Dragon ver.5 Front-end Board Document

Author	*Laboratory*
Yusuke KONNO	*Kyoto University*

Approved by	*Laboratory*

List of Abbreviations			

History		
Version	Date	Observation
0.1	20120725	
1.0	20130527	
1.5.1.04	20150426	firmware version 5.1.04
1.5.1.05	20150514	firmware version 5.1.05
1.5.1.08	20150723	firmware version 5.1.08
1.5.1.09		
1.5.1.0B	20151224	firmware version 5.1.0B
1.5.1.0B.0C	20160129	firmware version 5.1.0B.0C
1.5.1.0B.0E	20160314	firmware version 5.1.0B.0E
1.5.1.0B.0E.02	20160315	firmware version 5.1.0B.0E update the old version 3 contents (still temporary)
1.5.1.0C.00	20160414	firmware version 5.1.0C.00
1.5.1.0C.11	20160513	firmware version 5.1.0C.11
1.5.1.0C.13	20160531	firmware version 5.1.0C.13

Distribution	

Table of Contents

1 Software and Firmware Package

You can get the readout software and firmware of Dragon. Please contact Yusuke KONNO (konno@cr.scphys.kyoto-u.ac.jp) to get the package. The contents of the package (zip file) are as below. VV and SS mean the version number and the subversion number respectively.

- DominoSoft_v5_1_VV
  - software for data acquisition (Hereafter we call this directory DominoSoft)
- DRS_HDL
  - firmware source codes
- dv5_1_VV_SS.mcs
  - mcs file (firmware for writing to PROM)
- dv5_1_VV_SS_jtag.bit
  - bit file (firmware for JTAG download to FPGA)
- dragon_document_v5_1_VV_SS.pdf
  - this document
- ChangeLog
  - firmware change log

To compile the software, change the directory to DominoSoft_v5_1_VV and do a “make”. Before compiling, ROOT has to be installed in your PC.

```
[user@host]$ cd Dominosoft_v5_1_VV
[user@host DominoSoft]$ make clean
[user@host DominoSoft]$ make
```

After compiling, you can see the following programs in DominoSoft.

- Eth
  - Data acquisition through the Ethernet.
- EthDispCas
  - Data acquisition with the event display.
- rbcp
  - Slow control
- Mkoffset
  - Make pedestal table
- PlotRaw, PlotSubt, PlotSubtAll
  - Waveform plotter

You can find also other directories in DominoSoft.

- Macro
  - Macros to process the data
- RBCP
  - rbcp source codes

- table
  - Slow control configuration table
- SlowControl
  - Slow control monitor programs
- FPGA-Config
  - Programs for remote firmware installation through the Ethernet
- Other
  - Other programs

2 How to Operate

2.1 Setup

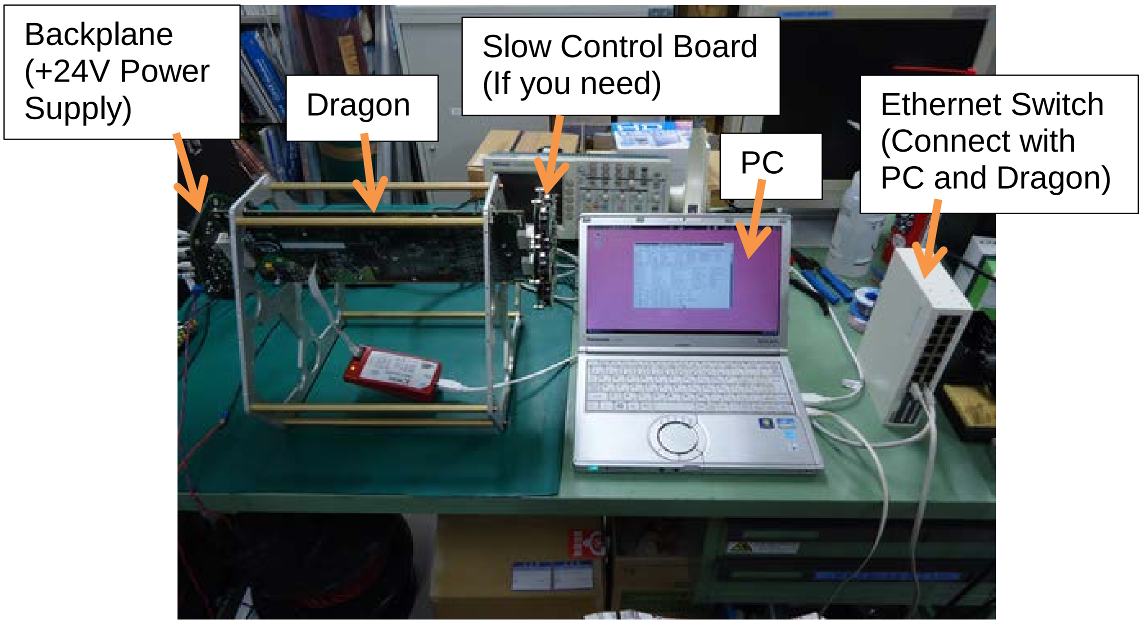


Figure 1: Setup for version 5.

Figure 1 shows an example of the setup for Dragon version 5. For the minimum setup you need **+24V power supply**, Dragon, backplane and PC. Connect the backplane to PC with a LAN cable. If you use Ethernet switch, take care that they are not connected to the outside network. Unless one purchase the MAC address and write it in EEPROM on the Dragon, Dragon does not have MAC address and it is prohibited to be connected to the global network. If you are using Dragon version 3, the power supply is +12V and please read the old version document.

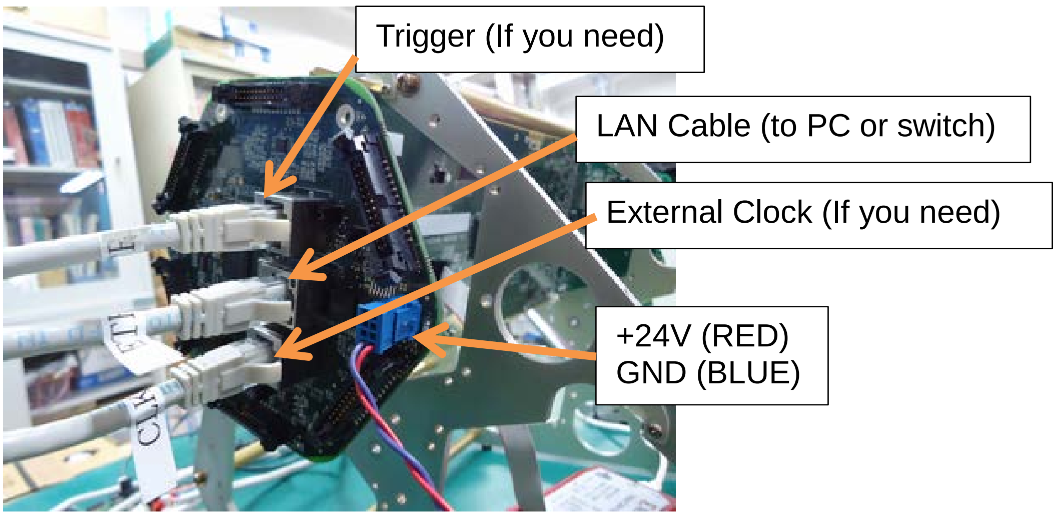


Figure 2: Backplane side (analog backplane)

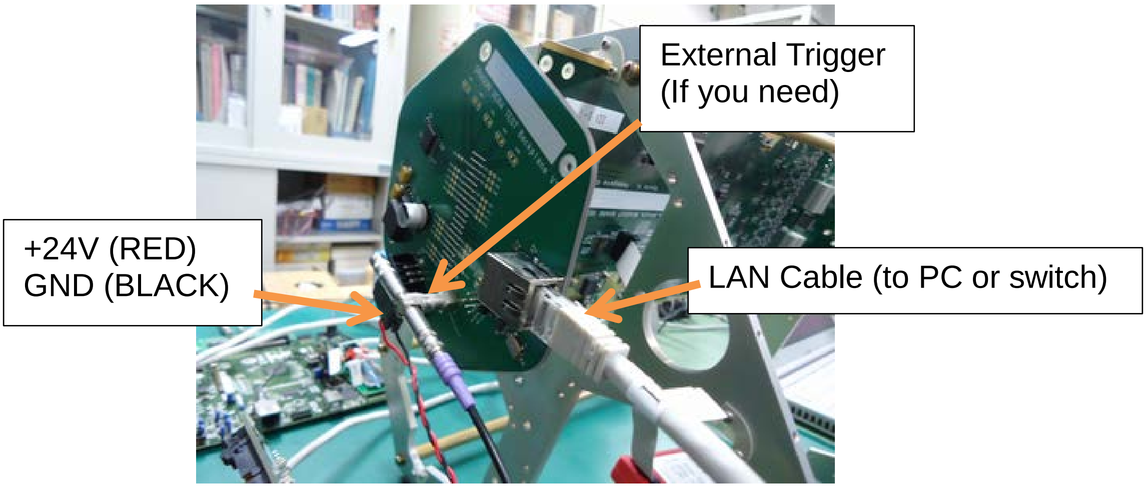


Figure 3: Backplane side (fake backplane)

For the backplane, you can use analog backplane (BP), which analog trigger team produced for the LST (Figure 2). For the test purpose, we produced the fake backplane, which also can be used for the test operation (Figure 3).

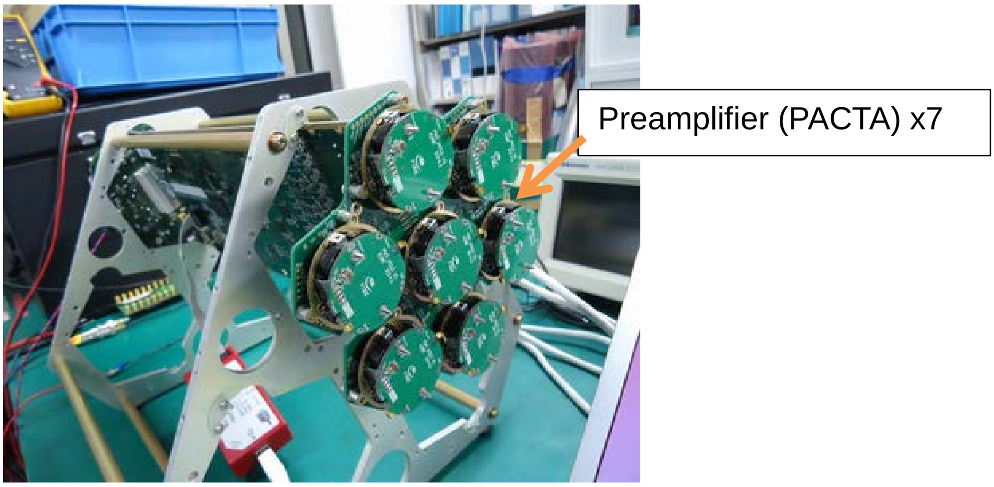


Figure 4: PMT side (Slow Control Board)

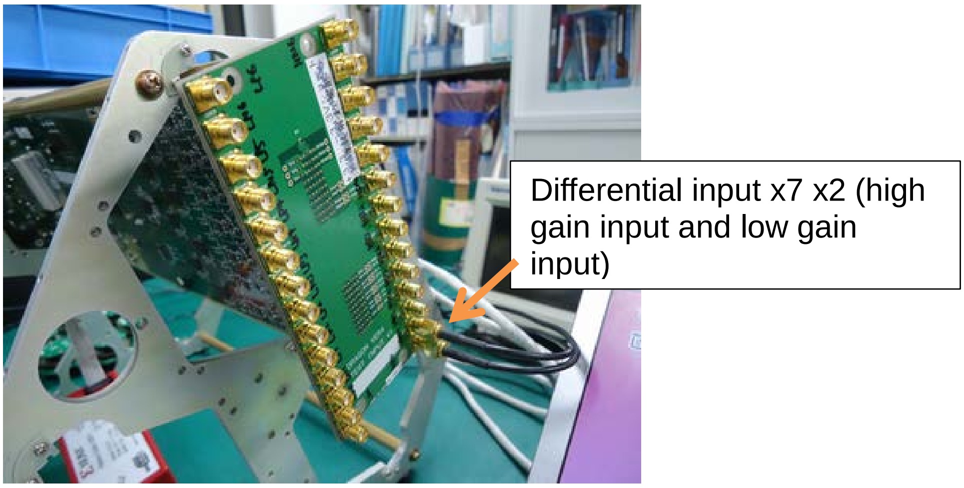


Figure 5: PMT side (test input board)

You can connect the Slow Contorl Board (SCB) to the PMT side of the Dragon (Figure 4). SCB controls the high voltages for PMT, monitors those high voltages, also monitors PMT anode currents and temperature and can generate test pulses. If you don’t need these functions, you don’t need to prepare the SCB. We also produced the test input board for the test purpose (Figure 5). It simply has 14 differential input SMA connectors and you can inject the signal to the Dragon with them.

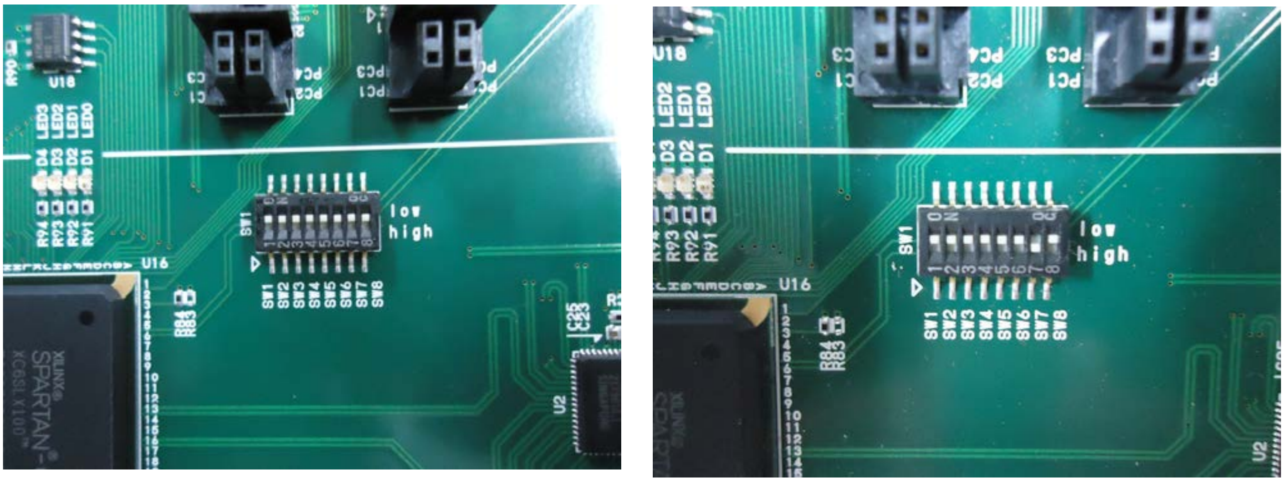


Figure 6: Configuration of DIP switches. Configuration using default IP address (left) or IP address written in EEPROM (right). The photos are from Dragon ver.3 but you can find the same switches for ver.5.

You should set DIP switches correctly. The 7th bit of the switches decides the board IP address. When it is set to low, the board works with default IP (192.168.10.16). When it is set to high, the board gets IP address written in EEPROM. MAC address has to be written in EEPROM when the board uses IP address in EEPROM. On the Dragon you may find the label on which IP address is written if someone already wrote the IP address. The 8th bit of the switches is the reset switch. Set it to high when you want to initialize the state machine during operation, and in other cases set it to low.

We buy board MAC address from a company and write the address in EEPROM. Some boards don’t have MAC address and in that case don’t connect the board to the global network.

Set IP Address

Set IP address of the PC to 192.168.10.***. and test the connection to the board.

```
[user@host DominoSoft]# ifconfig eth0 192.168.10.1
[user@host DominoSoft]$ ping 192.168.10.16
```

Here I use 192.168.10.16 for the board IP and 192.168.10.1 for the PC.

2.2 Pedestal Run

Slow Control

The Dragon board has the internal register for slow control. You can write and read bytes in this register by the software. You can find the register map (address and its content) at section 4 in this document. Since **the register map can be different for the different firmware version**, please check the map in the corresponding document version if you find problem with the command below.

```
[user@host DominoSoft]$ ./rbcp 192.168.10.16 4660
SiTCP_RBCP$
```

RBCP (remote bus control protocol) is the protocol for slow control. Port number for RBCP is 4660.

```
SiTCP_RBCP$ help
```

This command shows help.

Before taking pedestal run, you should set registers correctly.

```
SiTCP_RBCP$ wrb x100b 2
SiTCP_RBCP$ wrs x1090 100
```

In the slow control register inside Dragon, each address has 1byte data length. Command “wrb” means “write byte” and writes 1byte (=8bit) length of data at the address indicated. Command “wrs” means “write short” and writes 2bytes (=16bit) length of data from the address indicated. In the case above, the data 2 (=0x02) is written at address x100b and the data 100 (=0x0064) is written from x1090 (content is 0x00) to x1091 (content is 0x64).
For pedestal run, write 2 to the register x100b. This register decides the trigger type. 0: L1 trigger from BP, 1: locally generated L1 trigger, 2: pedestal run (periodical internal trigger) and so on. Register x1090-x1091 is for the length of readout window (Region of Interest, RoI). In this case each event has 100 sampling points. If you change the read depth (= length of readout window), you should change the Config.h in DominoSoft and make them again. In the Config.h you can find the

line “#define READDEPTH 100” and you should change this number.

```
SiTCP_RBCP$ wrs x1012 x0014
SiTCP_RBCP$ wrb x1011 x00
SiTCP_RBCP$ wrb x1010 xff
```

These settings are necessary to set ADC format appropriately for the software.

Instead of typing the command one by one, you can load the command from the file. In the directory DominoSoft_v5_1_VV_SS/table/ you can find such predefined command tables. Within these tables, the file “daq” is a table which includes the typical setting for the data taking. You may load this table for the pedestal run.

```
SiTCP_RBCP$ load table/daq
```

Before loading “daq”, you should check inside the file and edit the numbers if it is necessary.

```
[user@host DominoSoft]$ cat load table/daq
#daq set

#trigger enable
wrb x101e 1

#trigger select
#level1 (telescope)
wrb x100b 0                         <-You should put “#” for pedestal run
#level1 A async (local cluster)
#wrb x100b 1
#pedestal (self)
#wrb x100b 2                        <-You should remove “#” for pedestal run
…
```

Data Taking

To take data, you should run the program Eth or EthDispCas.

```
[user@host DominoSoft]$ ./Eth 192.168.10.16 24 10000 100 0 data.dat
100
200
300
...
10000
```

Data taking will start. The usage of the command can be found like:

```
[user@host DominoSoft]$ ./Eth
Usage: ./Eth <IP address> <Destination Port #> <event No> <READDEPTH> <option 0:datarun 1:warmup>
<filename>
```

<IP address>: IP address of the Dragon
<Destination Port#>: Port Number for TCP/IP. Set 24.
<event No>: How many events you want to take.
<READDEPTH>: length of the readout window
<option 0:datarun 1:warmup>: 0=usual data run, 1=warm up (data are not written in a file and run will continue until interrupted)
<filename>: data file name

Event Display

You can see the event display by running EthDispCas.

```
[user@host DominoSoft]$ ./EthDispCas

Usage: ./EthDispCas <IP address> <event No> <filename>

[user@host DominoSoft]$ ./EthDispCas 192.168.10.16 10000 data.dat
```

Then you will get more exciting view (Figure 7).

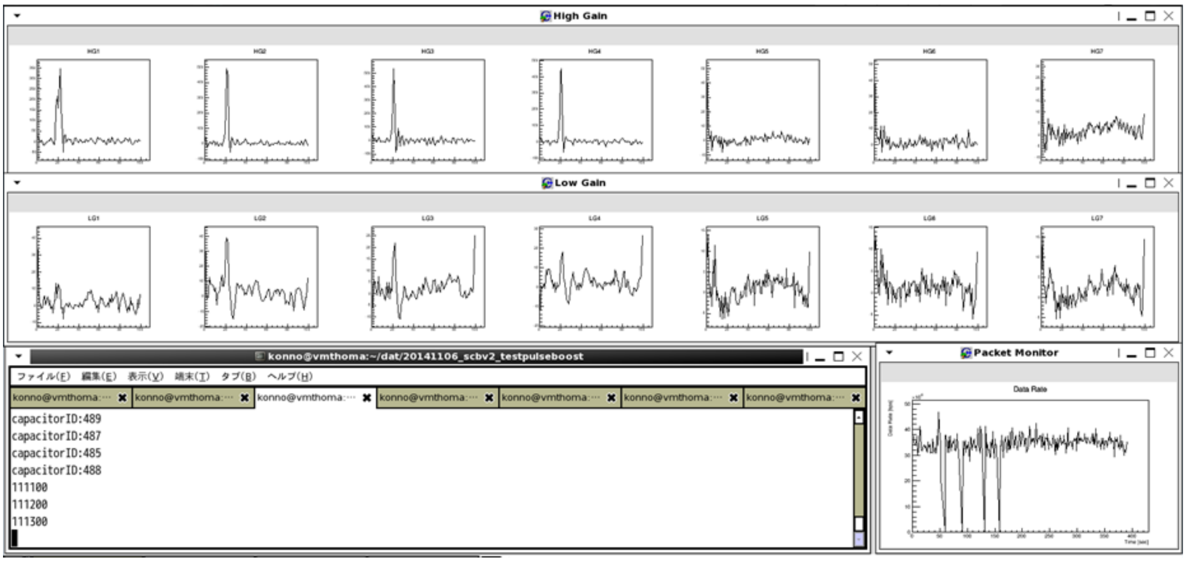


Figure 7: Event display example

Create Pedestal Table

Each capacitor in DRS4 has its own pedestal value. You should create such a pedestal table and subtract from the data to reduce a pedestal RMS. You can use the program Mkoffset for this purpose.

```
[user@host DominoSoft]$ ./Mkoffset
```

```
Usage: ./Mkoffset <input filename> <output filename> <read depth>

[user@host DominoSoft]$ ./Mkoffset data.dat offset.dat 100
```

Mean of pedestal is calculated for each capacitor from data file and offset.dat is created.

Plot Data

```
[user@host DominoSoft]$ ./PlotRaw

Usage: ./PlotRaw <data file> <channel(high gain:0-7 low gain:8-15)> <eventNo.>

[user@host DominoSoft]$ ./PlotRaw data.dat 0 100
```

Raw waveform is plotted for chnnel 0 high gain, event number 100 (Figure 8 left).

```
[user@host DominoSoft]$ ./PlotSubt

Usage: ./PlotSubt <data file> <pedestal file> <channel(high gain:0-7 low gain: 8-15)> <eventNo.>

[user@host DominoSoft]$ ./PlotSubt data.dat offset.dat 0 100
```

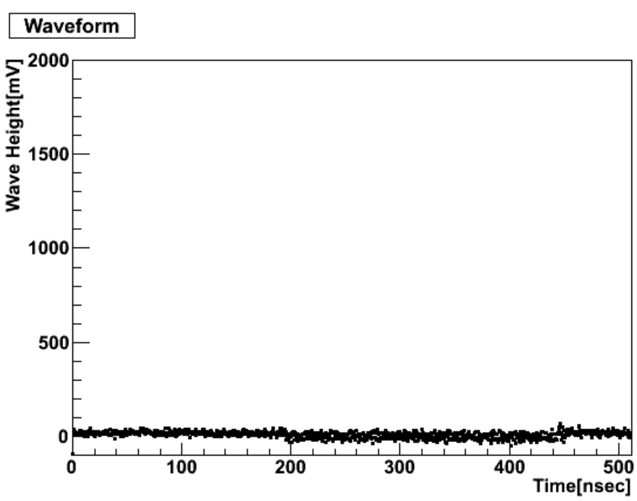
Waveform is plotted applying offset correction (pedestal subtruction) for chnnel 0 high gain, event number 100 (Figure 8 right).

```
[user@host DominoSoft]$ ./PlotSubtAll

Usage: ./PlotSubtAll <data file> <pedestal file> <gain (0:high 1:low)> <eventNo.>

[user@host DominoSoft]$ ./PlotSubtAll data.dat offset.dat 0 100
```

Waveforms are plotted for all channels, event number 100.



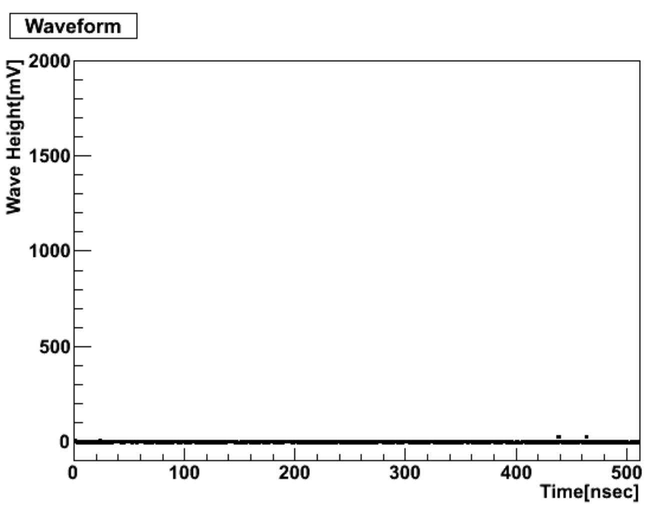


Figure 8: Raw data plot (left) and plot after offset correction (right)

2.3 Data Run

This is only for the firmware for the digital trigger and the instruction about how to get the data with the trigger locally generated by the L0 digital trigger mezzanine. Here we injected the signal with the test input board.

```
[user@host DominoSoft]$ ./rbcp 192.168.10.16 4660
SiTCP_RBCP$ wrb x100b 7
```

Set the trigger type to L0.

```
SiTCP_RBCP$ wrb x21 16
SiTCP_RBCP$ wrb x22 16
SiTCP_RBCP$ wrb x23 16
SiTCP_RBCP$ wrb x24 16
SiTCP_RBCP$ wrb x25 16
SiTCP_RBCP$ wrb x26 16
SiTCP_RBCP$ wrb x27 16
SiTCP_RBCP$ wrb x20 xff
```

Set the trigger thresholds (DAC values) for the digital L0 trigger. The range of DAC value is 0-255 (8 bits). Default value is 16. Each address (x21-x27) corresponds to each input channel. DAC values are overwritten when you set “xff” to the register x20.

```
[user@host DominoSoft]$ ./Eth 192.168.10.16 24 1000 100 0 data.dat
100
200
300
...
1000
[user@host DominoSoft]$ ./PlotSubtAll data.dat offset.dat 0 100
```

Take data run and plot waveform (Figure 9).

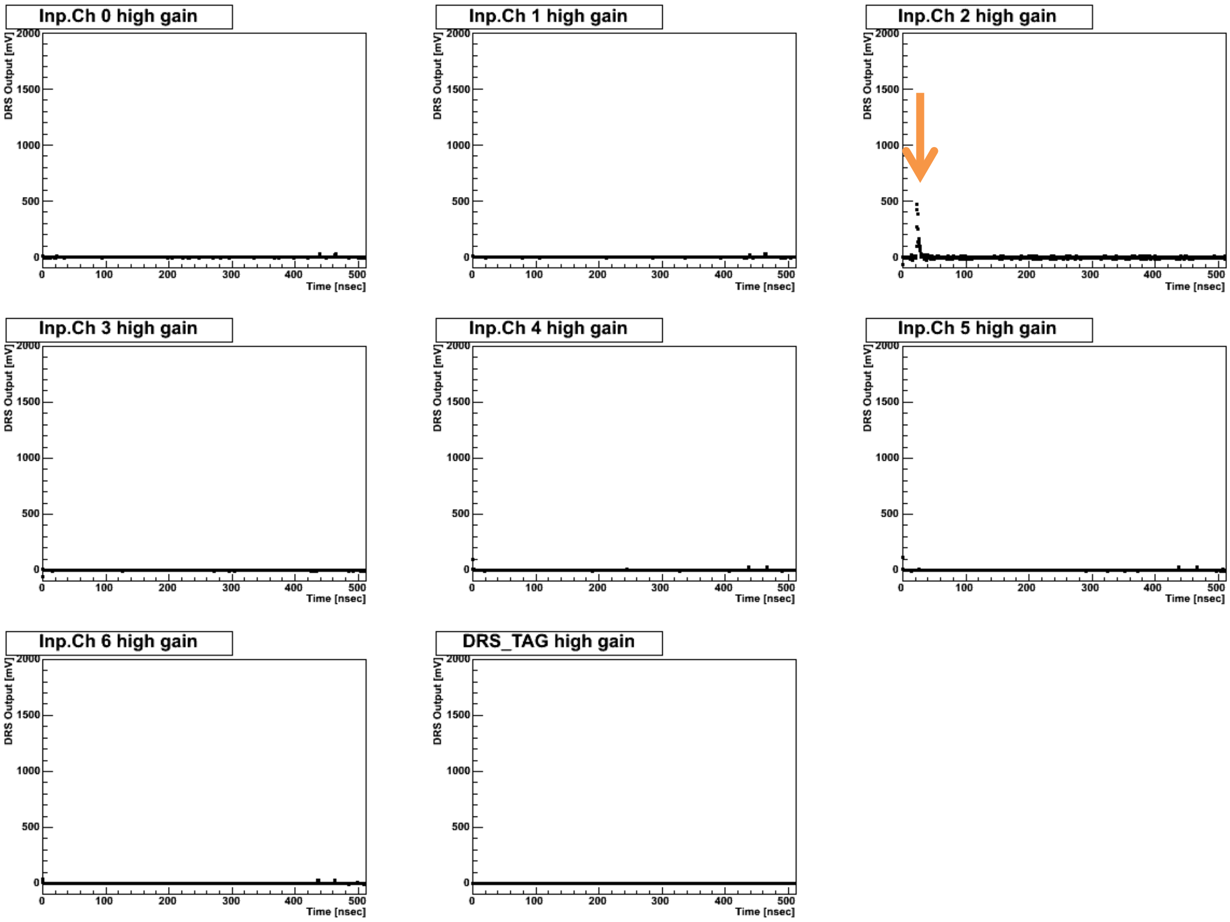


Figure 9: Plot of all channels for pulse input.

2.4 Slow Control

Check the Firmware Version Number

You can read the firmware version number from a slow control.

```
SiTCP_RBCP$ rd x1000 3
  [0x00001000] 51 0b 0e
```

This is the firmware version 5_1_0B and subversion 0E i.e. version 5_1_0B_0E. “5” means Dragon ver.5 (board version). “1” is after implementing the DRS4 channel cascade (before was 0). “0B” is major version number and “0E” is minor update or debugging version.

Set Parameters from a Configuration File

You can load slow control commands from a file.

```
[user@host DominoSoft]$ cat table/daq
#daq set

#trigger enable
wrb x101e 1

#trigger select
#level1 (telescope)
wrb x100b 0
#level1 A async (local cluster)
#wrb x100b 1
#pedestal (self)
#wrb x100b 2
…

[user@host DominoSoft]$ ./rbcp 192.168.10.16 4660
SiTCP_RBCP$ load table/daq
```

Reset Command

To reset and restart the board, set “xff” to the register x1008. After reset, the parameters for slow control are set to default value.

```
SiTCP_RBCP$ wrb x1008 xff
```

Trigger Frequency of Pedestal Run

Trigger frequency of pedestal run is written in the register x100C-x100F.

```
SiTCP_RBCP$ wrw x100c 133333
```

The value is trigger period in clocks (133MHz). When you set 133333, the frequency is 133MHz * 133333 clocks = 1 kHz.

Read Depth

Readout window is written in the register x1090-x1091. The default value is 100 (100 nsec for 1GSPs). You should take care about where the pulse is in the window. The pulse timing (i.e. when to stop sampling after receiving trigger) is written in the register x1092-x1093. For example:

```
SiTCP_RBCP$ wrs x1090 100
SiTCP_RBCP$ wrs x1092 512
```

This is roughly appropriate timing for 1GHz sampling and locally generated triggers. The number depends on how long it takes until the trigger arrives. Software needs to be compiled again if you change read depth from 100. Write the new value in Config.h.

```
#ifndef READDEPTH
    #define READDEPTH 100
#endif
```

Sampling Frequency

Default Sampling frequency is 1GSPs. The value is written in the register x1094. To change Sampling frequency to 2GSPs:

```
SiTCP_RBCP$ wrb x1094 34
```

For the other control parameters, see the register map in section 4.

3 Install the Firmware

You can install the firmware via the download cable offered by Xilinx. There are two different ways to install the firmware. One is downloading the file directly to the FPGA via JTAG chain. The other is writing the file to the SPI flash (PROM, non-volatile memory) through the FPGA. Downloading via JTAG is much faster than writing to the flash but FPGA forget the firmware once it is switched off (volatile). Thus JTAG download is for the debugging purpose.

3.1 JTAG Download to the FPGA

Connect the download cable to the Dragon (Figure 10) and launch the impact software offered by Xilinx. **If you are using the analog backplane ver. 3, you can also connect the download cable from the backplane to configure the Dragon FPGA.** After detecting the target devices you will see the window like Figure 11 left. Right click the FPGA, assign a configuration file (bit file), right click again and program the FPGA. On the Dragon you can see the PROG_B switch. If you press this, FPGA loses the firmware downloaded and starts to reload the firmware stored in the PROM.

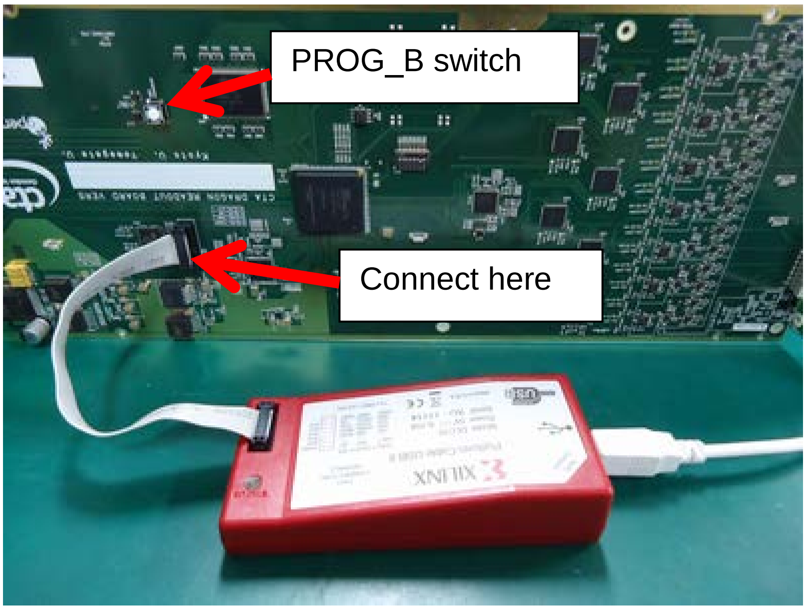


Figure 10: Connecting the Xilinx download cable

3.2 Write to the SPI PROM

Connect the download cable to the Dragon and launch the impact software offered by Xilinx. After detecting the target devices you will see the window like Figure 11 left. Right click the “FLASH” and assign new configuration file (mcs). Then you will see the window like Figure 11 right. Select the “SPI PROM”, “M25P64” and “1” like the screenshot. Finally right click the “FLASH” again and program the FLASH. After programming, you have to switch off the Dragon once for the FPGA to load the new firmware from the PROM.

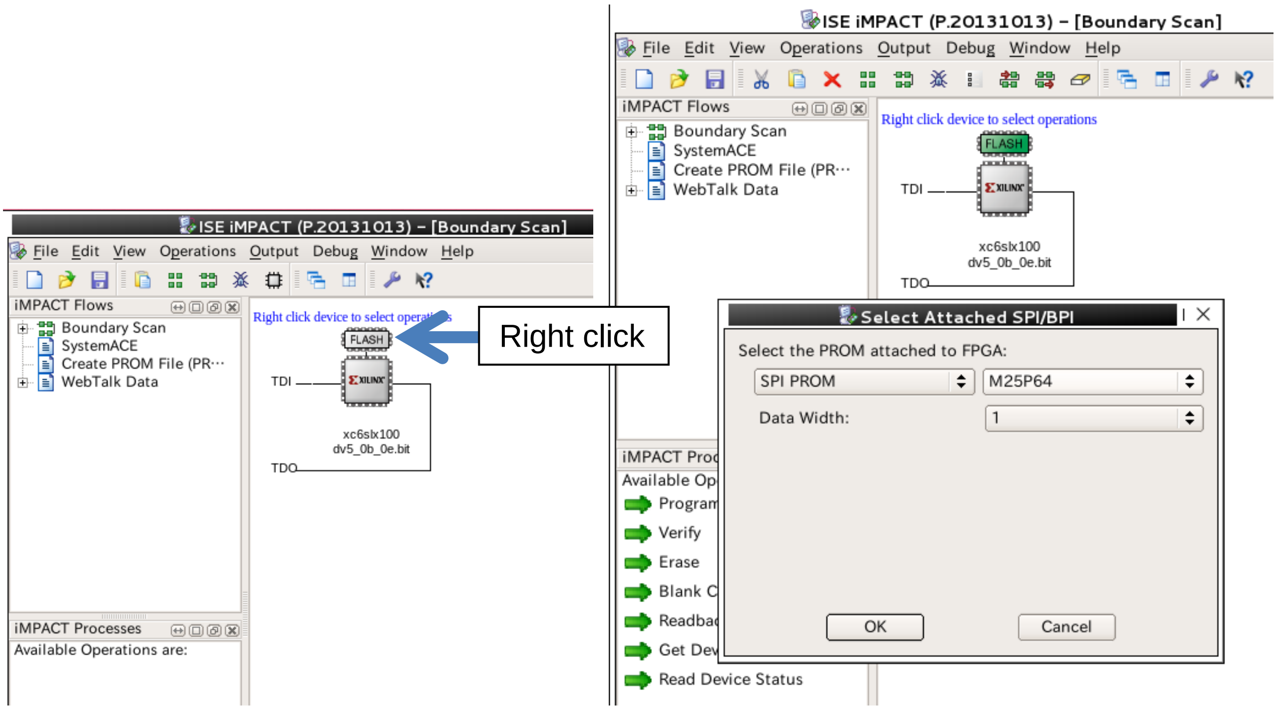


Figure 11: impact window to install the firmware to the SPI flash

3.3 *Write to the SPI PROM through the Ethernet (Remote Configuration)*

You can configure the FPGA from the Ethernet. However **this program has a bug and sometimes it fails to configure the FPGA at the moment.** In this case the PROM loses the firmware before trying the remote configuration and you have to download the firmware via the download cable.

To execute the remote configuration, go to the DominoSoft/FPGA-Config. Put the mcs file to the mcsfile_folder directory and modify the config_file.txt located in the config_folder directory. Then run the python script fpga_configuration.py and the remote configuration will starts.

```
[user@host DominoSoft]$ cd FPGA-Config
[user@host FPGA-Config]$ cp /home/user/tmp/dv5_1_0b_0e.mcs mcsfile_folder
[user@host FPGA-Config]$ cat config-folder/config-file.txt
dv5_1_0b_0e.mcs              <- set the file name of the mcs file
192.168.1.16                 <- set the IP address of the Dragon
4660                         <- port number for rbcp (always 4660)
[user@host FPGA-Config]$ python fpga_configuration.py
```

4 Memory Map

4.1 Network Parameters (stored in EEPROM)

The network parameters stored in EEPROM can be accessed via slow control in the same way as the other slow control parameters (Table 1). **The base address is xFFFFFF00.** For example, MAC address of the Dragon is stored in xFFFFFF00+(x12 to x17) = xFFFFFF12 to xFFFFFF17. The MAC address can be used as the board ID.

Table 1: Memory map for the network parameters (stored in EEPROM)

address #base address: xFFFFFF00	parameter	Length (bits)	Description
(XFFFFFF00+) X12-X17	MAC ADDRESS	6Byte	
X18-X1B	IP ADDRESS	4Byte	

4.2 Slow Control Parameters (stored in FPGA)

Table 2 shows the usual slow control parameters. **The base address is x1000.** For example, firmware version number is stored at x1000-x1001 and firmware subversion number is stored at x1002. Since **the register map can be different for the different firmware version**, please check the map in the corresponding document version if you find the problem.

Table 2: Memory map for the slow control parameters (stored in FPGA)

address #base address: x1000	parameter	Length (bits)	Description
(X1000+) X00-X01	FIRMWARE_VER	16	x5104 = boardVer5, firmwareVer1, firmwareSubver04
X02	FIRMWARE_SUBVER	8	[7:6] b00 = firmware for analog trigger b01 = firmware for digital trigger

			[5:0] firmware subsubversion number
X03	DIP_SWITCH_READ	8	status of DIP_SWITCH
X04-X07	DEBUG_IN	32	X05-X06 : drs_c[10:0] X07 : drs_state[3:0] (read only)
X08	command_rst	8	command for system reset 0xff : reset
X09-X0A	DEBUG_PARAM	16	input parameter for debug
X0B	TRIGGER_SELECT	8	0: level1 (telescope, asynchronous) 1: local level1 (local cluster, asynchronous) 2: pedestal (self, fixed frequency) 3: external (for dummy backplane, asynchronous) 4: scb test pulse 5: level1 sync (telescope, synchronous with local clock) 6: local level1 sync (local cluster, synchronous with local clock) 7: level0 sync 8: external sync
X0C-X0F	TRIGGER_FREQ	30	trigger period for pedestal run default: 444444 (clock) = 300Hz clock frequency: 133.333MHz
X10	command_adcspi	8	0xff: sending spi command to ADC
X11	ADC_SPI_DATA	8	ADC SPI write data
X12-X13	ADC_SPI_ADDR	13	ADC SPI write address
X14-X1D			Reserved
X1E	TRIGGER_ENABLE	1	triggers start/stop 0: stop, 1: start default: 1

X1F	BUSY_STATE	1	0: not busy, 1: busy (read only)
X20	command_dtrigset	8	0xff: update threshold
X21	DTRIG_THRESHOLD_0	8	DAC for Digital trigger threshold 0ch 8bits/1.25V
X22	DTRIG_THRESHOLD_1	8	1ch
X23	DTRIG_THRESHOLD_2	8	2ch
X24	DTRIG_THRESHOLD_3	8	3ch
X25	DTRIG_THRESHOLD_4	8	4ch
X26	DTRIG_THRESHOLD_5	8	5ch
X27	DTRIG_THRESHOLD_6	8	6ch
X28-X29	IPR_0	16	IPR of ch0
X2A-X2B	IPR_1	16	IPR of ch1
X2C-X2D	IPR_2	16	IPR of ch2
X2E-X2F	IPR_3	16	IPR of ch3
X30-X31	IPR_4	16	IPR of ch4
X32-X33	IPR_5	16	IPR of ch5
X34-X35	IPR_6	16	IPR of ch6
X36-X3D			Reserved
X3E-X3F	RATE_WINDOW	16	trigger rate counting window for [msec]
X40	command_l0_sc_write	8	
X41	command_l0_sc_read	8	

X42	command_l0_reset	8	
X43	command_l0dela_set	8	
X44	command_l0dela_reset	8	
X45	command_l1_sc_write	8	
X46	command_l1_sc_read	8	
X47	command_l1_reset	8	
X48-X49	RATE_WINDOWL1	16	L1 trigger rate counting window for [msec] default: 1000
X4A-X4B	RATE_L1OUT	16	trigger rate of L1out (read)
X4C-X4D	RATE_L1OUT2	16	trigger rate of L1out2 (read)
X4E-X4F	RATE_TRIGL1	16	trigger rate of TRIGL1 (read)
X50	L0_SC_ADDRESS	7	SPI address for analog L0
X51-X52	L0_SC_DATA	16	SPI send data for analog L0
X53-X55	L0_SC_READ	24	SPI read data for analog L0 (read)
X56	L1_SC_ADDRESS	7	SPI address for analog L1
X57-X58	L1_SC_DATA	16	SPI send data for analog L1
X59-X5B	L1_SC_READ	24	SPI read data for analog L1 (read)
X5C-X5E	L0_DELAYEXPAND_DATA	24	SPI send data to L0 expander for delay adjustment
X5F			Reserved
X60	command_bp_sc_write	8	0xff: send spi command to BP
X61-X64	BP_SC_SENDDATA	32	SPI send data for BP

X65-X68	BP_SC_READ	32	SPI read data for BP (read)
X69	command_bp_fpgaprogram	8	0xff: BP FPGA Reboot
X6A-X6F			Reserved
X70	command_sramwrite	8	0xff: write data to SRAM
X71	command_sramread	8	0xff: read data to SRAM (read)
X72	command_sramzz	8	0xff: SRAM sleep mode 0x00: SRAM operate
X73-X75	SRAM_ADDR	19	address for sram write/read
X76-X79	SRAM_WRITEDATA	32	write data for SRAM
X7A	SRAM_WRITEDATAP	4	another write data for SRAM
X7B-X7E	SRAM_READDATA	32	read data for SRAM (read)
X7F	SRAM_READDATAP	4	another read data for SRAM (read)
X80	command_dacset	8	0xff : upload DAC values 0x00: upload done
X81-X82	DAC_ROFS	16	DAC for DRS4 read offset default: 16'd28835=1.1V
X83-X84	DAC_OOFS	16	DAC for DRS4 out offset default: 16'd34078=1.3V
X85-X86	DAC_BIAS	16	DAC for DRS4 bias default: 16'd18350=0.7V
X87-X88	DAC_CALP	16	DAC for DRS4 cal_p default: 16'd20971=0.8V
X89-X8A	DAC_CALN	16	DAC for DRS4 cal_n default: 16'd20971=0.8V
X8B-X8D			Reserved
X8E-X8F	DRS_DWRITE_TO_READY	11	counts until trigger-ready-state (lowering busy) after enabling DWRITE (sampling start) 1count = 15nsec

			default: 273 = 4095nsec *after starting DRS4-sampling, there doesn't exist sampled waveform until the first DRS4-loop
X90-X91	DRS_READDEPTH	11	Data read depth default: 100
X92-X93	DSR_STOP_FROM_TRIG	11	Clock from Trigger to sampling stop
X94	DRS_SAMP_FREQ	8	reference clock for DRS4 default: 6'd67=1.004GSps Sampling frequency = 66.666MHz/(x+1)*1024 if you use external 10MHz as reference, Sampling frequency = 10MHz/(x+1)*1024
X95-X98	DRS_READ_FROM_STOP	32	Clock from sampling stop to readout (for the study of charge leakage)
X99	DRS_CLKOUT_ENABLE	2	bit 1-2: clock enable 2'b01: enable arrival time reference (TAG) clock (40 MHz, generated from external clock) 2'b10: enable DRS timing calibration clock (66.667 MHz) 2'b11: enable both
X9A	DRS_PLLLCK_CHECK	8	configuration for flag-check on PLLLCK default: 8'b1111_1111 = check all DRSs
X9B	DRS_CALREAD	8	0xff: readout ch8 of DRS4 for timing calibration
X9C	DRS_CASCADENUM	8	[7:4]: DRS stop channel reset enable/disable, [3:0]: number of cascaded channels x01: w/o cascade, x02: 2ch-cascade w/o channel reset, x04: 4ch-cascade w/o channel reset, x12: 2ch-cascade w/ channel reset, x14: 4ch-cascade w/ channel reset default: x04

X9D	DRS_REFCLK_RESET	8	0xff: reset DRS reference clock -> restart with the next PPS edge
X9E	DRS_REFCLK_SELECT	8	DRS reference clock select 0: local clock 1: external clock
9F			Reserved
XA0	command_scb_spisend	8	0xff: send spi command to SCBV2
XA1	command_tp_trig	8	0xff: inject test pulse trigger
XA2-XA4			Reserved
XA5-XB5	SCB_SPICMD	136	send command for spi to SCBV2
XB6	SCB_SPILENGTH	8	command length for spi to SCBV2 [0:1byte 1:2byre 2: 3byte …]
XB7-XC6	SCB_SPIREAD	128	read data for spi to SCBV2 (read)
XC7-XCA	SCB_TP_TRIG_FREQ	30	frequency(period) of test pulse if SCB_TP_CLKSELECT == 0 1count=7.5ns if SCB_TP_CLKSELECT == 1 1count=100ns default: 444444 (300Hz)
XCB-XCC	SCB_TP_TRIG_WIDTH	16	test pulse trigger width if SCB_TP_CLKSELECT == 0 1count=7.5ns if SCB_TP_CLKSELECT == 1 1count=100ns default: 3 ((3+1)*7.5=30ns)
XCD-XCE	TRIGGER_FREQ_OFFSET	16	timing offset for test pulse / pedestal trigger
XCF	SCB_TP_CLKSELECT	8	bit1: clock select for test pulse 0:local clock 133MHz 1:external clock 10MHz for external clock synchronization: 1.disable pps at backplane

			2. write xff in x109D this stop the test pulse 3.enable pps at backplane test pulse restarts at the rising edge of the pps and all test pulses in the camera should be synchronized bit8: enable test pulse without daq 1: test pulse injection starts without necessary to start daq 0: test pulse injection starts after starting daq (default) To control pulse inject position in DRS4 ring, you should select 0 and change the position by TRIGGER_FREQ_OFFSET because the timing is determined by when daq starts and when pulse injection starts.

5 Data Format

Table 3 shows the data format which is sent from Dragon. It consists of 128bits * (DRS_READDEPTH*2+4) words. In the table DRS_READDEPTH = 1024. **Before the firmware version 5_1_05, the header is different.** In this case see the Table 4. Since the format is different, you should modify the software or use the old version software.

5.1 Header

The data header is the first 128bits * 2 words. It includes the following contents.

- Header (2 bytes)

This is simply a header. Fixed number 0xAAAA is written.

- PPS counter (2 bytes)

Dragon FPGA gets PPS (pulse per second) signal from the backplane. The PPS signal is issued by the Trigger Interface Board (TIB) and distributed via the backplanes to the all Dragons in the camera. Dragon counts this PPS signal and writes the count when it gets the trigger. This PPS counter is reset to 0 when the run starts (i.e. when the TCP/IP connection is opened).

- Ten MHz counter (4 bytes)

Dragon FPGA gets also 10 MHz clock from the backplane. The clock is issued by the TIB and distributed via the backplanes to the all Dragons in the camera. Dragon counts this 10 MHz clock and writes the count when it gets the trigger. This 10 MHz counter is reset to 0 at every PPS.

- Event counter (4 bytes)

This is the number of events since the beginning of the run. It increases continuously from “1”.

- Trigger counter (4 bytes)

This is the number of triggers Dragon gets since the beginning of the run. Dragon counts the number of triggers even during the busy state i.e. readout dead time. Thus if there is a trigger issued during the busy state, trigger counter is incremented but the event counter is not. In this case you will find the jump of the trigger number in the next event data header.

- Local 133 MHz clock counter (8 bytes)

This is the counter for the 133 MHz clocks generated from the local oscillator on the Dragon. Since this is the local clock, it is not synchronized among different Dragons like PPS and 10 MHz counter. This counter is reset at the beginning of the run.

- Data header (8 bytes)

This is a header before the DRS4 data. The fixed number 0xDDDD_DDDD_DDDD_DDDD is written.

5.2 DRS4 Data

The DRS4 data includes the flag (128 bits), the first capacitor ID (128 bits) and the ADC counts (128 bits * READDEPTH * 2).

- Flag (2 bytes * 8 DRS4 chips)

This is the flag related to the readout behaviour of the Dragon FPGA. Currently only 2 bits are used.

The FPGA reads “stop channel” from DRS4 before read the capacitors. For the internal reason of the DRS4, if the “stop capacitor” of the DRS4 is >= 767, the true stop channel is one before the stop channel read from the DRS4. In other words, the stop channel which returns the DRS4 shifts after sampling to the capacitor ID 766. The first bit (LSB) of the flag indicates if the stop (first) capacitor ID is < 767 (0 in this case) or >= 767 (1 in this case).

The second bit of the flag indicates if the RoI is within the single DRS cahnnel (0 in this case) or the RoI is between two DRS channels (1 in this case).

- The first capacitor ID (2 bytes * 8 DRS4 chips)

The Dragon FPGA reads stop channel from the DRS4 (0 - 4) and then reads stop capacitor ID (0 - 1023) simultaneously with the sampled charge in the capacitors. In this field, 1024 * (stop channel) + stop capacitor ID is written.

- ADC counts (2bytes * READDEPTH * 2)

The single DRS4 chip has 8 DRS4 channels and we use cascaded 4 DRS4 channels for 1 PMT channel. Thus one DRS4 has 2 PMT channel inputs. Since Dragon reads DRS4 channels in a multiplexed way, it reads the first PMT channel at first and then the second PMT channel. The data format is written according to this order. For example, DRS4 chip 0 has PMT high gain channel 0 and PMT high gain channel 1 as inputs and the data is written from PMT high gain channel 0 (time slice 0-readdepth) to PMT high gain channel 1 (time slice 0-readdepth). The resolution of the ADC is 12 bits but FPGA write the 12 bits count in 16 bits field (just because it is easier to treat). Thus the first 4 bits from MSB is always 0.

Table 3: Data format

<table>
<tr><td>127-112</td><td>111-96</td><td>95-80</td><td>79-64</td><td>63-48</td><td>47-32</td><td>31-16</td><td>15-0bits</td></tr>
<tr><td>Header:
0xAAAA</td><td>PPS
counter
2byte</td><td colspan="2">Ten MHz counter
4byte</td><td colspan="2">Event counter 4byte</td><td colspan="2">Trigger counter
4byte</td></tr>
<tr><td colspan="4">Local 133MHz clock counter 8byte</td><td colspan="4">Data Header:
0xDDDD_DDDD_DDDD_DDDD</td></tr>
<tr><td>flag
chip0
bit0:read
prev ch</td><td>flag
chip1</td><td>flag
chip2</td><td>flag
chip3</td><td>flag
chip4</td><td>flag
chip5</td><td>flag
chip6</td><td>flag
chip7</td></tr>
</table>

bit1:bridge							
First capacitor ID for chip0	First capacitor ID for chip1	First capacitor ID for chip2	First capacitor ID for chip3	First capacitor ID for chip4	First capacitor ID for chip5	First capacitor ID for chip6	First capacitor ID for chip7
CH0 High gain, Time slice0	CH0 Low gain, Time slice0	CH2 High gain, Time slice0	CH2 Low gain, Time slice0	CH4 High gain, Time slice0	CH4 Low gain, Time slice0	CH6 High gain, Time slice0	CH6 Low gain, Time slice0
…	…	…	…	…	…	…	…
CH0 High gain, Time slice 1023	CH0 Low gain, Time slice 1023	CH2 High gain, Time slice 1023	CH2 Low gain, Time slice 1023	CH4 High gain, Time slice 1023	CH4 Low gain, Time slice 1023	CH6 High gain, Time slice 1023	CH6 Low gain, Time slice 1023
CH1 High gain, Time slice0	CH1 Low gain, Time slice0	CH3 High gain, Time slice0	CH3 Low gain, Time slice0	CH5 High gain, Time slice0	CH5 Low gain, Time slice0	DRS _TAG High gain, Time slice0	DRS _TAG Low gain, Time slice0
…	…	…	…	…	…	…	…
CH1 High gain, Time slice 1023	CH1 Low gain, Time slice 1023	CH3 High gain, Time slice 1023	CH3 Low gain, Time slice 1023	CH5 High gain, Time slice 1023	CH5 Low gain, Time slice 1023	DRS _TAG High gain, Time slice 1023	DRS _TAG Low gain, Time slice 1023

Table 4 Data header before the firmware version 5_1_05

127-112	111-96	95-80	79-64	63-48	47-32	31-16	15-0bits
Event counter 4 bytes		Trigger counter 4 bytes		Local 133MHz clock counter 8 bytes			
flag chip0 bit0:read prev ch bit1:bridge	flag chip1	flag chip2	flag chip3	flag chip4	flag chip5	flag chip6	flag chip7

6 Design Overview

6.1 Board Layout

The board layout of the Dragon and its components are shown in Figure 12. The PCB is made with 12 layers. The slow control board is connected to the PMT side and the backplane board is connected to the other side.

The analog trigger mezzanine is connected on the back side of the Dragon (Figure 13). The mezzanine gets the PMT signals from the main amplifier and also from other neighboring Dragons and generates the trigger signals.

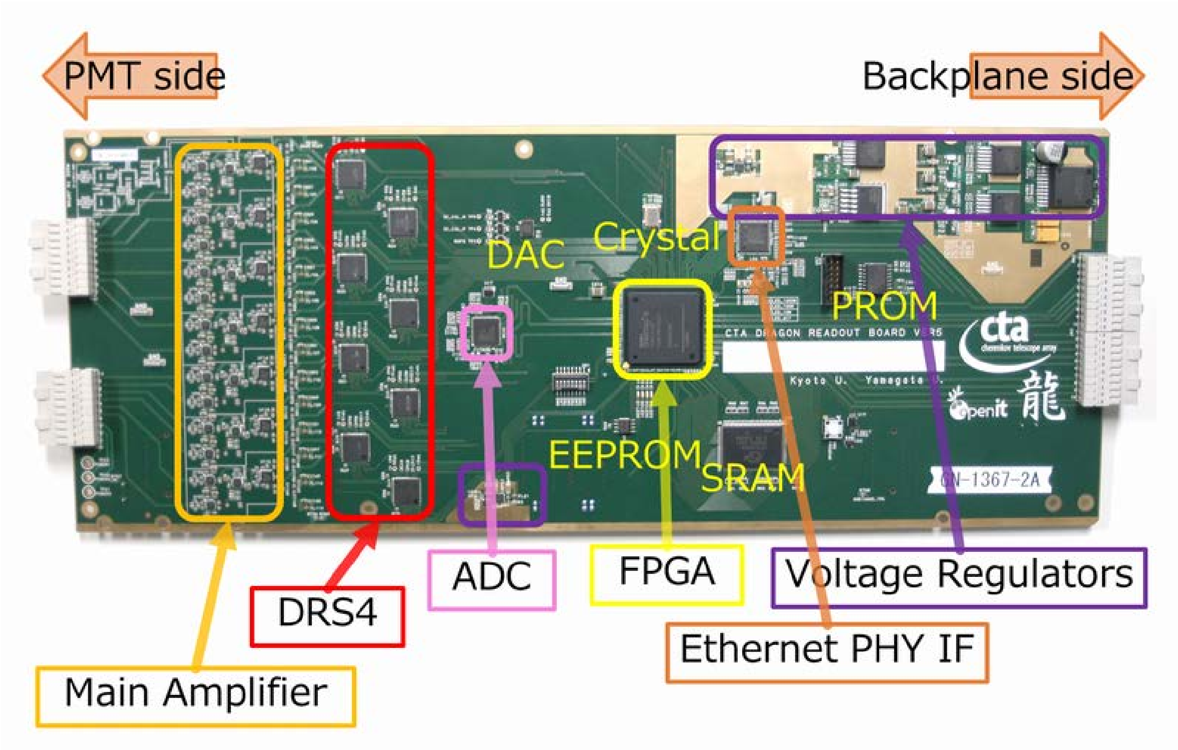


Figure 12: Front view of the Dragon ver.5

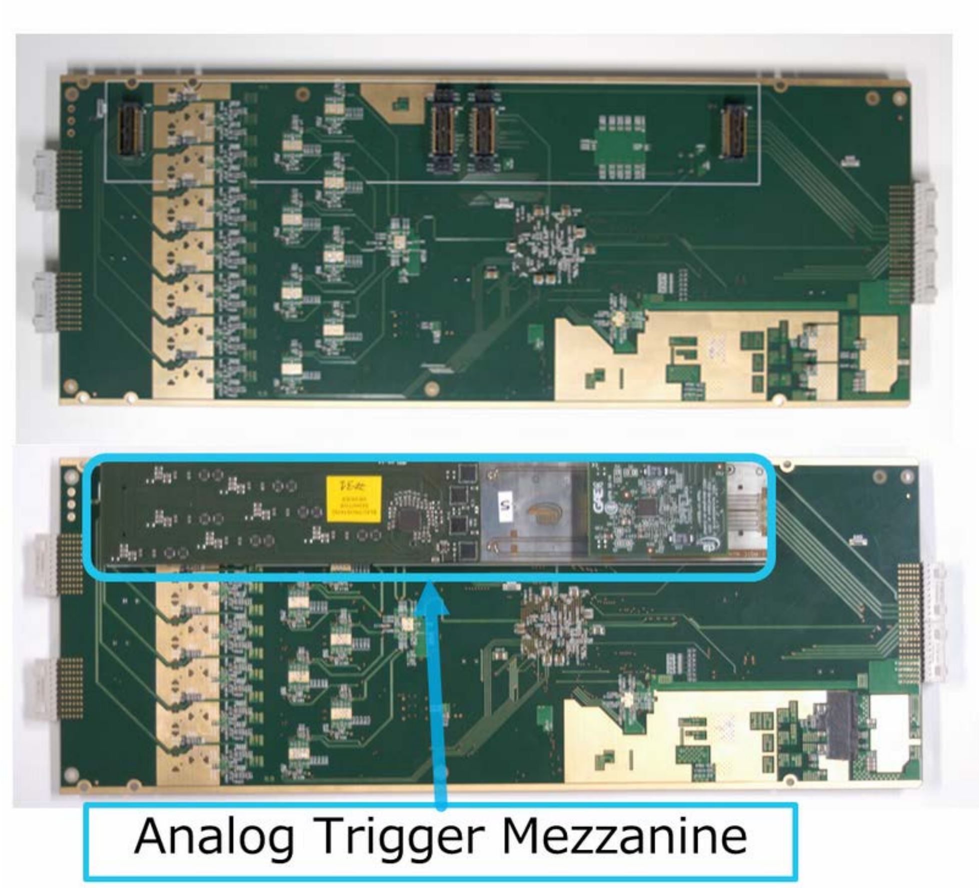


Figure 13: Back view of the Dragon ver.5 with the analog trigger mezzanine (bottom) and without the mezzanine (top)

6.2 Functional Block Diagram

Figure 14 shows the functional block diagram of the Dragon and the whole system. The main part of the Dragon is the 8 DRS4 chips which samples PMT signals with GHz frequency. The waveforms sampled by the DRS4s are digitized by the moderate-speed ADC, data-formatted by the FPGA and sent to the data server through the Gbit Ethernet.

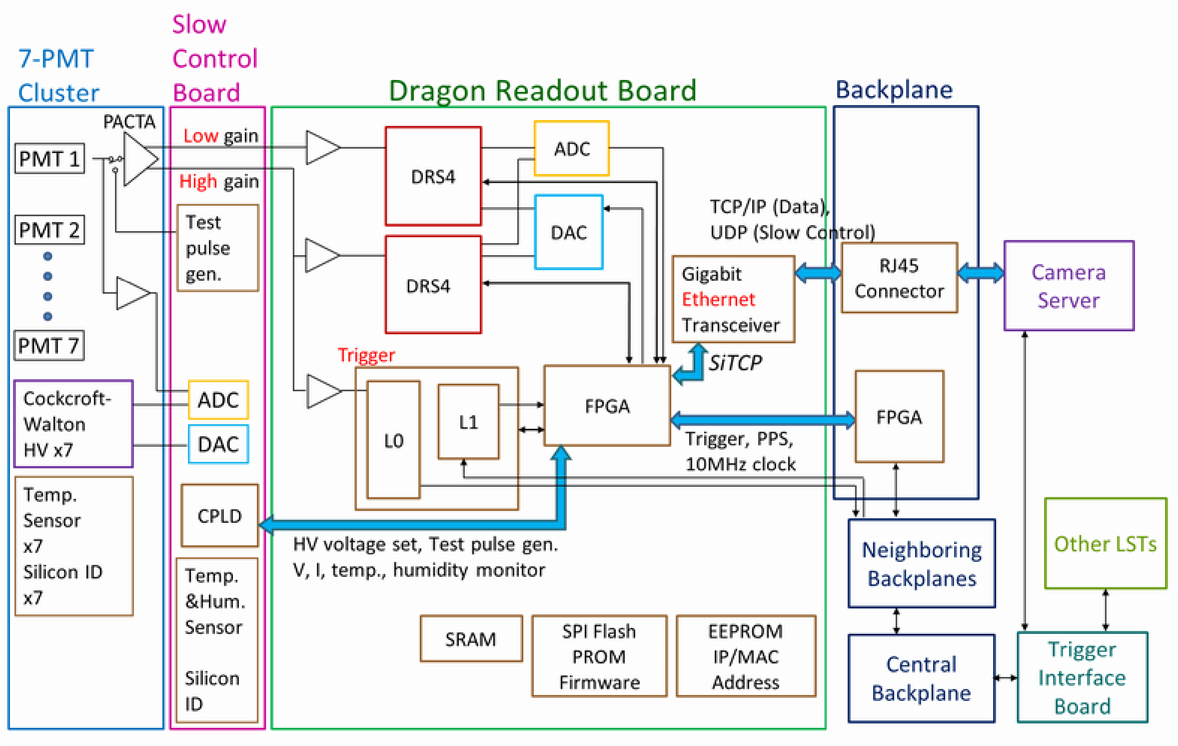


Figure 14: Functional block diagram

6.3 Voltage Regulators

Figure 15 shows the block diagram of the power supply. Dragon receives +24 V from the backplane. From this 24 V supply, the voltage regulators on the Dragon generate the various voltages necessary for the circuit (+3.3 V, -3.3 V, +2.5 V, +1.8 V and +1.2 V). The voltage regulators are the combination of the switching regulators (high efficiency but can cause switching noise) and linear regulators (low efficiency). Dragon supplies +6 V, +3.3 V and -3.3 V to the slow control board. These power supplies are also used to PMT-CW and PACTA. Ethernet magnetics needs +3.3 V and this is supplied from the Dragon to the backplane.

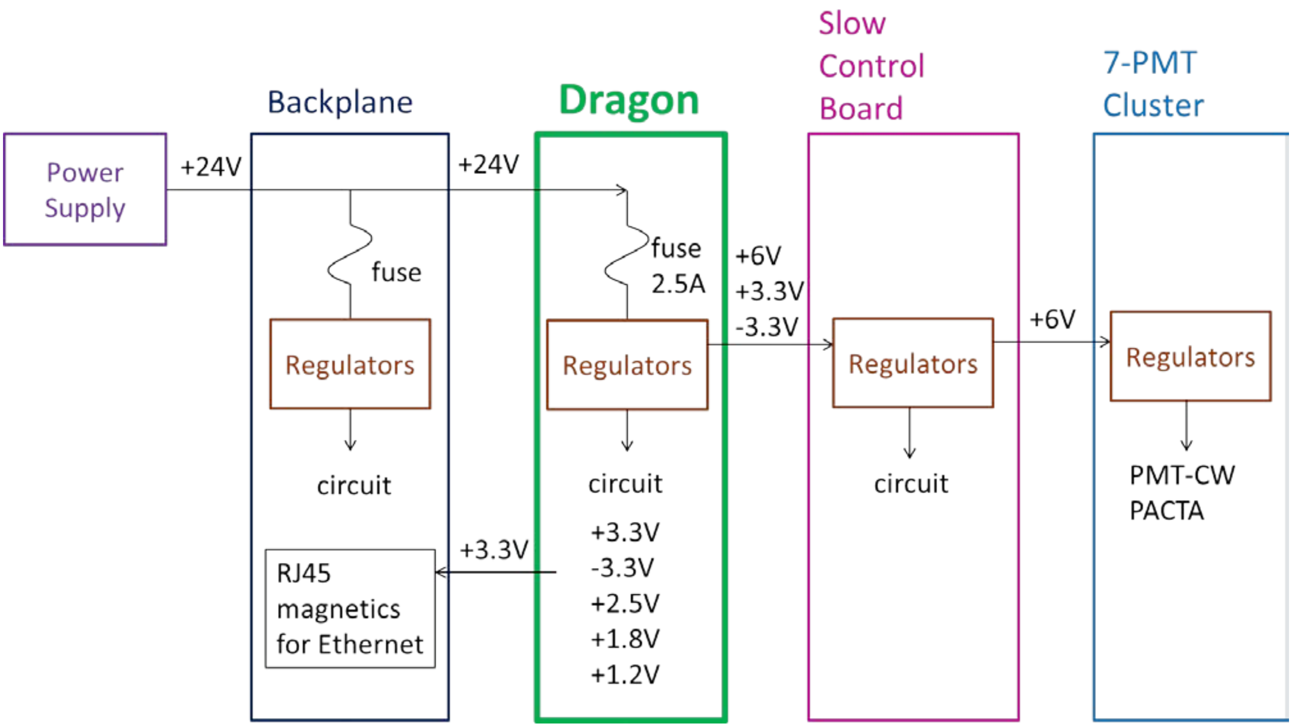


Figure 15: Block diagram of the power supply

6.4 Main Amplifier

Single Dragon module has 7 PMT modules connected on the SCB. Each PMT signal (anode output) goes into PACTA preamplifier on the PCB of the PMT module. PACTA has the bi-gain differential output (high gain, HG and low gain, LG) and both outputs go into Dragon through SCB. The first input stage on the Dragon is the main amplifier. Main amplifier is a combination of the commercial operational amplifier ADA4927 from Analog Devices and they amplify the signals from the PACTA into the appropriate range for the next stage. Here HG outputs from PACTA go into two branches, one is for the HG signal for DRS4 and the other is the signal for the trigger. LG outputs from PACTA go into LG signal for DRS4. The simplified schematics is shown in Figure 16. From the PACTA to DRS4 and finally from DRS4 to ADC, all the analog signals are differential to reject the common mode noise.

We use four cascaded DRS4 channels for the single PMT HG signals and the single PMT LG signals respectively. Because more cascaded DRS4 channels decrease the bandwidth of the signal due to larger input capacitance, two ADA4927 outputs are used for the HG DRS4 to improve the bandwidth. The dual channel chip ADA4927-2 is used. Each output is connected to two DRS4 channels.

PACTA is a transimpedance amplifier, which converts input current to output voltage according to Vout = Gti * Iin. Here Gti is the transimpedance gain and Gti for the HG is 1200 ohm and for the

LG is 80 ohm. Let's estimate the voltage swing of the signal. If we assume that the PMT signal is the triangle shape with the FWHM of 3 nsec (this is not very precise but rough estimation), the output current height (pk-pk) for the single photoelectron signal is derived like this:
3 nsec * Ipmt-pk = 4*10^4 (PMT gain) * 1.6*10^-19 (elementary charge).
Ipmt-pk = 2.13 uA/p.e..
Here we assumed that the PMT gain is 4*10^4, which is the nominal operation gain of the LST.
Then the output voltage height (pk-pk) from the PACTA is:
VpactaHG-pk =1200 * 2.13 uA/p.e. = 2.56 mV/p.e. (HG).
VpactaLG-pk = 80 * 2.13 uA/p.e. = 0.17 mV/p.e. (LG).

The gain of the main amplifier can be adjusted by the feedback resistor values of the ADA4927. The gain of the amplifier is adjusted to match the input range of the DRS4 and thanks to the bi-gain configuration Dragon achieves wider dynamic range. The input range of the DRS4 is 1 V and can be shifted by the external voltage from DAC. In the default configuration we operate DRS4 with the input range from -50 mV to +950 mV. The gain configuration for each branch is as follows:
HG x5.23, ~13.4 mV/p.e..
LG x4.02, ~0.68 mV/p.e..
Trigger x18.7, ~47.9mV/p.e..

Note that another DAC output is connected before the input of the DRS4 to supply the DC offset voltage to both of the differential lines. This DC voltage is necessary to shift the input common mode level to the optimum range of the DRS4. Since the same DC voltage is given to both of the differential lines, it does not change the level of the differential signal. However this DC offset part includes several resistors to separate the DC level and the AC signal. Due to this resistor divider, the signal amplitude is reduced by factor ~0.6. Finally,
HG ~8.04 mV/p.e., pulse height is saturated at ~118 p.e..
LG ~0.41 mV/p.e., pulse height is saturated at ~2330 p.e.

Again this is rough estimation because the pulse height depends on pulse shape and bandwidth. In addition, even if the pulse height is saturated still the integrated charge does not saturate. Thus the actual performance should be obtained by the measurements. Figure 17 and Figure 18 shows the dynamic range and bandwidth measurements respectively. From the measurements, the gain of the HG and the LG is *5.96 and *4.85 respectively. These values include the DRS4 output gain of *2. The bandwidth of the HG and LG is 350 MHz and 160 MHz at -3 dB respectively. Bandwidth of the LG is not very important because the signal is larger compared to the NSB level. The technical reason for the worse bandwidth of the LG is usage of the inner layer for the signal path due to high integrity and thus larger parasitic capacitance. Also more cascaded DRS4 channels for the single ADA4927 decreases the bandwidth.

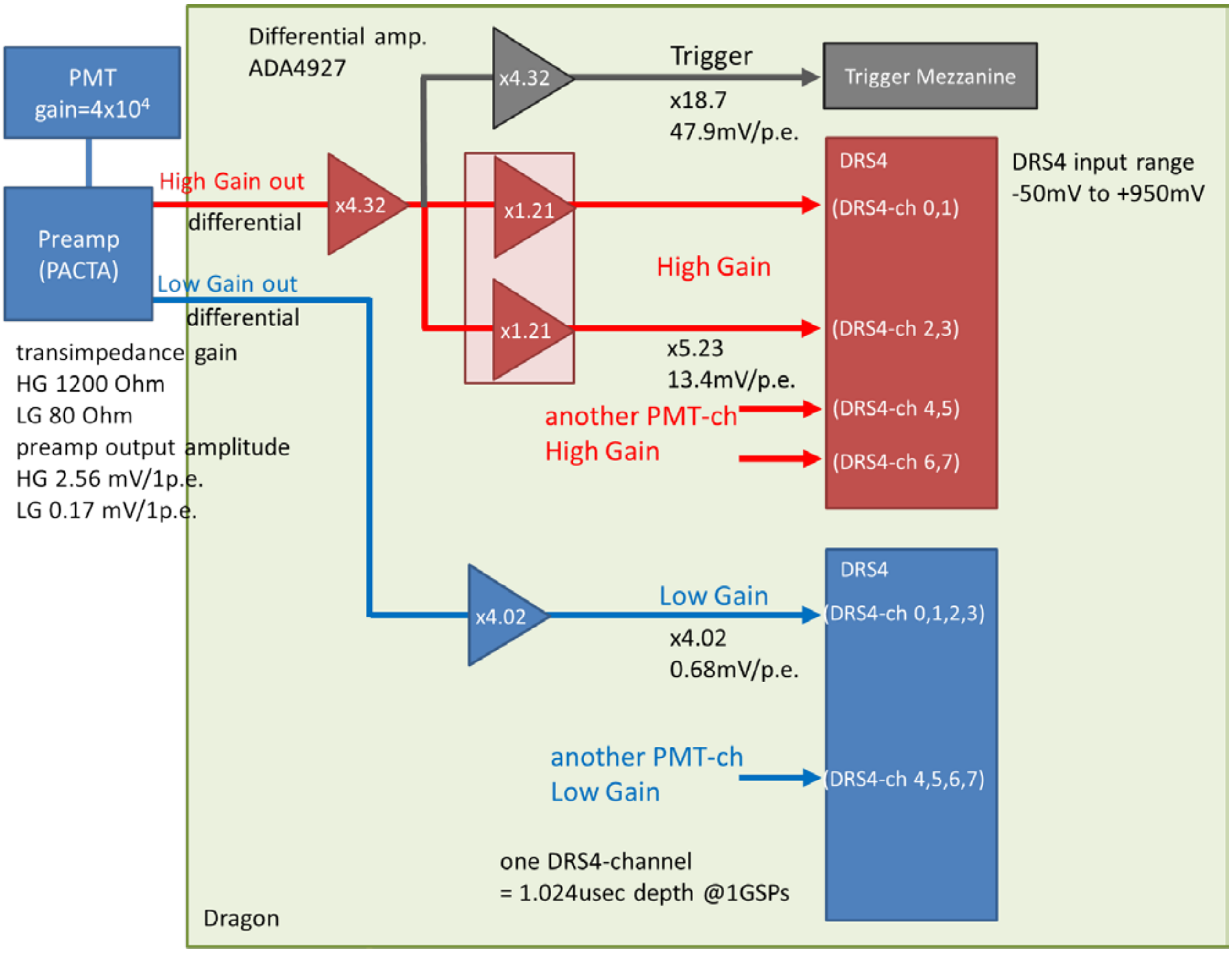


Figure 16: Block diagram of the main amplifier

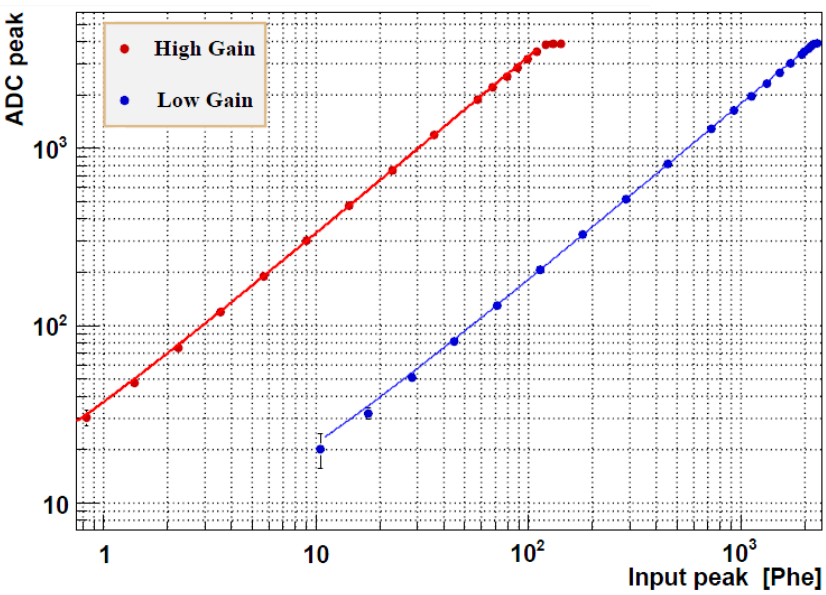


Figure 17: Dynamic range measurement of the Dragon

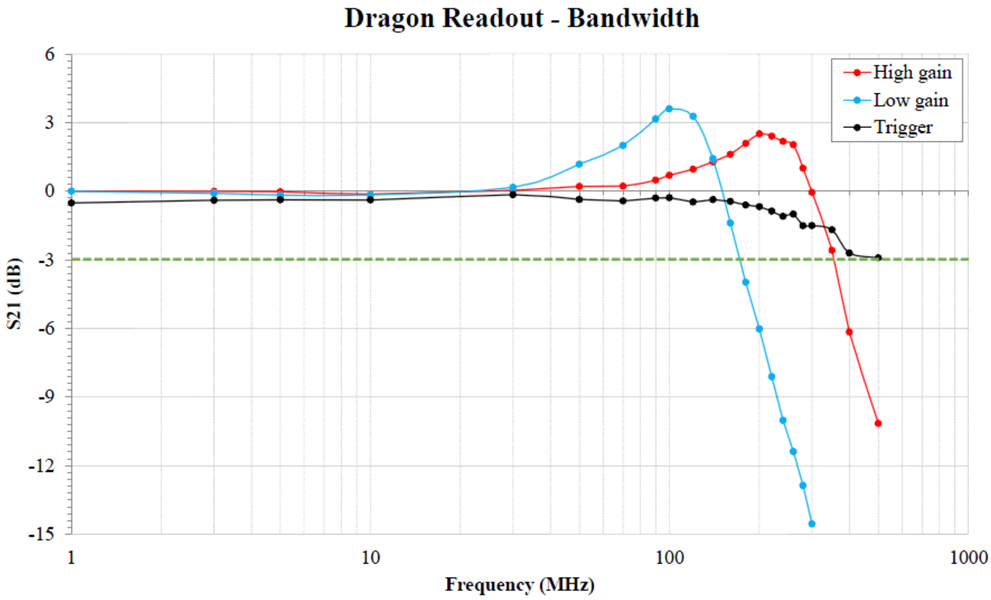


Figure 18: Bandwidth measurement of the Dragon