Nuclear Physics Division
Fast Electronics Group

Firmware for VXS Crate Trigger Processor Version 2 (CTPV2) Module

9 July 2013

Hai Dong

Table of Contents

1.0 Introduction

The firmware for the Crate Trigger Module (CTP) is written in VHDL. The CTP module has two XC5VLV50 (U1, U3) that has same VHDL code and one XC5VLV110 (U24) that has its own VHDL code. LV50 VHDL code receives data from five FADC, aligns the data, and sends the data in sync with system clock to the LV110. The LV110 receives data from six FADC and both LV50, aligns the data, processes the data, and sends the result to Fiber located at the front panel.

2.0 FX70T (U1,U3) VHDL Code

2.1 **Receive data from FADC**

The VHDL code receives data from five FADC using Xilinx Aurora operating at 2.5Gbits, aligns the data, and sends the data 80 bits synchronized with 250MHz clock. The Aurora protocol provides control and interface signals: Lane_up, Channel_up, Rx_src_rdy_n, Tx_src_rdy_n, Tx_dst_rdy_n. The functions for these signals are as follow:

- Lane_up : indicates FADC MGT and CTP GTP are able to send and receiving data from each other.
- Channel_up: indicates the data from the lanes if more than one are in alignment.
- Tx_dst_rdy_n: indicates that the transmitter is ready to send data
- Tx_src_rdy_n: tell the transmitter to send user data.
- Rx_src_rdy_n: indicates that the receiver has user data

2.2 **Aligning Data**

In the 12 Gev Trigger System, Sync is distributed in time to all sixteen FADC250V2 and CTP modules in a crate. When Sync goes high, FADC250 stops sending data and CTP reset all its circuitries and buffers. Sync has to be high for at least 500 nS to allow the MGT to completely flush its buffers. When Sync goes low, the FADC250 send value 2 for one 16 bits and value 1 for the other 16 bits on the 2 MGT lanes 0 and 1. Values 2 and 1 are sent 3 times. Which value is sent on lane 0 and 1 is depend on when the Sync arrived at the FADC250. The CTP uses these values to align data from all 16 FADC250 in time and words order. After aligning the data from 5 FADC250, U1 sends data to U24 via 80 LVDS lines. U3 does the same as U1. U24 aligns the data from 6 FADC250, wait for data from U1 and U3, processes, and sends data to SSP through fiber. First Word Fall Through FIFO is used to used to indicate when first values 2 and 1 are received.

2.3 **Demultiplexed Data to VLX110 (U24)**

The 32-bits data from each FADC is de-multiplexed into two 16-bits words with bits 31-16 send to U24 first. The 16-bits word is registered (F1_F3_D) and sends to U24 along with DatRdy. After Sync goes low, DatRdy goes high along with all values 2 output. DatRdy goes low when Sync goes high. The 250 MHz input clock is routed inside the FPGA to output pins (CLK_FPGA1_3). The LV110 uses CLK_FPGA1_3 to capture F1_F3_D and Datrdy.

2.4 **Disable FADC**

Data from FADC can be disabled by writing to Configuration register via I-square-C interface. When disabled, the code does not wait for data from that FADC250. The bits corresponded to that FADC are zero going.

2.5 **Processing Delay**

2.6 **Xilinx System Monitor.**

Xilinx System Monitor feature that indicates the die temperature and VCCINT and VCCAUX Supply is included. The data is mapped into register file that can be read back through I-Square-C. The host has to convert binary data to temperature and voltage (See Xilinx UG192).

3.0 FX100T (U24) VHDL code

3.1 **Receive data from FADC**

The procedure for receiving data from six FADC is identical to VLX50.

3.2 **Receive data from VLX50**

When Datrdy is high, 80 bits data is clocked into 512_80 FIFO. Five-hundred words deep can handle 2uS skewing between the two VLX50. The number of clock from Datrdy going high to data FIFO empty going low is 8 (32 ns).

3.3 **Aligning Data.**

Aligning data from six FADC250 is the same as U1 and U3. When Datrdy from U1 goes high, U24 allows words from U1 to be written into First-Word-Fall-Through FIFO. It does the same with U3. When words values of 2 are at the output of these FIFO, the FIFO are read at the same time.

3.4 **Summing Data.**

The data is de-multiplexed and add in stages as shown below. The stages are used to make 250 MHz clocking using Xilinx FPGA lowest cost (speed grade 1).

3.4.1 First stage

- 1^{st}_A0 = VLX50 #1 bit (15..0), (31..16), (47..32)
- 1^{st}_A1 = VLX50 #1 bit (63..48), (79..64)
- 2^{nd}_A0 = VLX50 #2 bit (15..0), (31..16), (47..32)
- 2^{nd}_A1 = VLX50 #2 bit (63..48), (79..64)

2.1.2 Second stage

- 1^{st}_B = 1^{st}_A0 + 1^{st}_A1
- 2^{nd}_B = 2^{nd}_A0 + 2^{nd}_A1
- SUM_A = FADC_0 + FADC_1 + FADC_2
- SUM_B = FADC_3 + FADC_4 + FADC_5

2.1.3 Third stage

- 1^{st}_2^{nd} = 1^{st}_B + 2^{nd}_B
- SUMA_B = SUMA + SUMB

2.1.4 Fourth stage

- Final_SUM = 1^{st}_2^{nd} + SUMA_B

3.5 **Multiplexing Final SUM**

In the CTP the FIBER data consisted of 4 Aurora Lanes operating at 2.5Gibits. Each lane provides 16bits for a total of 64-bits every 8 NS. Since the 20-bits Final Sum is occurring at 4 NS, two Final Sum is packed (multiplexed) into one transmission. Final Sum # even is packed into bits 39 to 30 while Final Sum # odd is packed into bits 19 to 0. As of this date 03/10/09 bits 63..40 are set to zeroes.

3.6 **Disable FADC**

An FADC can be Data from FADC can be disabled by writing to Configuration register via I-square-C interface. When disabled, the code does not wait for data from that FADC. The bits corresponded to that FADC are zero in the summing process.

3.7 **Processing Delay**

The number of clock from Datrdy (from VLX50) going high to FIFO empty going low is 8 clock (32 nS). Adder stages take 11 clock (44 ns).

3.8 **Threshold**

A register accessed through I-Square-C provides a threshold for the final sum . When the final sum is above the threshold, Trigger Output at the front panel is active high. If the History Buffer described below is armed, 256 additional sums are stored and ready to be read out.

3.9 **History Buffer**

The History Buffer stores 512 Final Sum values. Halves of the values are before Final Sum cross the threshold and halves are after. It is controlled and read out via I-Square-C. The steps to use the history buffer are as follow:

- Host sets ARM to one.
- Host sets ARM to zero.
- After 256 Final Sum are stored in Memory, waiting for Final Sum to go above threshold. Continues storing Final Sum
- When Final Sum reach threshold. 256 more values are stored.
- Stops storing.
- Host polled for DataReady for high.
- Read as much values as host wanted. Data read back is auto increment.
- Host sets ARM to clear DataReady.

3.10 **Automate Testing.**

The integrity of the data transfer from FADC250 to CTP can be verified. When bit 1 of MGT_CTRL register in FADC250 is 0, counting sequences (0,1,2,3,4, etc,). The CTP adds the data from all enable FADC250 and compare to expected sum. For example, if 8 FADC250 are enable, the expected Sum are 0, 8, 16, 24, 32, etc. If at any time the actual Sum and the expected Sum are not equaled, ERROR_LATCH_FS variable goes high and stay high until Sync goes high. To use this feature:

- Make bit 1 of MGT_CTRL register in FADC250 zero.
- Bring Sync high for at least 500 nS
- Bring Sync low.
- Wait for however long the testing is desire.
- Read Is2 status register bit ERROR_LATCH_FS. If it is zero the data from the FADC250 are OK.
- Make bit 1 of of MGT_CTRL register in FADC250 one. ERROR_LATCH_FS should be one because data from FADC250 is not counting sequence.

3.11 **Xilinx System Monitor**

Xilinx System Monitor feature that indicates the die temperature and VCCINT and VCCAUX Supply is included. The data is mapped into register file that can be read

back through I-Square-C. The host has to convert binary data to temperature and voltage (See Xilinx UG192).

4.0 I-Square-C interface

Data is written to and read from the CTP serially from VXS P1 connector. The serial interface consists of an input clock bit and a bidirectional data bit that connected to all three FPGA. The FPGA de-serializes the data bits and listens for its address. The FPGA have non-overlapping address and responses when it is addressed. Each Register is 16 bits wide.

5.0 Total Processing Delay

It takes a total of 27 clocks (8 from VLX50, 8 to receive data from VLX50, 11 from adder stages) from the time data arrived from the slowest FADC to produce the first Final Sum.

6.0 LED

LED(0) <= HEART_BEAT_250 ;
LED(1) <= HEART_BEAT_200;
LED(2) <= ABOVE_THREDSHOLD_BUF_Q(0); -- ; Heartbeat 200
LED(3) <= indicate all MGT lanes and all MGT Channels of U24 are OK.
LED(4) <= HEART_BEAT_INIT;
LED(5) <= Indicate there is not Fiber Fault and Fiber is ready---
LED(6) <= '1';

7.0 SD Test Code

7.1 Count Rising Edge of Sync, Trig1, Trig2 from SD.
7.2 Measure Clock from SD (should be 250 MHz)
7.3 Connect SD_IN_SPARE 0 to SD_OUT_SPARE 0
7.4 Connect SD_IN_SPARE 1 to SD_OUT_SPARE 1

8.0 Configuring FPGA from I-SQUARE-C Bus

This feature allows FPGA configuration data to be downloaded from I-Square-C and stored into one of the three Configuration ROMs which are AT45DB642. U1, U3, and U24 have a separate Configuration ROM. This feature also allows rebooting (reloading) the FPGA. **CTPV2 FpgaConfigExamples.C shows the routines. When writing C code the first time, it is recommend to write and verify the C Code for the** Read FPGA Config Data from ROM to SRAM. This is to minimize the mistake of continuously Erasing and Programming the ROM. ROM is rated for 100,000 erases/writes cycles.

8.1 The overall steps are (details steps follow):

8.1.1 Host issues Erase Commands to U1 to erase U1, U3, or U24.

8.1.2 Host downloads the entire FPGA Configuration Data (MCS file) for U1, U3, or U24's Configuration ROM to U1. U1 stores the data on onboard SRAM

8.1.3 Host read back data from SRAM for verification.

8.1.4 Host issues command to U1 to store data from SRAM into U1, U3, or U24's Configuration. ROM. Host wait until U1 finishes storing data to ROM

8.1.5 Host issues command to U1 to read Configuration ROM of U1, U3, or U24 to SRAM. Host wait for U1 to finish reading

8.1.6 Host reads data from SRAM and verifies that data is corrected. If data does not match the MCS file, host can redo steps 8.1.1 to 8.1.6.

8.1.7 If desire (not necessary) host can repeat steps 8.1.1 to 8.1.6 for the remaining FPGA.

8.1.8 Host issues command to U3 to reboot ALL FPGA. There is no command to reboot individual FPGA.

8.2 **Erase Command** (See Register File) Make sure U1 Register 12 bit 15 is a one:

8.2.1 For U1

8.2.1.1 Initialize variable Config_ROM_Block_Number_to_Erase to zero.

8.2.1.2 Write Config_ROM_Block_Number_to_Erase to U1 Register 7 bit 9..0

8.2.1.3 Write the following bits values to U1 Register 6 to start Erase U1 Rom

8.2.1.3.1 10..9 = **01**

8.2.1.3.2 8 = 1

8.2.1.3.3 7 = 0

8.2.1.3.4 6..3 = **1001**

8.2.1.4 Poll U1 Register 12 bit 15 until it is a one. Fpga Config is done erasing Config_ROM_Block_Number_to_Erase.

8.2.1.5 Write the following bits values to U1 Register 6 to ready for next command

8.2.1.5.1 10..9 = **01**

8.2.1.5.2 8 = **0**

8.2.1.5.3 7 = 0

8.2.1.5.4 6..3 = **1001**

8.2.1.6 Increment Config_ROM_Block_Number_to_Erase

8.2.1.7 WAIT for 220 milliSecond. This is the time the ROM takes to Erase one block.

8.2.1.8 Repeat 8.2.1.2 to 8.2.1.6 for Config_ROM_Block_Number_to_Erase from 1 to 1023 to erase all 1024 ROM's Blocks.

8.2.2 For U3

8.2.2.1 Initialize variable Config_ROM_Block_Number_to_Erase to zero.

8.2.2.2 Write Config_ROM_Block_Number_to_Erase to U1 Register 7 bit 9..0

8.2.2.3 Write the following bits values to U1 Register 6 to start Erase U1 Rom

8.2.2.3.1 10..9 = **10**
8.2.2.3.2 8 = 1
8.2.2.3.3 7 = 0
8.2.2.3.4 6..3 = **1010**

8.2.2.4 Poll U1 Register 12 bit 15 until it is a one. Fpga Config is done erasing Config_ROM_Block_Number_to_Erase.

8.2.2.5 Write the following bits values to U1 Register 6 to ready for next command

8.2.2.5.1 10..9 = **10**
8.2.2.5.2 8 = **0**
8.2.2.5.3 7 = 0
8.2.2.5.4 6..3 = **1010**

8.2.2.6 Increment Config_ROM_Block_Number_to_Erase.

8.2.2.7 Repeat 8.2.2.2 to 8.2.2.6 for Config_ROM_Block_Number_to_Erase from 1 to 1023 to erase all 1024 ROM's Blocks.

8.2.3 For U24

8.2.3.1 Initialize variable Config_ROM_Block_Number_to_Erase to zero.

8.2.3.2 Write Config_ROM_Block_Number_to_Erase to U1 Register 7 bit 9..0

8.2.3.3 Write the following bits values to U1 Register 6 to start Erase U1 Rom

8.2.3.3.1 10..9 = **11**
8.2.3.3.2 8 = 1
8.2.3.3.3 7 = 0
8.2.3.3.4 6..3 = **1011**

8.2.3.4 Poll U1 Register 12 bit 15 until it is a one. Fpga Config is done erasing Config_ROM_Block_Number_to_Erase.

8.2.3.5 Write the following bits values to U1 Register 6 to ready for next command

8.2.3.5.1 10..9 = **11**
8.2.3.5.2 8 = **0**
8.2.3.5.3 7 = 0
8.2.3.5.4 6..3 = **1011**

8.2.3.6 Increment Config_ROM_Block_Number_to_Erase

8.2.3.7 Repeat 8.2.3.2 to 8.2.3.6 for Config_ROM_Block_Number_to_Erase from 1 to 1023 to erase all 1024 ROM's Blocks.

8.3 Download FPGA Config Data to SRAM (See Register File) Make sure U1 Register 12 bit 15 is a one:

8.3.1 For U1, U3, U24

8.3.1.1 Write the following bits values to U1 Register 6 to select SRAM for writing.

8.3.1.1.1 10..9 = 00
8.3.1.1.2 8 = 0
8.3.1.1.3 7 = **1**
8.3.1.1.4 6..3 = 0000

8.3.1.2 Initialize 22 bits variable SRAM_Address to 0

8.3.1.3 Read 2 bytes from Xilinx MCS data bytes to variable SRAM_DATA. The first byte read go to bits 15..8 and the second byte read go to bits 7..0.

8.3.1.4 Write SRAM_DATA to U1 Register 8

8.3.1.5 Write bits 15..0 of SRAM_Address to U1 Register 9.

8.3.1.6 Write the following bits values to U1 Register 10 to write SRAM_Data to SRAM_Address.

8.3.1.6.1 15 = 1
8.3.1.6.2 14 = 0
8.3.1.6.3 5..0 = SRAM_Address bits 21..16

8.3.1.7 Write the following bits values to U1 Register 10 to ready for the next write.

8.3.1.7.1 15 = 0
8.3.1.7.2 14 = 0
8.3.1.7.3 5..0 = SRAM_Address bits 21..16

8.3.1.8 Increment SRAM Address

8.3.1.9 Repeat 8.3.1.2 to 8.3.1.8 for remaining bytes in MCS file

8.4 ReadBack FPGA Config Data to SRAM (See Register File) Make sure U1 Register 12 bit 15 is a one:

8.4.1 For U1, U3, U24

8.4.1.1 Write the following bits values to U1 Register 6 to select SRAM for reading.

8.4.1.1.1 10..9 = 00
8.4.1.1.2 8 = 0
8.4.1.1.3 7 = **1**
8.4.1.1.4 6..3 = 0000

8.4.1.2 Initialize 22 bits variable SRAM_Address to 0

8.4.1.3 Read 2 bytes from Xilinx MCS data bytes to variable SRAM_DATA. The first byte read go to bits 15..8 and the second byte read go to bits 7..0.

8.4.1.4 Write bits 15..0 of SRAM_Address to U1 Register 9.

8.4.1.5 Write the following bits values to U1 Register 10 to read SRAM_Data from SRAM_Address.

8.4.1.5.1 15 = 0
8.4.1.5.2 14 = **1**
8.4.1.5.3 5..0 = SRAM_Address bits 21..16

8.4.1.6 Poll U1 Register 12 bit 15 until it is a one.

8.4.1.7 Write the following bits values to U1 Register 10 to ready for the next write.

8.4.1.7.1 15 = 0
8.4.1.7.2 14 = 0

8.4.1.7.3 5..0 = SRAM_Address bits 21..16

8.4.1.8 Read Data From SRAM at U1 Register 11

8.4.1.9 Compare Data From SRAM (8.4.1.8) to SRAM_DATA (8.4.1.3)

8.4.1.10 Increment SRAM Address

8.4.1.11 Repeat 8.3.1.2 to 8.3.1.8 for remaining bytes in MCS file

8.5 Program FPGA Config Data from SRAM to ROM (See Register File) Make sure U1 Register 12 bit 15 is a one:

8.5.1 For U1

8.5.1.1 Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM

8.5.1.1.1 10..9 = 01

8.5.1.1.2 8 = 1

8.5.1.1.3 7 = 0

8.5.1.1.4 6..3 = **0000**

8.5.1.2 Poll U1 Register 12 bit 15 until it is a one. One indicates that Fpga Config is done storing Config Data to ROM. Opcodes 0,1,2 can only be issued at 10 minutes interval to prevent exceeding the ROM (AT45DB642) maximum write of 100,000 times.

8.5.1.3 Write the following bits values to U1 Register 6 to ready for next command

8.5.1.3.1 10..9 = 01

8.5.1.3.2 8 = 0

8.5.1.3.3 7 = 0

8.5.1.3.4 6..3 = **0000**

8.5.2 For U3

8.5.2.1 Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM

8.5.2.1.1 10..9 = **10**

8.5.2.1.2 8 = 1

8.5.2.1.3 7 = 0

8.5.2.1.4 6..3 = **0001**

8.5.2.2 Poll U1 Register 12 bit 15 until it is a one. One indicates that Fpga Config is done storing Config Data to ROM. Opcodes 0,1,2 can only be issued at 10 minutes interval to prevent exceeding the ROM (AT45DB642) maximum write of 100,000 times.

8.5.2.3 Write the following bits values to U1 Register 6 to ready for next command

8.5.2.3.1 10..9 = 10

8.5.2.3.2 8 = 0

8.5.2.3.3 7 = 0

8.5.2.3.4 6..3 = **0001**

8.5.3 For U24

8.5.3.1 Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM

8.5.3.1.1 10..9 = **11**
8.5.3.1.2 8 = 1
8.5.3.1.3 7 = 0
8.5.3.1.4 6..3 = **0010**

8.5.3.2 Poll U1 Register 12 bit 15 until it is a one. One indicates that Fpga Config is done storing Config Data to ROM. Opcodes 0,1,2 can only be issued at 10 minutes interval to prevent exceeding the ROM (AT45DB642) maximum write of 100,000 times.

8.5.3.3 Write the following bits values to U1 Register 6 to ready for next command

8.5.3.3.1 10..9 = 11
8.5.3.3.2 8 = 0
8.5.3.3.3 7 = 0
8.5.3.3.4 6..3 = **0011**

8.6 Read FPGA Config Data from ROM to SRAM (See Register File) Make sure U1 Register 12 bit 15 is a one:

8.6.1 For U1

8.6.1.1 Write the following bits values to U1 Register 6 to start program FPGA Config Data from SRAM to ROM

8.6.1.1.1 10..9 = 01
8.6.1.1.2 8 = 1
8.6.1.1.3 7 = 0
8.6.1.1.4 6..3 = **0011**

8.6.1.2 Poll U1 Register 12 bit 15 until it is a one. One indicates that Fpga Config is done reading Config Data from ROM and storing to SRAM.

8.6.1.3 Write the following bits values to U1 Register 6 to ready for next command

8.6.1.3.1 10..9 = 01
8.6.1.3.2 8 = 0
8.6.1.3.3 7 = 0
8.6.1.3.4 6..3 = **0011**

8.6.1.4 Follow steps under ReadBack FPGA Config Data to SRAM to verify that data is the ROM match MCS file.

8.7 Rebooting FPGA (Reloading FPGA with Config Data from ROM):

8.7.1 Before issuing this command, make sure the Config Data for U1 is corrected because U1 code contains the Configuration algorithm. If wrong code is loaded into U1, a local reload (with a lab top) has to be done. This command will reboot all three FPGA

8.7.2 For all FPGA

8.7.2.1 Write the following bits values to U3 Register 7 to reboot all FPGA

8.7.2.1.1 **2..0 = 101**

9.0 Board Serial Number

9.1 The board serial number is stored in Serial ROM (SROM) in ASCII format. The board serial number is CTPV2xxx where xxx is (ASCII) number of the board. The serial ROM is connected to U3.

9.2 Steps to read the board serial number

9.2.1 Set SROM Address variable to zero.

9.2.2 Write SROM Address variable to **U3** Config 2 bits 9..0

9.2.3 Set Bit 15 of **U3** Config 2 to read one byte at SROM Address

9.2.4 Poll bit 15 of **U3** Status 3 for Read Done

9.2.5 Read byte (in ASCII) from **U3** Status 3 bits 7..0.

9.2.6 ReSet Bit 15 of **U3** Config 2.

9.2.7 Increment SROM Address.

9.2.8 Repeat 9.2.1 to 9.2.7 for the remain Bytes.

10.0 Register File

I-Square-C Address Mapping

I-Square-C Board Address	I-Square-C Sub Address	R/W	Function
0 (U1)	0	R	Status 0 15: Payload 13 MGT Channel Up 14: Payload 11 MGT Channel Up 13: Payload 9 MGT Channel Up 12: Payload 7 MGT Channel Up 11: Undefine 10: Undefine 9: Payload 15 MGT Lane 1 Up 8: Payload 15 MGT Lane 0 Up 7: Payload 13 MGT Lane 1 Up 6: Payload 13 MGT Lane 0 Up 5: Payload 11 MGT Lane 1 Up 4: Payload 11 MGT Lane 0 Up 3: Payload 9 MGT Lane 1 Up 2: Payload 9 MGT Lane 0 Up 1: Payload 7 MGT Lane 1 Up 0: Payload 7 MGT Lane 0 Up
	1	R	Status 1 15..2: Undefine 1: Payload 15,13,11,9,7 MGT Channels Up 0: Payload 15 MGT Channel Up
	2	R/W	Config 0 6: Enable Payload 7 8: Enable Payload 9 10: Enable Payload 11 12: Enable Payload 13 14: Enable Payload 15
	3	R/W	Config 1 1: Init All MGT
	4	R	Die Temperature
	5	R	Vint
	6	R/W	Config 2 Fpga Configuration: 10..9 = 1 -> Sel U1 for configuration = 2 -> Sel U3 for configuration = 3 -> Sel U24 for configuration 8 = Rising edge Execute Opcode 7 = 1 Select SRAM for writing 6..3 = 0 -> PROGRAM_DATA_U1. 1 -> PROGRAM_DATA_U3.

			2 -> PROGRAM_DATA_U24. **3 -> READ_DATA_U1.** **4 -> READ_DATA_U3.** **5 -> READ_DATA_U24.** **6 -> READ_EPROM_ID_U1.** **7 -> READ_EPROM_ID_U3.** **8 -> READ_EPROM_ID_U24.** **9 -> ERASE_EPROM_U1** **A -> ERASE_EPROM_U3** **B -> ERASE_EPROM_U24** **C ->** **D ->** **E ->** **F ->** **2..0 = undefine**
	7	**R/W**	**Config 3** **Fpga Configuration:** **9..0 -> Config ROM Block Number to Erase**
	8	**R/W**	**Config 4** **Sram Data** **15..0 = Data to be written to SRAM**
	9	**R/W**	**Config 5** **Sram Address** **15..0 = Sram Address 15..0**
	10	**R/W**	**Config 6** **Sram Address, R/W** **15 = Rising edge write Sram Data to Sram Address** **14 = Rising edge read Data at Sram Address. Data is available at Data From Sram** **13..6 = undefined** **5..0 = Sram Address 21..16**
	11	**R**	**Status 3** **Data From Sram** **15..0 = Data read from Sram Address**
	12-16		**Undefine**
	17	**R**	**Status 2** **15: 1 -> Fpga Config is Idle. Ready for opcode** **14..0: Firmware Version**
----------	**----------**	**-------**	**----------**
1 (U3)	**0**	**R**	**Status 0** **15: Payload 14 MGT Channel Up** **14: Payload 12 MGT Channel Up** **13: Payload 10 MGT Channel Up**

			12: Payload 8 MGT Channel Up **11: Undefine** **10: Undefine** **9: Payload 14 MGT Lane 1 Up** **8: Payload 14 MGT Lane 0 Up** **7: Payload 12 MGT Lane 1 Up** **6: Payload 12 MGT Lane 0 Up** **5: Payload 10 MGT Lane 1 Up** **4: Payload 10 MGT Lane 0 Up** **3: Payload 8 MGT Lane 1 Up** **2: Payload 8 MGT Lane 0 Up** **1: Payload 6 MGT Lane 1 Up** **0: Payload 6 MGT Lane 0 Up**
	1	**R**	**Status 1** **15..2: Undefine** **1: Payload 16,14,12,10,8 MGT Channels Up** **0: Payload 16 MGT Channel Up**
	2	**R/W**	**Config 0** **7: Enable Payload 8** **9: Enable Payload 10** **11: Enable Payload 12** **13: Enable Payload 14** **15: Enable Payload 16**
	3	**R/W**	**Config 1** **1: Init All MGT**
	4	**R**	**Die Temperature**
	5	**R**	**Vint**
	6	**R/W**	**Config 2** **SROM** **15: Rising edge read byte from SROM Address** **9..0: SROM Address**
	7	**R/W**	**Config 3** **2..0 = 5 -> reboot ALL FPGA**
	8-10		**Undefine**
	11	**R**	**Status 3** **Data From Serial ROM** **15: ASCII Data from SROM Valid** **7..0: ASCII Data read from SROM Address**
	12=16		**Undefine**
	17	**R**	**Status 2** **15: Undefine** **14..0: Firmwarw Version**
----------	**----------**	**-------**	**----------**
2 (U24)	**0**	**R**	**Status 0** **15: Payload 2 MGT Channel Up**

			14: Payload 5 MGT Channel Up **13: Payload 1 MGT Channel Up** **12: Payload 3 MGT Channel Up** **11: Payload 6 MGT Lane 1 Up** **10: Payload 6 MGT Lane 0 Up** **9: Payload 4 MGT Lane 1 Up** **8: Payload 4 MGT Lane 0 Up** **7: Payload 2 MGT Lane 1 Up** **6: Payload 2 MGT Lane 0 Up** **5: Payload 5 MGT Lane 1 Up** **4: Payload 5 MGT Lane 0 Up** **3: Payload 1 MGT Lane 1 Up** **2: Payload 1 MGT Lane 0 Up** **1: Payload 3 MGT Lane 1 Up** **0: Payload 3 MGT Lane 0 Up**
	1	**R**	**Status 1** **15..2: Undefine** **8: Error in Sum in Test Mode** **7: Fiber Channel Ready** **6: Fiber Lane Remote Up** **5: Fiber Lane Byte Align** **4: Fiber Lane Channel Align** **3: History Data REady** **2: Payload 6 MGT Channel Up** **1: Payload 1,2,3,4,5,6 MGT Channels Up** **0: Payload 4 MGT Channel Up**
	2	**R/W**	**Config 0** **0: Enable Payload 1** **1: Enable Payload 2** **2: Enable Payload 3** **3: Enable Payload 4** **4: Enable Payload 5** **5: Enable Payload 6**
	3	**R/W**	**Config 1** **15..3:** **2: Reset Fiber MGT** **1: Init All MGT** **0: History Arm**
	4	**R**	**U24 FX100T Die Temperature**
	5	**R**	**U24 FX100T Vint**
	6	**R/W**	**Undefine**
	7	**R/W**	**Undefine**
	8	**R/W**	**Final Sum Threshold LSB**
	9	**R/W**	**Final Sum Threshold MSB**
	10	**R**	**History Buffer Data LSB**
	11	**R**	**History Buffer Data MSB**

	12	**R/W**	**SD Test Control Register** **Bits:** **0 : Reset SYNC Count Reg 14** **1 : Reset TRIG 1 Count Reg 15** **2 : Reset TRIG 2 Count Reg 16**
	13	**R**	**SD Clock Frequency (MHz)** **Bits (7.. 0) : Clock 250 Count (Should be 250 +/- 2)**
	14	**R**	**Count Rising Edge of Sync From SD** **Bits (15:0) Count Rising Edge of SYNC from SD. Sync must be high for at least 30 nS**
	15	**R**	**Count Rising Edge of Trig1 from SD** **Bits (15:0) Count Rising Edge of TRIG1 from SD. Sync must be high for at least 30 nS**
	16	**R**	**Count Rising Edge of Trig2 from SD** **Bits (15:0) Count Rising Edge of TRIG2 from SD. Sync must be high for at least 30 nS**
	17	**R**	**Status 2:** **15: Undefine** **14..0: Firmware Version**

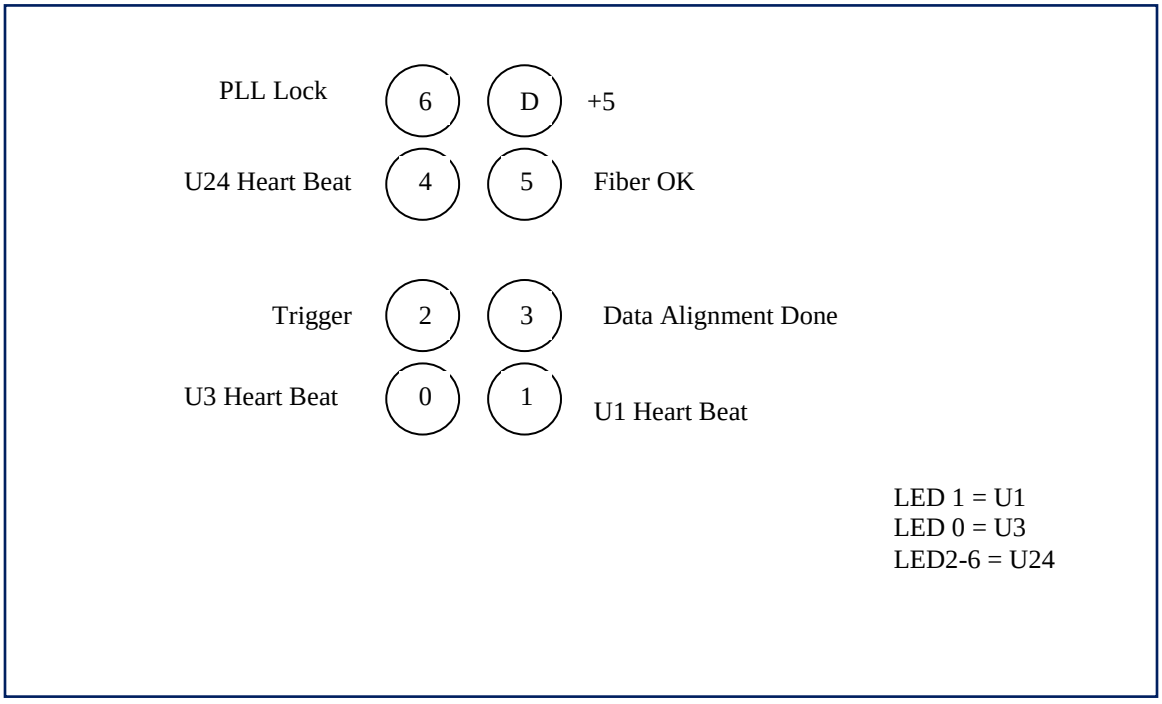
The history data advance after Data MSB is read. Hence read LSB then MSB.

Channel Number to PayLoad Mapping:

FPGA	FADC # Channel	Payload
U24	0	3
	1	1
	2	5
	3	2
	4	4
	5	6
U1	0	7
	1	9
	2	11
	3	13
	4	15
U3	0	8
	1	10
	2	12
	3	14
	4	16

```
Temperature_C = ((float)TempRegValue * 503.975/1024) - 273.15;
```

Front Panel LED:



Front Panel LVDS IN, LVDS OUT: