NUCLEAR PHYSICS DIVISION
FAST ELECTRONICS GROUP

Description and Requirements
for the

Global Trigger Processor

Benjamin Raydo (braydo@jlab.org)
Chris Cuevas (cuevas@jlab.org)
Scott Kaneta (skaneta@jlab.org)
Christopher Hewitt (hewittc@jlab.org)

5 February 2014

Table of Contents

1 - Introduction

The Global Trigger Processor (GTP) module is being designed for the Jefferson Lab 12GeV experimental halls. This unit is responsible for sending fixed latency trigger signals to the Trigger Supervisor (TS). The trigger signals are determined by evaluating the various detector subsystem Level 1 variable streams (derived from the Sub System Processors, SSP) in a set of trigger equations that reside in the GTP. Trigger equations are evaluated in the 250MHz system pipeline and can be reprogrammed to form different types of triggers by programming the Field Programmable Gate Array (FPGA) that performs the processing. The GTP is designed for a dual-star configured VXS crate and occupies one switch slot. The Global Trigger Crate (VXS) can only hold one GTP module located in the Switch A slot and can compute up to 30 simultaneous trigger equation evaluations. The TS located in the Trigger Distribution Crate makes the final trigger decision and will distribute a trigger word to the front-end crates when the desired trigger conditions are met.

Figure 1a is a logical diagram of the Global Trigger system, and shows where the GTP module resides in the Level 1 trigger system along with some of the critical signals interface it to the system.

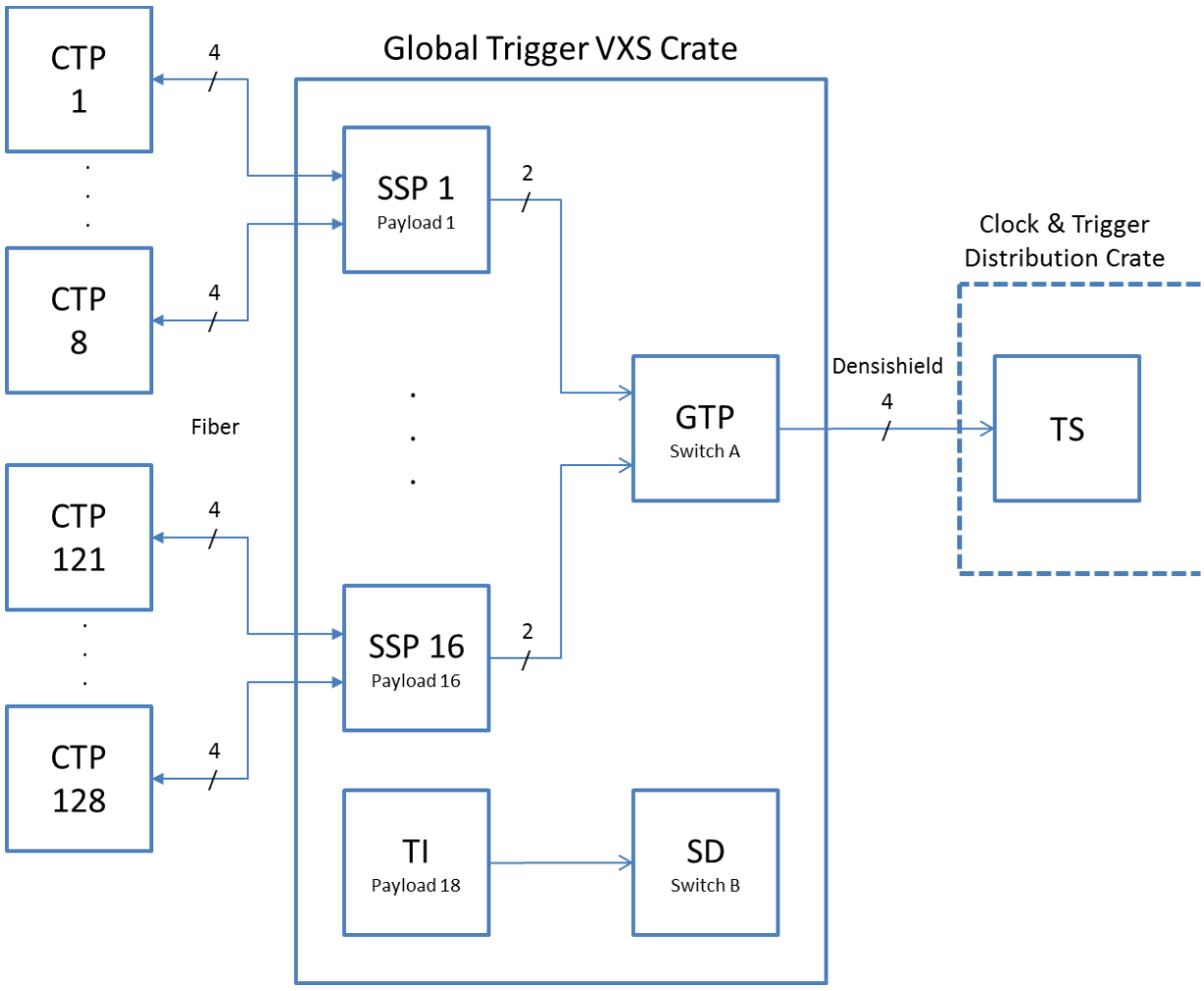


Figure 1a: Global Trigger Crate

2 - Purpose of the module

The GTP unit can receive up to 16 simultaneous subsystem variable streams to generate Level 1 triggers. Each subsystem variable stream provides a 32 bit quantity every 4ns (250MHz clock) indicating the current status of that subsystem (i.e. the total energy, number of tracks, or min/max energy for that subsystem in the current clock cycle). The eight 32 bit system variables are evaluated in up to 30 trigger equations and each can be mapped to a trigger output bit high or low as defined by the result of the trigger equation. All logic and computations run at the global 250MHz clock and are pipelined to allow continuous trigger decisions.

3 - Functional Description

Figure 3a shows a detailed block diagram of the Global Trigger Processor module. There are 4 major sections to this module and include: signal distribution, high speed VXS serial link management, trigger algorithm processing, and CPU management interface.

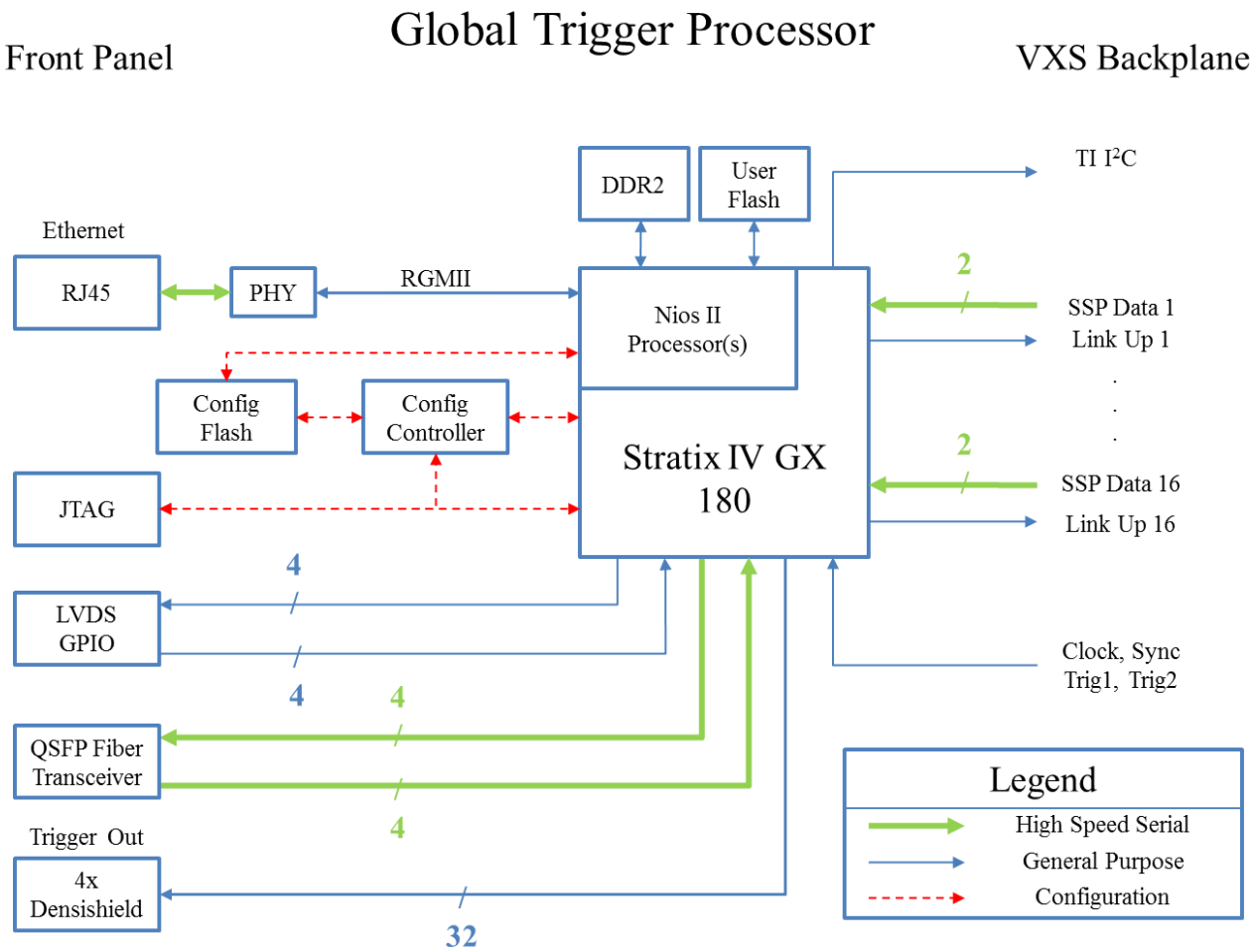


Figure 3a: Global Trigger Processor Hardware Block Diagram

3.1 - Signal Distribution

The standard configuration of the VXS crates in the data acquisition system requires distribution of the clock, trigger, and sync signals. This is done with the Signal Distribution (SD) module, which is placed in the VXS B switch slot.

Note: The transmit-to-receiver latency can vary by a few clock cycles between the Xilinx Aurora links each time the receive locks to the incoming data stream, but once locked the latency is fixed. Ultimately the GTP retimes the final trigger decision such that its latency is fixed with respect to the physics event (accurate to within roughly 4ns).

3.2 - High Speed VXS Serial Management

The SSP modules each communicate to the GTP via a 2 lane high speed serial VXS connection. Each lane operates at 5 Gbps providing a 10Gbps link. Using the Xilinx Aurora protocol this 10Gbps channel allows each SSP to provide a 32bit “instantaneous” subsystem quantity every 4ns

The Xilinx Aurora protocol uses the standard 8b10b line encoding scheme found in modern high speed networking protocols. Some of the main features of this encoding scheme include: DC balanced data stream, limited run length, and single bit error detection. Prototyping with the high speed serial links has revealed extremely low error rates when used on the VXS backplanes with the Xilinx Aurora protocol. While there is available bandwidth in the SSP->GTP links to provide capability for error correction, the error rates have shown to be low enough such that we can reliably use error detection and either ignore erroneous data or restart the Level 1 trigger system to avoid use of corrupted data.

As shown in Figure 3a, one FPGA will process all sixteen sets of SSP serial data streams. It is responsible for retiming the subsystem data streams such that a fixed latency trigger can be formed by the final computational stage. The retiming is done by using the fixed latency “SYNC” signal from the Trigger Inteface (TI). When the Level 1 trigger system starts, by deassertion of the “SYNC” signal, it will buffer the subsystem streams in a FIFO and after a fixed number of cycles (the number of cycles is determined by ensuring each SSP has been able to send at least to the GTP 2 subsystem words) the FIFOs will be read out simultaneously, which turns the SSP streams into a fixed latency stream for the final stage in the Level 1 trigger.

Deskewing the subsystem data streams such that correlated physics events arrive at the Trigger Algorithm Processors at the same time will simplify the final processing stage.

3.3 - Trigger Algorithm Processing

Subsystem trigger data flows from the SSP into the GTP. Currently the Hall D implementation requires the use of 8 SSP modules to support triggering for the following subsystems: barrel calorimeter (BCAL), forward calorimeter (FCAL), tagger microscope (TAGM), tagger hodoscope (TAGH), pair spectrometer (PS), start counter (ST), and time of flight (TOF). Each subsystem tends to have characteristics unique characteristics that require the trigger logic to be specific to that detector rather than generic.

The Hall D general trigger implementation is shown on figure 3.3a, where each subsystem data stream comes from specific SSP with a specific data format. Programmable delays are introduced to those data streams that allow the subsystems to be deskew with respect to each other. The deskew is intended to compensate for cable length differences, detector response time differences, detector position offsets, and allow intentional skew introduction in cases where the trigger decision timing is desired to be set by a particular detector. The next stage performs some post-processing on the subsystem data streams. In this case, only the FCAL requires this as to sum together the data from two SSP data streams into one. Finally, programmable coincidence windows are formed defined by the pulse extensions applied to hits (TAGM, TAGH, PS, ST, TOF cases) or by defining the time window of integration in the case of energy sums and number of simultaneous hits (BCAL, FCAL cases).

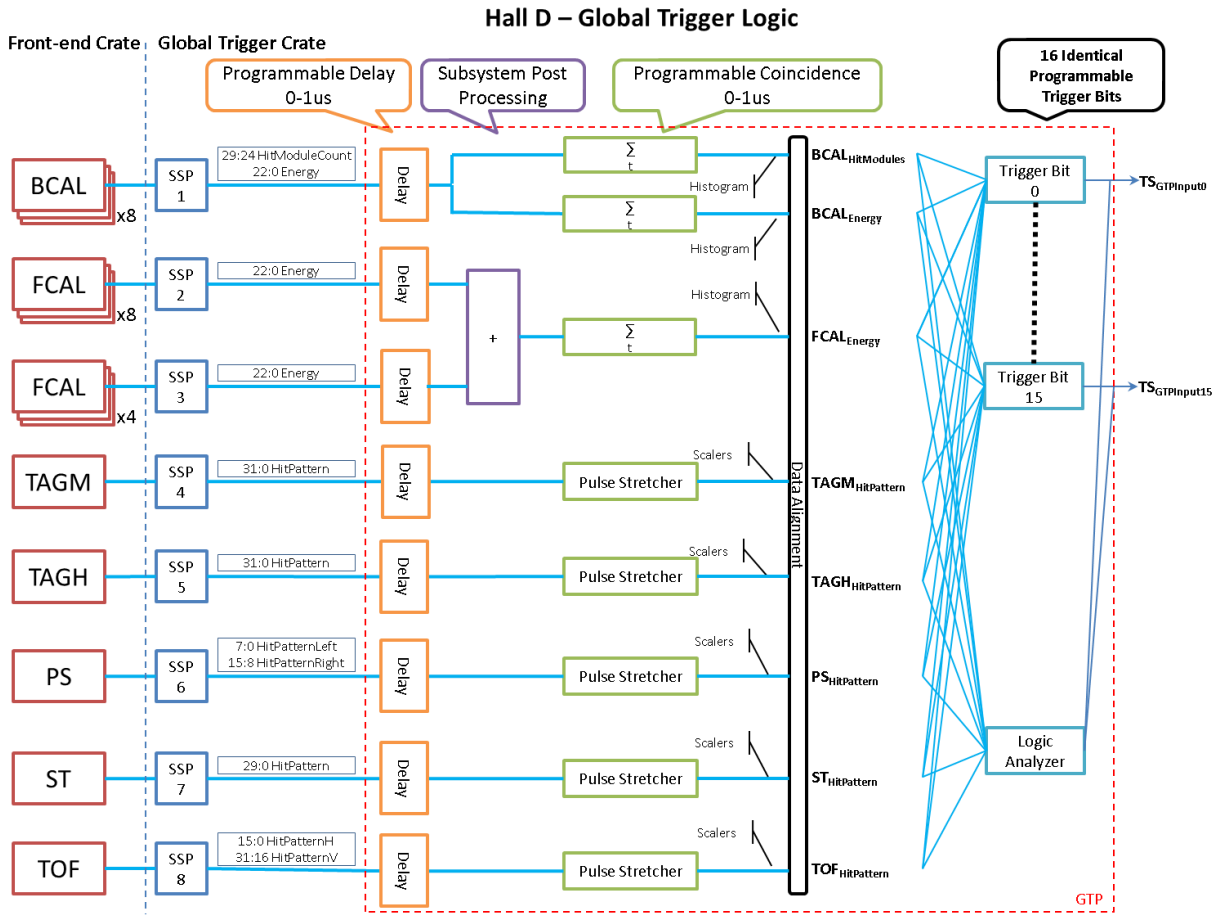


Figure 3.3a

Finally the copies of the subsystem data streams sent to each of the trigger bits where the logic is shown in figure 3.3b. The trigger bit logic has several thresholds, scale factors, and masks that can be applied to the subsystem data. An arbitrary selection of the desired subsystem logic equations can be put into coincidence that make the final trigger bit decision. The trigger bit decision has a programmable pulse extension that can be applied and also has its timing adjusted so that a fixed latency trigger decision is made. The fixed latency trigger decision is defined as a constant propagation time from a detector output signal to the respective trigger bit decision. This is critical to ensure the proper time window is captured on the front end (those time windows should capture the data that formed and agree with the trigger decision).

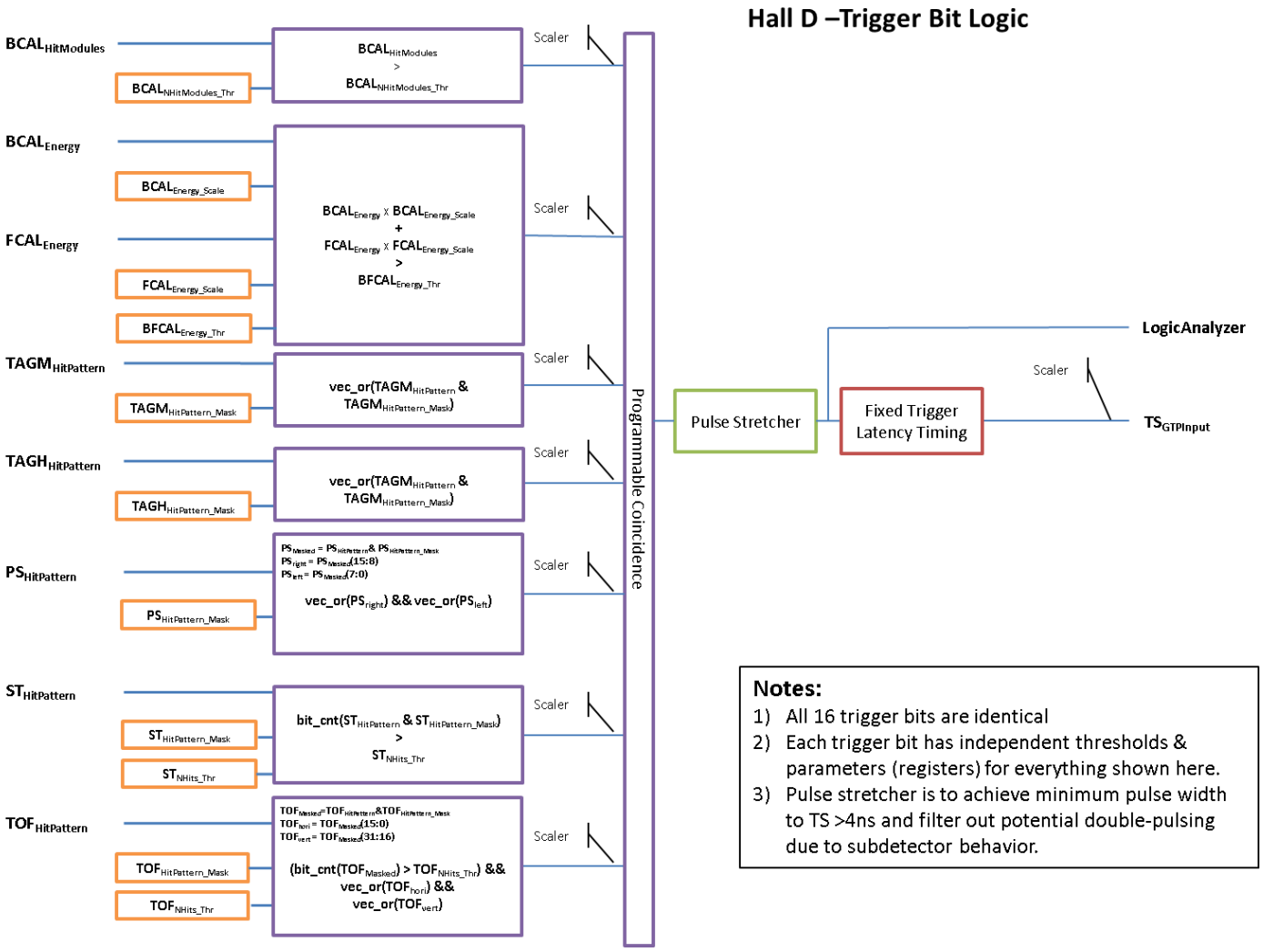


Figure 3.3b

3.4 - CPU management interface

To aid configuration, programming, and status reporting, the GTP will use its I^2C and Ethernet connections. VXS Switch modules in the level 1 trigger use an I^2C to communicate with the TI so their registers can be reported through the VME bus. Due to its limited speed and size, the I^2C interface will be used for board type, firmware version, and Ethernet status and configuration. Refer to the GTP Register spreadsheet for details.

The Ethernet connection will interface with the FPGA for all other configuration and control. These large FPGA devices will utilize 128 MB flash memory to store multiple FPGA images and a separate 128 MB for embedded processor code and other files. These devices will be programmable via the Ethernet interface to modify the firmware and software. This will be particularly timesaving if there are a host of trigger equations that need to be loaded for different experiments. The LAN interface will also allow real-time readout of onboard scalers, general reconfiguration relating to setting up SSP input ports, and programming of trigger equation coefficients.

The FPGA provides enough resource headroom to incorporate a reconfigurable Nios II processor. This "soft" processor enables the GTP to run an operating system without adding any additional physical components to the board. Running GNU/Linux on the GTP offers a number of advantages to the project, including access to its advanced networking capabilities, remote administration with secure shell, hardware peripheral access from user space, and a familiar environment with which software developers can extend functionality. Mentor Graphics provides a free cross-compiling Nios II toolchain based on the GNU Compiler Collection (GCC) called Sourcery CodeBench Lite, which is used to build the operating system from a development workstation. Remote debugging is also possible through the JTAG interface or over the network with "gdbserver".

3.4.1 – System integration

The Nios II platform is specified using Altera Qsys. The tool provides an interface to interconnect a Nios II processor and timers with various peripherals, such as memory and flash storage, which physically exist on the GTP board. Qsys generates a hardware description from the configuration that is included with the rest of the FPGA design project. A command line tool called "sopc2dts" converts the Qsys specification into header files and device tree source, which the bootloader and kernel use to map memory addresses for peripheral access.

3.4.2 – Barebox bootloader

Barebox is a bootloader for embedded systems containing support for the Nios II architecture. Despite its small binary size, Barebox offers a substantial set of features, including the ability to boot kernel images and mount root filesystems remotely using TFTP and NFS protocols. Remote booting expedites the testing process as images can be quickly loaded directly into memory, bypassing lengthy writes to flash storage. Barebox also provides a familiar Unix-like command line interface when

accessed via Altera's JTAG interface or the GPT's serial port and offers a lot of flexibility to script tasks such as network configuration and booting.

A board support package for the GTP board was derived from Barebox's generic Nios II platform example and the header file generated by the "sopc2dts" tool. Modifications were made to the relevant drivers to accommodate the pairing of the Altera Triple Speed Ethernet MAC with the Micrel KSZ9021RN Gigabit Ethernet PHY. Additionally, a partition table was created for the two flash memory devices on the GTP. The first flash device contains small partitions for the Barebox binary image and its configuration environment, as well as larger partitions for both the kernel and the root filesystem. A small general-purpose configuration partition resides at the end for any additional non-volatile parameters, such as a MAC address. The second flash device only contains one partition to contain loadable bitstreams for reprogramming the Stratix IV FPGA.

3.4.3 – Linux kernel

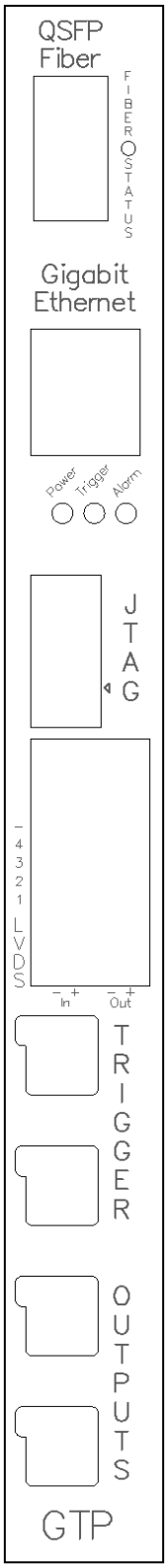
While the mainline Linux kernel does not support the Nios II architecture, a community named RocketBoards has made significant efforts to port to and support this platform. At the time of the GTP Linux porting effort, kernel 3.12 was the most current stable release and determined as the best starting point. As with Barebox, a board support package was created for the GTP. Configuration options were selected to best match the combination of attached peripherals and memory layout. Additionally, complications with the MAC and PHY kernel drivers had to be addressed before networking functioned reliably on the GTP.

3.4.4 – Root filesystem

Since there is no formal distribution of Linux for the Nios II architecture, the root filesystem, which Linux mounts at boot, must be built from source code. A package of scripts called Buildroot assists in this process and can automatically download and build the required files. Additionally, other packages such as OpenSSH and uEmacs can be selected for building. Once all source code is built, Buildroot generates a JFFS2 image, a filesytem optimized for NOR flashes, which is then flashed to the device.

Once Barebox, the Linux kernel, and the root filesystem exist on the flash memory, the board can be successfully booted into a usable Linux environment. Additional software can be developed in C or C++ and cross-compiled for the GTP Linux environment then transferred to the system via SCP, TFTP, or NFS.

Specification Sheet



MECHANICAL
- Single width VITA 41 (VXS) Switch Module

HIGH SPEED SERIAL I/O:

From Switch B (Signal Distribution module)
- 250MHz Clock (LVPECL)
- Trig1, Trig2, Sync (LVPECL)

To Payload Ports 1-16 (SSP/spare modules)
- 2x 5 Gbps SSP Stream (Xilinx Aurora)
- LINKUP out (LVTTL)
- BUSY out (LVTTL)

Outputs:
- Trigger output decision time <512ns

Front Panel:
- 1x RJ45: Ethernet
- 4x Densishield: Trigger Signals (LVPECL)
- 1x QSFP Fiber Transceiver
- 1x JTAG

Indicators: (Front Panel)
- Power – Blue LED
- Trigger – Amber LED
- Alarm – Red LED

Programming:
- On board JTAG Port
- 100/1000Mbps Ethernet Interface
- Trigger equation and processing for up to 16 JLAB SubSystem Processors

Power Requirements:
- +5v @ 10 Amps (From Backplane)
- Local regulators for other required voltages

Environment:
- Forced air cooling: N CFM
- Commercial grade components (85°C max)

4 - Registers

The GTP registers can be accessed two ways: Ethernet and I2C using the TI. The I2C register set is very limited intended to convey only enough information so that the Ethernet interface can be used. The Ethernet interface uses a custom protocol to facilitate register access as well as other functions that will be described in detail later.

I2C Register Summary:

Note: I2C accesses are 16bits each. Address offsets listed are 16bit word offsets. These registers are identical to the ones in the Ethernet interface (Cfg peripheral), please refer to the Ethernet versions for bit specific details. Since the Ethernet interface uses 32bit words (typically) the I2C uses 2x 16bit registers to provide the same information. The lower address access in the I2C words are the lower 16bits of the 32bit Ethernet word.

Register Name	Description	Address Offset
BoardId	Board identification	0x00-0x01
FirmwareRev	Firmware revision	0x02-0x03
CpuStatus	Linux CPU status	0x04-0x05
Hostname[]	Linux CPU hostname	0x06-0x0D

Ethernet Register Summary:

Note: Ethernet registers are 32bits each unless noted others. Unaligned non-32bit word accesses are not supported unless stated explicitly.

Register Name	Description	Address Offset
Cfg peripheral (offset 0x0000)		
BoardId	Board identification	0x0000
FirmwareRev	Firmware revision	0x0004
CpuStatus	Linux CPU status	0x0008
Hostname[]	Linux CPU hostname	0x000C
Clk peripheral (offset 0x0100)		
Ctrl	Clock control	0x0000
Status	Clock status	0x0004
Sd peripheral (offset 0x0200)		
ScalerLatch	Latch scalers	0x0000
Scalers[]	Scalers	0x0004
PulserPeriod	Pulser Period	0x0100
PulserLowCycles	Pulser low cycles	0x0104
PulserNPulses	Pulser pulse count	0x0108
PulserDone	Pulser status	0x010C
PulserStart	Pulser start	0x0110
SrlSel[]	Signal muxing	0x0120

La peripheral (offset 0x0400)		
Ctrl	Analyzer Control	0x0000
Status	Analyzer Status	0x0004
Data[]	Analyzer data	0x0020
CompareMode[]	Analyzer compare modes	0x0040
CompareThreshold[]	Analyzer compare thresholds	0x0060
MaskEn[]	Analyzer masks	0x0080
MaskVal[]	Analyzer match values	0x00A0
GxbConfig peripheral (offset 0x0500, 0x0600)		
Status	Gxb Config Status	0x0000
Ctrl	Gxb Config Control	0x0004
Ctrl2	Gxb Config Control	0x0008
TxRxIn	Gxb Config Data In	0x000C
TxRxOut	Gxb Config Data Out	0x0010
Serdes peripheral (0x1000, 0x1100, …, 0x1F00)		
Ctrl	Control	0x0000
Status	Status	0x0010
ErrTile	Tile 0 rx bit errors	0x0018
Status2	Status	0x001C
LaCtrl	Analyzer Control	0x0020
LaStatus	Analyzer Status	0x0024
LaData[]	Analyzer Data	0x0030
CompareMode	Analyzer Compare Mode	0x0050
CompareThreshold	Analyzer Compare Thresold	0x0070
MaskEn[]	Analyzer Masks	0x0090
MaskVal[]	Analyzer Compare Vals	0x00B0
Trigger peripheral (offset 0x2000)		
Ctrl	SSP trigger configuration	0x0000
BCal peripheral (offset 0x3000)		
Delay	Pulse delay	0x0000
Width	Pulse width	0x0004
HistDataEnergy	Energy histogram	0x0010
HistDataHits	Hit count histogram	0x0018
FCal peripheral (offset 0x3100)		
Delay	Pulse delay	0x0000
Width	Pulse width	0x0004
HistDataEnergy	Energy histogram	0x0010

GtpHitPattern peripheral (offset 0x3200, 0x3300, 0x3400, 0x3500, 0x3600)		
Delay	Pulse delay	0x0000
Width	Pulse width	0x0004
Scalers[]	Bit pattern scalers	0x0080
Trigbit peripheral (offset 0x4000, 0x4100, …, 0x4F00)		
Ctrl	Logic control	0x0000
TrigOutCtrl	Trigout control	0x0004
TrigOutStatus	Trigout status	0x0008
BCalCtrl0	BCal control	0x0010
BCalCtrl1	BCal control	0x0014
FCalCtrl	FCal control	0x0020
BFCalCtrl	BCal,FCAL control	0x0030
PSCtrl	PS control	0x0040
STCtrl0	ST control	0x0050
STCtrl1	ST control	0x0054
TOFCtrl0	TOF control	0x0060
TOFCtrl1	TOF control	0x0064
TagMCtrl	TAGM control	0x0070
TagHCtrl	TAGM control	0x0074
Scalers[]	Scalers	0x0080

4.1 Cfg Peripheral Registers Section (Peripheral offset = 0x0000)

Basic board information registers can be used to verify that this board is the GTP and check for the software revision, which should be checked for compatibility. When using the I2C access to these registers, the Ethernet hostname of the Linux CPU can be retrieve so that a connection can be established over Ethernet for access to the full register space.

Register: BoardId

Address Offset: 0x0000
Size: 32bits
Reset State: 0x47555020

31	30	29	28	27	26	25	24
BOARD_ID							
23	22	21	20	19	18	17	16
BOARD_ID							
15	14	13	12	11	10	9	8
BOARD_ID							
7	6	5	4	3	2	1	0
BOARD_ID							

BOARD_ID (RO):

0x47555020 = "GTP " in ASCII

Register: FirmwareRev

Address Offset: 0x0004
Size: 32bits
Reset State: 0xXXXXXXXX

31	30	29	28	27	26	25	24
GTPTYPE							
23	22	21	20	19	18	17	16
GTPTYPE							
15	14	13	12	11	10	9	8
FIRMWARE_REV_MAJOR							
7	6	5	4	3	2	1	0
FIRMWARE_REV_MINOR							

GTPTYPE(RO):

Firmware build type [0x0000 to 0xFFFF]
Defined types:
0x0001 HallD

FIRMWARE_REV_MAJOR (RO):

Major firmware revision number

FIRMWARE_REV_MINOR (RO):

Minor firmware revision number

Register: CpuStatus

Address Offset: 0x0008
Size: 32bits
Reset State: 0x0000FFFC

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	BOOTED

BOOTED (RO):

'0' – GTP Linux server not running
'1' – GTP Linux server is running

Register: Hostname[]

Address Offset: 0x000C, 0x0010, 0x0014, 0x0018
Size: 32bits
Reset State: 0x00000000, 0x00000000, 0x00000000, 0x00000000

31	30	29	28	27	26	25	24
HOSTNAME_CHAR3,7,11,15							
23	22	21	20	19	18	17	16
HOSTNAME_CHAR2,6,10,14							
15	14	13	12	11	10	9	8
HOSTNAME_CHAR1,5,9,13							
7	6	5	4	3	2	1	0
HOSTNAME_CHAR0,4,8,12							

HOSTNAME_CHARx(RO):

NULL terminated string (maximum length is 15 characters).
This string is the GTP Linux hostname intended to be read from I2C to then establish a connection via Ethernet.

4.2 Clk Peripheral Registers Section (Peripheral offset = 0x0100)

This peripheral is responsible for configuring and monitoring the clock signals used to drive the GTP logic and Serdes.

Register: Ctrl

Address Offset: 0x0000
Size: 32bits
Reset State: 0x80000000

31	30	29	28	27	26	25	24
CLKRESET	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	CLKSRC	

CLKRESET (RW):

'1' – GTP PLL reset asserted.
'0' – GTP PLL reset released.

CLKSRC (RW):

0 or 3 – Clock source disabled.
1 – VXS SWB 250MHz clock source used.
2 – Internal 250MHz clock source used.

Note: CLKRESET should be asserted when CLKSRC is being changed or there is no clock source. Only when CLKSRC has been set to a stable input clock should CLKRESET be released.

Register: Status

Address Offset: 0x0004
Size: 32bits
Reset State: 0xXXXXXXXX

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	R_LOCK	L_LOCK	GCLK_LOCK

GCLK_LOCK(RO): '1' - Global clock PLL is locked. '0' – not locked.
L_LOCK(RO): '1' – Left side Serdes transmit clock PLL is locked. '0' – not locked.
R_LOCK(RO): '1' – Right side Serdes transmit clock PLL is locked. '0' – not locked.

4.3 Sd Peripheral Registers Section (Peripheral offset = 0x0200)

Signal source muxing, generic pulser, and scalers are handled by the Sd peripheral.

Register: ScalerLatch

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DISABLE

DISABLE (RW):

'1' – Scalers/histograms on GTP are halted. This must be done before reading values.
'0' – Scalers/histograms on GTP are active and accumulating.

Note: Normally the GtpServer app running on the Linux CPU manages the scalers and their control. To read scaler values, see the GtpServer interface documentation in a following chapter.

Register: Scalers[]+SD_SCALER_x*4

Address Offset: 0x0004
Size: 32bits
Reset State: 0xXXXXXXXX

31	30	29	28	27	26	25	24
SCALER							
23	22	21	20	19	18	17	16
SCALER							
15	14	13	12	11	10	9	8
SCALER							
7	6	5	4	3	2	1	0
SCALER							

SCALER (RO): 32bit scaler value.

The index into Scalers[] correspond to scalers according to this table:

Scaler Name	Index in Scalers[]	Description
SD_SCALER_SYSCLK	0	50MHz clock. Used as reference for normalizing
SD_SCALER_GCLK	1	250MHz Global clock
SD_SCALER_SYNC	2	VXS SWB Sync input
SD_SCALER_TRIG1	3	VXS SWB Trig1 input
SD_SCALER_TRIG2	4	VXS SWB Trig2 input
SD_SCALER_BUSY	5	VXS SWB Busy output (number of times asserted)
SD_SCALER_BUSYCYCLES	6	VXS SWB Busy output (number of GCLK cycles asserted)
SD_SCALER_FP_IN0	7	Front panel anylevel input 0
SD_SCALER_FP_IN1	8	Front panel anylevel input 1
SD_SCALER_FP_IN2	9	Front panel anylevel input 2
SD_SCALER_FP_IN3	10	Front panel anylevel input 3

SD_SCALER_FP_OUT0	11	Front panel LVDS output 0
SD_SCALER_FP_OUT1	12	Front panel LVDS output 1
SD_SCALER_FP_OUT2	13	Front panel LVDS output 2
SD_SCALER_FP_OUT3	14	Front panel LVDS output 3

Register: PulserPeriod

Address Offset: 0x0100
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
PERIOD							
23	22	21	20	19	18	17	16
PERIOD							
15	14	13	12	11	10	9	8
PERIOD							
7	6	5	4	3	2	1	0
PERIOD							

PERIOD (RW):

Pulser period in 4ns ticks

Register: PulserLowCycles

Address Offset: 0x0104
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
LOW_CYCLES							
23	22	21	20	19	18	17	16
LOW_CYCLES							
15	14	13	12	11	10	9	8
LOW_CYCLES							
7	6	5	4	3	2	1	0
LOW_CYCLES							

LOW_CYCLES(RW):

Number of 4ns ticks pulser output is held low during the pulser period.

Register: PulserNPulses

Address Offset: 0x0108
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
COUNT							
23	22	21	20	19	18	17	16
COUNT							
15	14	13	12	11	10	9	8
COUNT							
7	6	5	4	3	2	1	0
COUNT							

COUNT (R/W):

0x00000000: disable pulser output
0x00000001 to 0xFFFFFFFE: number of periods to deliver pulser output
0xFFFFFFFF: infinite cycle count for pulser output

Notes:

1) When using fixed count of pulses the pulser must be trigger to start by writing to the **PulserStart** register

Register: PulserStart

Address Offset: 0x0110
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
PULSER_START							
23	22	21	20	19	18	17	16
PULSER_START							
15	14	13	12	11	10	9	8
PULSER_START							
7	6	5	4	3	2	1	0
PULSER_START							

PULSER_START (WO):

Write any value to start pulser operation. The pulse number counter is cleared.

Register: PulserDone

Address Offset: 0x010C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	DONE

DONE (RO):

'0' – pulser is still delivering pulses as defined in **PulserNPulses**
'1' – pulser is is not active (either disabled or has finished fixed pulse count)

Register: SrcSel[]

Address Offset: 0x0120 + 4*SD_SRC_x
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	SRC				

SRC (RW):

Selects the signal source for the output signal (output signal is indicated by the index into SrcSel[])

The SD_SRC_x ID map is used to determine which index in the SrcSel register array to use:

SD_SRC_x ID NAME	**Index in SrcSel**	**Description**
SD_SRC_TRIG	0	GTP trigger source
SD_SRC_SYNC	1	GTP sync source
SD_SRC_FP_LVDSOUT0	2	Front Panel LVDS output #0
SD_SRC_FP_LVDSOUT1	3	Front Panel LVDS output #1
SD_SRC_FP_LVDSOUT2	4	Front Panel LVDS output #2
SD_SRC_FP_LVDSOUT3	5	Front Panel LVDS output #3

Possible values for SRC contents of register:

SD_SRC_SEL_x	**Value**	**Source Signal Description**
SD_SRC_SEL_0	0	Drive constant ‘0’
SD_SRC_SEL_1	1	Drive constant ‘1’
SD_SRC_SEL_SYNC	2	VXS SWB Sync
SD_SRC_SEL_TRIG1	3	VXS SWB Trig1
SD_SRC_SEL_TRIG2	4	VXS SWB Trig2
SD_SRC_SEL_FPIN0	5	Front panel input#0
SD_SRC_SEL_FPIN1	6	Front panel input#1
SD_SRC_SEL_FPIN2	7	Front panel input#2
SD_SRC_SEL_FPIN3	8	Front panel input#3
SD_SRC_SEL_PULSER	18	Pulser output
SD_SRC_SEL_BUSY	19	Event builder busy
undefined	11-31	Undefined
SD_SRC_SEL_TRIGOUTx	32+x	Triggbit bit x output (x from 0 to 31)

4.4 La (Logic Analyzer) Peripheral Registers Section (Peripheral offset = 0x0400)

Provides the interface to the logic analyzer. This peripheral is for debugging purposes only. Information on this peripheral can be supplied upon request.

4.5 GxbConfig Peripheral Registers Section (Peripheral offset = 0x0500, 0x0600)

Interfaces the Serdes transceivers for low-level settings and testing. This peripheral is for debugging purposes only. Information on this peripheral can be supplied upon request.

4.6 Serdes Peripheral Registers Section (Peripheral offset = 0x1000, 0x1100, …, 0x1F00)

This peripheral configures and monitors the VXS serial links coming from the SSP. The following table indicates the peripheral address to payload mapping:

Peripheral Index	VXS Payload Port	Peripheral Address
0	PP1	0x1000
1	PP2	0x1100
2	PP3	0x1200
3	PP4	0x1300
4	PP5	0x1400
5	PP6	0x1500
6	PP7	0x1600
7	PP8	0x1700
8	PP9	0x1800
9	PP10	0x1900
10	PP11	0x1A00
11	PP12	0x1B00
12	PP13	0x1C00
13	PP14	0x1D00
14	PP15	0x1E00
15	PP16	0x1F00

Register: Ctrl

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000001

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	ERR_EN	ERR_RST	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	POWER_DOWN

POWER_DOWN (RW):

'0' – transceiver is disabled.
'1' – transceiver is enabled.

ERR_RST (RW):

'1' – reset bit error counters on link. Should be done once link is established

ERR_EN (RW):

'1' – enabled bit error counters on link.
'0' – disabled bit error counters on link.

Register: Status

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	RXSRCRDYN	TXLOCK	CHANNELUP	-	-	-	-
7	6	5	4	3	2	1	0
-	-	LANEUP1	LANEUP0	-	-	HARDERR1	HARDERR0

HARDERRx (RO):

'1' – transceiver lane x has a hard error. Serdes must be reset to recover
'0' – transceiver hard error condition not set.

LANEUPx (RO):

'1' – transceiver lane x is established.
'0' – transceiver lane x is not established.

CHANNELUP (RO):

'1' – transceiver channel is established.
'0' – transceiver channel is not established.

TXLOCK (RO):

'1' – transceiver reference transmitter PLL is locked
'0' – transceiver reference transmitter PLL is not locked.

RXSRCRDYN (RO):

'1' – no data is being received by transceiver.
'0' – data is being received by transceiver.

Notes:

1) TXLOCK must be '1' for transceiver to work. If not, check that clock sources are working/setup.
2) LANEUP signals must be '1' for transceiver channel to come up. If not, check connecting board to ensure its enabled and using the same reference clock.
3) Once CHANNELUP = '1', enable bit error monitors to track errors.

Register: ErrTile

Address Offset: 0x0018
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
LANE1_BITERRORS							
23	22	21	20	19	18	17	16
LANE1_BITERRORS							
15	14	13	12	11	10	9	8
LANE0_BITERRORS							
7	6	5	4	3	2	1	0
LANE0_BITERRORS							

LANEx_BITERRORS (RO):

16bit serdes bit error counter. Must be enabled in Ctrl to count.

Register: Status2

Address Offset: 0x001C
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	CRC_PASS
15	14	13	12	11	10	9	8
LATENCY							
7	6	5	4	3	2	1	0
LATENCY							

LATENCY (RO):

16bit latency measure from SYNC released to when first data word received by serdes

CRC_PASS (RO):

'1' – received data passed CRC verification for last SYNC period
'0' – received data failed CRC verification for last SYNC period

4.7 Trigger Peripheral Registers Section (Peripheral offset = 0x2000)

This peripheral selects the subsystem streams to enable and deskew then provides these a time coherent set of subsystem data streams for the trigger bit processor peripherals

Register: Ctrl

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
TRG_EN7	TRG_EN6	TRG_EN5	TRG_EN4	TRG_EN3	TRG_EN2	TRG_EN1	TRG_EN0

TRG_ENx (RW):

'0' – subsystem data stream is not enabled in GTP
'1' – subsystem data stream is enabled in GTP

The following table shows the mapping of TRG_EN bits to subsystem streams and their corresponding SSPs:

TRG_ENx	Subsystem Data Stream	SSP Payloads
0	BCal Energy	PP15
1	BCal Hit Modules	PP15
2	FCal Energy	PP13, PP11
3	TagM Hit Pattern	PP9
4	TagH Hit Pattern	PP7
5	PS Hit Pattern	PP5
6	ST Hit Pattern	PP3
7	TOF Hit Pattern	PP1

4.8 BCal Peripheral Registers Section (Peripheral offset = 0x3000)

The BCal peripheral performs some pre-processing and monitoring on the BCal SSP streams that are then fed as a common input to all trigger bit peripherals.

Register: Delay

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
DELAY							

DELAY (RW):

8bit delay value (0-255) in 4ns ticks that is added to the subsystem data stream. This is the delay mechanism to be used for deskewing subsystems with respect to each other.

Register: Width

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
WIDTH							

WIDTH (RW):

8bit width value (0-255) in 4ns ticks. This defines the integration time window for which the BCal Energy is formed as well as BCalHitModules. A value of 0 has an integration window equal to 1 sample, a value of 1 has an integration window equal to 2 samples, and so on…

Register: HistDataEnergy

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA (RO):

When scalers were enabled, then disabled (using the Sd.ScalerLatch enable/disable control) this register may be read 32 times, which extracts 32bins worth of the BCalEnergy histograms. Each bin represents the count of observed BCalEnergy integrals observed for that bin, where the bin is equal to 2^N (N is equal to the word number, 0-31, read out).

Note: Normally the GtpServer app running on the Linux CPU manages the scalers and their control. To read scaler values, see the GtpServer interface documentation in a following chapter.

Register: HistDataHits

Address Offset: 0x0018
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA (RO):

When scalers were enabled, then disabled (using the Sd.ScalerLatch enable/disable control) this register may be read 32 times, which extracts 32bins worth of the BCalHitModules histograms. Each bin represents the count of observed BCalHitModule integrals observed for that bin, where the bin is equal to the word number, 0-31, read out.

Note: Normally the GtpServer app running on the Linux CPU manages the scalers and their control. To read scaler values, see the GtpServer interface documentation in a following chapter.

4.9 FCal Peripheral Registers Section (Peripheral offset = 0x3100)

The FCal peripheral performs some pre-processing and monitoring on the FCal SSP streams that are then fed as a common input to all trigger bit peripherals.

Register: Delay

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
DELAY							

DELAY (RW):

8bit delay value (0-255) in 4ns ticks that is added to the subsystem data stream. This is the delay mechanism to be used for deskewing subsystems with respect to each other.

Register: Width

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
WIDTH							

WIDTH (RW):

8bit width value (0-255) in 4ns ticks. This defines the integration time window for which the FCal Energy is formed. A value of 0 has an integration window equal to 1 sample, a value of 1 has an integration window equal to 2 samples, and so on…

Register: HistDataEnergy

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

DATA (RO):

When scalers were enabled, then disabled (using the Sd.ScalerLatch enable/disable control) this register may be read 32 times, which extracts 32bins worth of the FCalEnergy histograms. Each bin represents the count of observed FCalEnergy integrals observed for that bin, where the bin is equal to 2^N (N is equal to the word number, 0-31, read out).

Note: Normally the GtpServer app running on the Linux CPU manages the scalers and their control. To read scaler values, see the GtpServer interface documentation in a following chapter.

4.10 SSGenPattern Peripheral Registers Section (Peripheral offset = 0x3200, 0x3300, …, 0x3600)

The SSGenPattern peripheral performs some pre-processing and monitoring on the generic pattern based SSP streams that are then fed as a common input to all trigger bit peripherals.

Register: Delay

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
DELAY							

DELAY (RW):

8bit delay value (0-255) in 4ns ticks that is added to the subsystem data stream. This is the delay mechanism to be used for deskewing subsystems with respect to each other.

Register: Width

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
WIDTH							

WIDTH (RW):

8bit width value (0-255) in 4ns ticks. This defines the number of ticks each bit in the pattern is pulse stretched. A value of 0 extends hits by 0ns, a value of 1 extends hits by 4ns, etc…

Register: Scalers[]

Address Offset: 0x0080+N*4
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
SCALER							
23	22	21	20	19	18	17	16
SCALER							
15	14	13	12	11	10	9	8
SCALER							
7	6	5	4	3	2	1	0
SCALER							

DATA[N] (RO):

N can range from 0 to 31 where SCALER[N] is the count of hits received on bit N of that hit pattern based subsystem data stream.

Note: Normally the GtpServer app running on the Linux CPU manages the scalers and their control. To read scaler values, see the GtpServer interface documentation in a following chapter.

4.11 Trigbit Peripheral Registers Section (Peripheral offset = 0x4000, 0x4100, …, 0x4F00)

The Trigbit peripheral performs the trigger bit logic that feeds the Trigger Supervisor, TS, GTP inputs. Currently only 16 Trigbit peripherals exists, mapping to the first 16 trigger bit inputs on the GTP data path of the TS. Each Trigbit peripheral receives an identical copy of the data stream as defined by the Trigger peripheral and BCal, FCal, SSGenPattern peripherals. Each Trigbit has its own set of registers to manipulate the input data streams and apply thresholds for define its trigger bit definition.

See section 3.3 for a detailed diagram on trigger bit logic and the registers below are used. A summary of the trigger bit equations are provided here where reference the register fields below:

BCalHitModules_Trig:
BCalHitModules >= *BCAL_HITMODULES_THR*

BFCalEnergy_Trig:
BCalEnergy**BCAL_ENERGY_SCALE* +
FCalEnergy**FCAL_ENERGY_SCALE* >= *BFCAL_ENERGY_THR*

TagMPattern_Trig:
or_vector(TagMHitPattern & *MASK*)

TagHPattern_Trig:
or_vector(TagHHitPattern & *MASK*)

PSPattern_Trig:
or_vector(PSHitPattern(7..0) & *MASK*(7..0)) &&
or_vector(PSHitPattern(15..8) & *MASK*(15..8))

STNHits_Trig:
Bit_count(STHitPattern & *MASK*) >= *ST_HITCOUNT_THR*

TOFNHits_Trig:
Bit_count(TOFHitPattern & *MASK*) &&
or_vector(TOFHitPattern(15..0) & *MASK*(15..0)) &&
or_vector(TOFHitPattern(31..16) & *MASK*(31..16))

Trig_Out:
Ctrl.En0 &&
(!*Ctrl.En1* || **BCalHitModules_Trig**) &&
(!*Ctrl.En2* || **BFCalEnergy_Trig**) &&
(!*Ctrl.En3* || **TagMPattern_Trig**) &&
(!*Ctrl.En4* || **TagHPattern_Trig**) &&
(!*Ctrl.En5* || **PSPattern_Trig**) &&
(!*Ctrl.En6* || **STNHits_Trig**) &&
(!*Ctrl.En7* || **TOFNHits_Trig**)

Register: Ctrl

Address Offset: 0x0000
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0

EN0 (RW):
'1' – enable trigger bit equation
'0' – disable trigger bit equation

EN1 (RW):
'1' – enable BCalHitModules logic in trigger bit equation
'0' – disable BCalHitModules logic in trigger bit equation

EN2 (RW):
'1' – enable BFCalEnergy logic in trigger bit equation
'0' – disable BFCalEnergy logic in trigger bit equation

EN3 (RW):
'1' – enable TagMPattern logic in trigger bit equation
'0' – disable TagMPattern logic in trigger bit equation

EN4 (RW):
'1' – enable TagHPattern logic in trigger bit equation
'0' – disable TagHPattern logic in trigger bit equation

EN5 (RW):
'1' – enable PSCoincidence logic in trigger bit equation
'0' – disable PSCoincidence logic in trigger bit equation

EN6 (RW):
'1' – enable STNHits logic in trigger bit equation
'0' – disable STNHits logic in trigger bit equation

EN7 (RW):
'1' – enable TOFNHits logic in trigger bit equation
'0' – disable TOFNHits logic in trigger bit equation

Register: TrigOutCtrl

Address Offset: 0x0004
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
WIDTH							
15	14	13	12	11	10	9	8
-	-	-	-	LATENCY			
7	6	5	4	3	2	1	0
LATENCY							

LATENCY (RW):

0-4095 (in 4ns ticks) for trigger bit latency. This time is measure from the global release of SYNC at the front-end crates to when the trigger bit sends decisions to the trigger supervisor. It is desirable to set this latency as large as the system can comfortably handle so that when future changes to the trigger logic are made no changes to the LATENCY or capture windows for the front-end crates need to be adjusted. In the case for Hall D, this value would perhaps be around 825 (825*4ns=3300ns).

WIDTH (RW):

0-255 (in 4ns ticks) that the trigger bit output pulse is extended. This may be useful to stretch pulses to ensure reliable capture by receiving end or to suppress multiple assertions of the trigger due to signals hovering around thresholds once fired.

Register: TrigOutStatus

Address Offset: 0x0008
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	LATENCY_ERR

LATENCY_ERR (RO):

This bit is asserted when the LATENCY in the CTRL register is not satisfied. For example, if the trigger logic or links are delaying the trigger bit decision past the desired latency set this bit will be set '1' to indicate this error condition. This condition will not be clear until another SYNC has been issued to flush and restart the trigger data pipeline.

Register: BCalCtrl0

Address Offset: 0x0010
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
BCAL_ENERGY_SCALE							

BCAL_ENERGY_SCALE (RW):

This is a scaling factor applied to the BCAL_ENERGY integral to be used in the BFCAL logic section of the trigger bit processing.

Register: BCalCtrl1

Address Offset: 0x0014
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	-	BCAL_HITMODULES_THR			

BCAL_HITMODULES_THR (RW):

This is a threshold applied to the BCAL_HITMODULES integral, making the BCAL_HITMODULES trigger logic decision.

Register: FCalCtrl0

Address Offset: 0x0020
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
FCAL_ENERGY_SCALE							

FCAL_ENERGY_SCALE (RW):

This is a scaling factor applied to the FCAL_ENERGY integral to be used in the BFCAL logic section of the trigger bit processing.

Register: BFCalCtrl0

Address Offset: 0x0030
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
BFCAL_ENERGY_THR							
23	22	21	20	19	18	17	16
BFCAL_ENERGY_THR							
15	14	13	12	11	10	9	8
BFCAL_ENERGY_THR							
7	6	5	4	3	2	1	0
BFCAL_ENERGY_THR							

BFCAL_ENERGY_THR (RW):

This is a threshold applied to the BFCAL logic section of the trigger bit processing.

Register: PSCtrl

Address Offset: 0x0040
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
MASK							
7	6	5	4	3	2	1	0
MASK							

MASK (RW):

This is a mask that is bitwise ANDed with the PSHitPattern data stream to disable certain bits from being used in the PS concidence section of the trigger bit processing.

Register: STCtrl0

Address Offset: 0x0050
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MASK							
23	22	21	20	19	18	17	16
MASK							
15	14	13	12	11	10	9	8
MASK							
7	6	5	4	3	2	1	0
MASK							

MASK (RW):

This is a mask that is bitwise ANDed with the STHitPattern data stream to disable certain bits from being used in the STNHits section of the trigger bit processing.

Register: STCtrl1

Address Offset: 0x0054
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	ST_HITCOUNT_THR				

ST_HITCOUNT_THR (RW):

This is a threshold applied to the STNHits section of the trigger bit processing.

Register: TOFCtrl0

Address Offset: 0x0060
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MASK							
23	22	21	20	19	18	17	16
MASK							
15	14	13	12	11	10	9	8
MASK							
7	6	5	4	3	2	1	0
MASK							

MASK (RW):

This is a mask that is bitwise ANDed with the TOFHitPattern data stream to disable certain bits from being used in the TOFNHits and coincidence section of the trigger bit processing.

Register: TOFCtrl1

Address Offset: 0x0064
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
-	-	-	-	-	-	-	-
23	22	21	20	19	18	17	16
-	-	-	-	-	-	-	-
15	14	13	12	11	10	9	8
-	-	-	-	-	-	-	-
7	6	5	4	3	2	1	0
-	-	-	TOF_HITCOUNT_THR				

TOF_HITCOUNT_THR (RW):

This is a threshold applied to the TOFNHits section of the trigger bit processing.

Register: TagMCtrl

Address Offset: 0x0070
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MASK							
23	22	21	20	19	18	17	16
MASK							
15	14	13	12	11	10	9	8
MASK							
7	6	5	4	3	2	1	0
MASK							

MASK (RW):

This is a mask that is bitwise ANDed with the TagMHitPattern data stream to disable certain bits from being used in the TagMHitPattern OR section of the trigger bit processing.

Register: TagHCtrl

Address Offset: 0x0074
Size: 32bits
Reset State: 0x00000000

31	30	29	28	27	26	25	24
MASK							
23	22	21	20	19	18	17	16
MASK							
15	14	13	12	11	10	9	8
MASK							
7	6	5	4	3	2	1	0
MASK							

MASK (RW):

This is a mask that is bitwise ANDed with the TagHHitPattern data stream to disable certain bits from being used in the TagHHitPattern OR section of the trigger bit processing.

Register: Scalers[]+TRIGBIT_SCALER_x*4

Address Offset: 0x0080
Size: 32bits
Reset State: 0xXXXXXXXX

31	30	29	28	27	26	25	24
SCALER							
23	22	21	20	19	18	17	16
SCALER							
15	14	13	12	11	10	9	8
SCALER							
7	6	5	4	3	2	1	0
SCALER							

SCALER (RO): 32bit scaler value.

The index into Scalers[] correspond to scalers according to this table:

Scaler Name	Index in Scalers[]	Description
TRIGBIT_SCALER_BCALHIT	0	BCalHitModules trigger bit logic
TRIGBIT_SCALER_BFCAL	1	BFCalEnergy trigger bit logic
TRIGBIT_SCALER_TAGM	2	TagM trigger bit logic
TRIGBIT_SCALER_TAGH	3	TagH trigger bit logic
TRIGBIT_SCALER_PS	4	PS trigger bit logic
TRIGBIT_SCALER_ST	5	ST trigger bit logic
TRIGBIT_SCALER_TOF	6	TOF trigger bit logic
TRIGBIT_SCALER_TRIGOUT	7	Trigbit output

5 Example sequence for board initialization

Certain sequences must be following in order for the GTP configuration to successfully store registers, acquire seriail links, and process trigger logic. The following information will outline a sequence for successful configuration for the typical VXS configuration.

1. **Identify and connect to GTP**
    a. Using I^2C, read Cfg.BoardId to confirm GTP board identity
    b. Using I^2C, read Cfg.FirmwareRev to confirm GTP firmware compatibility with driver
    c. Using I^2C, read Cfg.CpuStatus bit 0 to check if GTP Linux server app is running
    d. If all above checks are valid, use I^2C to read Cfg.Hostname and finally connect to the GtpServer app over Ethernet according to the outlined protocol described below.
    e. Verify Ethernet connection with GtpServer by reading Cfg.BoardId once again. All communication with the GTP shall now go over the Ethernet interface.
2. **Clock, Trig, Sync, Busy setup**
    a. Once the VXS crate clock from the TI is set to the running source and is stable, set Clk.Ctrl to use the VXS clock source and assert reset.
    b. Deassert Clk.Ctrl reset, delay >10ms, then confirm all clock PLLs are locked in Clk.Status (if not there is a problem with the source clock selection)
    c. Set Sd.SetSrcSel[SD_SRC_TRIG] to SD_SRC_SEL_TRIG1 (Trig is from SWB Trig1)
    d. Set Sd.SetSrcSel[SD_SRC_SYNC] to SD_SRC_SEL_SYNC (Sync is from SWB Sync)
3. **Enable/Reset Serial links from SSP**
    a. Serdes[].Ctrl = 0x1, then Serdes[].Ctrl = 0x0 for existing SSP payloads (enable populated serdes)
    b. Serdes[].Ctrl = 0x1 for non-existing SSP payloads (disable unpopulated serdes)
4. **Verify Serial links from SSP (once SSPs are configured)**
    a. Check Serdes[].Status bit12 to verify channel is up on all enabled SSP payloads. If channel is not up a reset may be needed (go back to step 3), possibly also on the SSP side. Print a bit decoding of the Serdes[].Status to help diagnose the issue.
    b. If channel is up, enable and reset the bit error counter: Serdes[].Ctrl = 0xC00
    c. Note, steps 1-3 only need to happen during crate initialization, step 4 and beyond can be repeated any number of times to change settings between runs
5. **General Trigger configuration**
    a. Write Trigger.Ctrl the desired pattern for the subsystems desired to be used in the trigger.
    b. Configure each of the subsystems desired to be used as defined in (a). That is, configure the subsystem deskew delays and coincidence widths on the BCal, FCal, and SSGenPattern peripherals if they are to be used.
6. **Trigger bit configuration**
    a. For disabled TriggerBits, set Trigbit[].Ctrl = 0. For enabled Trigbits, set Trigbit[].Ctrl to enable the logic elements desired for coincidence to create a trigger. The remain lines are for enabled TriggerBits…

b. Set Trigbit[].TrigOutCtrl to desired latency and width (these are probably identical to all trigbits)
c. Now configure the thresholds and parameters for all parts of used trigger logic

7. Starting the run

a. To begin the trigger data flow, Assert SYNC for a minimum of 4us, then release. Data will begin to flow from the FADC->CTP->SSP->GTP on the release of SYNC.
b. Wait for at least the trigger latency time after SYNC was released (a few us)
c. Check each enabled Serdes[].Status bit 14 to see that RXSRCRDYN = 0 indicating trigger data is flowing. If it is a 1, trace back the links to find where the data begins to flow to find the culprit modules. Restart step 7 or earlier depending on the action taken to correct if there was an issue.
d. Check each enabled Trigbit[].Status bit 0 to verify it is 0, if it is a 1 then the latency check has failed and delays/latency requirements likely need to be adjusted
e. At this point the Trigbit[].Scalers should be firing at expected physics rates. The TS input counters should also be firing…Release the triggers (in the TS of coarse)!

6 – Ethernet Interface to GTP

The Ethernet interface is the main path to be used for communication with the GTP. This interface provides the ability to: perform firmware updates, read/write registers, and retrieve buffered copies of scalers. Each of these will be discussed in detail.

6.1 Firmware updates

Firmware for the GTP follows a few steps:

1. After Quartus compilation, use the convert programming files to generate a 128Mbit GTP.pof file from the compiler generated GTP.sof
2. Run 'GTP_build_rbf.sh' to convert the GTP.pof file to GTP.rbf. Note that this .rbf file is slightly different from what you get if you convert the GTP.sof to GTP.rbf in the file converter…This is a critical different so these instructions must be followed!
3. The GTP.rbf generated in step 2 is now copied to the GTP Linux file system. Use the command: 'scp ./GTP.rbf root@dagtp1:~'
4. ssh into the GTP, e.g. 'ssh root@dagtp1'
5. In the root home directory (where the GTP.rbf file was uploaded) run the command: './gtp_burn GTP.rbf'
   This script will copy the GTP.rbf file to the FPGA flash memory. A simple program should be created at some point to do a file comparision between GTP.rbf and /dev/mtdblock5 to verify and reboot the FPGA. Currently a power cycle should be issued to force the GTP to reload the firmware (accessible register do exist to reboot the FPGA, but need to be tested!)

6.2 Register & Scalers Accesses

The GTP runs a TCP based socket server that defines several messages to access registers and scalers. Please refer to the CrateMsgClient.h file for a ROOT based version of the client that provides a good example that can easily be change to run under Linux with 'stdc' libraries. The client establishes a connect that is intended to remain open to issue request. The main protocol features are reading/writing single registers or blocks of registers and reading scalers.

7 - Power Supply and Current Consumption
8 - VXS Pinout Table

VXS Port	15	13	11	9	7	5	3	1
RX0	PP13	PP11	PP9	PP7	PP5	PP3	PP1	PP2
TX0								
RX1		PP15	PP13	PP11	PP9	PP7	PP5	PP3
TX1								
RX2	SSP	SSP	SSP	SSP	SSP	SSP	SSP	SSP
TX2								
RX3								
TX3								
SCL	Busy	Busy	Busy	Busy	Busy	Busy	Busy	Busy
SDA	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp

VXS Port	2	4	6	8	10	12	14	16
RX0	PP4	PP6	PP8	PP10	PP12	PP14	PP16	
TX0								
RX1	PP1	PP2	PP4	PP6	PP8	PP10	PP12	PP14
TX1								
RX2	SSP	SSP	SSP	SSP	SSP	SSP	SSP	SSP
TX2								
RX3								
TX3								
SCL	Busy	Busy	Busy	Busy	Busy	Busy	Busy	Busy
SDA	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp	LinkUp

VXS Port	17	18	B1	B2	B3	B4
RX0	Diff Pair*		Trig 1			Clock
TX0						
RX1			Trig 2			
TX1		TI GTP RX				
RX2			Sync			
TX2						
RX3		TI Busy				
TX3		TI GTP TX				
SCL	SCL*	TI_SCL				
SDA	SDA*	TI_SDA				