The Magic Bus Transceiver Card

Technical Design Report

Version 6 - January 1, 2000

Drew Baden, Rob Bard, John Giganti, David Toback

Abstract

The MBT Card plays a central role in both inter- and intra-crate communication in the DØ Level 2 System. This document serves as a Technical Design Report, as well as a specifications reference, for the MBT. Many thanks must go to Jim Linnemann for exquisitely useful feedback on this subject.

Contents

List of Tables

List of Figures

1 Introduction

The DØ trigger system is shown schematically in Figure 1. Trigger decisions at Level 1 use data (trigger, or more specifically Level 1, data) which is provided by the hardware for just this purpose. Such data is not as precise as the final raw data, but is provided in a very short time, allowing queries to take place per crossing time (132ns). Decisions at this level are based upon what is seen in the various detectors (calorimeter, tracker, muon, and preshower), but for the most part without detector cross-correlation. Level 1 Accepts (which occur at 10 kHz) cause trigger data to be sent to level 2, which then makes physics requirements using all detector elements (with the addition of the silicon detector). Level 2 Accepts (which cannot occur much faster than 1 kHz) cause a data acquisition readout cycle into Level 3, where a processor farm uses reconstructed raw data. Level 3 output rates are expected to be in the range of 10-50Hz.

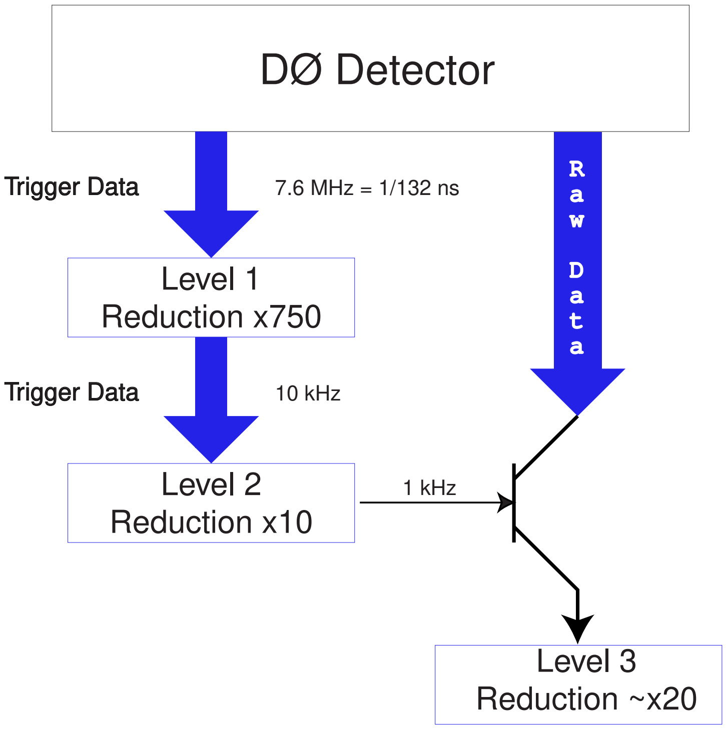

Figure 1: Schematic of the DØ Run II Trigger system.

Level 2 is built around the concept of preprocessor crates as outlined in figure 2. Working in parallel, each preprocessor feeds data into the Global Processor crate for Level 2 decision making. Each type of crate (both preprocessor and global) will consist of assorted VME cards for I/O and communication, with processing done on VME boards which consist of a Dec/ALPHA 21164 chip, running at 500MHz or greater (see figure 3). Note that in this figure, the term “Trigger Bus” is the known

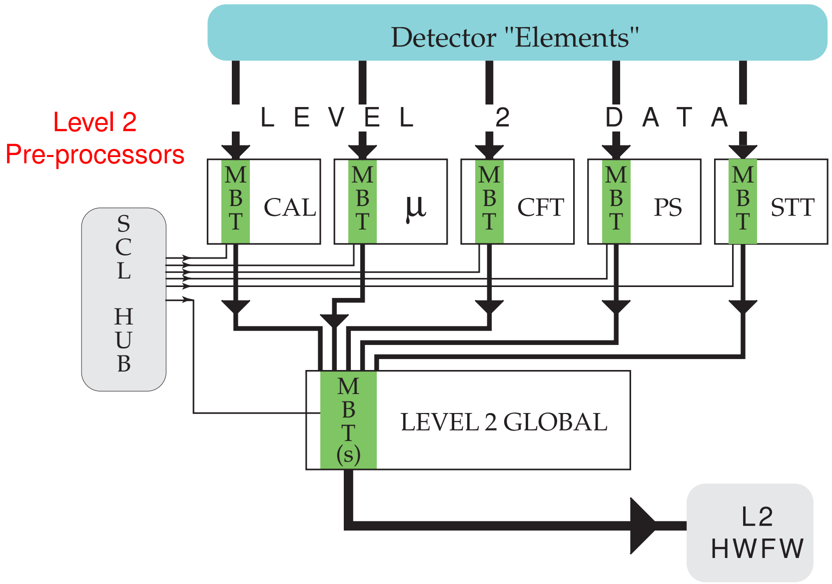

Figure 2: Schematic of the Level 2 trigger system.

as the MBus (see below).

For more information, see

http://umaxp1.physics.lsa.umich.edu/cdf/document/alphaproc.htm

Since each preprocessor crate will create data for the Level 2 global decision, they are obliged to have VBDs for Level 3 readout. VBDs are VME bus masters, and this suggests that intra-crate communication for Level 2 business should not rely on the VME backplane. Another data bus is therefore required. The Level 2 group will use the CDF solution to this problem, which is to build a custom bus onto the P3 backplane of VME64x crates as endorsed by VITA (VME International Trade Association). See

http://www.vita.com

These VME crates are a variant of the VME64x standard as specified by VIPA (VME International Physics Association), document number VITA 23-199x, and are referred to as VIPA crates (or VME-for-Physics). VIPA documents can be downloaded from

http://www-ese.fnal.gov/vipa/default.htm

The crates are 10U in height (1U=1.75in), 400mm deep, can contain up to 21 cards, and differ from standard VME64x crates in the way power is brought into the

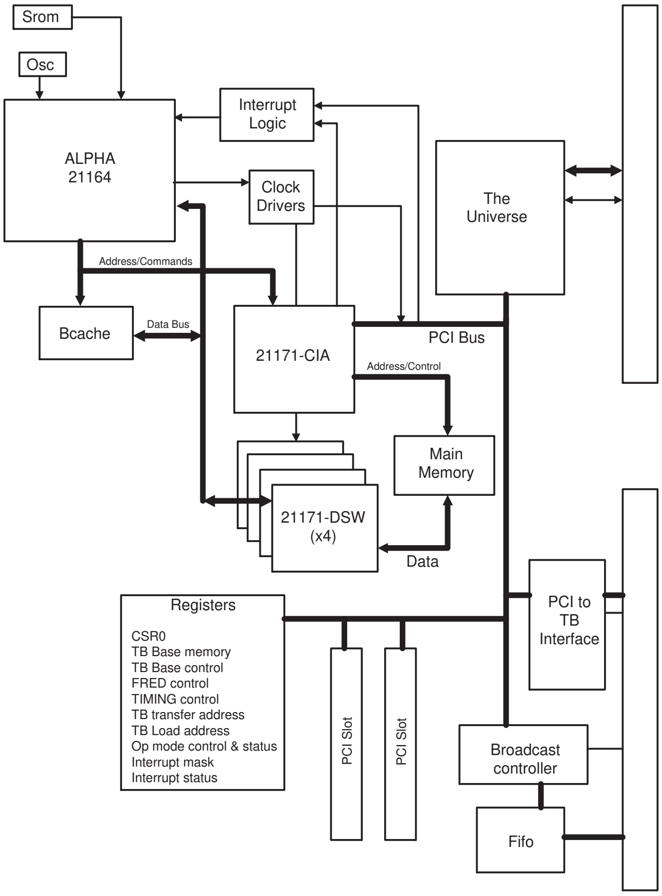

Figure 3: Schematic of the Level 2 ALPHA processor.

crate through the P0 backplane connector. The cards for these crates are actually 9U (see figure 4). A document on the 9U 400mm standard module can be obtained on the Web via

http://eng3.hep.uiuc.edu/vme.html.

Connector coordinates for the CDF/DØ level 2 crates are IEEE standards, document 1101.1, or VITA document 1.3. Figure 5 shows the connector layout. There are 4 backplanes, each with its own connector: J0, J1, J2, and J3. J0/J1/J2 form a monolithic, standard backplane, also used by CDF. J1 and J2 follow VME64x standards and uses a 5row x 32pin enhanced DIN and runs through all slots. J0 is a 2mm hard metric 5row x 19pin connector that, as mentioned, allows for non-standard analog voltages (*e.g.* -5.0 volts for ECL). J3 is completely user defined and will be used for a custom CDF-made bus described in the next paragraphs. The J3 connector is a non-standard VME64x 5row x 47 pin connector, with all pins on

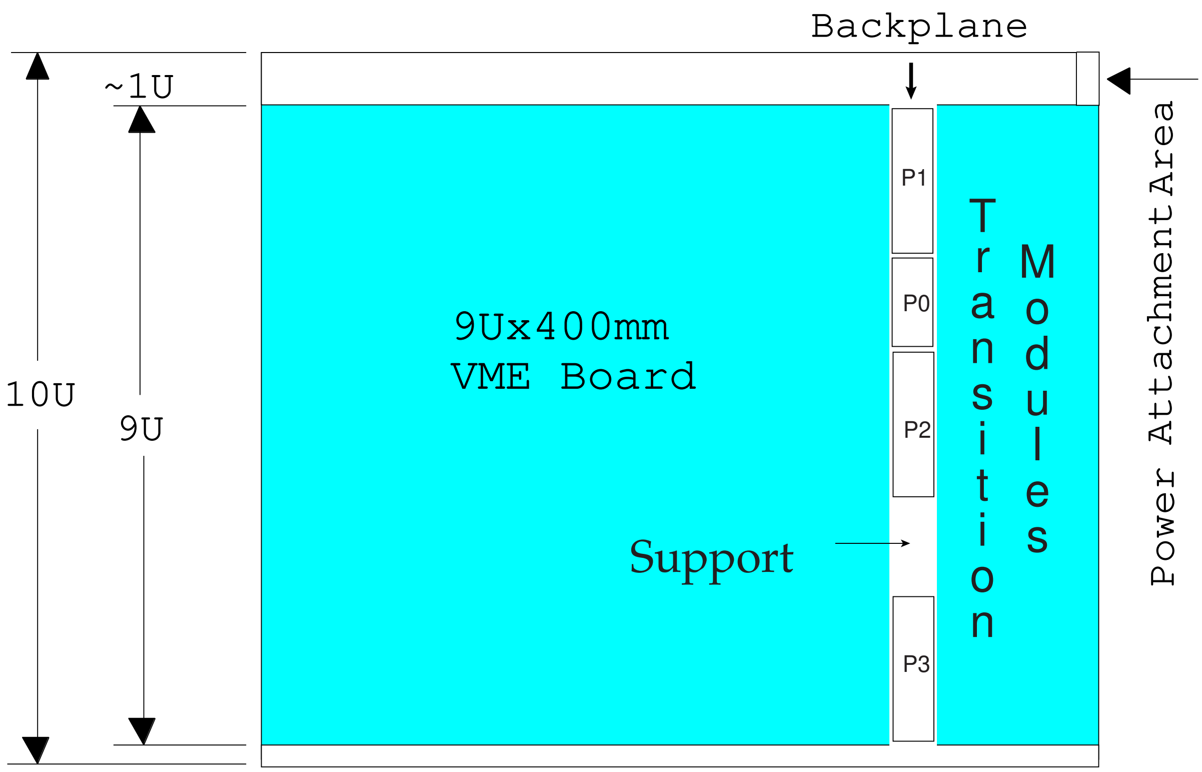

Figure 4: 9U VME64 card.

rows 34, 35, and 36 not used for data purposes. Table 1 lists the connectors and total pin count (of 645 pins). Using VIPA estimates for insertion and extraction forces, we can estimate that the forces required for insertion and extraction will be approximately 550N and 120N respectively. **As of this date, we haven't studied how we're going to pull these cards out,** *e.g.* **stiffeners, tools,** *etc.*

Connector	Rows	Pins/Column	Total Pins
J0	5	19	90
J1	5	32	160
J2	5	32	160
J3	5	47	235
TOTAL			645

Table 1: VME crate backplane pin count.

This new custom data bus is being designed and built by the CDF Michigan group:

http://umaxp1.physics.lsa.umich.edu/cdf/upgrade.htm

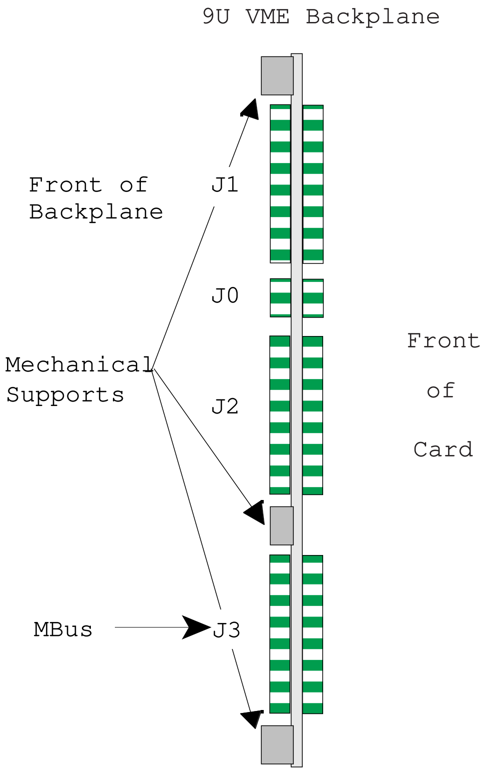

Figure 5: VME64 backplane connector layout.

The bus, called “Magic Bus”, or MBus, consists of a card mounted on the VME crate backplane, with bus lines connected to P3. The bus consists of 128 data bits and 32 address bits. See Table 2 for the current list of address, data, and special lines available as part of the MBus. The bus is asynchronous, and supports both broadcasting and programmed I/O. At this time there are no comprehensive measurements as to maximum bus rates, however bus operation is relatively simple: load address and data lines, raise DSTROBE, the receiver latches the 128 data bits and raises DDONE. DDONE high is the signal to lower DSTROBE (which releases DDONE) and begin another cycle. This sequence is the same for both broadcast mode, and programmed I/O. In fact there is really no difference between the two as far as the MBus is concerned – the difference is only evident on the ALPHA-PCI side of things. Note that a bus master cannot release the bus (release BOSS) until both DDONE and DSTROBE are released. Should an MBus card become hung (*e.g.* waiting for DDONE), an MBRESET will be necessary. When MBRESET is asserted, all MBus boards should pull their signals off the MBus. This line is controlled by the ALPHA processor card.

Estimates for cycle times are around 50ns, which would correspond to a nominal 20MHz cycle time, giving 20MHz$\times$16bytes=320Mbyte/sec throughput. The PCI (64 bits) bus in the ALPHA card is nominally rated at 33MHz, or 240MByte/sec,

Type	Bits	Mnemonic	Description
Data	128	MBDATA	Data lines
Address	32	MBAD	Address lines
Cycle type	1	RD/WR	If asserted, a read cycle, otherwise a write
Timing	1	DSTROBE	For a read, address is valid; for a write, address and data are valid (from master)
	1	DDONE	Open collector signal. For a read, data is valid; for a write, data has been latched by the slave
Arbitration	1	BOSS	A module is in control of the bus
(PECL)	1	BOSSREQ	A module requests control of the bus
	1	BOSSGRIN	A module may take control of the bus
	1	BOSSGROUT	The next module may control the bus
Special	1	MBRESET	Reset the backplane - remove all strobes and data
	1	START_LOAD	Administrator signal to MBTs to start broadcasting into the processors
	2	BUFFER(1:0)	Not used by DØ
	4	EV_LOADED(0)	Open collector ready signal driven by MBT
		EV_LOADED(1)	L2 SCL data ready line driven by MBT
		EV_LOADED(2:3)	Not used by DØ
	1	AP_FIFO_EMPTY	Open collector signal driven by ALPHA processors set whenever ALPHA finds itself with an empty FIFO
	19	MOD_DONE(18:0) (DONE_OUT)	Each module has a DONE_OUT signal that is routed on the backplane to the appropriate MOD_DONE bit. Slot number is n+2 (MOD_DONE(0)=slot 2)
Ground	18	GND	Signal return

Table 2: MBus signals.

however the University of Michigan experts agree that the realized performance will be lower than this (more like 80MByte/sec), and is thus the determining factor in getting data in and out of the ALPHA. An MBus derating would therefore have to be more than x2 to become the bottleneck.

Data will get in and out of the ALPHA cards via MBus, originating and terminating on the MBT card. The rest of this document concerns the MBT card. For a more detailed description of the DØ Level 2 system, see the TDR document DØ note 3402, February 1998,

http://www.pa.msu.edu:80/hep/d0/ftp/l2/overview/globaltdr/global_tdr.pdf

for PDF format, and .ps for POSTSCRIPT.

2 MBT Introduction

Table 3 summarizes the MBT Card. The MBT web page is maintained at Maryland, and can be found at http://macdrew.physics.umd.edu/dzero/trigger/mbt.html

A functional diagram of the MBT card is seen in Figure 6. The MBT card is a single-width 9U 400mm VME-64 for Physics card which collects all of the support functions that are needed in the ALPHA Processor based Level 2 Trigger Crates onto one card. In order to achieve this, the MBT will have 7 HotLinks serial digital data inputs running at 160 Mb/sec (or 16 MByte/sec in encoded mode, see below), 2 HotLinks data outputs running at the same rate, 1 HotLinks output for Level 1 SCL information fanout purposes, an SCL mezzanine card (for trigger hardware framework communication), a special 128 bit ECL output line which will deliver 16 bytes of information in about 0.5μsec or less (for Level 2 hardware framework communication and other purposes), and both VME and MBus interfaces. We will adopt the philosophy that all operations which do not occur in real time (with respect to data taking) will be over the VME bus, others will be over MBus.

The MBT card will be designed around the Altera 10k100 FPGA (henceforth referred to as MAIN). This gate array is quite large, and was chosen primarily to be able to eliminate internal data buses as much as possible. We therefore will keep as resident inside MAIN as many registers, counters, *etc.* as is possible. All MBus and VME data and address lines are connected to MAIN, as are the input FIFOs, and various other smaller gate arrays. The internal MBT clock will be 80MHz, or 12.5 ns "ticks". The version number of the firmware will be available for a VME read at address offset 22 decimal (see section 8).

Internal Clock	80 MHz (12.5 ns)
VME Crate Buses	VME R/W, MBus R/W
Input Data Sources	7 HotLinks receivers, 16MB/sec
Output Data Sources	2 HotLinks transmitters, 16MB/sec
	1 SCL Level 1 Fanout, 16MB/sec
SCL Mezzanine Card	(MBT Pilot only)
L2 HWFW Output	128 ECL lines @ 5 MHz, or
	16 PECL levels
MBT State Register	VME Readable
	Bit 0=1 indicates READY
	Bit 1=0 for data, 1 for test inject

Table 3: MBT Summary.

In each crate, there will be a "Pilot" MBT, and possibly one or more "Assistants", selectable via a switch on the board. A front panel LED will be lit for the PILOT. Differences between the two are detailed in section 12, but mainly have to do with functions not related to data taking (*e.g.* SCL communication).

The MBT card has four main functions:

1. **Receiver for Input**. This is used in the Global Crate to receive Preprocessor data and in some of the Preprocessors Crates to receive input data from their

Level 1 trigger data transmitters. There are 7 channels of Cypress HotLink input running differential over unshielded twisted pair at 16 MB/s (described below). Data received on these links is buffered and then sent via the MBus to the ALPHA processor boards.

2. **Transmitter for Output**. This is used in the Preprocessors to send their data to the Global Crate. There will be 2 channels of Cypress HotLink output, also running differential over unshielded twisted pair at 16MB/s. Data sent out on these links comes from the MBus, passes through a FIFO and is then sent out.

3. **Communication to the L2 Hardware Framework (L2HWFW)**. The Global processor will need a way to send the L2 decision result back to the L2HWFW. This will be via 128 lines of Parallel Digital I/O, driven by the MBT card. This can also be used in both the Preprocessor and Global Crates for real-time scaler and state control.

4. **Serial Command Link (SCL) Receiver**. This is used in both the Preprocessor and Global crates. The L1 Accept, L2 Decision, and SCL Initialize information is received and made available to the ALPHAs via the MBus (see below). In addition, the L1 Accept information will be sent out on shielded twisted pair using the same HotLinks transmitters. The status "Error" and "Busy" for both Level 1 and Level 2 are communicated back to the SCL hub through the SCL.

As detailed in section 9 below, the MBT must have a state register per card which reflects whether the card is "ready" or not. This register can be read over VME by the ALPHA (see section 8). The register will have a single bit, and if the MBT card is ready, that bit will be set to 1. This bit will be the LSB read over VME.

In the following subsections, we discuss the electrical characteristics of the HotLinks receiver/transmitter connections. See

http://macdrew.physics.umd.edu/dzero/trigger/hl-implementation.html

for details.

2.1 Cable Specification

Current plans call for using 1000Base-T unshielded twisted pair (UTP) cable, manufactured by Berk-Tek. Each cable (see figure 8) consists of 4 twisted pairs, 24 AWG, $\sim 100\Omega$ impedance, with thermoplastic insulation in a flame-retardant polymer alloy jacket with a 0.195 in diameter. Each pair is guaranteed to exceed

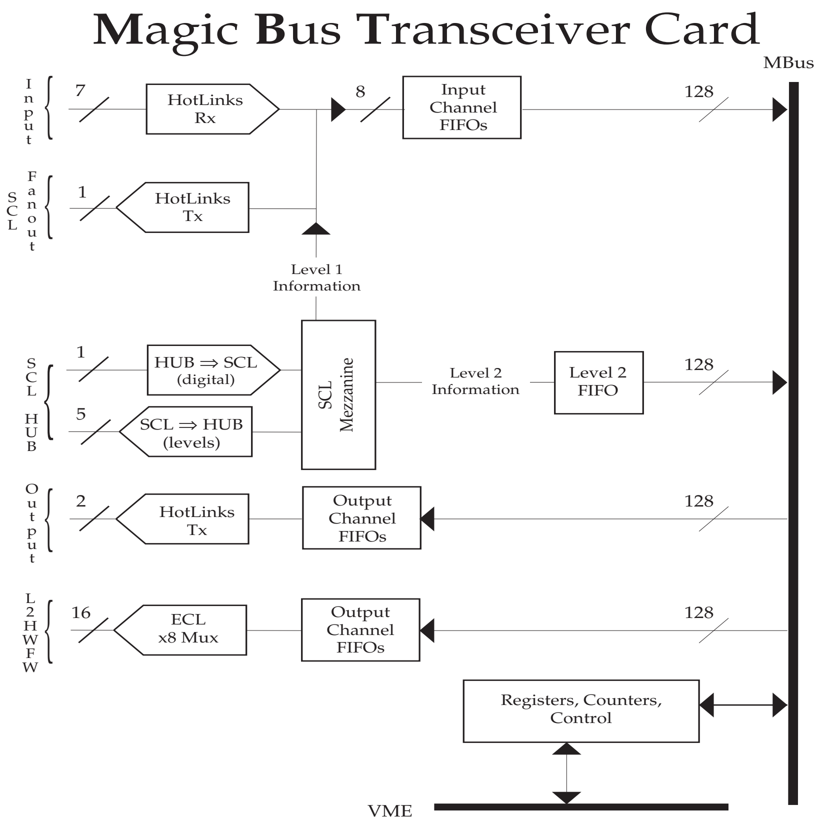

Figure 6: The MBT card.

Category 5 specifications. We will use 1 pair for data, grounding the other 3 through capacitors to eliminate DC ground offsets from crate to crate. The advantage of these cables is in the ubiquitous nature of 1000Base-T communication – the cables and connectors are very inexpensive, and should meet its specifications to have any impact in the commercial 1000Base-T market.

The actual electromagnetic bandwidth of each pair is of approximately 100MHz. However, the ratio of Mbit/sec to MHz can be large (in this case, a factor of about 5) if the cable is well. This is accomplished via manufacturing techniques which can result in, for instance, a low capacitance per meter (independent of twists in the overall cable), high db ratio in cross-talk between pairs, and well balanced pairs which result in a group delay which is frequency independent. As a result of many manufacturing techniques, the UTP quad-pair cable we are proposing are capable of running 500 Mb/sec on each pair in half duplex mode, with two pair sending at 500 Mb/sec and two pair receiving, resulting in an overall information transmit rate of 1000 Mb/sec.

Tests with this cable using a single transmitter and receiver pair running at 250 MB/s in BIST mode over 100 feet resulted in no errors after 17 hours of running overnight, or less than 1 error in in 1.53×10^{13} bits. This is translated into a bit error rate (BER) of less than 6.5×10^{-14}. The "eye" plot is shown in figure 7. Note that 3 of the 4 pairs are grounded.

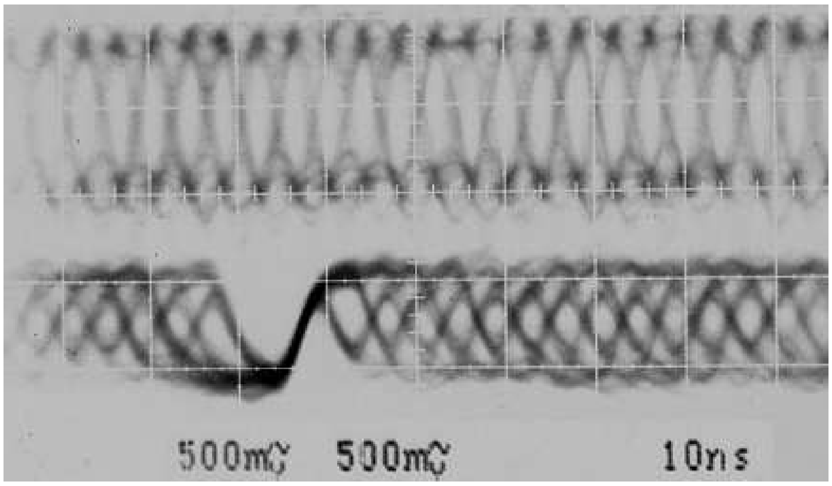

Figure 7: "Eye" plot for 250MHz in BIST mode over 100 feet of 1000Base-T UTP cable.

The exact Berk-Tek part number is 230716 for the tray and plenum rated cable. See

http://www.berktek.com/

for the Berk-Tek home page, and

http://macdrew.physics.umd.edu/dzero/trigger/cable.html

for the (borrowed) page that describes these cables.

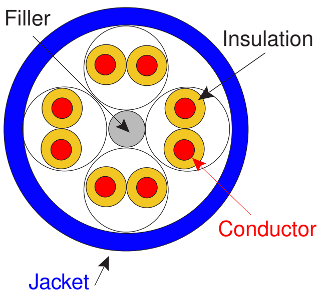

Figure 8: Schematic for proposed 1000Base-T cables to be used for Cypress HotLink transmission.

2.2 Connector Specification

The connector we will use with the 1000Base-T cable is an RJ45, and we will be using pins 4 and 5 for the "positive" and "negative" going PECL signals. All other pins should be grounded through a blocking capacitor. Figure 9 details the pins used, from the point of view of looking into the connector.

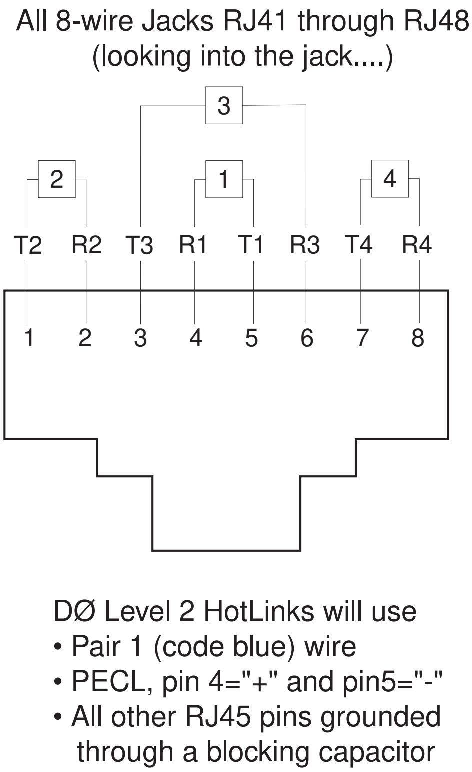

Figure 9: RJ45 connector as seen looking into the connector.

2.3 HOTLink Electrical Implementation Standards

Each UTP cable contains 4 twisted pair. Most systems in Level 2 will use a single pair for data, with 3 pair left for grounds. The muon SLIC card is the exception, and they will use 2 pair for data and 2 for ground. Figure 10 shows the recommended specification for data. We propose using transformers on both input and output, to be safe, with 4.7kΩ pulldown resistors on the cable side of the transmitter (downstream from the transformers) to keep the common mode potential at a reasonable level. Again, this is a conservative design. All transmitter/receivers which use a single pair (of the 4) should use pins 4 and 5 of the RJ45 for "+" and "-" PELC respectively. For those systems using 2 pair for data, we recommend using pins 1 and 2 for "+" and "-" PELC respectively.

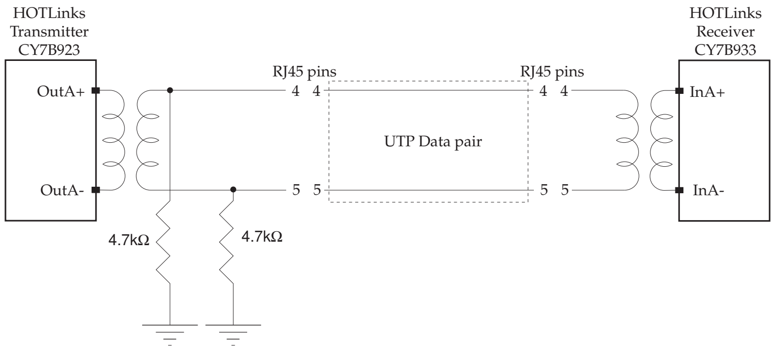

Figure 10: Specification for connecting one twisted pair for data.

The remaining 3 (or 2) twisted pair should all be grounded through blocking capacitors at the receiver, as shown in figure 11. We estimate that the shield cutoff frequency (which take into account the mutual inductance of the data and shield twisted pair) is approximately 1kHz. Using this estimate, we recommend using a 1μF blocking capacitor, as in the figure.

Figure 11: Specification for connecting grounds.

3 Receiver Section

The receiver section has 7 input channels of HotLink serial-to-parallel receiver chips (see figure 12). Data is sent over cables (currently, unshielded twisted pair as described below), and is stored in FIFOs. The entire input section is described below.

3.1 HotLink Transmitter/Receiver

HotLink is a trademark of Cypress. The transmitter/receiver chipset is used for point-to-point communication, and serves as building blocks for systems which transfer data over high-speed serial links (fiber, coax, and twisted pair). For technical information,

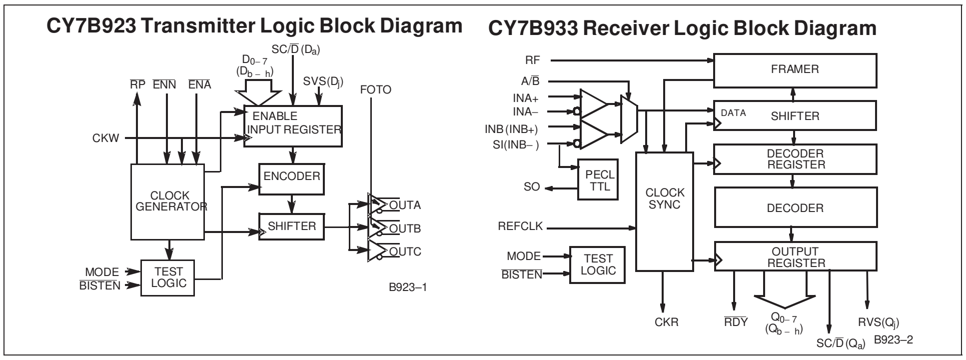

Figure 12: The Hot Links receiver and transmitter schematic.

http://www.cypress.com

has many documents, especially

http://www.cypress.com/hotlink/hotlinkappnotes.html

for relevant HotLink documents, and

http://www.cypress.com/pub/datasheets/cy7b923.html

for the HotLinks receiver transmitter description document. Note that we will probably want these documents around for awhile, so I have copied the relevant ones to the MBT home page, where you can access them from:

http://macdrew.physics.umd.edu/dzero/trigger/mbt.html

We will be using the transmission mode "Encoded mode" which consists of 8 data bits encoded with 2 control bits, transmitted serially over positive differential "pseudo" ECL (PECL). The data byte rate (16 MByte/sec) is 1/10th the transmission speed. The output register of the HotLinks receiver presents 8 data bits in parallel, a byte rate clock generated from the incoming (transmitted) data (CKR), a signal for "Data Ready" ($\overline{\mathrm{RDY}}$), a "Special Character" flag bit (SC/$\overline{\mathrm{D}}$), and an error bit (Received Violation Symbol, RVS). The state machine running in the PLD will be keying on $\overline{\mathrm{RDY}}$, SC/$\overline{\mathrm{D}}$, and RVS.

Table 4 lists the relevant input and output lines for the HotLinks receiver. The input reference clock is REFCLK, and this clock must be within $\pm 0.1\%$ of the transmitter clock CKW. For our purposes, the transmitter and receiver reference clock will be the same chip, running at approximately 16MHz. The HotLinks specification is that the receiver and transmitter clocks be within $\pm 0.1\%$ before synchronization, but the absolute rate is not critical. This is discussed in more detail below.) CKR is generated using REFCLK initially, and "stretched" to coincide with the transmitter byte clock (CKW) after (described below). All of the important receiver chip output lines switch synchronously with the rising edge of CKR. We will be using input lines A (INA). The other input lines will be tied to +5V to decrease noise. Note that INB and SO can be used in conjunction to accept TTL input, not implemented in the MBT.

Name		Configured	Description
MODE	In	LOW	Encoded Mode
REFCLK	In	16 MHz	External Byte Clock
CKR	Out		Internal Byte Clock
SC/$\overline{\text{D}}$	Out		HIGH=data, LOW=special character
RVS	Out		LOW=normal, HIGH=violation symbol
$\overline{\text{RDY}}$	Out		Data Output Ready (described below)
Q_{0-7}	Out		8-bit data received
$\overline{\text{BISTEN}}$	In		HIGH=normal, LOW=BIST mode
$\overline{\text{RP}}$	Out		Ready for next byte
INA			Serial Data Input lines
INB		+5V	Not Used
SO		+5V	Not Used

Table 4: HotLinks receiver pins.

Each 10-bit "frame" can assume one of 1024 possible values (10-bit "characters"), of which 8 bits, or 256 data bytes, are legal data. The remaining possibilities are either legal special characters flagged by SC/$\overline{\text{D}}$ high, or illegal special characters flagged by RVS high. Note that the HotLinks receiver internal clock time t_b is 10 times the external byte clock time t_B, that is, the rising edge of CKR occurs in a time of $10t_b$ after the previous rising edge. All lines rise and fall based on some integral number of t_b. During normal operation, $\overline{\text{RDY}}$ is used to signal when new data is available at the output pins.

Table 5 describes the state of the lines MODE, RF, and $\overline{\text{BISTEN}}$ for the different encoded modes normal, BIST, and reframing. Default values for our purposes will be: MODE=LOW (encoded mode), $\overline{\text{BISTEN}}$=HIGH (BIST mode off), and RF=LOW (RF disabled) when in standard data taking configuration (see figure 12). Each of these modes are described in subsequent sections.

	MODE	RF	$\overline{\text{BISTEN}}$
Normal Encoded	low	low	high
BIST	low	low	low
Framing	low	high	high

Table 5: MODE, RF, and $\overline{\text{BISTEN}}$usage.

3.1.1 Valid Data and Special Characters

Once the receiver and transmitter are synchronized (described below in section 3.1.2, the HotLinks receiver will frame the data onto the correct byte boundary (described below). The SC/$\overline{\text{D}}$ bit will be HIGH to signal valid "special character" when any special character is received (see "Valid Special Characters Table", from the HotLinks document, cy7b923.pdf on that document's page 31), and LOW to signal valid data. Should there not be data transmission over the line for any length of time, the transmitter will send the special character K28.5, the standard HotLinks "PAD" character, which will insure continual clock synchronization. For our purposes, we will use a subset of the special characters available as detailed in table 6 (as of this date, we have still not officially decided between the EOE character as proposed by Renardy and Baldin). Note that $\overline{\text{RDY}}$ is low (active) in encoded mode for all valid data and special characters except for the special case of PAD characters, in which $\overline{\text{RDY}}$ is low only for the last K28.5 in a stream, or a single one (see figure 13). This allows receivers to be stateless, and simply put all valid bytes into FIFOs, but avoiding filling those FIFOs up with a long stream of PADs. For our purposes, we will not be running stateless, as we will have to distinguish between the two states IDLE (between end and begin) and DATA (between begin and end, see section 3.2.1 below), and will therefore not be putting any K28.5 characters into our input FIFOs (see below).

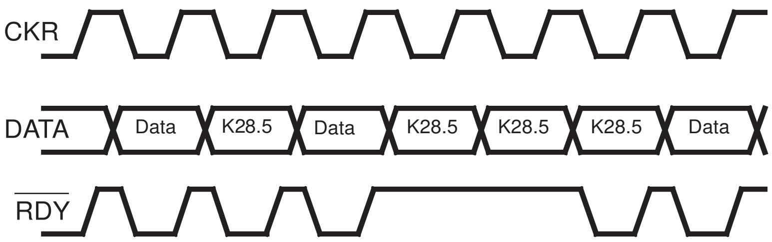

Figure 13: Normal $\overline{\text{RDY}}$ operation in encoded mode.

Special Character	Use	
K28.0	Begin-of-event (BOE)	
K28.3	End-of-event (EOE)	(RENARDY)
K23.7	End-of-event (EOE)	(BALDIN)
K28.5	PAD (filler character)	

Table 6: HotLinks special character usage.

3.1.2 Synchronization

Synchronization of the receiver with the transmitter consists of two parts:

- clock synchronization of the receiver to the transmitter (receiver tracks the incoming bit stream), and
- framing of the incoming bits in order to properly align to the intended byte boundaries.

Clock synchronization, or phase locking, requires that both the transmitter and receiver have onboard clocks which will initially have to run at the same frequency to 1 part in 1000 (or equivalently, ±500 PPM). Experience suggests that all DØ HotLinks implementation use the same onboard crystals. Following the muon group, we will use the standard HCMOS compatible quartz oscillator available from many distributors. In particular, we will use the RALTRON C06100 16 MHz oscillator which is rated at ±100 PPM, meeting the HotLink specification.

Synchronizing the phase and frequency of these two clocks is commenced by driving RF high. This enables circuitry which will "stretch" the receiver's internal default clock (generated by the crystal, derived from REFCLK) to the transmitted clock. The receiver latches RF at the falling edge of CKR, and at this point (RF=HIGH), $\overline{\mathrm{RDY}}$ leaves normal Encoded mode operation, and is forced HIGH a time t_b after the next rising edge of CKR. The only exception to this is when there is a K28.5 (PAD) in the framer already. In that case, an additional $\overline{\mathrm{RDY}}$ pulse will occur after RF is latched HIGH. Due to the latency of the receiver pipeline and control logic, $\overline{\mathrm{RDY}}$ will not pulse low before 4 CKR clock cycles, therefore external logic should be designed to not examine $\overline{\mathrm{RDY}}$ before 4 clock cycles have passed. Once locking is in place, the receiver chip searches for the PAD character K28.5 to establish the correct byte boundary (framing). Once framing has been established, $\overline{\mathrm{RDY}}$ resumes its normal encoded mode operation. The host can then disable framing (RF is set low) for normal operation if so desired, however if RF remains HIGH, the framer still continues to search for K28.5 to frame the data. If RF is high for more than 2048 CKR cycles, the framer converts to a double-byte framer requiring two K28.5s within five bytes for framing. $\overline{\mathrm{RDY}}$, however, remains in normal mode. Note that for our purposes, once $\overline{\mathrm{RDY}}$=LOW (reframing established) we will drive RF LOW for normal data taking, since once framing is established, we do not need to constantly reframe (see below). Figure 14 shows the timing sequence for $\overline{\mathrm{RDY}}$ in framing mode.

The experience of most HotLinks users in HEP suggests that the initial phase locking might take many PAD characters sent by the transmitter (milliseconds is a common claim), but that once the clocks are synchronized, they stay synchronized - phase unlocking is almost entirely absent. The establishment of the byte boundaries (byte framing) using the PAD character is, on the other hand, expected to occur quickly. Cypress claims that this can be accomplished using very few transmitted PADs as described above.

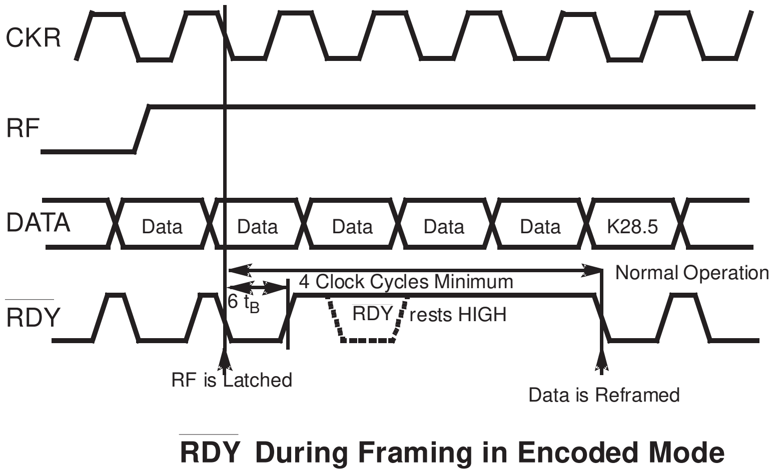

Figure 14: $\overline{\text{RDY}}$ operation in framing mode (RF=HIGH).

3.1.3 BIST Mode

HotLinks can be run in BIST (Built In Self Test) mode in order to measure the bit error rate (BER) for the physical layer. In BIST mode, the transmitter sends a 511-byte pseudo-random pattern (which constitutes a single BIST cycle)using its input register, and the receiver compares the BIST serial data stream with identical BIST patterns in its output register as generated by the same mechanism as the transmitter. All logic in the transmitter (except the input pins) and all logic in the receiver are checked by BIST, and all of the serial link interconnect components are exercised with normal data patterns, which are checked byte-by-byte in real time. The BIST cycle is recognized by the receiver when in BIST mode, allowing the receiver to synchronize. Errors are flagged by the same HotLinks output bit (RVS) as in data mode. These errors will be counted.

BIST mode is entered by asserting $\overline{\text{BISTEN}}$ LOW. $\overline{\text{RDY}}$ then enters its BIST mode of operation, and is asserted LOW for once CKR cycle. After the next CKR cycle (time t_B = 1/16MHz∼ 62.5ns), $\overline{\text{RDY}}$ is forced high to signal that the BIST logic is checking the incoming data for the start of the BIST cycle (D1.0 followed by D0.0), and remains high until the sequence is found, after which it will transition LOW a time t_b after CKR rises (this is due to the pipeline nature of the receiver). $\overline{\text{RDY}}$ will remain LOW for the duration of the BIST cycle except to pulse HIGH at the start of ech loop (every 511 bytes). One clock cycle after $\overline{\text{BISTEN}}$is asserted HIGH, $\overline{\text{RDY}}$ resumes its normal operation (depending upon the MODE selected). Figure 15 shows the timing for $\overline{\text{RDY}}$ entering, during, and leaving BIST mode. BIST errors are signaled on RVS (**check what $\overline{\text{RDY}}$ is doing when RVS is asserted**).

We will implement BIST mode in the following way (see figure 16):

1. BIST mode is entered after the MBT receives the correct VME command (see below). All active input channels, and both output channels, are affected.

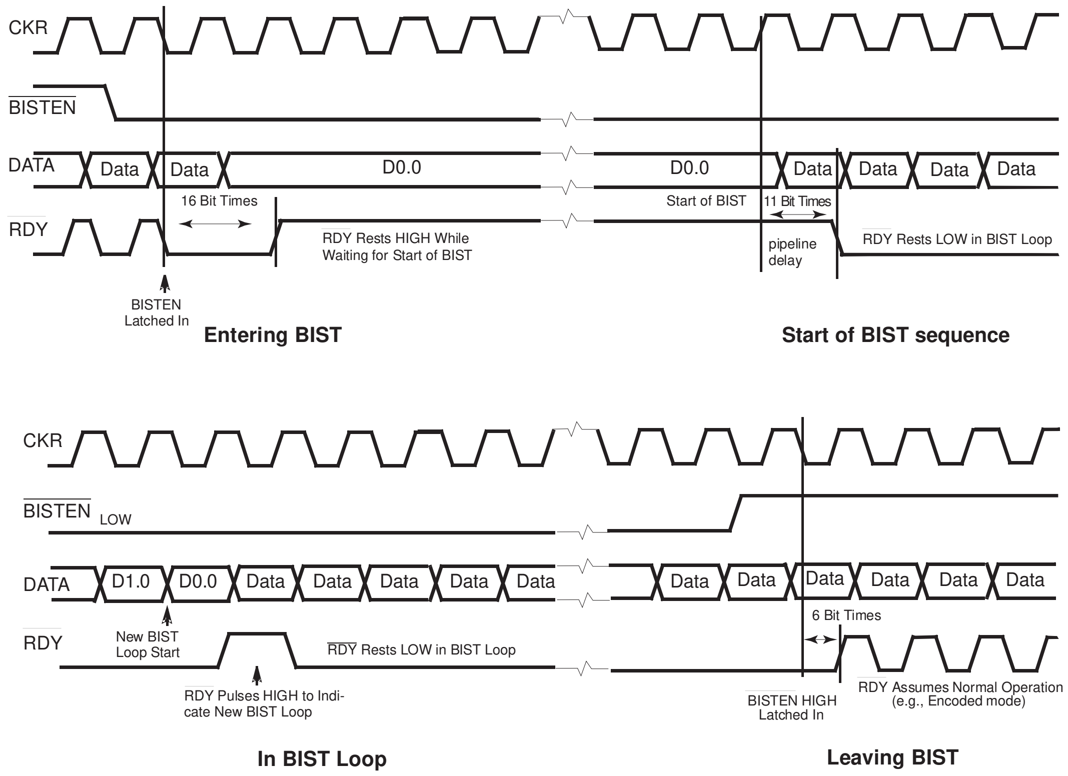

Figure 15: $\overline{\text{RDY}}$ operation in BIST mode ($\overline{\text{BISTEN}}$=LOW).

Upon entering BIST mode, the BIST error counter is cleared.

2. The MBT will lower the $\overline{\text{BISTEN}}$ line enabling BIST mode operation, for all **(active)** input, both output channels, and the SCL fanout channel. **If a receiver which has no connection goes into BIST mode, we believe that nothing will happen, since there will never be BIST synchronization. However, we are still not completely sure about this, and are investigating.**

3. Each BIST cycle (511-bytes at 16MByte/sec, or approximately every 32μs) will be noted by a pulse received on $\overline{\text{RDY}}$. After receiving 10 of these pulses (a precaution against transients in BIST startup), BIST cycles will not be counted. Bit error rates (BER) should be calculated by keeping track of the time. This should be adequate to get orders of magnitude estimates of the BER. BIST errors, however, will be counted in a BIST error counter. The BIST error counter will be the same as the error counter for the DATA state, and the BIST cycle counter will be the same as the error counter for the IDLE state (**but we're really not certain yet**).

4. BIST mode is canceled via the correct VME command (see section 8 below).

5. Errors can be collected via the correct VME command (see section 8 below).

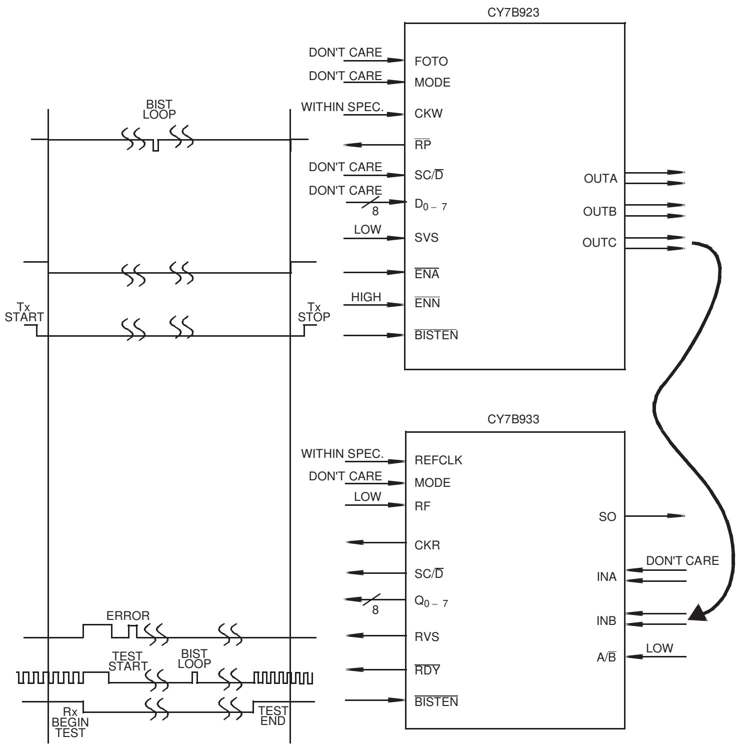

Figure 16: Built In Self Test (BIST) illustration.

3.1.4 Input Channel State Register

Each of the 7 HotLinks input channels will have an 8-bit state register which can be read over VME in order to ascertain the state of the input channel state machine (see section 8 below for more detail on how to read these registers). The bits in the state register are described in table 7. Some of these bits are control bits, but are not directly accessible via programmed I/O by the ALPHA. Instead, these bits are written to by the main FPGA (MAIN) whenever MAIN receives the appropriate communication via programmed I/O from the ALPHA as described in section 8.

At this time, the bits in these registers are:

0 **State** – 0 is IDLE, 1 is DATA. The default is IDLE (0). This bit can only be read over VME, and cannot be changed externally except during a programmed reset or powerup (see section 9.1 and 9.2).

1 **MODE** – this should always be 0, or low, indicating normal HotLinks encoded mode. This bit cannot be set, and is provided for debugging purposes.

2 **RF** – reflects what the logic in the PLD believes should be the state of the RF line, used for reframing. This bit is meant to be read as an indication that the input channel is (or is not) framing.

3 $\overline{\textbf{BISTEN}}$ – drives whether the input channel is in BIST mode or not. If this bit is driven high (1), the channel goes into BIST mode. Default is 0. Note that to set a channel mode to BIST, you write a 1 into this bit, which will cause the line $\overline{\text{BISTEN}}$ to be driven low, therefore in effect this bit is equivalent to .NOT.$\overline{\text{BISTEN}}$. This bit is cleared during a programmed reset or powerup.

4 **SCL Reset** – set for SCL Reset. Default is 0.

5 **SPARE**

6 **Type 1 Errors** – set if there is at least 1 Type 1 error for the state reflected in bit 0. See section 3.2.2 for a description of Type 1 and Type 2 errors. This bit is read only, and is 0 by default.

7 **Type 2 Errors** – set if there is at least 1 Type 2 error for the state reflected in bit 0. See section 3.2.2 for a description of Type 1 and Type 2 errors. This bit is read only, and is 0 by default.

Bit	Description	Value		Default
0	State	0=IDLE, 1=DATA	R/O	0
1	MODE	0=Encoded, 1=Bypass	R/O	0
2	RF	0=normal, 1=reframe	R/W	0
3	BISTEN	0=normal,1=BIST	R/W	0
4	SCL Reset	0=normal, 1=init	R/W	0
5	Spare			
6	$\geq$ 1 Type 1 Errors	0=none, 1=$\geq$ 1	R/O	0
7	$\geq$ 1 Type 2 Errors	0=none, 1=$\geq$ 1	R/O	0

Table 7: Input channel state register description.

3.2 MBT Input Channel Description

The MBT will have 7 channels of HotLinks input sections as described above. All HotLinks channels will operate in Encoded mode, selected by setting the lines MODE and RF low, and $\overline{\text{BISTEN}}$ high. All inputs receive data in parallel, asynchronous to each other. Each of the 7 channels will feed a set of input FIFOs. An 8th channel (channel 0) will be used to buffer the SCL Level 1 data from the SCL input mezzanine card (as described in section 7.1 below). This channel will behave exactly like each input channel in terms of the way data is buffered and read out.

Each channel will be controlled by a PLD finite state machine. The state of the machine is controlled by the MAIN FPGA. Figure 17 shows the various states, and how these states are entered and exited. Each state is described in this document. Note that the states **IDLE** and **DATA** are defined for normal experimental running configuration, while the states **BIST** (see section 3.1.3), **QUIET** (see section 7.3), and **Test Data Injection** (see section 4.3) are defined for the non-normal running conditions concerning diagnostics and SCL resets, as described.

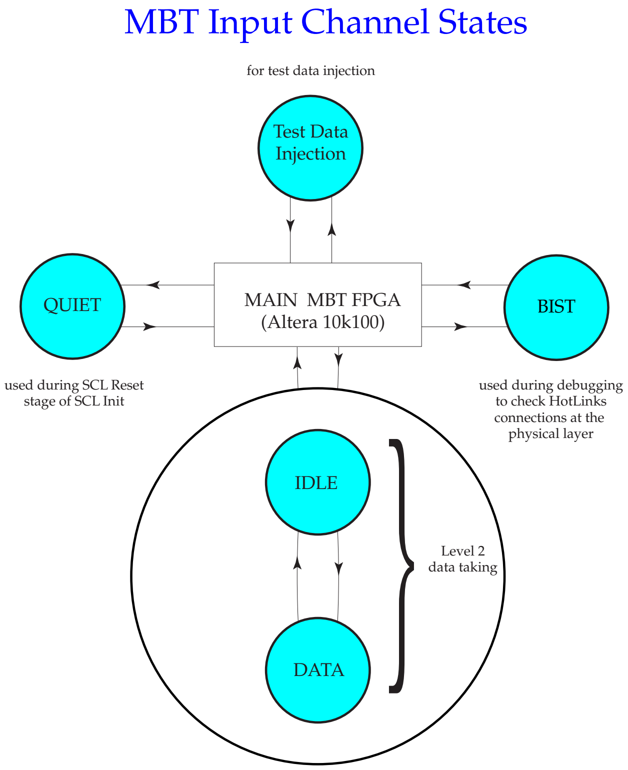

Figure 17: Input channel states

Each FIFO set will be large enough to hold 64k bytes of data, or 16 events at 4k bytes/event on average. This number is, we believe way beyond the expected event size, and is determined from a Monte Carlo calculation using ISAJET low-p_T jet data with 7x multiple interactions and considering tracks in the CFT. Calculations suggest that even an event with 7 additional interactions will rarely, if ever, result in as many as 400 tracks. If we require that the MBT be able to hold 16 events in its input FIFO, we will need these FIFOs to be 400 tracks/event $\times$ 4 bytes/track$\times$16 events = 25,600 bytes in a single channel. In fact, these data will be spread over the seven MBT inputs, however there are many Level 2 pre-processor crates which will effectively take Level 1 data spread over its inputs, transform the data in the

ALPHA (and in some cases, *e.g.* the CFT pre-processor, will actually increase the data size) and output the data over fewer output channels than are input channels.

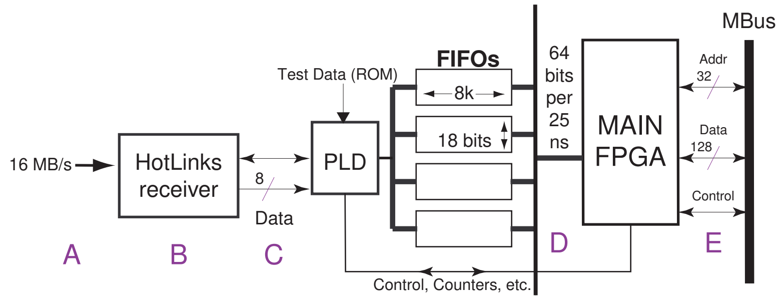

Figure 18: Schematic for a single MBT input channel.

The receiver section for each channel is shown in Figure 18. At point A, data is received differentially over twisted pair into a front panel connector, entering a single HotLinks receiver (point B). The HotLinks chip will take 10 bits serial, decode the extra 2 bits, and produce 1 byte of data (point C) at 16MHz. The HotLinks also provides a clock and a status bit derived from any special character sent. A PLD (Altera EPM7064, 64 logic cells) will monitor the data flow, multiplexing the data bytes into the proper place in a set of 4 FIFOs (FIFO bank). The PLD will buffer every other byte so that it can write 2 bytes (plus two possible control bits, making a total of 18 bits) at a time into the FIFOs. As discussed in section 4.3 below, test data can be injected via VME into the MBT front end FIFOs. Multiple events, or loops, will will have to originate in a separate VME card to be built, and will be described in section 10. Legal special characters will not be put into the FIFOs, but will be intercepted by the PLD and passed along in the appropriate way. At this time, only the PAD character indicating "idle" (also used for framing as described above) and the begin/end-of-event (BOE/EOE) special characters are legal special characters (see table 6), however other special characters might could be considered.

Each FIFO is 18 bits wide, with only 16 to be used for data, and can be read out every 25ns (manufacturer's design spec). To understand why 4 FIFOs per channel are used, consider the MBus goal of being able to broadcast 128 bits (16 bytes) every 50ns or so (20 MHz) into the ALPHA. The FIFOs are 2 bytes wide, therefore we will have to have addressed 8 FIFOs in the 50ns to get the 16 bytes required by MBus. Since we can address 4 FIFOs in parallel, all we have to do is read each of the 4 FIFOs, twice. We require 64k bytes total buffering capacity per channel, or 16k bytes per FIFO. Since each FIFO is 2 bytes wide, we need 8k time slices deep. The MBT input channel FIFOs will therefore be 8k deep, and 2 bytes (plus 2 extra bits) wide. Since the MBT onboard clock will be 80 MHz, or 12.5 ns, we will use the 25 ns to read each FIFO, which will realize a maximum MBus rate of 320 MHz. If we find that there are problems, we can always change the clock to 60 MHz, or 16.7 ns "ticks", or 33 ns for FIFO reads, which will limit the MBus to about 240

MB/sec (down from the maximum advertised guess of 320 MB/sec). This reduction would most likely not cost anything in terms of Level 2 dead time, since the real bottleneck in getting data from the MBT input FIFOs into the ALPHAs will be in the transfer over the ALPHA PCI bus.

At point D, 64 bits of data (and 8 bits of status) is read from the FIFO block (of 4) every 25ns into a single Altera FPGA (EPF10k100, 100k gates) which has access to all 8 of the above described channels' FIFO bank. It will clock 128 bits of data from a single channel onto the MBus every 50 ns as described above. This design is optimized for cost and success.

When each single channel broadcast is complete, the MBT will proceed to the next channel. Since each MBus cycle will consist of 2 FIFO reads of the 4-block of FIFOs in any single channel, each MBus 16 byte broadcast will consist of data only for a single channel. That is, we will not be able to combine data from 2 separate channels into a single MBus cycle broadcasting 128 bits. After 2 FIFO reads, we will have 128 bits latched into the MBus data lines, ready for MBus broadcast, (point E). MBus bus arbitration will take place between channel broadcasts, so the MBT will keep the MBus busy during an entire broadcast of all of the data from a given event from a single channel.

All devices transmitting to any MBT input are required to send data in "blocks" of 128 bits. Since the HotLinks chipset frames the byte boundaries automatically, this ensures that data will enter the MBT input 16 bytes at a time. The MBT gate arrays (input channel PLD and MAIN) maintains event boundaries inside the MBT input FIFOs in the following way:

1. Since each FIFO is 2 bytes wide (plus 2 extra bits), 8 FIFO slices are needed to store 16 bytes. Since the PLD will write 18 bits at a time into a given FIFO, it will have to buffer every other data byte from the HotLinks in order to construct the necessary 16 bits of data. As all normal MBT input will occur with 16 byte "blocks", during normal transmission/receiving the 16-byte boundary should always occur in the same place within the FIFOs (see Figure 19) modulo 2 slices.

2. A normal complete event transmission will result in the last byte being stored in the FIFOs in the proper place (the last byte in the 4th FIFO, see item above), after which the PLD should see an end-of-event character (EOE) after 5 (or more) PAD characters. The PLD will then rewrite the previous 2 bytes, setting the 18th bit (out of 18) indicating end of event, and setting bit 17 if an error condition has occurred (to be used by the Altera in the physical trailer, **but we haven't settled on this as yet**). Under normal operations, this FIFO position should be in the last byte of the 4th FIFO (see Figure 19). If error sin transmission which required reframing occurred, and reframing was established before the next EOE was received, the pointer for this EOE character will not necessarily be at the front of the 1st FIFO.

3. Begin-of-event (BOE) character, when received, will always result in a setting of the PLD FIFO pointer to the beginning of the 2x4 FIFO block (2 deep, 4 wide). The very next 2 data bytes will be sent to this location.

4. The Altera will always read 72 bits (4 FIFOs times 18 bits/FIFO) twice in order to put 128 data bits onto the MBus. The Altera will search for the EOE marker in the 18th bit of every FIFO data. If the flag for the event boundary does not occur in the correct place (due to a transmission or receiving error such as a framing error), the Altera will be responsible for padding up to 128 bits for the MBus broadcast.

The only exception to the 16 byte rule comes from the fact that physical trailers are mostly composed of 1 or 2 bytes of parity calculations. The MBT will pad zeros in the remaining MBus data bytes to make 16 total.

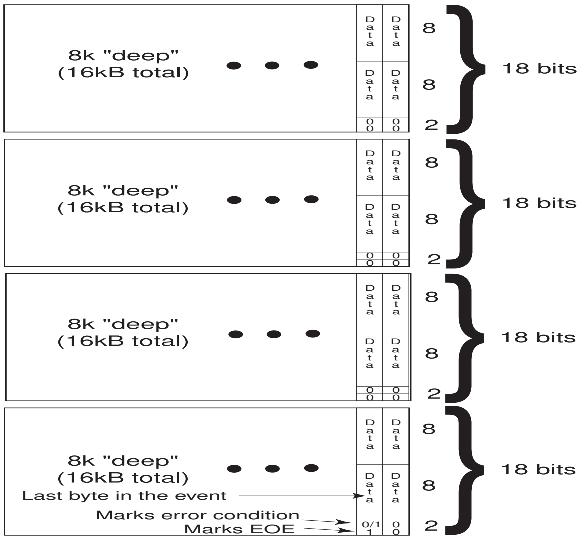

Figure 19: A single MBT input channel FIFO block.

3.2.1 The MBT 2-State Input Machine

The existence of both the BOE and EOE marker allows the MBT receiver section to behave as a two-state machine. The state diagram is seen in figure 20. The two states are **IDLE** and **DATA**. In each of these two states, the machine will have to deal with (see table 8 below: begin event special characters (BOE); end of event special character (EOE); pad character (PAD); valid data (DATA); and invalid data (RVS, or ERROR). In addition, a criteria for reframing must be established. Finally, the state of all active channels will be reflected instantaneously in the input channel state register which can be read via VME. The format of this register is described in section 3.1.4 and the command to read it is described in section 8. A bank of 7 LEDs on the front panel will also reflect the state of the channels. We will use the convention that 0=IDLE, 1=DATA. Note that upon initialization, the default state will be IDLE, and this will be reflected in the state register (and the front panel). Resets are discussed further in section 9. Given that the PLD will buffer every other byte before writing into the FIFO (18 bits), we believe that the PLD 2-state machine will have enough time to do its chores.

Condition	Line State				
	$\overline{\text{RDY}}$	SC/$\overline{\text{D}}$	RVS	MODE	BISTEN
SC (special character)	LOW	HIGH	LOW	LOW	HIGH
DATA	LOW	LOW	LOW	LOW	HIGH
REC_ERR	LOW	HIGH	HIGH	LOW	HIGH

FSM Input	Description using above table "Conditions"
Valid Data	DATA=TRUE
BOE	SC=TRUE.and.K28.0
EOE	SC=TRUE.and.K23.7 or possibly K28.3 (to be determined)
PAD	SC=TRUE.and.K28.5
Error Received	(REC_ERR=TRUE).or.($\overline{\text{BOE}}$.and.$\overline{\text{EOE}}$.and.$\overline{\text{PAD}}$)
Reframe	"n" consecutive REC_ERR=TRUE (programmable)

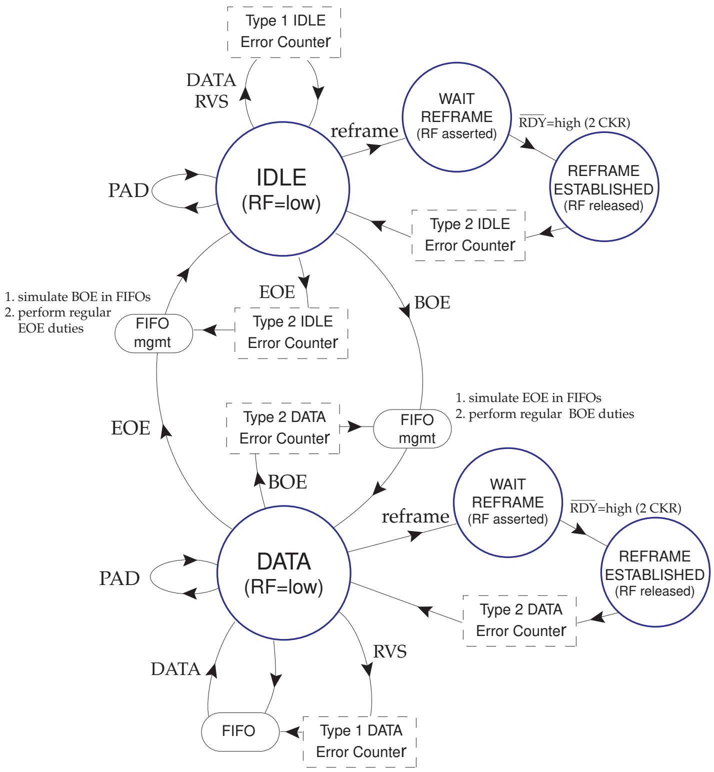

Figure 20: MBT input channel state machine.

The handling of all possible conditions while in **IDLE** state is described in table 8. Receipt of either PAD or BOE is considered "normal". PAD will send the machine back into the IDLE state, BOE will send the machine into the DATA state after FIFO management chores are complete (*e.g.* resetting pointers). Receipt of anything else (DATA, EOE, or ERROR) signifies an error condition. DATA and ERROR are most likely both due to a single bit transmission error which maps the new character into either a valid data or an invalid special character. Receipt of these two conditions will be counted, and the machine will return to the IDLE state. Receipt of an EOE while in IDLE state could be due to either an error which maps into the EOE character, or a possible loss (or lack of detection) of a previously sent BOE character. It is impossible to distinguish which of these two happened, therefore the machine will assume the latter (loss of a previous BOE), and will treat the EOE character in IDLE state in the same way that it will while in DATA state (see below) with respect to managing the FIFOs, with the addition that the error condition will be reported in the error counter. Reframing will be attempted upon receipt of two or more (programmable, see section 8) consecutive errors (valid DATA, invalid special characters, or RVS), and is described in detail in section 3.2.2.

The handling of all possible conditions while in **DATA** state is described in table 8. Receipt of EOE sends the machine into the IDLE state after FIFO management chores are complete (*e.g.* rewriting the last 2 bytes with the end-of-event bit set in position 18 of the FIFO slice). Receipt of PAD will send the machine back into the DATA state. Receipt of DATA results in a FIFO write and return to DATA state. Receipt of anything else (BOE, or ERROR) will cause the appropriate error counter to be incremented. Reframe will occur upon receipt of more than one consecutive RVS, and is described in detail in section 3.2.2. In addition, RVS will also cause a FIFO write. This is done in order to try to preserve the event length integrity. Note that the MBT will attempt to inform the ALPHA as to which byte had the error, using the 128-bit physical trailer which contains the parity calculation. **More on this later - how to do this?.**

	IDLE State		DATA State	
Received	Action	Condition	Action	Condition
PAD	Return	Normal	Return	Normal
BOE	Goto DATA	Normal	Count, Simulate EOE, IFO mgmt, Return	Type 2
EOE	Count, Simulate BOE, FIFO mgmt, Return	Type 2	FIFO mgmt, Goto IDLE	Normal
DATA	Count, Return	Type 1	FIFO, Return	Normal
RVS	Count, Return	Type 2	Count, FIFO, Return to IDLE	Type 1
2RVS	Count, Reframe, Return	Type 2	Count, Reframe, Return	Type 2

Table 8: Condition handling for the 2-state MBT input state machine.

3.2.2 Error Handling

There are many possible error types which can occur, and as shown in figure 20, these errors can occur in both the DATA and IDLE state. We propose that we not differentiate between these various errors with too much specificity, and will keep track of errors of certain "types". The two types we believe are the most appropriate are:

- Type 1 ("Transmission errors"). Errors which preserve the event size
- Type 2 ("Event errors"). Errors which will result in an incorrect event size, or loss of event boundary.

Type 1 errors are simple bit errors occurring stochastically during transmission, resulting in RVS=HIGH). In addition, receipt of any special character which is not PAD, BOE, or EOE is also an error condition, as is receipt of valid data while in IDLE state. Type 2 errors include reframing conditions, loss of bytes, improper special characters (BOE in DATA state or EOE in IDLE state). If the error results in the receiver chip losing track of the byte boundary, we will attempt to deal with these "framing errors" in such a way that at worst, the entire event will be rejected, as opposed to causing a full reset of the experiment (SCL Initialize). To do this, we note that if the error does not result in a loss of the event boundary, then the ALPHA will be able to recognize a full event which has some damage. On the other hand, should the error cause the ALPHA to get confused as to which buffers belong to which events, the entire system will have to be reset.

In HotLinks terminology, there are only data, special characters, and illegal special characters. Since HotLinks does not have a parity check, reporting of errors is limited to reporting the receipt of either an illegal special character or a framing error. The latter case results in either a loss of phase lock (extremely unlikely) or loss of byte boundary (much more likely). In either case, the error condition is signalled by RVS (received violation symbol) high.

Illegal special characters can result from one or more bit flips in transmission. In case of an illegal special character, the circuitry does not know the difference between whether a bit (or bits, or bytes) has been dropped, or a bit has been flipped – all it knows is that it got an unexpected character. It is possible that such an error will map into a legal data character. This is expected to occur approximately 25% ($2^8/2^{10}$) of the time. If it is the case that there was a simple random bit flip, but synchronization is indeed not lost, then one would expect that subsequent transmission would be alright. We therefore will simply log each RVS in the appropriate counter, and write the received byte into the FIFO as if it was data. This preserves both the total event size as well as the event boundaries. However, if we receive "n" (programmable) consecutive RVS, then this is an indication of an error which could have resulted in a loss of framing, which almost certainly will result in a corruption of the data for the event for this channel. The reframing scheme employed is optimized to be able to recover in time to see the subsequent EOE character, marking the event boundary. The reframing scheme is as follows: (see Figure 21):

1. Drive RF high to re-enable framing circuitry,
2. Reframe upon receipt of PAD characters. Note that the convention for end-of-event transmission calls for 5 or more consecutive PAD characters proceeding the end-of-event character EOE (see table 6). (Note that as described below, the output section and SCL fanout HotLinks will also adhere to this convention.) Reframing should therefore be re-established before EOE is received, thus preserving the end-of-event. The logic detects successful reframing when $\overline{\text{RDY}}$ is low for 2 clock cycles (CKR).
3. Drive RF low to disable framing.

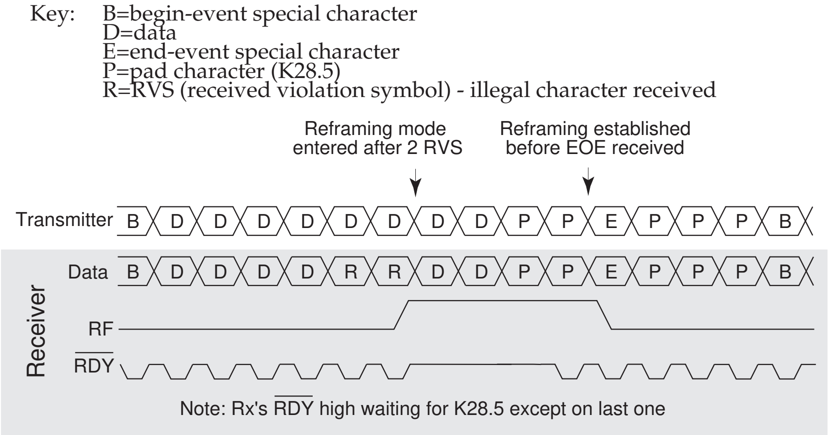

Figure 21: Transmission, receiving, and RF with regards to framing (and reframing).

If the reframing scheme successfully reestablishes synchronization before the transmitter sends the 2 (or greater) PADs followed by the EOE character, then the event boundary will be preserved, and framing errors will not cause a full SCL Init of the system. The existence of a framing error will have to be detected in the ALPHA card by examining the data itself (which will contain necessary header and trailer information about event size, *etc.*). The ALPHA can deal with this error via the Level 2 Global (if the error occurred in the preprocessor crate) or directly inside the L2 Global crate, by rejecting the event (if so desired).

3.3 Channel Event Occupancy

The MBT card will keep track of the number of events in the MBT input buffers. This will be accomplished in the following way. Between the HotLinks receiver and the FIFO is a PLD as described above (point C in figure 18). The job of this PLD is mainly to present 16 bits of data to the correct destination FIFO in the channel FIFO block. This gate array also looks at SC/$\overline{\mathrm{D}}$ (see figure 12) for the special character flag in order to detect the end-of-event (EOE) character. Upon receiving EOE, the PLD will increment a 5-bit occupancy counter which has to allow for 0 to 16 events in the FIFO (see figure 22) which will contain the number of events in the buffer for that channel. This word will be decremented by MAIN after the data for the particular event has been broadcast to the ALPHA (for all channels). The value of this counter is sampled at some predetermined rate. The specific I/O specification is detailed in section 8.

There will be 3 occupancy histograms available. Each histogram will have 17 bins (0 to 16 events), with each bin having a size of 16 bits. These histograms are:

- "Least" channel, defined as the channel with the fewest number of events in its FIFO. This will be implemented using the EV_LOADED counter as described above. Note that if we define the event counter for channel "i" as c_i, then this histogram will be filled with the result of the calculation $\min(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$.

- "Max" channel not including the channel which will contain the SCL L1 data (input channel 0). Using the above notation, this histogram will contain the result of the calculation $\max(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$. Channel 0, which contains the SCL information, is not included in this calculation. At present, we have not decided how to determine the value to be used to histogram, however a sorting algorithm (sorting the individual channel occupancies) can be implemented.

- Specific channel. The channel number will have to be supplied to the MBT by the ALPHA via programmed I/O. It will also be changeable via some kind of front-panel thumbwheel.

At some predetermined programmable rate (say 100 Hz), the MAIN gate array will form the 3 values to be histogrammed ("min","max", and "specific"), and use its value for an offset pointer to increment the correct bin (which can be thought of as a separate counter) in each histogram. The maximum number of bits set aside for each "bin" is calculated as follows. We anticipate that the histograms will be read out by the trigger system at least every 10 minutes. If the "sampling" rate of the 5-bit counter is 100Hz, or once every 0.01 sec, a single histogram bin could have accumulated as many as 600sec/0.01sec=60,000 "hits". We will therefore reserve 16 bits (2 bytes) per histogram bin. This results in 2 bytes $\times$ 17 bins = 34 bytes per channel occupancy histogram, or 102 bytes for the 3 histograms.

A front-panel display that will allow a person to get a quick look at the channel occupancy. We plan to install a set of 5 LEDs, and a thumbwheel for channel selection, so the user can dial up the channel of interest and get a view of the occupancy as presented on the LEDs as a 5-bit word. In addition, an analog output will can be plugged into a scope to view the (thumbwheel selected) histogram directly.

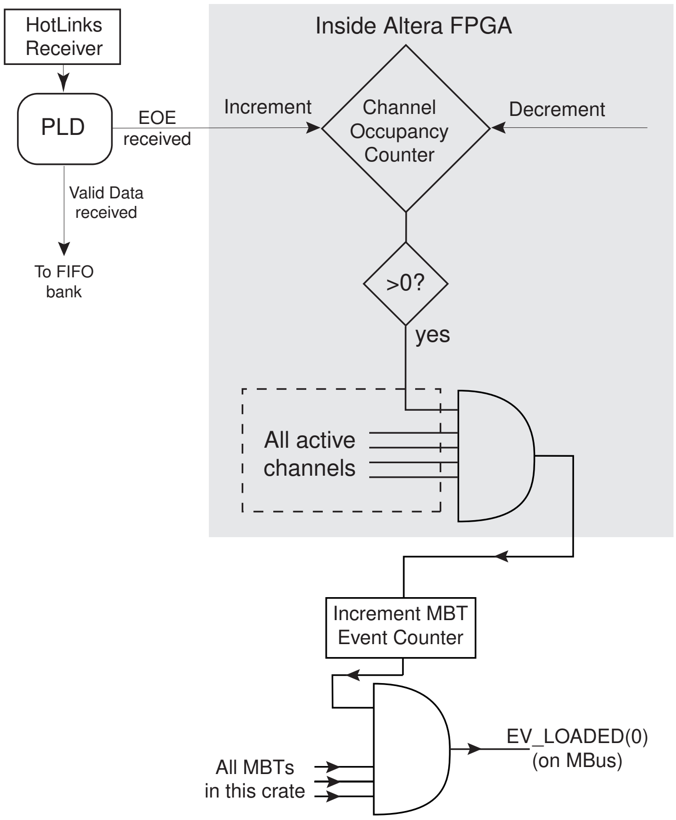

Figure 22: Schematic of how MBT channel occupancy is controlled, and controls EV_LOADED.

3.4 MBus lines used in Receiver Control

Table 2 summarizes the MBus lines. All 128 data lines will be used. Of the 32 address lines, only 10 will be used in broadcast mode (but all 32 can be used for programmed I/O). The remaining 22 address lines are required to be set to zero by the MAIN Altera. The 10 bits of non-zero address correspond to the following bits used as in table 9, and as described below:

9876543210
eeeessssss

Table 9: HMBus Address lines used in Broadcasting.

- Bit 0:5 - "Source ID". A "source" is defined as 1 or more MBT input channels. Each MBT channel will have a unique source ID, as long as there are fewer than 64 channels of MBT input in a crate. Since there are 8 MBT inputs (including the SCL channel 0), that means that each source will be unique as long as there are 8 or fewer MBTs per crate. If there are more than 8, then some source IDs will not be unique, but this is ok as far as the ALPHA DMA architecture is concerned. Source ID's are downloaded into each MBT by the ALPHA at initialization (see section 8).
- Bit 6:9 - "Event ID". The "event" counter is a round-robin 4-bit counter that depicts the "ith" event (between 0 and 15).

MBT broadcasting of data (DMA) into the ALPHAs will use this address, corresponding to a particular channel and particular event. START_LOAD will be controlled by the ALPHA Administrator, used to initiate an event transfer (ready). AP_FIFO_EMPTY is an open collector MBus line, formed from an .AND. of all ALPHAs, and will be used by the Administrator to generate an interrupt when the event has been transferred (all ALPHAs have it). This line comes from signals from each ALPHA, each signal reflecting the state of the DMA FIFO in that particular card. EV_LOADED is also an open collector signal .AND. from the MBT cards, indicating if high that all MBTs have an event ready to go. Its state is controlled by the values of the occupancy counters (each which count the number of complete fragments in the particular channel) for each of the active MBT sources (channels) (see Figure 22). Note that for diagnostic reasons, a VME readable register (see section 8) will contain a mask which will have bit "i" set if the channel has at least 1 fragment in the input buffer. MOD_DONE(n) is set by MBT "n" to signal that MBT "n" has finished its DMA. The following section describes the sequence of events which result in a full event transfer from the MBT into the ALPHA.

4 MBus Broadcasting

The receiver sections on a number of MBT cards will need to be able to work together to minimize the amount of communication required with the Administrator processor during actual event processing. In order to describe this section, we will walk through the receiving and MBus broadcasting of a single event. Note that in each crate, there will be a "Pilot" MBT, and "Assistants", selectable via a switch on the board. A front panel LED will be lit for the PILOT. Differences between the two are detailed in section 12 below.

The entire broadcast cycle for a complete event has two basic parts: 1) broadcasting (DMA) of MBT data into the ALPHA DMA engine, and 2) ALPHAs servicing the broadcast via interrupt routines. Figure 23 shows the sequence for the broadcasting step, and Figure 24 shows the sequence for the interrupt handling. The overall timing diagram for both parts is shown in figure 25. The entire sequence is described below.

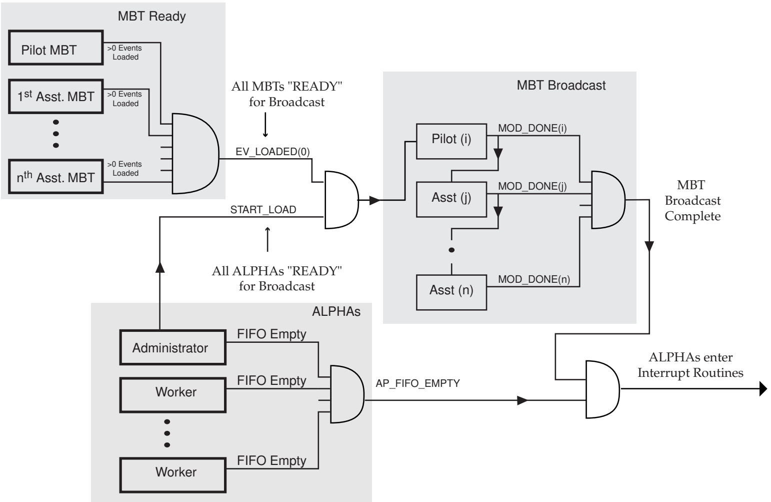

Figure 23: Schematic of the MBT to ALPHA DMA broadcast sequence.

4.1 MBTs' Role in the Broadcasting Sequence

Figure 26 shows the state machine diagram for the MBT broadcast cycle. Each MBT broadcast engine is in the IDLE state, waiting for the enable condition to be satisfied. For the Pilot MBT, enable is satisfied when both of the MBus lines START_LOAD and EV_LOADED(0) are asserted. EV_LOADED(0) is derived from the logical AND of all active MBT having at least one full event in the input FIFOs ready to go (see Figure 22). For the other MBTs in the crate (the Assistants), enable is satisfied for assistant i when START_LOAD and MOD_DONE(*i-1*) are asserted. By *i-1* we mean that the assistants wait until the previous MBT has completed transfer of its event. The first assistant therefore waits for the Pilot, the second assistant waits for the first, *etc.* The Pilot therefore serves the purpose of initiating the event transfer into the ALPHA DMA engine. Note that due to the MBT hardware design, *the particular MOD_DONE line which each Assistant needs for its enable is set via a jumper on the board.*

Upon receipt of the proper enable condition, each MBT then requests access to the MBus. This is done in order to allow the ALPHA to perform programmed I/O to the MBT between MBT source (or channel) broadcasting, When MBus access is granted, the MBT will begin transferring data from the first active channel, and will hold the MBus until all data is sent, after which the MBus is released. The entire channel data consists of the data in the FIFOs, followed by the physical trailer (described below **,but where?**). If all sources' data have been transferred, the machine goes into the DONE state. If there are remaining sources of data, the

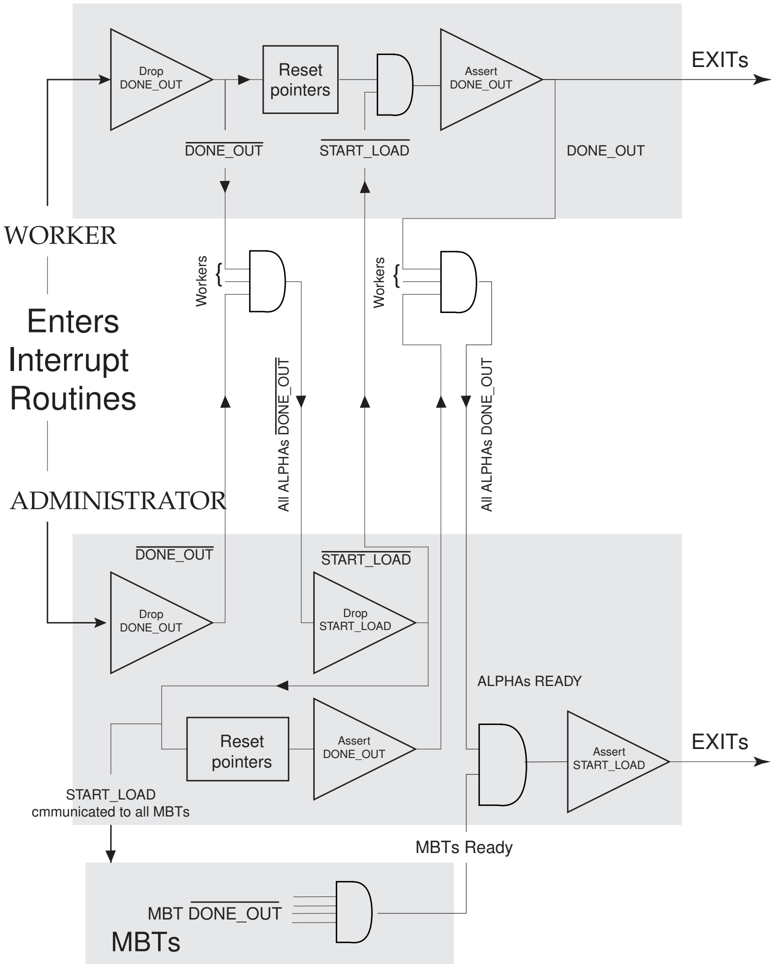

Figure 24: Schematic of what the ALPHA Workers and Administrators do inside the interrupt routine.

transfer cycle is repeated – MBus request, data broadcast followed by the physical trailer, bus release.

The MBT input channel physical trailer will consist of 128 bits, broadcast in a single MBus cycle. This trailer will contain information from previous physical trailers from any device which is upstream of the MBT input, as well as the 8-bit longitudinal parity calculation for the channel's data. **We still need to settle on what bits goes where, but Jim suggests we pack the parity from upstream devices in the lowest order bytes, and put our parity at the top.** This calculation is kept in the MAIN Altera, and is therefore calculated during broadcast (as opposed to during data receiving). The parity is reset after the physical trailer is sent to the ALPHA. **We are also going to send information about errors, but we haven't settled on a scheme for this as yet.**

In the DONE state, the machine for MBT i asserts MOD_DONE(i) via asserting

MBT_DONE = all MBT MOD_DONE lines asserted (all MBTs finished broadcasting)
AP_FIFO_EMPTY = all ALPHA DMA FIFOs empty (drained of broadcasted data)
ALPHA_REC_DONE = MBT_DONE.and.AP_FIFO_EMPTY (all ALPHAs enter interrupt routines)
Pilot_ENABLE = START_LOAD.and.EV_LOADED(0)
Assistant_ENABLE = START_LOAD.and.MOD_DONE(previous)

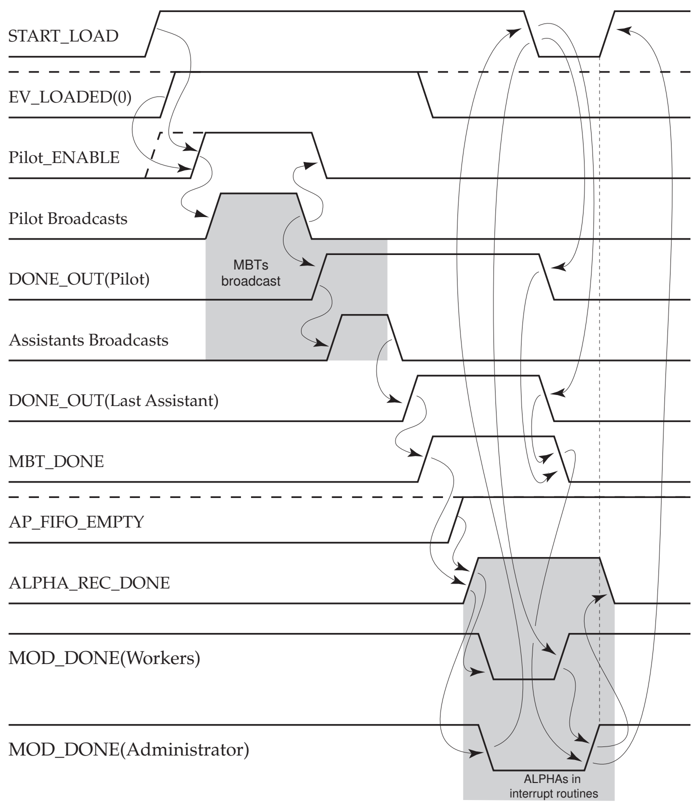

Figure 25: Timing diagram for the entire broadcast sequence.

its DONE_OUT line. If MBT i is the not the last MBT in the broadcast sequence to transmit its data, then by asserting its MOD_DONE line it allows the next MBT to begin broadcasting. The machine in each MBT will then wait for the release of the MBus line START_LOAD. As described below, START_LOAD is released by the Administrator when all MBTs have completed their broadcasts (determined from the set of MOD_DONE bits which are asserted) and all ALPHAs have lowered their MOD_DONE lines, which signifies that the ALPHAs have received the data, moved it into physical memory, and are servicing the broadcast in their respective interrupt routines.

When START_LOAD is released, each MBT goes into the RESET state, where it decrements the local event counter, updates its contribution to EV_LOADED(0) (so that it reflects the presence of a complete event, if one exists), and releases its respective MOD_DONE line. The machine then goes back into the IDLE state.

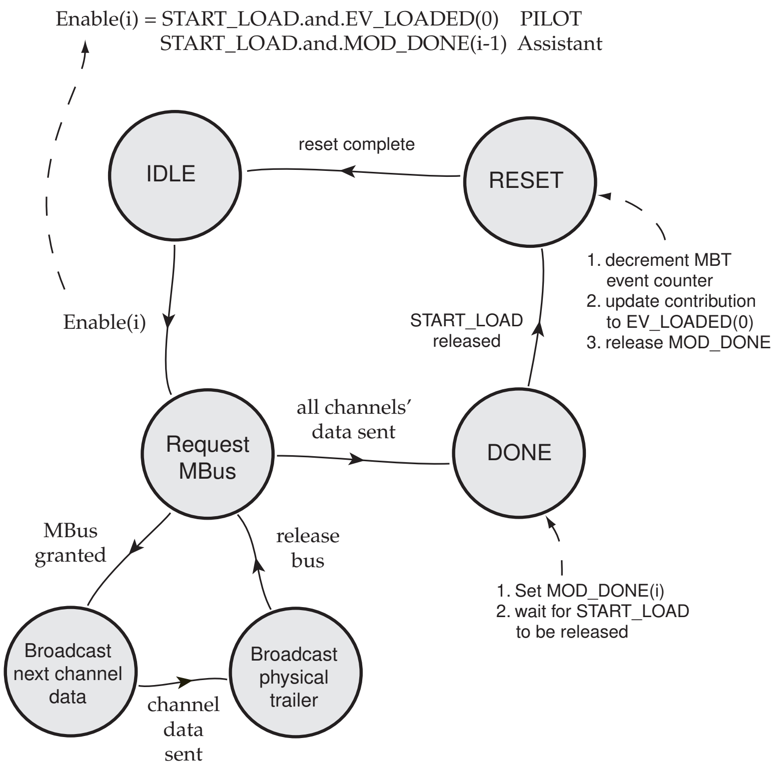

Figure 26: State machine diagram for each MBT broadcast sequence.

4.2 ALPHAs' Role in the Broadcasting Sequence

Broadcasting data from any particular MBT input source into the ALPHA DMA engine is via MBus I/O, using an MBus address constructed from the "Event ID" (a round-robin counter cycling from 0 to 15 and back to 0) in the first 4 bits and the "Source ID" for the next 4 bits. The other 24 MBus address bits are required to be zero. The ALPHA DMA engines service the DMA FIFO (128 bits wide), and ALPHA moves data from the FIFO into ALPHA physical memory using pointers constructed from the MBus address value. The state machine diagram is shown in figure 27.

All ALPHAs must know how many MBTs are in the crate, and which MBus slots they occupy (which MOD_DONE lines are active). This allows the construction of a "MOD_DONE mask" to be used to compare against those MOD_DONE lines which are driven by each MBT upon completion of the broadcast. Satisfaction of the MOD_DONE indicates that all MBTs have finished the transfer of a complete event. Note that the transfer is asynchronous - each MBT has moved its data into the ALPHA DMA engine as fast as it can, and the DMA engine has moved data from its FIFOs into physical memory as fast as it can. The only handshaking done is at the hardware level - there is a handshake on every MBus transfer cycle,

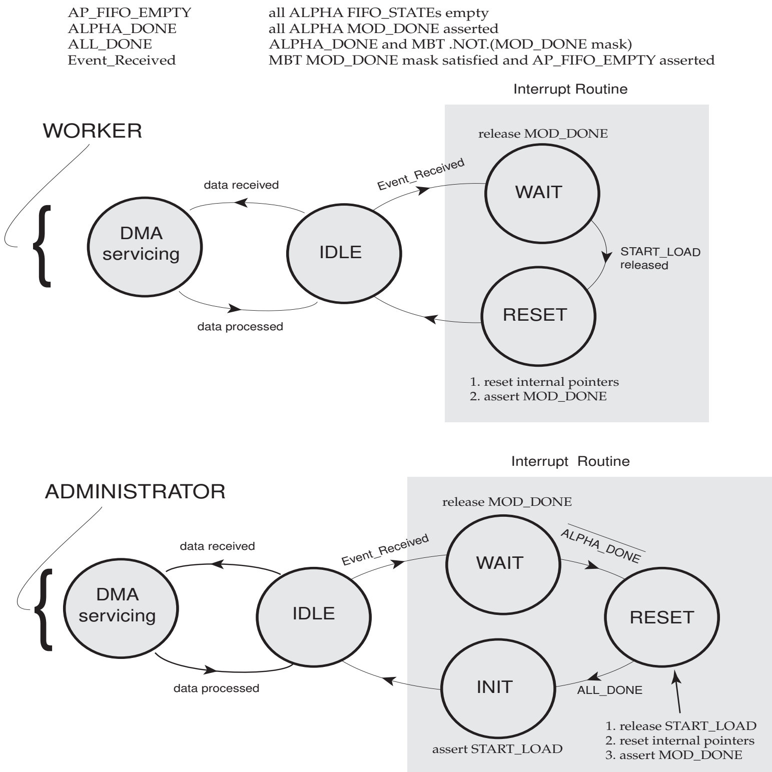

Figure 27: State machine diagram for each ALPHA broadcast sequence.

with an address strobe and a DDONE (see Table 2). There is no direct message passing between ALPHAs, or between ALPHA and MBT, which serve as handshaking communication concerning when the event transfer is finished. The state of each ALPHA's input DMA FIFO is used to determine the state of an internal line, which participates in generating the MBus line AP_FIFO_EMPTY via a logical AND. When each ALPHA finds that the MOD_DONE mask is satisfied (all MBTs have finished), *and* all ALPHA FIFOs are empty, a hardware interrupt is generated over the ALPHA PCI bus.

Inside this interrupt handler, the ALPHA Worker state (labelled RESET in figure 27) drops its DONE_OUT line (causing a particular MOD_DONE line to be released), resets pointers governing where the DMA engine is to put the next event in memory, and raises DONE_OUT. The interrupt routine is then exited.

The Administrator node, however, has more work to do than the Worker. In the Administrator WAIT state, the ALPHA releases its DONE_OUT line causing its MOD_DONE to be released, and waits for the signal that all Worker ALPHAs have entered their interrupt routines. When the proper set of MOD_DONE lines are released, the Administrator goes into the RESET state, where it releases START_LOAD, resets its own internal pointers, and asserts its own DONE_OUT line (contributing to the MOD_DONE mask). When all ALPHAs are done (MOD_DONE lines are all asserted), and all MBTs have released their MOD_DONE lines (all MBTs ready for a next event) the Administrator goes into the INIT state, where it releases START_LOAD, after which the Administrator exits the interrupt routine and enters IDLE.

4.3 Test Data Injection

Test data can be injected directly into any input channel FIFO via VME writes to the appropriate address offset. The following describes the sequence of steps needed. All of these steps are described as well in section 8.

1. Set the MBT mode to "test data injection" via a VME write to address offset 25, with the LSB bit (bit 0) set to 1. When this bit in this register is set, the MBT will set bit 1 to 1 in its state register, readable via VME at address offset 17. In this mode, the MBT will ignore all input coming in over the HotLinks receivers, or in other words, the PLD state machine will ignore the HotLinks receiver output. The MBT will stay in this mode until the ALPHA writes to address offset 25 with bit 0 cleared.

2. In order to write test data into any particular channel, the ALPHA must first specify the channel of interest by a VME write to address offset 14, with the channel number specified in the lowest order 3 bits (0-7). Subsequent injections will be routed into this particular FIFO.

3. In order to specify the beginning of the event, the ALPHA must issue a 9-bit VME write to address offset 28 with the 9th bit (bit 8) set to 1 and with the proper HotLinks BOE special character (K28.0) encoded in bits 0:7. The PLD in the input channel (as specified by the value in register 14) treats this in the same way it would treat a BOE character received by the HotLinks receiver. To specify end of the event, write to the same address offset with bit 8 set to 1 and bits 0:7 set to the HotLinks EOE special character K23.7. This is treated as an EOE character received by the HotLinks receiver. To write data, set bit 8 to 0 and put the data in bits 0:7.

4. When the ALPHA has injected all test data (with appropriate BOE and EOE) into all chosen channels of all MBT cards, a broadcast over MBus can begin. At this point, there is no difference between the data taking and test injection states. If there are complete event fragments in each enabled channel of all MBT cards, the ALPHA can enable the MBus broadcast in the same way as for data. See section 4 and subsections 4.1 and 4.2 above for details.

5 Output Section

As in figure 28, each MBT card has two output channels for transmitting data to another device via HotLinks. These two output channels are each completely separate, controlled by a state machine running in the MAIN FPGA (described below). Note that each output channel will not have an 8-bit state register analogous to that of the input channel, since it is effectively stateless in that it simply processes according to what the ALPHA told it to do, without regard to what came previously.

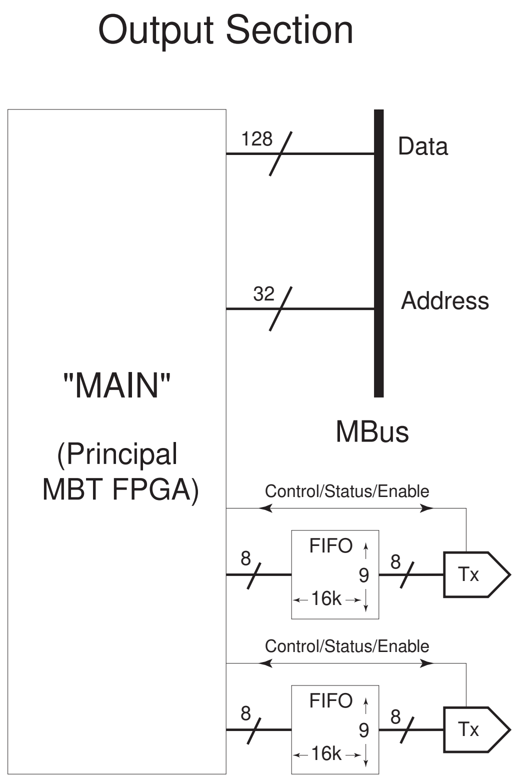

Figure 28: Schematic of the MBT output channel.

Data that is to be sent out from the ALPHAs is received on the MBT from the MBus via programmed digital I/O to a particular MBus address (see section 8 and table 16). This data originates in the ALPHA, travels over PCI to the ALPHA MBus controller, over MBus after bus arbitration, into the MBT MAIN FGPA, into the output channel FIFO, then out the HotLinks output. (As explained above, it is required that the ALPHA can grab the MBus and initiate output to one of the two MBT output channels between MBT broadcasts.) Each output channel will contain its own FIFO which serves to both buffer the data and assure synchronous transfer of the data into the HotLinks transmitter. Since programmed I/O from the ALPHA does not have a burst mode, the ALPHA will have to do wait for bus arbitration between each 16 byte MBus transfer. The current budget for all Level 2 pre- processors is of order a few hundred bytes, which means that the MBT output FIFO need not be very large. However, since FIFOs are relatively cheap, we will put a 16kByte FIFO (9bits wide, 16k deep) on the output so that any upgrade (*e.g.* enabling burst mode programmed I/O coupled with an increase in the byte budget) path is easily taken.

5.1 Output State Machines

The state machine running in MAIN is shown in figure 30. Programmed I/O from the ALPHA (see section 8) informs this state machine about which channel is selected and whether the operation is to be a begin signal, end signal, or data to be sent out the HotLinks transmitter. The special characters BOE and EOE are sent whenever the ALPHA writes to the appropriate address offsets corresponding to the output control address (see section 8 item list) using bit 0 to select whether the operation is begin or end. Data is sent whenever the ALPHA sends 16 bytes of output to the appropriate address.

Figure 29 shows the sequence of characters which constitute a single "event", put into the FIFO by the MAIN FPGA according to what was sent to it by the ALPHA over MBus. The 9th bit in each the FIFO slot will be used to control SC/$\overline{\text{D}}$ on the transmitter to distinguish between sending data or special characters. The first character will be BOE, with bit 9=1, followed by data (bit9=0), followed by 5 PAD characters (bit9=1), a single EOE (bit9=1), and 5 more PADs (bit9=1). PAD characters can be automatically clocked onto the UTP output cable by the HotLinks transmitter by setting $\overline{\text{ENA}}$ into the "false" state ($\overline{\text{ENA}}$=HIGH) for 5 CKW cycles, however for reasons of control, we prefer to put the PAD bit sequence into the FIFO by hand and set bit9=1.

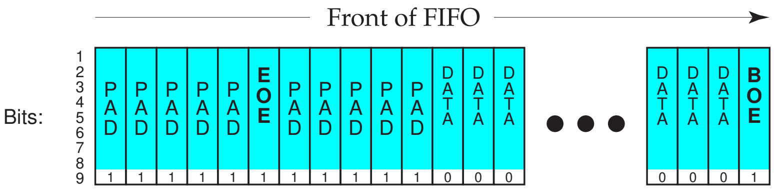

Figure 29: Sequence of characters packed into the MBT output channel FIFO for a single output "event".

The state machine has very little to do during normal running. The three messages (BOE, Data, and EOE), are described next:

- **BOE** – The state machine merely enables the correct transmitter (channel 0 or channel 1) and loads the correct BOE character into the channel FIFO. BOE is automatically sent.

- **Data** – The correct transmitter enable line is asserted, and the data is transferred from the MBus data lines (MBDATA) straight into the FIFO, keeping care to update the 8-bit longitudinal parity calculation along the way. Since the programmed I/O rate into the MBT is similar to the HotLinks transmitter speed, the output FIFO will almost certainly spend most of its time empty. However, in case of an upgrade which consists of a burst mode in programmed I/O, a mechanism to throttle back the MBus transfers exists for this state machine to implement. It can simply hold off on the MBus DDONE line after any 16 byte transfer if the FIFO half-full line indicates that the FIFO is half full (or 1/3, 1/4, there are often many of these lines). For the next run (Run II), we expect not to have to use this path (hence we will follow the dashed line in figure 30.

- **EOE** – For EOE, the state machine has to enable the correct transmitter. It then has to build the physical trailer, which consists of 16 bytes, with the first byte consisting of the longitudinal parity calculated from the output data, one byte at a time. The physical trailer is placed into the FIFO and automatically sent as data (bit9=0), and the parity reset. At this point, 5 PAD characters are inserted with bit 9=1. The EOE character is then inserted, followed by 5 more PADs, all with bit 9=1 to signal special character.

5.2 HotLinks Transmitter Control

Figure 12 shows the HotLinks transmitter. Table 10 shows the various lines, and how they will be configured. The MODE line will be set LOW for encoded mode, FOTO will be grounded as it is not used. SVS is normally LOW, but can be driven high if one wants to test the receivers RVS function by sending a violation symbol. SC/$\overline{\mathrm{D}}$ tells the transmitter whether the byte presented for output should be encoded as data or a special character, and $\overline{\mathrm{BISTEN}}$ is used to put the transmitter into BIST mode **(see below)**. We will be driving the OUTA lines, and for noise purposes will tie OUTB and OUTC to +5V.

The input clock (16 MHz) is provided by the MBT on the CKW line (the transmitter derives a x10 clock from CKW for bit transfers) and controls all byte-wise timing on the transmitter chip. The transmitter uses this clock to derive the output pulse $\overline{\mathrm{RP}}$ (Read Pulse). This pulse signals that the chip is ready for another byte, and can be used as the enable into a FIFO (either asynchronous or clocked). We will use the enable line $\overline{\mathrm{ENA}}$ (Enable Parallel Data), which causes the transmitter to load, encode, and send the next byte on the rising edge of CKW. ($\overline{\mathrm{ENN}}$ is the virtually same, with slight differences in timing, and will not be used.) Setting $\overline{\mathrm{ENA}}$ high causes the transmitter to ignore the data inputs and send the PAD character.

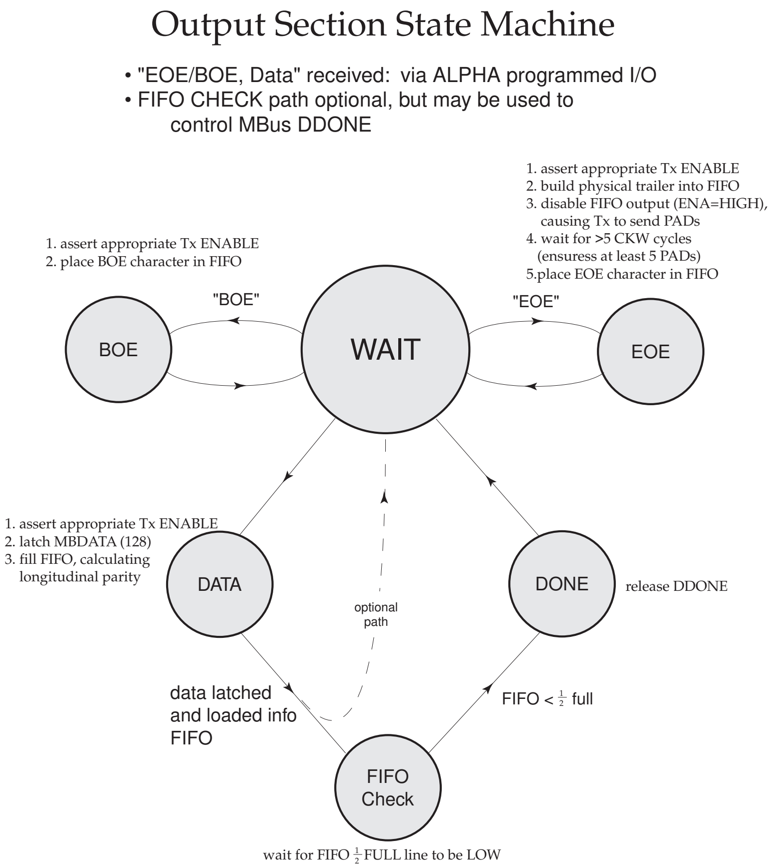

Figure 30: Schematic of the MBT output channel state machine.

This will be the mechanism whereby we can force PADs to be send (*e.g.* before sending EOE).

6 L2 Global HWFW Communication

A simple parallel digital I/O function is required for sending the answer to the L2 Hardware Framework (L2HWFW), and can also be used to get state monitoring control lines into a logic analyzer or scalers. The output will consists of the following lines on a 40-pin front panel connector: 16 PECL lines of data, with each data line containing as much as one byte worth (multiplexed by 8) of information. The transmission rate will be approximately 40MHz, or 25ns, resulting in the full information being sent to the L2HWFW in of order 200ns. There will also be 3 address lines used to signal which multiplex slice is being sent, a strobe line indicating information is present, and a done line to indicate information has been received by the DeMux card (see below). Note that if some kind of error in transmission causes the L2HWFW to miss the transmission, ther eis no way to report this back to the

Name		Configured	Description
MODE	In	LOW	Encoded Mode
CKW	In		16 MHz clock
SC/$\overline{\mathrm{D}}$	In		HIGH=data, LOW=special character
SVS	In		LOW=normal, HIGH=send violation symbol
$\overline{\mathrm{ENA}}$	In		Enable for send
$\overline{\mathrm{ENN}}$	In	+5V	Not Used
D_{0-7}	In		8-bit data to transmit
FOTO	In	GND	Not Used
$\overline{\mathrm{BISTEN}}$	In		HIGH=normal, LOW=BIST mode
$\overline{\mathrm{RP}}$	Out		Ready for next byte
OUTA			Serial PECL output lines
OUTB		+5V	Not Used
OUTC		+5V	Not Used

Table 10: HotLinks transmitter pins.

MBT. In such a case, the error will have to be detected via some other route (than via the MBT and Level 2 crate), with an SCL reset.

On each MBT card, output to these 128 lines of Parallel Digital I/O will set by an MBus write. We expect the latency to be of the order of a microsecond. To send 128 bits, write to address 30 (command 30). Reading the value in the 128 bit register is accomplished via an MBus read the same address. Data will be strobed onto the output line as soon as it is received. If it is necessary that we be able to send 128 1's or 128 0s, the ALPHA will have to send them explicitly.

In order to save space on the MBT card, the 128 bits will be multiplexed into 16 PECL lines. A demultiplexing (DeMux) card will be made available to the L2HWFW. Since only 16 pairs of data lines leave the MBT, it might be useful to use those 16 lines ending 16 bits (which for some of the non-global crate MBTs for debugging purposes. These lines will therefore be usable in 16-bit mode but only as levels, not pulses, via issuing a VME (not MBus) write to address 31 (command 31). Note that this involves another line to the DeMux card, informing it of whether it is in 16-level or 128-bit mode. Since the output is 16 pair of PECL levels, we will build a very simple 16 LED card that plugs into this cable. The PECL MBT drivers can drive the LEDs. This might be a very useful tool in debugging and commissioning. Also, in case it is needed, the strobe lines will be active when the output lines (in 16 bit mode) change state.

6.1 DeMux Card

A single demultiplexing (DeMux) card will be used by the online system to take the L2 GLOBAL HWFW information from the Global ALPHA to the framework. We will build a minimum of these cards.

As described, 16 PECL lines with 16 bits/line at 25 ns between bits will contain the Level 2 Global's "answer". These lines will go into a 6U VME card which uses

VME for power only, through the P2 VME backplane. An advantage to a 6U VME format is realized if some other use is found for digital outputs from then MBT, which will probably be the case at least for debugging and commissioning. For the currently planned use connecting to the Framework we could use it either way.

The output will consist of 8 34-pin standard 17 pair ribbon cable which carry 16 signals on each cable, standard differential ECL with the pull down current supplied from the DeMux card. By using ECL, the ribbon cables can be plugged directly into the L2 Framework or directly into standard scaler cards if people end up wanting to scale things via this path. (Note that the MBT will send PECL, the DeMux card will send ECL.)

A strobe indicating new valid data is present on one of the 128 signal lines. This strobe will be generated locally on the DeMux card after it has finished demuxing the final chunk of data in a full 128 bit transfer. This strobe will be present on the 17th pair of all 8 ribbon connectors, for convenience. There are, at present, no requirements for any handshaking from the DeMux to the MBT, nor is there any requirement for error checking.

7 Serial Command Link Receiver Section

The MBT pilot will use a standard SCL mezzanine card to receive information from the SCL Hub. The information which the MBT will be interested is: the L1 accept information, the L2 Decision information, and the SCL Initialize messages from the SCL data stream. The mezzanine card plugs into the MBT board using two 64-pin SMT connectors. 128 bits of information is clocked into this card every 132ns, appearing on the two 64-pin connectors. Signal functions on these connectors are described in table 11, and pin assignments in figure 31. Items in this list which are marked are differential RS485 (differential) signals that go from the mezzanine card to the SCL Hub – all other signals originate at the HUB and are presented on these pins to the host (the MBT). The line CLK_7 (128 ns clock) indicates new information is present, SCL_READY indicates that the mezzanine card is in synch with the HUB for this transmission, and SCL_SYNCERROR indicates that at some point, a loss of phase locking occurred (but could have been subsequently resynched). Each SCL_SYNCERROR will be counted in a counter which can be read over VME using 09 (see section 8 above), and will be reset after each read.

Table 12 summarizes input lines used in the state machine which monitors the SCL mezzanine card input. Table 13 summarizes which input lines determine if the cycle contains Level 1 or Level 2 information (ECB requires that there will never be both Level 1 and Level 2 information advertised in the same 128 ns) as well as which bytes which will be saved for each. To be clear, the information presented by the MBT will always be referring to the original level 1 crossing ID of the event, and **not** the current event under “discussion”.

Signal Name	Needed For	Description
SCL_READY	Data Valid	SCL Ready Status
INIT_SECTION	SCL init	Initialize Geo Section Flag
SCL_SYNCERROR		SCL Synch Error
SCL_ACK	**not sure**	Acknowledge & Clear SCL Error Flags
CLK_53		53MHz Clock
CLK_7	New Data	1/132ns = 7.5MHz Clock
CURRENT_TURN[0:15]		Current Turn Number
CURRENT_BX[0:7]		Current BX Number this Turn
FIRST_PERIOD		1st Period in a Turn Marker
BEAM_PERIOD		Period with Beam Marker
SYNC_GAP		Sync Gap Marker (no L1 Accepts)
COSMIC_GAP		Cosmic Gap Marker (only Cosmic L1 Accepts)
SPARE_PERIOD		Spare Period Marker
L1_PERIOD	L1 info present	Period with L1 Accept Issued
L1_ACCEPT		L1 Accept to this Geo Section
L1_TURN[0:15]	L1/L2 info	Level 1 Turn Number
L1_BX[0:7]	L1/L2 info	Level 1 BX Number this Turn
L1_QUAL[0:15]	L1/L2 info	L1 Accept Qualifiers
L2_PERIOD	L2 info present	Period with L2 Decision Issued
L2_REJECT	L2 info	This Geo Section L2 Reject
L2_ACCEPT	L2 info	This Geo Section L2 Accept
L1_BUSY	set/cleared	Busy L1 Status
L2_BUSY		Busy L2 Status
L1_ERROR	set/cleared	Error L1 Flag
L2_ERROR	set/cleared	Error L2 Flag
INIT_ACK	MBT for init	Init acknowledge signal to Hub
SYNC_LOST	counted	SCL Receiver Synch Lost
SPARE_STATUS[0:1]		Spare Status signals to Hub

Table 11: SCL Mezzanine card I/O connections.

Figure 32 shows the architecture employed for handling the SCL mezzanine card. Input from the SCL Hub enters the mezzanine card via the front panel, is parallelized and presented onto the pins as in figure 31 by an onboard FPGA. A state machine running in the PLD on the MBT responds accordingly. Level 1 and Level 2 information is separated, FIFO'd, and in the case of Level 1 sent out via a HotLinks transmitter. Figure 33 contains the state diagram for the machine which responds to SCL input.

7.1 SCL Level 1 Information

All of the ALPHA processors in a crate need to receive the L1 Accept information. It is the L1 Accept information that contains the event's "Geographic Section L1 Event Number" and the L1 Qualifier Mask that indicates whether or not a given event is a "Normal", "Monitor", "Unbiased Sample", or "Forced Write" event, or any other qualifiers defined for particular level 2 processors. For each event, the ALPHA Processors need to receive this L1 Accept information in a way that associates it with the rest of the input data that these processors receive for the event. Thus this L1 Accept information is FIFO'd and then broadcast to the Alpha Pro-

Pin	P1 <- Connectors ->	P2
1	+5V	+5V
2	+5V	+5V
3	SCL_READY	INIT_SECTION
4	SCL_SYNCERROR	SCL_ACK
5	GND	GND
6	CURRENT_TURN[0]	L1_TURN[0]
7	CURRENT_TURN[1]	L1_TURN[1]
8	CURRENT_TURN[2]	L1_TURN[2]
9	CURRENT_TURN[3]	L1_TURN[3]
10	GND	GND
11	CURRENT_TURN[4]	L1_TURN[4]
12	CURRENT_TURN[5]	L1_TURN[5]
13	GND	GND
14	CURRENT_TURN[6]	L1_TURN[6]
15	CURRENT_TURN[7]	L1_TURN[7]
16	CURRENT_TURN[8]	L1_TURN[8]
17	CURRENT_TURN[9]	L1_TURN[9]
18	GND	GND
19	CURRENT_TURN[10]	L1_TURN[10]
20	CURRENT_TURN[11]	L1_TURN[11]
21	GND	GND
22	CURRENT_TURN[12]	L1_TURN[12]
23	CURRENT_TURN[13	L1_TURN[13]
24	CURRENT_TURN[14]	L1_TURN[14]
25	CURRENT_TURN[15]	L1_TURN[15]
26	GND	GND
27	CURRENT_BX[0]	L1_BX[0]
28	CURRENT_BX[1]	L1_BX[1]
29	GND	GND
30	CURRENT_BX[2]	L1_BX[2]
31	CURRENT_BX[3]	L1_BX[3]
32	CURRENT_BX[4]	L1_BX[4]
33	CURRENT_BX[5]	L1_BX[5]
34	GND	GND
35	CURRENT_BX[6]	L1_BX[6]
36	CURRENT_BX[7]	L1_BX[7]
37	GND	GND
38	FIRST_PERIOD	L1_QUAL[0]
39	BEAM_PERIOD	L1_QUAL[1]
40	L1_PERIOD	L1_QUAL[2]
41	L1_ACCEPT	L1_QUAL[3]
42	GND	GND
43	SPARE_PERIOD	L1_QUAL[4]
44	L2_PERIOD	L1_QUAL[5]
45	GND	GND
46	L2_REJECT	L1_QUAL[6]
47	L2_ACCEPT	L1_QUAL[7]
48	SCL_DATAERROR	L1_QUAL[8]
49	reserved	L1_QUAL[9]
50	GND	GND
51	• L1_BUSY	L1_QUAL[10]
52	• L1_ERROR	L1_QUAL[11]
53	GND	GND
54	• L2_BUSY	L1_QUAL[12]
55	• L2_ERROR	L1_QUAL[13]
56	• INIT_ACK	L1_QUAL[14]
57	SYNC_LOST	L1_QUAL[15]
58	GND	GND
59	SPARE_STATUS[0]	SYNC_GAP
60	SPARE_STATUS[1]	COSMIC_GAP
61	GND	GND
62	GND	GND
63	GND	GND
64	CLK_53	CLK_7

Figure 31: SCL mezzanine card output pins.

cessors by circuitry that is identical to that used by the Receiver Section channels, appearing in MBT Channel 0. The information for this channel is also immediately sent to a 3rd HotLinks transmitter for SCL fanout purposes.

The MBT will also have to add a standard header and trailer, but I don't know what these are yet.... It could be just header, trailer, and no data since the headers/trailers for each event in each channel should contains this very SCL L1 information.

The MBT will monitor the 128 bits of SCL information, and will key on the indicator L1_ACCEPT. If this bit is asserted, the MBT will then latch the 3 signals L1_TURN (16 bits), L1_BX (8 bits), and L1_QUAL (16 bits). These signals provide information as to the Level 1 event number as well as qualifiers used by the ALPHA for various data acquisition states such as data, monitoring, *etc.*. The MBT will use these 5 bytes to constuct a standard piece of a Level 1 event by either constructing a template header and trailer, or using these 5 bytes to make up the header and

SCL Line	Purpose
CLK_7	new ready for latching
SCL_READY	information is valid
SCL_SYNCERROR	loss of phase lock synch

Table 12: SCL mezzanine output lines needed by MBT.

	Level 1		Level 2	
	Data	Bytes	Data	Bytes
Tag Bit	L1_PERIOD		L2_PERIOD	
Buffer:	L1_BX	1	L1_BX	1
	L1_TURN	2	L1_TURN	2
	L1_QUAL	2	L1_QUAL	2
			Acc/Rej	1
Total Bytes		5		6

Table 13: SCL Level 1 and Level 2 information needed by MBT.

trailer (resulting in a piece of an event with no data), adding additional bytes to pad to 16 bytes total (if needed), and setting the 9th bit in the last word as the EOE marker, place these bytes into the Level 1 input FIFO as in figure 32. This FIFO constitutes the channel 0 input FIFO, and is read out as an additional channel just like the other input channels (see section 3.2). Since the data for this channel will be of known length, and small, the input FIFO need only be large enough to hold 16 events worth of Level 1 data. We have determined that a 1kByte FIFO has more than enough room for this purpose.

7.2 SCL Level 2 Information

The handling of the L2 Decision information is different because only the Administrator Processor needs access to this data, since the L2 decision refers to data that has already been processed. The L2 Decision information is flagged on the SCL input by the bit L2_PERIOD, and monitored by the PLD (see figure 32). If this bit is set, the PLD will FIFO the same 5 bytes as it does for the Level 1 information (2 bytes for L1_TURN, 1 byte forL1_BX byte, and 2 bytes for L1_QUAL, which in this case will be the geographical section Level 3 transfer number), as well as a single byte which will contain all zeros if the (SCL) L2_REJECT bit was set, or all 1s if the L2_ACCEPT bit was set (or something similar to this, possibly determined later), constituting 6 bytes total. The 6 bytes a FIFO which is 1 kByte deep, enough to hold 133 events, much more than the canonical 16 events. A counter (resident inside the MAIN FPGA) indicating the number of SCL L2 events present in the FIFO will be incremented, and the MBT will then inform the ALPHA that there is new SCL Level 2 information is present for readout. There are two possibilities for informing

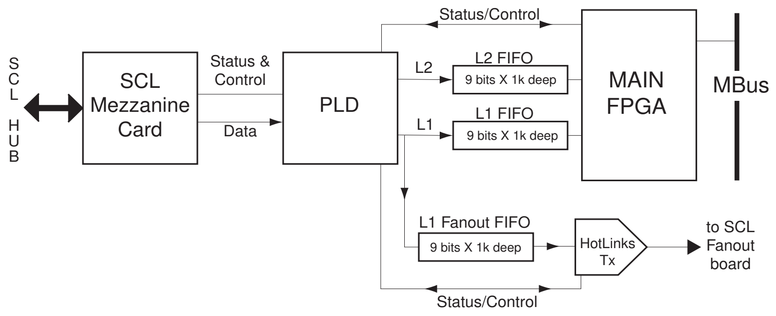

Figure 32: Schematic of the SCL section of the MBT.

the alpha: 1) by asserting the MBus line EV_LOADED(1) (see table 2) or 2) by asserting a line on the J2 backplane connected to a pin **to be determined**. On receiving the signal that Level 2 information is present, the ALPHA will read the information over MBus using register 06 as in table 16. The information sent over MBus will have the format: bits 0-15 is the level 1 turn number (L1_TURN), bits 16-23 is the level 1 beam crossing number (L1_BX), bits 24-39 will be the transfer number (L1_QUAL), and bits 40-47 will be either all 0 for reject or all 1 for accept. After the data is read, the MBT will release EV_LOADED(1) or the P2 line, decrement the counter, and reassert EV_LOADED(1) or the P2 line if the counter is not zero, indicating another Level 2 decision is present. The value of the counter will not be readable, but will remain internal.

7.3 SCL Init and other SCL Output to HUB

The MBT Pilot receives the SCL Initialize messages from the SCL data stream, and transmits any SCL Initialize request to the Hub, all via the SCL mezzanine card output lines. These lines are shown in table 14.

SCL Output	Description
L1_BUSY	Halt/Resume L1 Accepts
L2_BUSY	Halt/Resume L2 Accepts
L1_ERROR	Request SCL Init
L2_ERROR	Request SCL Init
INIT_ACK	SCL Init Received/Complete

Table 14: SCL output lines.

The SCL output lines L1_BUSY, L2_BUSY, L1_ERROR, and L2_ERROR will reflect the state of the SCL control register (SCLCR), which changes according to the VME programmed I/O write to address 07-10 (see section 8 and table 16). However, of these 4 lines, L1_BUSY is also controlled by the state of the input line into P2 (A1) which can be driven by any external card (most likely only the FIC

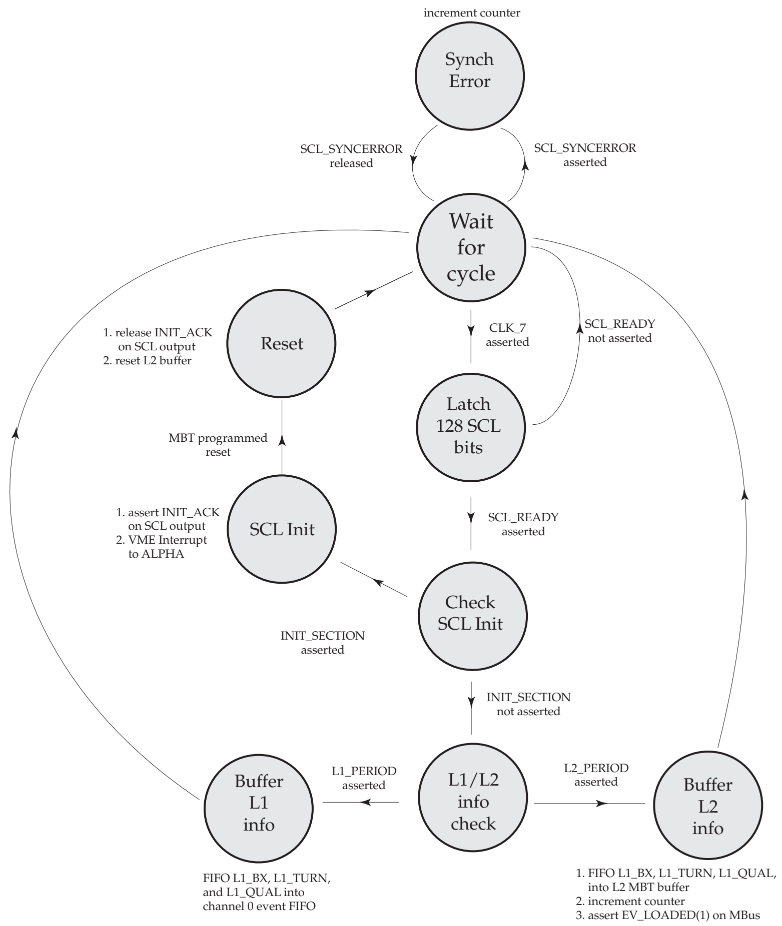

Figure 33: State machine which monitors and responds to SCL mezzanine card input.

card). The control register SCLCR can be read via a VME read to address 11. Table 15 details what bits in the SCLCR controls what output lines on the SCL output. There is currently no plan to detect L1 Busy in either the L2 Global or any of the preprocessor crates. This is because there is no way for these systems to actually know in realtime how many events are in their various buffers and there is nothing that they can do to stop an overflow once it would begin to occur. However these L2 system can indicate L1 Busy as they are required to during the SCL Initialization process also via command 07.

Any Level 1 or Level 2 error which results in a loss of synchronization (or which requires an initialization of the entire DØ trigger and DAQ system) will result in an SCL Init. As just discussed, these requests can be initiated by the ALPHA Administrator in any Level 2 crate via programmed I/O to the Pilot MBT, which will result in assertion of either L1_ERROR or L2_ERROR lines on the SCL mezzanine card. Specifically, to set (clear) L1_ERROR, the ALPHA must issue a VME write

SCL Output	Controlled by...	Causes...	
L1_BUSY	SCLCR.OR."P2"	Set:	Halt L1 Accepts
	$\overline{\text{SCLCR}}$.AND.$\overline{\text{"P2"}}$	Clear:	Resume L1 Accepts
L2_BUSY	SCLCR	Set:	Halt L2 Accepts
	SCLCR	Clear:	Resume L2 Accepts
L1_ERROR	SCLCR	Set:	Request SCL Init
	SCLCR	Clear:	no meaning
L2_ERROR	SCLCR	Set:	Request SCL Init
	SCLCR	Clear:	no meaning
INIT_ACK	Internal to Pilot MBT	Set:	SCL Init Received
		Clear:	SCL Init Complete

Table 15: Control of the SCL output lines. SCLCR is the SCL Control Register.

to command 08 (command 10 for L2_ERROR), using bit 0 (LSB) to set (1) or clear (0). Note that we have to prepare each MBT Pilot for SCL Inits which did, and did not, originate within its crate.

The sequence of events occurring during the SCL Init (from the point of view of the SCL mezzanine card controller) is shown in figure 34, and is itemized as follows:

- When L1_ERROR or L2_ERROR is asserted on the SCL mezzanine output, the HUB will process the request, and if appropriate, issue a global SCL Init via the SCL input back to the Pilot MBT through the SCL mezzanine card.

- The Pilot MBT will see any SCL Init request from the HUB by monitoring the appropriate bit on the mezzanine card (INIT_SECTION). If this bit is set, the Pilot MBT will inform the ALPHA Administrator that the SCL_INIT has been requested. There are two possibilities for how this information can be passed to the ALPHA: 1) via a VME interrupt (either level 6 or 7, to be determined) which will be wired on the ALPHA to the SYS_FAIL input in order to avoid confirmation) or 2) via asserting a line connected to one of the pins on P2 (to be determined). Presently we have decided to implement the latter - a single P2 TTL level will be asserted, and this line has to be wired directly to the ALPHA Administorator. The Pilot MBT will immediately assert INIT_ACK on the SCL mezzanine output, telling the HUB it has the request and is dealing with it.

- When the ALPHA begins the SCL init for the crate, it informs the each MBT in the crate via an MBus write to address 12 (SCL Init) asserting bit 0. This causes each MBT to perform SCL Reset (containing a subset of all possible resets, as described in 9.3), release the P2 line whichever was being used, clear (if set) the L1_ERROR or L2_ERROR bits in the SCLCR (which will cause the corresponding SCL mezzanine output lines to be released), and clear the MBT State register READY bit (see section 2. Each MBT will then place all active input channels into the QUIET state to wait for all transmitters

(MBT inputs) be cease transmitting data before resuming normal operation. The QUIET state is a very simple state – all 4 error counters are cleared, and any non-PAD character appearing at the HotLinks output will cause one of the error counters to be incremented. The ALPHA must poll each MBT active channel error counters via a VME write to address 14 (to select the channel) and a VME read to address 16 (see section 8) to determine if the channel is indeed quiet, and each time the channel's error counter is read, it is immediately cleared. After several pollings which result in error counters which are clear, (the number to be determined by experience), the channel can be considered quiet. After all channels are quiet, the ALPHA can check that particular MBT card as being ready for the completion of the SCL Init.

- When the ALPHA determines that the SCL Init is completed in the crate, the ALPHA informs each MBT using the same MBus write to address 12 with bit 0 cleared. Each MBT will put all active input channels into the IDLE state, and set the READY bit in the MBT state register. The Pilot MBT then releases INIT_ACK on the SCL mezzanine card output, informing the HUB that the init has been successfully processed.

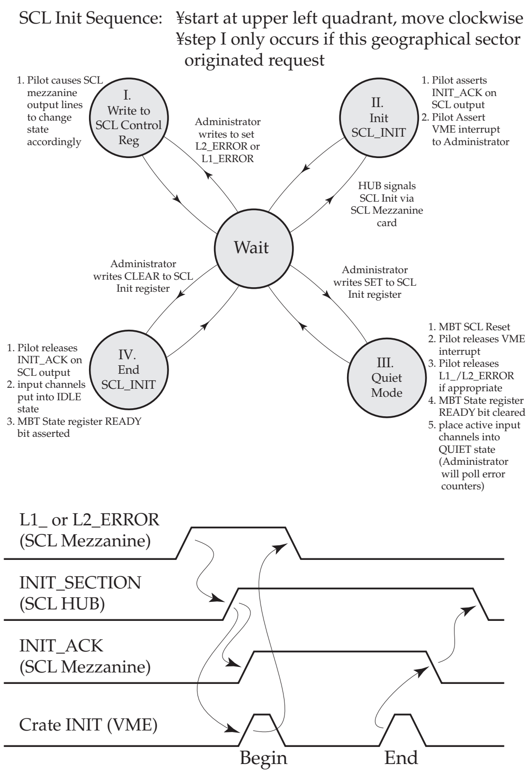

Figure 34: SCL Init state machine and timing.

8 Programmed I/O from the ALPHA to the MBT

Programmed I/O, as explained above, involves the same sequence as in broadcasting: assert data lines, assert address, assert DSTROBE, wait for DDONE (data latched) which releases DSTROBE and DDONE. The main differences in how we will use the MBus are:

- Broadcasts will be an MBT write of 128 data bits, however Programmed I/O will be an ALPHA read or write with variable numbers of data bits, and
- Broadcasts will use an 8 bit MBus address, with the other 24 bits set to zero. Programmed I/O will cause addresses (as offsets to the geographical sector, or MBus slot) to be mapped into commands.

Table 16 lists the address offsets, function, whether a read or write (from the point of view of the ALPHA), whether the operation is via VME or MBus, and the data width of all programmed I/O. The policy implemented is that all communication between the ALPHA and the MBT which occurs in real time (during Level 2 decision making) will be via MBus, while all other communication (*e.g.* monitoring, setup, initialization) will be via VME.

Each of the above items are described next. Note that in both VME and MBus, programmed I/O can be either a read or write operation controlled by asserting the proper R/W lines. We can therefore use a single command address for both a write (*e.g.* setting a parameter) and a read (*e.g.* reading that parameter back). Note however that each address will involve either VME or MBus, but not both. In the following, itemization numbers are the register offsets, in decimal.

01 **Programmed Reset** – A write to address 01 causes a general MBT reset (Programmed Reset) and is described in section 9.2 below.

02 **Active Channels** – The setting (reading back) of the specific MBT input channel occupancy to histogram is via a VME write (read) to address 04. Note that the value is encoded in the lowest order 8 bits, forming a bit mask – setting bit "n" will select channel "n". The same applies to the read.

03 **Sampling Frequency** – The sampling frequency is set by specifying a countdown counter value. The MBT will maintain an internal 50ns clock. The sampling will occur when the countdown counter, counting internal 200ns ticks, is zero. A default value of 50,000 decimal (C350 hex) will result in a sampling frequency of 100Hz.

04 **Specific Channel Histogram specification** – As described elsewhere, each MBT will make among other things an occupancy histogram for any specified channel. To set this channel number, the ALPHA will issue a VME write to

Address Code (Decimal)	Function	Bus	Width	Comments
01	Programmed Reset	VME W	0	Reset MBT Card
02	Set Active Channels	VME R/W	8	Bit mask, 0-7≡ on/off for Ch 0-7
03	Sampling Freq	VME R/W	32	Set the histogram sampling rate
04	Specific Channel	VME R/W	3	Channel number for specific histogram
05	BIST mode	VME R/W	1	0=off, 1=on
06	Rd SCL L2 data	MBus R	25	SCL L2 data read: L1_TURN (0:15), L1_BX(16:23), L2_ACCEPT(24)
07	L1_BUSY	VME W	1	0=clear, 1=set
08	L1_ERROR	VME W	1	0=clear, 1=set
09	L2_BUSY	VME W	1	0=clear, 1=set
10	L2_ERROR	VME W	1	0=clear, 1=set
11	SCL Control Reg	VME R	4	bit 0:3=L1_BUSY:L1_ERROR:L2_BUSY: L2_ERROR – 0=on, 1=off
12	SCL Reset	VME W	1	bit 0, 1=set, 0=clear
13	SCL Sync Err Cntr	VME R	8	SCL Link Errors
14	Input Channel Pntr	VME R/W	3	channel number (0-7)
15	Input Chan State Reg.	VME R	8	see section 3.1.4
16	Channel Error Cntr	VME R	32	see text
17	MBT State Register	VME R	1	see table 3
18	MOD_DONE Mask	VME R/W	32	Pointer used for
19	Input Ch Source ID	VME R/W	6	channel source ID broadcast enable
20	Reframe Parameter	VME R/W	8	Value for consec. reframe
21	L2 Data Type	VME R/W	8	Value for L1 SCL chunk header
22	MAIN Firmware Version	VME R	8	For MAIN Altera
23	Input Channel Fragments	VME R	8	Mask of ≥ 1 fragments
25	Test Data Inject Mode	VME W	1	set/clear bit 0=test/data
28	Inject Data	VME W	10	data for injection
30	L2 Global Result 128	MBus R/W	128	Output 128 bits to FW
31	L2 Global Result 16	VME R/W	16	Output 16 bits to FW
33	Output Ch1	MBus W	1	0=BOE, 1=EOE
34	Output Ch1 Data	MBus W	128	Data for output Ch1
35	Output Ch2	MBus W	1	0=BOE, 1=EOE
36	Output Ch2 Data	MBus W	128	Data for output Ch2
70-78	Input Ch "Min" Hist	VME R	32	See text
80-88	Input Ch "Max" Hist	VME R	32	See text
90-98	Specific In Ch Hist	VME R	32	See text

Table 16: MBus and VME I/O address offsets.

address 04 with the histogram number in the lowest order 3 bits of the data sent (representing channels 0 through 7).

05 **BIST mode** BIST mode is toggled on/off via address 05, controlled by data bit 0 (0=off and 1=on). All active input, and both output, channels are affected. A VME write to address 05 with data bit 0 set to 0 (1) will cause the BIST bit in each channel's state register (see section 3.1.4) to be asserted (released). The circuitry will recognize the change of state and set the $\overline{\text{BISTEN}}$ lines for each channel accordingly, thus toggling BIST mode. As described in section 3.1.4, a VME read of the input channel state register (see description for item 14 and 15) will allow the ALPHA to determine if the MBT channel is in BIST mode or not.

06 **MBT SCL L2 data** – The read of the SCL Level 2 data (described in section 7.2 below) is via address 06.

07 **L1_BUSY control** – A single bit VME write to address 07 will change the value in the MBT's internal SCL Control Register. The MBT circuitry will automatically drive the SCL mezzanine card output line L1_BUSY accordingly (send 0 will result in a clear, send 1 will assert). To read the state of this control, the ALPHA reads the SCL Control register (see item 11 below).

08 **L1_ERROR control** – A single bit VME write to address 08 will change the value in the MBT's internal SCL Control Register. The MBT circuitry will automatically drive the SCL mezzanine card output line L1_ERROR accordingly (send 0 will result in a clear, send 1 will assert). To read the state of this control, the ALPHA reads the SCL Control register (see item 11 below).

09 **L2_BUSY control** – A single bit VME write to address 09 will change the value in the MBT's internal SCL Control Register. The MBT circuitry will automatically drive the SCL mezzanine card output line L2_BUSY accordingly (send 0 will result in a clear, send 1 will assert). To read the state of this control, the ALPHA reads the SCL Control register (see item 11 below).

10 **L2_ERROR control** A single bit VME write to address 10 will change the value in the MBT's internal SCL Control Register. The MBT circuitry will automatically drive the SCL mezzanine card output line L1_BUSY accordingly (send 0 will result in a clear, send 1 will assert). To read the state of this control, the ALPHA reads the SCL Control register (see item 11 below).

11 **Read SCL Control Register** To read the state of the MBT's internal SCL control register, the ALPHA issues a VME read to address 11, with the lowest order 4 bits in the data returned corresponding to the state of the L1_BUSY, L1_ERROR, L2_BUSY, and L2_ERROR bits in the register. Note that the MBT circuity is designed to drive the corresponding mezzanine card output line according to the state of the appropriate bits in the Control Register (with the exception of the mezzanine card's L1_BUSY output line, which is also controlled by the state of pin A1 on P2. There is no way for the MBT to know what the state of the mezzanine card output lines are.

12 **SCL Reset** – As described below, an SCL Init request from the SCL Hub will come in on the SCL mezzanine card and be passed to the ALPHA via either a VME interrupt or an asserted P2 line to be determined. The ALPHA then, at some point, will inform the MBT to begin the SCL Reset sequence. When the entire crate has completed its SCL reset sequences card by card, the ALPHA must also inform the MBT that the reset is complete, and the MBT will pass the information along to the SCL Hub. The communication

from the ALPHA to the MBT of starting and finishing an SCL Init will be via a VME write using command 12, with bit 0 set to 1 to begin the reset, and bit 0 set to 0 to signal completion. See section 7.3.

13 **SCL Link Errors** – SCL link errors will be counted (see section 7 below). The number of errors is reported by a VME read to address 13. The counter will be 8 bits, and will be reinitialized after each read. The rationale for clear on read is that the counter counts the SCL_SYNCERROR (see section 7.3), which is only by the mezzanine card FPGA when synch has been lost. It could be that synch was lost, and re-established. There is therefore no need to keep the actual value for synch errors.

14 **MBT Input Channel Pointer** – Items 15, 16, and 27-29 below require that a particular input channel be specified first, accomplished by a VME write to address 14 with the lowest order 3 bits set to the channel number of interest.

15 **Input Channel State Register** – As described in section 3.1.4, each HotLinks input channel has an associated state register which reflects the state of (and sometimes controls) the input channel state machine (and hence reflects the state of the input channel receiver). After specifying which channel to read, the ALPHA can read each state register over VME using a read to address 17, latching the 8 bits of each register in the 8 lowest order bits.

16 **MBT Input Channel Error Counters** – There are currently 4 error counters (8 bits each) defined per channel. These counters count the number of Type 1 and Type 2 errors as discussed in section 3.2.2 for the two states **DATA** and **IDLE**. Channel specification is via a VME write to address 15 as above. After setting the channel number, the ALPHA can read all 4 of channel's error counters in a single 32 bit VME read to address offset 16. The order of packing is (bits 0:31) = Type 1 error DATA state[0], Type 2 error DATA state[1], Type 1 error DATA state[2], and Type 2 error IDLE state[3].

17 **MBT State Register** – Each MBT will have a State Register. This is necessary after powerup, and can also be used during SCL Initialization to verify that the card is ready for data taking. The register can be read via a VME read to address 17, returning an 8 bit number. At present, bit 0 (LSB) is clear/set according to whether the MBT card is "ready", and bit 1 is clear/set according to whether the MBT card is in normal/test injection mode.

18 **MOD_DONE Mask** – Each MBT will begin broadcasting when the proper enable condition is satisfied (see section 4. For the Assistants, enable is derived from the logical AND of EV_LOADED(0) and the successful completion of the previous MBT, defined by the MOD_DONE Mask. For instance, if the Pilot is connected to MOD_DONE(10), then the 1st assistant (11) will wait until

EV_LOADED(0) and MOD_DONE(10) to be asserted before broadcasting. Then this MBT is finished, the next MBT (12) will wait for EV_LOADED(0) and MOD_DONE(10) to be asserted before broadcasting and so on. The Pilot does not derive its enable from MOD_DONE, instead using EV_LOADED(0) and START_LOAD.

19 **SOURCE ID** – Each active input channel of each MBT will need a Source ID, to be used to construct the destination address for DMA broadcasting (see section 4). To set the source ID for a specific channel (0-7), the ALPHA must first set the MBT Input Channel Pointer (see item 14 above) via a VME write to address offset 14, then set the source ID for the channel via a VME write to address offset 19, using the lowest 4 bits for the source ID (to be used directly by the MAIN FPGA in configuring the MBus address for broadcasting that particular source). The ALPHA has to take care to change the channel pointer before each source id transfer.

20 **Reframe Parameter** – Input channel reframing will occur whenever some number of consecutive errors (defined as either RVS or non-EOE/BOE/PAD special character received) have been received. This parameter is set via VME read to address 20 using the lowest 8 bits, and read via VME at the same address. Note that 8 bits are indeed overkill (probably 2 bits would suffice) however 8 gives the ALPHA to opportunity to effectively "turn off" reframing by setting the counter high.

21 **Level 2 Data Type Header** – The header for all data coming into the ALPHA must have a Level 2 data type byte. This byte will be placed into the header for the Level 1 SCL information which is FIFO'd into channel 0. This byte is delivered to the MBT via a VME write to address 21, with 8 bits of data. The byte is read at the same address via a VME read.

22 **MAIN FPGA Firmware Version** – The logic code which is downloaded into the MAIN FPGA resides in Flash-ROM on the MBT board itself. This ROM is read/write over VME, and is non-volatile. Changes in the code can be tracked via a firmware version number. This version number will be available for read over VME in a register at address offset 22.

23 **Channel Contents** – The ALPHA can tell which MBT source (channel) has 1 or more fragments in the input buffers via a VME read of the register at address offset 23. This register will contain a mask, with bit "i" indicating 0 (1) for channel "i" having 0 (greater than 0) complete fragments in the input buffer.

25 **Test Data Injection Mode** – Test data can be injected into any input channel FIFO directly via VME. To begin test data injection, the ALPHA

must first put the MBT input state machines into test data injection mode via a VME write to address offset 25 with bit 0 set to 1. To take the MBT out of test injection mode, the ALPHA must write to offset 25 with bit 0 set to 0. Note that the state of the MBT (test data injection vs. normal data mode) is reflected in bit 1 of the MBT state register (see item 17 above).

28 **Inject Data** – The data for test data injection is sent to the MBT for the channel pointed to by the value in address register offset 14 (see above) by a 9-bit VME write to address offset 28. The nine bits include 8 data bits (in the lowest 8 significant bits). To send data into the channel input FIFO, the 9th bit should be cleared, with data residing in the lowest order 8 bits. To make a BOE equivalent, the 9th bit should be set, and the lowest 8 bits should be the Cypress BOE character hex code. Similary, to make an EOE equivalent, assert the 9th bit and send the Cypress EOE character hex code. It will be up to the ALPHA to get these codes correct, the MBT will not check that the hex code is legitimate. The reason for this is driven by the hardware - the Cypress HotLink output consists of 8 data bits, and a single special character bit, which is fed into the front-end PLD. What we are doing here is bypassing the Cypress output with whatever is in the VME register, sent by the ALPHA.

30 **Write L2 Global Result 128 bits** – 128 bits of programmed digital I/O will be available in 200 ns to the framework, delivering the "answer" from the Global Level 2 processor. This information is delivered to the MBT via an MBus write to address 30. Note that the output is actually 16 lines with a time multiplex of 8, one every 25 ns.

31 **Write 16 Levels onto same line as above** – Some non-global level 2 systems may want to use these lines for other purposes. For this, we provide the ability to write 16 levels on these lines (no multiplexing). Note that the 16 lines will be PECL, and will reflect the state of the 16 bits of this write. There will be no pulses or multiplexing. This information is delivered to the MBT via a VME write to address 31.The strobe line will assert and be released when these levels change, however since the levels are intended to be "latched" by a scope, the strobe will only be useful for logic analyzers, *etc.*

33 **MBT Output, Channel 0, Control** – Output of any data to channel 0 must be proceeded by a begin event (BOE) character and terminated by an end event (EOE) character. This is accomplished via an MBus write to address 33, with the lowest order bit equal to 0 for sending BOE, and 1 for EOE.

34 **MBT Output, Channel 0, Data** – Data is sent to the MBT output channel 0 via an MBus write using all 128 bits of data to address offset 34. The ALPHA can pack the data any way it wishes, however the MBT will put the lowest

MBus data byte into the FIFO first, followed by the next, and so on until all 16 bytes are FIFO'd.

35 **MBT Output, Channel 1, Control** – Output of any data to channel 1 must be proceeded by a begin event (BOE) character and terminated by an end event (EOE) character. This is accomplished via an MBus write to address 35, with the lowest order bit equal to 0 for sending BOE, and 1 for EOE.

36 **MBT Output, Channel 0, Data** – Data is sent to the MBT output channel 1 via an MBus write using all 128 bits of data to address offset 36.

- **MBT Occupancy Histograms** As described in section 3.3 above, each of the 3 histogram have 2 bytes per bin, 17 bins=34 bytes per histogram. This requires 9 32-bit reads per histogram. Command 70-78 reads the histogram for the "min" occupancy, 80-88 reads the "max" histogram, and 90-98 reads the specific channel histogram. The histograms bin contents are packed 2 bins per read, with the 1st read containing the value for bin 0 and 1 (occupancy=0 and occupancy=1) and so on up to bin 17 (occupancy=16 events).

  70 occ=0 in bytes 0:1, occ=1 in bytes 2:3 for "min" histogram

  71 occ=2 in bytes 0:1, occ=3 in bytes 2:3 for "min" histogram

  72 occ=4 in bytes 0:1, occ=5 in bytes 2:3 for "min" histogram

  73 occ=6 in bytes 0:1, occ=7 in bytes 2:3 for "min" histogram

  74 occ=8 in bytes 0:1, occ=9 in bytes 2:3 for "min" histogram

  75 occ=10 in bytes 0:1, occ=11 in bytes 2:3 for "min" histogram

  76 occ=12 in bytes 0:1, occ=13 in bytes 2:3 for "min" histogram

  77 occ=14 in bytes 0:1, occ=15 in bytes 2:3 for "min" histogram

  78 occ=16 in bytes 0:1 for "min" histogram

  80 occ=0 in bytes 0:1, occ=1 in bytes 2:3 for "max" histogram

  81 occ=2 in bytes 0:1, occ=3 in bytes 2:3 for "max" histogram

  82 occ=4 in bytes 0:1, occ=5 in bytes 2:3 for "max" histogram

  83 occ=6 in bytes 0:1, occ=7 in bytes 2:3 for "max" histogram

  84 occ=8 in bytes 0:1, occ=9 in bytes 2:3 for "max" histogram

  85 occ=10 in bytes 0:1, occ=11 in bytes 2:3 for "max" histogram

  86 occ=12 in bytes 0:1, occ=13 in bytes 2:3 for "max" histogram

  87 occ=14 in bytes 0:1, occ=15 in bytes 2:3 for "max" histogram

  88 occ=16 in bytes 0:1 for "min" histogram

  90 occ=0 in bytes 0:1, occ=1 in bytes 2:3 for "specific" histogram

91 occ=2 in bytes 0:1, occ=3 in bytes 2:3 for "specific" histogram

92 occ=4 in bytes 0:1, occ=5 in bytes 2:3 for "specific" histogram

93 occ=6 in bytes 0:1, occ=7 in bytes 2:3 for "specific" histogram

94 occ=8 in bytes 0:1, occ=9 in bytes 2:3 for "specific" histogram

95 occ=10 in bytes 0:1, occ=11 in bytes 2:3 for "specific" histogram

96 occ=12 in bytes 0:1, occ=13 in bytes 2:3 for "specific" histogram

97 occ=14 in bytes 0:1, occ=15 in bytes 2:3 for "specific" histogram

98 occ=16 in bytes 0:1 for "min" histogram

9 Initialization

There are many levels of initialization required for the MBT as summarized in table 17. These levels are summarized here, and detailed in sections below:

Level	Description
Powerup	Crate power cycled
Programmed Reset	Parameters left intact
SCL Reset	Parameters and SCL left intact
FPGA Download	User intervention required

Table 17: Various MBT initialization levels.

- Powerup – When the power to the card is cycled. Everything is initialized, except the FPGA instruction downloads which are user controlled via the TCC.

- Programmed Reset – Via programmed I/O , the ALPHA can reset the MBT. All registers, counters, etc. are reset, however registers which are parameters (*e.g.* active masks, *etc.*) are not affected. This is meant to be used as a reset without having to reconfigure. There will also be a front-panel button available for this reset.

- SCL Reset – For the special case of an SCL Init, the SCL Reset is similar to the programmed reset, with the exception of the Pilot MBT SCL section (which has to stay active to handle the SCL Init communication to and from the SLC Hub) and the all MBT input sections, which must go into a special "quiet" state to wait for all inputs to be quiet during an SCL reset (described in section 7.3).

- FPGA Downloading – MAIN FPGA instructions from the flash-EPROM will be nonvolatile, but can be downloaded via VME from the TCC. Note that this downloading will not occur during any automatic online sequence - it must be initiated by personnel in control of the experiment, and will hence not exist as part of the usual downloading tasks for the TCC. We anticipate a standalone program for this purpose. Instructions for the VME FPGA will be downloaded automatically from onboard EPROMs, and hence will be permanently non-volatile, only changeable by removing the EPROM from the socket and replacing with another version. We anticipate that once this EPROM program proves correct (established during the debugging period), it will not have to be changed since the great majority of its work is involved in the MBT VME interface.

Table 18 lists all registers, counters, and states which need to be initialized, which reset conditions (itemized above) affects them, and their default values. Note that the value 50000 (decimal, or C350 HEX) is actually a countdown counter marking 200ns ticks, giving a default value of once per 10 ms (or 100 Hz sampling). All registers and counters are 8 bits unless specified otherwise. Each histogram has 17 bins (0-16) with 2 bytes per bin, or 34 bytes of counters per histogram. The 10 parity counters include 8 input and 2 output. When all resets have been accomplished, the MBT State Register ready bit will be asserted (see section 2 above).

9.1 Powerup

Powerup is the lowest level init state – everything gets reset to the default setting. See table 18. An itemization by function is described below:

- **Input Section.** All error counters are set to zero, and the 3 histograms (min, max, and specific) bins are zeroed. The event ID (used in the address for the MBus broadcast) is set to zero, as is the sampling rate and the active channel list. The Altera will set the parity calculation values for all channels to zero. The state machine for all input channels is put into the IDLE state. The MOD_DONE mask is set to zero, and all source IDs are set to zero.

- **Output Section.** The output parity calculation is set to zero, and the state machine is set to IDLE.

- **L2 Global Information Section.** The contents of the L2 Global Result register (address 30) will be set to zero. The hardware will not automatically transmit all zeros on the 128 bit output line, since the transmission will not be enabled.

- **SCL Section.** The MBT will reset the 4 bits of the SCL Control register to zero (address offset 11). This will cause the SCL lines L1_BUSY, L1_ERROR,

Registers	Programmed Reset	Powerup	SCL Reset	Default
Input Channel State(7)	$\surd$	$\surd$	$\times$	IDLE(0)
Specific Channel	$\times$	$\surd$	$\times$	0
Sampling Freq	$\times$	$\surd$	$\times$	50000 (dec)
Active Ch Mask	$\times$	$\surd$	$\times$	0
Input Channel Source ID(7)	$\times$	$\surd$	$\times$	0
BIST Mode Register	$\surd$	$\surd$	$\surd$	0
SCL Control Register	$\surd$	$\surd$	$\times$	0
Input Ch pntr	$\surd$	$\surd$	$\surd$	0
Reframe Parameter	$\times$	$\surd$	$\times$	2
MOD_DONE Mask	$\times$	$\surd$	$\times$	0
L2 Data Type	$\times$	$\surd$	$\times$	0
Counters				
Input Ch error(7@4 each)	$\surd$	$\surd$	$\times$	0
Input Ch occ(7)	$\surd$	$\surd$	$\surd$	0
MIN Histogram(34)	$\surd$	$\surd$	$\surd$	0
MAX Histogram(34)	$\surd$	$\surd$	$\surd$	0
Specific Ch Histogram(34)	$\times$	$\surd$	$\times$	0
SCL Sync Err	$\surd$	$\surd$	$\times$	0
Parity(10)	$\surd$	$\surd$	$\surd$	0
L2 Event Counter	$\surd$	$\surd$	$\surd$	0
State				
Input	$\surd$	$\surd$	$\times$	IDLE
EV_LOADED(1) contribution	$\surd$	$\surd$	$\surd$	released
Output Channels	$\surd$	$\surd$	$\surd$	WAIT
SCL Output Lines	$\surd$	$\surd$	$\times$	released
Other				
L2HWFW Output	$\surd$	$\surd$	$\surd$	Reset lines
L2HWFW Mode	$\surd$	$\surd$	$\surd$	128-bit

Table 18: Initialization list.

L2_BUSY, and L2_ERROR to be released (set to zero). The SCL sync error counter will be set to zero.

- **Card Level.** At the card level, each MBT will make release its contribution to EV_LOADED(0), and its DONE_OUT line, and be sure it is not driving DSTROBE or DDONE. The Pilot will also release EV_LOADED(1). All MBTs will then determine their VME and MBus slot via VME64x VIPA geographical addressing (**I need to dig up details of this...**). The ALPHA can poll the MBT State Register over VME to determine if the card is initialized by checking the lowest order bit (see section 2). The ALPHA will then download the appropriate parameter information into each MBT (see table 18). Note that powerup does **not** cause the MAIN FPGA to be downloaded via VME. All VME downloading (which changes the FPGA coding) is done via user intervention, and is not part of the DØ TCC configuration. The FPGA coding will exist in a flash-EPROM which is nonvolatile.

9.2 Programmed Reset and Run Start

A Programmed Reset is meant to be used as a reset without having to reconfigure. There will also be a front-panel button available for this reset, and causes all of the same resets as the Powerup with the exception of all registers which represent parameters (see table 18).

To begin using the MBT in real-time DØ running, the ALPHA Administrator may or may not issue a Programmed Reset to be sure the MBT is initialized. However, this should not be necessary, since the MBT will be initialized at Powerup, and during an SCL Reset. However, the Administrator may want to change some of the register values which are used for parameters to configure the card for running (*e.g.* active channel, *etc*). Table 19 lists the registers which the ALPHA may want to configure for these purposes, since these registers affect data taking. All of these parameters are written toand read back over VME. See table 16 for more detail.

Register	Default
Sampling Frequency	50000 (decimal)
Active Channel Mask	0
Input Channel Source IDs	0
Reframe Parameter	2
MOD_DONE mask	0
L2 Data Type	0
Specific Channel to Histogram	0

Table 19: BT parameters (all accessible over VME)which affect data taking.

9.3 SCL Reset

When the MBT receives the SCL Init request from the ALPHA, it must first perform the list of initializations as in table 18. These initializations are somewhat different from those of a Programmed Reset, in that 1) for the MBT Pilot, the SCL Output section is not reset, and 2) all active input channels go into a QUIET state (see section 7.3. The SCL is not reset for the Pilot since during a crate SCL Init, the Pilot's SCL section must remain in contact with the SCL Hub to communicate that the Init is complete for the particular geographic section.

10 Input Test Card

It is required for running debugging purposes that a well defined data set be injected into the MBT front-end FIFOs. We have chosen not to provide circuitry onboard the MBT to do this, but instead opt for the building of a 9U VME card with 7 HotLinks outputs. This card will have 7 input FIFOs, filled via VME commands.

Other VME commands control output: sending BOE, data, physical trailer, EOE.
We have not designed this card at this time.

11 Front Panel

Figure 35 shows the front panel of the MBT card, not to scale. The following items appear on the front panel:

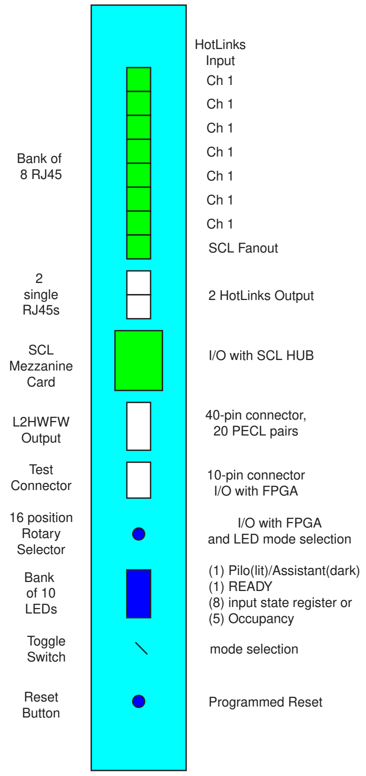

Figure 35: MBT Front Panel.

- HotLinks input receivers – 7 inputs in a bank of 8 RJ45 connectors (banks of 8 are more compact than 8 single RJ45s).
- SCL Fanout HotLinks transmitter – this will be the 8th RJ45 in the bank of 8.
- HotLinks output transmitters – 2 single RJ45s
- SCL Mezzanine card input
- L2HWFW output – 40 pin cable carrying 20 PECL signals

- Test Connector – a 10 pin shrouded connector to carry test signals directly to and from the MAIN FPGA for debugging and commissioning.
- 16 position rotary – for debugging and LED display mode selection.
- Bank of 10 LEDs – 1 LED for Pilot (lit)/Assistant (dark), 1 LED to display the READY bit in the MBT state register (see section 2), and 8 LEDs to be used for either the input channel occupancy or the input channel state register depending upon the position of the rotary switch and one of the toggle switch
- 1 Toggle Switch – mode selection (input channel occupancy/state register)
- Programmed Reset Button – to perform a Programmed Reset (see section 9)

12 Differences between Pilot and Assistant MBTs

As discussed above, the Pilot and Assistant MBTs differ in the following ways:

- SCL Mezzanine card – only on the Pilot
- L2 HWFW communication bus – run in 128 bit digital mode by MBTs in the Global Level 2 crate. All other MBTs are, however, allowed to use this communication path as they choose for debugging and commissioning, however there will only be a few of the DeMux cards (see section 6.1) available, reserved for the Global crate. This path is available to all MBTs in 16 bit PECL level mode also as described in section 6.1.

The selection of Pilot or Assistant will be via a front panel toggle switch. **More on this later....**

13 Programming Summary

In this section we present a summary of how the programmer will program the ALPHA to "talk" to the MBT. Note that all of this information is detailed elsewhere, this section is only intended as a summary. All address offsets are in decimal.

- <u>Programmed Reset</u> – VME W, address offset 01, no data. See section 8 and 9.2.
- <u>Active Channels</u> – VME R/W, address offset 02, bit 0 asserted to write, clear to read, data read as a mask on bits 0:7 for read. See section 8.
- <u>Sampling Frequency</u> – VME R/W, address offset 03, data is a multiple of 200 ns countdown counter. Default is 50000 decimal for 100Hz. See section 8.

- Specific Channel to Histogram – VME R/W, address offset 04, data is 3 bits indicating channel 0-7 (note channel 0 will be the SCL L1 data). See section 8.

- MOD_DONE Mask – VME R/W, 32 bits, address offset 18, contains a single bit set indicating which MOD_DONE line to sample for the broadcast enable. Note that for the Pilot doesn't use this mask for anything, it gets its broadcast enable by looking at START_LOAD instead of any particular MOD_DONE mask. This item is only relevant for MBT Assistants.

- BIST Mode – VME W, address offset 05, bit 0 asserted to set BIST mode, cleared to clear BIST mode. See section 8 and 3.1.3.

- Read SCL L2 Data – ALPHA must poll EV_LOADED(1), and if TRUE, issue VME R to MBT address offset 06 with 128 bits returned in the following order: L1_TURN in bits 0-15, L1_BX in bits 16-23, and L2_ACCEPT in bit 24 (if bit 24=1, then accepted, if 0, then rejected). See section 7.2.

- Level 1 Busy (L1_BUSY) – VME W, address offset 07, bit 0 asserted to set, cleared to clear. See section 8 and 7.3.

- Level 2 Busy (L1_BUSY) – same as above, address offset 08.

- SCL Init – The steps detail the progression of events relating to an SCL Init are described in detail in section 7.3 above.

- SCL Link Errors – VME R, address offset 13, 8 bit counter. See section 8 and 7.

- SCL Control Register – VME R, address offset 11, bits 0:3 indicate L1_BUSY, L1_ERROR, L2_BUSY, and L2_ERROR bits respectively. See section 8 and 7

- Input Channel State Register – VME W, address offset 14, lowest 3 bits indicate channel number of interest, then VME R, address offset 15, lowest 8 bits are that channel's state register. See section 3.1.4.

- L2G HWFW Result – MBus W, address offset 30, all 128 data bits clocked onto the L2HWFW output line. An MBus R to the same address will read back the last 128 bits sent. See section 6.

- Set 16 Levels on same cable as above – VME W, address offset 31, 16 bits of data sent, results in 16 levels on the output. See section 6.

- Output Channel 0 – A sequence of begin, data, end is accomplished via:

  - **Begin**. MBus W, address offset 33, data bit 0 cleared for BOE

- **Data**. MBus W, address offset 34, 128 data bits
- **End**. MBus W, address offset 33, data bit 1 asserted for EOE

- Output Channel 1 – same as channel 0 using address offsets 35 and 36 for BOE/EOE and Data respectively.