weeroc

High-end Microelectronics Design

Maud: Datasheet system

Prepared by	Florent PEREZ
Controlled by	Salleh AHMAD
Approved by	Salleh AHMAD

Version document record

Issue	Date	Total pages	Modified pages	Notes

Table of content

Table of figure

1 References

1.1 Applicable documents

The following documents are applicable to this report.

Ref	Document name	Reference
AD-1		
AD-2		
AD-3		

Table 1 - Applicable document

1.2 Reference documents

The following documents are referenced to this report.

Ref	Document name	Reference
RD-1	CITIROC1A - Datasheet	V2.4
RD-2	CITIROC - User Guide	
RD-3	A7585 User's Manual	A7585_rev0
RD-4	Cyclone V Device Datasheet	CV-51002
RD-5	LTC2325-14 Datasheet	232514
RD-6	Maud_FPGA.xls	

Table 2 - Reference document

2 Introduction

This datasheet describes the electrical characteristics, configuration specifications and I/O specifications for a 64 channels Read-Out System with two Citiroc1a embedded.
In total, there are three boards designed for this project:

- FE BOARD – This is the front-end board housing the 64-channel detector, readout ASICs and ADC
- BE BOARD – The backend board contains the FPGA for controlling the ASICs and also acts as the connection point between the external power supply, PC and HV voltage module.
- HV BOARD – This board is used to house the HV module supplying the biasing voltage of the detector.

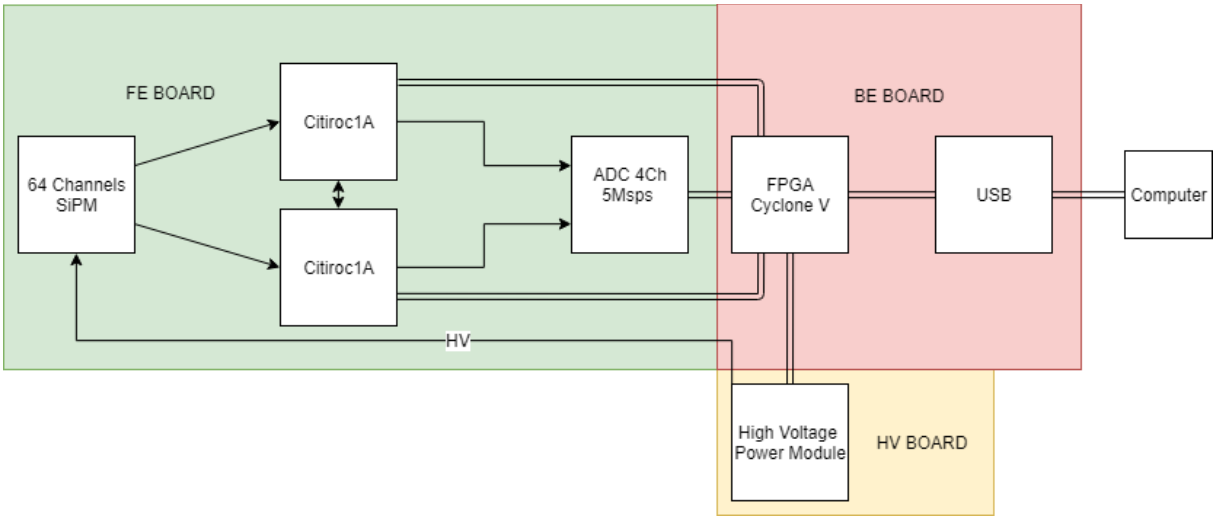


Figure 1 - Synoptic Diagram of the system

This document will include the programming phase for both Citiroc1a, the different ADC code that will need to be decoded, the protocol of communication between the software and the firmware and the different register the firmware will give you in term of data or configuration.

This Datasheet is completed by the *Citiroc1A - Datasheet* which will help you to configure and program your ASIC in order to perform measurements.

3 CITIROC1A

3.1 Programmable parameters

3.1.1 Slow control

The slow control and probe register in this system are put in a serial way. Instead of having 1144 bits of slow control (*Citiroc1A – Datasheet, 8. Programmable parameters, internal probing and power pulsing*), there will be 2288 bits of programming because it is a duplicate of 2 citiroc1a:

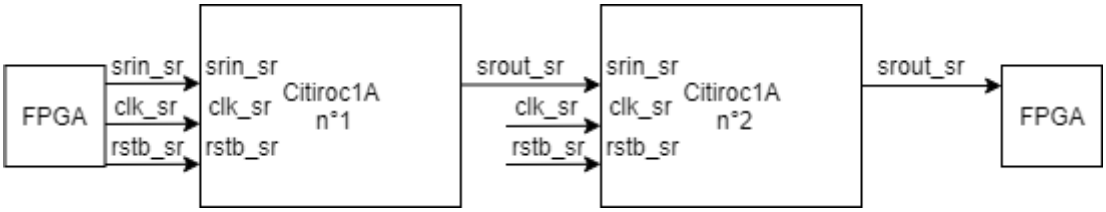


Figure 2 - Slow control in serial

The slow control allows programming Citiroc1A with all parameters as :

- The different DAC Value, the threshold value
- Switching on or off a different part of the ASIC
- Activating a few parameters as latching trigger, etc..

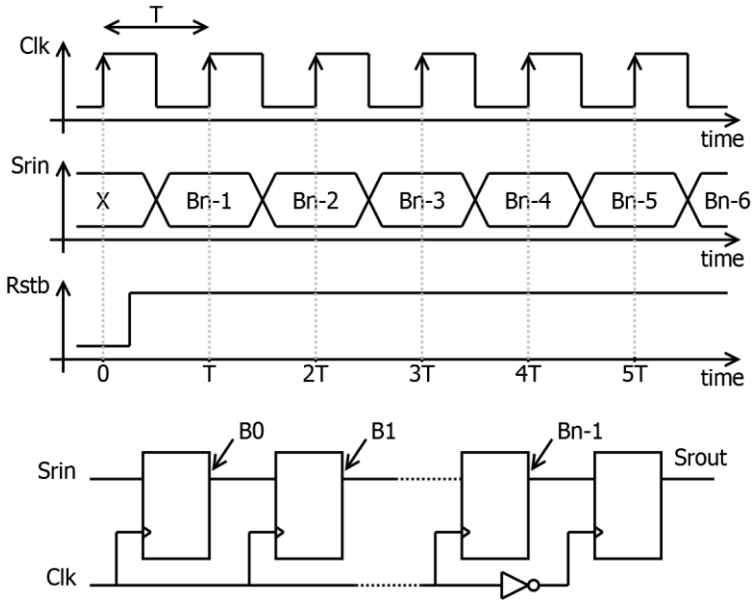


Figure 3 - Slow Control Chronogram and explanation (Citiroc1A - Datasheet)

The slow control in Citiroc1A is a shift register composed of 1144 flip-flops, in this system the slow control is composed of 2288 flip-flops (first bit is B_0 = DAC_t<0>_1 and last B_{2287} = EN_32triggers_2).
The first bit of the slow control register is the last bit to enter in the ASIC:

The table for the slow control register can be found in *Citiroc1A- Datasheet Section 8.*

3.1.2 Read register

See CITIROC1A – Datasheet Section 7 p.22

Unlike for the slow control, the read register is parallel between both Citiroc1A.
This way when there is an acquisition sequence, both ASIC are working at the same time, delivering the same channel to the analog output which will be converted by the ADC.

For example, the ADC will convert Ch2_LG, Ch2_HG, Ch34_LG, CH34_HG at the same time. And it will take the same amount of time than converting one Citiroc1A analog output value.

3.2 I/Os connection

3.1.3 SiPM connection

Each channel of a SiPM is connected to an Input of CITIROC1A like it is recommended in the Datasheet (10.1 SiPM connection) expect that the 100nF capacitors are replaced by a 220nF one.
This way the input DAC can adjust the DC value on the input point and it can correct the non- uniformity in the breakdown voltage of the different SiPM channel connected to Citiroc1A.

The input DAC have a 4.5V dynamic range, allowing the user to adjust the virtual ground of each SiPM channel that is connected to the same high voltage supply.

3.1.4 Trigger connection

In this system, every trigger is connected to the FPGA to be able to perform an S-curve-test which will help in order to calibrate the input DAC and correct the non-uniformity from the SiPM.
Both NOR32_OC and NOR32T_OC from the 2 Citiroc1A are connected separately to the FPGA. This way there is the choice to connect them into the FPGA if there is a need for a common trigger, for an acquisition system for example.

3.1.5 Control Signals

Every signal that is used for control is the same for the 2 Citiroc1A :

- Signal for PeakSensing (PS_modeb_xt, PS_global)
- Val_evt & Raz_chn
- Signal for slow control and read register

When a validation window has been opened it will be opened for both Citiroc1A. If there is a need of disabling one Citiroc, it is possible to deactivate all the stage from one ASIC in the slow control.

Same for the analog reset, both ASIC will be reset when the analog reset (raz_chn) is equal to '1'.

4 System Functionality

4.1 I/O Systems and Mechanical part

There is 2 external connection on the BE Board :

- One for the external supply, communication to PC, debug and PT100 connection
- One for high voltage power supply: A7585

4.1.1 External connection

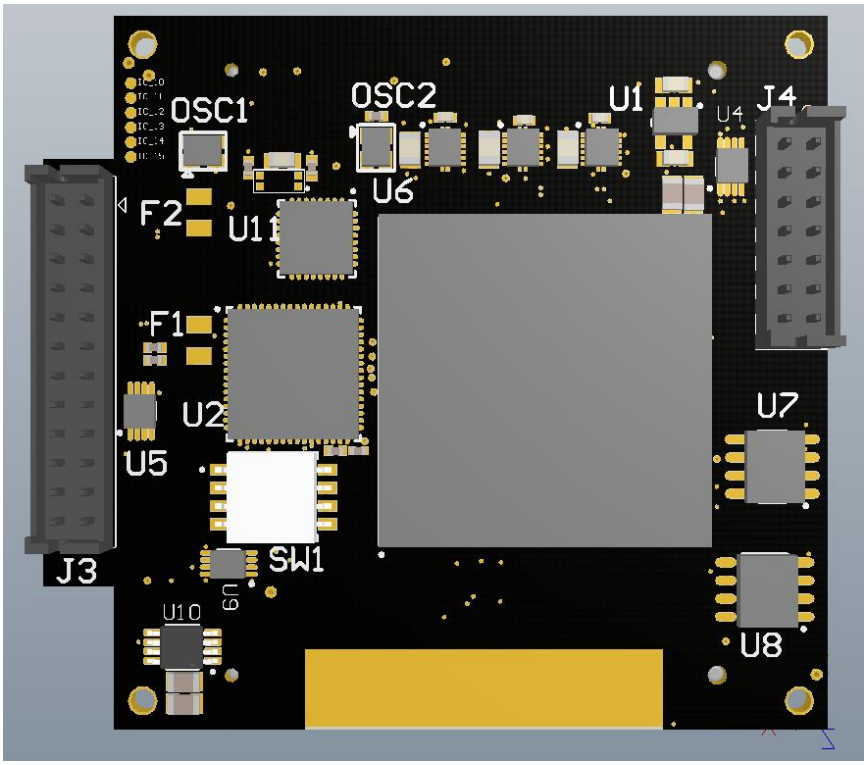


Figure 4 - 3D Bottom View BE Board

Header J3 (see above) is used for the USB connection, debug, supply and PT100.

Pin n°	Name	Description
1	USB_RS+	DATA(p) for FT232R
2	USB_RS-	DATA(n) for FT232R
3	GND	Ground
4	VBUS_RS	5V for FT232R
5	USB-	DATA(n) for FT2232H
6	USB+	DATA(p) for FT2232H
7	GND	Ground
8	VBUS	5V for FT2232H
9	RX	UART RX
10	TX	UART TX
11	GND	Ground
12	GND	Ground
13	IO_FPGA1	I/O connected to FPGA
14	IO_FPGA2	I/O connected to FPGA
15	IO_FPGA3	I/O connected to FPGA
16	IO_FPGA4	I/O connected to FPGA
17	IO_FPGA5	I/O connected to FPGA
18	IO_FPGA6	I/O connected to FPGA
19	IO_FPGA7	I/O connected to FPGA
20	IO_FPGA8	I/O connected to FPGA
21	5VEXT	External Supply (5V 1A suggested)
22	GND	Ground
23	PT+	PT100 connection wire (+)
24	PT-	PT100connection wire (-)

Table 1 – External connection pinout list Header J3

In order to communicate with a PC, USB+ and USB- shall be used all the time. In future USB_RS+/- could be used to create another communication system using UART with the FTDI: FT232R.

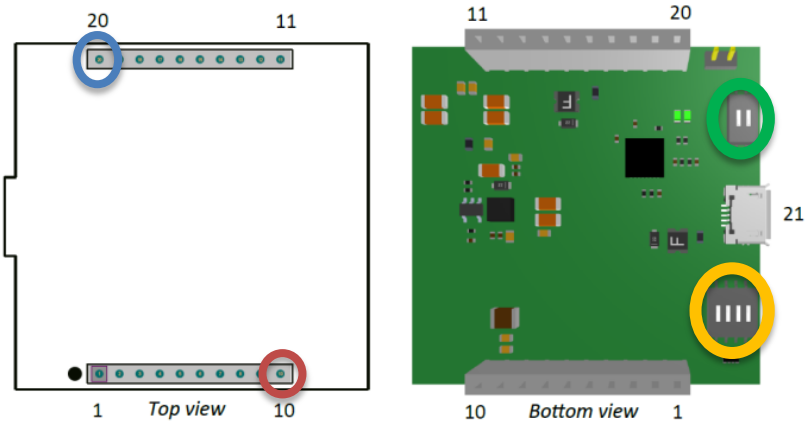
If there is no use of TX/RX UART, 6Volts could be used as an External Supply (Pin.21). Because of the fact that every power supply for the FPGA, ASICs and HV Power supply use a DC/DC converter in order to regulate the supply.

In the end, IO_FPGA will be used as a shield for the connector as they could be connected to the ground.

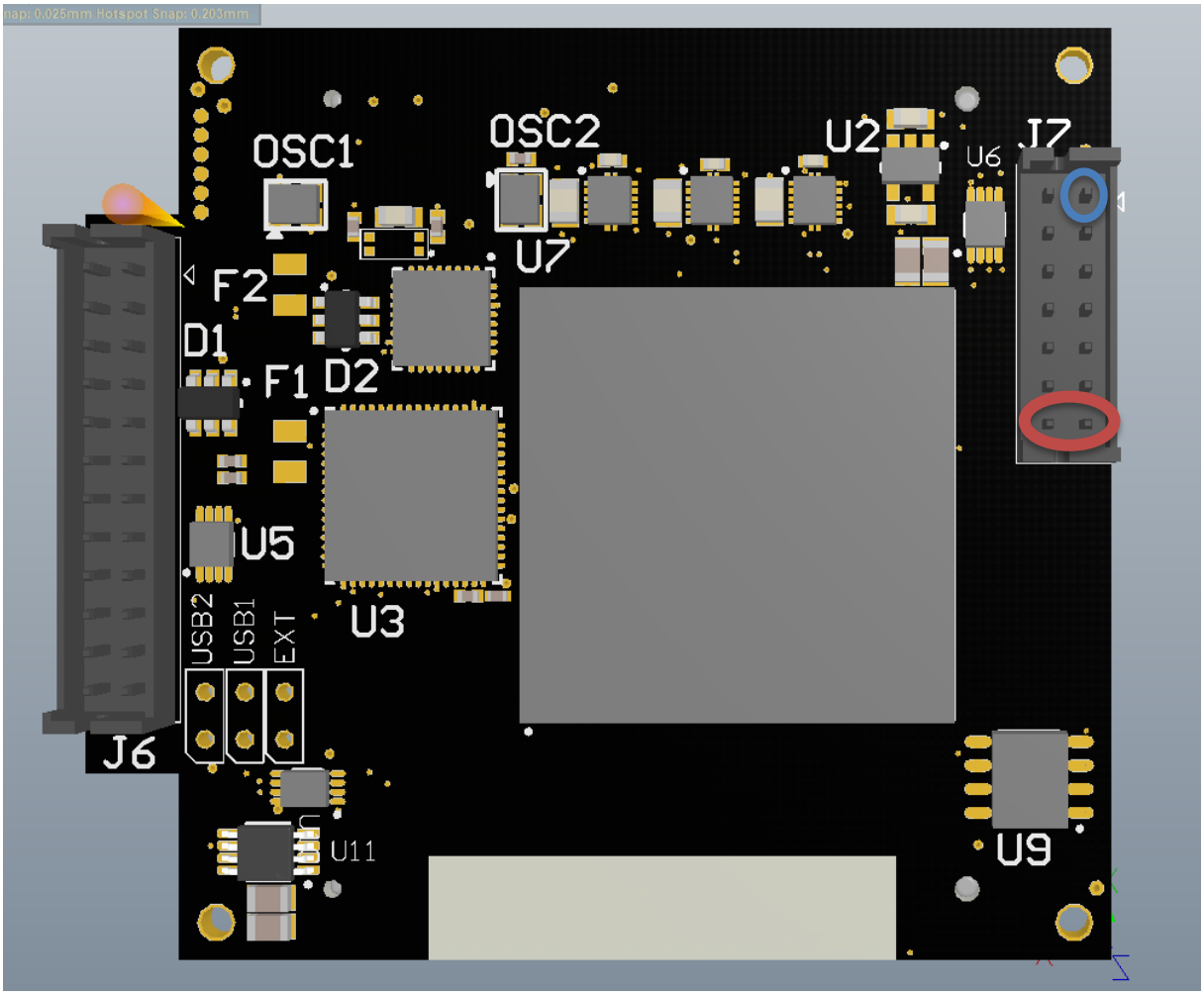
The connector that will be used to do the connection with the header on the board is :
DF11-24S-2C
In order to be well connected, the wire should be crimped with DF11-2428-SCF and the wire should be in standard AWG#24 to AWG#28.

4.1.2 High Voltage Connection

On the 2 PIN Jumper, both are "ON".
For the 4 Pin Jumper, we put them all to "OFF" except for the one towards the center 'Cfg0" which need to be put to "On".



The connection between both boards is pretty easy to realize:
On the TOP View of the PowerModule Pin 10 (HV) and Pin 20 (GND) will be wired to the Back End Board.
On the BackEnd Board the connector J7 will be used to wire the High Voltage:



High voltage got to be wired to the 2 pins on the Bottom of the Connector.
The ground needs to be connected to the pin n°1 Top Right of the Connector.
The connector that will be used to do the connection with the header on the board is :
DF11-14DS-2C
And it will use the same wire as we saw before as well as the same crimping stuff: DF11-2428-SCF.

4.1.3 Connecting BE to FE Board

Using a couple of male/female connector SAMTEC(TEM-140-02-03.0-H-D-A/ SEM-140-02-03.0-H-D-A) there are 80 contacts per connector, allowing us to connect a lot of signal between those two boards. This way the FPGA is connected to both Citiroc1A and to the ADC LTC2325-14. And the connector also carries some supply for digital part, allowing the design to be less noisy as there is DC/DC converter on the Front End board.

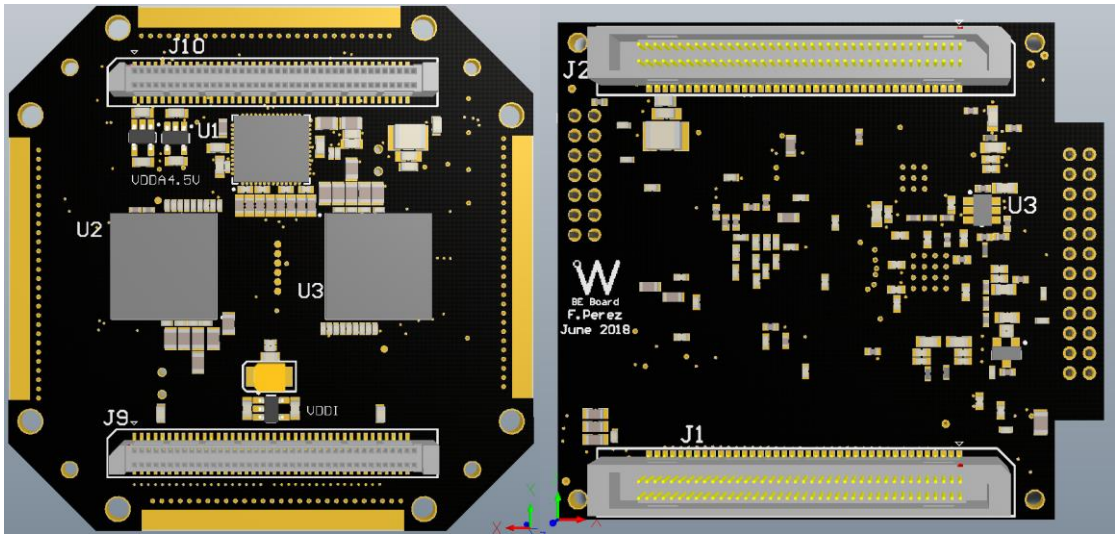


Figure 5 - 3D Bottom View FE Board & 3D Top View BE Board

In order to connect the good signals connections, **J1**(BE) should be connected to **J9**(FE) and **J2**(BE) should be connected to **J10**(FE).
It is recommended to do not plug and unplug the 2 cards too many times, possible loss of efficiency.
There is also a possibility to block the two boards using 4 M2 screws in the holes (2mm diameter) which are on each corner of each board.

4.1.4 Mechanical possibilities

In a mechanical point of view, you can be able to screw the BE Board to your mechanical support with 8 M3 screws, using the 2 bigger holes (2.9 mm diameter) in every corner of the boards.

Same for HV board, in each corner you will have 1 hole(2.9 mm diameter) which will allow you to fix the board to your mechanical structure.

4.2 FPGA

The FPGA which have been chosen is the 5CEBA9 :

	Product Line	5CEA9
Resources	LEs (K)	301
	ALMs	113,560
	Registers	454,240
	M10K memory blocks	1,220
	M10K memory (Kb)	12,200
	MLAB memory (Kb)	1,717
	Variable-precision DSP blocks	342
	18 x 18 multipliers	684

Figure 6 - FPGA characteristics

The Firmware which is developed by Weeroc will use up to 50% of the memory part for Data Acquisition and 1-2% for ALM (Adaptive Logic Module). In fact, the data rate is based on the available memory.
Letting the space to add any VHDL program in the FPGA, even for the heaviest one.

It will combine a few principal functions which will be useful for the measurement part :

- USB interface for communication between Software and Firmware
- Slow Control
- I2C for HV Module
- Data Acquisition system (ADC and both CITIROC1A are configured)
- ADC for external PT100
- Temperature Sensor TMP275

4.2.1 Communication between Software and Firmware

In order to communicate with FPGA, the software uses a protocol which has been developed by LAL.
It is using a FIFO Asynchronous mode with the FTDI2232H device.
This protocol uses 2 important .dll which can be downloaded on:
LALUSB Drivers: for LALUSB2.0.dll
D2XX Drivers for FTD2XX_NET.dll
They will be needed for every computer that will exchange data with the FPGA in order to get or send data.

Figure 7 - 7Bits Address corresponding to 8Bits Data

In order to communicate with the FPGA, there are multiple words of 8 bits which are corresponding to signals on the FPGA. Each Word corresponds to an address of 7 Bits.

For example, when the software is reading the word'100', the firmware is responding with his version number which has been written in the Firmware. This is the way we know we are well connected to the board when we connect it to our C# :

```
94          while (rdSubAdd100 == "00000000" && testCon < 10)
95          {
96              testCon++;
97              // Sub address 100 contain the firware version. If rdSubAdd100 == 0 the board failed to connect
98              rdSubAdd100 = Firmware.readWord(100, usbDevId);
99              // Wait 20 ms
100             Thread.Sleep(20);
101         }
102
103         // If rdSubAdd100 == 0, the board failed to connect 10 times in a row
104         if (rdSubAdd100 == "00000000")
105         {
106             MessageBox.Show("Looks like there is an issue with the connection. Please verify the board is well plugged and powered and click again.");
```
Figure 8 - Read word 100 in Software

Those words are like register with :
Sometimes the integrality of a word is used as the word at the address 21 contains data from an acquisition (High_Gain_Ct2(7 downto0)). And sometimes it's one word for 8 bit with a different function, for example
The Bit 1 from the word 0 is a software analog reset for the ASIC.

Bit 'Y' of the word 'X' is written WordX(Y) in order to be more fluent. Word 'X' corresponds to the Data at the Address 'X'.

With both DLL the software is able to communicate with the FPGA through those words. There are many programmable words. Which can be found on the Annexe "Maud_fpga.xls".

There are two functions for reading a word and two functions for writing on a word. In fact, you can read a word multiple amounts of times: It will be useful when the Software need to extract a lot of data from a FIFO (multiple bytes have to be read in order to get all the data) which is connected to the Data Acquisition.

```
public static void sendWord(byte subAd, string word, int usbDevId)
{
    Int32 wrByte;
    byte[] tempTx = new byte[1];
    tempTx[0] = Convert.ToByte(Convert.ToUInt32(word, 2));
    wrByte = USB.UsbWrt(usbDevId, subAd, tempTx, 1);
}

public static void sendWord(byte subAd, byte[] stream, int nbSend, int usbDevId)
{
    USB.UsbWrt(usbDevId, subAd, stream, nbSend);
}
```

Figure 9 – Functions for sending a word under C#

```
public static string readWord(byte subAd, int usbDevId)
{
    byte[] tempRx = new byte[1];
    byte rdAdd = 0;
    unsafe
    {
        fixed (byte* arrayRd = tempRx)
        {
            Int32 rdByte = USB.UsbRd(usbDevId, subAd, arrayRd, 1);
        }
        rdAdd = tempRx[0];
    }

    return IntToBin(Convert.ToInt32(rdAdd), 8);
}

public static byte[] readWord(byte subAd, int nbRead, int usbDevId)
{
    byte[] rdAdd = new byte[nbRead];
    unsafe
    {
        fixed (byte* arrayRd = rdAdd)
        {
            int rdByte = USB.UsbRd(usbDevId, subAd, arrayRd, nbRead);
        }
    }
    return rdAdd;
}
```

Figure 10 - Functions for reading the word under C#

It is working like a register table where Subaddress (7bit Adress) correspond to value in the Firmware/Software (see Annex 1).

The USB module has been electronically designed with the possibility to switch to FIFO Synchronous to upgrade the data flow. It's possible to develop a new protocol in order to improve data speed.

To perform this communication it is an **obligation** to have USB-, USB+, GND, 5VBUS (Table 1) wired to the board because of even if there is 5VEXT there will be no supply for the FTDI2232H and it will not work at all.

4.2.2 Slow Control

In order to send the Slow control configuration to the ASIC, the user will need to go through the FPGA using a FIFO which will shift all the bits to the Internal Register of the ASIC.
To do this important operation, the user will have to use 3 words: Word 0, Word 1 and Word 10.

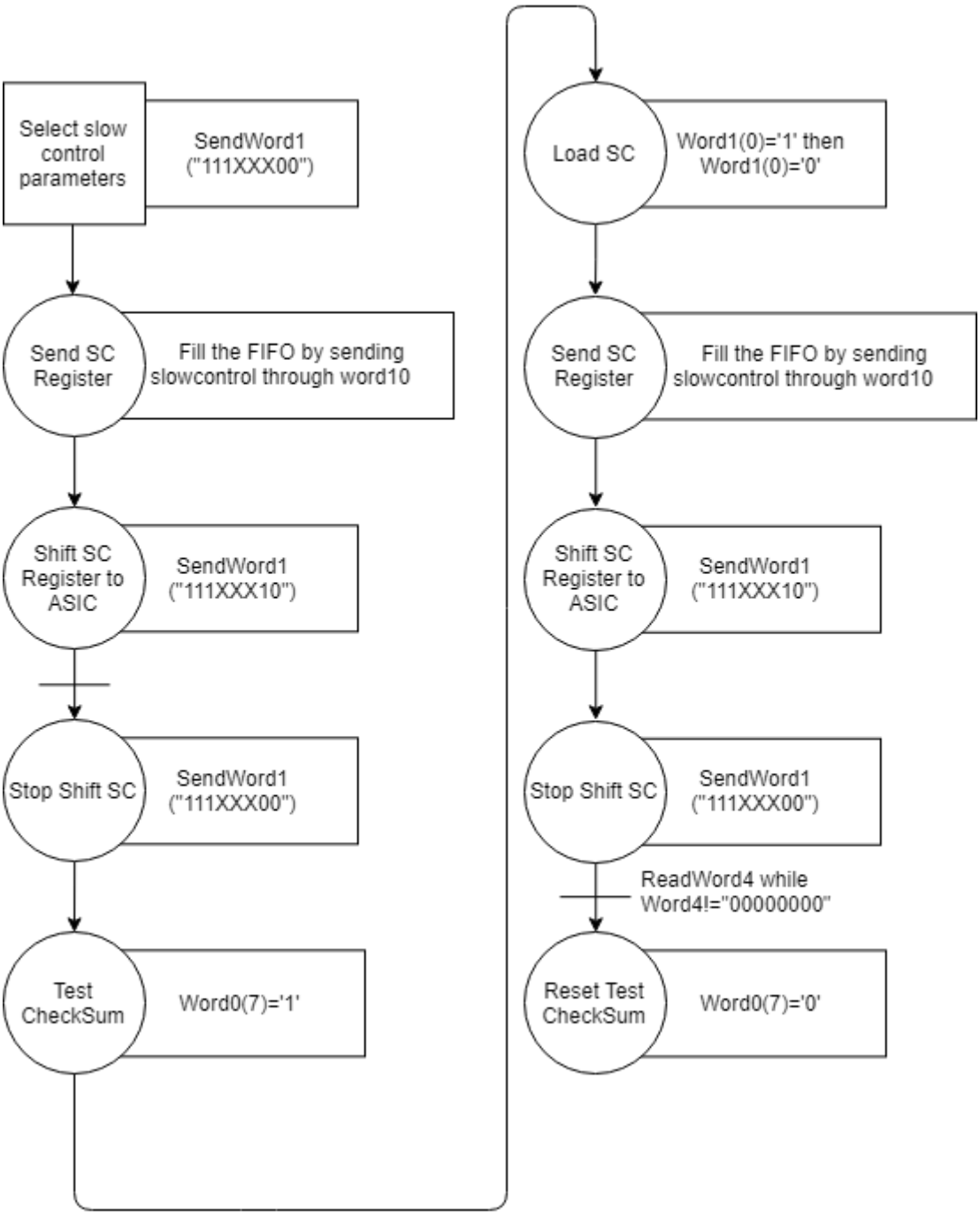


Figure 11 - StateMachine for Slow Control

Once the Slow control of 2288 bits is fixed, it will be possible to use the Word10 as a register for the entire String of data. When the Word10 have been filled by the Software, the Software will give the order to shift all the Data.
In order to test the value that we have to send to the ASIC, we are going to test the value by pushing the Slow Control from the register out of the ASIC through "srout", by sending the same Value of Slow control a second time. The Firmware will be able to compare the value of the output register "srout" with the FIFO which contains the Slow control register.

Comparing the value from "srout" with the Slow control will give you a result through Word4(7). When Word4(7) is equal to '1' the slow control passed well.

4.2.3 SiPM Staircase

The SiPM staircase measurement is used to locate the pedestal and 1, 2, 3 photoelectrons positions relative to the threshold DAC code. The SiPM must be plugged on the board and set in the dark with no parasitic light. A scan through the threshold DAC values will allow locating the pedestal and the first photoelectrons values thanks to the dark noise of the SiPM.

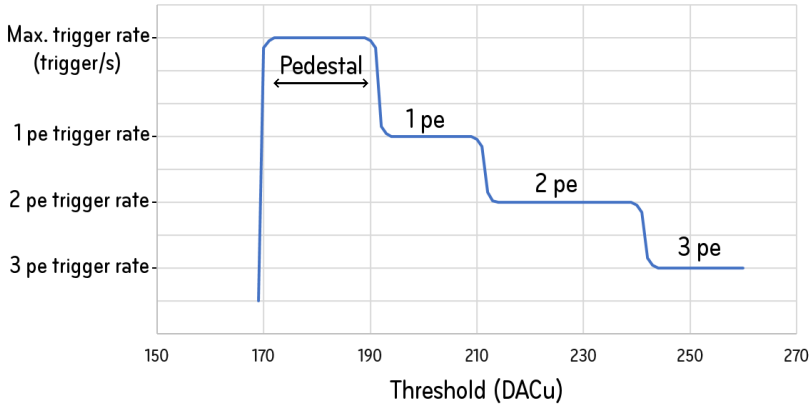


Figure 12 – Staircase Principle

Those values can be used to choose the threshold value or to calibrate the gain of the 64 SiPM cells thanks to the input DACs with the Slow control Register.

In order to perform this test, first a channel need to be chosen by the user using Word31 (see *Maud_FPGA.xls*) then the following state machine can be started:

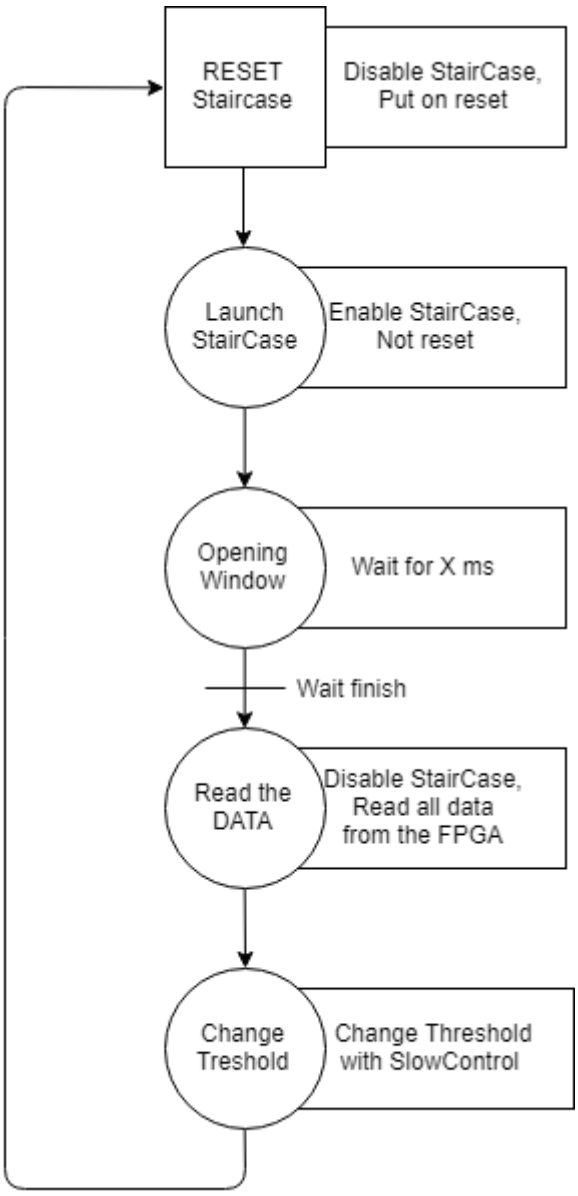


Figure 13 - State Machine for StairCase Software Point of View

The Opening Window, at this time, as to be done on the Software because it is a very long period of acquisitions (100ms minimum), not useful to implement it on the Firmware part. The longer the acquisitions is the more precise the staircase will be.
The Word which will be used for the Staircase is the Word6 :

WORD 6

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
	reset_staircase1	enable_staircase1	reset_staircase2	enable_staircase2			
	'0' reset the	'1' Enable the	'0' reset the	'0' Enable the			

	counter of Staircase for CT1	Staircase CT1	counter of Staircase for CT2	Staircase CT2

When the test is done on the value of CT1, the value of the counter (number of triggers) is contained in Words 70(7 down to 0),71(15 down to 8),72(23 down to 16), 73(31 down to 24) and the Software needs to change the value from bit5 and bit6.

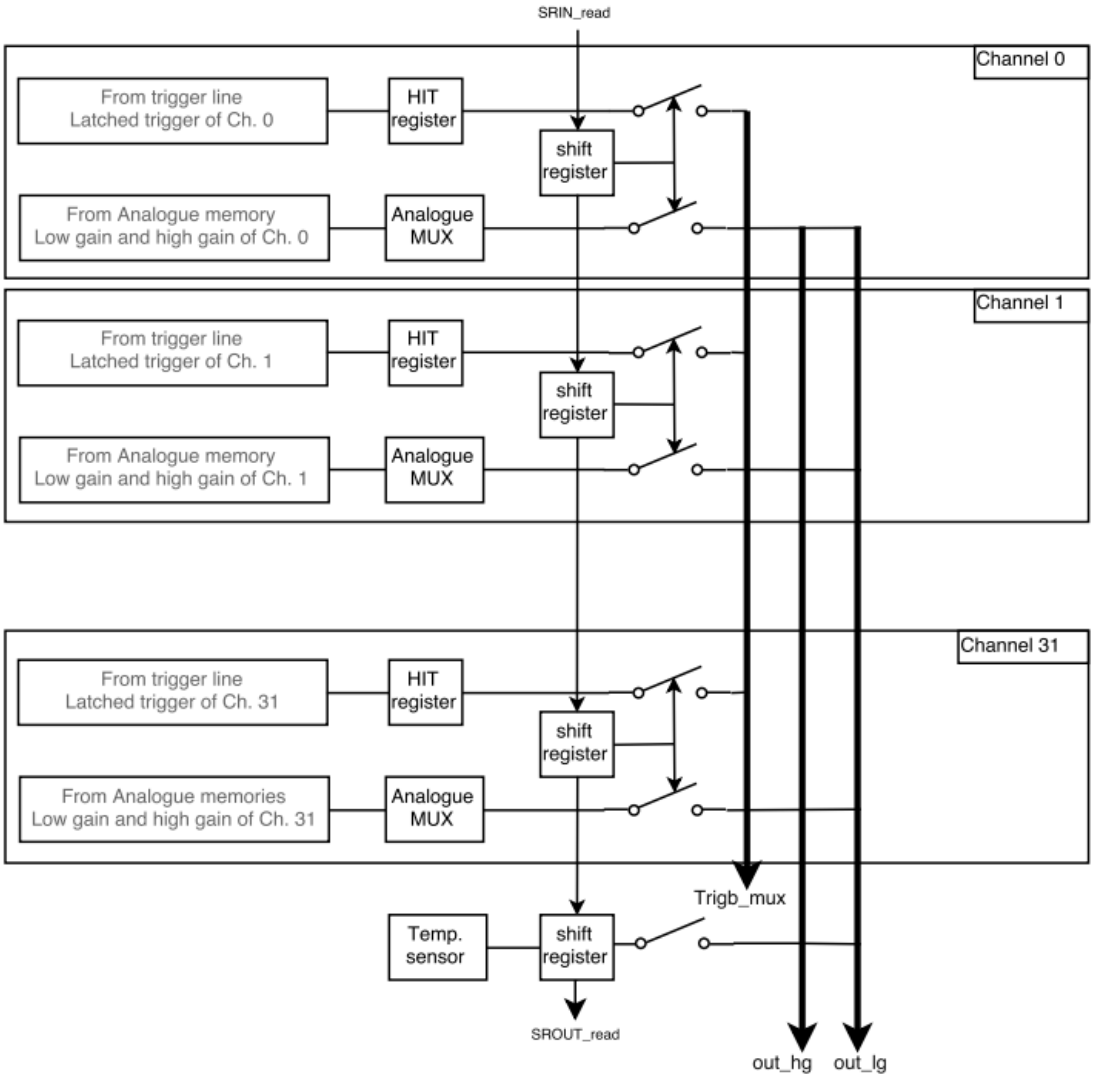
Repeating this test on every channel will give information about the threshold corresponding to the pedestal of every firsts photoelectron in every channel. The user will be able to change the DAC value through Slow Control in order to modify the pedestal in order to have them at the same threshold.

4.2.4 DAQ system

Citiroc1A got two analog charge measurement that can be read in serial using a multiplexed output providing sequentially 32 charges measurement as well as charge trigger status for each channel,

In order to get Data from Citiroc1A measurement, an ADC has to convert 4 analogs output (Out_LG1, Out_HG1, Out_LG2, Out_HG2) before sending them to the FPGA/PC for analyzing.

As it is done for Citiroc1A test board, read register will be used in order to place the output signal multiplexer on every channels shaper output. The channel selection on the analog output is done with a shift register equivalent to the slow control. It only has 33 (32 channels and Temperature) flip-flop and you need to have only one '1' in the shift register at the same time. The state machine for the read register, including the 32 channels Temperature, is managed by the Firmware.



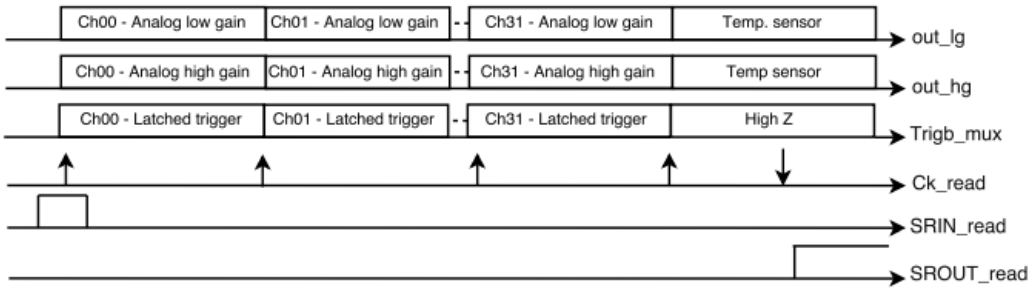


Figure 14 - Read register description scheme

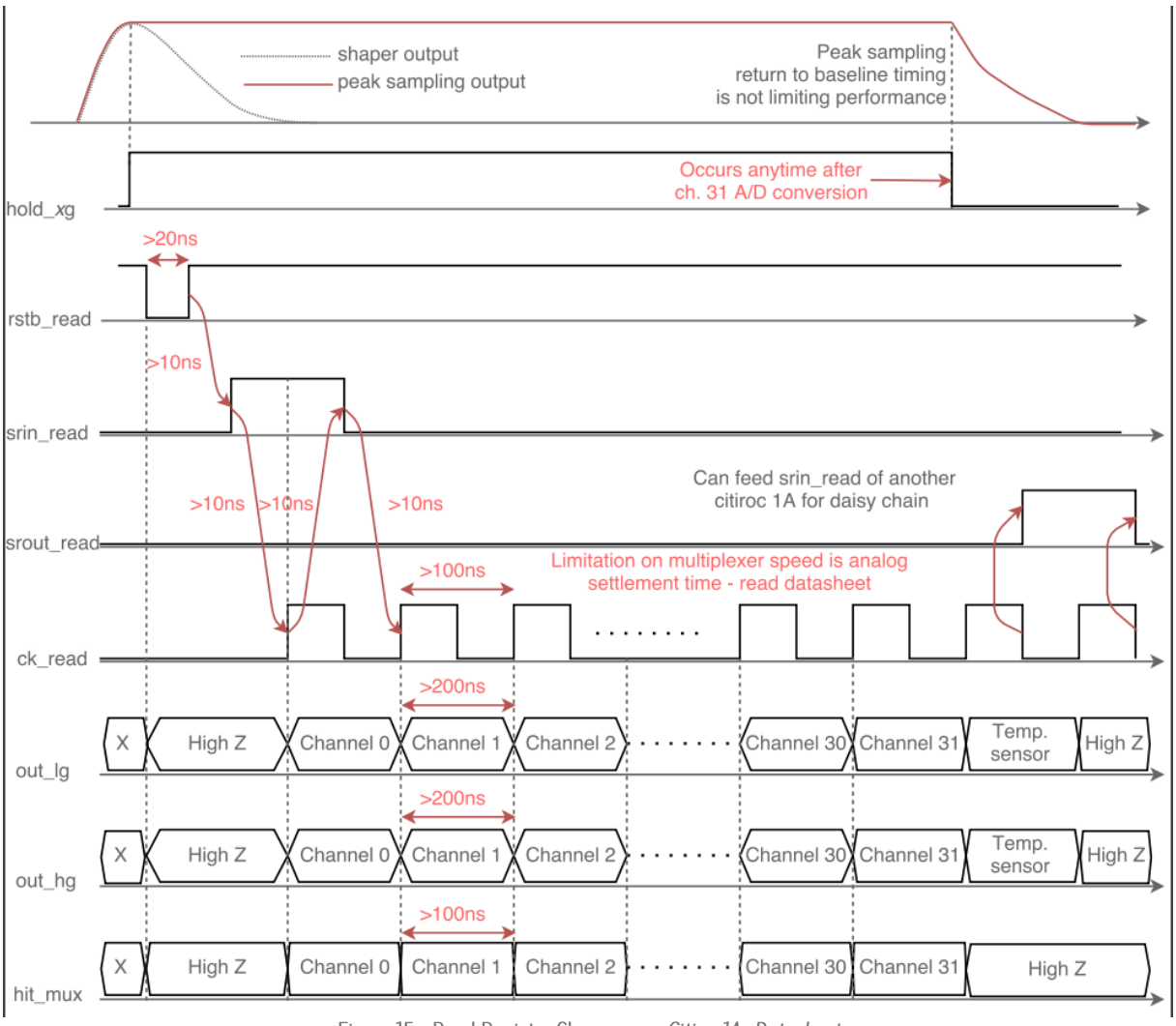


Figure 15 - Read Register Chronogram *Citiroc1A- Datasheet*

The hold_xg (hold_lg and hold_hg) will put on hold the value of every channel until it is released by the firmware either for both the track and hold and the peak detector mode.

4.2.4.1 Peak Sampling: Track and hold or peak detector

For more information see *Citiroc1a – Datasheet Section 6.3*

Two peak sampling are available in Citiroc1a. A peak detector that automatically stores the highest value of the slow shaper after being armed, and a track and hold mode that both requires a hold for storing the analog value a long time. Both options cannot be applied at the same time, it depends on a slow control bit which will choose between both. It is also possible to choose one option for the LowGain and the other for HighGain.

4.2.4.1.1 Track & hold

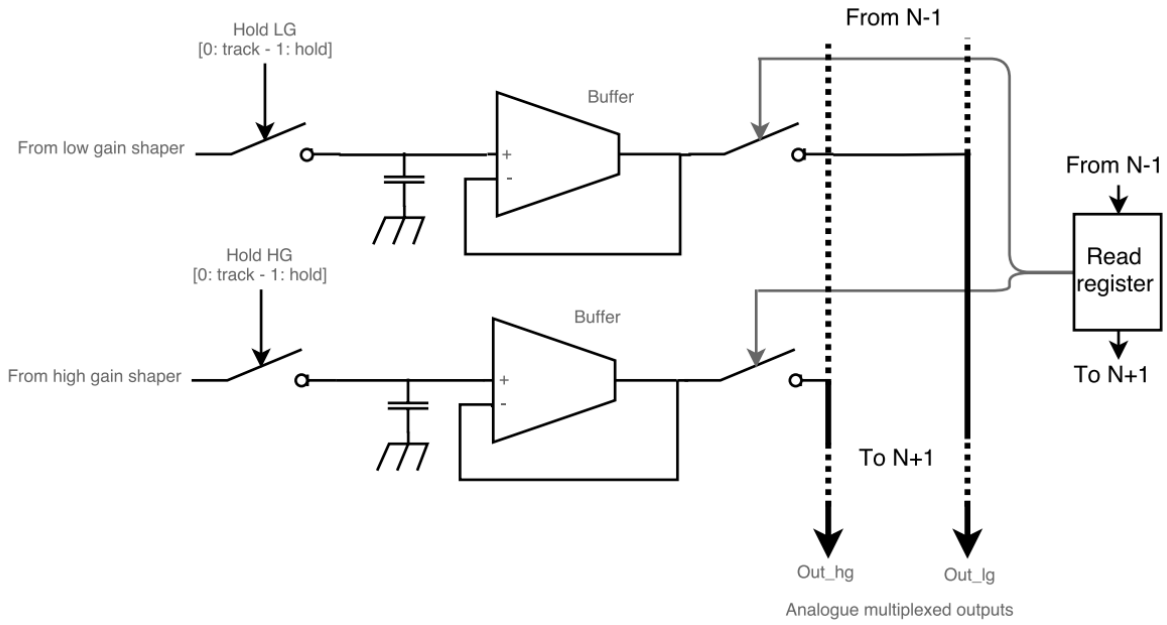


Figure 16 - Schematic of track and hold cell

When the hold signal is high level it will hold the value on every shaper at the moment the ASIC sees the '1' :

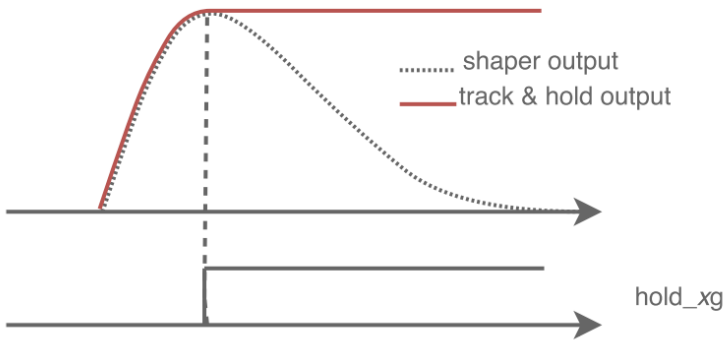


Figure 17 - Standard Track & Hold working mode

In read out mode, the hold signal must remain to a high level during the acquisition of the 66 channels.

4.2.4.1.2 Peak detector

Instead of using the track and hold solution, it is possible to sample the maximum of the shaped signal. It will be useful for detecting the maximum of the shape without knowing precisely the perfect timing.

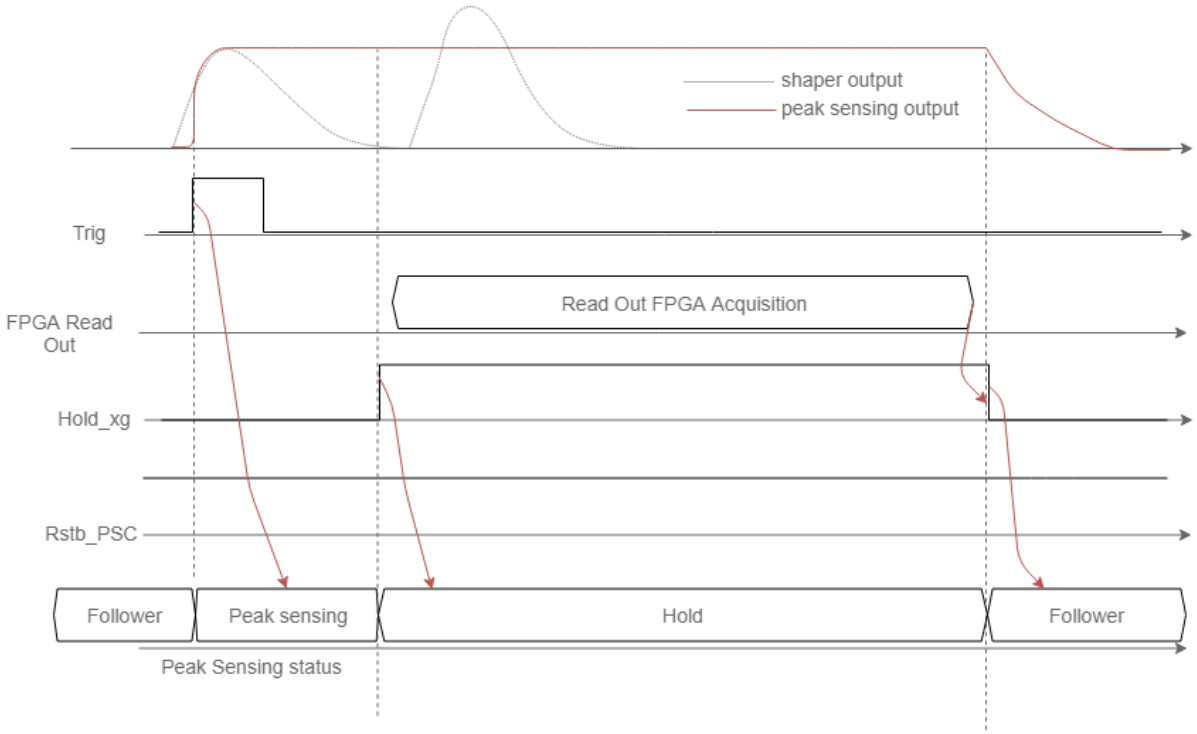


Figure 18 - Peak Detector mode

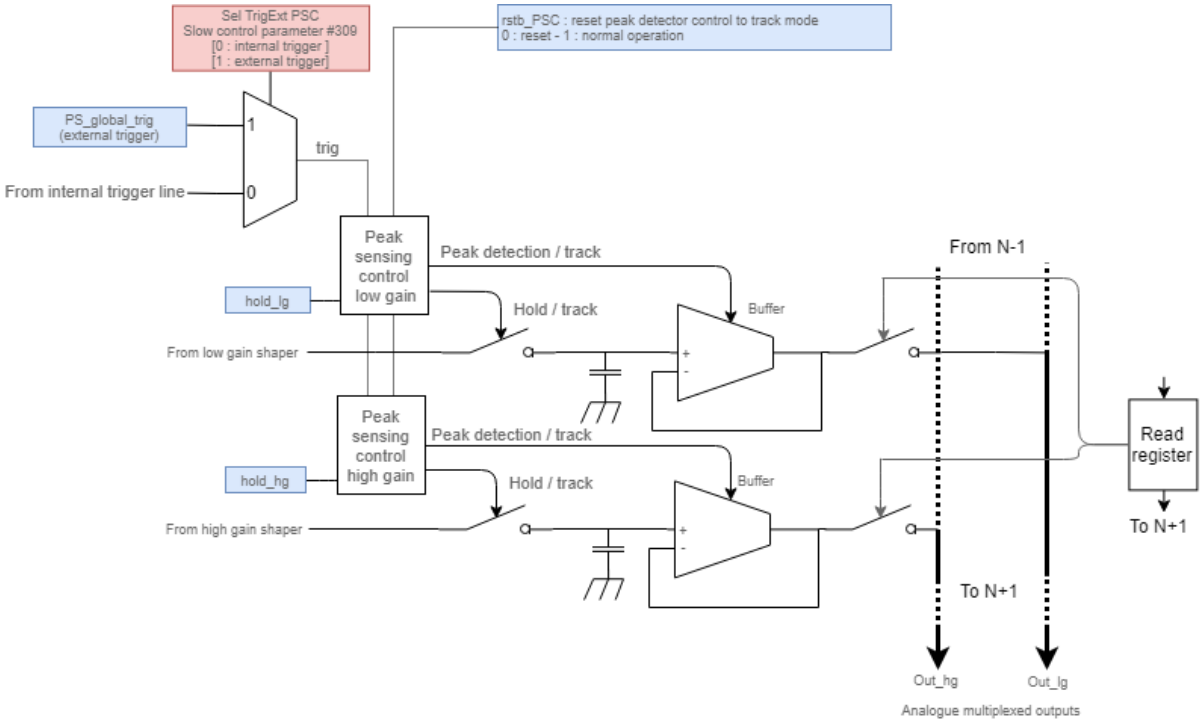


Figure 19 - Peak Detector block scheme

The peak sensing mode will only be effective when there is a trigger in the first place if there is no trigger the analog output while being close to 0 Volt. It's possible to start the Peak Detector process using an external trigger through a bit on the FPGA: PS_global_trig.

The peak detector switch to hold phase when that signal occurs, it will not memorize a higher value of input signal.

The reset pin (rstb_PSC) doesn't need to be used at each acquisition, the system automatically goes back to track mode, where the peak sensing cell wait for the trigger to rise. When the hold_xg signal goes to a low level it will go back to the track mode.

In the slow control, it is important to place bits #308 & #1452 to 0 in order to not bypass the Peak sensing cell.

Those signals are the same for both ASICs.

Either for Track&hold or PeakDetectore is important to fix a delay before sending the hold from the FPGA:

- The track and hold can be used, by the user, to trace the output shaper for low gain or high gain by varying the "hold" delay value.

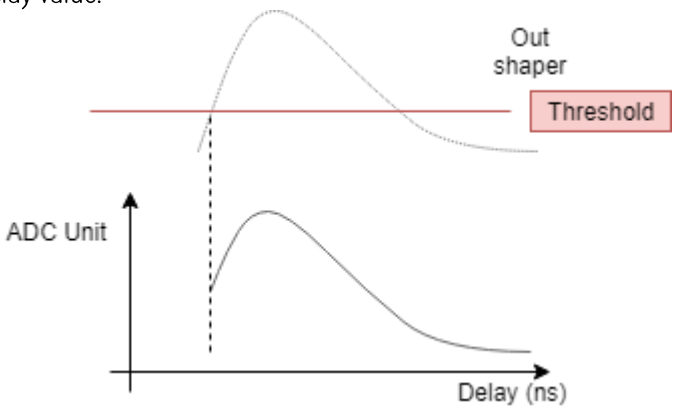


Figure 20 - HoldScan using Track&hold

  This measure will help in the second mode because the user knows the timing between the trigger and the peak of the signal.
- After getting the timing from the holdscan, the user can be able to position the delay in order to be sure to have the maximum of the signal before doing the readout. This way is the best for having a good readout system. If the delay is too early, the data which came from ADC will not be the maximum of the signal and it will lose information in term of charge.

This delay is programmable using the **Word30**:
The delay will be equal to 2ns*Word30(7 down to 0) with a range of 0 to 510ns delay. It is possible to change the clock in FPGA in order to reduce the "2ns" step. But be careful the FPGA can't use the 1Ghz clock in a proper way to perform a 1ns step.

In this configuration there are two analogs output, (LOW & HIGH Gain) of every channel, which is going to the ADC in order to be read.
ADC takes up to 200 ns to convert a value (30ns of CNV launch and 170 ns of Conversion Time). Same has Citiroc1A rate.

4.2.4.2 ADC configuration for DAQ

The ADC chosen for this system is LTC2325-14 provided by Linear Technology :
Providing simultaneous 4 Channels data conversion and 5Msps/channel data throughput rate (enough for the data rate of Citiroc1A), this ADC is one of the best options available on the market.

The fact that the converted data are sent to the FPGA through SPI serial interface does facilitate the routing process for this ADC. Including fewer signals to the FPGA through the SAMTEC connector than with a serial output for the Data.

The ADC has been configured to be able to convert in the range or the analog output from Citiroc1:
Basically, the span for the analog output from Citiroc is 1,2V to 3V. Subtracting 1V to this value put us between 200mV and 2V which is in the range of the ADC.
The value coming from the ADC will contain between "1000110 0100 0000" for the lowest value of HG/LG and "'11 1110 1000 0000" for the highest value of the analog output.

In this configuration, we use the Sign bit in order to have information when the analog part of the ASIC is undershooting, making the voltage value go under 1Volts. When 4 SiPM Matrix are wired to the board, the undershoot is not happening.

4.2.4.3 Software point of view for DAQ

The first thing to take into account is the Trigger which will be used to start the acquisition process, in order to have useful value Citiroc1a need to have a trigger. Depending on the use of the system there are two possibilities :

- Charge Trigger in order to have a Hit on each channel where the signal is going over the charge threshold
- Time Trigger, where there is the possibility to have 0 Hit in an acquisition. Because the Hit is only depending on the Charge Trigger and the Trigger need to be latched using the Slowcontrol. Without latch, the Hit will not be memorized.

The fact that there are 2 ASICs on the same board, keep in mind that it's important to have the external trigger "on" (SlowControl) allowing one ASIC to Trigger the other one.

In the DAQ part, the software/firmware uses exactly 10 words: Words 20, 21, 23, 24, 25, 26, 27, 28 for Data. Word 22 for FIFO control, Word 4 in order to tell the software that data are ready, Words 45, 46 for the number of acquisitions and Word 43 in order to launch the process when the Software is ready.

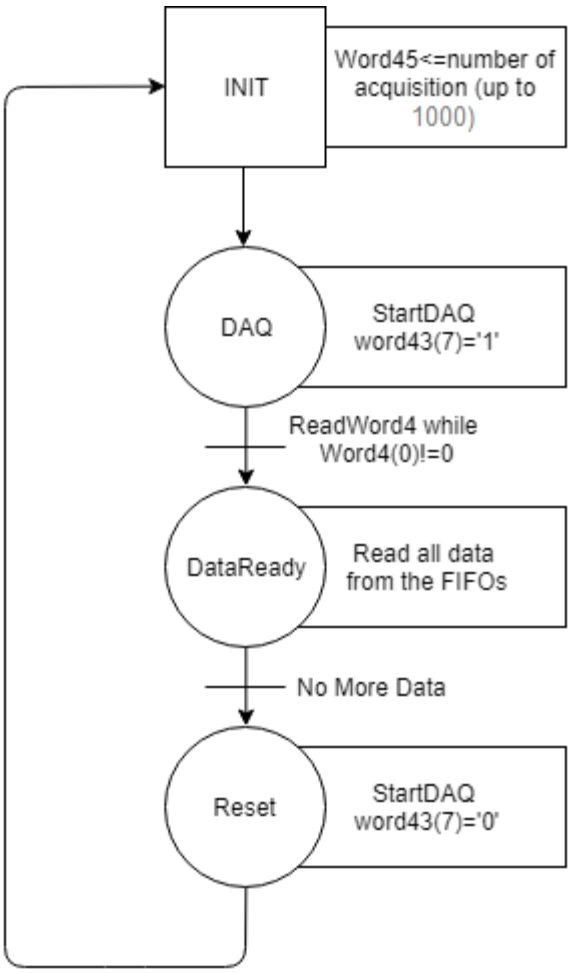


Figure 21 - Simplified DAQ State Machine from the Software point of view

The sequence from the Software is to load the number of acquisition wanted (1000 maximum) then launch DAQ with a SendWord43='10000000'. Then when the firmware has done every acquisition it will tell the software that the data are ready in the FIFO putting Word4 (0) to '1'. The Firmware also allows the possibility to stop the Acquisition when it is needed without losing data:
Reading Word50 and Word51 will allow the user to know how many acquisitions have been done:

Number of Acquisitions = Word51*256 + Word50 -100. We subtract the flushing data from the FPGA (100 at this time).
If this value is under 0, there is no data in the FIFOs.
Reading the Number of Acquisitions before "RESET" is important, if not the Data will be cleared.

The data are stored into FIFOs as it is described under:

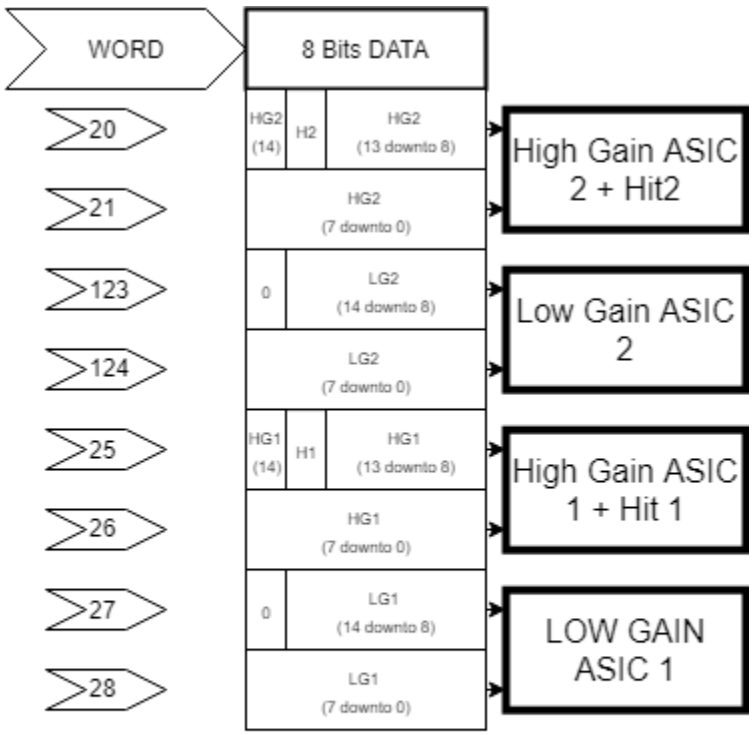


Figure 22 – Different Word for the DAQ process

Every Channel needs to be converted from analog to digital in order to get the data from the ASIC. This will give you 33 Channels to convert per ASIC. Basically, for each acquisition, there will be 33(32Ch+T°) * 8Bits in every FIFO.
Each FIFO can be emptied by reading the equivalent Word with the Software as it is shown in Figure 19.
Let's take the example where a user asks for N acquisitions:

- Each FIFO contains N*33 *8 Bits which (N up to 1000).
- In order to get all these data, the user will need to read the data from the FIFO.
- The Software allows the user to use multiple reading function in order to get the data from the acquisitions:
    - byte[] FIFO20 = **readWord(20,N, UsbDevID)**
    - byte[] FIFO21 = **readWord(21,N, UsbDevID)**
    - byte[] FIFO23 = **readWord(123,N, UsbDevID)**
    - byte[] FIFO24 = **readWord(124,N, UsbDevID)**
    - byte[] FIFO25 = **readWord(25,N, UsbDevID)**
    - byte[] FIFO26 = **readWord(26,N, UsbDevID)**
    - byte[] FIFO27 = **readWord(27,N, UsbDevID)**
    - byte[] FIFO28 = **readWord(28,N, UsbDevID)**

This functions will give you a table of Bytes in order to process all the data.
The only thing the software need to do is to concatene :

- Byte[] FIFO20 & Byte[] FIFO21 for High Gain 2 + Hit2,

- Byte[] FIFO23 & Byte[] FIFO24 for Low Gain 2,
- Byte[] FIFO25 & Byte[] FIFO26 for High Gain 1 + Hit1,
- Byte[] FIFO27 & Byte[] FIFO28 for Low Gain 1.

Then it will be possible to process all the data through the Software.

When every FIFO has been emptied, putting Word43(7) to '0' will reset the state machine allowing the user to launch another series of acquisitions.
At any time in the process, you can go back to the init State by switching Word43(7) from '1' to '0'. This action will clear all the FIFO and reset the ADC part.

For example, the software asks for 2 acquisitions, every FIFO (**word 20 ,21,123,124,25,26,27,28**) are filled as shown below:

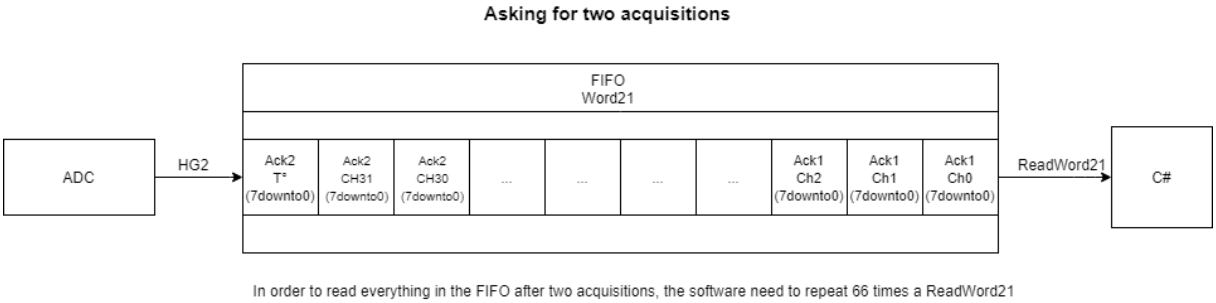


Figure 23 - FIFO example for 2 acquisitions

In the C# you will have 66 Bytes of 8 bits for every FIFO which can be decoded as you want.

4.3 High-Voltage Module

4.3.1 The configuration of the PowerModule

In order to use the Power Module, Zeus should be installed :
https://www.caen.it/?downloadfile=1364

This Software allows to put control value :

- High Voltage value
- Clamping Current (the Output turn Off when the output current is higher than the clamping value)
- Ramps Speed for the High Voltage
- The Enable/Disable for High Voltage

It is possible to know in real time the Real High Voltage at the Output of the Module and also the Current at the Output, allowing the user to know the behavior of the module. And of the SiPM.

Basically, this is the default configuration that should be used at the start, some adjustment should be done to do precise measurement :

Channel	Output Voltage Enable	Status	Bias Voltage (V)	Compliant Voltage (V)	Maximum Current (mA)	Operation Mode	Output Voltage (V)	Output Current (mA)	Set Point (V)	Reference Voltage (V)	Sensor Temperature (°C)	SiPM Temperature Coefficient (mV/°C)	Peltier Controller Enable	Peltier Controller Set Point (°C)	Peltier Controller Temperature (°C)	Apply Settings
COM10\\CH0	☑		56,25	80,000	0,8000	Digital	56,155	0,14533	56,250	0,540	0,00	0,00000	☐	25,000		Apply

Do not take Output Current as a reference, it is depending on the SlowControl part (InputDac) and on the Matrice itself.

The maximum current should be placed at a value where there is no risk of damaging the matrices. 10uA/SiPM Channel is a nice start allowing the detector to go up if there is a lot of light. This maximum current will allow the module to power off if there is to much light on the Matrice.

0.64mA is a good value to start with.
56.25V as a Bias Voltage is only coming from the Breakdown of the SiPM + the Input DAC. Here BreakDown Voltage is 54V and The Input Dac is 2.2V. It is interesting to be above the BreakDown Voltage.
If the ASIC is used with the 2.5V Input DAC the value at the middle ('128') will be 1.2V, which implicate that the breakdown voltage will be obtained at a lower voltage from the power module.
There is the possibility to put a Compliant Voltage, in order to be sure to protect the SiPM.

4.3.2 Communication with the Power Module through Serial

At this time the module use simply AT commands via the serial port.
In principle this is the principal command :

In principle, you can just open the serial port COMxxx and send the following commands (all command should be confirmed with entering \n\r)

```
AT+MACHINE      //enter in machine control mode
AT+SET,1,0      //force digital control mode
AT+SET,2,45.3   //set the voltage to 45.3V
AT+SET,0,1      //power on the module
```

to read voltage back, for example, you can send
AT+GET,231

Serial port should be opened at 115200 bps,8,1,n

You can find the other command on the Section 4.Uart Interface of the A7585_rev0.

5 Conclusion

Using this document, the excel annex, the Citiroc datasheet and the software as an example, give all the needed information to communicate with the Boards and make the equivalent measure as it can be done on the Citiroc_Test Board.

There are few other functions that have been implemented in the Firmware, but there is no point in presenting them in this document.