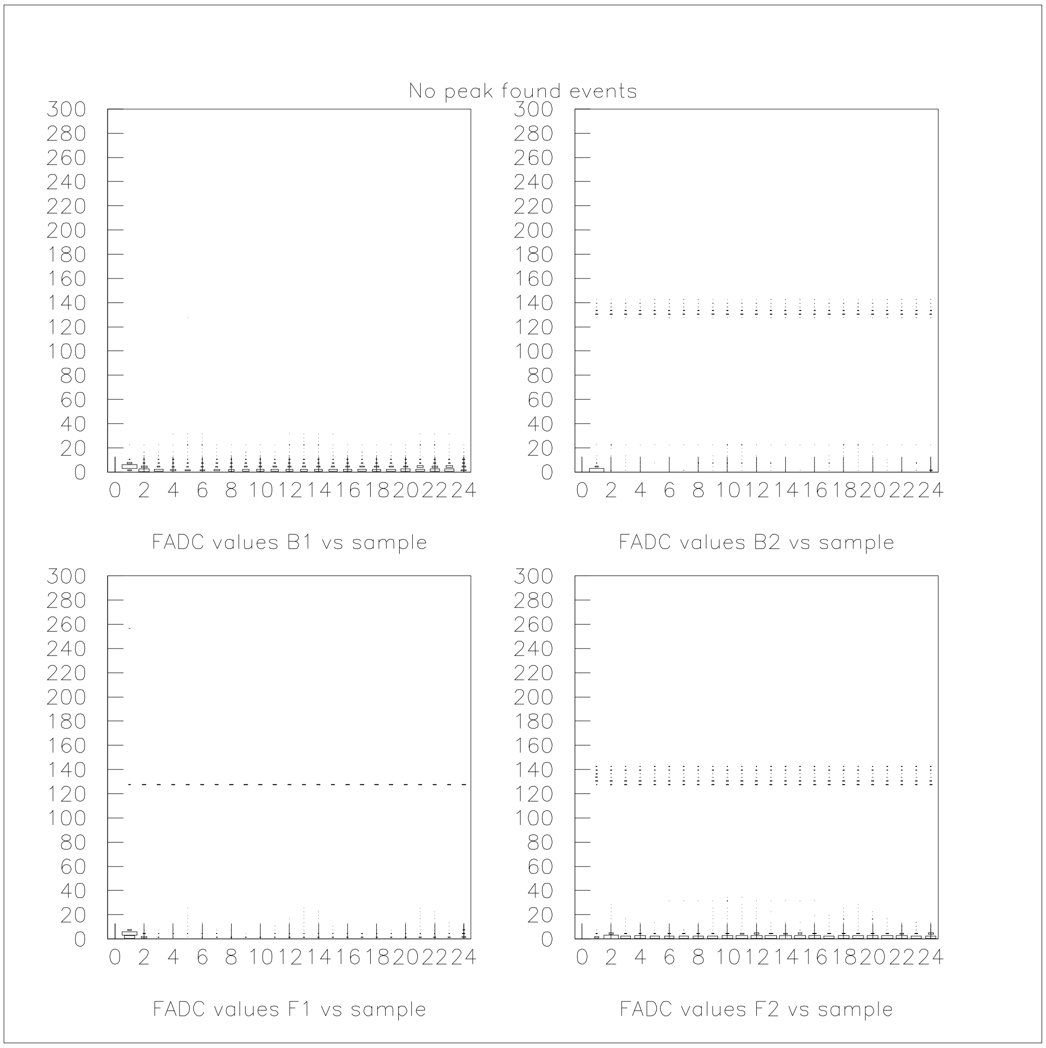


Figure 10: FADC data when no peak is found