

# **D0 Trigger System Description**

## **(Serial Control Crate)**

**Preliminary Draft**  
**September 24, 1999**

**Bill Haynes, Walter Knopf, Thinh Pham,**  
**Neal Wilcer, Ted Zmuda**

# D0 Trigger System

OVERVIEW .....	3
PHYSICAL CONFIGURATION .....	3
CONNECTIVITY .....	4
GENERAL OPERATION .....	5
TRIGGER SYSTEM MODULES .....	6
<i>THC (Trigger Hub Controller)</i> .....	6
VME Interface .....	6
Addressing the THC.....	6
Registers.....	6
Emulation Mode .....	8
<i>Description</i> .....	8
<i>Emulation Mode Data Format</i> .....	9
<i>Operation</i> .....	10
<i>Front Panel Description</i> .....	12
<i>TSC (Trigger Status Concentrator)</i> .....	13
<i>SLF (Serial Link Fanout)</i> .....	16
<i>J3 Backplane</i> .....	16
<i>Serial Transmitter</i> .....	16
<i>Serial Link Receiver Test Module (SRTM)</i> .....	17
VME Interface .....	<b>Error! Bookmark not defined.</b>
Addressing the SRTM.....	<b>Error! Bookmark not defined.</b>
CHANGE HISTORY: .....	28

# D0 Trigger System

## Overview

D0 requested a system be built to transport L1 and L2 signals to 128 geographic sections and to return status signals from the 128 sections to the L1 and L2 frameworks. In addition, these status signals need to be available via VME. The D0 trigger system was designed to satisfy these needs.

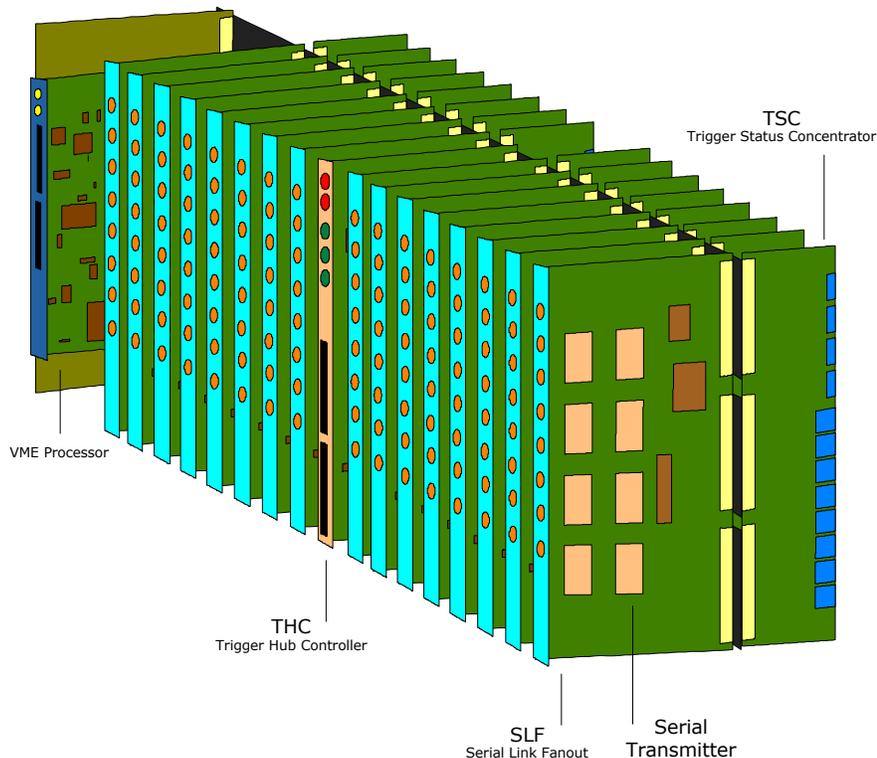
The trigger system consists of 5 modules and a custom VME J3 backplane in a VIPA VME crate. The modules are the THC (Trigger Hub Controller), SLF (Serial Link Fanout), TSC (Trigger Status Concentrator), Serial Transmitter Board, and the Serial Receiver Board.

## Physical Configuration

The trigger system consists of a VIPA crate with a custom J3 backplane, D0's choice of VME controller, 16 SLFs, 16 TSCs, 1 THC, 8 Serial Transmitters mounted on each SLF (128), and 128 Serial Receivers on the 128 geographic destinations. The THC sits in slot 13. Eight SLFs sit to the right of the THC and eight to the left. Each SLF has an associated TSC plugged in behind it in the transition module side of the crate. Slot 1 contains the crate controller, and slot 2,3, and 4 can hold any VME module as long as a 6U to 9U adapter is used. See figure 1. CAUTION – The J3 connections in slot 5 through 21 are customized for the D0 trigger system modules. Do not attempt to insert any 9U non-trigger system modules in these slots. Damage to the system will likely be the result.

Figure 1

## D0 Trigger System Crate Configuration





## General Operation

There are 3 types of data being transferred from the frameworks to the 128 geographic sections:

System wide data – This is information that all 128 sections (channels) need. This data is carried over 2 64-pin cables to the THC, latched and multiplexed, and then sent over the custom J3 backplane to the SLFs.

16 channel group data – Certain data is the same for groups of 16 channels. This data comes from the frameworks over 8 34-pin cables. Each of the cables is plugged into 2 SLFs.

Individual channel data – Data that is specific to a certain channel is brought into each TSC over 3 16-pin cables. This data falls through to the associated SLF via the J2 connector for the 2 modules.

All of this data is latched into the all of the modules using the 7.59 Mhz BSYNC clock. After the data is latched, the THC multiplexes it's data and fans it out to each of the SLFs using 53.104Mhz D0 clock (RFCLK). The SLF will take that data and mix it with the data it received from it's front panel and from it's associated TSC. The end result is a bucket of data that has 7 frames of 16 bit words, with the 7th frame containing parity information. See figure 3.

Figure 3  
SLF Frame Definition

Frame 0

**Bits      Signals**

[15:0]      Reserved for system use.

Frame 1

**Bits      Signals**

[15:0]      Level 1 Accept Qualifier/Level 3 Transfer Number.

Frame 2

**Bits      Signals**

[0]      Level 1 Accept.

[1]      Level 2 Accept.

[2]      Level 2 Reject.

[3]      Marker - Spare.

[4]      Marker – First Period in Turn.

[5]      Marker – Real Beam in Period.

[6]      Marker – Sync Gap.

[7]      Marker – Cosmic Gap.

[15:8]      Current BX Number (1-159).

Frame 3

**Bits      Signals**

[15:0]      Current Turn Number.

Frame 4

**Bits      Signals**

[0]      Level 1 Accept Advisor.

[1]      Level 2 Decision Advisor.

[2]      Initialize Section.

[7:3]      Reserved.

[15:8]      Level 1 BX Number.

Frame 5

**Bits      Signals**

[15:0]      Level 1 Turn Number.

Frame 6

**Bits      Signals**

[15:0]      Parity for Frames 0 thru 5 (generated by SCL Transmitter).

These frames are serialized by the Serial Transmitters, and sent over coax cable to Serial Receivers on each of the 128 geographic sections. The Serial Receivers repackage the data into one 84 bit word that is read out by that geographic section.

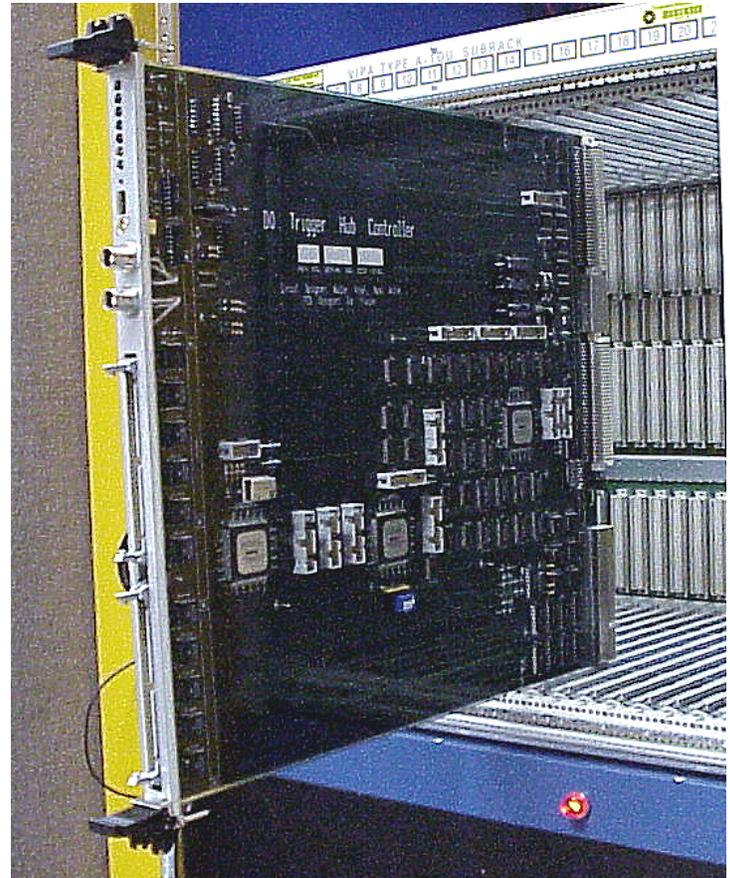
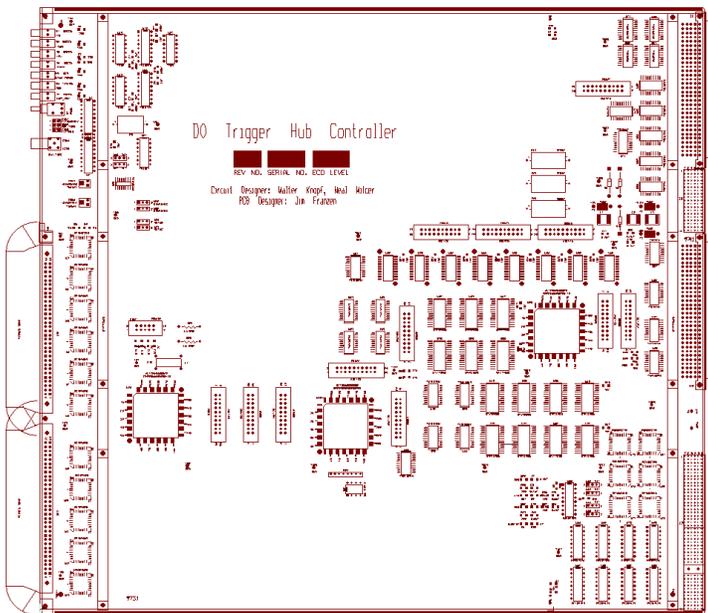
Each of the geographic sections needs to return status information back to the frameworks. In addition to being able to receive serialized data from the Serial Transmitters, the Serial Receivers send status back through a cable to the TSCs. The TSCs take that status information and make all of it available in onboard registers, and sends some of the information directly to the frameworks. These registers are accessible over VME. If necessary, any of the status signals can be set to trigger a VME interrupt.

## Trigger System Modules

### THC (Trigger Hub Controller)

The THC is a 9U x 400mm module that has 3 functions. Its main function is to fan out trigger data from the frameworks to all 16 of the SLFs. Furthermore, it communicates with status registers on the TSCs. Finally, it utilizes onboard SRAM as a source of synthetic data to be used in testing when no actual trigger data is available.

All VME access is through the THC. The THC can communicate with it's own status registers, ram, and the TSC's registers.



### VME Interface

The THC responds only to single transfer A24 D16 VME access with address modifier of 0x39 or 0x3D.

### Addressing the THC

The THC uses a 4-bit dipswitch to set the modules base address. The dipswitch setting is compared to the A20..A23 address lines during VME access.

### Registers

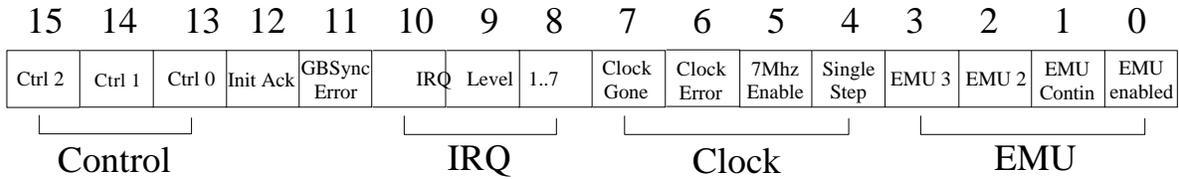
THC Registers – *base addr + 0x0 through base addr + 0xFE*

A “1” is active, “0” is inactive for all registers.

**READ ONLY**

Status Register – base addr + 0x0

Status Register is read only. A “1” in a bit field signifies active. Values read from the Status Register correspond to values written to the Write Only registers listed below with the exception of the “Init Ack”, “GB Sync Err”, “Clock Gone”, and “Clock Error” bits. These bits are fed directly from onboard THC signals. Default (after reset) value of the Status Register with no clock signals attached to the THC front panel is 0x00C0.



**WRITE ONLY**

Control Set Register – base addr + 0x0

Control Reset Register – base addr + 0x2

- Control[0], Store current ram data
- Control[1], Recall stored ram data.
- Control[2], Not defined

Clock Set Register – base addr + 0x4

Clock Reset Register – base addr + 0x6

- Clock[0], Single Step Mode
- Clock[1], Enable 7Mhz clock
- Clock[2..3], Not writeable

Emulation Set Register – base addr + 0x8

Emulation Reset Register – base addr + 0xA

- EMU[0], Emulation Mode enable
- EMU[1], Emulation Mode runs continuously
- EMU[2..3], Currently undefined

Emulation Pulse Register – base addr + 0xC

- EMU\_Pulse [0], Pulse start emulation mode

Interrupt Request Level Register – base addr + 0xE

- IRQ Level[0..2], Set THC’s IRQ request level, 1 through 7

Reserved – base addr + 0x10 through base addr + 0x3FE

**Definition of register bits**

Control[0..2] –

Control[0] - Store ram data. When a “1” is written to this location a sequence of ram reads occurs that takes the current data in the ram and stores it into the ram’s onboard eeprom. Writing a “1” to this location results in a pulse that starts the store sequence. This location is not readable via the status register.

Control[1] – Recall stored ram data. When a “1” is written to this location, a sequence of ram reads occurs, resulting in a movement of the data that is stored in the ram’s eeprom into the ram. Writing a “1” to this location results in a pulse that starts the recall sequence. This location is not readable via the status register.

Control[2] – Not defined

Clock[0..3]

Clock[0] – Single Step Mode. This mode allows the user to use an external clock plugged into the front panel differential ECL ECLK. Emulation mode must be active (EMU[0] = 1).

Clock[1] – 7Mhz Clock Enable. When this value is “0”, no 7Mhz clocks will be fed to the serial link fanouts. This allows the serial link transmitters and receivers time to “sync up”. When a “1” is written to this location, the 7Mhz clock is sent to SLFs, and whatever data is present at the THC front panel or in emulation ram will be sent. In the event of a “sync error”, this bit can be reset to “0”, allowing the receiver/transmitter to sync up again.

Clock[2] – Clock Error. A comparator circuit on the THC checks the RFCLK at the front panel for missing ticks. If more than a couple ticks are missing, this bit will go active. It is not writeable from VME, and cannot be reset. The bit will go inactive once the RFCLK is working correctly.

Clock[3] – Clock Gone. If the RFCLK completely disappears, this bit will go active. It is not writeable from VME, and cannot be reset. The bit will go inactive once the RFCLK is working correctly.

#### EMU\_Pulse[0]

If EMU[0] is active, writing a “1” to this location will start emulation mode.

#### EMU[0..3]

EMU[0] – Emulation mode enable. This bit must be set before any of the diagnostic features can be utilized.

EMU[1] – Emulation Continuous. Allows the emulation mode to run continuously until a zero is written to this location.

EMU[2..3] - Currently undefined.

#### IRQ[0..2]

This allows the user to select the IRQ interrupt level he/she wants to generate. Valid values are 1 through 7.

*Emulation Ram – base addr + 0x80000 through base addr + 0x9FFFC*

*Reserved – base addr + 0xA0000 through base addr + 0xFFFFE*

The Emulation Ram is non-volatile ram SRAM that can be used as regular ram or non-volatile depending on what the user’s needs are. When used as regular ram, addressing and data transfers are done by directly addressing the ram. When used as non-volatile ram, the user enters the data he/she wants saved into the ram. Once all data is written to the ram, the user must *store* that data in the ram’s onboard eeprom. When the stored data needs to be loaded back into the ram, the user must do a *recall*. Both the store and recall are done using a sequence of ram address reads. This procedure has been simplified so that all is needed for a ram store to occur is to write control register 0 with a “1”. A recall is done by writing a “2” to control register 0.

Note:

Due to physical ram configuration, address base addr + 0x8XXxE or 0x9XXxE does not exist and will yield invalid data. Accessing these addresses will *not* result in a VME bus error, however.

### *Emulation Mode*

#### **Description**

Emulation Mode allows testing of the trigger information path through the THC, onto the backplane, in to the SLF, through the transmitters to the receivers. If an SRTM (Serial Receiver Test Module) is used, data can be read out over VME, or viewed with a logic analyzer using the analyzer connectors on the SRTM.

The data used in emulation mode is contained in the THC onboard RAM. The THC during emulation mode generates the 53Mhz and 7Mhz clocks needed for data transfer. Ram data is loaded using VME writes to the addresses specified in the previous section, and can be saved by storing it in non-volatile ram. The THC will reload the stored data into ram during power up.

### **Emulation Mode Data Format**

Due to physical ram configuration, timing within the THC, THC sequencer op codes, and data packet format, the ram onboard the THC needs to be loaded in a certain way so that what is viewed at the receiver end makes sense. Two charts are shown below. The upper chart is the data format that needs to be loaded into the THC. The chart below that is the format for the data that will be read out of the SRTM (if used) once the THC is triggered. Note that the SRTM module id of 0xA is used in the chart's example.

Data format consists of a "packet" whose size is determined by how many "frames" of data are sent. A "frame" of data is a group of seven 16-bit data word "buckets" that comes every 132ns. During actual online data taking, data from bucket 1 is considered reserved, or unusable. This holds true for emulation mode also. Bucket 7 is the parity bucket for that frame that is generated by the SLF. The THC uses the equivalent bucket 7 address to hold the sequencer op codes that control emulation operation. That leaves 5 buckets of genuine data that the THC has to generate.

Using the first chart below, the frame construction is as follows:

**Addr 0x8XXXX0 – Bucket 1** Reserved, unusable, don't care  
**Addr 0x8XXXX2 – Bucket 2** 1<sup>st</sup> true data word  
**Addr 0x8XXXX4 – Bucket 3** 2<sup>nd</sup> true data word  
**Addr 0x8XXXX6 – Bucket 4** 3<sup>rd</sup> true data word  
**Addr 0x8XXXX8 – Bucket 5** 4<sup>th</sup> true data word  
**Addr 0x8XXXXA – Bucket 6** 5<sup>th</sup> true data word  
**Addr 0x8XXXXC – Bucket 7** Parity word generated by SLF. The THC uses this location for the emulation mode's sequencer op codes.  
**Addr 0x8XXXXE** – Non-existent ram, illegal address.

Note that "don't care" data is also in the first and last "frame". The THC timing can't guarantee that that first and last frame will contain correct data. Be sure to take this fact into account when analyzing the data at the receiver end.

There are multiple op codes available for emulation mode, but the set that is used most is shown below. Notice address 0x80000C has a value of 0x1001. This is the op code value that starts the sequencer operating. Address 0x8000FC has a value of 0x4000 that tells the sequencer to stop sending data. The value of 0x1001 should always be entered into address 0x80000C, but the sequencer op code of 0x4000 can be positioned at any 0x8XXXXC address, depending on when you want the sequencer to stop sending data. All values between the 0x1001 and 0x4000 must always be 0x0000.

## Data Format for the THC

	BKT#1	BKT#2	BKT#3	BKT#4	BKT#5	BKT#6	BKT#7	
	0	2	4	6	8	A	C	E
800000	don't care	0x1001	ill addr					
Frame# 0 800010	don't care	data0	data1	data2	data3	data4	0x0000	ill addr
Frame# 1 800020	don't care	data5	data6	data7	data8	data9	0x0000	ill addr
Frame# 2 800030	don't care	data10	data11	data12	data13	data14	0x0000	ill addr
Frame# 3 800040	don't care	data15	data16	data17	data18	data19	0x0000	ill addr
Frame# 4 800050	don't care	data20	data21	data22	data23	data24	0x0000	ill addr
Frame# 5 800060	don't care	data25	data26	data27	data28	data29	0x0000	ill addr
Frame# 6 800070	don't care	data30	data31	data32	data33	data34	0x0000	ill addr
Frame# 7 800080	don't care	data35	data36	data37	data38	data39	0x0000	ill addr
Frame# 8 800090	don't care	data40	data41	data42	data43	data44	0x0000	ill addr
Frame# 9 8000A0	don't care	data45	data46	data47	data48	data49	0x0000	ill addr
Frame# 10 8000B0	don't care	data50	data51	data52	data53	data54	0x0000	ill addr
Frame# 11 8000C0	don't care	data55	data56	data57	data58	data59	0x0000	ill addr
Frame# 12 8000D0	don't care	data60	data61	data62	data63	data64	0x0000	ill addr
Frame# 13 8000E0	don't care	data65	data66	data67	data68	data69	0x0000	ill addr
8000F0	don't care	0x4000	ill addr					

## SRTM Data Format

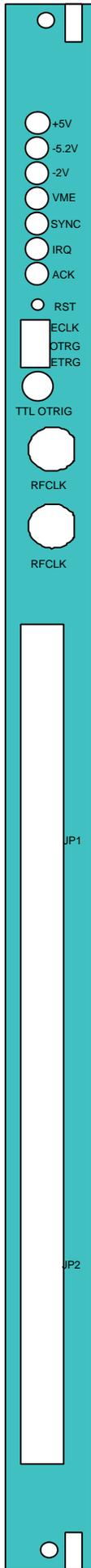
	A00000	A00200	A00400	A00600	A00800
0	data0	data1	data2	data3	data4
2	data5	data6	data7	data8	data9
4	data10	data11	data12	data13	data14
6	data15	data16	data17	data18	data19
8	data20	data21	data22	data23	data24
A	data25	data26	data27	data28	data29
C	data30	data31	data32	data33	data34
E	data35	data36	data37	data38	data39
10	data40	data41	data42	data43	data44
12	data45	data46	data47	data48	data49
14	data50	data51	data52	data53	data54
16	data55	data56	data57	data58	data59
18	data60	data61	data62	data63	data64
1A	data65	data66	data67	data68	data69

### Operation

Assuming you have all modules inserted, the dip switch on the THC set for address 0x8, and cables attached, initialization and running of the emulation mode requires the following steps:

1. Load the emulation data you wish to run, including sequencer op codes, in to the THC ram (or use what you have saved in the non-volatile ram).
2. Start the 7Mhz clock – Write 0x2 to location 0x800004
3. Enable emulation mode – Write 0x1 to location 0x800008
4. Trigger emulation – Write 0x1 to location 0x80000C

This will send a burst of data that can be retrieved from SRTM ram or, if designed to do so, by the user's end module. For scope or analyzer looping, the data can be sent repeatedly by changing the value in step 3 to a value of 0x3. This enables emulation mode and continuous data. Writing a reset bit to the "emulation reset register" can halt looping – Write 0x2 to location 0x80000A.



## Front Panel Description

### LEDs

+5V	+5 volt power OK when lit. Green
-5.2V	-5.2 volt power OK when lit. Green
-2V	-2 volt power OK when lit. Green
VME	Flashes when a DTACK was transmitted. Green
SYNC	Synchronization error detected by any serial link. Red
IRQ	VME interrupt request active. Yellow
ACK	Init acknowledge received from any serial receiver. Green
RST	Board reset pushbutton

### 2-pin Connectors

ECLK	Allows and external differential ECL clock to be fed into THC
OTRG	Outputs a differential ECL pulse signifying the start of emulation mode.
ETRG	Input for differential ECL pulse to trigger emulation mode.

TTL OTRIG TTL version of OTRG

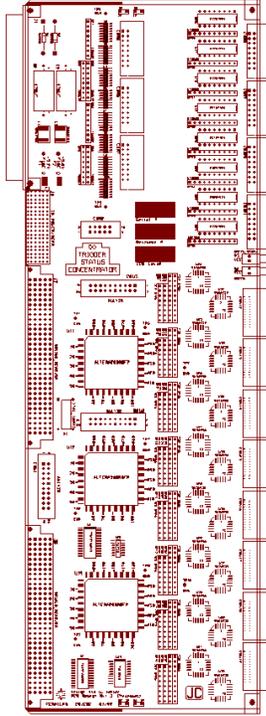
### Twinaxial Connectors

BSYNC	7.59Mhz secondary clock
RFCLK	53.104Mhz system clock

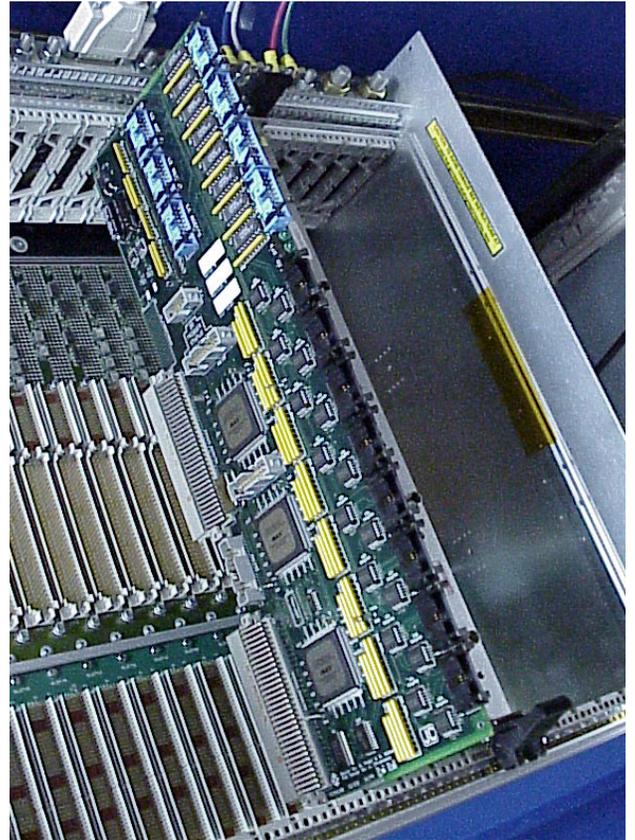
### 64-pin Connectors

JP1,2	Differential ECL data from the frameworks
-------	---

## TSC (Trigger Status Concentrator)



The TSC is a 9U x 120mm transition module that has 3 functions. Status information from groups of eighth Serial Receivers is passed into each TSC. This status information is loaded into registers on the TSC and is available over VME. Some of the status signals get transferred back to the frameworks. When enabled, any of these status signals can be programmed to initiate a VME interrupt. Lastly, framework data is loaded into the TSC



that passes it directly to it's associated SLF.

### Registers

Each TSC holds status information for eight channels. Each channel has 3 registers – Status, Interrupt Mask, and Interrupt. All registers are read/write. There is also one register per TSC that can contain board level information. This, too, is read/write.

All register data is read and written using the THC's base address along with an offset described below. Data transfer is done through the THC, onto the custom J3 backplane, ending at the TSC.

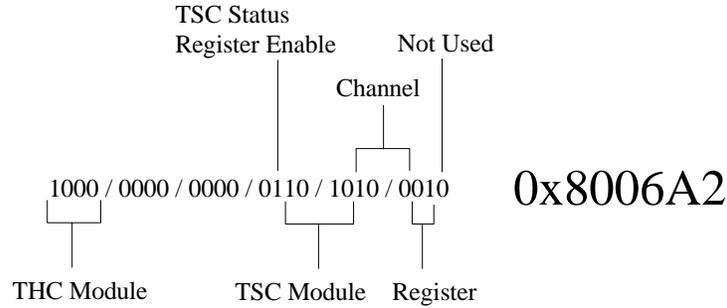
Each status signal connector is 26-pin .050" pitch. The status signals are defined as follows for each of the eight connectors:

- Bit 0 – Level 1 Busy**
- Bit 1 – Level 1 Error**
- Bit 2 - Level 2 Busy**
- Bit 3 – Level 2 Error**
- Bit 4 – GB Sync Error**
- Bit 5 – Init-Ack**
- Bit 6 – Spare1**
- Bit 7 – Spare2**
- Bit 8 – Cable Disconnected**
- Bit 9 thru Bit 25 - Reserved**

### **TSC Registers** – *base addr + 0x400 through base addr + 0x7FE*

The 128 geographic sections (or channels) are connected to 16 Trigger Status Concentrators (TSC) modules that have 8 channels of status information per module. Each channel has 3 readable/writeable registers. In addition, each TSC has one register for board specific status. Access to the registers is achieved by combining slot location, channel, and register number information with the THC base address and setting the TSC status register enable bit. Example for

accessing register 1 of channel 4 on TSC 10 with a THC base address of 0x800000:



### Channel Status Register

*THC Module base addr + TSC Status Enable Bit + TSC Module + Channel + 0x0.*

This register contains the current state of the status signals. Read this register to view the current state of the status signals. Though not useful for online operation, this register can be written to for diagnostic purposes. Before normal use, this register should be reset to 0x0000.

### Channel Interrupt Mask Register

*THC Module base addr + TSC Status Enable Bit + TSC Module + Channel + 0x2.*

Writing a one in any of bit locations enables the associated status signal to generate an interrupt. Writing a zero will disable interrupts for that status signal.

### Channel Interrupt Request Register

*THC Module base addr + TSC Status Enable Bit + TSC Module + Channel + 0x4.*

If a status signal goes active (high) and the associated interrupt mask register bit is enabled, the rising edge of the status signal will result in a VME interrupt. This interrupt is latched, and can only be cleared by writing a “0” to this bit, or by a reset. Once a bit in this register is cleared, the only way another interrupt will be generated by that status signal is if the signal changes from inactive to active. If a signal is still active when you clear the interrupt, another interrupt will not be generated.

### Board Status Register

*THC Module base addr + TSC Status Enable Bit + TSC Module + (Any) Channel + 0x6.*

Currently, only bit zero is defined in this register. If bit zero is high, interrupts from this TSC are enabled. If bit zero is low, this TSC will not generate any interrupts.

Reserved – base addr + 0x800 through base addr + 0x7FFFE

See chart 1 at the end of this document for a list of all TSC addresses.

The registers on the TSC are in conformance with the “Serial Link Interface Module” specification, revision 15/Jan/98. The spec requires 3 registers per channel, and 1 board wide register. The channel registers view the status signals coming from the serial receivers and load them into the status register and if requested to, into the interrupt registers. The board wide register has, at this time, only one function. That function is to allow or disallow interrupts from the TSC.

An example of register usage –

1. Set the global interrupt enable to allow interrupts from the TSC. This allows interrupts (if individual channel interrupts are enabled) to be generated over VME. Note: Once this bit is set (or reset) for the particular TSC, it doesn't have to be written to again

*Write data value 0x0001 to address 0x800406 (for TSC 0).*

2. Write the interrupt request level you want to go active in event of an interrupt. This is the THC Interrupt Request Level register. Assume the request level you want is 6. Set this value once and forget it until the THC is reset, powered down, or the value is overwritten.

*Write data value 0x0006 to address 0x80000E (for any interrupt from any TSC).*

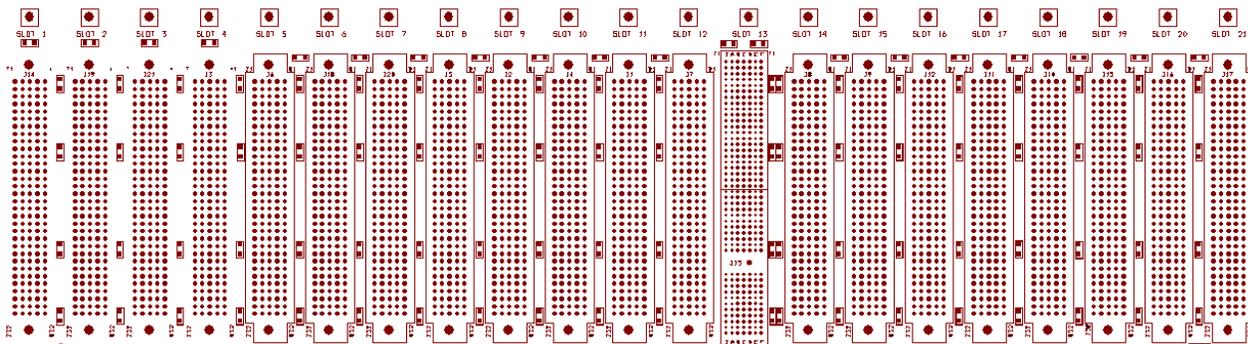
3. If you want a “SYNC ERR” from status signal group 0 on TSC 0 to generate an interrupt (assuming all registers start at a value of 0x0, and the THC module address is 0x8), you would:

Set the interrupt mask register to allow a SYNC ERR to generate an interrupt.

*Write data value 0x0010 to address 0x800402.*

4. When the SYNC ERR signal goes high on status signal group 0, an IRQ6 will be generated on VME and out the THC's front panel. Additionally, an LED will light on the front panel. All interrupt requests are latched and can only be reset with a VME reset, front panel reset, or a write of '0' to the offending bit in the TSC's interrupt request register (as per the SLIM spec). The interrupt request register can be read as well as written. Therefore, once the interrupt request register that is activating the IRQ is found, the interrupt can be cleared by:

*Write data value 0xFFFE to address 0x800404.*



## SLF (Serial Link Fanout)

### J3 Backplane

In order to transfer framework data and clocks from the THC to the SLFs and access the status registers on the TSCs, a custom J3 backplane had to be designed.

The backplane is laid out so that (1) slots 1 – 4 have no signal or plane connection, (2) Slots 5 – 12 and 14 – 21 house 16 SLFs, and (3) slot 13 holds the THC. All of the backplane connectors are the 160-pin din variety with the exception of slot 13, which utilizes the 2mm hard metric connector. In addition, this slot is keyed to allow only the THC to be plugged in. All of the 160-pin din connectors have long pins protruding through the rear of the backplane that make it possible for a 160-pin shell to be installed on the backside to allow the TSCs to be plugged in.

The THC generates a differential ECL RFCLK and BSYNC clock for each of the SLFs. These clocks are passed through the 2mm connector in slot 13 to all of the SLFs. Trace lengths on the backplane have been matched to provide minimum skew between clock edges for all slots. These clocks are terminated on each SLF into  $-2V$ . The THC also provides single ended ECL framework data to the SLFs. In the case of this data, however, there is a left and right path for 2 copies of the data. The left path feeds SLFs 1 – 8, and the right path feeds SLFs 9 – 16. Termination for the ECL signals is to  $-2V$  on the backplane itself. Finally, the THC talks to status registers on the TSCs via this custom J3 backplane. All 16 of the TSCs are read or written to through this bus. All 16 of the TSCs monitor the status enable and address lines. If status is enabled, and they are being addressed, that particular TSC will enable itself for data transfers.

Power for the  $-2V$  used in terminating the data lines is provided by a fused circuit on the THC and fed to the backplane via the 2mm connector.

### Serial Transmitter

Reference University of Arizona doc – *I/O Specifications for Serial Transmitter Daughter Board (PCB-0140-XMIT) Aug 21, 1997 Rev 2.1*

**Serial Receiver** (extracted from D0 SCL Receiver Preliminary Specification, with thanks to Mark Bowden)

The SCL receiver duplicates the University of Arizona serial data link receiver, with additional on-card demultiplexing. The demultiplexing logic consists of a single Altera 9xxx (208 pin QFP) which will be in-circuit reprogrammable through JTAG. An on-board 53.1 MHz oscillator will provide the SCL receiver local reference clock. Available standard frequencies of 53.125 and 53.088 MHz may be suitable (although they exceed the  $\pm 100$  PPM receiver specification). The CLK\_53 output will be driven by the local reference clock when the receiver is not synchronized. +3.3 Volt power for the receiver will be supplied by an on-board linear regulator. The front panel status connector will be (subject to change) a 21 pin Micro-D Subminiature (<http://www.molex.com/product/ultimate/8-18.html>). This provides 16 pins for the eight differential RS-485 status signals plus 5 pins for a JTAG front-panel interface. Drivers for the eight RS-485 signals will be bi-directional (they may be programmed as either input control, or output

status signals). All outputs will be valid 15 ns before the rising edge of CLK\_7 and will remain valid for 15 ns after the rising edge of CLK\_7.

## Serial Link Receiver Test Module (SRTM)

SRTM Vers 2.4

### Features

- Single Width 6U VIPA Module
- Supports Only Single Word Transfers ( 24-bit Address, 16-bit Data )
- Functionality Implemented in One FPGA (VME & SCLR Interface, Memory and Control)
- Universal VIPA Interface Module Capability (Add Daughter Card)
- Initially Designed for the D0 Trigger Distribution System
- On-The-Fly BER Testing of 1-Gbit Serial Control Link Receiver (SCLR)
- Supports Slow Computer BER Testing (Write to Transmitter Memory, Read From SRTM Memory)
- Capable of Non-stop BER Testing for 312-yrs (48-bit Loop Counter @ 35us/Loop)
- 16-bit Error Counters (Frame Error & Sync Error)
- Reports Location of First Error In Data Stream (16-bit Error Summary Register)
- Supports Multiple SRTMs for Testing up to 16-Channels of 1-Gbit Serial Links.

The Serial Link Receiver Test Module (SRTM) is a 6U VIPA module designed to check the integrity of the 1Gbps serial link transmitters & receivers via coaxial cable for the D0 Trigger Distribution System. The SRTM permits a serial block transfer of a length up to 256 80-bit data frames to be checked on the fly (in hardware) or under program control. One data frame consists of five 16-bit words or buckets (corresponding to the accelerator beam bucket). The results of the serial command link receiver frame parity check (SCL\_DataError) are stored in bit3 of the Bucket4 receiver memory on a frame by frame basis. A data parity error is stored as a logic high. A simplified block diagram is shown in figure 1.

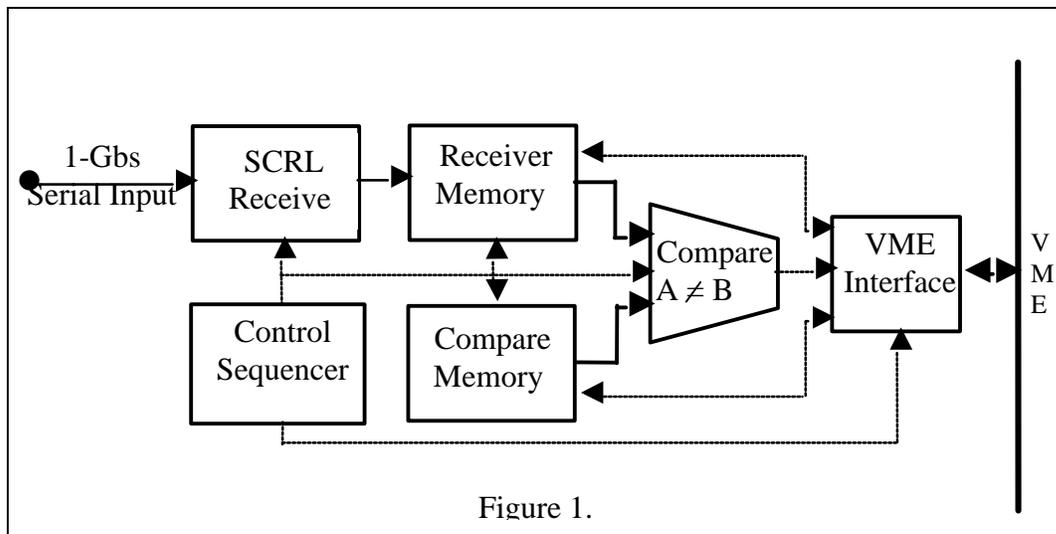


Figure 1.

### On-The-Fly Bit Error Rate (BER) Testing

The On-the-Fly mode of operation performs hardware data checking of the 1-Gbs serial data from the Transmitter to the Receiver at full speed with minimal serial data interruptions (except for the approx. 80ns burst retrigger time). In this mode the SRTM uses two memory banks; one called the receiver data memory and the other the compare memory. The receiver memory stores the incoming 1-Gbs serial data in 80-bit frames of five 16-bit buckets at a 7Mhz rate (beam sync). The compare memory is loaded with a copy of the transmitted data and compared to the receiver memory data on a frame by frame (bucket by bucket) basis. The statistics of the data comparisons are provided in several VME registers & counters to determine the ongoing Bit Error Rate (BER) which also helps in system level debugging.

The SRTM was designed to permit up to sixteen 1-Gbs links to be tested simultaneously by daisy-chaining the Done pulse through each of the SRTMs as shown in the simplified connection diagram of figure 2. This synchronizes the block data transfers of the system by forcing the SRTM on the right to wait for the SRTM on the left until the received data is tested.

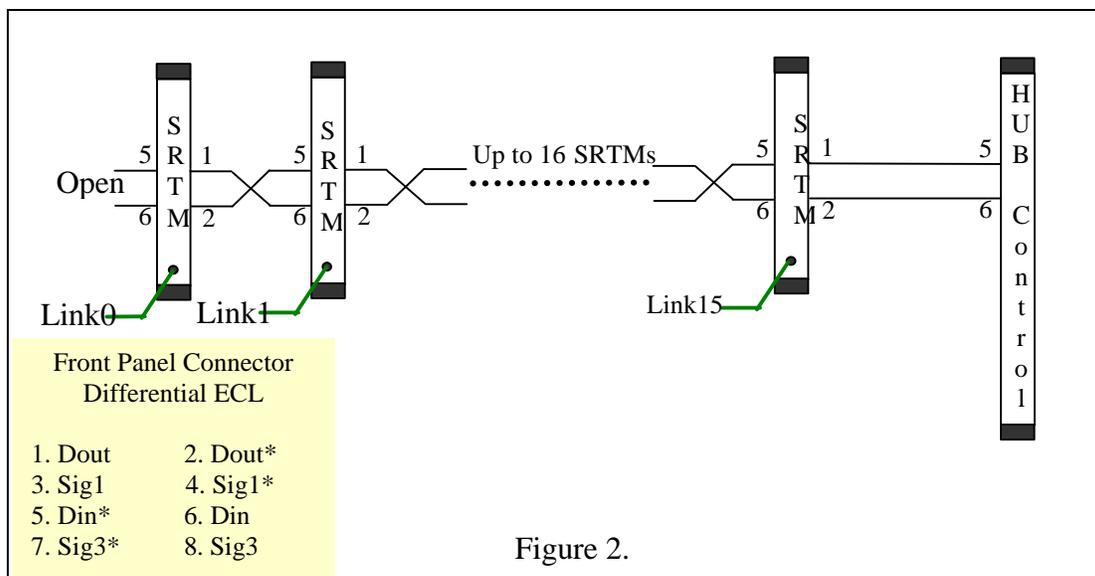


Figure 2.

The following basic SRTM procedure (written in Visual Basic for MS Excel, file = SRTM\_Sync\_tst.xls) controls the on-the-fly BER testing:

- 1) Write a block of data into the HUB Controller (up to 255 80-bit words).
- 2) Write the same block of data into the SRTM Compare memory (up to 255 80-bit words).
  - Due to the unpredictable results of the a header and a trailer frames of the Trigger Hub Controller (THC), in emulation mode, the on-the-fly data checking for these two frames must be masked off. To accomplish this, bit4 of the Bucket4 compare memory is asserted to disable the data checking at the locations of the header and trailer frames. For example, to disable data checking for frames 1 & 10 simply set bit4 of Bucket4 Compare memory locations 1 & 10. The remaining locations in

- the block of memory, where the data is to be checked, bit4 should be set to a logical zero.
- 3) Set the bits CSR0[12] (Arm SCLR), CSR0[11] ( Loop Mode) and optionally CSR0[10] (Error Mode).
    - Arm SCLR arms the SRTM module to store data from the SCLR receiver into receiver memory.
    - Loop mode causes the SRTM to automatically rearm itself to store the next block of data from the SCLR.
    - Error Mode causes the SRTM to stop at the end of the current data block when a compare error occurs after storing the first frame# and bucket # of the error in CSR6 (0xN81014). Where 'N' is the VME base address selectable via an on board 4-position DIP Switch.
  - 4) Trigger the HUB Controller to send the block of data in continuous mode.
  - 5) Read the statistics from the VME counters and registers (0xN81004 to 0xN81018) and calculate BER.
    - Loop Counter (0xN81004 to 0xN8100C) is incremented at the end of each block store in receiver memory.
    - Frame Error (0xN81010) is incremented for each frame error.
    - Error Summary Register (0xN81014) contains the first the location in memory of the first error within the block of data received. This is most useful in Stop-on-Error Mode.
    - Sync Error Counter (0xN81018) contains a count of Sync Errors the SCLR receiver has reported.
  - 6) Repeat step 5 as often as necessary.
  - 7) Loop back to step 1 to change the data pattern frequently (currentlt set to once per minute). Before looping back to step 1:
    - Disable the Hub Controller from sending data.
    - Check the Done bit of CSR0: if it is high, the current block is complete and it is ok to loop back to step1.

It takes about 35us to transfer a burst of 250 words (80-bits ea) to the SRTM data memory. This means that a BER of better than  $1 \times 10^{-14}$  can be accomplished in 2.07 days and  $1 \times 10^{-15}$  is possible in 20.7days assuming no errors occur. What can be accomplished in a few days in this (on-the-fly) mode would require years in program control mode.

### **Program Control Mode**

Program control is the simplest receiver test mode; where the computer writes a block of data into the HUB controller and then reads the data transferred to the SRTM's memory and compares it with the data written. All statistics (under program control) is left to the imagination of the programmer. The following basic procedure describes what is required by the program:

- 8) Write a block of data into the HUB Controller (up to 255 80-bit words).
- 9) Arm the SRTM for data taking mode and stop on error (write 0x1000 to CSR #0xN81000).
- 10) Trigger the HUB Controller to send the block of data.
- 11) Read the data from the SRTM memory (read 0xN80000 to 0xN80000 + #words\_sent).
- 12) Compare data read with data send.
- 13) Repeat steps 1 – 5 to accumulate error statistics.

## Version History

**Version 2.4** of the firmware incorporates a hard coded bit(19) of the VME base address to allow up to 16 SRTM modules to be connected in one VME crate. This results in a change of the VME base address of CSR0 to 0xN81000, where 'N' is selectable via an on board 4-position DIP Switch. As a result of Version 2.3, the serial link receiver firmware was changed to assert the Link\_Error signal when three consecutive data errors occur. Therefore, the Link\_Error check was removed from the SRTM.

**Version 2.3** of the firmware incorporates an additional counter (Sync Error Counter) to display the statistics of both data errors and sync errors. Also, firmware was added to increment the Sync Error Counter when three consecutive data errors occur from the serial receiver. To perform this test, the following procedure should be followed using the excel test procedures written by Bob Angstadt:

1. Use Excel program SRTM\_Sync\_Tst.xls on worksheet tab Acktst.
  - Issue a VME Reset.
  - Download a random pattern into the Hub Controller.
  - Start the 7Mhz oscillator on the Hub Controller.
  - Start the Hub Controller in a continuous send loop.
  - Set the Arm-Slr, Loop-Mode & Err\_Mode bits in CSR0 (N01000).
  - Read the Loop Counter (N01004 & N01008) and maximum receiver reset time (N01010 & N01014). The Loop Counter gives the ongoing counter value of the current reset time. The register at N01010 & N01014 contains the maximum time (in 53Mhz pulse) required to resync the receiver after a reset is issued by the SRTM in this test mode.
  - Continue to issue the above to display an update of the maximum reset time.

**Version 2.2** of the SRTM firmware was designed to test the maximum time required to reset the Fermilab 1-Gbs serial link receiver (based on the University of Arizona design). The maximum time for the receiver to resync after a reset is contained in the register at locations N01010 (High) & N01014 (Low). This test should be run for several hours to determine the worst case resync time. To perform this test, the following procedure should be followed using the excel test procedures written by Bob Angstadt:

1. Use Excel program SRTM\_Sync\_Tst.xls on worksheet tab Acktst.
  - Issue a VME Reset.
  - Download a random pattern into the Hub Controller.
  - Start the 7Mhz oscillator on the Hub Controller.
  - Start the Hub Controller in a continuous send loop.
  - Set the Arm-Slr, Loop-Mode & Err\_Mode bits in CSR0 (N01000).
  - Read the Loop Counter (N81004 & N81008) and maximum receiver reset time (N01010 & X01014). The Loop Counter gives the ongoing counter value of the current reset time. The register at N01010 & N01014 contains the maximum time (in 53Mhz pulse) required to resync the receiver after a reset is issued by the SRTM in this test mode.
2. Then continue to issue above read to get an update of the maximum reset time.

Based on running the above test over night it was determined that the maximum time required to resync after a reset (assertion of SCL\_Ack) was **10.1sec.** with data only and no sync patterns. However, when sync frames were supplied, the maximum time in an over night test was **12.3ms.**

### VME Registers

**VME Register/Memory Space (N8m000h):** Where N is selectable via a four position DIP switch, when m=1 selects register space, when m=0 selects receiver memory or when m=2 selects compare memory.

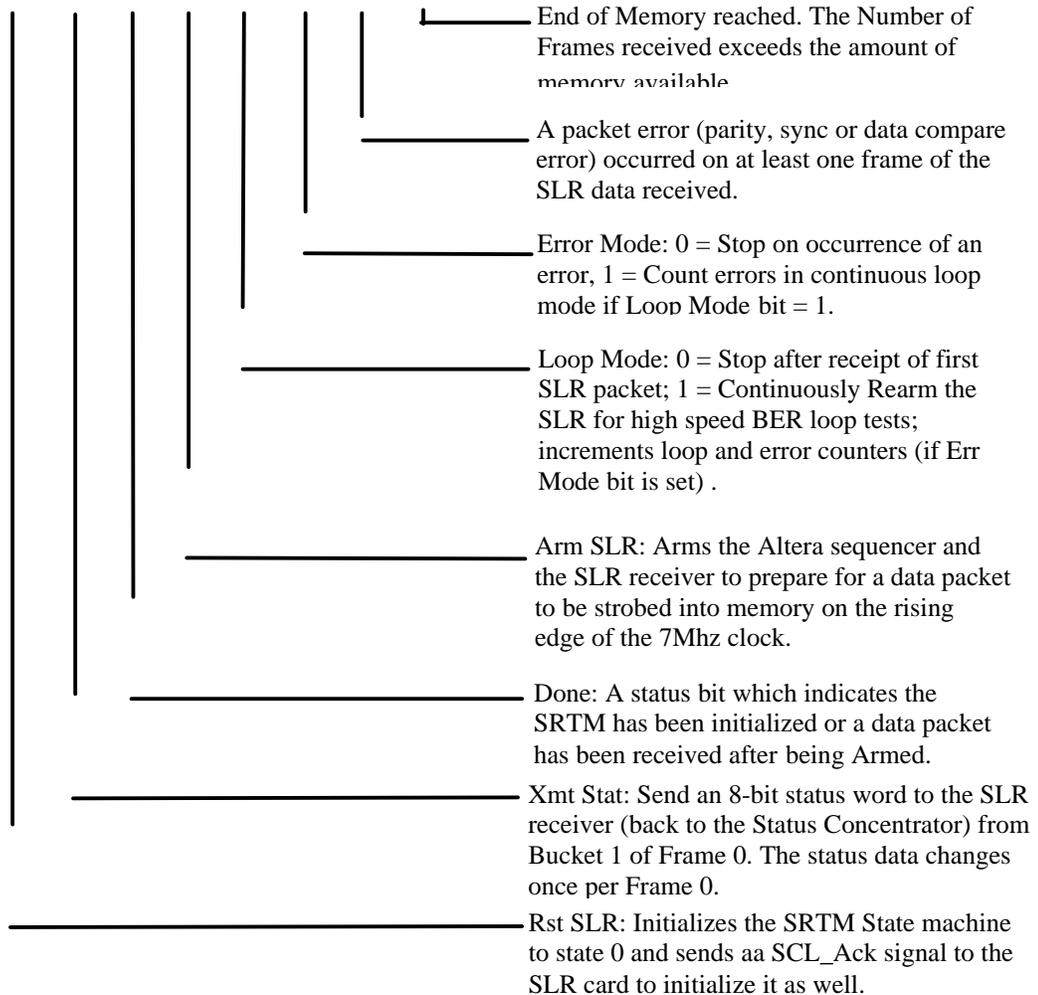
VME Address Format																								
23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
Logical Address				Logical Zeros								R	M	B	B	B	Internal Module Address: CSR # or SLR Frame Memory							
												e	e	u	u	u								
												c	m	c	c	c								
												-	-	k	k	k								
												C	R	e	e	e								
												m	e	t	t	t								
												P	g	2	1	0								



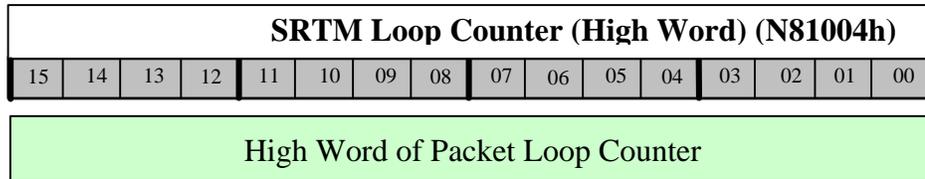
## High Speed SRTM Registers and Memory

**VME Control and Status Register (N81000h):** Where X selectable via a four position DIP switch. All bits indicated in yellow of this register are setable via writing bit combination to N81000 and resetable via writing the bit combination to N81002.

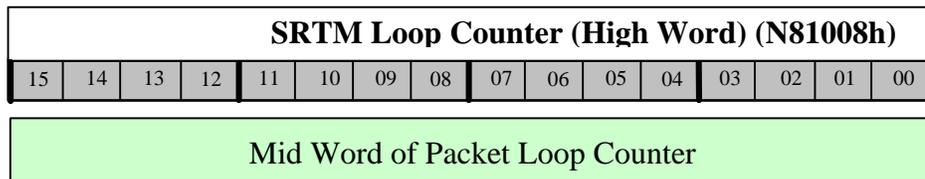
VME Control & Status Register (N81000h)															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
R S T - S L R	X t m  S t a t	D o n e	A r m  S L R	L o o p  M o d e	E r r  M o d e	P a c k e t - E r r	E n d  O f  M e m	Number of SLR Frames Received							



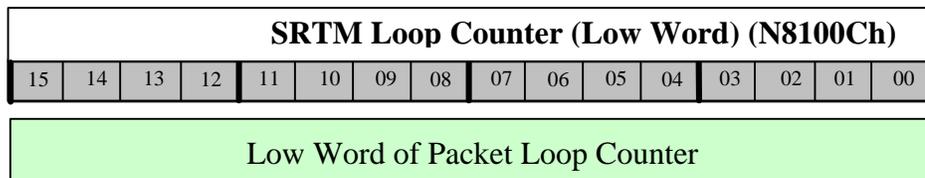
**SRTM High Word of Loop Counter (N81004h):** Where 'N' selectable via a four position DIP switch.



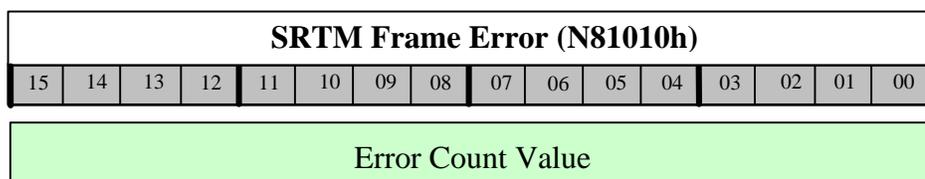
**SRTM Mid Word of Loop Counter (N81008h):** Where 'N' selectable via a four position DIP.



**SRTM Low Word of Loop Counter (N8100Ch):** Where 'N' selectable via a four position DIP.



**SRTM Frame Error Counter (N81010h):** Where 'N' selectable via a four position DIP switch.



**SRTM Error Summary Register (N81014h):** Where ‘N’ selectable via a four position DIP switch.

SRTM Error Summary Register (N81014h)															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
P	S	D	B	B	B	B	B	Address of The First Error Within the Packet							
k	y	a	k	k	k	k	k								
t	n	t	t	t	t	t	t								
E	E	E	E	E	E	E	E								
r	r	r	r	r	r	r	r								
r	r	r	r	r	r	r	r								

**SRTM Sync Error Counter (N81018h):** Where ‘N’ selectable via a four position DIP switch.

SRTM Sync Error (N81018h)															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Sync Error Count Value															

## Chart1

**TSC address list (assumes THC base address of 0x800000)**

	Status	Interrupt Mask	Interrupt Request	TSC Interrupt Enable
TSC0				800406
Chan0	800400	800402	800404	
Chan1	800408	80040A	80040C	
Chan2	800410	800412	800414	
Chan3	800418	80041A	80041C	
Chan4	800420	800422	800424	
Chan5	800428	80042A	80042C	
Chan6	800430	800432	800434	

TSC1	Chan7	800438	80043A	80043C	800446
	Chan0	800440	800442	800444	
	Chan1	800448	80044A	80044C	
	Chan2	800450	800452	800454	
	Chan3	800458	80045A	80045C	
	Chan4	800460	800462	800464	
	Chan5	800468	80046A	80046C	
	Chan6	800470	800472	800474	
TSC2	Chan7	800478	80047A	80047C	800486
	Chan0	800480	800482	800484	
	Chan1	800488	80048A	80048C	
	Chan2	800490	800492	800494	
	Chan3	800498	80049A	80049C	
	Chan4	8004A0	8004A2	8004A4	
	Chan5	8004A8	8004AA	8004AC	
	Chan6	8004B0	8004B2	8004B4	
TSC3	Chan7	8004B8	8004BA	8004BC	8004C6
	Chan0	8004C0	8004C2	8004C4	
	Chan1	8004C8	8004CA	8004CC	
	Chan2	8004D0	8004D2	8004D4	
	Chan3	8004D8	8004DA	8004DC	
	Chan4	8004E0	8004E2	8004E4	
	Chan5	8004E8	8004EA	8004EC	
	Chan6	8004F0	8004F2	8004F4	
TSC4	Chan7	8004F8	8004FA	8004FC	800506
	Chan0	800500	800502	800504	
	Chan1	800508	80050A	80050C	
	Chan2	800510	800512	800514	
	Chan3	800518	80051A	80051C	
	Chan4	800520	800522	800524	
	Chan5	800528	80052A	80052C	
	Chan6	800530	800532	800534	
TSC5	Chan7	800538	80053A	80053C	800546
	Chan0	800540	800542	800544	
	Chan1	800548	80054A	80054C	
	Chan2	800550	800552	800554	
	Chan3	800558	80055A	80055C	
	Chan4	800560	800562	800564	
	Chan5	800568	80056A	80056C	
	Chan6	800570	800572	800574	
TSC6	Chan7	800578	80057A	80057C	800586
	Chan0	800580	800582	800584	
	Chan1	800588	80058A	80058C	
	Chan2	800590	800592	800594	
	Chan3	800598	80059A	80059C	
	Chan4	8005A0	8005A2	8005A4	
	Chan5	8005A8	8005AA	8005AC	
	Chan6	8005B0	8005B2	8005B4	
TSC7	Chan7	8005B8	8005BA	8005BC	8005C6
	Chan0	8005C0	8005C2	8005C4	
	Chan1	8005C8	8005CA	8005CC	

	Chan2	8005D0	8005D2	8005D4	
	Chan3	8005D8	8005DA	8005DC	
	Chan4	8005E0	8005E2	8005E4	
	Chan5	8005E8	8005EA	8005EC	
	Chan6	8005F0	8005F2	8005F4	
	Chan7	8005F8	8005FA	8005FC	
TSC8					800606
	Chan0	800600	800602	800604	
	Chan1	800608	80060A	80060C	
	Chan2	800610	800612	800614	
	Chan3	800618	80061A	80061C	
	Chan4	800620	800622	800624	
	Chan5	800628	80062A	80062C	
	Chan6	800630	800632	800634	
	Chan7	800638	80063A	80063C	
TSC9					800646
	Chan0	800640	800642	800644	
	Chan1	800648	80064A	80064C	
	Chan2	800650	800652	800654	
	Chan3	800658	80065A	80065C	
	Chan4	800660	800662	800664	
	Chan5	800668	80066A	80066C	
	Chan6	800670	800672	800674	
	Chan7	800678	80067A	80067C	
TSC10					800686
	Chan0	800680	800682	800684	
	Chan1	800688	80068A	80068C	
	Chan2	800690	800692	800694	
	Chan3	800698	80069A	80069C	
	Chan4	8006A0	8006A2	8006A4	
	Chan5	8006A8	8006AA	8006AC	
	Chan6	8006B0	8006B2	8006B4	
	Chan7	8006B8	8006BA	8006BC	
TSC11					8006C6
	Chan0	8006C0	8006C2	8006C4	
	Chan1	8006C8	8006CA	8006CC	
	Chan2	8006D0	8006D2	8006D4	
	Chan3	8006D8	8006DA	8006DC	
	Chan4	8006E0	8006E2	8006E4	
	Chan5	8006E8	8006EA	8006EC	
	Chan6	8006F0	8006F2	8006F4	
	Chan7	8006F8	8006FA	8006FC	
TSC12					800706
	Chan0	800700	800702	800704	
	Chan1	800708	80070A	80070C	
	Chan2	800710	800712	800714	
	Chan3	800718	80071A	80071C	
	Chan4	800720	800722	800724	
	Chan5	800728	80072A	80072C	
	Chan6	800730	800732	800734	
	Chan7	800738	80073A	80073C	
TSC13					800746
	Chan0	800740	800742	800744	
	Chan1	800748	80074A	80074C	
	Chan2	800750	800752	800754	
	Chan3	800758	80075A	80075C	
	Chan4	800760	800762	800764	
	Chan5	800768	80076A	80076C	

	Chan6	800770	800772	800774	
	Chan7	800778	80077A	80077C	
TSC14					800786
	Chan0	800780	800782	800784	
	Chan1	800788	80078A	80078C	
	Chan2	800790	800792	800794	
	Chan3	800798	80079A	80079C	
	Chan4	8007A0	8007A2	8007A4	
	Chan5	8007A8	8007AA	8007AC	
	Chan6	8007B0	8007B2	8007B4	
	Chan7	8007B8	8007BA	8007BC	
TSC15					8007C6
	Chan0	8007C0	8007C2	8007C4	
	Chan1	8007C8	8007CA	8007CC	
	Chan2	8007D0	8007D2	8007D4	
	Chan3	8007D8	8007DA	8007DC	
	Chan4	8007E0	8007E2	8007E4	
	Chan5	8007E8	8007EA	8007EC	
	Chan6	8007F0	8007F2	8007F4	
	Chan7	8007F8	8007FA	8007FC	

## **Change History:**

9/20/99 – NGW

Modified the Altera programming to allow the 7Mhz clock to the backplane to be enabled or disabled by writing a “1” at bit location [1] of address (base address +) 0x4 to enable the clock, 0x6 to disable the clock. The serial receiver and transmitter need the 7Mhz clock to be disabled for a time to allow them to “sync up”. Once they’re in sync (yellow LED on receiver card lit) the 7Mhz clock can be enabled. This works in day to day running mode, and in emulation mode.