REQUIREMENTS AND FUNCTIONAL SPECIFICATION

Station Board Timing FPGA

RFS Document: **A25052N0000**

Revision: 3.5

Heng Zhang and Dave Fort, 28 May 2012

National Research Council Canada
Herzberg Institute of Astrophysics
Dominion Radio Astrophysical Observatory

P.O. Box 248, 717 White Lake Rd
Penticton, B.C., Canada
V2A 6J9

Table of Contents

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations and Acronyms

CMIB	Correlator Module Interface Board
FPGA	Field Programmable Gate Array
JTAG	Joint Test Access Group
LVDS	Low Voltage Differential Signaling
MCB	Monitor/Control Bus
PAR	Place And Route
PCB	Printed Circuit Board
PCMC	PC/104 Monitor/Control
PECL	Positive Emitter-Coupled Logic
PLL	Phase-Locked Loop
PPS	Pulse Per Second

1 Revision History

Revision	Date	Changes/Notes	Author
Draft	03 Feb 2004	Initial Draft	H. Zhang
1.0	28 Jul 2005	Update registers, 2 > 1 DUMPTRIG	H. Zhang
2.0	25 Sep 2006	Updates due to Filter FPGA and TGB	D. Fort
2.1	25 Jan 2007	Additional Test Features, TC-FB	D. Fort
2.2	25 Apr 2007	Correct CRC in control output…	D. Fort
2.3	15 Jan 2008	Add self-incrementing mode to TCSTAMP. Add counters for synchronizing the Output FPGAs to the Timing FPGA. Add multiple DUMPTRIG generators.	D. Fort
2.4	15 Jan 2009	Changes for V1.1 of the Station Board. Added ability to use external SMA clock. Added O/P chopping for Baseline Board. Added PPS Time interval Counter. Range of interrupt delay to increased.	D. Fort
2.5	15 Feb 2009	Added hop count. Added command registers for Crossbar Board.	D. Fort
2.6	05 Apr 2009	Clarified what to do with hop count. Added automatic PPSCODE features.	D. Fort
2.7	02 Sep 2009	Change Test Outputs. Added individual PPSCODE registers. Added a PPSCODE time interval register for trouble shooting.	D. Fort
2.8	15 Sep 2009	Updated Status4 register. Changed PPSCODE clocking scheme (always in AUTO). Removed TIC introduced in 2.7.	D. Fort
2.9	02 Nov 2009	Changed description of PCSTATE register0x53.	D. Fort
3.0	02 Nov 2009	Changes for the no Raltron case. Modified TICs to detect overflow.	D. Fort

3.1	30 Jun 2010	Added status bits to PCSTATE register. Added control bits to PCCTLSTS register.	D. Fort
3.2	07 Dec 2010	Added bit to SYSDLY register.	D. Fort
3.3	01 Apr 2011	Pinouts, etc added.	D. Fort
3.4	25 Jan 2012	Double Speed write	D. Fort
3.5	28 May 2012	DUMPTRIG HES extensions for looping.	D. Fort

2 Introduction

This document describes detailed requirements and design concepts for the Station Board Timing FPGA. The Timing FPGA receives PPSCODE and CLOCK from a Crossbar Board which are used to provide the timing for the whole Station Board and, indirectly, to the Baselines Boards. The Timing FPGA generates signals to provide accumulation dump control, phase models, phase errors due to fractional sample delay errors and timing information to the Baseline Board.

Background information on the functionality of the Timing FPGA can be found in NRC-EVLA Memo 14. More detailed information on the interface of the Timing FPGA can be found in Filter FPGA RFS (A25044N0000). This document presents the concepts in more detail. Note that the TGB described in Memo 14 has been abandoned. Some of the TGB functions (copies of the External Time Code and the generation of a 128 MHz clock) now reside on a suitably populated Crossbar Board. The remaining functions now reside in the Timing FPGA.

This design will be developed and tested in an Altera FPGA (target device is EP1SGX25C but may fit in an EP1SGX10C). Functional/post-PAR simulation will be conducted during the FPGA development and final testing/verification will be done on the Station Board with other designs.

3 Overview

The Timing FPGA resides on the Station Board. It chooses one of two PPSCODEs and associated 128 MHz clocks it receives from a Crossbar Board to establish the Station Board time and clock rate. Both the PPSCODE and CLOCK pairs are sent out to the next Station Board in a chain of up to eight Station Boards. With the help of the CMIB, the Timing FPGA generates a more detailed TIMECODE to be used by the Baseline Boards. In addition, the Timing FPGA receives 18 sets of sub-band phase errors from each of two basebands and generates 18 PHASERR signals, one for each baseband pair. The Timing FPGA also generates DUMPTRIG and PHASEMOD based on the information sent by the CMIB. The DUMPTRIG, PHASEMOD, PHASERR and TIMECODE are combined and serialized onto a single differential pair which is sent to Baseline Boards (via Crossbar Boards). This control signal together with the two associated sub-band serialized data and 128 MHz clock (all generated by the Station Board Output FPGAs) go to the Crossbar Board via a high-speed cable. For detailed information on DUMPTRIG, PHASEMOD, PHASERR and TIMECODE signals, refer to HM Gbps Cable Signaling Specification (Document A25022N0041).

A simplified block diagram of the Station Board is shown in Figure 3-1.

A simplified input/output diagram of the Timing FPGA is shown in Figure 3-2. The JTAG interface and FPGA configuration interfaces are not shown in this diagram.

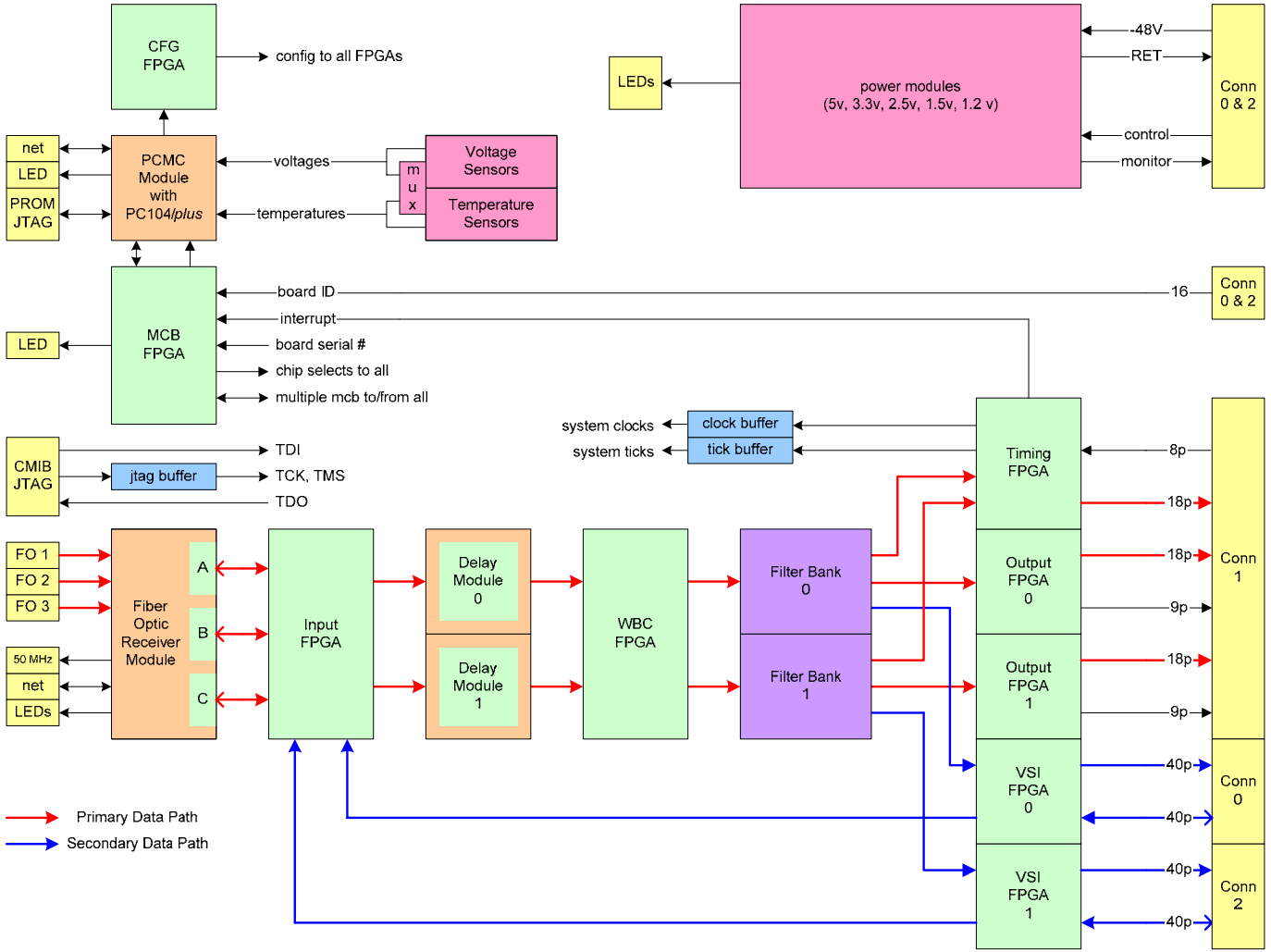


Figure 3-1 Block diagram of the Station Board.

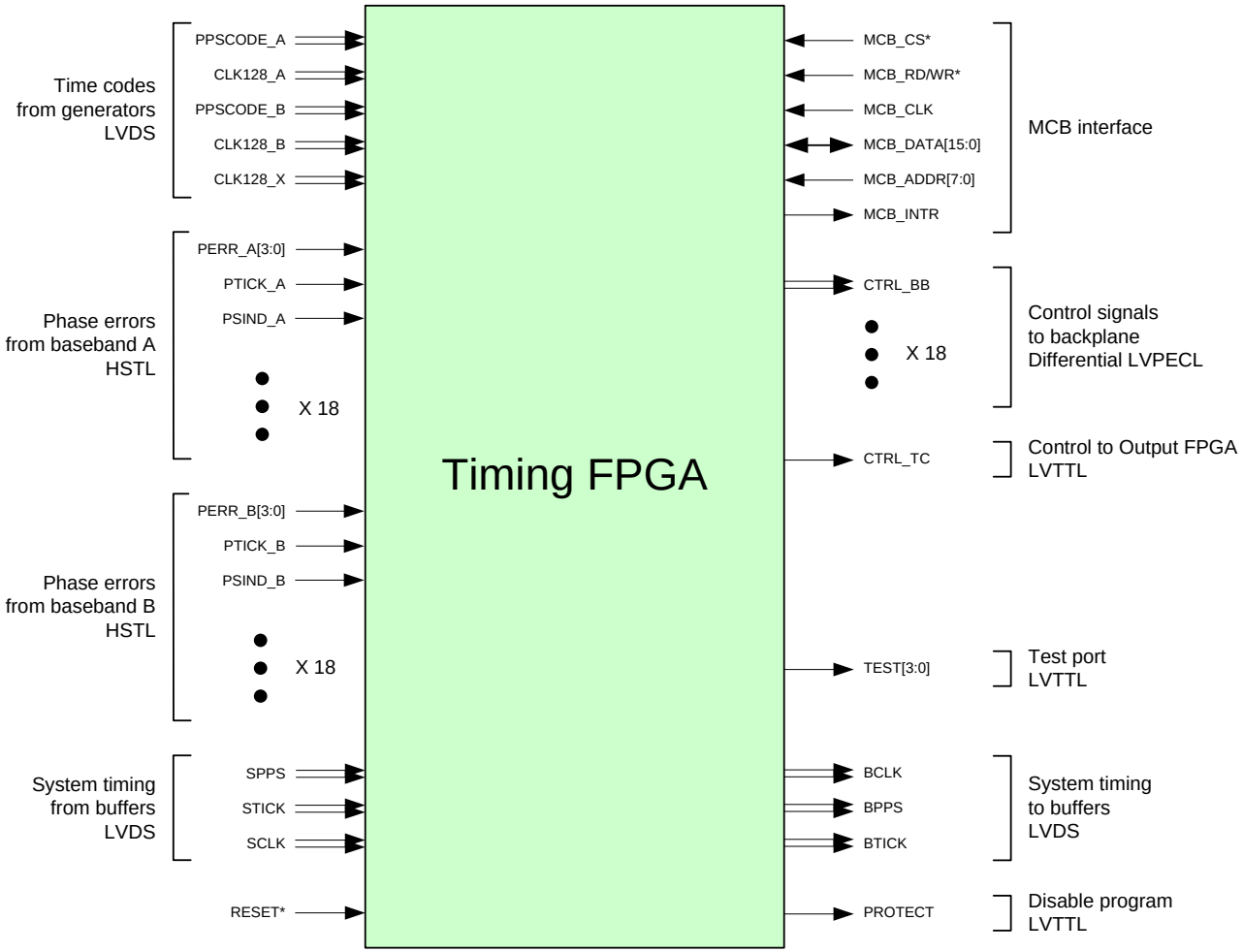


Figure 3-2 Input/Output diagram of the Timing FPGA.

The PPSCODE_A/B and CLK128_A/B inputs come from two independent sources. This scheme is used to mitigate the consequences of a failure in one of the sources. The Timing FPGA will default to A but choose B if A disappears. After selecting which PPSCODE and CLK128 to use, the Timing FPGA generates the timing signals BPPS (one pulse per second), BTICK (100 pulses per second) and BCLK (128 MHz). BPPS and BTICK are sent to buffers which distribute the system timing signals SPPS and STICK to all the FPGAs on the Station Board (including the Timing FPGA). BCLK is sent to a buffer and distributed to all the FPGAs on the Station Board. The External PPSCODE identifies the second and the time to one minute. The derived TIMECODE identifies the ten millisecond marker and the time to some specified past epoch. An additional input CLK128_X can be provided by a coaxial connector. This input can be used for testing but is not used by the operational EVLA.

The PERR input interface consists of phase error inputs from the Filter FPGAs, one set from each of the 18 Filter FPGAs in baseband A and the other set from each of the 18 Filter FPGAs in baseband B. Along with each phase error comes its associated ten millisecond tick and 128 MHz clock. The Timing FPGA will combine corresponding baseband pairs to generate 18 PHASERRs as part of the control signals that go through the backplane, on to the Crossbar Boards and then to the Baseline Boards (CTRL_BB).

The CTRL_BB output interface consists of 18 sub-band control signals. Each control signal contains DUMPTRIG, PHASEMOD, PHASERR and TIMECODE for two sub-bands, one from each baseband and is serialized to 1.024 Gb/s. A 128 MHz clock signal will be sent out by the Output FPGA (due to Timing FPGA pin limitations) and used to recover the control signal at the receiver end.

The system input interface consists of several system signals. The STICK, SPPS and SCLK signals are the system ten millisecond, one second and 128 MHz signals from the LVDS buffers on the Station Board.

The RESET input is the Timing FPGA external reset signal that brings the internal registers to a known state.

A brief description of each I/O signal is as follows. More detailed descriptions are contained in the following sections.

3.1 Master Reset

- RESET* is the input low-true master reset signal.
- RESET* is LVTTL.

3.2 FPGA Protect

- PROTECT, when high prevents the FPGA from being reprogrammed. It is set low by a master reset and can be set high by the CMIB. This signal prevents an

inadvertent de-programming of the FPGA. It can also be used to re-program individual FPGAs.

- PROTECT is LVTTL.

3.3 System Input Signals

- SCLK is the system 128 MHz clock input supplied as a LVDS pair from an LVDS buffer. The Timing FPGA chooses which of two time code 128 MHz clocks to send to the clock buffers. The 128 MHz output of the clock buffers is sent to every FPGA on the Station Board.

- SPPS is the system 1 Hz input supplied as an LVDS pair from a LVDS buffer. The Timing FPGA generates the system PPS from the chosen time code and sends it to the SPPS buffer. The output of the buffer is sent to some FPGAs on the Station Board.

- STICK is the system 100 Hz input supplied as an LVDS pair from an LVDS buffer. The Timing FPGA generates the system tick from the chosen time code and sends it to the STICK buffer. The output of the buffer is sent to every FPGA on the Station Board.

- RESET* is an active low signal to bring the Timing FPGA internal registers to a known state. It comes from the PC/104 Monitor/Control (PCMC) card.

3.4 Buffer Output Timing Signals

- BCLK is a clock (128 MHz) output chosen From the A, B or X input clocks in the Timing FPGA and goes to Station Board buffers to produce SCLK.

- BPPS is a PPS (1 Hz) output derived from the chosen time code in the Timing FPGA that goes to Station Board buffers to produce SPPS.

- BTICK is a tick (100 Hz) output derived from the chosen TIMECODE in the Timing FPGA that goes to Station Board buffers to produce STICK.

3.5 Time Code Input Signals

- TIMECODE_A/B are Time Code inputs supplied as two LVDS pairs to the Timing FPGA. They come from a Crossbar Board and contain a serial encoding of the master one second pulse and which second in the minute it is (see A25022N0090).

- CLK128_A/B are 128 MHz clock inputs supplied as two LVDS pairs to the Timing FPGA. They come from a Crossbar Board which ultimately produces them from an 128 MHz clock from a hydrogen maser frequency standard.

3.6 MCB Interface Signals

- MCB_ADDR[7:0] is the input 8-bit address bus for accessing internal Timing FPGA configuration, monitor and control registers.
- MCB_DATA[15:0] is the bi-directional 16-bit microprocessor data bus.
- MCB_CS* is the input low-true FPGA select that enables the MCB interface drivers.
- MCB_CLK is the input clock for the synchronous MCB interface. The phase and frequency of MCB_CLK is independent of system clock.
- MCB_RD/WR* is the input read/write enable. High enables read transactions and low enables write transactions when MCB_CS* is low.
- MCB_INTR is a pulse with a width of one MCB_CLK cycle, asserted every 10 milliseconds as an interrupt signal to the CMIB.

3.7 PERR Input Interface Signals

- PTICK_A/B[0:17] are 18 sub-bands 10 millisecond ticks from the baseband A/B filter FPGAs. This is a delayed version of STICK to match the data timing and any other pipeline delays in the Filter FPGA.
- PERR_A/B[0:17][3:0] are 18 sub-bands phase errors from the baseband A/B filter FPGAs, clocked in by a 2X version of SCLK. The 8-bit parallel phase error is clocked in as two nibbles at 256 MHz. The LS nibble of the phase error is clocked in first when the corresponding PTICK is high.
- PSIND_A/B[0:17] are 18 sub-bands sample indicators from the baseband A/B filter FPGAs. These signals are checked but not used in the timing FPGA.

3.8 Control Output Interface Signals

- The 18 sub-band control outputs are the serialized PECL output that contains DUMPTRIG, PHASEMOD, PHASERR and TIMECODE signals. A detailed description of these signals is given in A25022n0041.

3.9 Test Port Signals

These four outputs can be attached to a number of internal signals defined in Table 5-1 to provide a simple diagnostic capability.

3.10 JTAG Signals

Four dedicated pins on the FPGA are used to conduct JTAG test.

3.11 Configuration signals

8-bit configuration data bus and two control/monitor signals are used to program the Timing FPGA.

4 Requirements

The following is a list of the Timing FPGA requirements. For the most part, the Timing FPGA provides internal services to the Station Board; therefore, the requirements consist of a list of functions that must be performed with some additional description.

4.1 Functional Requirements

1. Receive two sets of PPSCODE bit streams with 128 MHz clocks from NRAO. Allow the CMIB to choose one of the two sets. Check the CRC of the time code, check that the 128 MHz clock is toggling and measure the PPSCODE period. Generate output PPSCODEs with incremented hop counts and output clocks.

2. Derive a PPS from the time code and provide the information contained in the time code to the CMIB.

3. Produce a delayed version of the PPS derived from the time code and generate PPS and TICK (100PPS) signals to send to external LVDS buffers. After buffering, these signals become SPPS and STICK for every FPGA on the Station Board.

4. Delay STICK by the requested amount and send it to the CMIB as an interrupt.

5. Generate 18 sub-band PHASERRs. Each PHASERR encodes the phase error from the two base-bands in a 20-bit frame synchronous with the T-bit (see A25022N0041). The phase error input cross-bar switch can select any 1 of 18 phase error inputs from each baseband to generate PHASERR. The settings of this switch should match those of the Output FPGA to ensure that the phase errors go with the correct data.

6. Generate DUMPTRIGs which will can be individually attached to each sub-band output. Each DUMPTRIG can contain different dumping instructions for any sub-band (see A25022N0041). Ideally, there would be one DUMPTRIG generator for each sub-band pair; however, there will not be enough RAM or logic in the Timing FPGA to implement 18 generators so the number has been limited to a maximum of 16. This limitation means that it is necessary to allow each output to choose which generators to use. Large RAM buffers (32k x 16-bits) may be necessary for pulsar observations while smaller ones (2k x 16-bits) could be used for normal observations; therefore, a mixture of RAM buffer sizes will be necessary to fit into the Timing FPGA.

7. Generate one PHASEMOD which will be the same for each sub-band but can contain different models for each sub-band (see A25022N0041).

8. Generate 18 distinct COMMAND outputs to control the Crossbar Board. There is no need to send commands on different wafers simultaneously (see A25022N0041).

9. Generate 18 sub-band control outputs and send them to the backplane. The generated DUMPTRIG, PHASEMOD, PHASERR, COMMAND and TIMECODE signals are combined into a 1.024Gbps output using Source-Synchronous Signaling provided by Altera Stratix GX device.

4.2 Performance Requirements

1. The Timing FPGA shall operate on the input sampled data and control signals with a clock rate of 128MHz. The 4-bit phase error nibbles will be clocked on both edges of the 128 Hz clock to form an 8-bit phase error samples at 128 Ms/s.

2. The synchronous MCB interface shall be capable of operating with a clock that is neither frequency nor phase synchronous with the 128MHz clock. The FPGA will support an MCB interface clock with a maximum rate of 33MHz.

3. The power dissipation of the FPGA, running at 128 MHz shall not exceed 3 Watts; however, the goal for the maximum power dissipation in 2 Watts.

4.3 Environmental Requirements

1. The Timing FPGA will be surface mounted on the Station Board PCB. Additional heat sinks may be attached to the FPGA to reduce temperature.

2. The board will use forced-air cooling with a normal operating ambient temperature of 20 °C and with a maximum ambient temperature of 40 °C.

4.4 Interface Requirements

As shown in Figure 3-2, the Timing FPGA has several interfaces. The following sections show the functional relative timing between the signals comprising each interface.

4.4.1 Time Code Interface Requirements

The time code interface provides the timing (time and clock) for the Station Board. Two sets of time codes are provided for redundancy. The CMIB will choose one of them to use. TIMECODE_A/B are running at 128Mb/s data rate. CLK128_A/B is 128 MHz and has a positive going edges aligned with the transitions in TIMECODE_A/B. The functional timing requirements of the TIMECODE interface are shown in Figure 4-1.

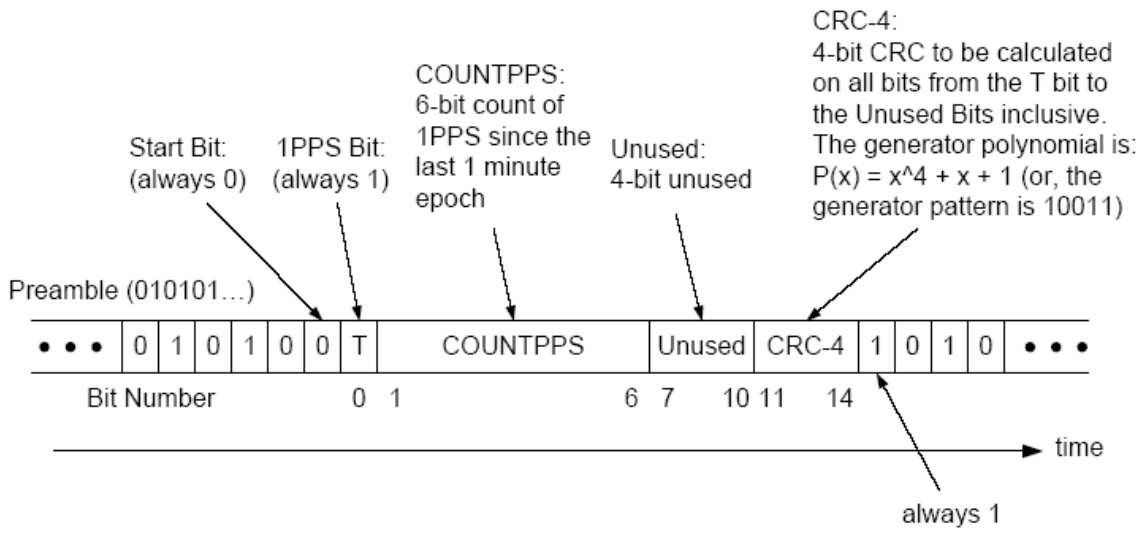


Figure 4-1 Time Code interface functional timing.

Each of the Time Code bits is described in detail as follows.

Bit[0] (Tick Bit) – 1 at 1PPS Time Tick. This is the fundamental epoch in the system.

Bit[1:6] (COUNTPPS Bits) – Counts of 1PPS time tick since last minute. Bit[1] is the LSB and Bit[6] is the MSB. This 6-bit counter increases by one every second and is zero on every minute.

Bit[7:10] (Unused Bits) – Bit 7 is 1, bit 8 is 0, bit 9 is 1 and bit 10 is 0.

Bit[11:14] (CRC-4 Bits): 4-bit CRC that is calculated on all bits from Tick Bit to the Major EPOCH Bits inclusive. The generator pattern is 10011.

4.4.2 Sub-band Phase Error Interfaces

Sub-band PERR interfaces provide the phase error due to the fractional bit delay error from the filter FPGA, one set of 18 from each baseband. PERR[3:0] is clocked in by a suitable phased version of SCLK on both rising and falling edges. PSIND is a sample indicator that accompanies the data. Figure 4-2 shows the phase error functional timing. For 128 MHz and 64 MHz bandwidth data, PSIND will always be high (upper half of figure) but for narrower bandwidths, PSIND will mark only one of the repeated phase errors (lower half of figure shows the 32 MHz bandwidth case).

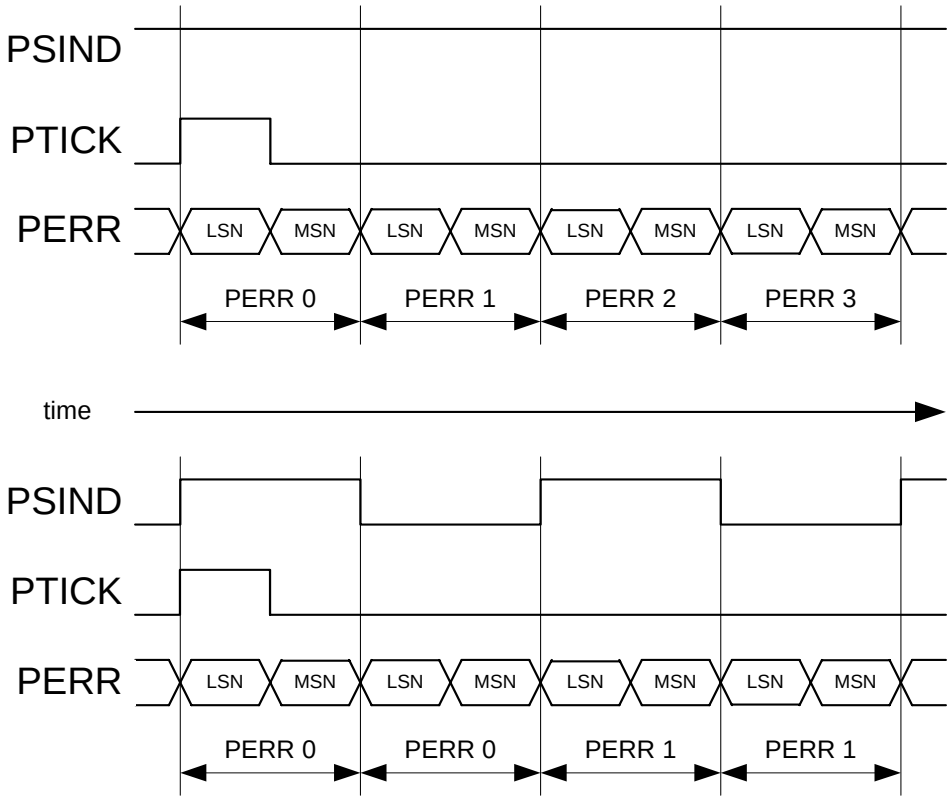


Figure 4-2 Sub-band Phase Error functional timing.

4.4.3 MCB Interface Requirements

The MCB (Monitor & Control Bus) interface allows the CMIB to write into the Timing FPGA to configure, monitor and control it.

When the microprocessor wants to write to the Timing FPGA, it puts the data on the MCB_DATA bus, the target register address on the MCB_ADDR bus and drives MCB_RD/WR* and MCB_CS* low some time before a rising edge of the MCB_CLK and keeps the signals stable until some time after the rising edge of the MCB_CLK. If MCB_CS* and MCB_RD/WR* are both low, the Timing FPGA then writes the data into the specified register on the rising edge of MCB_CLK. A write requires one clock cycle as shown below.

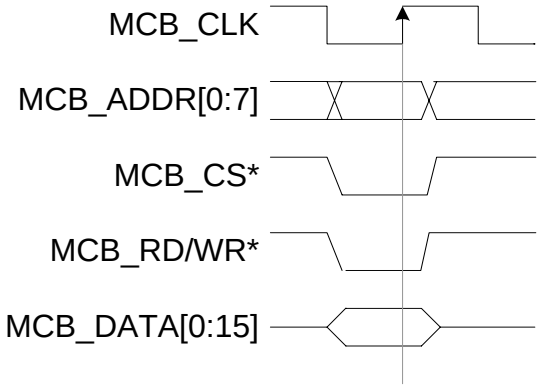


Figure 4-3 MCB Interface WRITE Functional Timing

When the microprocessor wants to read from the Timing FPGA, it drives MCB_RD/WR* high, puts the desired register address on the MCB_ADDR bus and drives the corresponding MCB_CS* low some time before a rising edge of MCB_CLK. If MCB_CS* is low and MCB_RD/WR* is high, the Timing FPGA then clocks the address into a register on the rising edge (A) of MCB_CLK and places the contents of the address contained in the register onto the MCB_DATA bus as long as MCB_CS* is low. The microprocessor then takes the data on the next rising edge (B) of MCB_CLK. The address says valid until the microprocessor takes the data. Read cycles require one clock cycle to setup and an additional clock cycle to read as shown below. Subsequent reads at the same address may take only one clock cycle depending on the microprocessor.

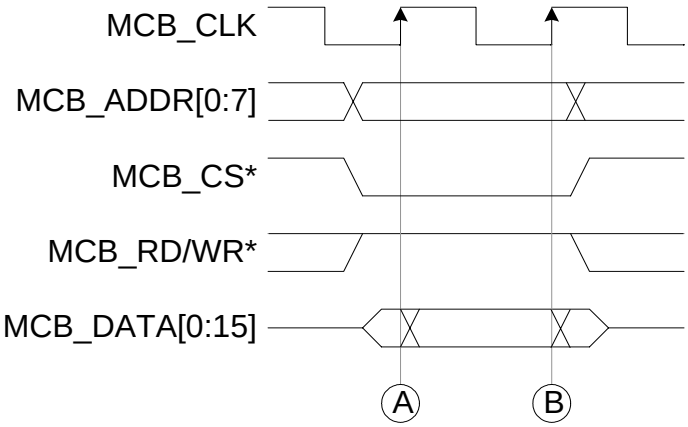


Figure 4-4 MCB Interface READ Functional Timing

5 Functional Specifications

A simplified block diagram of the Timing FPGA is shown below.

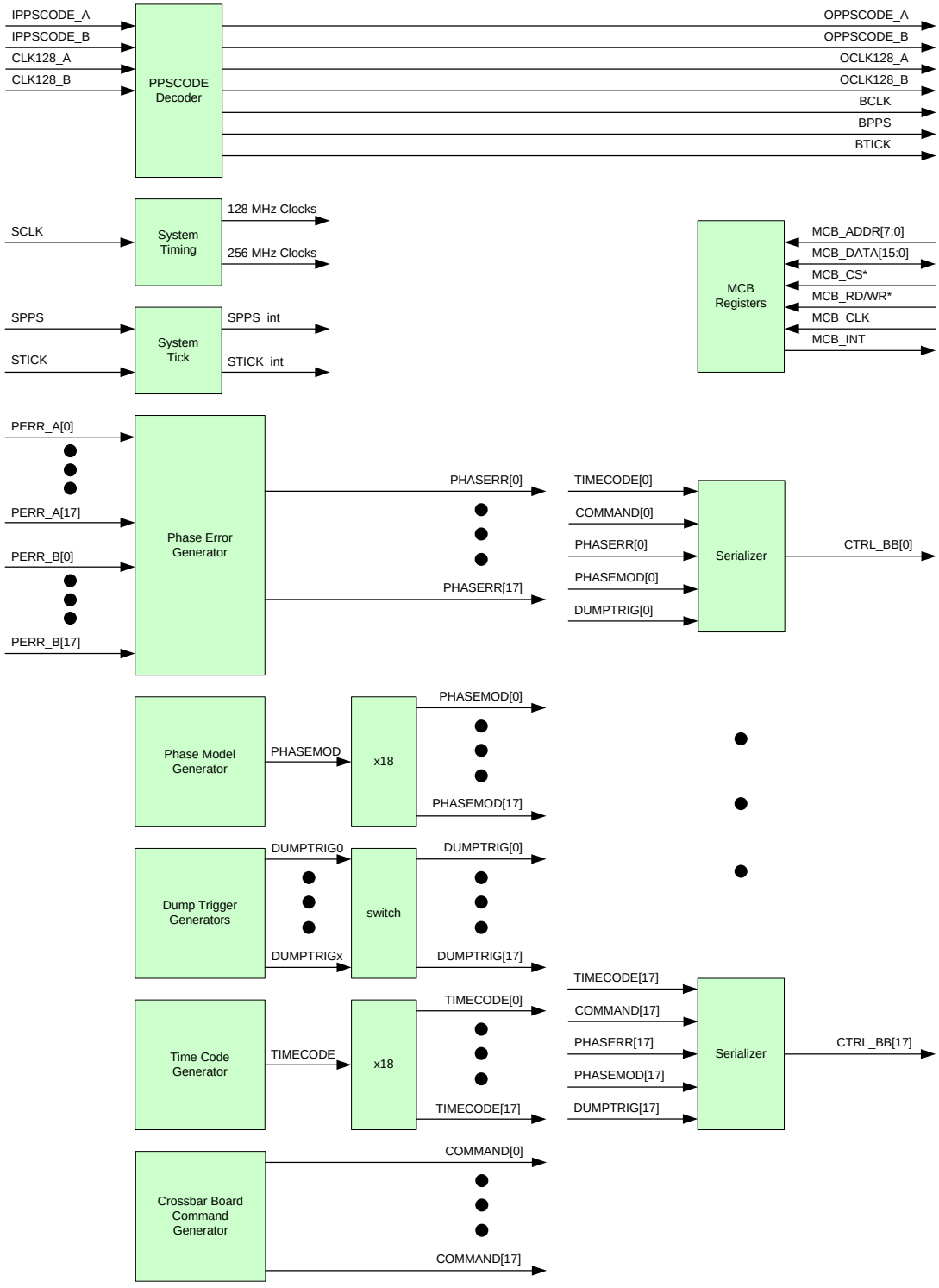


Figure 5-1 Block diagram of the Timing FPGA.

Each block in the Timing FPGA will now be described in more detail.

5.1 PPSCODE Decoder

Two sets of PPSCODE and CLK128 are received by the Timing FPGA. One set will be chosen automatically. A 128 MHz clock (BCLK) is produced for output to all the FPGAs on the board as SCLK as well as several status bits that can be read by the CMIB.

A clock signal input has been added to the Station Board in the form of an external 128 MHz signal. If the clock carried along with the PPSCODE contains too much jitter, then the external clock may be used.

The selected input PPSCODE (from the same set as CLK128) will be used to extract the second (0-59) and hop count (0-255) and generate output PPSCODEs with the hop count incremented for use by other Station Boards. In addition, delayed BPPS (1 Hz) and BTICK (100 Hz) timing signals are generated and sent to external buffers and returned to the Timing FPGA as SPPS and STICK respectively. The CRC of the incoming PPSCODE is checked. The hop count will be used by the software to set the delay of the whole board timing system so that different Station Boards can be aligned in time in spite of getting PPSCODES with different delays.

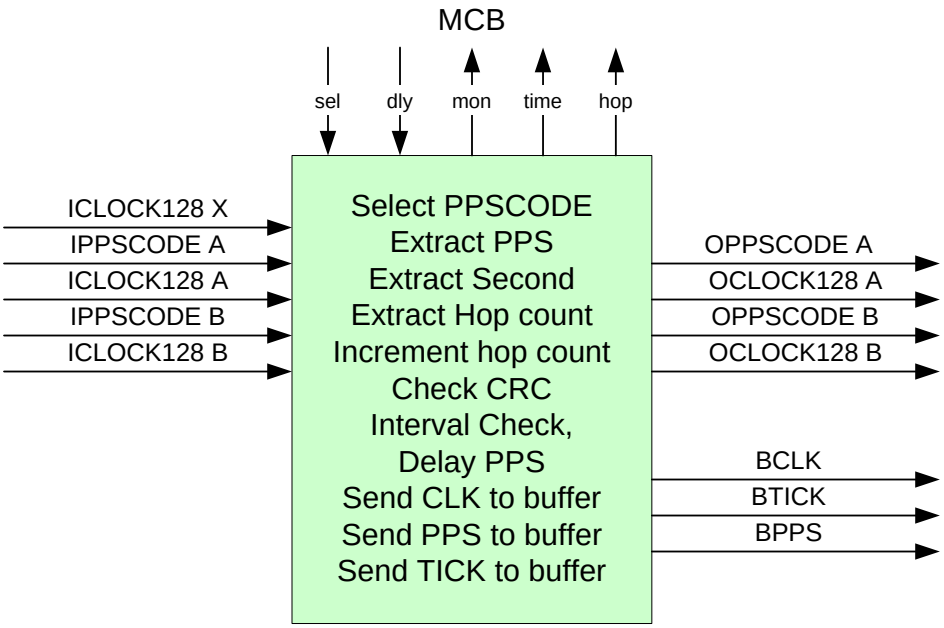


Figure 5-2 Block diagram of the PPSCODE Decoder.

5.2 System Timing

The System Timing block uses an Altera PLL to take the incoming SCLK and produces internal 128 MHz and 256 MHz clocks.

5.3 System Tick

In this block, the Timing FPGA will check the SPPS and STICK intervals. A 10 ms interrupt is also generated by delaying STICK by a programmable amount and sent to the PCMC. A functional block diagram of the System Tick block is shown below.

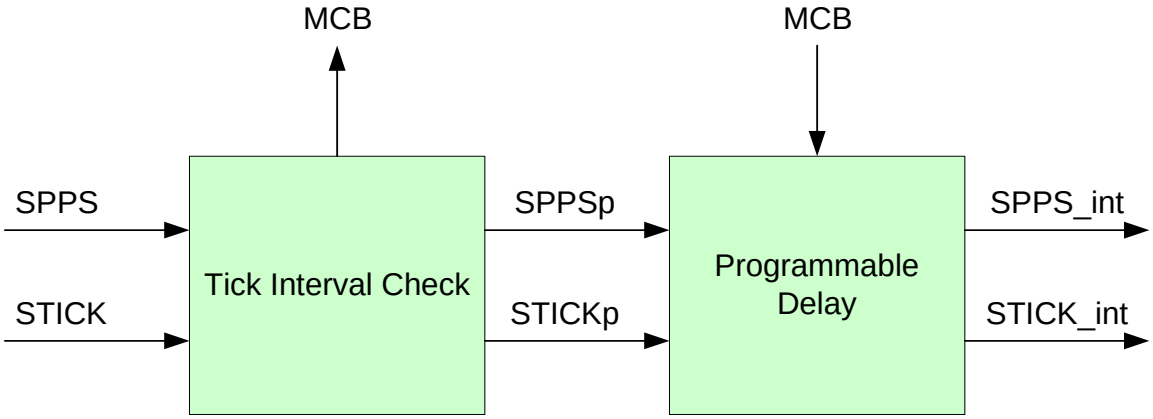


Figure 5-3 Block diagram of System Tick.

5.4 Time Code Generator

The Time Code Generator block receives the time (tick count, second count and epoch identifier) as well as a data header control bit from the CMIB and formats this information into TIMECODE as per A25022n0041 (“HM Gbps Cable Signal Specification”).

5.5 Phase Error Generator

The Timing FPGA generates a serial bit stream of phase errors used to correct fractional sample errors in delay (see A25022n0041). The correction is performed on the Baseline Board. The Timing FPGA receives PERR from the filter FPGAs and generates 18 sub-band PHASERR signals. A simplified block diagram of the PHASERR generator is shown below.

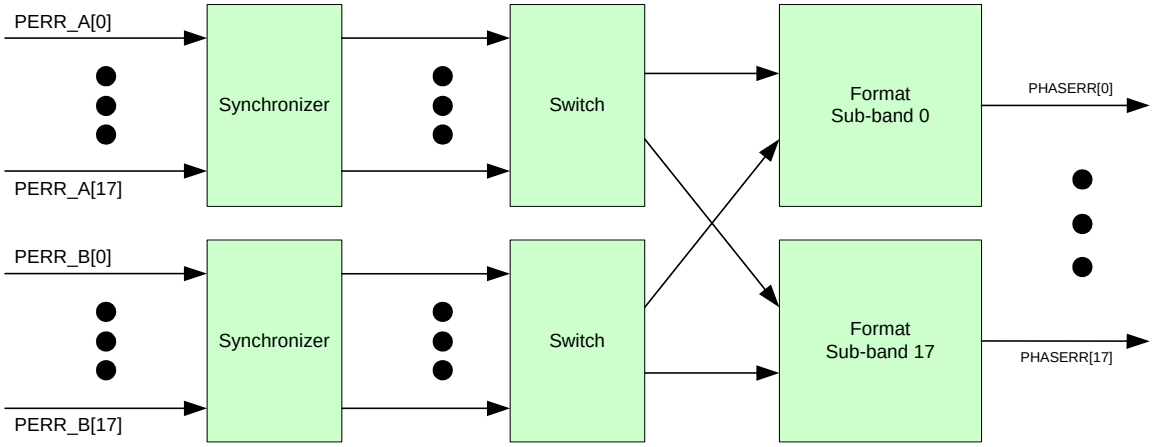


Figure 5-4 Block diagram of the Phase Error Generator.

The synchronizer block is used to align the incoming PERR signals (which may have differing delays from each Filter FPGA due to PCB trace lengths) to SCLK. In addition, two consecutive 4-bit nibbles are combined to produce an 8-bit phase error.

The switch block allows any baseband pair to be connected to any PHASERR output. The setting of the switch should match those in the Output FPGA.

The Format block produces PHASERR for each sub-band according to A25022N0041. The format combines the two sub-band phase error inputs with the “F bit” and a CRC.

5.6 Phase Model Generator

The Timing FPGA generates a serial bit stream of instructions for controlling the phase rotation on a Baseline Board (see A25022n0041). The phase model data is sent from the CMIB and stored in the PHASEMOD memory. The generator reads the phase model data from the PHASEMOD memory and builds the PHASEMOD frames. The PHASEMOD state machine starts reading the PHASEMOD memory on every internal STICK and stops reading when it receives an END command. It is only necessary that the memory consist of one dual port RAM, even in synchronous operation, because all the phase models can be transmitted in less than 20 µsecs whereas the interrupt will be delayed by considerably more than this for other reasons (Filter FPGA). This means that reading and writing to the memory never overlap and, more importantly, it means that the whole operation can be reset every interrupt. This is not the case for the DUMPTRIG generator (see next section). A functional diagram of the PHASEMOD generator block is shown in below.

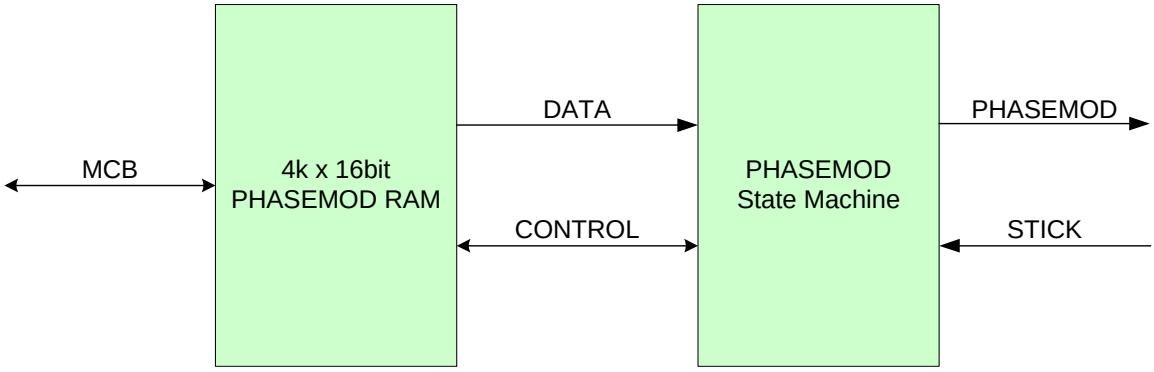


Figure 5-5 Block diagram of the Phase Model Generator.

5.7 Dump Trigger Generators

The Timing FPGA produces a serial bit stream of instructions for each sub-band pair used to control the dumping of the accumulators in the Correlator Chips on a Baseline Board (see A25022N0041). Ideally, there are 18 DUMPTRIG generators, one for each sub-band pair, but it may not be possible to accommodate this many in the Timing FPGA due to logic limitations. One DUMPTRIG generator which can include dump triggers for any or all sub-bands is described here. DUMPTRIG instructions are sent from the CMIB and stored in the DUMPTRIG memory. A state machine reads instructions from the DUMPTRIG memory and builds the DUMPTRIG frames for the Baseline Board. It starts reading the DUMPTRIG memory on an internal STICK when it was enabled. After that, it will continuously read the data to build the DUMPTRIG frames without explicitly synchronizing to the internal STICK. The state machine will stop building the DUMPTRIG frame when it receives an END command, usually at the end of an observation. A simplified block diagram of a DUMPTRIG generator is shown below.

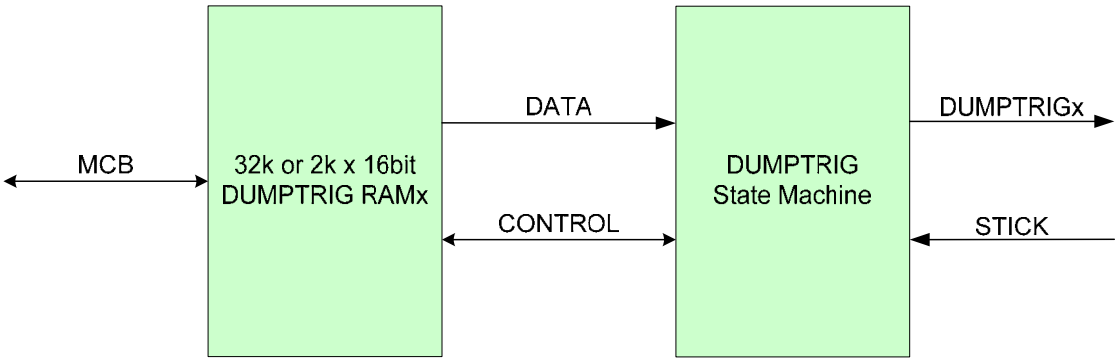


Figure 5-6 Block diagram of the Dump Trigger Generator.

5.8 Serializer

This block serializes DUMPTRIG, PHASEMOD, PHASERR, COMMAND and TIMECODE signals into a control output by using Altera Source-Synchronous Signaling in the Stratix GX device. Since we only use 4 of 8 parallel inputs in the serializer, one of the unused input can be defined as the synchronization pattern that will be used to align the de-serialized word boundary at the receiver. For detailed information of using Altera Source-Synchronous Signaling, refers to Altera application note AN236,

http://www.altera.com/literature/an/an236.pdf

and Stratix GX FPGA Data Sheet,

http://www.altera.com/literature/ds/ds_sgx.pdf.

6 Software Design Information

6.1 Register Map Summary

Register	Access	Address	Description
DESIGNID	R	00h	Design ID
TESTPIN0	R/W	01h	Test Pin Register 0 and 1
TESTPIN1	R/W	02h	Test Pin Register 2 and 3
PECRCSEL	R/W	03h	PERR Input CRC Select (A and B)
PECRCA0	R	04h	PERR A Inputs 0-3 CRCs
PECRCA1	R	05h	PERR A Inputs 4-7 CRCs
PECRCA2	R	06h	PERR A Inputs 8-11 CRCs
PECRCA3	R	07h	PERR A Inputs 12-15 CRCs
PECRCA4	R	08h	PERR A Inputs 16-17 CRCs
PECRCB0	R	09h	PERR B Inputs 0-3 CRCs
PECRCB1	R	0Ah	PERR B Inputs 4-7 CRCs
PECRCB2	R	0Bh	PERR B Inputs 8-11 CRCs
PECRCB3	R	0Ch	PERR B Inputs 12-15 CRCs
PECRCB4	R	0Dh	PERR B Inputs 16-17 CRCs
PTICKSEL	R/W	0Eh	PTICK Select (A and B)
PTICKCNTA0	R	0Fh	PTICK Count A [15:0]
PTICKCNTA1	R	10h	PTICK Count A [20:16]
PTICKCNTB0	R	11h	PTICK Count B [15:0]
PTICKCNTB1	R	12h	PTICK Count B [20:16]
PPSDLY0	R/W	13h	PPS Delay [15:0]
PPSDLY1	R/W	14h	PPS Delay [26:16]
SYSDLY	R/W	15h	System Delay [15:0]
INTDLY	R/W	16h	Interrupt Delay [15:0]
TCSTAMP0	R/W	17h	Seconds since EPOCH [15:0]
TCSTAMP1	R/W	18h	Seconds since EPOCH [31:16]
TCSTAMP2	R/W	19h	T, C, EPOCH, TCOUNT[9:0]
PPSCODE	R	1Ah	Received PPS and hop counts
STATUS0	R/W	1Bh	Status 0

Register	Access	Address	Description
STATUS1	R/W	1Ch	Status 1
STATUS2	R/W	1Dh	Status 2
STATUS3	R/W	1Eh	Status 3
STATUS4	R/W	1Fh	Status 4
CONTROL	R/W	20h	Control
DTWADDR	R	21h	DUMPTRIG Buffer Write Address
DTRADDR	R	22h	DUMPTRIG Buffer Read Address
DTTRIGCNT0	R	23h	DUMPTRIG TRIG Count [15:0]
DTTRIGCNT1	R	24h	DUMPTRIG TRIG Count [20:16]
PESWCFG0	R/W	25h	Phase Error Switch 0 Configuration
PESWCFG1	R/W	26h	Phase Error Switch 1 Configuration
PESWCFG2	R/W	27h	Phase Error Switch 2 Configuration
PESWCFG3	R/W	28h	Phase Error Switch 3 Configuration
PESWCFG4	R/W	29h	Phase Error Switch 4 Configuration
PESWCFG5	R/W	2Ah	Phase Error Switch 5 Configuration
PESWCFG6	R/W	2Bh	Phase Error Switch 6 Configuration
PESWCFG7	R/W	2Ch	Phase Error Switch 7 Configuration
PESWCFG8	R/W	2Dh	Phase Error Switch 8 Configuration
PESWCFG9	R/W	2Eh	Phase Error Switch 9 Configuration
PESWCFG10	R/W	2Fh	Phase Error Switch 10 Configuration
PESWCFG11	R/W	30h	Phase Error Switch 11 Configuration
PESWCFG12	R/W	31h	Phase Error Switch 12 Configuration
PESWCFG13	R/W	32h	Phase Error Switch 13 Configuration
PESWCFG14	R/W	33h	Phase Error Switch 14 Configuration
PESWCFG15	R/W	34h	Phase Error Switch 15 Configuration
PESWCFG16	R/W	35h	Phase Error Switch 16 Configuration
PESWCFG17	R/W	36h	Phase Error Switch 17 Configuration
PMPORT	R/W	37h	PHASEMOD Memory Port
DTPORT	R/W	38h	DUMPTRIG Memory Port
CLKSELA	R/W	39h	Select DDR clock for PERR A 0-15
CLKSELB	R/W	3Ah	Select DDR clock for PERR B 0-15

Register	Access	Address	Description
CLKSELABT	R/W	3Bh	Select DDR clock for TO, PERR 16-17
PLEN0	R/W	3Ch	PPS length in 128 MHz clocks [15:0]
PLEN1	R/W	3Dh	PPS length in 128 MHz clocks [26:16]
TLEN0	R/W	3Eh	TICK length in 128 MHz clocks [15:0]
TLEN1	R/W	3Fh	TICK length in 128 MHz clocks [20:16]
MCBTEST	R/W	40h	16 bit register for testing MCB interface.
INTRIND	R/W	41h	Poor man's interrupt indicator.
TIMER0	R/W	42h	TIMER count in microsecond [15:0]
TIMER1	R/W	43h	TIMER count in microsecond [31:16]
PMWADDR	R	44h	PHASEMOD Buffer Write Address
PMRADDR	R	45h	PHASEMOD Buffer Read Address
TCOUNT	R	46h	Tick Counter
TPESEL	R/W	47h	Tick Phase Error Select
TPEOUT	R	48h	Tick Phase Error Output
TIMOUTA0	R	49h	OUTA/TIM timing difference [15:0]
TIMOUTA1	R	4Ah	OUTA/TIM timing difference [21:16]
TIMOUTB0	R	4Bh	OUTB/TIM timing difference [15:0]
TIMOUTB1	R	4Ch	OUTB/TIM timing difference [21:16]
DTSELECT	R/W	4Dh	DUMPTRIG Generator select.
DTSWITCH0	R/W	4Eh	DUMPTRIG Generator switch 0 to 3.
DTSWITCH1	R/W	4Fh	DUMPTRIG Generator switch 4 to 7.
DTSWITCH2	R/W	50h	DUMPTRIG Generator switch 8 to 11.
DTSWITCH3	R/W	51h	DUMPTRIG Generator switch 12 to 15.
DTSWITCH4	R/W	52h	DUMPTRIG Generator switch 16 to 17.
PCSTATE	R/W	53h	PPSCODE status bits and selections.
CSRR	R/W	54h	Chopper Seed Root Register.
PPSCNT0	R/W	55h	PPS Time Interval[15:0].
PPSCNT1	R/W	56h	PPS Time Interval[26:16].
XBADDR	R/W	57h	Address for Crossbar Board command.
XBDATA	R/W	58h	Data for Crossbar Board command.

Register	Access	Address	Description
PPSCODE_A	R	59h	Received second and hop count from A
PPSCODE_B	R	5Ah	Received second and hop count from B
TOGCOUNT_A	R	5Bh	A clock toggle count (test)
TOGCOUNT_B	R	5Ch	B clock toggle count (test)
TOGCOUNT_S	R	5Dh	S clock toggle count (test)
TOGCOUNT_X	R	5Eh	X clock toggle count (test)
PCCTLSTS	R	5Fh	PPSCODE control/status (test)
SYSDLY_MS	R	60h	System Delay [16]
DTPBIN	R	61h	Phase Bin and Bank for HES extensions
DTPORTP	R/W	FFh	Double wide DTPORT

Table 5-1 Register map of the MCB Interface.

6.2 Detailed Register Description

Each register in the map will be described in the following pages.

6.2.1 Design ID Register (DESIGNID)

This register specifies the Timing FPGA Design/Version/Revision.

DESIGNID (read only) Addr = 00h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Design								Version				Revision			
Day (hex)								Month(hex)				Year(hex)			

Revision[3:0] – The revision number is to be incremented whenever the new FPGA bit file incorporates new functionality but does not require software upgrades.

Version[7:4] – The version number is to be incremented whenever the new FPGA bit file is incompatible with the previous version from the software standpoint and would require change to the driver.

Design[15:8] – The design number is not likely to used for the Timing FPGA. If more than one design is used simultaneously then this number should be different to allow the software to behave differently if desired.

6.2.2 Test Pin Registers (TESTPIN)

There are four 6-bit addresses to select an internal signal to be routed to Test Pins 0 to 3. The mapping of these address values to internal signals is given in Table 5-2. The four selected signals are routed to pads unregistered. This means that the relative timing of the four test pins is only accurate to a few nanoseconds.

TESTPIN0 (read/write) Addr = 01h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		Address 1						Reserved		Address 0					

Address 0[5:0] – 6-bit address that maps the FPGA signals to the Test Pin 0.

Address 1[5:0] – 6-bit address that maps the FPGA signals to the Test Pin 1.

TESTPIN1 (read/write) Addr = 02h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		Address 3						Reserved		Address 2					

Address 2[5:0] – 6-bit address that maps the FPGA signals to the Test Pin 2.

Address 3[5:0] – 6-bit address that maps the FPGA signals to the Test Pin 3.

The address mapping of the internal FPGA signals is shown as follows:

Address	Internal Signal Description	Address	Internal Signal Description
00h	SYS_RST	10h	ZERO
01h	SCLK128	11h	ZERO
02h	SCLK256	12h	ZERO
03h	SYS_TICK_INT	13h	ZERO
04h	SYS_PPS_INT	14h	ZERO
05h	PREAMBLE_INT	15h	ZERO
06h	PC_CLK_A_I	16h	ZERO
07h	PC_CLK_B_I	17h	ZERO
08h	PC_CLK_X_I	18h	ZERO
09h	PC_DAT_A	19h	ZERO
0Ah	PC_DAT_B	1Ah	ZERO
0Bh	PC_PPS_O	1Bh	COMMAND_O[0]
0Ch	TX_SYNCBIT	1Ch	PHASERR_O[0]
0Dh	ZERO	1Dh	TIMECODE_O
0Eh	ZERO	1Eh	PHASEMOD_O
0Fh	ZERO	1Fh	DUMPTRIG_O

Address	Internal Signal Description	Address	Internal Signal Description
20h	MCB_CLK	30h	ZERO
21h	MCB_CS*	31h	ZERO
22h	MCB_RD/WR*	32h	ZERO
23h	PHASEMOD_EN	33h	ZERO
24h	PHASEMOD_WR_EN	34h	ZERO
25h	PHASEMOD_CLR	35h	ZERO
26h	DUMPTRIG_ARM	36h	ZERO
27h	DUMPTRIG_WR_EN	37h	ZERO
28h	DUMPTRIG_CLR	38h	ZERO
29h	DUMPTRIG_T	39h	ZERO
2Ah	PTICK_INT_A	3Ah	ZERO
2Bh	PTICK_INT_B	3Bh	COMMAND_F
2Ch	MCB_IND	3Ch	PHASERR_F
2Dh	ZERO	3Dh	TIMECODE_F
2Eh	ZERO	3Eh	PHASEMOD_F
2Fh	ZERO	3Fh	DUMPTRIG_F

Table 5-2 Address mapping for the Test Pin Registers.

6.2.3 PERR CRC Select Register (PECRCSEL)

This register selects the phase error bit stream of each baseband on which to perform CRC calculations. The CRC is calculated between two consecutive system ticks on the selected bit streams and is stored in the PCRC register. The CMIB then checks that the CRCs on both transmitter side and receiver side match to verify that the connection is OK and, therefore, there are no signal integrity problems.

PECRCSEL (read/write) Addr = 03h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					SEL B								SEL A		

B_SEL A[1:0] – 2-bit selects 1 of 4 bit streams from a phase error input that comes from the baseband A filter FPGAs. Value of 0-3 refers to as bit stream #0-3. If B_SEL_A[2]=1, then SIND_A is substituted for stream 0.

B_SEL B[1:0] – 2-bit selects 1 of 4 bit streams from a phase error input that comes from the baseband B filter FPGAs. Value of 0-3 refers to as bit stream #0-3. If B_SEL_B[2]=1, then SIND_B is substituted for stream 0.

6.2.4 PERR CRC Registers (PECRCnA/B n = 0, 1, 2, 3, 4)

These registers contain the results of the CRC calculation on the selected bit for the 18 sub-bands of each of the two basebands. The PECRCnA group contains the CRCs for baseband A and the PECRCnB group contains the CRCs for baseband B.

PECRCnA/B (read/write) Addr = 04h to 0Dh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRC[4*n+3]				CRC[4*n+2]				CRC[4*n+1]				CRC[4*n]			

CRC[x][3:0] – the 4-bit CRC calculated between two consecutive system ticks for four sub-bands where n = 0, 1, 2, 3, 4. PECRC4A/B contain only two CRCs.

6.2.5 PTICK Select Register (PTICKSEL)

This register contains two parts. PTICK_SEL A selects 1 of 18 PTICK inputs from the baseband A filter FPGAs and PTICK_SEL B selects 1 of 18 PTICK inputs from the baseband B filter FPGAs. The Timing FPGA counts the distance between the selected PTICK and STICK. The counter value for the selected PTICK from baseband A and system tick is stored in the PTICK Count A register. The counter value for the selected PTICK from baseband B and system tick is stored in the PTICK Count B register.

PTICKSEL (read/write) Addr = 0Eh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PTICK_SEL B					Reserved			PTICK_SEL A				

PTICK_SEL A/B[4:0] – 5-bit select will choose 1 of 18 PTICK inputs from the baseband A/B filter FPGAs. Value of 0-17 refers to as the phase error input 0-17.

6.2.6 PTICK Count A/B Registers (PTICKCNTA/B)

These registers contain the interval from the selected PTICK (from the baseband A/B filter FPGA) to the STICK in 256 MHz clocks. This interval can be read by the CMIB so that the system can set the proper output delay value in the filter FPGAs. After changing PTICKSEL in the ISR, the intervals read during the next ISR may not be correct.

PTICKCNTA0 (read only) Addr = 0Fh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTICKCNT A[15:0]															

PTICKCNTA1 (read only) Addr = 10h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL_A		Reserved								PTICKCNT A[21:16]					

PTICKCNTB0 (read only) Addr = 11h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTICKCNT B[15:0]															

PTICKCNTB1 (read only) Addr = 12h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL_B		Reserved								PTICKCNT B[21:16]					

SEL_A[1:0] and SEL_B[1:0] determine which time intervals are measured:
00=>dtick to stick, 01=>dtick to dtick, 10=>stick to stick and 11=>stick to dtick.

6.2.7 PPS Delay Register (PPSDLY)

This register is used to set the PPS delay in clock cycles at 128 MHz. The PPS derived from the time code is delayed to produce the system PPS and TICK. The delay is needed

to move the time reference to near the middle of the Delay Module delay range so that the Delay Module can implement both positive and negative delays. The following two addresses set a 27-bit delay count value (>1 second); however, the delay will probably be about 130 milliseconds.

PPSDLY0 (read/write) Addr = 13h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSDLY[15:0]															

PPSDLY1 (read/write) Addr = 14h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					PPSDLY[26:16]										

6.2.8 System Delay Register (SYSDLY)

This register is used to set the system delay in clock cycles at 128 MHz. STICK is delayed to match the pipeline delay of previous FPGAs (especially the Delay Module and the Filter FPGAs). This ensures that the TIMECODE (and header in the Output FPGA) occurs at the correct time.

SYSDLY (read/write) Addr = 15h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SYSDLY															

6.2.9 Interrupt Delay Register (INTDLY)

This register is used to set the interrupt delay in clock cycles at 64 MHz after the delayed system tick. This delay makes sure that all the models written to the Delay Module and (especially) to the Filter FPGAs have been used before new ones are written. In addition, this delay makes sure that all measurements are complete before the corresponding registers are read. The largest value of the delay is forced by the narrowest bandwidth for which the delay should be 0xC000 (768 microseconds).

INTDLY (read/write) Addr = 16h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INTDLY															

6.2.10 Time Stamp Registers (TCSTAMP)

These three registers are used to allow the CMIB to set the time stamp that will appear in the output of the Baseline Board (every interrupt). The time stamp values set will be sent on the next 10-millisecond tick in the TIMECODE going to the Baseline Board. For the content of TIMECODE information, refer to A25022N0041.

TCSTAMP0 (read/write) Addr = 17h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCOUNT[15:0]															

SCOUNT[15:0] – PPS Count Bit[15:0]. Counts of 1PPS time tick since last EPOCH. Bit[0] is the LSB and Bit[31] is the MSB.

TCSTAMP1 (read/write) Addr = 18h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCOUNT[31:16]															

SCOUNT[31:16] – PPS Count Bit[31:16]. Counts of 1PPS time tick since last EPOCH.

TCSTAMP2 (read/write) Addr = 19h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	T	C	EPOCH			TCOUNT[9:0]									

TCOUNT[9:0] – 100PPS Count Bit[9:0]. Counts of 100PPS time ticks since last 1PPS. Bit[0] is the LSB and Bit[9] is the MSB.

EPOCH[2:0] – Major EPOCH Bit[2:0]. (3'b000) equals 00:00:00 Jan 1/1970.

C – Control Bit for headers 1 =>headers are inserted.

T – Tick Bit. 1 at 1PPS Time Tick, 0 at 100PPS Time Tick. This is the fundamental epoch in the system.

A – Auto bit. Normally this bit is set to 0 and the CMIB sets the values of all the TCSTAMP bits every interrupt. Alternatively, the TCSTAMP bits can be set once with the auto bit set to 1. In this case, the values of T, TCOUNT and SCOUNT are set automatically using the internal TICK and PPS. The given values of T and TCOUNT are ignored. The given value of SCOUNT is set and then incremented automatically using the internal PPS. New values of SCOUNT are set each time TCSTAMP2 is written (at the next PPS). In addition, the values read back from the TCSTAMP registers are the updated values (as opposed to the sent values).

6.2.11 Time Code Register (PPSCODE)

This register contains the one second time code supplied to the Station Board and the number of hops that the time code has been through on its way. It should be read by the ISR. See A25022n0041 for a detailed description.

PPSCODE (read only) Addr = 1Ah

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTHOP								**COUNTPPS**						0	0

COUNTPPS [5:0] – Counts of seconds since last minute. Bit[0] is the LSB and Bit[5] is the MSB. COUNTPPS should be in the range 0 to 59. COUNTPPS should be ignored if the PCCRC-Err bit is high in the STATUS0 register.
COUNTHOP [7:0] – Number of hops that the selected External Station Timecode (PPSCODE) has been through to get to this board. Bit[0] is the LSB and Bit[7] is the MSB. COUNTHOP should be in the range 0 to 255 (with PCOVF-Err bit = 0 in STATUS0) and should be a constant unless the source of the PPSCODE is changed (PC-Sel bit in the CONTROL register). COUNTHOP should be ignored if the PCCRC-Err bit is high in the STATUS0 register. COUNTHOP is used by the CMIB to determine this part of PPSDLY so that all Station Boards will have nearly the same timing (within range of the Baseline Board FIFOs). Each Station Board delays the PPSCODE by 18 128 MHz clocks; however, the edge used in the DDR input can alter this by 0.5 clocks (affected by the external PPSCODE cable lengths). The board that suffers the least number of hops will need the biggest delay in PPSDLY to align the on-board timing and , therefore, the timing of the signals going to the Baseline Board. PPSDLY may compensate for other delays (PPSDLY0) but the part due to the hop count is given by:

PPSDLY = PPSDLY0 + (MAX HOP COUNT POSSIBLE - hop_count) * 18

For the EVLA MAX HOP COUNT POSSIBLE can be set to 16.

6.2.12 Status Registers

These registers contain monitor bits that reflect the status of the Timing FPGA. Normally, you would not expect to write to status registers. The bitwise *XOR* of the value written and the actual value of the status bits is returned on read so that errors can be created for test purposes.

STATUS0 (read/write) Addr = 1Bh

7	6	5	4	3	2	1	0
STICK-Mis	SPPS-Intv	STICK-Intv	SCLK-Bad	XCLK-Bad	CLKTX-Bad	R	R

15	14	13	12	11	10	9	8
R	PCOVF-Err	SPPS-Bad	STICK-Bad	PCPPS-Intv	PCCRC-Err	CLKB-Bad	CLKA-Bad

CLKTX-Bad – A value of 1 indicates that the Gigabit transmitter PLL is not locked.

XCLK-Bad – A value of 1 indicates that the external clock is not toggling.

SCLK-Bad – A value of 1 indicates that the Altera SCLK PLL is not locked.

STICK-Intv – STICK interval error. A counter in the Timing FPGA counts the clock cycles between two consecutive STICK inputs to check the STICK interval. If the counts between two consecutive STICK inputs are not a 10-millisecond interval, the Timing FPGA will set this bit to (1). Set at tick so ISR reads.

SPPS-Intv – SPPS interval error. A counter in the Timing FPGA counts the clock cycles between two consecutive SPPS inputs to check the SPPS interval. If the counts between two consecutive SPPS inputs are not a 1 second interval, the Timing FPGA will set this bit to (1). Set at tick so ISR reads.

STICK-Mis – STICK and SPPS misalign error. If SPPS is not coincident with an STICK, then this bit will be set to (1). Set at tick so ISR reads.

CLKA-Bad – A value of 1 indicates that the PPSCODE_A clock is not toggling.

CLKB-Bad – A value of 1 indicates that the PPSCODE_B clock is not toggling.

PCCRC-Err - If the Timing FPGA finds a CRC error on the chosen PPSCODE input, it will set this bit to (1). Set at PPS so ISR reads.

PCPPS-Intv – A counter in the Timing FPGA counts the clock cycles between two consecutive PPS inputs decoded from the chosen PPSCODE. If the count is not correct, this bit will be set to (1). Set at PPS so ISR reads.

STICK-Bad – A 1 indicates that the width of STICK is incorrect.

SPPS-Bad – A 1 indicates that the width of SPPS is incorrect.

PCOVF-Err - If the Timing FPGA finds that the hop count overflowed (0xFF => 0x00) on the chosen PPSCODE input, it will set this bit to (1). Set at PPS so ISR reads.

STATUS1 (read/write) Addr = 1Ch

7	6	5	4	3	2	1	0
SIND - ErrA7	SIND - ErrA6	SIND - ErrA5	SIND - ErrA4	SIND - ErrA3	SIND - ErrA2	SIND - ErrA1	SIND - ErrA0

15	14	13	12	11	10	9	8
SIND - ErrA15	SIND - ErrA14	SIND - ErrA13	SIND - ErrA12	SIND - ErrA11	SIND - ErrA10	SIND - ErrA9	SIND - ErrA8

STATUS2 (read/write) Addr = 1Dh

7	6	5	4	3	2	1	0
SIND- ErrB7	SIND - ErrB6	SIND - ErrB5	SIND - ErrB4	SIND - ErrB3	SIND - ErrB2	SIND - ErrB1	SIND - ErrB0

15	14	13	12	11	10	9	8
SIND - ErrB15	SIND - ErrB14	SIND - ErrB13	SIND - ErrB12	SIND - ErrB11	SIND - ErrB10	SIND - ErrB9	SIND - ErrB8

STATUS3 (read/write) Addr = 1Eh

7	6	5	4	3	2	1	0
Reserved						SIND - ErrA17	SIND - ErrA16

15	14	13	12	11	10	9	8
Reserved						SIND - ErrB17	SIND - ErrB16

SIND-Err(A/B)(n) Sample Indicator error on baseband A/B sub-band n.

STATUS4 (read/write) Addr = 1Fh

7	6	5	4	3	2	1	0
R	R	R	PMW-Err	PMR-Err	PMS-Err	PMF-Err	PMC-Err

15	14	13	12	11	10	9	8
R	R	R	R	DTR-Err	DTS-Err	DTT-Err	DTC-Err

PMC-Err – Phase Mod Command error. When the PHASEMOD generator is building a PHASEMOD frame, if it finds an illegal command in the phase model data, it will set this bit to (1). Set at tick so ISR reads.

PMF-Err – Phase Mod Frame error. When the PHASEMOD generator is building a PHASEMOD frame, if it finds the frame will cross into the next 10 millisecond interval, it will set this bit to (1). Set at tick so ISR reads.

PMS-Err – Phase Mod Start bit error. If a start bit is set where the preamble would not have been a one, this bit is set to (1). Set at tick so ISR reads.

PMW-Err – Phase Mod write overflow error. If the RAM write address ever reaches the end of the buffer (overflow), this bit is set to (1). Set at tick so ISR reads.

PMR-Err – Phase Mod read overflow error. If the RAM read address ever reaches the end of the buffer (overflow), this bit is set to (1). Set at tick so ISR reads.

DTC-Err – Dump Trig Command error. When the DUMPTRIG generator is building a DUMPTRIG frame, if it finds an illegal command in the dump trigger model data, it will set this bit to (1) and enter the IDLE state. Set at tick so ISR reads.

DTT-Err – Dump Trig Trigger error. If a trigger bit is set where the preamble would not have been a zero, this bit is set to (1). Set at tick so ISR reads.

DTS-Err – Dump Trig Start bit error. If a start bit is set where the preamble would not have been a one, this bit is set to (1). Set at tick so ISR reads.

DTR-Err – Dump Trig RAM error. If the RAM read and write addresses are ever identical (overflow or underflow), this bit is set to (1). Set at tick so ISR reads.

6.2.13 Control Register (CONTROL)

This register contains the individual control bits to set the Timing FPGA operation. Note that these bits remain in force until changed by the CMIB. Care must be taken to avoid inadvertently changing some bits while modifying others. This register is not changed by a hardware reset in order that a software reset can be held for use as a standby mode.

CONTROL (read/write) Addr = 20h

7	6	5	4	3	2	1	0
SYS-Sel	DT-Arm	DT-Clr	PM-En	PM-Clr	XB-Err	R	R

15	14	13	12	11	10	9	8
PLL-Rset	PE-Err	TC-Err	PM-Err	DT-Err	RST-SW	INTR-En	TX-Bit

XB-Err[2] – If this bit is set to (1), the COMMAND CRC will be wrong.

PM-Clr[3] – PHASEMOD generator clear bit. (1) clears the PHASEMOD generator.

PM-En[4] – PHASEMOD generator enable bit. (1) enables the PHASEMOD generator.

DT-Clr[5] – DUMPTRIG generator clear bit. (1) clears DUMPTRIG generator.

DT-Arm[6] – DUMPTRIG generator arm bit. (1) arms DUMPTRIG generator.

SYS-Sel[7] – Selects which DDR output to use for SPPS and STICK. The value will be determined during testing.

TX-Bit[8] – This bit should be set to a (1). It allows the BSLB to byte align.

INTR-En[9] – If this bit is set to (1), the CMIB interrupt will be enabled.

RST-SW[10] – If this bit is set to (1), the FPGA will be held in reset state.

DT-Err[11] – If this bit is set to (1), the DUMPTRIG CRC will be wrong.

PM-Err[12] – If this bit is set to (1), the PHASEMOD CRC will be wrong.

TC-Err[13] – If this bit is set to (1), the TIMECODE CRC will be wrong.

PE-Err[14] – If this bit is set to (1), the PHASERR CRC will be wrong.

PLL-Rset[15] – If this bit is set to (1), The SCLK PLL will be reset.

6.2.14 DUMPTRIG Address Registers (DTWADDR & DTRADDR)

These registers hold the values of the RAM buffer write and read addresses as they were at the last STICK. These values could be used for checking correct operation.

DTWADDR (read only) Addr = 21h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DTWADDR [14:0]														

DTRADDR (read only) Addr = 22h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DTRADDR [14:0]														

R – Reserved bit.

6.2.15 DUMPTRIG TRIG Count Register (DTTRIGCNT)

In the DUMPTRIG generator, there is a counter to count the distance between the generation of the trigger bit and the STICK. This can be used to check that DUMPTRIG is keeping the correct timing.

DTTRIGCNT0 (read only) Addr = 23h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTTRIGCNT[15:0]															

DTTRIGCNT1 (read only) Addr = 24h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											DTTRIGCNT[20:16]				

6.2.16 Phase Error Switch Configuration Registers (PESWCFG0-17)

The sub-band control output contains DUMPTRIG, PHASEMOD, PHASERR and TIMECODE signals. PHASEMOD and TIMECODE are identical for all sub-bands but DUMPTRIG and PHASERR can be different for each sub-band. PHASERR is composed of 1 of 18 phase error inputs from the baseband A Filter FPGAs and 1 of 18 phase error inputs from the baseband B filter FPGAs. The sub-band configuration registers control the selection of the PHASERR signals and the selection should match the selection of the data in the Output FPGA. 18 sub-band configuration registers control each sub-band output configuration separately.

PESWCFG (read/write) Addr = 25-36h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved			PECFG-B[4:0]					Reserved			PECFG-A[4:0]				

PECFG-A[4:0] – 5-bit control to select 1 of 18 phase error inputs from the baseband A filter FPGAs. Value of 0-17 refers to as the phase error input 0-17. Others are reserved.

PECFG-B[4:0] – 5-bit control to select 1 of 18 phase error inputs from the baseband B filter FPGAs. Value of 0-17 refers to as the phase error input 0-17. Others are reserved.

6.2.17 PHASEMOD Memory Port Register (PMPORT)

This is the memory port address that allows the microprocessor to write and read phase model data to the PHASEMOD memory. Read on this register will return the last written data.

PMPORT (read/write) Addr = 37h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHASEMOD[15:0]															

PHASEMOD[15:0] – phase model data to be written to the PHASEMOD memory.

6.2.18 DUMPTRIG Memory Port Register (DTPORT)

This is the memory port address that allows the microprocessor to write and read dump trigger model data to the DUMPTRIG memory. Read on this register will return the last written data.

DTPORT (read/write) Addr = 38h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DUMPTRIG[15:0]															

DUMPTRIG[15:0] –dump trigger data to be written to the DUMPTRIG memory.

6.2.19 Clock Select Registers

These registers allow the CMIB to select one of two clock phases for the input of the phase error data from the Filter FPGAs, the synchronizing ticks from the Output FPGAs and the Time Code edge connector. This choice is needed because the Timing FPGA is run from SCLK while the input signals have an arbitrary phase relative to SCLK due to differing PCB trace lengths. The PCB traces (6) from a given Filter FPGA will be nearly the same length by design and all six will use the same clock phase. The actual values will be determined empirically during testing and should be the same for all Station Boards.

CLKSELA (read/write) Addr = 39h

7	6	5	4	3	2	1	0
PERR-ClkA7	PERR-ClkA6	PERR-ClkA5	PERR-ClkA4	PERR-ClkA3	PERR-ClkA2	PERR-ClkA1	PERR-ClkA0

15	14	13	12	11	10	9	8
PERR-ClkA15	PERR-ClkA14	PERR-ClkA13	PERR-ClkA12	PERR-ClkA11	PERR-ClkA10	PERR-ClkA9	PERR-ClkA8

CLKSELB (read/write) Addr = 3Ah

7	6	5	4	3	2	1	0
PERR-ClkB7	PERR-ClkB6	PERR-ClkB5	PERR-ClkB4	PERR-ClkB3	PERR-ClkB2	PERR-ClkB1	PERR-ClkB0

15	14	13	12	11	10	9	8
PERR-ClkB15	PERR-ClkB14	PERR-ClkB13	PERR-ClkB12	PERR-ClkB11	PERR-ClkB10	PERR-ClkB9	PERR-ClkB8

CLKSELABT (read/write) Addr = 3Bh

7	6	5	4	3	2	1	0
R	TO-ClkA	R	R	R	R	PERR-ClkA17	PERR-ClkA16

15	14	13	12	11	10	9	8
R	TO-ClkB	R	R	R	R	PERR-ClkB17	PERR-ClkB16

TO-ClkA(B) selects the edge for the tick from the OUTPUT FPGA A(B) input.

6.2.20 PPS and TICK Length Registers

These registers are used to shorten simulation times and are set to their normal values on reset. Normal values are PPSLEN = 128000000-1 = 127999999 = 0x07a11fff = 27'h7a11fff and TICKLEN = 1280000-1 = 1279999= 0x001387ff = 21'h1387ff.

PPSLEN (read/write) Addr = 3Ch

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSLEN[15:0]															

PPSLEN (read/write) Addr = 3Dh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved					PPSLEN [26:16]										

TICKLEN (read/write) Addr = 3Eh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TICKLEN[15:0]															

TICKLEN (read/write) Addr = 3Fh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved											TICKLEN [20:16]				

6.2.21 MCB Test Register

This 16-bit register can be used to check that the FPGA is reading and writing correctly.

MCBTEST (read/write) Addr = 40h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Whatever your little heart desires															

6.2.22 Interrupt Indicator Register (INTRIND)

This 16-bit register can be used as a poor man's (polling) interrupt.

INTRIND (read only) Addr = 41h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															IND

IND – A 1 in this bit indicates that an interrupt has occurred. Any write to this register clears the indicator bit. The interrupts do not have to be enabled. This can be used for a poor man's polled interrupt.

6.2.23 TIMER Registers (TIMER0/TIMER1)

This 32-bit register can be used by the CMIB to time intervals in microseconds.

6.2.24 PHASEMOD Address Registers (PMWADDR & PMRADDR)

These registers hold the values of the RAM buffer write and read addresses as they were at the last END command. These values could be used for checking correct operation.

PMWADDR (read only) Addr = 44h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				PMWADDR [11:0]											

PMRADDR (read only) Addr = 45h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				PMRADDR [11:0]											

R – Reserved bit.

6.2.25 TCOUNT register

This register contains the system tick count from 0 to 99 and can be used by the CMIB to identify the interrupt number and check correct operation.

TCOUNT (read only) Addr = 46h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved									TCOUNT[6:0]						

6.2.26 TPESEL register

This register selects which of the 18 phase errors to store in TPEOUT at the tick. It can be used for testing phase error transfer from the Filter FPGA .

TPESEL (read/write) Addr = 47h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved			TPESEL_B					reserved			TPESEL_A				

6.2.27 TPEOUT register

This register contains the value of the selected phase error at the tick. It can be used for testing phase error transfer from the Filter FPGA .

TPEOUT (read only) Addr = 48h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPEOUT_B								TPEOUT_A							

6.2.28 TIMOUT Count A/B Registers (TIMOUTA/B)

These registers contain the interval from the tick at the serial output time from each of the A/B Output FPGAs to the equivalent tick in the Timing FPGA in 256 MHz clocks. This interval can be read by the CMIB so that the system can set the proper output delay value in the filter FPGAs that keep the serial outputs from the Timing FPGA and the Output FPGAs in sync.

TIMOUTA0 (read only) Addr = 49h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT A[15:0]															

TIMOUTA1 (read only) Addr = 4Ah

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL_A		Reserved								COUNT_A[21:16]					

TIMOUTB0 (read only) Addr = 4Bh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT B[15:0]															

TIMOUTB1 (read only) Addr = 4Ch

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL_B		Reserved								COUNT_B[21:16]					

SEL_A[1:0] and SEL_B[1:0] determine which time intervals are measured: 00=>dtick to stick, 01=>dtick to dtick, 10=>stick to stick and 11=>stick to dtick. If COUNT_A|B = 0x3FFFFF, then an overflow has occurred which means that one of the ticks was missing. Setting SEL_A|B to 01 or 10 can help decide which tick is missing.

6.2.29 DUMPTRIG Select Register (DTSELECT)

This register determines with which of the DUMPTRIG generators the CMIB is communicating. It affects the following registers: CONTROL, DTWADDR, DTRADDR, DTPORT, DTTRIGCNT0 and DTTRIGCNT1.

DTSELECT (read/write) Addr = 4Dh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved												DTGEN			

6.2.30 DUMPTRIG Switch Register (DTSWITCH)

This register determines which of the DUMPTRIG generators feeds which output.

DTSWITCH0 (read/write) Addr = 4Eh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTGEN to output 3				DTGEN to output 2				DTGEN to output 1				DTGEN to output 0			

DTSWITCH1 (read/write) Addr = 4Fh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTGEN to output 7				DTGEN to output 6				DTGEN to output 5				DTGEN to output 4			

DTSWITCH2 (read/write) Addr = 50h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTGEN to output 11				DTGEN to output 10				DTGEN to output 9				DTGEN to output 8			

DTSWITCH3 (read/write) Addr = 51h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DTGEN to output 15				DTGEN to output 14				DTGEN to output 13				DTGEN to output 12			

DTSWITCH4 (read/write) Addr = 52h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved								DTGEN to output 17				DTGEN to output 16			

6.2.31 PPSCODE STATE Register (PCSTATE)

This register contains the result of the automatic PPSCODE clock and edge selects.

PCSTATE (read/write) Addr = 53h

7	6	5	4	3	2	1	0
PPSb-ici	PPSa-ici	PPSb-sci	PPSa-sci	DAT-Sel[1:0]		CLK-Sel[1:0]	

15	14	13	12	11	10	9	8
BADb-int	BADa-int	SEL-rst	MAN-sel	OVFb-err	OVFa-err	CRCb-err	CRCa-err

CLK-Sel[1:0] – Shows which clock is being used as the board system clock. CLK-Sel = 00, then the A clock is being used. If CLK-Sel = 01, then the B clock is being used. If CLK-Sel = 10, then the X (external) clock is being used. If CLK-Sel = 11, then no clock is being used (board clock off). If an external clock (X clock) has always been present, it will be always used as the board system clock. If the X clock appears while in an existing OK state, then the existing state will remain until this register is written to. Otherwise, one of the A or B clocks will be sent depending on which one is being used for extracting the PPSCODE. If none of A, B or X clocks is present, then the board system clock will disappear.

DAT-Sel[1:0] – Shows which PPSCODE (A or B or neither) clock and data is being used to get the PPS, the second count and the hop count. If DAT-Sel = 00, then the A clock and data is being used. If DAT-Sel = 01, then the B clock and data is being used. If DAT-Sel = 1x, then PPSCODE (second count and hop count) will be meaningless (off).

PPSa/b-sci – PPS system clock interval error for PPSCODE A/B.

PPSa/b-ici – PPS input clock interval error for PPSCODE A/B.

CRCa/b-err – CRC error for PPSCODE A/B.

OVFa/b-err – hop count overflow error for PPSCODE A/B.

MAN-sel – if this bit is 1 then the board clock and PPSCODE selection will be what is written to the **CLK-Sel** and **DAT-Sel** bits.

SEL-rst – allow normal selection for board clock and for PPSCODE. If, while using the A clock and data, a Station Board in the middle of the PPSCODE chain is powered down, the B chain will be used by the boards that can no longer use the (broken) A chain. If the A chain is made OK again, the boards using the B chain will stay using the B chain (to prevent further interruptions). Setting this bit to 1 and then back to 0 will cause the Timing FPGA to restart the selection process.

BADa/b-int – Integrated errors for PPSCODE A/B. CRC and interval errors are integrated over 8 seconds and declared bad if all 8 are bad.

6.2.32 Chopper Seed Root register – CSRR

This register sets the chopper seed root for each of the two filter banks.

CSRR (read/write) Addr = 54h

7	6:4	3:1	0
CID0	SID0	R	FID0

15	14:12	11:9	8
CID1	SID1	R	FID1

The bit assignments here are suggested. Any 16-bit value may be sent. The Timing FPGA will add the wafer number (0 to 17) to whatever is sent and will protect against the either seed being 0xFF. The aim here is to provide as many different chopping waveforms to the RXP FPGAs on the baseline board. The same values for the two seed roots must be set in the Output FPGAs for the scheme to work.

CID0/1[7] Crate ID (0 or 1) for bank 0/1

SID0/1[6:4] Slot ID (0 to 7)) for bank 0/1

FID0/1[0] FPGA ID (0 or 1) for bank 0/1 (FID0 should=0 and FID1 should = 1)

6.2.33 PPS Count Registers (PPSCNT0/1)

These two registers contain the interval from the PPS received from the External Timecode (PPSCODE) input to the Station Board PPS received by the Timing FPGA in 256 MHz clocks. This interval should remain constant but will be altered if PPSDLY is changed.

PPSCNT0 (read only) Addr = 55h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT A[15:0]															

PPSCNT1 (read only) Addr = 56h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				COUNT A[27:16]											

6.2.34 XBAR Address Register (XBADDR)

This register determines to which sub-band pair (0-17) the Crossbar Board command data is sent. On read, bit-15 = 1 indicates that the command is still being sent (busy) and bit-14 = 1 indicates that the address is out of range.

XBADDR (read/write) Addr = 57h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
busy	error	reserved									address				

6.2.35 XBAR data Register (XBDATA)

This register contains the Crossbar Board command data to be sent. The address must be sent first and then the data.

XBDATA (read/write) Addr = 58h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
command[15:0]															

6.2.36 Individual PPS Code Registers (PPSCODE_A/B)

These registers contain the one second time code supplied to the Station Board and the number of Station Boards the time code has been through on its way as individual quantities, not just the working values.

PPSCODE_A/B (read only) Addr = 59h/5Ah

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTHOP								**COUNTPPS**						**OVF**	**CRC**

COUNTPPS [5:0] – Counts of seconds since last minute. Bit[0] is the LSB and Bit[5] is the MSB. COUNTPPS should be in the range 0 to 59. COUNTPPS should be ignored if the CRC bit is high.
COUNTHOP [7:0] – Number of hops that the selected External Station Timecode (PPSCODE) has been through to get to this board. Bit[0] is the LSB and Bit[7] is the MSB. COUNTHOP should be in the range 0 to 255 (with OVF bit = 0) and should be a constant unless the source of the PPSCODE is changed.
OVF – a 1 in this bit indicates that a hop count overflow has occurred (255 to 0).
CRC – a 1 in this bit indicates that a time code CRC error has occurred.

6.2.37 Toggle Count Registers (TCOUNT_A/B/S/X)

These registers contain the 'toggle' count for each of the external clocks coming in – A(PPSCODE A), B(PPSCODE B), S(board clock) and X(external coax).

TOGCOUNT_A/B/S/X (read only) Addr = 5Bh/5Ch/5Dh/5Eh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOGCOUNT															

TOGCOUNT [15:0] – Counts of 128 MHz in 4091 33 MHz clock cycles. This count should be ~15709 (0x3D5D). The FPGA considers the clock to be bad if the count < 0x3D50 or it the count > 0x3D70. This register is meant to be used for testing only.

6.2.38 PPSCODE Control/ Status (PCCTLSTS)

These registers contain status and control bits for the PPSCODE clocks and data.

PCCTLSTS (read/write) Addr = 5Fh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved										datb	data	**X**	**S**	**B**	**A**

A/B/S/X – Shows the results of comparing TOGCOUNT with its limits. 1 => bad.

data/b – disable code if 1.

6.2.39 MS bit of SYSDLY register (SYSDLY_MS)

SYSDLY_MS **(read/write) Addr = 60h**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															MS

6.2.40 DUMPTRIG Phase Bin register (DTPBIN)

DTPBIN **(read only) Addr = 61h**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
B	**Reserved**				**Phase Bin**										

The HES extensions for DUMPTRIG are meant for looping and hence require that the phase bin and bank be updated in the FPGA. This register allows the software to see what the value of phase bin and bank were at the last 10 millisecond tick and hence should be read and checked in the ISR.

B - bank.

6.2.41 Double wide DTPORT register (DTPORTP)

Addr = FFh

The MCB is 16 bits wide but the PCI bus is 32 bits wide. A 32-bit word sent by the software to a register must first pass through the PCMC which would normally only send the lower 16 bits out on the MCB. For this particular address in the Timing FPGA address space, however, the PCMC will divide the 32-bit word into two 16-bit words and send them to register 0x38 (DTPORT) sequentially, LS part first. This operation effectively speeds up sending HES by a factor of two which is highly desirable because a lot of HES must be sent in a short time.

On read, DTPORT contains the last (MS) 16-bit word and DTPORTP contains the first (LS) 16-bit word of the last double wide write. Checking these registers allows the user to check the correct operation off the double wide write feature.

6.3 PHASEMOD Generator

This section will explain how the PHASEMOD generator works in more detail. The concept of building PHASEMOD frame is to include a command with the phase model data and the embedded command tells the state machine how to put the phase model data together to form a PHASEMOD frame. The width bits are introduced to cope with the case when not all 8 data bits are used. In order to understand the whole process, first we need to define the phase model data formats.

Phase model data format definition:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit#
command					**width**			**data**								Content
command					**length**											Content

- **Data**: The 8-bit *data* field contains the real phase model data to build the PHASEMOD frame. Data[0] will be put in the PHASEMOD frame first.

- **Width**: The 3-bit *width* field specifies how many data bits are actually used to build the frame. It is defined in the following table.

Width	Data bits actually used
000	data [0:0]
001	data [1:0]
010	data [2:0]
011	data [3:0]
100	data [4:0]
101	data [5:0]
110	data [5:0]
111	data [7:0]

- **Length:** specifies the number of preamble bits for NOP (see below).

- **Command**: The 5-bit *command* field tells the state machine how to put bits into the PHASEMOD frame. Each command is defined as the following table.

Command	Command description
00000	Reserved.
00001	**SBIT** - insert 1 start bit.
00010	**DATA** - insert *width* data bits.
00011	**CRC** - insert 4 zero bits as a placeholder for the CRC.
00100	**END** - insert continuous preamble bits.
00101	**NOP** - insert *length* preamble bits.
00110	Reserved.
All others	All others are reserved for possible future use.

Use of each command will be explained in detail as follow.

SBIT: This command is used to tell the state machine to start a frame by inserting a zero. Other bits are ignored.

DATA: This command is used to tell the state machine to insert data bits in the frame. Width specifies how many data bits are inserted.

CRC: This command is used to tell the state machine to insert the 4 CRC bits in the frame. The CRC is generated in the Timing FPGA. It also indicates the end of a frame. Other bits are ignored in this word.

END: This command is used to tell the state machine to stop reading instructions and go to the IDLE state which continues to insert preamble until the next system tick. Other bits are ignored in this word.

NOP: This command is used to tell the state machine to insert preamble. Bits [10:0] Length specifies the number of preamble bits to be inserted.

The following state diagram illustrates the PHASEMOD state machine's operation.

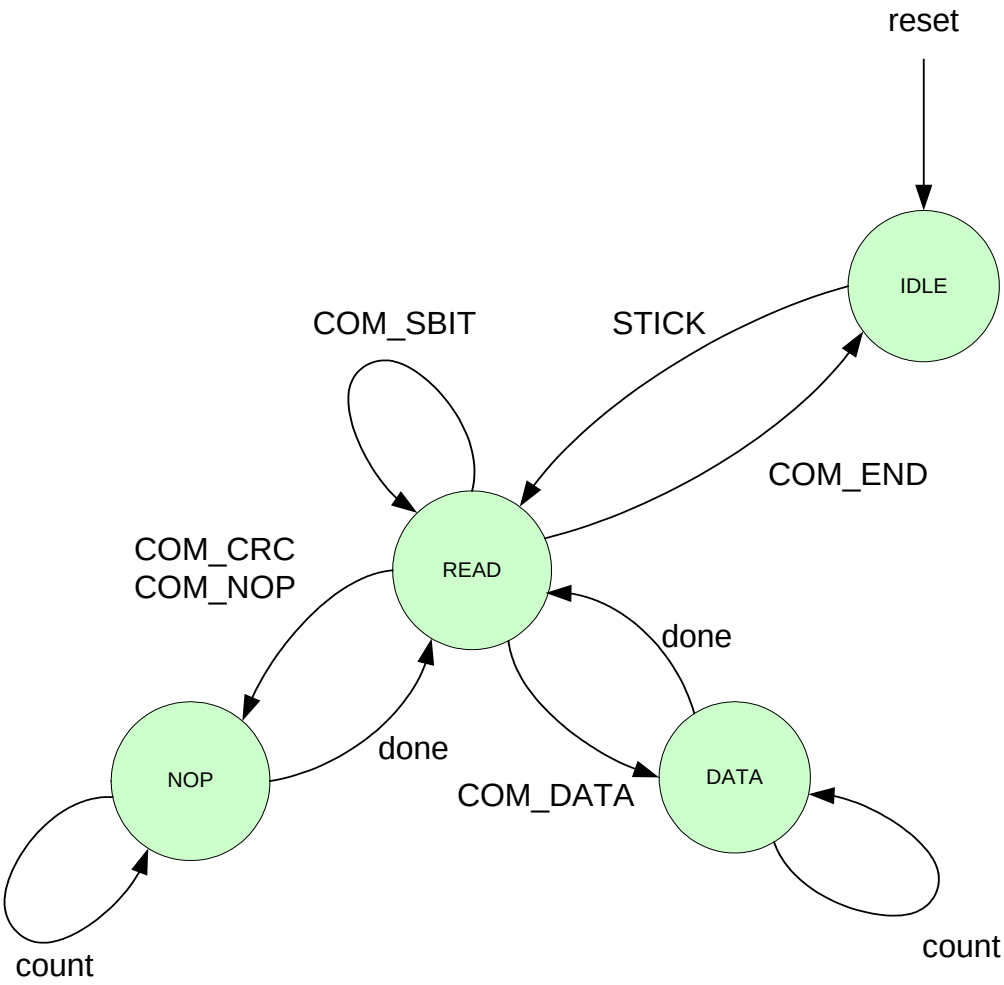
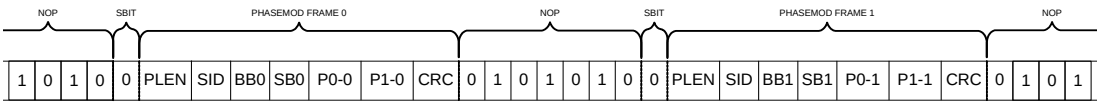


Figure 6-1 State diagram of the PHASEMOD generator.

Instructions are written into the PHASEMOD memory by the ISR during the previous interrupt. On each interrupt, before writing, the state machine must be cleared to reset the write and read addresses to zero and set the state machine to IDLE. This is only possible if the state machine has finished transmitting all the PHASEMOD bits and is already in IDLE state. This is normally the case because the interrupt occurs well after the STICK. In IDLE state, preamble, synchronized to STICK, is inserted into the PHASEMOD bit stream. On the next 128 MHz clock after an enabled STICK, the first word from the memory is read and the embedded command is decoded. If the command is SBIT, a zero will be inserted, the CRC calculation is cleared and the state remains READ. If the command is NOP, the first preamble bit will be inserted and the state will be set to NOP to do the rest of the bits and then return to READ. If the command is DATA, data[0] will be inserted and a frame flag is set. If the number of data bits to insert is 1 (width = 0), the state remains READ, otherwise the state is set to DATA to do the rest of the bits and then return to READ. If the command is CRC, a zero will be inserted and the state set to DAT to do the remaining 3 zeroes (the CRC is actually inserted later) and then return to

READ. If the command is END the first preamble bit will be inserted, the state will be set to IDLE to continue inserting preamble.

All of the phase models for a given station ID can be written to a single frame but here is an example of using two PHASEMOD frames. The actual insertion time of the phase models is not important because they are not used by the Baseline Board until the next STICK. At least TBD[SBIT] bits of preamble must be inserted before SBIT and at least TBD[frame] bits of preamble must be inserted between frames for the convenience of the Recirculation FPGA on the Baseline Board. SBIT + frame contains an odd number of bits and SBIT must always be placed where the preamble would have been 1. This condition is guaranteed by the hardware if SBIT were the first command in the memory; therefore, TBD[SBIT] must be even and TBD[frame] must be odd. The STATUS4 register contains error bits that indicate that SBIT has been placed incorrectly relative to the preamble.



To build the PHASEMOD frames as above, the corresponding phase model data should be encoded as follows:

SBIT ; // *insert start bit into PHASEMOD frame 0*

DATA PLEN; // *insert PLEN into PHASEMOD frame 0*

DATA SID; // *insert SID into PHASEMOD frame 0*

DATA BB0; // *insert BB0 into PHASEMOD frame 0*

DATA SB0; // *insert SB0 into PHASEMOD frame 0*

DATA P0-0[7:0]; // *insert P0-0[7:0] into PHASEMOD frame 0*

DATA P0-0[15:8]; // *insert P0-0[15:8] into PHASEMOD frame 0*

DATA P0-0[23:16]; // *insert P0-0[23:16] into PHASEMOD frame 0*

DATA P0-0[31:24]; // *insert P0-0[31:24] into PHASEMOD from 0*

DATA P1-0[7:0]; // *insert P1-0[7:0] into PHASEMOD frame 0*

DATA P1-0[15:8]; // *insert P1-0[15:8] into PHASEMOD frame 0*

DATA P1-0[23:16]; // *insert P1-0[23:16] into PHASEMOD frame 0*

DATA P1-0[31:24]; // *insert P1-0[31:24] into PHASEMOD frame 0*

CRC; // *insert CRC and Low bit into PHASEMOD frame 0*

NOP 7; // *insert seven preamble bits into PHASEMOD frame 0.*

SBIT ; // *insert start bit into PHASEMOD frame 1*

DATA PLEN; // *insert PLEN into PHASEMOD frame 1*

DATA SID; // *insert SID into PHASEMOD frame 1*

DATA BB1; // *insert BB0 into PHASEMOD frame 1*

DATA SB1; // *insert SB0 into PHASEMOD frame 1*

DATA P0-1[7:0]; // *insert P0-1[7:0] into PHASEMOD frame 1*

DATA P0-1[15:8]; // *insert P0-1[15:8] into PHASEMOD frame 1*

DATA P0-1[23:16]; // *insert P0-1[23:16] into PHASEMOD frame 1*

DATA P0-1[31:24]; // *insert P0-1[31:24] into PHASEMOD from 1*

DATA P1-1[7:0]; // *insert P1-1[7:0] into PHASEMOD frame 1*

DATA P1-1[15:8]; // *insert P1-1[15:8] into PHASEMOD frame 1*

DATA P1-1[23:16]; // *insert P1-1[23:16] into PHASEMOD frame 1*

DATA P1-1[31:24]; // *insert P1-1[31:24] into PHASEMOD frame 1*

CRC; // *insert 4-bit CRC into PHASEMOD frame 1*

END; // *stop building frame set and return to IDLE.*

Note: DATA BBx and DATA SBx can be combined as DATA BBx+SBx. Because BBx is 3-bit and SBx is 5-bit, they can be put together into 8-bits.

6.4 DUMPTRIG Generator

This section will explain how the DUMPTRIG generator works in more detail. The concept of building the DUMPTRIG frame is the same as building the PHASEMOD frame with two more commands. The dump trigger model data format is defined as follows.

Dump trigger model data format definition:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit#
Command					**Width**			**Data**								Content
Command					**Length**											Content

- **Data**: The 8-bit *data* field contains the data bits to be inserted into the DUMPTRIG frame. Data[0] will be put in the DUMPTRIG frame first. How many data bits are actually used to build the frame is defined in Width Bit.

- **Width**: The 3-bit *width* field specifies how many data bits are actually used to build the frame. It is defined as the following table.

Width	**Data bits used**
000	Data [0:0]
001	Data [1:0]
010	Data [2:0]
011	Data [3:0]
100	Data [4:0]
101	Data [5:0]
110	Data [5:0]
111	Data [7:0]

- **Length:** The 11-bit *length* field specifies the number of preamble bits for NOP and NOPL (see below).

- **Command**: The 5-bit *command* field tells the state machine how to put bits into the DUMPTRIG frame. Each command is defined as the following table.

Command	Command description
00000	**TRIG** - insert the trigger bit.
00001	**SBIT**- insert the start bit.
00010	**DATA-** insert *width* data bits.
00011	**CRC** - inserts 4 zero bits as a placeholder for the CRC.
00100	**END** - insert continuous preamble bits.
00101	**NOP** - insert *length* preamble bits.
00110	**NOPL** - insert *length* x2048 preamble bits.
All others	All others are reserved for possible future use.

Table 6-1 DUMPTRIG Commands.

Use of each command will be explained in detail below.

TRIG: This command tells the state machine to insert the trigger bit. Width and Data bits are ignored in this word. The trigger must occur where there would have been a zero in the preamble. This condition is guaranteed by the hardware if the first command in the memory after an enabled STICK is TRIG.

SBIT: This command tells the state machine to insert the start bit. It also indicates the start of a DUMPTRIG frame. Other bits are ignored in this word. The start bit must occur where there would have been a 1 in the preamble.

DATA: This command tells the state machine to insert data bits into the frame. Width specifies the number of data bits to be inserted.

CRC: This command tells the state machine to insert the preamble bits into the frame as a placeholder for the CRC. The CRC is generated in the Timing FPGA. It also indicates the end of a DUMPTRIG frame. Other bits are ignored in this word.

NOP: This command tells the state machine to insert preamble. Bits [10:0] specify the number of bits of preamble to insert.

NOPL: This command tells the state machine to insert lots of preamble. Bits [10:0] shifted left by 11 bits (time 2048) specify the number of bits of preamble to insert.

END: This command is used to tell the state machine to stop reading instructions and go to the IDLE state which continues to insert preamble until the next system tick. Other bits are ignored in this word. Unlike the PHASEMOD generator, the END command is only used at the end of an observation.

Following state diagram illustrates the DUMPTRIG state machine's operation.

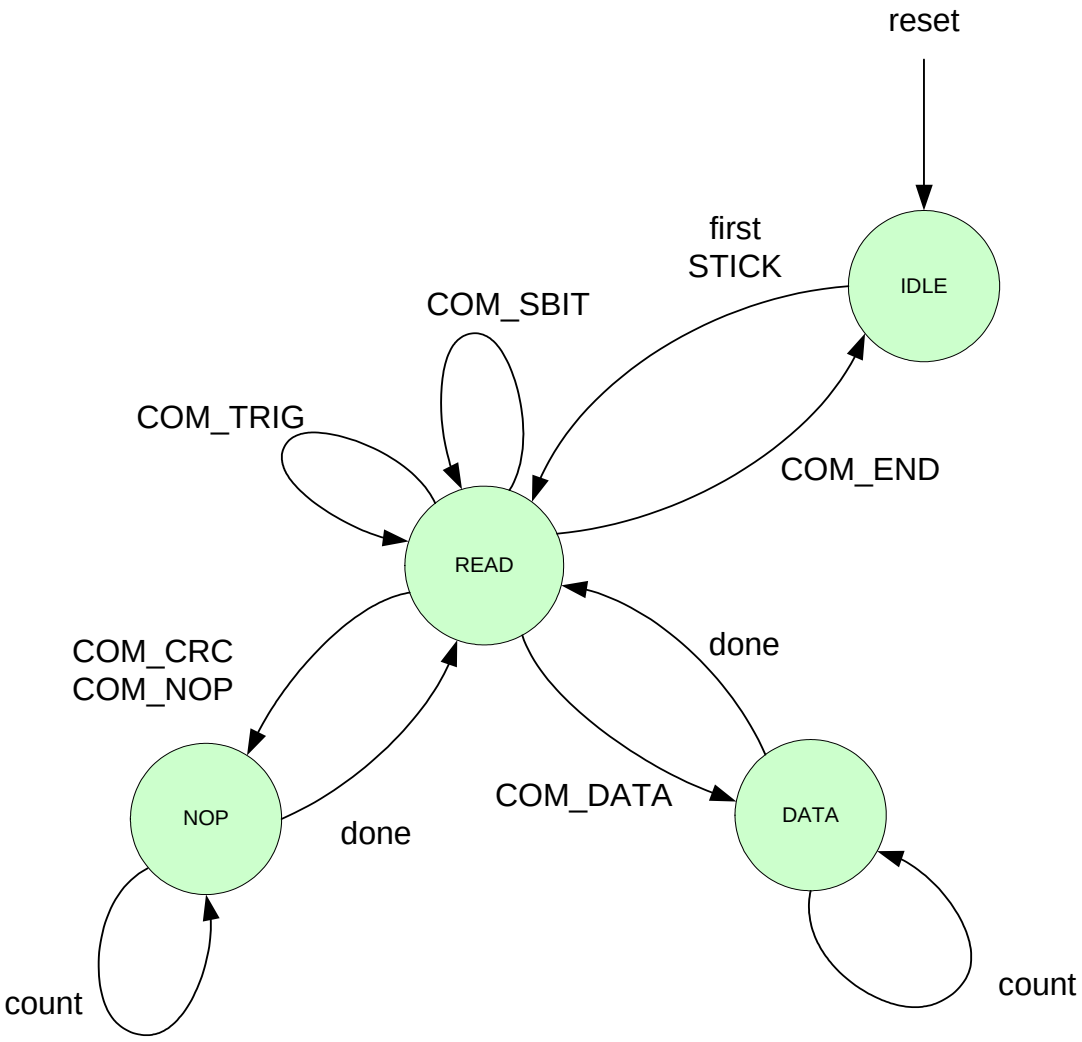


Figure 6-2 State diagram of the DUMPTRIG generator.

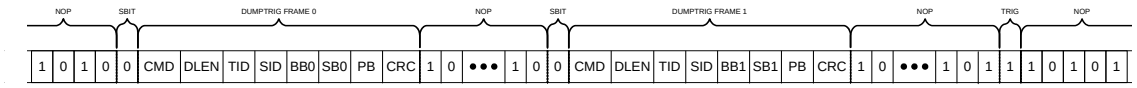
Before writing instructions to the DUMPTRIG memory for the first time, the state machine must be cleared. This clearing resets the read and write addresses to zero. Unlike the PHASEMOD case, this clearing is only done before the first write. Triggers must occur several times across a 10 millisecond tick interval and, even worse, in the case of pulsars the triggers are not synchronous with STICK. The DUMPTRIG state machine is only started once by the first STICK that occurs after the DUMPTRIG arm control bit is set and then follows the instructions written to the circular memory after that until it is sent back to the IDLE state by software clear or END instruction (usually at the end of the observation). This means that the DUMPTRIG arm bit in the CONTROL register should only be set for one interrupt and that 20 milliseconds of instructions should be written the first time. The writing of commands has to be done very carefully during the ISR to make sure that the triggers occur when they are needed and that the buffer never becomes empty or full. To make the trigger occur at the desired time, the

number of bits of preamble (NOP) is set accordingly before and after frames to keep everything timed correctly relative to STICK. There is a counter (DTTRIGCNT**)** to measure the time interval between the last trigger produced before an STICK and the STICK itself. This can be used to verify that the triggers occurred when they were expected.

To start a sequence of triggers from a disabled and cleared state, the ISR would write 20 milliseconds worth of instructions into the memory and set the DUMPTRIG arm bit. At the next STICK the state will change to READ and the first bit of the first instruction would be inserted into the DUMPTRIG bit stream aligned with the T bit in the TIMECODE. If the first command is TRIG, the resulting trigger bit will be aligned with the TIMECODE T bit. The next interrupt would clear the DUMPTRIG arm bit and write the next 10 milliseconds worth of instructions into the memory. Subsequent interrupts would continue writing instructions into the memory until the observation is complete. At this time, an END command or a software clear will return the state machine to IDLE inserting continuous preamble.

All of the dump commands can be written to a single frame but here is an example using two DUMPTRIG frames. Like PHASEMOD, at least TBD[frame] bits of preamble must be inserted between frames. The number of bits of preamble between frames must be odd because SBIT + frames contain an odd number of bits and the start bit must always be placed where the preamble would have been 1. Similarly, the trigger bit must be placed where the preamble would have been zero. This condition is guaranteed by the hardware if TRIG is the first command to be written to the RAM (note that this is different than the PHASEMOD case). The STATUS4 register contains error bits that indicate that SBIT and TRIG have been placed incorrectly relative to the preamble. For normal operation the number of bits inserted between STICKS must be 1280000, the number of bits inserted between triggers no more than ~64000 (0.5 milliseconds) to prevent Correlator Chip overflows, the triggers evenly spaced and STICK always aligned with one of the triggers. The period between triggers must be even because the triggers must occur where there is a “1” in the preamble; therefore, if the desired dump interval is an even number of bits then the trigger interval will be constant but if the desired dump interval is an odd number of bits then the trigger interval will have to “dither” plus and minus one bit.

For pulsar observations all bets are off but, when possible, a synchronizing frame with an STICK aligned trigger should be placed in the DUMPTRIG bit stream. Even for pulsars, the position of TRIG and SBIT must be correct relative to the preamble; therefore, the position of TRIG will normally have to “dither” plus and minus one bit around where you would like to put it.



To build the DUMPTRIG frames as above, the corresponding dump trigger model data should be encoded as follows:

	#bits
SBIT ; // *insert start bit into DUMPTRIG frame 0*	1
DATA CMD; // *insert CMD into DUMPTRIG frame 0*	3
DATA DLEN; // *insert DLEN into DUMPTRIG frame 0*	8
DATA TID; // *insert TID into DUMPTRIG frame 0*	4
DATA SID; // *insert SID into DUMPTRIG frame 0*	8
DATA BB0; // *insert BB0 into DUMPTRIG frame 0*	5
DATA SB0; // *insert SB0 into DUMPTRIG frame 0*	5
DATA PB[7:0]; // *insert PB[7:0] into DUMPTRIG frame 0*	8
DATA PB[15:8]; // *insert PB[15:8] into DUMPTRIG frame 0*	8
CRC; // *insert CRC into DUMPTRIG frame 0*	4
NOP 19; // *insert 19 preamble bits after DUMPTRIG frame 0*	19
SBIT ; // *insert start bit into DUMPTRIG frame 1*	1
DATA CMD; // *insert CMD into DUMPTRIG frame 1*	3
DATA DLEN; // *insert DLEN into DUMPTRIG frame 1*	8
DATA TID; // *insert TID into DUMPTRIG frame 1*	4
DATA SID; // *insert SID into DUMPTRIG frame 1*	8
DATA BB1; // *insert BB1 into DUMPTRIG frame 1*	5
DATA SB1; // *insert SB1 into DUMPTRIG frame 1*	5
DATA PB[7:0]; // *insert PB[7:0] into DUMPTRIG frame 1*	8
DATA PB[15:8]; // *insert PB[15:8] into DUMPTRIG frame 1*	8
CRC; // *insert CRC into DUMPTRIG frame 1*	4
NOP 45; // *insert 45 preamble bits after DUMPTRIG frame 1*	45
TRIG; // *insert trigger bit for DUMPTRIG frame set*	1
NOP 339; // *insert 339 preamble bits after DUMPTRIG frame 1*	339
NOPL 156; // *insert 156*2048 preamble bits after DUMPTRIG trigger*	319488
Total bits (2.5 milliseconds)	320000
Back to SBIT and repeat with next phase bin	

6.5 Extended DUMPTRIG Generator

The good news is that the previous description of the DUMPTRIG Generator is still true and DUMPTRIGs can be generated in the same manner as before. Everything described in this section is in addition to that described above. The reason for the change is to reduce the number of HES writes required to run an experiment. This would allow for faster dumping of the correlator and reduce the network traffic. The main element of the change is the incorporation of a loop structure in the HES allowing a group of instructions to be processed multiple times. In fact two loops are possible – an inner loop and an outer loop. The inner loop would normally be used to add correlator dumps to an LTA integration and the outer loop would be used to produce multiple integrations. The bad news is that the Timing FPGA is becoming rather full and this limits flexibility. For example, there is no way of dealing with the Trigger ID which would normally be incremented after each trigger. This deficiency should not be a problem, however, because the Trigger ID is not checked by the Crossbar Boards or the Baseline Boards (Brent Carlson – Private Communication).

Command	Command description
00000	**TRIG** - insert the trigger bit.
00001	**SBIT**- insert the start bit.
00010	**DATA**- insert *width* data bits.
00011	**CRC** - inserts 4 zero bits as a placeholder for the CRC.
00100	**END** - insert continuous preamble bits.
00101	**NOP** - insert *length* preamble bits.
00110	**NOPL** - insert *length* x2048 preamble bits.
00111	**PMAX** - insert preamble bit, set phase bin maximum.
01000	**PSET** - insert preamble bit, set phase bin number.
01001	**BSET** - insert preamble bit, set bank number and switching state.
01010	**CNT0** - insert preamble bit, set inner loop count.
01011	**CNT1** - insert preamble bit, set outer loop count.
01100	**PINC** - insert preamble bit, increment phase bin.
01101	**PPUT** - insert phase bin (16 bits).
01110	**JMP0** - insert preamble bit, inner loop jump back.
01111	**JMP1** - insert preamble bit, outer loop jump back.

Table 6-2 Extended DUMPTRIG Commands.

Use of each command will be explained in detail below.

TRIG: This command tells the state machine to insert the trigger bit. Width and Data bits are ignored in this word. The trigger must occur where there would have been a zero in the preamble. This condition is guaranteed by the hardware if the first command in the memory after an enabled STICK is TRIG.

SBIT: This command tells the state machine to insert the start bit. It also indicates the start of a DUMPTRIG frame. Other bits are ignored in this word. The start bit must occur where there would have been a 1 in the preamble.

DATA: This command tells the state machine to insert data bits into the frame. Width specifies the number of data bits to be inserted.

CRC: This command tells the state machine to insert the preamble bits into the frame as a placeholder for the CRC. The CRC is generated in the Timing FPGA. It also indicates the end of a DUMPTRIG frame. Other bits are ignored in this word.

NOP: This command tells the state machine to insert preamble. Bits [10:0] specify the number of bits of preamble to insert.

NOPL: This command tells the state machine to insert lots of preamble. Bits [10:0] shifted left by 11 bits (time 2048) specify the number of bits of preamble to insert.

END: This command is used to tell the state machine to stop reading instructions and go to the IDLE state which continues to insert preamble until the next system tick. Other bits are ignored in this word. Unlike the PHASEMOD generator, the END command is only used at the end of an observation.

PMAX: This command is used set the maximum value of phase bin. The value is placed in the length bits (11 bits). To cycle phase bin from 0 to 99 put in 99. Must be less than 2000. One preamble bit is inserted in the DUMPTRIG stream.

PSET: This command is used set the value of phase bin. The value is placed in the length bits (11 bits). Must be less than PMAX. One preamble bit is inserted in the DUMPTRIG stream.

BSET: This command is used set the value of the phase bin bank (1 bit) and whether bank switching is on (1 bit). The bank is placed in bit 0 and the switching flag in bit 1. . One preamble bit is inserted in the DUMPTRIG stream.

CNT0: This command is used set the value of the inner loop counter. The value is placed in the data bits (8 bits). If the width bits equal 0 then the LS 8 bits are set. If the width bits equal 1 then the MS 8 bits are set. The inner loop count is limited to 65535 and is one less than the number of times the following instructions are to be executed. If the loop is to be executed less than 256 times only one CNT0 instruction is necessary. One preamble bit is inserted in the DUMPTRIG stream.

CNT1: This command is used set the value of the outer loop counter. The value is placed in the data bits (8 bits). If the width bits equal 0 then the LS 8 bits are set. If the

width bits equal 1 then the MS 8 bits are set. The outer loop count is limited to 65535 and is one less than the number of times the following instructions are to be executed. If the loop is to be executed less than 256 times only one CNT1 instruction is necessary. One preamble bit is inserted in the DUMPTRIG stream There is a working version of the FPGA that allows the outer loop counter to be 20 bits. If this version is adopted then the extra 4 bits are written when the width bits equal 2. In this case three CNT1 instructions are necessary to convey all 20 bits.

PINC: This command is used increment the value of the phase bin. The phase bin will reset to zero after it reaches the maximum. If bank switching is on then the bank will switch when the phase bin reaches the maximum. One preamble bit is inserted in the DUMPTRIG stream.

JMP0: This command is used to jump back to the start of the inner loop. The number of HES instructions to jump back is placed in the length bits (11 bits). The actual number does not include the jump instruction but must be increased by 4 instructions because of the latency of the RAM and its associated FIFO. For the same reason the jump instruction must be followed by four NOP 1 instructions. One preamble bit is inserted in the DUMPTRIG stream.

JMP1: This command is used to jump back to the start of the outer loop. The number of HES instructions to jump back is placed in the length bits (11 bits). The actual number does not include the jump instruction but must be increased by 4 instructions because of the latency of the RAM and its associated FIFO. For the same reason the jump instruction must be followed by four NOP 1 instructions. One preamble bit is inserted in the DUMPTRIG stream.

Other looping considerations:

Once one tries to write the script for a real experiment it will be realized that the number of bits of preamble inserted before the trigger is likely different the first time through the loop than all subsequent times. All CNT instructions set a flag which will cause the following (~mandatory) NOP instruction to insert 16 bits the first time through and the specified number of bits after that.

Watch the placement of the SBIT instruction to make sure it lands where the preamble would have been a 1. Similarly, the TRIG bit must be inserted where the preamble would have been a 0.

The current set of instructions must be followed by at least four more instructions or else a RAM error will occur. If the whole experiment can be done in one outer loop then follow by four END instructions. If not, then start by writing and extra set of instructions at the beginning.

An example set of instructions is given below.

```
00 PMAX 1999                    1  set phase bin Maximum to 1999
01 PSET 0                       1  set phase bin to 0
02 BSET 2                       1  set bank to 0, switching on
03 NOP  10                     10 set even number of NOPs for SBIT
04 SBIT                         1  start-bit on odd bit
05 DAT  2 DAD                   3  Dump and Discard
06 DAT  7 1                     8  DLEN
07 DAT  3 0                     4  TID
08 DAT  7 sid                   8  SID
09 DAT  4 bbid                  5  BBID
10 DAT  4 sbid                  5  SBID
11 PPUT                        16 insert phase bin
12 CRC                          4  insert CRC placeholder
13 NOPL (nclk-nbit)/2048
14 NOP  (nclk-nbit)%2048           bit count is now even
15 TRIG                         1  DAD trigger bit
16 CNT1 0 LS(nint-1)            1  set LS part of outer loop count
17 CNT1 1 MS(nint-1)            1  set MS part of outer loop count
18 NOP  13                     16 set 16 NOPs first time, 13 after
19 NOP  10                     10 set even number of NOPs for SBIT
20 SBIT                         1  start bit
21 DAT  2 FDS                   3  First Dump and Save
22 DAT  7 1                     8  DLEN
23 DAT  3 0                     4  TID
24 DAT  7 sid                   8  SID
25 DAT  4 bbid                  5  BBID
26 DAT  4 sbid                  5  SBID
27 PPUT                        16 insert phase bin
28 CRC                          4  insert CRC placeholder
29 NOPL (nclk-nbit)/2048
30 NOP  (nclk-nbit)%2048           bit count is now even
31 TRIG                         1  FDS trigger bit
32 CNT0 0 LS(ndmp-3)            1  set LS part of inner loop count
33 CNT0 1 MS(ndmp-3)            1  set MS part of inner loop count
34 NOP  13                     16 set 16 NOPs first time, 13 after
35 NOP  10                     10 set even number of NOPs for SBIT
36 SBIT                         1  start bit
37 DAT  2 AFS                   3  Add Dump and Save
38 DAT  7 1                     8  DLEN
39 DAT  3 0                     4  TID
40 DAT  7 sid                   8  SID
41 DAT  4 bbid                  5  BBID
42 DAT  4 sbid                  5  SBID
43 PPUT                        16 insert phase bin
44 CRC                          4  insert CRC placeholder
45 NOPL (nclk-nbit)/2048
46 NOP  (nclk-nbit)%2048           bit count is now even
47 TRIG                         1  AFS trigger bit
48 JMP0 18                      1  jump back to start of inner loop
49 NOP  1                       1  latency NOP
50 NOP  1                       1  latency NOP
51 NOP  1                       1  latency NOP
```

```
52 NOP  1                       1  latency NOP
53 NOP  11                      11 set odd number of NOPs for SBIT
54 SBIT                         1  start bit
55 DAT  2 LDS                   3  Last Dump and Save
56 DAT  7 1                     8  DLEN
57 DAT  3 0                     4  TID
58 DAT  7 sid                   8  SID
59 DAT  4 bbid                  5  BBID
60 DAT  4 sbid                  5  SBID
61 PPUT                         16 insert phase bin
62 CRC                          4  insert CRC placeholder
63 PINC                         1  increment phase bin
64 NOPL (nclk-nbit)/2048
65 NOP  (nclk-nbit)%2048           bit count is now even
66 TRIG                         1  LDS trigger bit
67 JMP1 53                      1  jump back to start of outer loop
68 NOP  1                       1  latency NOP
69 NOP  1                       1  latency NOP
70 NOP  1                       1  latency NOP
71 NOP  1                       1  latency NOP
72 END  1                       1  RAM error prevention
73 END  1                       1  RAM error prevention
74 END  1                       1  RAM error prevention
75 END  1                       1  RAM error prevention
```

Notes:

Instruction 48: the jump offset is 48 – 34 + 4 = 18

Instruction 67: the jump offset is 67 – 18 + 4 = 53

Instruction 67: if the jump offset were 55 then instructions 16 and 17 would be executed every outer loop effectively producing an endless loop.

Instruction 18: the bit count from instructions 15-17 is 3 and the bit count from line 66-71 is 6 so instruction 18 inserts 16 NOPS the first time and 13 NOPs the rest of the time.

Instruction 34: the bit count from instructions 31-33 is 3 and the bit count from lines 47-52 is 6 so instruction 34 inserts 16 NOPs the first time (every outer loop) and 13 NOPs the rest of the time.

If it is desired to add more instructions to prolong the experiment then instructions 72-75 have to be replaced by these extra instructions. At least four of them have to be written before starting the generator to avoid a RAM error.

7 Pinouts, Pin Locations and Programming Notes

7.1 Pinouts by signal name

Note: UNUSED and NC pins have been removed.

```
Quartus II Version 9.1 Build 304 01/25/2010 Service Pack 1 SJ Full Version
CHIP  "timing_top"  ASSIGNED TO AN: EP1SGX25CF672C5

-------------------------------------------------------------------------------------------
Pin Name/Usage               : Location  : Dir.   : I/O Standard       : Voltage : I/O Bank  :
-------------------------------------------------------------------------------------------
control_out[0]               : H19       : output : LVDS              :         : 2         :
control_out[0](n)            : H20       : output : LVDS              :         : 2         :
control_out[1]               : K17       : output : LVDS              :         : 2         :
control_out[1](n)            : K18       : output : LVDS              :         : 2         :
control_out[10]              : T19       : output : LVDS              :         : 1         :
control_out[10](n)           : T20       : output : LVDS              :         : 1         :
control_out[11]              : T21       : output : LVDS              :         : 1         :
control_out[11](n)           : T22       : output : LVDS              :         : 1         :
control_out[12]              : U17       : output : LVDS              :         : 1         :
control_out[12](n)           : U18       : output : LVDS              :         : 1         :
control_out[13]              : U19       : output : LVDS              :         : 1         :
control_out[13](n)           : U20       : output : LVDS              :         : 1         :
control_out[14]              : U21       : output : LVDS              :         : 1         :
control_out[14](n)           : U22       : output : LVDS              :         : 1         :
control_out[15]              : V21       : output : LVDS              :         : 1         :
control_out[15](n)           : V22       : output : LVDS              :         : 1         :
control_out[16]              : V19       : output : LVDS              :         : 1         :
control_out[16](n)           : V20       : output : LVDS              :         : 1         :
control_out[17]              : W19       : output : LVDS              :         : 1         :
control_out[17](n)           : W20       : output : LVDS              :         : 1         :
```

control_out[2]	J19	output	LVDS		2	
control_out[2](n)	J20	output	LVDS		2	
control_out[3]	K19	output	LVDS		2	
control_out[3](n)	K20	output	LVDS		2	
control_out[4]	K21	output	LVDS		2	
control_out[4](n)	K22	output	LVDS		2	
control_out[5]	L19	output	LVDS		2	
control_out[5](n)	L20	output	LVDS		2	
control_out[6]	L21	output	LVDS		2	
control_out[6](n)	L22	output	LVDS		2	
control_out[7]	M17	output	LVDS		2	
control_out[7](n)	M18	output	LVDS		2	
control_out[8]	M19	output	LVDS		2	
control_out[8](n)	M20	output	LVDS		2	
control_out[9]	M21	output	LVDS		2	
control_out[9](n)	M22	output	LVDS		2	
mcb_addr[0]	AD4	input	3.3-V LVTTL		7	
mcb_addr[1]	AE4	input	3.3-V LVTTL		7	
mcb_addr[2]	AF4	input	3.3-V LVTTL		7	
mcb_addr[3]	AC5	input	3.3-V LVTTL		7	
mcb_addr[4]	AD5	input	3.3-V LVTTL		7	
mcb_addr[5]	AE5	input	3.3-V LVTTL		7	
mcb_addr[6]	AF5	input	3.3-V LVTTL		7	
mcb_addr[7]	AC6	input	3.3-V LVTTL		7	
mcb_clk	AB12	input	3.3-V LVTTL		7	
mcb_cs_l	AE3	input	3.3-V LVTTL		7	
mcb_data[0]	AD6	bidir	3.3-V LVTTL		7	
mcb_data[1]	AE6	bidir	3.3-V LVTTL		7	
mcb_data[10]	AE7	bidir	3.3-V LVTTL		7	
mcb_data[11]	AF7	bidir	3.3-V LVTTL		7	
mcb_data[12]	U8	bidir	3.3-V LVTTL		7	
mcb_data[13]	V8	bidir	3.3-V LVTTL		7	
mcb_data[14]	W8	bidir	3.3-V LVTTL		7	
mcb_data[15]	Y8	bidir	3.3-V LVTTL		7	

mcb_data[2]	AF6	bidir	3.3-V LVTTL		7
mcb_data[3]	V7	bidir	3.3-V LVTTL		7
mcb_data[4]	W7	bidir	3.3-V LVTTL		7
mcb_data[5]	Y7	bidir	3.3-V LVTTL		7
mcb_data[6]	AA7	bidir	3.3-V LVTTL		7
mcb_data[7]	AB7	bidir	3.3-V LVTTL		7
mcb_data[8]	AC7	bidir	3.3-V LVTTL		7
mcb_data[9]	AD7	bidir	3.3-V LVTTL		7
mcb_int	AE2	output	3.3-V LVTTL		7
mcb_rd_wr_l	AC4	input	3.3-V LVTTL		7
out_tick_a	E19	input	3.3-V LVTTL		2
out_tick_b	Y21	input	3.3-V LVTTL		1
pc_clk_a_i	AE14	input	LVDS		8
pc_clk_a_i(n)	AF14	input	LVDS		8
pc_clk_a_o	AA19	output	LVDS		1
pc_clk_a_o(n)	AA20	output	LVDS		1
pc_clk_b_i	AC14	input	LVDS		8
pc_clk_b_i(n)	AD14	input	LVDS		8
pc_clk_b_o	AB19	output	LVDS		1
pc_clk_b_o(n)	AB20	output	LVDS		1
pc_dat_a_i	N23	input	LVDS		2
pc_dat_a_i(n)	N24	input	LVDS		2
pc_dat_a_o	AC19	output	LVDS		1
pc_dat_a_o(n)	AC20	output	LVDS		1
pc_dat_b_i	J25	input	LVDS		2
pc_dat_b_i(n)	J26	input	LVDS		2
pc_dat_b_o	AC21	output	LVDS		1
pc_dat_b_o(n)	AC22	output	LVDS		1
pclk_0_a	A5	input	1.5-V HSTL Class I		4
pclk_0_b	AF8	input	1.5-V HSTL Class I		7
pclk_1_a	B6	input	1.5-V HSTL Class I		4
pclk_1_b	AD9	input	1.5-V HSTL Class I		7
pclk_10_a	L16	input	1.5-V HSTL Class I		3
pclk_10_b	AD18	input	1.5-V HSTL Class I		8

pclk_11_a	F17	input	1.5-V HSTL Class I		3
pclk_11_b	AD20	input	1.5-V HSTL Class I		8
pclk_12_a	F18	input	1.5-V HSTL Class I		3
pclk_12_b	AF22	input	1.5-V HSTL Class I		8
pclk_13_a	D19	input	1.5-V HSTL Class I		3
pclk_13_b	Y25	input	1.5-V HSTL Class I		1
pclk_14_a	B21	input	1.5-V HSTL Class I		3
pclk_14_b	V26	input	1.5-V HSTL Class I		1
pclk_15_a	B24	input	1.5-V HSTL Class I		3
pclk_15_b	U26	input	1.5-V HSTL Class I		1
pclk_16_a	H26	input	1.5-V HSTL Class I		2
pclk_16_b	R24	input	1.5-V HSTL Class I		1
pclk_17_a	J22	input	1.5-V HSTL Class I		2
pclk_17_b	R22	input	1.5-V HSTL Class I		1
pclk_2_a	C7	input	1.5-V HSTL Class I		4
pclk_2_b	AE10	input	1.5-V HSTL Class I		7
pclk_3_a	J7	input	1.5-V HSTL Class I		4
pclk_3_b	AE11	input	1.5-V HSTL Class I		7
pclk_4_a	G8	input	1.5-V HSTL Class I		4
pclk_4_b	AC13	input	1.5-V HSTL Class I		11
pclk_5_a	G9	input	1.5-V HSTL Class I		4
pclk_5_b	AE15	input	1.5-V HSTL Class I		8
pclk_6_a	H11	input	1.5-V HSTL Class I		4
pclk_6_b	T16	input	1.5-V HSTL Class I		8
pclk_7_a	G12	input	1.5-V HSTL Class I		4
pclk_7_b	Y15	input	1.5-V HSTL Class I		8
pclk_8_a	F14	input	1.5-V HSTL Class I		10
pclk_8_b	AA17	input	1.5-V HSTL Class I		8
pclk_9_a	B16	input	1.5-V HSTL Class I		3
pclk_9_b	AB16	input	1.5-V HSTL Class I		8
pdata_0_a[0]	A4	input	1.5-V HSTL Class I		4
pdata_0_a[1]	B4	input	1.5-V HSTL Class I		4
pdata_0_a[2]	C4	input	1.5-V HSTL Class I		4
pdata_0_a[3]	D4	input	1.5-V HSTL Class I		4

pdata_0_b[0]	AA8	input	1.5-V HSTL Class I		7
pdata_0_b[1]	AB8	input	1.5-V HSTL Class I		7
pdata_0_b[2]	AC8	input	1.5-V HSTL Class I		7
pdata_0_b[3]	AD8	input	1.5-V HSTL Class I		7
pdata_1_a[0]	B5	input	1.5-V HSTL Class I		4
pdata_1_a[1]	C5	input	1.5-V HSTL Class I		4
pdata_1_a[2]	D5	input	1.5-V HSTL Class I		4
pdata_1_a[3]	E5	input	1.5-V HSTL Class I		4
pdata_1_b[0]	U9	input	1.5-V HSTL Class I		7
pdata_1_b[1]	W9	input	1.5-V HSTL Class I		7
pdata_1_b[2]	Y9	input	1.5-V HSTL Class I		7
pdata_1_b[3]	AB9	input	1.5-V HSTL Class I		7
pdata_10_a[0]	C16	input	1.5-V HSTL Class I		3
pdata_10_a[1]	D16	input	1.5-V HSTL Class I		3
pdata_10_a[2]	E16	input	1.5-V HSTL Class I		3
pdata_10_a[3]	G16	input	1.5-V HSTL Class I		3
pdata_10_b[0]	W18	input	1.5-V HSTL Class I		8
pdata_10_b[1]	Y18	input	1.5-V HSTL Class I		8
pdata_10_b[2]	AA18	input	1.5-V HSTL Class I		8
pdata_10_b[3]	AB18	input	1.5-V HSTL Class I		8
pdata_11_a[0]	A17	input	1.5-V HSTL Class I		3
pdata_11_a[1]	B17	input	1.5-V HSTL Class I		3
pdata_11_a[2]	C17	input	1.5-V HSTL Class I		3
pdata_11_a[3]	D17	input	1.5-V HSTL Class I		3
pdata_11_b[0]	AE18	input	1.5-V HSTL Class I		8
pdata_11_b[1]	AF18	input	1.5-V HSTL Class I		8
pdata_11_b[2]	AD19	input	1.5-V HSTL Class I		8
pdata_11_b[3]	AE19	input	1.5-V HSTL Class I		8
pdata_12_a[0]	A18	input	1.5-V HSTL Class I		3
pdata_12_a[1]	B18	input	1.5-V HSTL Class I		3
pdata_12_a[2]	C18	input	1.5-V HSTL Class I		3
pdata_12_a[3]	D18	input	1.5-V HSTL Class I		3
pdata_12_b[0]	AE20	input	1.5-V HSTL Class I		8
pdata_12_b[1]	AF20	input	1.5-V HSTL Class I		8

pdata_12_b[2]	AD21	input	1.5-V HSTL Class I		8
pdata_12_b[3]	AE21	input	1.5-V HSTL Class I		8
pdata_13_a[0]	G18	input	1.5-V HSTL Class I		3
pdata_13_a[1]	H18	input	1.5-V HSTL Class I		3
pdata_13_a[2]	A19	input	1.5-V HSTL Class I		3
pdata_13_a[3]	B19	input	1.5-V HSTL Class I		3
pdata_13_b[0]	AD22	input	1.5-V HSTL Class I		8
pdata_13_b[1]	AE22	input	1.5-V HSTL Class I		8
pdata_13_b[2]	AF23	input	1.5-V HSTL Class I		8
pdata_13_b[3]	Y23	input	1.5-V HSTL Class I		1
pdata_14_a[0]	A20	input	1.5-V HSTL Class I		3
pdata_14_a[1]	B20	input	1.5-V HSTL Class I		3
pdata_14_a[2]	C20	input	1.5-V HSTL Class I		3
pdata_14_a[3]	D20	input	1.5-V HSTL Class I		3
pdata_14_b[0]	Y26	input	1.5-V HSTL Class I		1
pdata_14_b[1]	W23	input	1.5-V HSTL Class I		1
pdata_14_b[2]	W24	input	1.5-V HSTL Class I		1
pdata_14_b[3]	W25	input	1.5-V HSTL Class I		1
pdata_15_a[0]	A22	input	1.5-V HSTL Class I		3
pdata_15_a[1]	B22	input	1.5-V HSTL Class I		3
pdata_15_a[2]	C22	input	1.5-V HSTL Class I		3
pdata_15_a[3]	A23	input	1.5-V HSTL Class I		3
pdata_15_b[0]	V23	input	1.5-V HSTL Class I		1
pdata_15_b[1]	V24	input	1.5-V HSTL Class I		1
pdata_15_b[2]	V25	input	1.5-V HSTL Class I		1
pdata_15_b[3]	U24	input	1.5-V HSTL Class I		1
pdata_16_a[0]	F25	input	1.5-V HSTL Class I		2
pdata_16_a[1]	F26	input	1.5-V HSTL Class I		2
pdata_16_a[2]	G25	input	1.5-V HSTL Class I		2
pdata_16_a[3]	G26	input	1.5-V HSTL Class I		2
pdata_16_b[0]	U23	input	1.5-V HSTL Class I		1
pdata_16_b[1]	T23	input	1.5-V HSTL Class I		1
pdata_16_b[2]	T24	input	1.5-V HSTL Class I		1
pdata_16_b[3]	T25	input	1.5-V HSTL Class I		1

pdata_17_a[0]	H23	input	1.5-V HSTL Class I		2
pdata_17_a[1]	H24	input	1.5-V HSTL Class I		2
pdata_17_a[2]	J23	input	1.5-V HSTL Class I		2
pdata_17_a[3]	J24	input	1.5-V HSTL Class I		2
pdata_17_b[0]	P23	input	1.5-V HSTL Class I		1
pdata_17_b[1]	P24	input	1.5-V HSTL Class I		1
pdata_17_b[2]	P25	input	1.5-V HSTL Class I		1
pdata_17_b[3]	R18	input	1.5-V HSTL Class I		1
pdata_2_a[0]	C6	input	1.5-V HSTL Class I		4
pdata_2_a[1]	D6	input	1.5-V HSTL Class I		4
pdata_2_a[2]	E6	input	1.5-V HSTL Class I		4
pdata_2_a[3]	A7	input	1.5-V HSTL Class I		4
pdata_2_b[0]	AE9	input	1.5-V HSTL Class I		7
pdata_2_b[1]	AF9	input	1.5-V HSTL Class I		7
pdata_2_b[2]	R10	input	1.5-V HSTL Class I		7
pdata_2_b[3]	W10	input	1.5-V HSTL Class I		7
pdata_3_a[0]	D7	input	1.5-V HSTL Class I		4
pdata_3_a[1]	E7	input	1.5-V HSTL Class I		4
pdata_3_a[2]	F7	input	1.5-V HSTL Class I		4
pdata_3_a[3]	G7	input	1.5-V HSTL Class I		4
pdata_3_b[0]	AF10	input	1.5-V HSTL Class I		7
pdata_3_b[1]	U11	input	1.5-V HSTL Class I		7
pdata_3_b[2]	W11	input	1.5-V HSTL Class I		7
pdata_3_b[3]	Y11	input	1.5-V HSTL Class I		7
pdata_4_a[0]	A8	input	1.5-V HSTL Class I		4
pdata_4_a[1]	B8	input	1.5-V HSTL Class I		4
pdata_4_a[2]	C8	input	1.5-V HSTL Class I		4
pdata_4_a[3]	D8	input	1.5-V HSTL Class I		4
pdata_4_b[0]	AC12	input	1.5-V HSTL Class I		7
pdata_4_b[1]	AD12	input	1.5-V HSTL Class I		11
pdata_4_b[2]	AE12	input	1.5-V HSTL Class I		11
pdata_4_b[3]	AA13	input	1.5-V HSTL Class I		11
pdata_5_a[0]	H8	input	1.5-V HSTL Class I		4
pdata_5_a[1]	A9	input	1.5-V HSTL Class I		4

pdata_5_a[2]	B9	input	1.5-V HSTL Class I		4
pdata_5_a[3]	C9	input	1.5-V HSTL Class I		4
pdata_5_b[0]	AD13	input	1.5-V HSTL Class I		11
pdata_5_b[1]	AA14	input	1.5-V HSTL Class I		12
pdata_5_b[2]	AB14	input	1.5-V HSTL Class I		12
pdata_5_b[3]	AC15	input	1.5-V HSTL Class I		8
pdata_6_a[0]	A10	input	1.5-V HSTL Class I		4
pdata_6_a[1]	E10	input	1.5-V HSTL Class I		4
pdata_6_a[2]	G10	input	1.5-V HSTL Class I		4
pdata_6_a[3]	E11	input	1.5-V HSTL Class I		4
pdata_6_b[0]	R13	input	1.5-V HSTL Class I		8
pdata_6_b[1]	R15	input	1.5-V HSTL Class I		8
pdata_6_b[2]	T15	input	1.5-V HSTL Class I		8
pdata_6_b[3]	U15	input	1.5-V HSTL Class I		8
pdata_7_a[0]	E13	input	1.5-V HSTL Class I		9
pdata_7_a[1]	F13	input	1.5-V HSTL Class I		9
pdata_7_a[2]	D12	input	1.5-V HSTL Class I		4
pdata_7_a[3]	E12	input	1.5-V HSTL Class I		4
pdata_7_b[0]	U16	input	1.5-V HSTL Class I		8
pdata_7_b[1]	V16	input	1.5-V HSTL Class I		8
pdata_7_b[2]	W16	input	1.5-V HSTL Class I		8
pdata_7_b[3]	Y16	input	1.5-V HSTL Class I		8
pdata_8_a[0]	A13	input	1.5-V HSTL Class I		10
pdata_8_a[1]	B13	input	1.5-V HSTL Class I		10
pdata_8_a[2]	C14	input	1.5-V HSTL Class I		3
pdata_8_a[3]	D14	input	1.5-V HSTL Class I		3
pdata_8_b[0]	AC16	input	1.5-V HSTL Class I		8
pdata_8_b[1]	AD16	input	1.5-V HSTL Class I		8
pdata_8_b[2]	AE16	input	1.5-V HSTL Class I		8
pdata_8_b[3]	W17	input	1.5-V HSTL Class I		8
pdata_9_a[0]	B15	input	1.5-V HSTL Class I		3
pdata_9_a[1]	C15	input	1.5-V HSTL Class I		3
pdata_9_a[2]	D15	input	1.5-V HSTL Class I		3
pdata_9_a[3]	E15	input	1.5-V HSTL Class I		3

pdata_9_b[0]	AB17	input	1.5-V HSTL Class I		8
pdata_9_b[1]	AC17	input	1.5-V HSTL Class I		8
pdata_9_b[2]	AD17	input	1.5-V HSTL Class I		8
pdata_9_b[3]	AE17	input	1.5-V HSTL Class I		8
pll_clk_o	AE13	output	LVDS		12
pll_clk_o(n)	AF13	output	LVDS		12
ptick_0_a	E4	input	1.5-V HSTL Class I		4
ptick_0_b	AE8	input	1.5-V HSTL Class I		7
ptick_1_a	A6	input	1.5-V HSTL Class I		4
ptick_1_b	AC9	input	1.5-V HSTL Class I		7
ptick_10_a	H16	input	1.5-V HSTL Class I		3
ptick_10_b	AC18	input	1.5-V HSTL Class I		8
ptick_11_a	E17	input	1.5-V HSTL Class I		3
ptick_11_b	AF19	input	1.5-V HSTL Class I		8
ptick_12_a	E18	input	1.5-V HSTL Class I		3
ptick_12_b	AF21	input	1.5-V HSTL Class I		8
ptick_13_a	C19	input	1.5-V HSTL Class I		3
ptick_13_b	Y24	input	1.5-V HSTL Class I		1
ptick_14_a	A21	input	1.5-V HSTL Class I		3
ptick_14_b	W26	input	1.5-V HSTL Class I		1
ptick_15_a	B23	input	1.5-V HSTL Class I		3
ptick_15_b	U25	input	1.5-V HSTL Class I		1
ptick_16_a	H25	input	1.5-V HSTL Class I		2
ptick_16_b	R23	input	1.5-V HSTL Class I		1
ptick_17_a	J21	input	1.5-V HSTL Class I		2
ptick_17_b	R21	input	1.5-V HSTL Class I		1
ptick_2_a	B7	input	1.5-V HSTL Class I		4
ptick_2_b	Y10	input	1.5-V HSTL Class I		7
ptick_3_a	H7	input	1.5-V HSTL Class I		4
ptick_3_b	AA11	input	1.5-V HSTL Class I		7
ptick_4_a	E8	input	1.5-V HSTL Class I		4
ptick_4_b	AB13	input	1.5-V HSTL Class I		11
ptick_5_a	D9	input	1.5-V HSTL Class I		4
ptick_5_b	AD15	input	1.5-V HSTL Class I		8

ptick_6_a	G11	input	1.5-V HSTL Class I		4
ptick_6_b	U14	input	1.5-V HSTL Class I		8
ptick_7_a	F12	input	1.5-V HSTL Class I		4
ptick_7_b	AA16	input	1.5-V HSTL Class I		8
ptick_8_a	E14	input	1.5-V HSTL Class I		10
ptick_8_b	Y17	input	1.5-V HSTL Class I		8
ptick_9_a	F15	input	1.5-V HSTL Class I		3
ptick_9_b	AF17	input	1.5-V HSTL Class I		8
sclk	B14	input	LVDS		3
sclk(n)	A14	input	LVDS		3
spps	L23	input	LVDS		2
spps(n)	L24	input	LVDS		2
stick	K23	input	LVDS		2
stick(n)	K24	input	LVDS		2
sys_rst_N	R25	input	3.3-V LVTTL		1
system_pps_o	C12	output	LVDS		9
system_pps_o(n)	B12	output	LVDS		9
system_tick_o	D13	output	LVDS		9
system_tick_o(n)	C13	output	LVDS		9
tc_cbit	C1	output	3.3-V LVTTL		4
test_port[0]	R19	output	3.3-V LVTTL		1
test_port[1]	M16	output	3.3-V LVTTL		3
test_port[2]	R17	output	3.3-V LVTTL		1
test_port[3]	AA9	output	3.3-V LVTTL		7
xclk	Y12	input	LVDS		7
xclk(n)	AA12	input	LVDS		7

Table 8-1 Pinout by Signal Name.

7.2 Pinout by pin number

Note: UNUSED and NC pins have been removed.

```
Quartus II Version 9.1 Build 304 01/25/2010 Service Pack 1 SJ Full Version
CHIP  "timing_top"  ASSIGNED TO AN: EP1SGX25CF672C5

-------------------------------------------------------------------------------------------
Pin Name/Usage               : Location  : Dir.   : I/O Standard       : Voltage : I/O Bank  :
-------------------------------------------------------------------------------------------
GND                          : A2        : gnd    :                    :         :           :
VCCIO4                       : A3        : power  :                    : 3.3V    : 4         :
pdata_0_a[0]                 : A4        : input  : 1.5-V HSTL Class I : 	       : 4         :
pclk_0_a                     : A5        : input  : 1.5-V HSTL Class I :         : 4         :
ptick_1_a                    : A6        : input  : 1.5-V HSTL Class I :         : 4         :
pdata_2_a[3]                 : A7        : input  : 1.5-V HSTL Class I :         : 4         :
pdata_4_a[0]                 : A8        : input  : 1.5-V HSTL Class I :         : 4         :
pdata_5_a[1]                 : A9        : input  : 1.5-V HSTL Class I :         : 4         :
pdata_6_a[0]                 : A10       : input  : 1.5-V HSTL Class I :         : 4         :
GND                          : A11       : gnd    :                    :         :           :
VCCIO4                       : A12       : power  :                    : 3.3V    : 4         :
pdata_8_a[0]                 : A13       : input  : 1.5-V HSTL Class I :         : 10        :
sclk(n)                      : A14       : input  : LVDS               :         : 3         :
VCCIO3                       : A15       : power  :                    : 3.3V    : 3         :
GND                          : A16       : gnd    :                    :         :           :
pdata_11_a[0]                : A17       : input  : 1.5-V HSTL Class I :         : 3         :
pdata_12_a[0]                : A18       : input  : 1.5-V HSTL Class I :         : 3         :
pdata_13_a[2]                : A19       : input  : 1.5-V HSTL Class I :         : 3         :
pdata_14_a[0]                : A20       : input  : 1.5-V HSTL Class I :         : 3         :
ptick_14_a                   : A21       : input  : 1.5-V HSTL Class I :         : 3         :
pdata_15_a[0]                : A22       : input  : 1.5-V HSTL Class I :         : 3         :
pdata_15_a[3]                : A23       : input  : 1.5-V HSTL Class I :         : 3         :
VCCIO3                       : A24       : power  :                    : 3.3V    : 3         :
```

GND	A25	gnd				
GXB_VCC*	AA1			1.5V	15	
GXB_GND*	AA2				15	
GXB_GND	AA3					
GXB_VCC*	AA4			1.5V	15	
GXB_GND*	AA5				15	
GND	AA6	gnd				
mcb_data[6]	AA7	bidir	3.3-V LVTTL		7	
pdata_0_b[0]	AA8	input	1.5-V HSTL Class I		7	
test_port[3]	AA9	output	3.3-V LVTTL		7	
RESERVED_INPUT	AA10				7	
ptick_3_b	AA11	input	1.5-V HSTL Class I		7	
xclk(n)	AA12	input	LVDS		7	
pdata_4_b[3]	AA13	input	1.5-V HSTL Class I		11	
pdata_5_b[1]	AA14	input	1.5-V HSTL Class I		12	
RESERVED_INPUT	AA15				8	
ptick_7_b	AA16	input	1.5-V HSTL Class I		8	
pclk_8_b	AA17	input	1.5-V HSTL Class I		8	
pdata_10_b[2]	AA18	input	1.5-V HSTL Class I		8	
pc_clk_a_o	AA19	output	LVDS		1	
pc_clk_a_o(n)	AA20	output	LVDS		1	
RESERVED_INPUT	AA21				1	
RESERVED_INPUT	AA22				1	
GND+	AA23				1	
GND+	AA24				1	
GND+	AA25				1	
GND+	AA26				1	
GXB_GND	AB1					
GXB_GND	AB2					
GXB_GND	AB3					
GXB_GND	AB4					
GXB_GND	AB5					
VCCINT	AB6	power		1.5V		
mcb_data[7]	AB7	bidir	3.3-V LVTTL		7	

pdata_0_b[1]	AB8	input	1.5-V HSTL Class I		7
pdata_1_b[3]	AB9	input	1.5-V HSTL Class I		7
RESERVED_INPUT	AB10				7
RESERVED_INPUT	AB11				7
mcb_clk	AB12	input	3.3-V LVTTL		7
ptick_4_b	AB13	input	1.5-V HSTL Class I		11
pdata_5_b[2]	AB14	input	1.5-V HSTL Class I		12
RESERVED_INPUT	AB15				8
pclk_9_b	AB16	input	1.5-V HSTL Class I		8
pdata_9_b[0]	AB17	input	1.5-V HSTL Class I		8
pdata_10_b[3]	AB18	input	1.5-V HSTL Class I		8
pc_clk_b_o	AB19	output	LVDS		1
pc_clk_b_o(n)	AB20	output	LVDS		1
RESERVED_INPUT	AB21				1
RESERVED_INPUT	AB22				1
GND+	AB23				1
GND+	AB24				1
GND+	AB25				1
GND+	AB26				1
GXB_VCC*	AC1			1.5V	15
GXB_GND*	AC2				15
GND	AC3	gnd			
mcb_rd_wr_l	AC4	input	3.3-V LVTTL		7
mcb_addr[3]	AC5	input	3.3-V LVTTL		7
mcb_addr[7]	AC6	input	3.3-V LVTTL		7
mcb_data[8]	AC7	bidir	3.3-V LVTTL		7
pdata_0_b[2]	AC8	input	1.5-V HSTL Class I		7
ptick_1_b	AC9	input	1.5-V HSTL Class I		7
RESERVED_INPUT	AC10				7
RESERVED_INPUT	AC11				7
pdata_4_b[0]	AC12	input	1.5-V HSTL Class I		7
pclk_4_b	AC13	input	1.5-V HSTL Class I		11
pc_clk_b_i	AC14	input	LVDS		8
pdata_5_b[3]	AC15	input	1.5-V HSTL Class I		8

pdata_8_b[0]	AC16	input	1.5-V HSTL Class I	8	
pdata_9_b[1]	AC17	input	1.5-V HSTL Class I	8	
ptick_10_b	AC18	input	1.5-V HSTL Class I	8	
pc_dat_a_o	AC19	output	LVDS	1	
pc_dat_a_o(n)	AC20	output	LVDS	1	
pc_dat_b_o	AC21	output	LVDS	1	
pc_dat_b_o(n)	AC22	output	LVDS	1	
GND+	AC23			1	
GND+	AC24			1	
GND+	AC25			1	
GND+	AC26			1	
GXB_GND	AD1				
GXB_GND	AD2				
GND	AD3	gnd			
mcb_addr[0]	AD4	input	3.3-V LVTTL	7	
mcb_addr[4]	AD5	input	3.3-V LVTTL	7	
mcb_data[0]	AD6	bidir	3.3-V LVTTL	7	
mcb_data[9]	AD7	bidir	3.3-V LVTTL	7	
pdata_0_b[3]	AD8	input	1.5-V HSTL Class I	7	
pclk_1_b	AD9	input	1.5-V HSTL Class I	7	
RESERVED_INPUT	AD10			7	
RESERVED_INPUT	AD11			7	
pdata_4_b[1]	AD12	input	1.5-V HSTL Class I	11	
pdata_5_b[0]	AD13	input	1.5-V HSTL Class I	11	
pc_clk_b_i(n)	AD14	input	LVDS	8	
ptick_5_b	AD15	input	1.5-V HSTL Class I	8	
pdata_8_b[1]	AD16	input	1.5-V HSTL Class I	8	
pdata_9_b[2]	AD17	input	1.5-V HSTL Class I	8	
pclk_10_b	AD18	input	1.5-V HSTL Class I	8	
pdata_11_b[2]	AD19	input	1.5-V HSTL Class I	8	
pclk_11_b	AD20	input	1.5-V HSTL Class I	8	
pdata_12_b[2]	AD21	input	1.5-V HSTL Class I	8	
pdata_13_b[0]	AD22	input	1.5-V HSTL Class I	8	
GND+	AD23			1	

GND+	AD24				1	
GND+	AD25				1	
VCCIO1	AD26	power		3.3V	1	
GND	AE1	gnd				
mcb_int	AE2	output	3.3-V LVTTL		7	
mcb_cs_l	AE3	input	3.3-V LVTTL		7	
mcb_addr[1]	AE4	input	3.3-V LVTTL		7	
mcb_addr[5]	AE5	input	3.3-V LVTTL		7	
mcb_data[1]	AE6	bidir	3.3-V LVTTL		7	
mcb_data[10]	AE7	bidir	3.3-V LVTTL		7	
ptick_0_b	AE8	input	1.5-V HSTL Class I		7	
pdata_2_b[0]	AE9	input	1.5-V HSTL Class I		7	
pclk_2_b	AE10	input	1.5-V HSTL Class I		7	
pclk_3_b	AE11	input	1.5-V HSTL Class I		7	
pdata_4_b[2]	AE12	input	1.5-V HSTL Class I		11	
pll_clk_o	AE13	output	LVDS		12	
pc_clk_a_i	AE14	input	LVDS		8	
pclk_5_b	AE15	input	1.5-V HSTL Class I		8	
pdata_8_b[2]	AE16	input	1.5-V HSTL Class I		8	
pdata_9_b[3]	AE17	input	1.5-V HSTL Class I		8	
pdata_11_b[0]	AE18	input	1.5-V HSTL Class I		8	
pdata_11_b[3]	AE19	input	1.5-V HSTL Class I		8	
pdata_12_b[0]	AE20	input	1.5-V HSTL Class I		8	
pdata_12_b[3]	AE21	input	1.5-V HSTL Class I		8	
pdata_13_b[1]	AE22	input	1.5-V HSTL Class I		8	
GND+	AE23				1	
GND+	AE24				1	
GND+	AE25				1	
GND	AE26	gnd				
GND	AF2	gnd				
VCCIO7	AF3	power		3.3V	7	
mcb_addr[2]	AF4	input	3.3-V LVTTL		7	
mcb_addr[6]	AF5	input	3.3-V LVTTL		7	
mcb_data[2]	AF6	bidir	3.3-V LVTTL		7	

mcb_data[11]	AF7	bidir	3.3-V LVTTL		7
pclk_0_b	AF8	input	1.5-V HSTL Class I		7
pdata_2_b[1]	AF9	input	1.5-V HSTL Class I		7
pdata_3_b[0]	AF10	input	1.5-V HSTL Class I		7
GND	AF11	gnd			
VCCIO7	AF12	power		3.3V	7
pll_clk_o(n)	AF13	output	LVDS		12
pc_clk_a_i(n)	AF14	input	LVDS		8
VCCIO8	AF15	power		1.5V	8
GND	AF16	gnd			
ptick_9_b	AF17	input	1.5-V HSTL Class I		8
pdata_11_b[1]	AF18	input	1.5-V HSTL Class I		8
ptick_11_b	AF19	input	1.5-V HSTL Class I		8
pdata_12_b[1]	AF20	input	1.5-V HSTL Class I		8
ptick_12_b	AF21	input	1.5-V HSTL Class I		8
pclk_12_b	AF22	input	1.5-V HSTL Class I		8
pdata_13_b[2]	AF23	input	1.5-V HSTL Class I		8
VCCIO8	AF24	power		1.5V	8
GND	AF25	gnd			
GND	B1	gnd			
RESERVED_INPUT	B2				4
RESERVED_INPUT	B3				4
pdata_0_a[1]	B4	input	1.5-V HSTL Class I		4
pdata_1_a[0]	B5	input	1.5-V HSTL Class I		4
pclk_1_a	B6	input	1.5-V HSTL Class I		4
ptick_2_a	B7	input	1.5-V HSTL Class I		4
pdata_4_a[1]	B8	input	1.5-V HSTL Class I		4
pdata_5_a[2]	B9	input	1.5-V HSTL Class I		4
RESERVED_INPUT	B10				4
RESERVED_INPUT	B11				4
system_pps_o(n)	B12	output	LVDS		9
pdata_8_a[1]	B13	input	1.5-V HSTL Class I		10
sclk	B14	input	LVDS		3
pdata_9_a[0]	B15	input	1.5-V HSTL Class I		3

pclk_9_a	B16	input	1.5-V HSTL Class I		3
pdata_11_a[1]	B17	input	1.5-V HSTL Class I		3
pdata_12_a[1]	B18	input	1.5-V HSTL Class I		3
pdata_13_a[3]	B19	input	1.5-V HSTL Class I		3
pdata_14_a[1]	B20	input	1.5-V HSTL Class I		3
pclk_14_a	B21	input	1.5-V HSTL Class I		3
pdata_15_a[1]	B22	input	1.5-V HSTL Class I		3
ptick_15_a	B23	input	1.5-V HSTL Class I		3
pclk_15_a	B24	input	1.5-V HSTL Class I		3
GND+	B25				2
GND	B26	gnd			
tc_cbit	C1	output	3.3-V LVTTL		4
RESERVED_INPUT	C2				4
RESERVED_INPUT	C3				4
pdata_0_a[2]	C4	input	1.5-V HSTL Class I		4
pdata_1_a[1]	C5	input	1.5-V HSTL Class I		4
pdata_2_a[0]	C6	input	1.5-V HSTL Class I		4
pclk_2_a	C7	input	1.5-V HSTL Class I		4
pdata_4_a[2]	C8	input	1.5-V HSTL Class I		4
pdata_5_a[3]	C9	input	1.5-V HSTL Class I		4
RESERVED_INPUT	C10				4
RESERVED_INPUT	C11				4
system_pps_o	C12	output	LVDS		9
system_tick_o(n)	C13	output	LVDS		9
pdata_8_a[2]	C14	input	1.5-V HSTL Class I		3
pdata_9_a[1]	C15	input	1.5-V HSTL Class I		3
pdata_10_a[0]	C16	input	1.5-V HSTL Class I		3
pdata_11_a[2]	C17	input	1.5-V HSTL Class I		3
pdata_12_a[2]	C18	input	1.5-V HSTL Class I		3
ptick_13_a	C19	input	1.5-V HSTL Class I		3
pdata_14_a[2]	C20	input	1.5-V HSTL Class I		3
RESERVED_INPUT	C21				3
pdata_15_a[2]	C22	input	1.5-V HSTL Class I		3
GND+	C23				2

GND+	C24				2
GND+	C25				2
VCCIO2	C26	power		3.3V	2
GXB_GND	D1				
GXB_GND	D2				
GND	D3	gnd			
pdata_0_a[3]	D4	input	1.5-V HSTL Class I		4
pdata_1_a[2]	D5	input	1.5-V HSTL Class I		4
pdata_2_a[1]	D6	input	1.5-V HSTL Class I		4
pdata_3_a[0]	D7	input	1.5-V HSTL Class I		4
pdata_4_a[3]	D8	input	1.5-V HSTL Class I		4
ptick_5_a	D9	input	1.5-V HSTL Class I		4
RESERVED_INPUT	D10				4
RESERVED_INPUT	D11				4
pdata_7_a[2]	D12	input	1.5-V HSTL Class I		4
system_tick_o	D13	output	LVDS		9
pdata_8_a[3]	D14	input	1.5-V HSTL Class I		3
pdata_9_a[2]	D15	input	1.5-V HSTL Class I		3
pdata_10_a[1]	D16	input	1.5-V HSTL Class I		3
pdata_11_a[3]	D17	input	1.5-V HSTL Class I		3
pdata_12_a[3]	D18	input	1.5-V HSTL Class I		3
pclk_13_a	D19	input	1.5-V HSTL Class I		3
pdata_14_a[3]	D20	input	1.5-V HSTL Class I		3
RESERVED_INPUT	D21				2
RESERVED_INPUT	D22				2
GND+	D23				2
GND+	D24				2
GND+	D25				2
GND+	D26				2
GND*	E1				
GND*	E2				
GXB_GND	E3				
ptick_0_a	E4	input	1.5-V HSTL Class I		4
pdata_1_a[3]	E5	input	1.5-V HSTL Class I		4

pdata_2_a[2]	E6	input	1.5-V HSTL Class I		4
pdata_3_a[1]	E7	input	1.5-V HSTL Class I		4
ptick_4_a	E8	input	1.5-V HSTL Class I		4
RESERVED_INPUT	E9				4
pdata_6_a[1]	E10	input	1.5-V HSTL Class I		4
pdata_6_a[3]	E11	input	1.5-V HSTL Class I		4
pdata_7_a[3]	E12	input	1.5-V HSTL Class I		4
pdata_7_a[0]	E13	input	1.5-V HSTL Class I		9
ptick_8_a	E14	input	1.5-V HSTL Class I		10
pdata_9_a[3]	E15	input	1.5-V HSTL Class I		3
pdata_10_a[2]	E16	input	1.5-V HSTL Class I		3
ptick_11_a	E17	input	1.5-V HSTL Class I		3
ptick_12_a	E18	input	1.5-V HSTL Class I		3
out_tick_a	E19	input	3.3-V LVTTL		2
RESERVED_INPUT	E20				2
RESERVED_INPUT	E21				2
RESERVED_INPUT	E22				2
GND+	E23				2
GND+	E24				2
GND+	E25				2
GND+	E26				2
GXB_GND	F1				
GXB_GND	F2				
GXB_GND	F3				
GXB_GND	F4				
GXB_GND	F5				
VCCINT	F6	power		1.5V	
pdata_3_a[2]	F7	input	1.5-V HSTL Class I		4
RESERVED_INPUT	F8				4
TMS	F9	input			4
TDI	F10	input			4
TDO	F11	output			4
ptick_7_a	F12	input	1.5-V HSTL Class I		4
pdata_7_a[1]	F13	input	1.5-V HSTL Class I		9

pclk_8_a	F14	input	1.5-V HSTL Class I		10
ptick_9_a	F15	input	1.5-V HSTL Class I		3
RESERVED_INPUT	F16				3
pclk_11_a	F17	input	1.5-V HSTL Class I		3
pclk_12_a	F18	input	1.5-V HSTL Class I		3
RESERVED_INPUT	F19				2
RESERVED_INPUT	F20				2
RESERVED_INPUT	F21				2
RESERVED_INPUT	F22				2
GND+	F23				2
GND+	F24				2
pdata_16_a[0]	F25	input	1.5-V HSTL Class I		2
pdata_16_a[1]	F26	input	1.5-V HSTL Class I		2
GND*	G1				
GND*	G2				
GXB_GND	G3				
GND*	G4				
GND*	G5				
GND	G6	gnd			
pdata_3_a[3]	G7	input	1.5-V HSTL Class I		4
pclk_4_a	G8	input	1.5-V HSTL Class I		4
pclk_5_a	G9	input	1.5-V HSTL Class I		4
pdata_6_a[2]	G10	input	1.5-V HSTL Class I		4
ptick_6_a	G11	input	1.5-V HSTL Class I		4
pclk_7_a	G12	input	1.5-V HSTL Class I		4
GNDA_PLL5	G13	gnd			
RESERVED_INPUT	G14				3
RESERVED_INPUT	G15				3
pdata_10_a[3]	G16	input	1.5-V HSTL Class I		3
RESERVED_INPUT	G17				3
pdata_13_a[0]	G18	input	1.5-V HSTL Class I		3
RESERVED_INPUT	G19				2
RESERVED_INPUT	G20				2
RESERVED_INPUT	G21				2

RESERVED_INPUT	G22				2
GND+	G23				2
GND+	G24				2
pdata_16_a[2]	G25	input	1.5-V HSTL Class I		2
pdata_16_a[3]	G26	input	1.5-V HSTL Class I		2
GXB_GND	H1				
GXB_GND	H2				
GXB_GND	H3				
GXB_GND	H4				
GXB_GND	H5				
VCCINT	H6	power		1.5V	
ptick_3_a	H7	input	1.5-V HSTL Class I		4
pdata_5_a[0]	H8	input	1.5-V HSTL Class I		4
TRST	H9	input			4
RESERVED_INPUT	H10				4
pclk_6_a	H11	input	1.5-V HSTL Class I		4
VCCG_PLL5	H12	power		1.5V	
VCCA_PLL5	H13	power		1.5V	
RESERVED_INPUT	H14				3
RESERVED_INPUT	H15				3
ptick_10_a	H16	input	1.5-V HSTL Class I		3
GND	H17				
pdata_13_a[1]	H18	input	1.5-V HSTL Class I		3
control_out[0]	H19	output	LVDS		2
control_out[0](n)	H20	output	LVDS		2
RESERVED_INPUT	H21				2
RESERVED_INPUT	H22				2
pdata_17_a[0]	H23	input	1.5-V HSTL Class I		2
pdata_17_a[1]	H24	input	1.5-V HSTL Class I		2
ptick_16_a	H25	input	1.5-V HSTL Class I		2
pclk_16_a	H26	input	1.5-V HSTL Class I		2
GND*	J1				
GND*	J2				
GXB_GND	J3				

GND*	J4					
GND*	J5					
GND	J6	gnd				
pclk_3_a	J7	input	1.5-V HSTL Class I		4	
TCK	J8	input			4	
RESERVED_INPUT	J9				4	
TEMPDIODEn	J10					
TEMPDIODEp	J11					
GNDG_PLL5	J12	gnd				
VCC_PLL5_OUTA	J13	power		3.3V	9	
RESERVED_INPUT	J14				3	
RESERVED_INPUT	J15				3	
VREF0B3	J16			0.75V	3	
VREF1B3	J17			0.75V	3	
VREF2B3	J18			0.75V	3	
control_out[2]	J19	output	LVDS		2	
control_out[2](n)	J20	output	LVDS		2	
ptick_17_a	J21	input	1.5-V HSTL Class I		2	
pclk_17_a	J22	input	1.5-V HSTL Class I		2	
pdata_17_a[2]	J23	input	1.5-V HSTL Class I		2	
pdata_17_a[3]	J24	input	1.5-V HSTL Class I		2	
pc_dat_b_i	J25	input	LVDS		2	
pc_dat_b_i(n)	J26	input	LVDS		2	
GXB_GND	K1					
GXB_GND	K2					
GXB_GND	K3					
GXB_GND	K4					
GXB_GND	K5					
VCCINT	K6	power		1.5V		
RREFB15A	K7					
RESERVED_INPUT	K8				4	
VREF0B4	K9			0.75V	4	
VREF1B4	K10			0.75V	4	
VREF2B4	K11			0.75V	4	

VCC_PLL5_OUTB	K12	power		1.5V	10	
DCLK	K13				3	
nSTATUS	K14				3	
CONF_DONE	K15				3	
RESERVED_INPUT	K16				3	
control_out[1]	K17	output	LVDS		2	
control_out[1](n)	K18	output	LVDS		2	
control_out[3]	K19	output	LVDS		2	
control_out[3](n)	K20	output	LVDS		2	
control_out[4]	K21	output	LVDS		2	
control_out[4](n)	K22	output	LVDS		2	
stick	K23	input	LVDS		2	
stick(n)	K24	input	LVDS		2	
GND+	K25				2	
GND+	K26				2	
GND*	L1					
GND*	L2					
GXB_GND	L3					
GND*	L4					
GND*	L5					
VCCT_B15	L6	power		1.5V	15	
VCCR_B15	L7	power		1.5V	15	
VCCP_B15	L8	power		1.5V	15	
VCCINT	L9	power		1.5V		
VCCINT	L10	power		1.5V		
VCCINT	L11	power		1.5V		
VCCIO4	L12	power		3.3V	4	
VCCIO3	L13	power		3.3V	3	
nCONFIG	L14				3	
RESERVED_INPUT	L15				3	
pclk_10_a	L16	input	1.5-V HSTL Class I		3	
VREF0B2	L17			0.75V	2	
VREF1B2	L18			0.75V	2	
control_out[5]	L19	output	LVDS		2	

control_out[5](n)	L20	output	LVDS			2	
control_out[6]	L21	output	LVDS			2	
control_out[6](n)	L22	output	LVDS			2	
spps	L23	input	LVDS			2	
spps(n)	L24	input	LVDS			2	
GND+	L25					2	
GND	L26	gnd					
GXB_GND	M1						
GXB_GND	M2						
GXB_GND	M3						
GXB_GND	M4						
GXB_GND	M5						
VCCT_B15	M6	power			1.5V	15	
VCCR_B15	M7	power			1.5V	15	
VCCP_B15	M8	power			1.5V	15	
VCCINT	M9	power			1.5V		
VCCINT	M10	power			1.5V		
GND	M11	gnd					
GND	M12	gnd					
GND	M13	gnd					
VCCINT	M14	power			1.5V		
RESERVED_INPUT	M15					3	
test_port[1]	M16	output	3.3-V LVTTL			3	
control_out[7]	M17	output	LVDS			2	
control_out[7](n)	M18	output	LVDS			2	
control_out[8]	M19	output	LVDS			2	
control_out[8](n)	M20	output	LVDS			2	
control_out[9]	M21	output	LVDS			2	
control_out[9](n)	M22	output	LVDS			2	
GND+	M23					2	
GND+	M24					2	
GND+	M25					2	
VCCIO2	M26	power			3.3V	2	
GND*	N1						

GND*	N2					
GXB_GND	N3					
GND*	N4					
GND*	N5					
GND	N6	gnd				
VCCA_B15	N7	power		3.3V	15	
VCCG_B15	N8	power		1.5V	15	
VCCINT	N9	power		1.5V		
GND	N10	gnd				
VCCINT	N11	power		1.5V		
GND	N12	gnd				
VCCINT	N13	power		1.5V		
GND	N14	gnd				
VCCINT	N15	power		1.5V		
VCCINT	N16	power		1.5V		
GND	N17	gnd				
VCCIO2	N18	power		3.3V	2	
GNDG_PLL1	N19	gnd				
VCCA_PLL1	N20	power		1.5V		
VCCG_PLL1	N21	power		1.5V		
GNDA_PLL1	N22	gnd				
pc_dat_a_i	N23	input	LVDS		2	
pc_dat_a_i(n)	N24	input	LVDS		2	
GND+	N25				2	
GND+	N26				2	
GXB_GND	P1					
GXB_GND	P2					
GXB_GND	P3					
GXB_GND	P4					
GXB_GND	P5					
GND	P6	gnd				
VCCG_B15	P7	power		1.5V	15	
VCCA_B15	P8	power		3.3V	15	
nCE	P9				7	

VCCINT	P10	power		1.5V		
GND	P11	gnd				
VCCINT	P12	power		1.5V		
GND	P13	gnd				
VCCINT	P14	power		1.5V		
GND	P15	gnd				
VCCINT	P16	power		1.5V		
GND	P17	gnd				
VCCIO1	P18	power		3.3V	1	
GNDG_PLL2	P19	gnd				
VCCG_PLL2	P20	power		1.5V		
VCCA_PLL2	P21	power		1.5V		
GNDA_PLL2	P22	gnd				
pdata_17_b[0]	P23	input	1.5-V HSTL Class I		1	
pdata_17_b[1]	P24	input	1.5-V HSTL Class I		1	
pdata_17_b[2]	P25	input	1.5-V HSTL Class I		1	
GND+	P26				1	
GXB_VCC*	R1			1.5V	15	
GXB_GND*	R2				15	
GXB_GND	R3					
GXB_VCC*	R4			1.5V	15	
GXB_GND*	R5				15	
VCCT_B15	R6	power		1.5V	15	
VCCR_B15	R7	power		1.5V	15	
VCCP_B15	R8	power		1.5V	15	
VCCSEL	R9				7	
pdata_2_b[2]	R10	input	1.5-V HSTL Class I		7	
VCC_PLL6_OUTB	R11	power		3.3V	12	
GND	R12	gnd				
pdata_6_b[0]	R13	input	1.5-V HSTL Class I		8	
GND	R14	gnd				
pdata_6_b[1]	R15	input	1.5-V HSTL Class I		8	
VREF0B1	R16			0.75V	1	
test_port[2]	R17	output	3.3-V LVTTL		1	

pdata_17_b[3]	R18	input	1.5-V HSTL Class I		1	
test_port[0]	R19	output	3.3-V LVTTL		1	
RESERVED_INPUT	R20				1	
ptick_17_b	R21	input	1.5-V HSTL Class I		1	
pclk_17_b	R22	input	1.5-V HSTL Class I		1	
ptick_16_b	R23	input	1.5-V HSTL Class I		1	
pclk_16_b	R24	input	1.5-V HSTL Class I		1	
sys_rst_N	R25	input	3.3-V LVTTL		1	
VCCIO1	R26	power		3.3V	1	
GXB_GND	T1					
GXB_GND	T2					
GXB_GND	T3					
GXB_GND	T4					
GXB_GND	T5					
VCCT_B15	T6	power		1.5V	15	
VCCR_B15	T7	power		1.5V	15	
VCCP_B15	T8	power		1.5V	15	
nCEO	T9				7	
nIO_PULLUP	T10				7	
VCCA_PLL6	T11	power		1.5V		
VCCIO7	T12	power		3.3V	7	
MSEL1	T13				8	
VCCIO8	T14	power		1.5V	8	
pdata_6_b[2]	T15	input	1.5-V HSTL Class I		8	
pclk_6_b	T16	input	1.5-V HSTL Class I		8	
VREF2B1	T17			0.75V	1	
VREF1B1	T18			0.75V	1	
control_out[10]	T19	output	LVDS		1	
control_out[10](n)	T20	output	LVDS		1	
control_out[11]	T21	output	LVDS		1	
control_out[11](n)	T22	output	LVDS		1	
pdata_16_b[1]	T23	input	1.5-V HSTL Class I		1	
pdata_16_b[2]	T24	input	1.5-V HSTL Class I		1	
pdata_16_b[3]	T25	input	1.5-V HSTL Class I		1	

GND	T26	gnd				
GXB_VCC*	U1			1.5V	15	
GXB_GND*	U2				15	
GXB_GND	U3					
GXB_VCC*	U4			1.5V	15	
GXB_GND*	U5				15	
GND	U6	gnd				
RREFB15	U7					
mcb_data[12]	U8	bidir	3.3-V LVTTL		7	
pdata_1_b[0]	U9	input	1.5-V HSTL Class I		7	
PORSEL	U10				7	
pdata_3_b[1]	U11	input	1.5-V HSTL Class I		7	
GNDG_PLL6	U12	gnd				
MSEL0	U13				8	
ptick_6_b	U14	input	1.5-V HSTL Class I		8	
pdata_6_b[3]	U15	input	1.5-V HSTL Class I		8	
pdata_7_b[0]	U16	input	1.5-V HSTL Class I		8	
control_out[12]	U17	output	LVDS		1	
control_out[12](n)	U18	output	LVDS		1	
control_out[13]	U19	output	LVDS		1	
control_out[13](n)	U20	output	LVDS		1	
control_out[14]	U21	output	LVDS		1	
control_out[14](n)	U22	output	LVDS		1	
pdata_16_b[0]	U23	input	1.5-V HSTL Class I		1	
pdata_15_b[3]	U24	input	1.5-V HSTL Class I		1	
ptick_15_b	U25	input	1.5-V HSTL Class I		1	
pclk_15_b	U26	input	1.5-V HSTL Class I		1	
GXB_GND	V1					
GXB_GND	V2					
GXB_GND	V3					
GXB_GND	V4					
GXB_GND	V5					
VCCINT	V6	power		1.5V		
mcb_data[3]	V7	bidir	3.3-V LVTTL		7	

mcb_data[13]	V8	bidir	3.3-V LVTTL		7
VREF2B7	V9			0.75V	7
VREF1B7	V10			0.75V	7
VREF0B7	V11			0.75V	7
VCCG_PLL6	V12	power		1.5V	
VCC_PLL6_OUTA	V13	power		1.5V	11
MSEL2	V14				8
VREF2B8	V15			0.75V	8
pdata_7_b[1]	V16	input	1.5-V HSTL Class I		8
VREF1B8	V17			0.75V	8
VREF0B8	V18			0.75V	8
control_out[16]	V19	output	LVDS		1
control_out[16](n)	V20	output	LVDS		1
control_out[15]	V21	output	LVDS		1
control_out[15](n)	V22	output	LVDS		1
pdata_15_b[0]	V23	input	1.5-V HSTL Class I		1
pdata_15_b[1]	V24	input	1.5-V HSTL Class I		1
pdata_15_b[2]	V25	input	1.5-V HSTL Class I		1
pclk_14_b	V26	input	1.5-V HSTL Class I		1
GXB_VCC*	W1			1.5V	15
GXB_GND*	W2				15
GXB_GND	W3				
GXB_VCC*	W4			1.5V	15
GXB_GND*	W5				15
GND	W6	gnd			
mcb_data[4]	W7	bidir	3.3-V LVTTL		7
mcb_data[14]	W8	bidir	3.3-V LVTTL		7
pdata_1_b[1]	W9	input	1.5-V HSTL Class I		7
pdata_2_b[3]	W10	input	1.5-V HSTL Class I		7
pdata_3_b[2]	W11	input	1.5-V HSTL Class I		7
GNDA_PLL6	W12	gnd			
RESERVED_INPUT	W13				8
RESERVED_INPUT	W14				8
PLL_ENA	W15				8

pdata_7_b[2]	W16	input	1.5-V HSTL Class I		8	
pdata_8_b[3]	W17	input	1.5-V HSTL Class I		8	
pdata_10_b[0]	W18	input	1.5-V HSTL Class I		8	
control_out[17]	W19	output	LVDS		1	
control_out[17](n)	W20	output	LVDS		1	
RESERVED_INPUT	W21				1	
RESERVED_INPUT	W22				1	
pdata_14_b[1]	W23	input	1.5-V HSTL Class I		1	
pdata_14_b[2]	W24	input	1.5-V HSTL Class I		1	
pdata_14_b[3]	W25	input	1.5-V HSTL Class I		1	
ptick_14_b	W26	input	1.5-V HSTL Class I		1	
GXB_GND	Y1					
GXB_GND	Y2					
GXB_GND	Y3					
GXB_GND	Y4					
GXB_GND	Y5					
VCCINT	Y6	power		1.5V		
mcb_data[5]	Y7	bidir	3.3-V LVTTL		7	
mcb_data[15]	Y8	bidir	3.3-V LVTTL		7	
pdata_1_b[2]	Y9	input	1.5-V HSTL Class I		7	
ptick_2_b	Y10	input	1.5-V HSTL Class I		7	
pdata_3_b[3]	Y11	input	1.5-V HSTL Class I		7	
xclk	Y12	input	LVDS		7	
RESERVED_INPUT	Y13				8	
RESERVED_INPUT	Y14				8	
pclk_7_b	Y15	input	1.5-V HSTL Class I		8	
pdata_7_b[3]	Y16	input	1.5-V HSTL Class I		8	
ptick_8_b	Y17	input	1.5-V HSTL Class I		8	
pdata_10_b[1]	Y18	input	1.5-V HSTL Class I		8	
RESERVED_INPUT	Y19				1	
RESERVED_INPUT	Y20				1	
out_tick_b	Y21	input	3.3-V LVTTL		1	
RESERVED_INPUT	Y22				1	
pdata_13_b[3]	Y23	input	1.5-V HSTL Class I		1	

ptick_13_b	Y24	input	1.5-V HSTL Class I		1
pclk_13_b	Y25	input	1.5-V HSTL Class I		1
pdata_14_b[0]	Y26	input	1.5-V HSTL Class I		1

Table 8-2 Pinout by Pin Number.

7.3 Altera Stratix GX – EP1SGX25CF672C5 Pin Locations

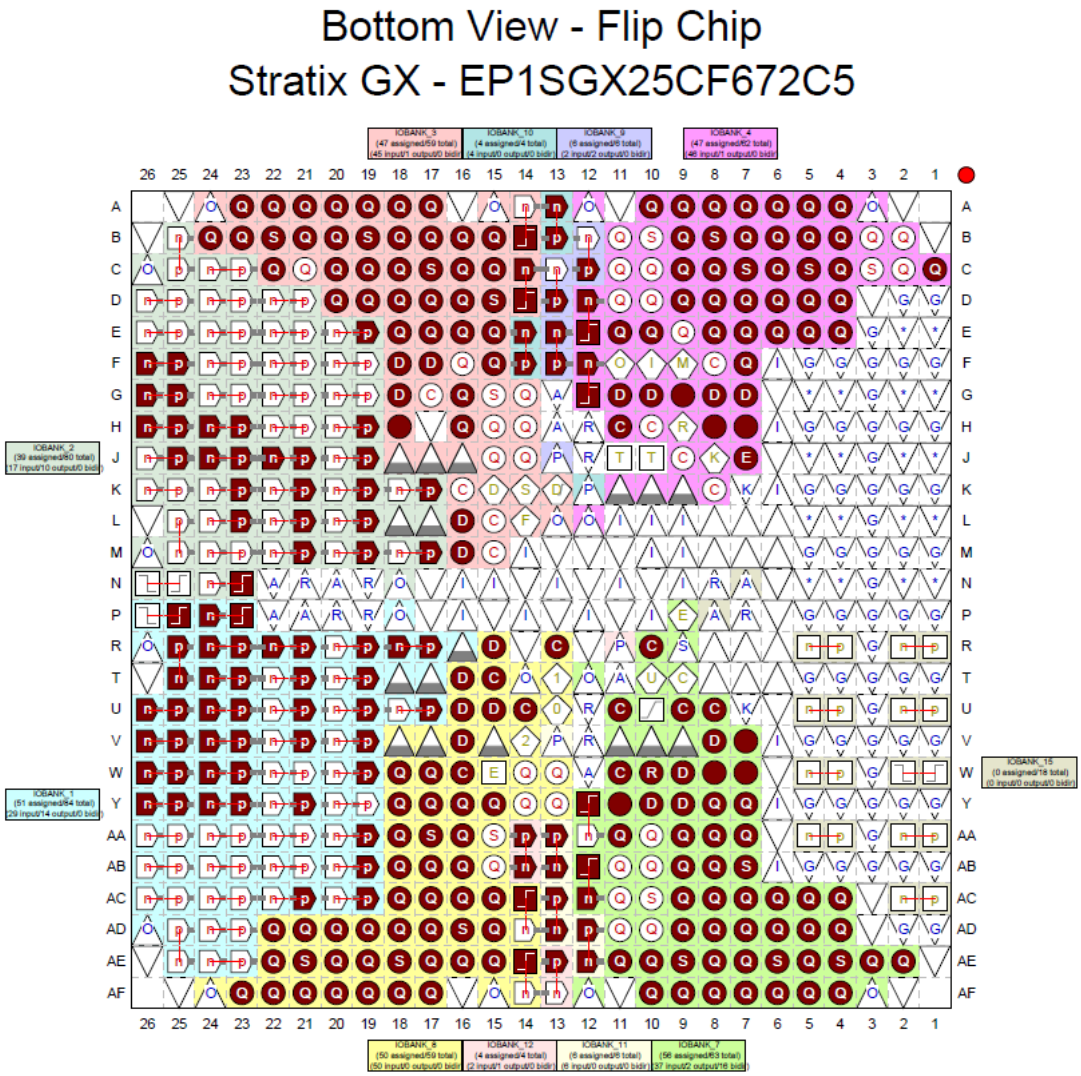


Figure 8-2 Pin Locations.

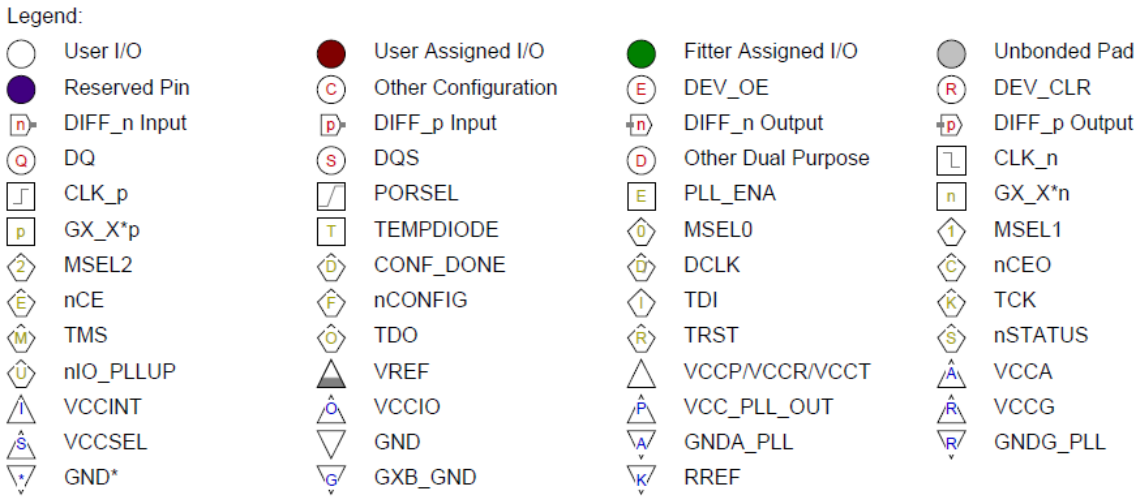


Figure 8-3 Pin Locations Legend.

7.4 Programming Notes

All FPGAs on the Station Board are programmed through their 8-bit wide configuration port. The Station Board CMIB software requires the Raw Binary File (.rbf) output file to program the Altera FPGAs. This is set by choosing “Device” from the “Assignments” menu and pushing the “Device and Pin Options” button in the Altera Quartus II software. Select the “Programming Files” tab and check “Raw Binary File (.rbf)”.

8 References

[1] Brent Carlson, “Refined EVLA WIDAR Correlator Architecture”, NRC-EVLA Memo# 014, October 2, 2001.

[2] Brent Carlson, “HM Gbps Cable Signaling Specification”, PROTOCOL Document A25022N0041, August 18, 2005.

[3] Zhang Heng, “TIMECODE and Clock External Interface Specification”, ICD Document A25022N0090, Rev 1.0, February 22, 2005.

[3] Dave Fort, “Requirements and Functional Specification: Filter FPGA”, RFS Document A25044N0000, Rev 0.5 Draft, November 1, 2003.

9 Index

L

M

N

P

S

T

V

W