Fermi National Accelerator Laboratory

SVX II Silicon Strip Detector Upgrade Project

FIBER INTERFACE BOARD (FIB)

-- PRODUCTION --

June 24, 2003
Version 21

Originator: K. Woodbury
Revised: K. Treptow

Document #ESE-SVX-951010

1. GENERAL INFORMATION

1.1 Forward

This document describes the production version of the "Fiber Interface Board", hereafter referred to as the FIB.

1.2 System Introduction

The FIB is a 9U x 400 VME module that interfaces to the J1/J2/J0 VME64xP[1] backplane and to a custom J3 backplane[2]. It is one of the modules designed to control and readout the SVX3 chips[3, 4]. Figure 1 illustrates the basic components of the FIB crate; the FIB (up to twelve per crate), the FIB Fan Out Board (FFO)[5], and the MVME2301 CPU board. The FIB connects to the Port Card (PC)[6] via the Optical FIB Transition Board (OFTM). The FIB interfaces with the FFO over a custom J3 backplane. The FFO receives timing and command information from the Silicon Readout Controller (SRC)[7] through a Gigabit Fiber-optic Data Link (G-Link)[8].

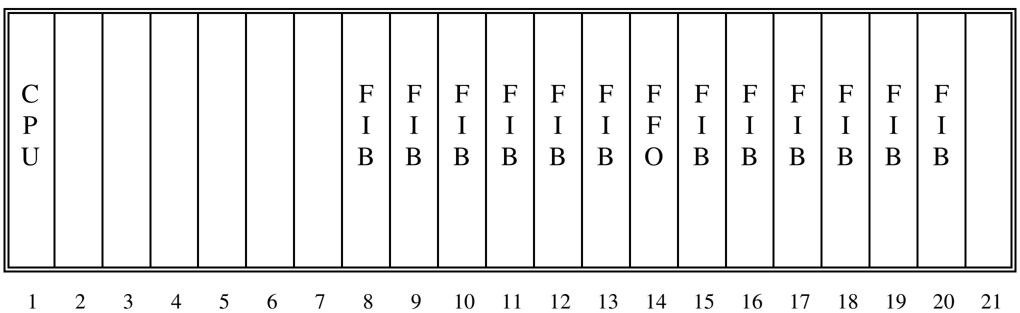


Figure 1 - FIB Crate Components

1.3 Description of Component & How it fits into the System

The FIB is used to both control the SVX3 chips, and to transfer data to the Level 2 VME Readout Buffers (VRB)[9] and to the Silicon Vertex Trigger (SVT) [10] system. The FIB executes commands from one of three sources:

- Commands delivered by the SRC board through the J3 backplane.
- Commands previously stored in the Command FIFO by the VME CPU.
- Commands processed immediately as written to the Command FIFO by the VME CPU.

The FIB interprets these in-coming commands and delivers encoded control information to the PC.

The state/control information is translated on the PC and generates the logic levels to control the internal features of the SVX3 chips. One PC can control up to five chains of SVX3 chips that are mounted on a hybrid. These hybrids are connected to the PC through the High Density Interconnect (HDI) cable [11].

1.3.1 Command Flow

The following is the flow of commands and clocks from the SRC to the SVX3 chips.

- The SRC sends a 53 MHz clock (Master Clock), timing signals and commands to the FFO via a dedicated G-Link fiber.
- The FFO distributes the 53 MHz clock and SRC commands through the J3 backplane.
- The FIB interprets the commands from the SRC and sends the SVX3 clocks (FE_CLK, BE_CLK, L1A, PIPE_RD2), the Control Clock (C_CLK), and the encoded control information (C[4:0]) to the PC using low voltage differential levels. *Note these drivers are on the Optical FIB transition module (OFTM)* [12].
- The PC controller interprets the encoded control information, using the Control Clock (C_CLK) and delivers the proper CMOS logical levels to the SVX3 chips through the HDI.
- The clock sequences for the SVX3 chips are generated directly by the FIB. The PC fans out the clocks for the five HDIs.

1.3.2 Data Flow

The data flow from the SVX3 chips travels via the following path:

- SVX3 chips transmit the data to the PC, through the HDI..
- The PC converts these signals, places them on a parallel optical link and transmits the data to the FIB through five 9 bit parallel optical links (8 bits of data and a Data Valid signal). *Note: These receivers are on the OFTM* [12].
- The FIB board accepts data from ten of these HDI links.
- Data from the HDI is processed (header frames and End Of Records added) and then transferred to the Level 2 buffers (VME Readout Buffer (VRB) modules) and Silicon Vertex Trigger (SVT) system.

1.4 List Of Component Requirements

The major components of the FIB design are as follows:

- The FIB is a 9U x 400mm VME64xP card.
- The FIB is connected to the J1/J2/J0 and J3 backplanes of the VMExP crate.
- The core of the J1/J2 backplane follows the VMEbus specification (ANSI/IEEE STD1014) [13].
- Additions have been made to the J1/J2 backplane to provide several enhancements as per the VME for Physics recommendations (add reference).
- The J3 backplane is a custom made backplane [ref. **Error! Bookmark not defined.**]. The FFO board sends commands and clocks to the FIB through this backplane.
- Pins on the J0 connector provide -5.2 V for the ECL logic on the FIB.
- The SRC through the FFO supplies one command every 132 ns.
- The FIB controls two PCs. The FIB is capable of reading from all ten HDIs connected to the two PCs. The readout **data rate** is 26.5 MBytes/sec in each HDI.
- The digital electrical interface between FIB and PC is done on a transition module which is described in another document [12].
- Four G-Link transmitter daughter boards can be connected to the FIB. The G-Link daughter boards are used to send data that has been read by the FIB to the VRB board.
- Cooling for the FIB is supplied by fans mounted external to the VME crate. Additional cooling will be required in the form of external air-water heat exchangers.
- The FIB requires +5.0 V and -5.2 V power supplies.

2. THEORY OF OPERATION

2.1 Basic Features & Operation

Figure L1 below, is a simplified block diagram of the FIB*.

(figure L1 -FIB simplified block diagram)

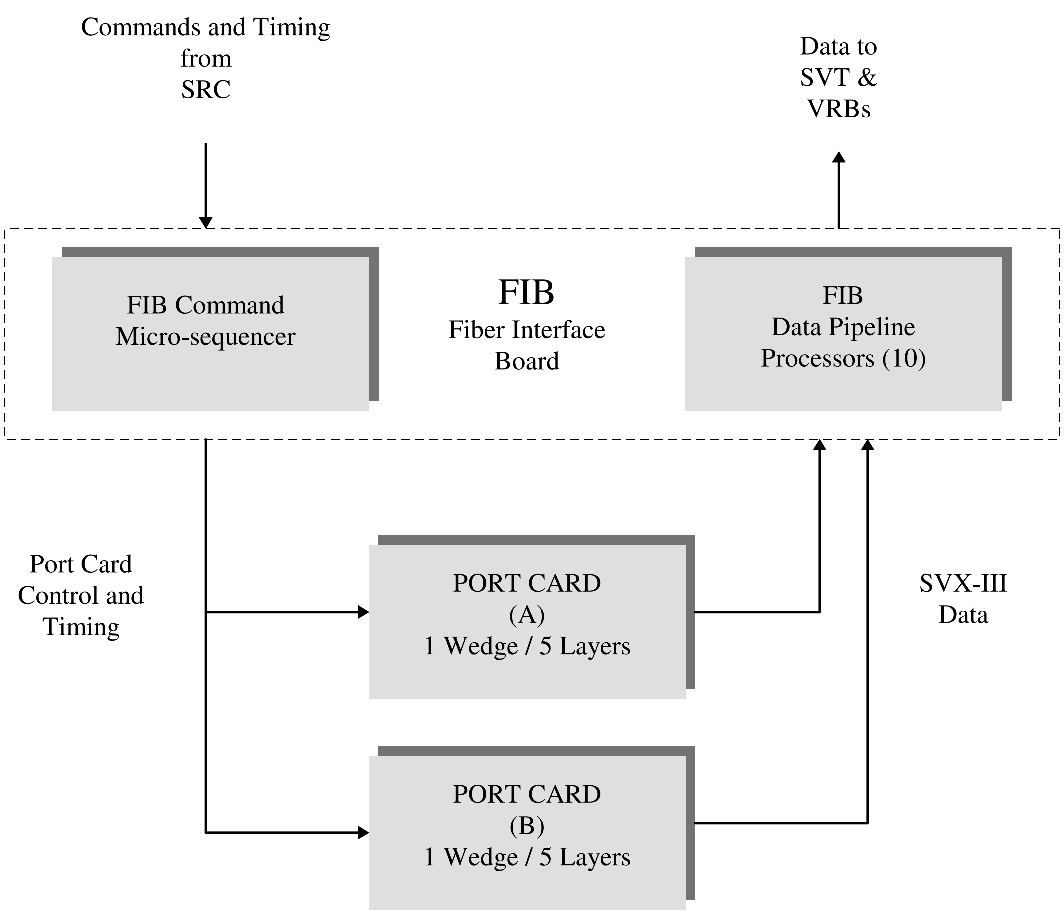


Figure L1 - FIB Simplified Block Diagram

The main functions of the FIB are the following:

- The execution of commands sent by the SRC.
- Translates these commands into encoded control sequences to the Port Cards (PC).
- Synchronizes and shapes the appropriate clock for the SVX3 chips.
- Allows diagnostic testing of the SVX3 chips and Port Cards.
- Delivers configuration commands to the Port Card.
- Controls the configuration of the SVX3 chips.
- Provides read-back of the configuration of the SVX3 chips.
- Receives, processes and stores, into on board Output FIFOs, the parallel data from the Port Card.

- Appends Bunch Crossing and HDI identification frames to the head of the data from the PC.
- Appends a minimum of four EOR frames to the tail of the data from the PC. Either EORn - End of Record Normal, EORt - End of Record Truncate, or EORa - End of Record Abort, followed by a minimum of two EORf - End of Record Fills.
- Transfers the data collected from the PC to 4 G-Links, to be transmitted to the VRB and SVT systems.

All units of the FIB operate synchronously with a 53 MHz clock. This 53 MHz clock is supplied by the SRC via the FFO and through the J3 backplane *.

2.1.1 Detailed Block Diagram

Figure 2., on the next page, shows a more detailed block diagram of the FIB.

(figure 2 - FIB Block Diagram)

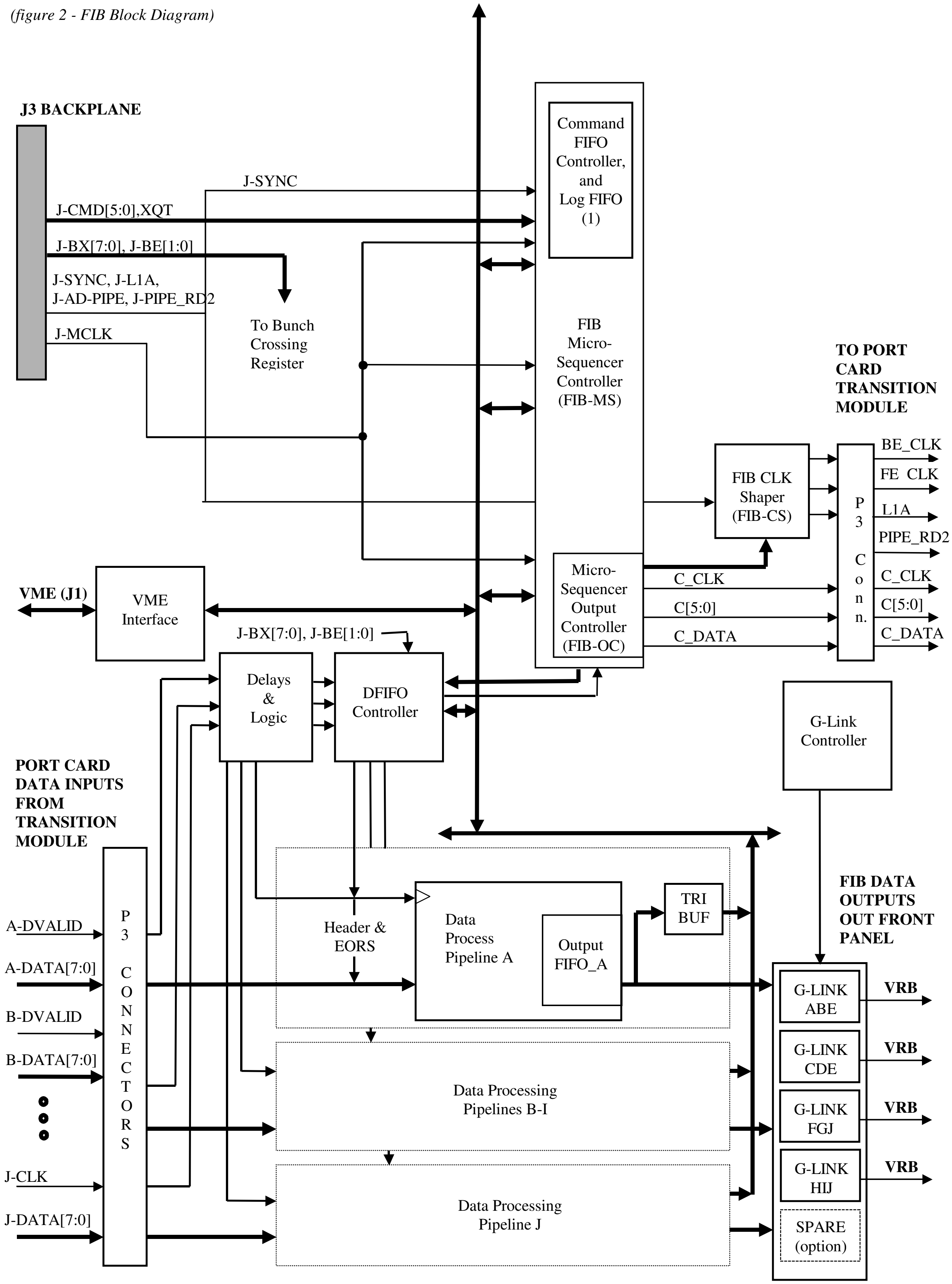


Figure 2. FIB Block Diagram

2.2 Command Processing

This section describes how the FIB processes commands from the SRC/VMEbus, and focuses primarily on the FIB Micro-sequencer as shown in the detailed block diagram on the previous page.

2.2.1 FIB Micro-sequencer

The block diagram below shows the basic design of the FIB Micro-sequencer. The output timing for all commands are stored in fast Non-volatile SRAM. The timing of each command is stored graphically on a central file server, so it can be quickly updated. The specific software that will be used to store this information is not documented at this time.

(Figure L2 - Micro-sequencer Simplified Block Diagram)

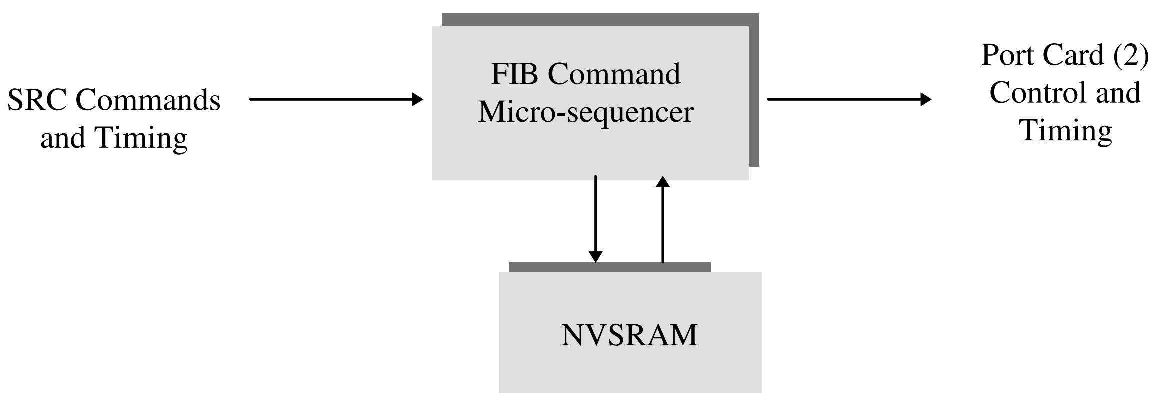


Figure L2 - Micro-sequencer Simplified Block Diagram

The FIB-MS has several VME registers. One of the functions of these VME registers is to configure the FIB-MS as to the source from which to execute commands.

- Commands synchronously entered directly into the Command FIFO, from the SRC through the J3 backplane.
- The list of commands previously downloaded into the Command FIFO.
- Commands asynchronously entered directly into the Command FIFO.

Commands are entered from the VMEbus for either initialization or testing purposes only.

The during *normal operation*, SRC commands are processed which control the data acquisition of the SVX3 chips. However, there are specific commands for initialization of the SVX3 chips, Port Card and for diagnostic testing, which must be issued from the VMEbus.

This mode, where the VMEbus CPU is issuing the commands, is also implemented to allow the FIB-MS to test all commands that the SRC can request independently of the SRC module. This “Testing Mode” is a powerful tool for diagnostic testing as well as for the development of the FIB. *(Please note, running without the SRC requires that the*

FFO be used to generate clock and timing signal inputs). The VME interface writes to the Command FIFO the list of commands to be executed. The VMEbus CPU then issues an Enable FIB Micro-sequencer command which then initiates the processing the list of commands.

With the Micro-sequencer enabled, the VMEbus CPU can also asynchronously issue new commands which will be immediately processed. This “Immediate Mode” is used for configuration of the SVX3 chips and the DACs of the PC. This mode may also be used for diagnostic testing.

At any time the VMEbus can read the last command that was processed in the FIB-MS. If commands are being processed continuously from the SRC, there is no guarantee as to the accuracy of reading this register “asynchronously”. The SRC can, however, issue a latch status command which then stops this latch from updating. This may be especially useful during system configuration/testing; for example, to detect if a particular FIB is not receiving commands.

When the FIB-MS receives a command, it executes the appropriate operation. If the command is associated with the PC, the FIB-MS combined with the FIB-MS sends the necessary encoded control sequences (through the C_Clk, C_Data, and C[4:0] bus) to the PC.

The FIB receives commands, interprets these commands, and when necessary sends encoded control sequences to the PC via the FIB Micro-sequencer. The bits of the encoded control bus are validated by the Control Clock (C-CLK). As the micro-sequence is delivered to the PC-CT, the FIB, PC and SVX3 chips work synchronously. The FIB Micro-sequencer sends C_CLK and C[4:0] to advance/change the state of the SVX3 chips, in addition the Micro-sequencer delivers the FE_CLK, BE_CLK, PIPE_RD2, and L1A clocks with the appropriate timing required by the SVX3 chips and PC-CT.

2.2.2 Pre-amplifier Reset Command Processing

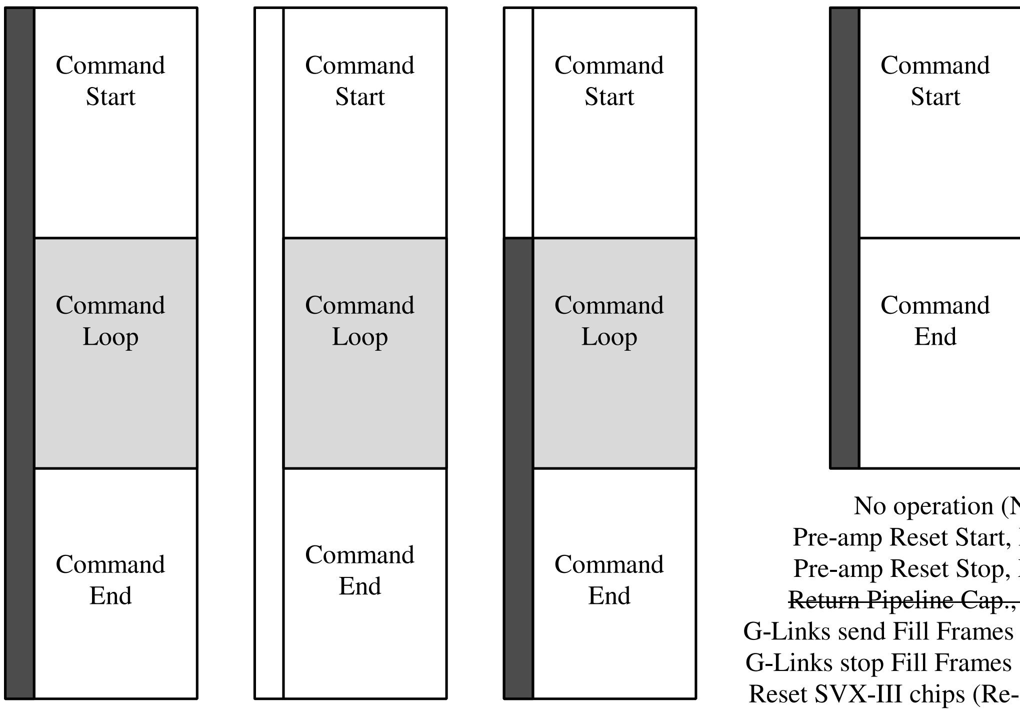
Unlike a number of the commands such as Digitization and Readout, the Pre-amplifier Reset Start and Pre-amplifier Stop commands must be synchronous with the beam; specifically the Start and Stop command must be issued within the duration of the abort gaps (3 per turn 16x132ns long = 2.11us). To handle this low latency they can be handled as “preemptive interrupts”, where the FIB-MS can be interrupted during its operation.

These two commands use a command modifier bit from the SRC (XQT), which indicates that the command should be processed immediately. If this command modifier bit is not set, the commands are placed in the Command FIFO queue, to be processed in the normal sequence of commands.

Command Processing Structure

This command processing structure is based on the PA RESET STOP command being issued (from the SRC) immediately following a PA RESET START command. Either PA RESET START or PA RESET STOP can preempt any other command (except during CALIBRATION INJECT and at the START of DIGITIZE).

Note: RETURN PIPELINE CAPACITOR COMMAND now happens independent of the command stream, and has be replaced by a direct connection from the SRC to all of the PIPE-RD2 lines on the SVX-III chips.



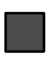
Readout Command

Initialize and Calibration Inject Commands

Digitize Command

No operation (NOP)
Pre-amp Reset Start, XQT = off
Pre-amp Reset Stop, XQT = off
~~Return Pipeline Cap., XQT = off~~
G-Links send Fill Frames (FIB -> VRBs)
G-Links stop Fill Frames (FIB -> VRBs)
Reset SVX-III chips (Re-sync. Pipeline)
Reset Port Card Decoders
(go into Data Acquisition Mode)
Reset FIB (pending)
Latch status
Abort (digitize or readout)

Key:

XQT enable = 

For immediate execution of:
PA RESET START and
PA RESET STOP

<u>VME only Commands</u>

Enable all HDIs
Enable HDIx (1-5)
Enable Port Cardx (1-2)
Initialize SVX chips
Configure HDI

2.3 SVX3 clock processing

The SVX3 chip acquisition clock or Front End Clock (FE_CLK) can have different shapes and frequencies. The frequency of the clock is controlled by the mode of operation of the Tevatron, i.e. what the frequency of interaction is, either 396ns between beam interactions, or 132 ns between beam interactions. The goal in either of these beam scenarios is to provide as much integration time as possible to the SVX3 front end chips; specifically, the time from falling edge to falling edge of the Front End Clock should be maximized as much as possible. Another variable is the amount of time used to reset the input channels, which is also controlled by the Front End Clock high time. In addition, the maximum skew between the SVX3 chip acquisition clock has to be controlled within strict boundaries in order to guarantee that all SVX3 chips controlled by different FIBs and PCs will be acquiring data synchronously.

FIB Clock Shaper formats the SVX3 chip acquisition clock using the Master Clock (J-MCLK), advance pipeline signal J-AD-PIPE, and the J-SYNC pulse delivered by the J3 backplane. For all other operations (such as initialization), the clock is supplied directly by the FIB Micro-sequencer and the SVX3 Clock Shaper logic. Differential drivers on the FIB/PC transition module [ref. 12] are used to send the FE_CLK, BE_CLK, L1A, PIPE_RD2, C_CLK, C_DATA and C[4:0] to the PC.

2.3.1 Calibration Inject

One of the commands executed from the FIB-MS is the Calibration Inject. This command forces the SVX3 chip to inject charge into its own pre-amplifiers. The FIB Clock Shaper FIB-CS, times this command in relation to the acquisition clock. Specifically, when the FIB-MS recognizes this command, it prepares for a Calibration Inject and executes a wait loop, until the proper timing signal arrives from the FIB-CS, which times the change of the state of the SVX3 chips in relation to the Master Clock and the FE_CLK. Figure 5 shows the relationship among the several signals associated with the calibration inject and the SVX3 data sampling.

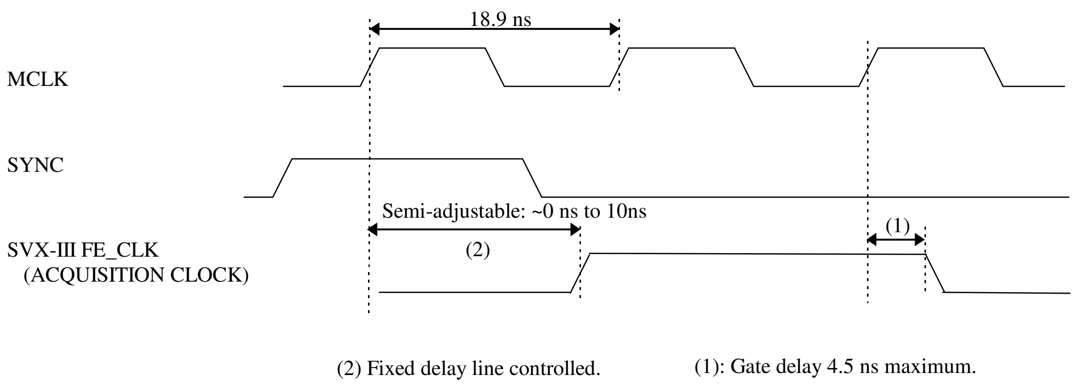


Figure 5. FE_CLK and Calibration inject timing
(Note: Calibration Injection Timing not yet shown!)

2.3.2 Number of Clocks for the SVX III Analog to Digital Converter

The SVX3 chips have an analog-to-digital Gray Code counter, which counts until it reaches the Counter Module value programmed inside the configuration parameter bits [174:167] of the SVX3 chip. The analog to digital conversion is completed when the Gray counter reaches the Counter Module value, and the FIB can proceed with the readout.

Presently, the FIB can be programmed to the number of digitization clocks to deliver to the Port Card; as a result it can deliver just enough clocks to convert the value programmed in the Counter Module register of the SVX3 chip. To be conservative, the FIB can always be programmed to the maximum value that can be programmed into the Counter Module register, however, this lengthens the time for the digital conversion command.

2.4 Draft Data Processing

As shown in the block diagram (figure 2), the FIB can read out and process data from a total of ten High Density Interconnects (HDI), i.e. the data from two (2) Port Cards. The data is transmitted to the FIB via optical or differential receivers which translate this data to TTL electrical levels [12].

In addition to appropriate headers and End-Of-Record information being added to the "data stream", each data frame can have an independent pedestal/baseline subtraction taken. All of the processing is handled by ten (10) independent Data Processing Pipelines.

An overview of the Data Processing Pipeline is shown in the next section.

2.4.1 Data Processing Pipelines - Overview

The above figure (N1) shows the general topology of the data processing pipeline. As each new channel/data set is

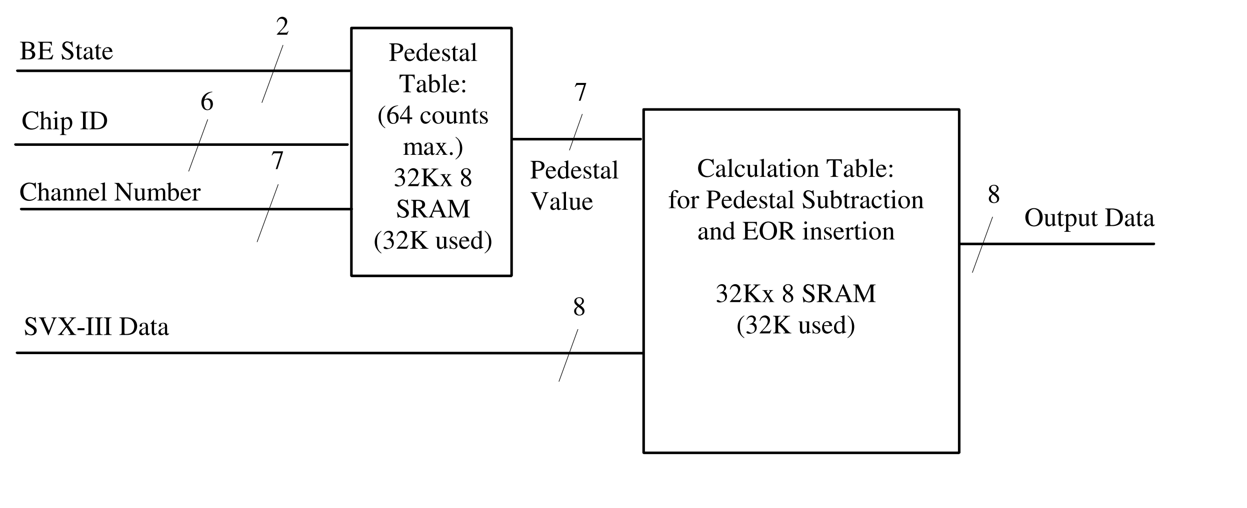


Figure N1 - Data Processing Pipeline (2.1 - 6/99)

presented, first the channel pedestal is determined, then the appropriate calculation value is found given the SVX3 input data. The pedestal/baseline is only subtracted from the ADC data and not from the other fields in the data stream.

2.4.2 Data Processing Pipelines - Internal Structure

Each Data Processing Pipeline has the following internal structure as shown below:

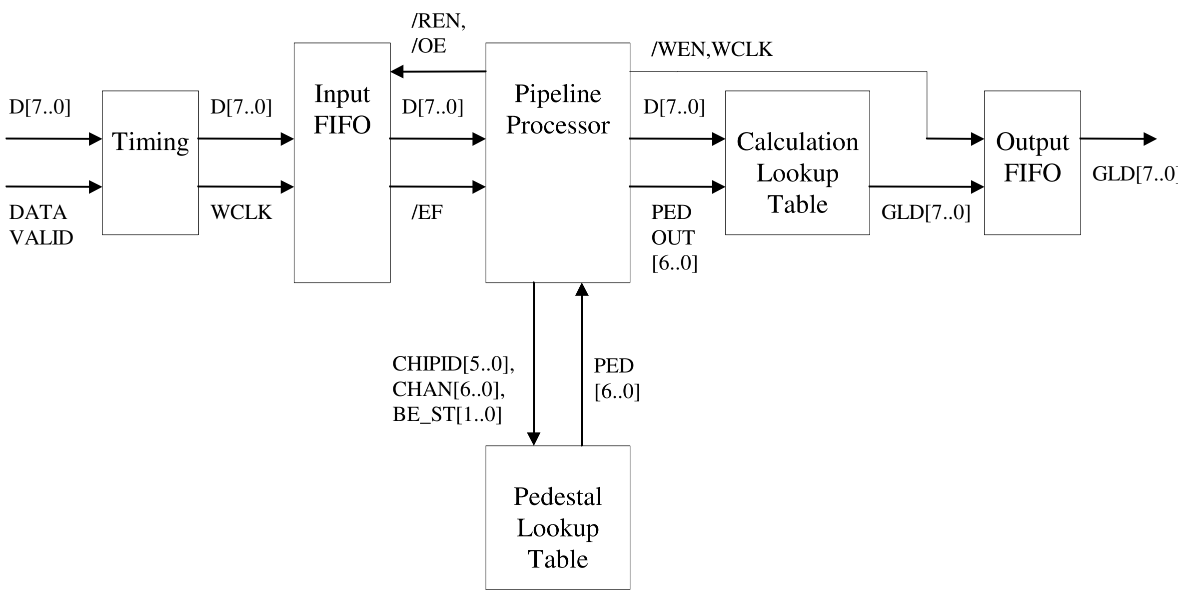


Figure M1- Data Processing Pipeline Schematic

The main components of the Data Processing Pipeline is the set of ten Input FIFOs and Pipelines Processors (A through J). The Output FIFOs are used to synchronize the data from the different HDIs and for diagnostic testing.

The control of these Input FIFOs (ex. testing vs. data readout) are set by a control register located inside the FIB FIFO Controller FIB-FC. The FIB-FC communicates with the FIFO controller logic which then generates the proper clocking for the various components of this data readout section of the logic. During testing or calibration the data can be held in the Output FIFOs for read back via the VMEbus processor. This is done by disabling G-Link output. To disabling the G-Link output is done by accessing a control register in the FIB G-Link Controller (FIB-GC).

The FIB reads out data from the SVX3 chips and transmits this data to the VRB through the G-Links. Further details on the format of this data are in the Output FIFO section of this document (Section 0). The Output FIFOs are primarily used to synchronize data from the different HDIs.

2.4.3 Pipeline Operation

Prior to data processing, all header data is pre-stored in the Input FIFOs. The VMEbus CPU, during initialization, downloads the HDI identification number into the assigned registers. The Bunch Crossing Number and Back End State is down-loaded from the command bus into the FIB-FC immediately following the transmission of a Digitize Command.

Immediately prior to the readout of the data, a signal that indicates that a readout is about to start is transferred to the FIB-FC, which then stores the Bunch Crossing Number, Back End State and HDI ID into the Input FIFOs. Then, during readout, the FIB-MS enables the readout clock for the SVX3 chips and informs the FIB-FC to enable the data write operation.

Most data passing through the Pipeline Processors is untouched, except for the ADC data which has the pedestal subtracted from it.

As soon as all of the Output FIFOs connected to one G-Link have data available more words, the FIB-GC reads the data out of the FIFOs and sends the information through the G-Links.

2.4.4 Odd-byte Data Valid

To properly clock the data from the Port Cards an Odd-byte Data Valid signal is bundled in with the each of the input streams. This data valid signal is then used to generate a write strobe for the input FIFOs.

The SVX3 data valid signals (A-DVALID through J-DVALID signals shown in Figure 2) arrive in parallel with the data from the PC Transition Module. Figure 3 shows the relation between this data and clock. New data is available at each transition of the SVX3 data valid signal which operates at approximately 26.5 MHz. The delay between the transition of the data valid signals and data from the output of the SVX3 chips should be approximately 0 ns, therefore a fixed delay can be used to set the proper relationship between data valid signal and write strobes to the input registers of the Data Processing Pipeline.

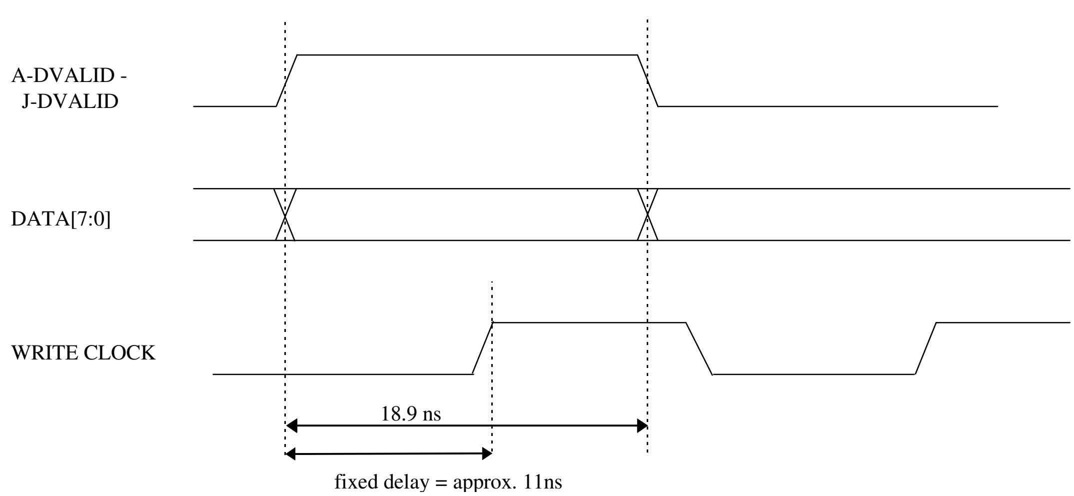


Figure O1 - Odd Byte Data Valid Timing

Output FIFOs

There are a set of ten 4K x 8 FIFOs (Note: this has been increased from the TFIB size of 2K x 8 to hold an entire data readout from the largest HDI) with each FIFO associated with one HDI. The Data FIFOs are shown in Figure 2. These FIFOs receive data from the Port Card Transition Modules and can be read by the VMEbus.

The Data FIFOs have two main functions:

- The primary function is the synchronization of the data coming from different HDIs. The data comes from the PC in parallel with a data valid and is stored inside the FIFOs, after going through the Data Processing Pipeline. When all FIFOs have data available, the data is then transmitted through to the G-Links.
- Other functions are associated with the following diagnostic operations:
  a) the VMEbus CPU board can download a known data pattern into these FIFOs and use the G-Links to ship the data to the VRB.
  b) the G-Links may be disabled with the data that comes from the PC being read by the VMEbus CPU board instead of being transmitted to the VRB.

2.4.5 Pipeline testing

One of the commands that the FIB can emulate is the readout of the SVX3 chips. The Disable G-Links bit in the Control Register (this implementation may change) disables the G-Links and forces the data to remain in the Output FIFOs. Normally, the Output FIFOs are used to synchronize different data paths from the PC. The VMEbus CPU board can read this data from the Output FIFOs and check data consistency. This allows the VMEbus CPU board to read the data from the SVX chips without the SRC board.

Note, the data that the Pipelines process during testing is forced into all of the input FIFOs from the FIB-FC, i.e. all data packets are the same.

2.4.6 Data Output Format

The data from the PC has the format shown in Table 2 (a). End Of Readout (EOR) is differentiated from CHIP ID and Channel Number by the two most significant bits (see Table 1). Further details are provided within the SVX3 and PC specifications.

Control Byte	Bit Assignment (d=data bit)
Chip ID	10dddddd
Channel #	0ddddddd
EOR	11dddddd

Table 1. Control Byte and Bit Assignment

The FIB board appends the Bunch Crossing number, Back End State and HDI identification number to the beginning of the data coming from the PC. This is shown in Table 2 (b).

(a)

CHIP ID
STATUS
CH #
DATA
CH #
:
CHIP ID
STATUS
CH #
:
CH # 127
DATA

(b)

ID0	ID_0 = HDI ID MSB [7:0] (FIB ID A [7:0])
ID1	ID_1 = HDI ID LSB [7:0] LADDER ID [3:0] (value = 0-9) FIB ID B [7:4]
ID2	ID_2 = Bunch Crossing Number [7:0]
ID3	ID_3 = Back End State [1:0] (The byte values are: 0, quiescent; 1, digitize mode; 2, read mode.)
CHIP ID	
STATUS	
CH #	
:	
CHIP ID	
STATUS	
CH #	
:	
CH # 127	
DATA	
EORn	
EORn	
EORf	
EORf	

Table 2. Data format (a) from Port Card, (b) with Bunch Crossing Number, Back End State, HDI ID and EOR information appended.

2.4.6.1 End of Record formats

The table below shows the different EOR frames that are defined. Note there is a minimum of four End of Record frames appended onto each data packet; the first two EOR frames indicate a normal data set, or if there were any errors. The EORf frames are used to pad adjacent links, and there are a minimum of two present in every data packet.

Bit	7	6	5	4	3	2	1	0	
Channel #	0	x	x	x	x	x	x	x	
Phi chip ID	1	0	1	x	x	x	x	x	
Z chip ID	1	0	0	x	x	x	x	x	
EOR	1	1	x	x	x	x	x	x	
EOR_0	1	1	0	0	0	0	0	1	EOR_0 = EOR_n = End Of Record - Normal
EOR_1	1	1	0	0	0	0	1	0	EOR_1 = EOR_t = End Of Record - Truncate
EOR_2	1	1	0	0	0	1	0	0	EOR_2 = EOR_f = End Of Record - Fill
EOR_3	1	1	0	0	1	0	0	0	EOR_3 = EOR_a = End Of Record - Abort

Table 3 -End of Record data patterns

2.5 G-Link Boards

The G-Link and Finisar transmitters are assembled on daughter boards which are plugged into the front of the FIB. Four of these daughter boards are connected to the FIB, with the ST output connector protruding through the front panel. Figure 4 is a diagram of this daughter card.

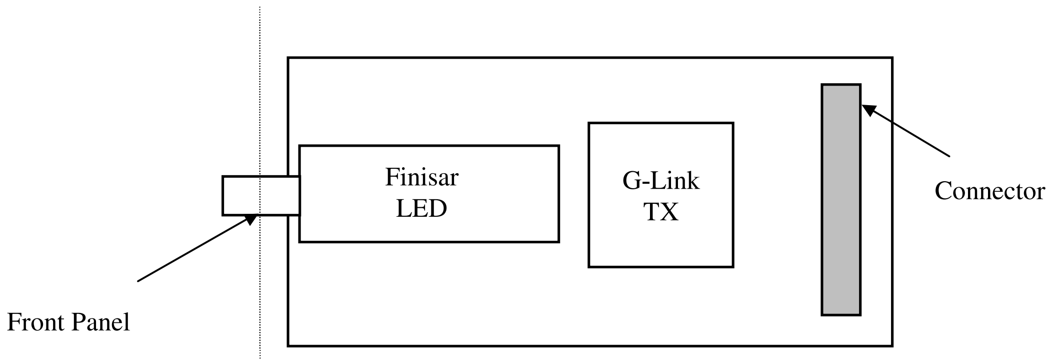
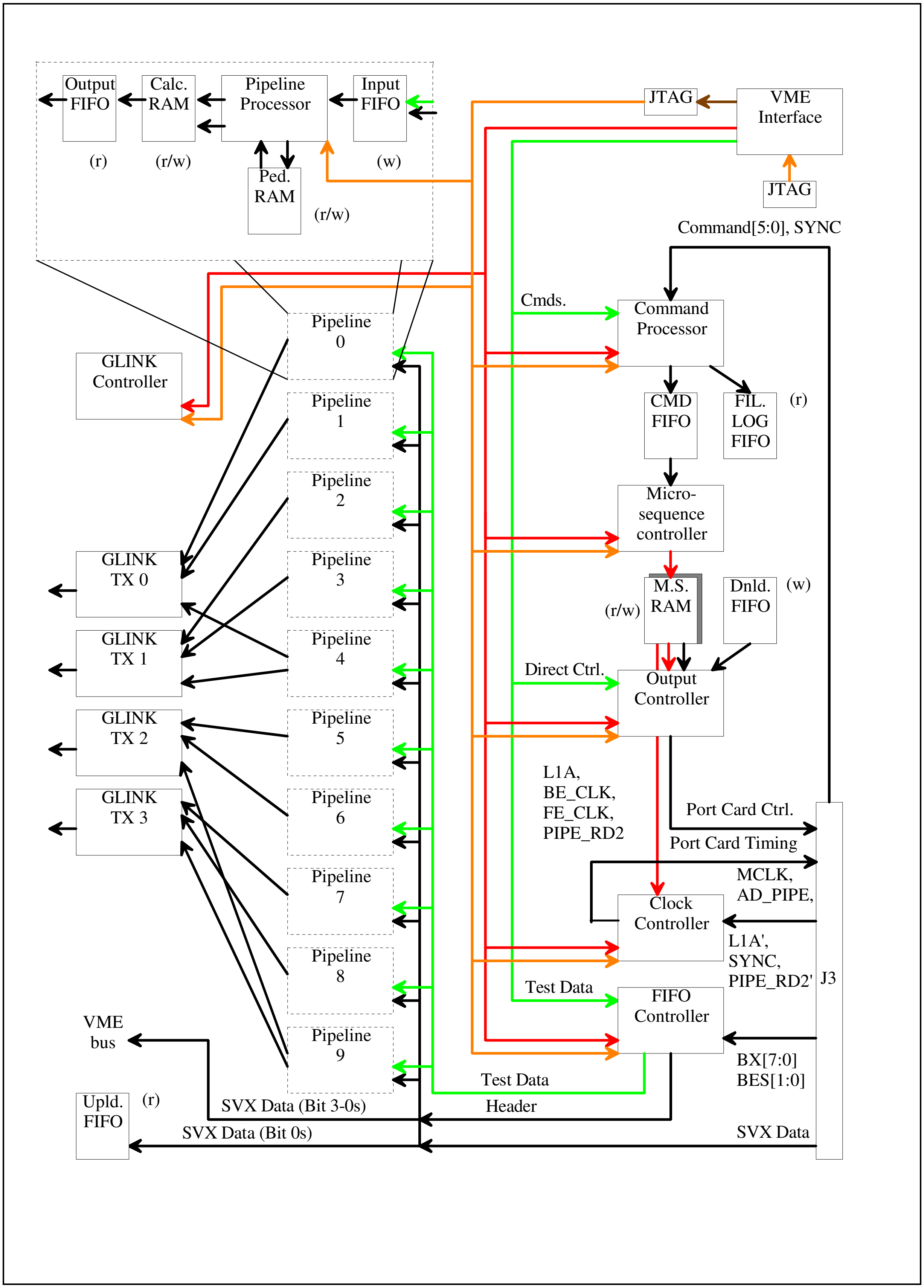


Figure 4. G-Link Board

The implementation of this high speed data link as a self-contained board makes for greater flexibility of the overall data acquisition system in view of future upgrades. The daughter card concept permits convenient independent testing of the high speed communication system. Maintenance is simplified due to the boards' ease of replacement. The approximate dimensions of this daughter board is 4.5" x 2". For more information about this interface, refer to the G-Link/Finisar Transmitter and Receiver Boards specifications [14].

2.6 Data Flow Diagram

As a summary of all of the FIB board components, the Data Flow Diagram on the next page, shows an overview of the control and data flow through the FIB.


Output FIFO
Calc. RAM
Pipeline Processor
Input FIFO
(r)
(r/w)
(w)
Ped. RAM
(r/w)
JTAG
VME Interface
JTAG
Command[5:0], SYNC
Cmds.
Command Processor
GLINK Controller
Pipeline 0
Pipeline 1
Pipeline 2
Pipeline 3
Pipeline 4
Pipeline 5
Pipeline 6
Pipeline 7
Pipeline 8
Pipeline 9
CMD FIFO
FIL. LOG FIFO
(r)
Micro-sequence controller
GLINK TX 0
GLINK TX 1
GLINK TX 2
GLINK TX 3
M.S. RAM
(r/w)
Dnld. FIFO
(w)
Direct Ctrl.
Output Controller
L1A, BE_CLK, FE_CLK, PIPE_RD2
Port Card Ctrl.
Port Card Timing
MCLK, AD_PIPE,
Clock Controller
L1A', SYNC, PIPE_RD2'
J3
Test Data
FIFO Controller
BX[7:0] BES[1:0]
VME bus
Test Data
Upld. FIFO
(r)
SVX Data (Bit 3-0s)
Header
SVX Data (Bit 0s)
SVX Data

3. DIAGNOSTIC/DEVELOPMENT SOFTWARE

A description of the FIB test software will be included here.

3.1 FIB API

The FIB API header file is included below. This shows all of the function calls associated with controlling/operation of the FIB. The test calls (noted 3apr97) have been eliminated from this listing.

```
/* SVX_FibApi.h   -   FIB API HEADER FILE
 *
 * NOTE: Type INT is assumed 32 bits!!!
 *
 * naming convention: fib_<Functional block> (Action target) (Action)
 *
 * 3apr97 DS          added tests translated for IRIX-Tcl/Tk
 */

/* buffer unsigned int*, all functions below */

/* VME Interface (Bank 0) */

int fib_IDRead(VISION_SLAVE slave, unsigned int* bid, unsigned int*,
                                        unsigned int*, unsigned int*);    /* fib_IDRegRead */
int fib_JtagStatus(VISION_SLAVE slave,unsigned int* buffer);
int fib_JtagEnable(VISION_SLAVE slave);
int fib_JtagDisable(VISION_SLAVE slave);
int fib_JtagRead(VISION_SLAVE slave,unsigned int* buffer);
int fib_JtagTMSTDIWrite(VISION_SLAVE slave,unsigned int* buffer);
int fib_JtagTMSTDIRead(VISION_SLAVE slave,unsigned int* buffer);          /* fib_JtagReadTMSTDI */
int fib_JtagTMSTDIClock(VISION_SLAVE slave,unsigned int* buffer);         /* fib_JtagForceTMSTDI */
int fib_JtagTDORead(VISION_SLAVE slave,unsigned int* buffer);                  /* fib_JtagReadTDO */
int fib_MasterClockVerify(VISION_SLAVE slave, unsigned int* buffer);   /* fib_CheckMasterClock */
int fib_MasterReset(VISION_SLAVE slave);

/* General Status Register (Bank 1) */

int fib_GeneralStatus(VISION_SLAVE, unsigned int*);

/* FIB Micro-sequencer 1 - Command Processor (Bank 2) */

int fib_CommandCsrRead(VISION_SLAVE slave, unsigned int* vce,
                                              unsigned int* mse);
        /* fib_ReadCommandCsr */
int fib_VmeCommandEnable(VISION_SLAVE slave);                             /* fib_VmeCommandLoadEnable */
int fib_VmeCommandDisable(VISION_SLAVE slave);                            /* fib_VmeCommandLoadDisable */
int fib_ManualStrobeEnable(VISION_SLAVE slave);
int fib_ManualStrobeDisable(VISION_SLAVE slave);
int fib_VmeCommandRead(VISION_SLAVE slave,unsigned int* buffer);     /* fib_ReadVmeCommand */
int fib_VmeCommandWrite(VISION_SLAVE slave,unsigned int* buffer);    /* fib_WriteVmeCommand */
int fib_VmeCommandStrobe(VISION_SLAVE slave);                             /* fib_ManualStrobeCommand */
int fib_LatchedStatusClear(VISION_SLAVE slave);                           /* fib_ClearLatchedStatus */
int fib_CommandFifoReset(VISION_SLAVE slave);                             /* fib_ResetCommandFifo */
int fib_FilLogFifoReset(VISION_SLAVE slave);                              /* fib_ResetFilLogFifo */
int fib_CommandFifoRead(VISION_SLAVE slave, unsigned int* buffer);   /* NOT COMPLETE! fib_ReadCommandFifo */
int fib_FilLogFifoRead(VISION_SLAVE slave, unsigned int* buffer);    /* fib_ReadFilLogFifo */

/* FIB Micro-sequencer 2 - Address Controller (Bank 3) */

int fib_LastCommandRead(VISION_SLAVE slave, unsigned int* buffer);   /* fib_ReadLastCommand */
int fib_AddrControlCsrRead(VISION_SLAVE slave,
                              unsigned int* cmd_processing_enable,
                              unsigned int* vme_addr_asserted);       /* fib_ReadAddressControlCsr */
```

```
int fib_CommandProcessingEnable(VISION_SLAVE slave);                    /* fib_EnableCommandProcessing */
int fib_CommandProcessingDisable(VISION_SLAVE slave);                   /* fib_DisableCommandProcessing */
int fib_VmeAddrAssertedRead(VISION_SLAVE slave,unsigned int* buffer);   /* fib_ReadVMEAddrAsserted */


/* FIB Micro-sequencer 3 - Output Controller (Bank 4) */

int fib_OutputControlCsrRead(VISION_SLAVE slave,
                                unsigned int* vme_ctrl_enable,
                                unsigned int* c_clock1_enable,
                                unsigned int* c_clock2_enable);  /* fib_ReadOutputControlCsr */
int fib_DirectPortCardControlEnable(VISION_SLAVE slave);        /* fib_EnableDirectPortCardControl */
int fib_DirectPortCardControlDisable(VISION_SLAVE slave);       /* fib_DisableDirectPortCardControl */
int fib_ControlClock1Enable(VISION_SLAVE slave);                /* fib_MasterEnableControlClock1 */
int fib_ControlClock1Disable(VISION_SLAVE slave);               /* fib_MasterDisableControlClock1 */
int fib_ControlClock2Enable(VISION_SLAVE slave);                /* fib_MasterEnableControlClock2 */
int fib_ControlClock2Disable(VISION_SLAVE slave);               /* fib_MasterDisableControlClock2 */
int fib_PortCardControlRegWrite(VISION_SLAVE slave, unsigned int* buffer); /* fib_WriteVMEControlRegister */
/* Do we need read-back added to production version ? */

/* Clock Shaper / Controller (Bank 5) */

int fib_TotalDigitizeRegRead(VISION_SLAVE slave, unsigned int* buffer);  /* fib_ReadTotalDigitizeRegister */
int fib_TotalDigitizeRegWrite(VISION_SLAVE slave, unsigned int* buffer);/* fib_WriteTotalDigitizeRegister */
int fib_MaxReadoutRegRead(VISION_SLAVE slave, unsigned int* buffer);  /* fib_ReadMaxReadoutRegister */
int fib_MaxReadoutRegWrite(VISION_SLAVE slave, unsigned int* buffer);  /* fib_WriteMaxReadoutRegister */
int fib_AccelModeRegRead(VISION_SLAVE slave, unsigned int* buffer);    /* fib_ReadAccelModeRegister */
int fib_Accel396NsMode(VISION_SLAVE slave);                            /* fib_SetFibTo396NsMode */
int fib_Accel132NsMode(VISION_SLAVE slave);                            /* fib_SetFibTo132NsMode */

/* Read-back Controller (Bank 6) */

int fib_DoimOeRegRead(VISION_SLAVE slave, unsigned int* buffer);        /* fib_ReadDoimOeReg */
int fib_DoimOutputEnable(VISION_SLAVE slave);                           /* fib_EnableDoimOutput */
int fib_DoimOutputDisable(VISION_SLAVE slave);                          /* fib_DisableDoimOutput */
int fib_DownloadCountRegRead(VISION_SLAVE slave, unsigned int* buffer);           /* fib_ReadDownloadCountReg */
int fib_DownloadCountRegWrite(VISION_SLAVE slave, unsigned int* buffer);/* fib_WriteDownloadCountReg */
int fib_ReadBackSelectRegRead(VISION_SLAVE slave, unsigned int* buffer);/* fib_GetReadBackSelectReg */
int fib_ReadBackSelectRegWrite(VISION_SLAVE slave, unsigned int* buffer);/* fib_SetReadBackSelectReg */
int fib_SerialReadBack(VISION_SLAVE slave, unsigned int* buffer);       /* fib_ReadSerialReadBackValue */
int fib_UploadFIFOSelectRegRead(VISION_SLAVE slave, unsigned int* buffer);/* fib_ReadUploadFIFOSelectReg */
int fib_UploadSelectDoim(VISION_SLAVE slave);                           /* fib_SetUploadSelectDoimData */
int fib_UploadSelectVme(VISION_SLAVE slave);                            /* fib_SetUploadSelectVmeData */
int fib_UploadFifoWrite(VISION_SLAVE slave, unsigned int* buffer);      /* fib_WriteUploadFifoData */
int fib_SerialFifosReset(VISION_SLAVE slave);                           /* fib_ResetSerialFifos */
int fib_DownloadFifoWrite(VISION_SLAVE slave, unsigned int* buffer);    /* fib_WriteDownloadFifo */
int fib_UploadFifoRead(VISION_SLAVE slave, unsigned int* buffer);       /* fib_ReadUploadFifo */
int fib_DownloadFifoClock(VISION_SLAVE slave);                          /* fib_ClockDownloadFifo */
int fib_UploadFifoClock(VISION_SLAVE slave);                            /* fib_ClockUploadFifo */
int fib_DOIMRead(VISION_SLAVE slave, int hdi_number, unsigned int* buffer);/* fib_ReadDOIMData */

/* FIFO controller (Bank 7) */

int fib_FifoControllerCsrRead(VISION_SLAVE slave,
                                unsigned int* vme_ctrl_req,
                                unsigned int* vme_ctrl_grant,
                                unsigned int* pipe_tristate);           /* fib_ReadFifoControllerCsr */
int fib_VmeFifoControlRequest(VISION_SLAVE slave);                      /* fib_RequestVmeFifoControl */
int fib_VmeFifoControlVerify(VISION_SLAVE slave, unsigned int* buffer); /* fib_VerifyVmeFifoControlGrant */
int fib_VmeFifoControlRelease(VISION_SLAVE slave);                      /* fib_ReleaseVmeFifoControl */
int fib_PipelineTristateEnable(VISION_SLAVE slave);                     /* fib_EnablePipelineTristate */
int fib_PipelineTristateDisable(VISION_SLAVE slave);                    /* fib_DisablePipelineTristate */
int fib_InputFifosWrite(VISION_SLAVE slave, unsigned int* buffer);      /* fib_WriteInputFifos */
int fib_ForceDataProcessing(VISION_SLAVE slave);                        /* fib_ForceDataProcessing */
int fib_InputFifosReset(VISION_SLAVE slave);                            /* fib_ResetAllInputFifos */
int fib_HeaderARead(VISION_SLAVE slave, unsigned int* buffer);          /* fib_ReadHeaderA */
int fib_HeaderAWrite(VISION_SLAVE slave, unsigned int* buffer);         /* fib_WriteHeaderA */
```

```
int fib_HeaderBRead(VISION_SLAVE slave, unsigned int* buffer);                    /* fib_ReadHeaderB */
int fib_HeaderBWrite(VISION_SLAVE slave, unsigned int* buffer);                   /* fib_WriteHeaderB */
int fib_BunchCrossingRegRead(VISION_SLAVE slave, unsigned int* buffer);           /* fib_ReadBunchCrossingRegister */
int fib_BEStateRegRead(VISION_SLAVE slave, unsigned int* buffer);         /* fib_ReadBEStateRegister */

/* FIFO controller (Bank 8) */

//Read G-Link Controller CSR (0x00)
//Enable VME Control Links 0-1
//Force FF0 transmission Links 0-1

int fib_GlinkControllerCsrRead(VISION_SLAVE slave, unsigned int* vme_enable_01,
                                        unsigned int* vme_enable_23,
                                        unsigned int* force_ffz_01,
                                        unsigned int* force_ffz_23);/* fib_ReadGlinkControllerCsr */
int fib_Glink01VmeControlEnable(VISION_SLAVE slave);                              /* fib_VmeEnableGlink01 */
int fib_Glink01VmeControlDisable(VISION_SLAVE slave);                             /* fib_VmeDisableGlink01 */
int fib_Glink23VmeControlEnable(VISION_SLAVE slave);                              /* fib_VmeEnableGlink23 */
int fib_Glink23VmeControlDisable(VISION_SLAVE slave);                             /* fib_VmeDisableGlink23 */
int fib_Glink01FillFrameZeroHold(VISION_SLAVE slave);                       /* fib_VmeForceFfz01 */
int fib_Glink01FillFrameZeroRelease(VISION_SLAVE slave);              /* fib_VmeReleaseFfz01 */
int fib_Glink23FillFrameZeroHold(VISION_SLAVE slave);                       /* fib_VmeForceFfz23 */
int fib_Glink23FillFrameZeroRelease(VISION_SLAVE slave);              /* fib_VmeReleaseFfz23 */
int fib_GlinkOutputFifoRead(VISION_SLAVE slave, unsigned int hdi_number,
                                        unsigned int* buffer); /* fib_ReadGlinkOutputFifo */

/* Bank 9 commands */

int fib_MsCommandLoad(VISION_SLAVE slave, int command_number,
                                        char *command_file_name); /* fib_LoadCommand */
int fib_MsRamBlockWrite(VISION_SLAVE slave, int RamNum, unsigned int* buffer);  /* fib_FillMsRam */
int fib_MsRamBlockRead(VISION_SLAVE slave, int RamNum, unsigned int* buffer); /* fib_CheckFillMsRam */
int fib_MsNVRamStore(VISION_SLAVE slave);                                   /* fib_StoreNVRam */
int fib_MsNVRamRecall(VISION_SLAVE slave, unsigned int *buffer); /* fib_RecallNVRam */

/* Pedestal RAMs (Bank 10 - Bank 19) */

int fib_PedRamRead(VISION_SLAVE slave,int RamNum, unsigned int offset,
                                        unsigned int* buffer); /* fib_ReadPedRam */
int fib_PedRamWrite(VISION_SLAVE slave,int RamNum, unsigned int offset,
                                        unsigned int* buffer); /* fib_WritePedRam */
int fib_PedRamBlockWrite(VISION_SLAVE slave,int RamNum, unsigned int* array);   /* fib_FillPedRam */
int fib_PedRamBlockRead(VISION_SLAVE slave,int RamNum, unsigned int* array);/* fib_CheckFillPedRam */
//int fib_WritePedValue(VISION_SLAVE slave,int RamNum, int ChipId,
                                        //int ChanNum, int BeState);
//int fib_ReadPedValue(VISION_SLAVE slave,int RamNum, int ChipId,
                                        //int ChanNum, int BeState);

/* Calculation RAMs (Bank 20 - Bank 29) */

int fib_CalcRamRead(VISION_SLAVE slave,int RamNum,unsigned int offset,
                                        unsigned int* buffer); /* fib_ReadCalcRam */
int fib_CalcRamWrite(VISION_SLAVE slave,int RamNum,unsigned int offset,
                                        unsigned int* buffer); /* fib_WriteCalcRam */
int fib_CalcRamBlockWrite(VISION_SLAVE slave,int RamNum, unsigned int* array);/* fib_FillCalcRam */
int fib_CalcRamBlockRead(VISION_SLAVE slave,int RamNum, unsigned int* array);/* fib_CheckFillCalcRam */
//int fib_FillSpecCodesCalcRam(VISION_SLAVE slave, int RamNum);

/* Utility functions */

int fib_HexToInt(char *);
int fib_BinToInt(char *);
unsigned int fib_HexToUInt(char *);
int fib_ErrorString(int,char *);              /* error number,return string   fib_GetErrorString */

unsigned int fib_BinToGray(unsigned int bn);
unsigned int fib_GrayToBin(unsigned int gn);
```

```
int fib_0to9(int input);            /* returns 0 if in 0to9 range else returns 1 */
```

4. INTERFACE SPECIFICATIONS

The FIB board has the following interfaces, which we will now describe:

- With the Port Card, through the J3 connector.
- With the SRC through the specially designed J3 backplane.
- With VMEbus through the J1,J0,J2 backplane.
- With G-Link transmitters boards, through on board connectors, which transmit data to the VRB and SVT boards.

4.1 VMEbus Interface

The TFIB is a VMEbus slave which implements A32:D32 addressing. Block transfers are not implemented. The FIB does not generate VMEbus errors neither VMEbus interrupts. The VME interface has the following characteristics:

- The FIB board is always a VMEbus slave. The FIB board has no VME master capability.
- The VME interface is implemented through the J1 and J2 connectors.
- Only a 32 bit addressing scheme is allowed.
- The VME interface only executes single data transfers.
- The VME interface allows only 32 bit data transfers.
- The VME interface does not generate interruptions.
- The VME interface generates VMEbus errors.

4.1.1 Addressing Modes

The TFIB is addressed using a 32 bit address mode and the VME Address Modifiers either 0x09 or 0x0A The base address has a fixed pattern, set by a DIP switch, and a variable one, which depends if the board is connected to a VME-P type J1 backplane. We will call Addressing Mode 1 when the board is connected to the VME-P type J1 backplane, and Addressing Mode 2 when it is not.

If the board is connected to a VME-P type J1 backplane (Addressing Mode 1), the address lines A[31:27] are compared with the geographic address supplied by the backplane. Table 4 shows the relationship between A[31:27] and GA[4:0]*. When the higher order bits of the VME address, A[31:27], compare with the six bits GAP*,GA[4:0]*, the FIB recognizes that it is being addressed by the VMEbus CPU board and performs the requested action. Lower address bits are used to address both memories segments and control registers within the FIB.

GEOGRAPHIC ADDRESS	ADDRESS LINE
GAP*	not used
GA4*	A31
GA3*	A30
GA2*	A29
GA1*	A28
GA0*	A27

Table 4. Geographic Address And Address Line Relationships

If the board is *NOT* connected to a VME-P type J1 backplane (Addressing Mode 2), the address lines A[31:27] are compared with the setting of DIP switch S1. Bits are set to zero by having the switch in the on position. Bits are set to one by having the switch in the off position. When the higher order 5 its of the VME address, A[31:27], compare with the DIP switch positions, the FIB recognizes that it is being addressed by the VMEbus CPU board and performs the requested action.

4.1.2 Data Cycles Types

The FIB only implements 32 bit data transfers. The following section contains a general description of the FIB address map followed by a detailed description of each FIB register and/or memory location.

4.1.3 FIB VME Register/Memory Descriptions

4.1.3.1 General FIB VME Address Map

Board Sub-Section	Bank Number	Base Offset
VME Interface	0	0x000000
General Status	1	0x040000
Micro-sequencer 1 - Command Processor	2	0x080000
Micro-sequencer 2 - Address Controller	3	0x0C0000
Micro-sequencer 3 - Output Controller	4	0x100000
Clock Shaper	5	0x140000
Read-back Controller	6	0x180000
FIFO Controller	7	0x1C0000
G-Link Controller	8	0x200000
Micro-sequencer RAM	9	0x240000
Pedestal RAM 0	10	0x400000
Pedestal RAM 1	11	0x440000
Pedestal RAM 2	12	0x480000
Pedestal RAM 3	13	0x4C0000
Pedestal RAM 4	14	0x500000
Pedestal RAM 5	15	0x540000
Pedestal RAM 6	16	0x580000
Pedestal RAM 7	17	0x5C0000
Pedestal RAM 8	18	0x600000
Pedestal RAM 9	19	0x640000
Calculation RAM 0	20	0x800000
Calculation RAM 1	21	0x840000
Calculation RAM 2	22	0x880000
Calculation RAM 3	23	0x8C0000
Calculation RAM 4	24	0x900000
Calculation RAM 5	25	0x940000
Calculation RAM 6	26	0x980000
Calculation RAM 7	27	0x9C0000
Calculation RAM 8	28	0xA00000
Calculation RAM 9	29	0xA40000
Pipeline Controller 0	30	0xC00000
Pipeline Controller 1	31	0xC40000
Pipeline Controller 2	32	0xC80000
Pipeline Controller 3	33	0xCC0000
Pipeline Controller 4	34	0xD00000
Pipeline Controller 5	35	0xD40000
Pipeline Controller 6	36	0xD80000
Pipeline Controller 7	37	0xDC0000
Pipeline Controller 8	38	0xE00000
Pipeline Controller 9	39	0xE40000

4.1.3.2 FIB Sub-section register/memory summary

This is a brief summary of the registers/memory in each of the address banks of the FIB.

- VME Interface Registers:
  - Board ID register
  - JTAG2 control register
  - VME clock status
- General Status Register:
  - General Status Register
- Micro-sequencer Controller 1/ Command Processor:
  - Micro-sequencer control registers
- Micro-sequencer Controller 2/ Address Controller:
  - Micro-sequencer control registers
- Micro-sequencer Controller 3/ Output Controller:
  - Output Latch control registers
  - Manual control registers
- Clock Shaper/Controller:
  - BE_CLK maximum count registers
- Read-back Controller:
  - Download FIFO controls
  - Upload FIFO controls
  - Port Card Read-back control
- FIFO Controller:
  - HDI ID registers
  - Bunch Crossing register
  - Back-end State register
  - Input FIFO control registers
  - Manual write control registers
  - Pipeline control registers
- G-Link Control registers:
  - G-Link Control registers
  - Output FIFO readout
  - Channel Mask registers
- Micro-sequencer RAM:
  - 8K x 32 Non-volatile Micro-sequencer memory
- Pedestal SRAM:
  - 32K x 8 volatile Pedestal memory
- Calculation SRAM:
  - 32K x 8 volatile Calculation memory
- Pipeline Controllers:
  - Pipeline ID register
  - Data Timeout registers

4.1.3.3 VME Register Maps

FIB - VME Register Map(Section 1/4)

Register	Identifier	Access	Address Offset
VME Interface **(4 registers)**	(**Bank 0**)		*Starting Offset: 0x000000*
ID register (32 bits) Bit 31:24 FIB ID = 02 Bit 23:16 PCB Version Bit 15:08 ECO Level Bit 07:00 Serial Number	(Bank0, r0)	read only	*0x000000*
JTAG Register (5 bits) Bit 0 - TDO Bit 1 - TCLK Bit 2 - TMS Bit 3 - TDI Bit 4 - JTAG enable (1=enable)	(Bank0, r1) (Bank0,r1,b0) (Bank0,r1,b1) (Bank0,r1,b2) (Bank0,r1,b3) (Bank0,r1,b4)	 read only read/write read/write read/write read/write	*0x000004*
Clock Status Register (1 bit) Bit 0 - Master Clock Present (1=clock present)	(Bank0, r2)	read only	*0x000008*
Board Reset (n/a)	(Bank0, r3)	write only	*0x00000C*
General Status Register **(1 register)**	(**Bank 1**)		*Starting Offset: 0x040000*
General Status Register (32 bit) Bit 0 - G-Link Lock 0 Bit 1 - G-Link Lock 1 Bit 2 - G-Link Lock 2 Bit 3 - G-Link Lock 3 Bit 4 - CMD FIFO WERR Bit 5 - Latch Status	(Bank1, r0)	read only	*0x040000*
FIB Micro-sequencer 1 / Command Processor **(9 registers)**	(**Bank2**)		*Starting Offset: 0x080000*
VME Control Register (2 bits) Bit 0 - VME Command Enable (1=enable) Bit 1 - Manual Strobe Enable (1=enable) Bit 2 - Command Processing on Overflow (1= Continue on Cmd FIFO full) Bit 3 - FIB Busy	(Bank2, r0)	 read / write read / write read / write read only	*0x080000*
VME Command Register (8 bits)	(Bank2, r1)	read /write	*0x080004*
Manual Strobe Register *Note: must be enabled by (Bank2,r0,b1)*	(Bank2, r2)	write only	*0x080008*
Clear Latch Status	(Bank2, r3)	write only	*0x08000C*
Reset Command FIFO	(Bank2, r4)	write only	*0x080010*
Reset FIL FIFO	(Bank2, r5)	write only	*0x080014*
Command FIFO (9 bits) Bit 7:0 - Data Bit 8 - Empty Flag (0=empty) *Note: Command Processing must be disabled (Bank3,r1,b0)*	(Bank2, r8)	read only	*0x080020*
FIL FIFO (9 bits) Bit 7:0 - Data Bit 8 - Empty Flag (0=empty)	(Bank2, r9)	read only	*0x080024*

FIB - VME Register Map(Section 2/4)

Register	Identifier	Access	Address Offset
FIB Micro-sequencer 2 / Address Controller **(2 registers)**	(**Bank3**)		*Starting Offset: 0x0C0000*
Last Command Processed Register (8 bits)	(Bank3, r0)	read only	*0x0C0000*
Micro Sequencer Control Register (2 bits) Bit 0 - Enable Command Processing Bit 1 - VME address asserted	(Bank3, r1) (Bank3, r1, b0) (Bank3, r1, b1)	 read/write read only	*0x0C0004*
FIB Micro-sequencer 3 / Output Controller **(2 registers)**	(**Bank4**)		*Starting Offset: 0x100000*
VME control register (1 bit) Bit 0 - VME control Enable Bit 1 - Enable Control Clock 1 Bit 2 - Enable Control Clock 2	(Bank4, r0) (Bank4,r0,b0) (Bank4,r0,b1) (Bank4,r0,b2)	read / write	*0x100000*
VME command register (12 bits) Bit 0 - Command 0 Bit 1 - Command 1 Bit 2 - Command 2 Bit 3 - Command 3 Bit 4 - Command 4 Bit 5 - Command 5 Bit 6 - Command Data (C_Data)		write only	*0x100004*
Clock Shaper / Controller **(3 registers)**	(**Bank5**)		*Starting Offset: 0x140000*
Maximum Digitize Count Register (14 bit) *Note: value is written in Gray Code*	(Bank5, r0)	read / write	*0x140000*
Maximum Readout Count Register (14 bit) *Note: value is written in Gray Code*	(Bank5, r1)	read / write	*0x140004*
Accelerator Mode Register (1 bit) Bit 0 - Accelerator Mode (0 = 396ns mode, default) (1 = 132ns mode) *Note: This bit does not currently do anything* Bit 1 – BE Clock hold on Mode (0 = Normal, default) (1 = BE Hold On Mode) *.Note: This mode has never been tested.*	(Bank5, r3) (Bank5,r3,b0)	read / write	*0x140008*

FIB - VME Register Map(Section 3/4)

Register	Identifier	Access	Address Offset
Read-back Controller **(19 registers)**	(**Bank6**)		*Starting Offset: 0x180000*
DOIM output enable control register (1 bit) Bit 0 - Enable control bit (1=Output enabled/manual)	(Bank6, r0) (Bank6,r0,b0)	read/write	*0x180000*
Max. Download bit count register (12 bit) *Note: value is written in Gray Code.* *See r10 for 13th bit added for Run2b.*	(Bank6, r1)	read/write	*0x180004*
BN select register (4 bit) (0-9 = BNRB0-9) (15 = Download FIFO)	(Bank6, r2)	read/write	*0x180008*
BN read-back register (1 bit) Bit 0 - BN read back *Note: Read-back of data selected by register (Bank6,r2)*	(Bank6, r3) (Bank6,r3,b0)	read only	*0x18000C*
Upload FIFO data select register (1 bit) Bit 0 - Read-back FIFO select bit (0 = DOIM data) (1 = VME data)	(Bank6, r4) (Bank6,r4,b0)	read/write	*0x180010*
VME write to upload FIFO register (1 bit) Bit 0 - VME write test bit	(Bank6, r5) (Bank6,r5,b0)	write only	*0x180014*
Reset FIFO register (0 bit) *Note: resets download and upload registers*	(Bank6, r6)	write only	*0x180018*
Download FIFO write register (8 bit)	(Bank6, r7)	write only	*0x18001C*
Read-back FIFO read register (8 bit)	(Bank6, r8)	read only	*0x180020*
Test read clock for Download FIFO	(Bank6, r9)	write only	*0x180024*
Max. Download bit count register (13th bit) *Note: value is written in Gray Code.* *This is the MSB added for Runb2.*	(Bank6, r10)	read/write	*0x180028*
DOIM read-back registers 0-9 (4 bit)	(Bank6, r16) - (Bank6, r25)	read only	*0x180040 -* *0x180064*
FIFO Controller **(19 registers)**	(**Bank7**)		*Starting Offset: 0x1C0000*
Control Status Register (3 bits) Bit 0 - VME control request Bit 1 - VME control granted Bit 2 - Pipeline tri-state control (1=Pipeline tri-stated)	(Bank7, r0) (Bank7,r0,b0) (Bank7,r0,b1) (Bank7,r0,b2)	 read/write read only read/write	*0x1C0000*
Write FIFO register (8 bits)	(Bank7, r1)	write only	*0x1C0004*
Force processing of data (0 bits)	(Bank7, r2)	write only	*0x1C0008*
Reset FIFO register (0 bits) *Note: this resets all input FIFOs*	(Bank7, r3) (Bank7,r3,b0)	write only	*0x1C000C*
Header A Register (8 bits)	(Bank7, r4)	read/write	*0x1C0010*
Header B Register (8 bits)	(Bank7, r5)	read/write	*0x1C0014*
Bunch Crossing Register (8 bits)	(Bank7, r6)	read only	*0x1C0018*
Back-end State/Tag Register (8 bits) Bit 0-1 = Back-end state Bits 2-7 = Programmable ID tag	(Bank7, r7)	 read only read/write	*0x1C001C*

FIB - VME Register Map(Section 4/4)

Register	Identifier	Access	Address Offset
G-Link Controller **(11 registers)**	(**Bank8**)		*Starting Offset: 0x200000*
Control Status Register (0 bits) Bit 0 - VME control G-Link 0/1 enable Bit 1 - VME control G-Link 2/3 enable (1=VME enable) Bit 2 - FF0 G-Link 0/1 enable Bit 3 - FF0 G-Link 2/3 enable (1=FF0 enable)	(Bank8, r0) (Bank8,r0,b0) (Bank8,r0,b1) (Bank8,r0,b2) (Bank8,r0,b3)	 read/write read/write read/write read/write	*0x200000*
Note 1: To read G-Link output FIFOs, enables in (Bank8,r0,b0) and/or (Bank8,r0,b1) must first be set. *Note 2: Read G-Link Registers map as follows:* Bit 7-0 – Data Bit 8 – FIFO Empty Flag (0 = empty)	 (b7-b0) (b8)	 read only read only	
Read G-Link output FIFO 0	(Bank8, r1)	read only	*0x200004*
Read G-Link output FIFO 1	(Bank8, r2)	read only	*0x200008*
Read G-Link output FIFO 2	(Bank8, r3)	read only	*0x20000C*
Read G-Link output FIFO 3	(Bank8, r4)	read only	*0x200010*
Read G-Link output FIFO 4	(Bank8, r5)	read only	*0x200014*
Read G-Link output FIFO 5	(Bank8, r6)	read only	*0x200018*
Read G-Link output FIFO 6	(Bank8, r7)	read only	*0x20001C*
Read G-Link output FIFO 7	(Bank8, r8)	read only	*0x200020*
Read G-Link output FIFO 8	(Bank8, r9)	read only	*0x200024*
Read G-Link output FIFO 9	(Bank8, r10)	read only	*0x200028*
Mask A Register Bit 3-0 = FIFO 3-0 *Note: No longer used disables are in now pipeline registers.*	(Bank8, r11)	read/write	*0x20002C*
Mask B Register Bit 3-0 = FIFO 7-4 *Note: No longer used disables are in now pipeline registers*	(Bank8, r12)	read/write	*0x200030*
Mask C Register Bit 1-0 = FIFO 9-8 *Note: No longer used disables are in now pipeline registers*	(Bank8, r13)	read/write	*0x200034*
Reset Output FIFOs	(Bank8, r14)	write only	*0x200038*

4.1.3.4 VME Memory Maps

Draft Note: The sections describing the memory organization will be included in the sections following this table.

FIB - VME Memory Map(Section 1/2)

Register	Identifier	Access	Address Offset
Micro-sequencer RAM 8K x 32 memory (32 bits) Bit 0 - Command 0 Bit 1 - Command 1 Bit 2 - Command 2 Bit 3 - Command 3 Bit 4 - Command 4 Bit 5 - Command 5 Bit 6 - Command Data Bit 7 - Command Done Bit 8 - Command Loop Start Bit 9 - Command Loop End Bit 10 - External Command Data Bit 11 - Download FIFO Read Clock Bit 12 - Upload FIFO Write Clock Bit 13 - Micro-sequencer Latch Enable Bit 14 - Abort Acknowledge Bit 15 - Enable Calibration Inject Bit 16 - Enable Control Clock Bit 17 - 1=Readout/0=Digitize Bit 18 - Mask Interrupt Bit 19 - BE_CLK High Bit 20 - BE_CLK Low Bit 21 - BE_CLK Enable Bit 22 - L1A_High Bit 23 - L1A_Low Bit 24 - L1A_Enable Bit 25 - PRD2_High Bit 26 - PRD2_Low Bit 27 - PRD2_Enable Bit 28 - FE_High Bit 29 - FE_Low Bit 30 - FE_Enable Bit 31 - Pause (Bit 0-31, 1=active) *Note: Bit 17 is used to qualify Bit 21, i.e. to select the correct BE_CLK max. count..*	(Bank9)	read/write	*Starting Offset: 0x240000*

FIB - VME Memory Map(Section 2/2)

Register	Identifier	Access	Address Offset
Pedestal RAMs 32K x 8 SRAM (8 bits) Bit 0:5 - Pedestal 0:5 Bit 6 - reserved Bit 7 - unused or parity	(Bank10)- (Bank19)	read/write	*Starting Offset: 0x400000*
Pedestal RAM 0	(Bank10)	read/write	*Starting Offset: 0x400000*
Pedestal RAM 1	(Bank11)	read/write	*Starting Offset: 0x440000*
Pedestal RAM 2	(Bank12)	read/write	*Starting Offset: 0x480000*
Pedestal RAM 3	(Bank13)	read/write	*Starting Offset: 0x4C0000*
Pedestal RAM 4	(Bank14)	read/write	*Starting Offset: 0x500000*
Pedestal RAM 5	(Bank15)	read/write	*Starting Offset: 0x540000*
Pedestal RAM 6	(Bank16)	read/write	*Starting Offset: 0x580000*
Pedestal RAM 7	(Bank17)	read/write	*Starting Offset: 0x5C0000*
Pedestal RAM 8	(Bank18)	read/write	*Starting Offset: 0x600000*
Pedestal RAM 9	(Bank19)	read/write	*Starting Offset: 0x640000*
Calculation RAMs 32K x 8 SRAM (8 bits) Bit 0:7 - Output data	(Bank20) - (Bank29)	read/write	*Starting Offset: 0x800000*
Calculation RAM 0	(Bank20)	read/write	*Starting Offset: 0x800000*
Calculation RAM 1	(Bank21)	read/write	*Starting Offset: 0x840000*
Calculation RAM 2	(Bank22)	read/write	*Starting Offset: 0x880000*
Calculation RAM 3	(Bank23)	read/write	*Starting Offset: 0x8C0000*
Calculation RAM 4	(Bank24)	read/write	*Starting Offset: 0x900000*
Calculation RAM 5	(Bank25)	read/write	*Starting Offset: 0x940000*
Calculation RAM 6	(Bank26)	read/write	*Starting Offset: 0x980000*
Calculation RAM 7	(Bank27)	read/write	*Starting Offset: 0x9C0000*
Calculation RAM 8	(Bank28)	read/write	*Starting Offset: 0xA00000*
Calculation RAM 9	(Bank29)	read/write	*Starting Offset: 0xA40000*
Last Chip ID Registers 10 Registers (8 bits) Bit 0:7 - Last Chip ID			
Pipeline 0 Last Chip ID	(Bank30, r1)	read/write	*Starting Offset: 0xC00000*
Pipeline 1 Last Chip ID	(Bank31, r1)	read/write	*Starting Offset: 0xC40000*
Pipeline 2 Last Chip ID	(Bank32, r1)	read/write	*Starting Offset: 0xC80000*
Pipeline 3 Last Chip ID	(Bank33, r1)	read/write	*Starting Offset: 0xCC0000*
Pipeline 4 Last Chip ID	(Bank34, r1)	read/write	*Starting Offset: 0xD00000*
Pipeline 5 Last Chip ID	(Bank35, r1)	read/write	*Starting Offset: 0xD40000*
Pipeline 6 Last Chip ID	(Bank36, r1)	read/write	*Starting Offset: 0xD80000*
Pipeline 7 Last Chip ID	(Bank37, r1)	read/write	*Starting Offset: 0xDC0000*
Pipeline 8 Last Chip ID	(Bank38, r1)	read/write	*Starting Offset: 0xE00000*
Pipeline 9 Last Chip ID	(Bank39, r1)	read/write	*Starting Offset: 0xE40000*

Register	Identifier	Access	Address Offset
Pipeline Disable Registers 10 Registers (1 bits) Bit 0 - (0=Enabled, 1 = Disabled)			
Pipeline 0 Last Chip ID	(Bank30, r2)	read/write	*Starting Offset: 0xC00004*
Pipeline 1 Last Chip ID	(Bank31, r2)	read/write	*Starting Offset: 0xC40004*
Pipeline 2 Last Chip ID	(Bank32, r2)	read/write	*Starting Offset: 0xC80004*
Pipeline 3 Last Chip ID	(Bank33, r2)	read/write	*Starting Offset: 0xCC0004*
Pipeline 4 Last Chip ID	(Bank34, r2)	read/write	*Starting Offset: 0xD00004*
Pipeline 5 Last Chip ID	(Bank35, r2)	read/write	*Starting Offset: 0xD40004*
Pipeline 6 Last Chip ID	(Bank36, r2)	read/write	*Starting Offset: 0xD80004*
Pipeline 7 Last Chip ID	(Bank37, r2)	read/write	*Starting Offset: 0xDC0004*
Pipeline 8 Last Chip ID	(Bank38, r2)	read/write	*Starting Offset: 0xE00004*
Pipeline 9 Last Chip ID	(Bank39, r2)	read/write	*Starting Offset: 0xE40004*
No Data Timeout Registers 10 Registers (8 bits) Bit 0:7 - Number of clocks to wait for data before truncating the channel.			
Pipeline 0 Last Chip ID	(Bank30, r3)	read/write	*Starting Offset: 0xC00004*
Pipeline 1 Last Chip ID	(Bank31, r3)	read/write	*Starting Offset: 0xC40004*
Pipeline 2 Last Chip ID	(Bank32, r3)	read/write	*Starting Offset: 0xC80004*
Pipeline 3 Last Chip ID	(Bank33, r3)	read/write	*Starting Offset: 0xCC0004*
Pipeline 4 Last Chip ID	(Bank34, r3)	read/write	*Starting Offset: 0xD00004*
Pipeline 5 Last Chip ID	(Bank35, r3)	read/write	*Starting Offset: 0xD40004*
Pipeline 6 Last Chip ID	(Bank36, r3)	read/write	*Starting Offset: 0xD80004*
Pipeline 7 Last Chip ID	(Bank37, r3)	read/write	*Starting Offset: 0xDC0004*
Pipeline 8 Last Chip ID	(Bank38, r3)	read/write	*Starting Offset: 0xE00004*
Pipeline 9 Last Chip ID	(Bank39, r3)	read/write	*Starting Offset: 0xE40004*
Repeating Data Timeout Registers 10 Registers (3 bits) Bit 0:2 - Number of clocks to wait when the data repeats before truncating the channel.			
Pipeline 0 Last Chip ID	(Bank30, r4)	read/write	*Starting Offset: 0xC00004*
Pipeline 1 Last Chip ID	(Bank31, r4)	read/write	*Starting Offset: 0xC40004*
Pipeline 2 Last Chip ID	(Bank32, r4)	read/write	*Starting Offset: 0xC80004*
Pipeline 3 Last Chip ID	(Bank33, r4)	read/write	*Starting Offset: 0xCC0004*
Pipeline 4 Last Chip ID	(Bank34, r4)	read/write	*Starting Offset: 0xD00004*
Pipeline 5 Last Chip ID	(Bank35, r4)	read/write	*Starting Offset: 0xD40004*
Pipeline 6 Last Chip ID	(Bank36, r4)	read/write	*Starting Offset: 0xD80004*
Pipeline 7 Last Chip ID	(Bank37, r4)	read/write	*Starting Offset: 0xDC0004*
Pipeline 8 Last Chip ID	(Bank38, r4)	read/write	*Starting Offset: 0xE00004*
Pipeline 9 Last Chip ID	(Bank39, r4)	read/write	*Starting Offset: 0xE40004*

4.1.3.5 Memory Organization

Draft Note: These sections are currently being written.

4.1.3.5.1 Micro-sequencer RAM

Bit Number - Name	Usage	Comments
0 - Command 0	DDR-D Flip-Flop selection bit	
1 - Command 1	DDR-D Flip-Flop selection bit	
2 - Command 2	DDR-D Flip-Flop selection bit	
3 - Command 3	DDR-D Flip-Flop selection bit	
4 - Command 4	DDR-D Flip-Flop selection bit	
5 - Command 5	DDR-D Flip-Flop selection bit	Not used by current DDR chip
6 - Command Data	DDR-D Flip-Flop set or reset	0 = reset, 1= set
7 - Command Done	End Micro-sequencer command	Last address is 1 or 2 past marking point (verify!)
8 - Command Loop Start	Mark beginning of internal loop in Micro-sequencer command	1) Starting address is 2 past marking point (verify!) 2) Loop terminated by completion of trigger task: • 21 - BE_CLK Enable (BE done - counter) • 15 - Enable Calibration Inject (CAL inject done) • 10 - External Command Data (Initialization done - counter) 3) Must be 3 addresses away from Loop End (verify!)
9 - Command Loop End	Mark ending of internal loop in Micro-sequencer command	1) Starting address is 2 past marking point (verify!) 2) Loop terminated by completion of trigger task: • 21 - BE_CLK Enable (BE done) • 15 - Enable Calibration Inject (CAL inject done) • 10 - External Command Data (Initialization done) 3) Must be 3 addresses away from Loop Start (verify!)
10 - External Command Data	A) Triggers start of Initialization B) Replaces Command Data (bit 6) with output of Download Parallel to Serial FIFO	1. Issue before Loop Start

11 - Download FIFO Read Clock	Clocks data out of Download Parallel to Serial FIFO	Used during Initialization loop
12 - Upload FIFO Write Clock	Clocks data into Upload Serial to Parallel FIFO	Used during Initialization loop
13 - Micro-sequencer Latch Enable	Enables Commands and Data bits to change to the Port Card (puts output latch into transparent mode)	1. 0=latched, 1=transparent 2. Command clock should be enabled 1 or 2 cycles after output latch is enabled into transparent mode (verify!) 3. Command clock should be disabled 2 cycles before output latch is disabled.
14 - Abort Acknowledge (Reset)	Clears Abort FF which is set by SRC Abort Command	Always set at the end of Readout Command after loop
15 - Enable Calibration Inject	Triggers start of Calibration Inject - enables sychronizing circuitry.	Issued before loop start (wait loop required in Calibration command)
16 - Enable Control Clock	Enables DDR-D Flip-Flop clock	1. Command clock should be enabled 1 or 2 cycles after output latch is enabled into transparent mode (verify!) 2. Command clock should be disabled 2 cycles before output latch is disabled.
17 - Readout/Digitize	Selects load register for BE_CLK down counter If bit set to one (1) starts the data processing (inserts header into input FIFOs)	1. 1=Readout, 0=Digitize 2. Used in conjunction with BE_CLK Enable (bit 21) 3. Note set to 1, 6 cycles prior to BE_CLK Enable (verify!)
18 - Mask Interrupt	Prevents PA_START and PA_STOP interrupts	Used during charge transfer in the Digitize command
19 - BE_CLK High	Forces BE_CLK high	Overrides BE_CLK enable
20 - BE_CLK Low	Forces BE_CLK low	Overrides BE_CLK enable
21 - BE_CLK Enable	Enables BE_CLK for duration of BE_CLK down counter	Used in conjunction with Readout/Digitize (bit 17)
22 - L1A High	Forces L1A high	Overrides L1A enable
23 - L1A Low	Forces L1A low	Overrides L1A enable
24 - L1A Enable	Enables L1A	
25 - PRD2 High	Forces PRD2 high	Overrides PRD2 enable
26 - PRD2 Low	Forces PRD2 low	Overrides PRD2 enable
27 - PRD2 Enable	Enables PRD2	
28 - FE_CLK High	Forces FE_CLK high	Overrides FE_CLK enable
29 - FE_CLK Low	Forces FE_CLK low	Overrides FE_CLK enable
30 - FE_CLK Enable	Enables FE_CLK	
31 - Pause	Pause for one MCLK cycle	Used to align signal to MCLK

4.1.3.5.2 Pedestal RAM

4.1.3.5.3 Calculation RAM

4.2 SRC Interface

SRC communication is done through the FFO to the J3 backplane of the VME crate. The J3 backplane is a custom design for the VRB and FIB and has more lines available than the FIB requires [ref. **Error! Bookmark not defined.**].

The next sections describe the electrical and logical protocol of this interface.

4.2.1 J3 Backplane Control Connector

The following is a short description of the J3 backplane SRC/FFO interface signals that the FIB will use. For further details, refer to specific documentation [ref. **Error! Bookmark not defined.**]. The signals being distributed from the FFO are the following:

ECL Clock Signals

- The 53 MHz differential ECL clock signal named MCLK.
- A 32 ns wide, 132 ns period, "SYNC" pulse in differential ECL.

TTL Data Bus Signals (these change every 132 ns)

- Command(4:0), named CMD[4:0].
- Execute Immediate command modifier, name XQT.
- Advance Pipeline, named AD-PIPE which indicates when the SVX III pipeline is to advance.
- Level 1 Accept, named L1A which indicates that a particular physics event should be digitized.
- Beam/Bunch Crossing number [7:0], named BXING[7:0].
- Back-end Chip State[1:0], named RDQ[1:0].
- Pipeline Capacitor Return Signal, named PIPE_RD2, which returns an input cell to the pipeline.

Note: Power Connections

- J1, J0 and J2 provide +5.0 V DC, -5.2 V DC, and Ground. J3 does NOT supply power.

The table on the next page shows the pin assignment of the J3 VME backplane connector. This signals listed above are high-lighted.

FIB J3 Connector Pin-out - (rev F.1 - 10/29/96)

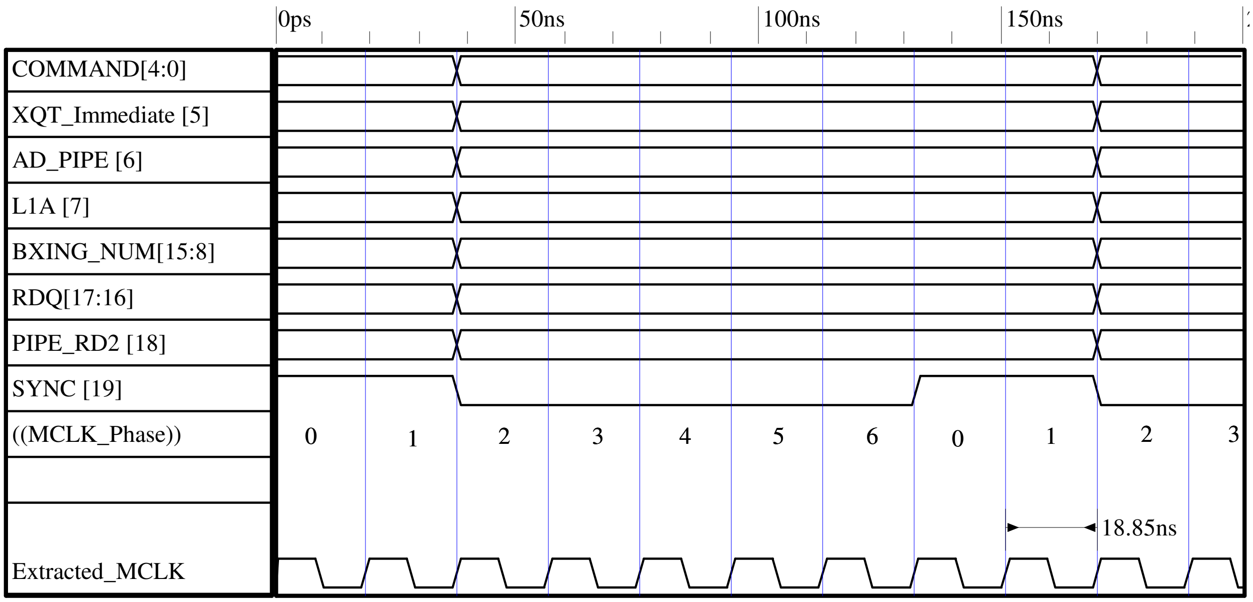
Pin #	Row 'a'	Row 'b'	Row 'c'	Row 'd'	Row 'e'	Row 'f'
1	GND	GND	**CMD[0]**	GND	GND	GND
2	B_MCLK	HDI-A-D0	**CMD[1]**	NC	HDI-B-D0	GND
3	GND	HDI-A-D1	**CMD[2]**	GND	HDI-B-D1	GND
4	B_AD_PIPE	HDI-A-D2	**CMD[3]**	NC	HDI-B-D2	GND
5	GND	HDI-A-D3	**CMD[4]**	GND	HDI-B-D3	GND
6	B_SYNC	HDI-A-D4	**XQT**	NC	HDI-B-D4	GND
7	GND	HDI-A-D5	**ADPIPE**	GND	HDI-B-D5	GND
8	FTM_CLK	HDI-A-D6	**L1A**	NC	HDI-B-D6	GND
9	GND	HDI-A-D7	**BXING[0]**	GND	HDI-B-D7	GND
10	NC	HDI-A-OBDV	**BXING[1]**	NC	HDI-B-OBDV	GND
11	GND	HDI-C-D0	**BXING[2]**	GND	HDI-D-D0	GND
12	NC	HDI-C-D1	**BXING[3]**	NC	HDI-D-D1	GND
13	GND	HDI-C-D2	**BXING[4]**	GND	HDI-D-D2	GND
14	NC	HDI-C-D3	**BXING[5]**	NC	HDI-D-D3	GND
15	GND	HDI-C-D4	**BXING[6]**	GND	HDI-D-D4	GND
16	NC	HDI-C-D5	**BXING[7]**	NC	HDI-D-D5	GND
17	GND	HDI-C-D6	**RDQ[0]**	GND	HDI-D-D6	GND
18	C0	HDI-C-D7	**RDQ[1]**	NC	HDI-D-D7	GND
19	GND	HDI-C-DVAL	**PRD2**	GND	HDI-D-DVAL	GND
20	C1	HDI-E-D0	**TTL_SYNC**	NC	HDI-F-D0	GND
21	GND	HDI-E-D1	**BUS21**	GND	HDI-F-D1	GND
22	C2	HDI-E-D2	**STAT0**	NC	HDI-F-D2	GND
23	GND	HDI-E-D3	**STAT1**	GND	HDI-F-D3	GND
24	C3	HDI-E-D4	**STAT2**	NC	HDI-F-D4	GND
25	GND	HDI-E-D5	**STAT3**	GND	HDI-F-D5	GND
26	C4	HDI-E-D6	RESERVED	NC	HDI-F-D6	GND
27	GND	HDI-E-D7	SYNC(n)	GND	HDI-F-D7	GND
28	C5	HDI-E-DVAL	/SYNC(n)	NC	HDI-F-DVAL	GND
29	GND	HDI-G-D0	RESERVED	GND	HDI-H-D0	GND
30	C_DATA	HDI-G-D1	MCLK(n)	NC	HDI-H-D1	GND
31	GND	HDI-G-D2	/MCLK(n	GND	HDI-H-D2	GND
32	C_CLK1	HDI-G-D3	RESERVED	NC	HDI-H-D3	GND
33	GND	HDI-G-D4	GND	GND	HDI-H-D4	GND
34	C_CLK2	HDI-G-D5	BNRB-A	NC	HDI-H-D5	GND
35	GND	HDI-G-D6	BNRB-B	GND	HDI-H-D6	GND
36	PIPE_RD2	HDI-G-D7	BNRB-C	NC	HDI-H-D7	GND
37	GND	HDI-G-DVAL	BNRB-D	GND	HDI-H-DVAL	GND
38	L1A	HDI-I-D0	BNRB-E	NC	HDI-J-D0	GND
39	GND	HDI-I-D1	BNRB-F	GND	HDI-J-D1	GND
40	FE_CLK1	HDI-I-D2	BNRB-G	NC	HDI-J-D2	GND
41	GND	HDI-I-D3	BNRB-H	GND	HDI-J-D3	GND
42	FE_CLK2	HDI-I-D4	BNRB-I	NC	HDI-J-D4	GND
43	GND	HDI-I-D5	BNRB-J	GND	HDI-J-D5	GND
44	BE_CLK1	HDI-I-D6	GND	NC	HDI-J-D6	GND
45	GND	HDI-I-D7	NC	GND	HDI-J-D7	GND
46	BE_CLK2	HDI-I-DVAL	NC	NC	HDI-J-DVAL	GND
47	GND	GND	NC	GND	GND	GND

Table 5. J3 backplane pin assignment

4.2.2 SRC to FFO G-Link Command and Timing Frame Structure and Error Detection

4.2.2.1 Frame Structure

The G-Link frame structure for the SRC to FFO is shown in the figure below.



The table below shows the G-Link frame data bits and their meaning. Frame position indicates position relative to the start of the SYNC signal.

SRC G-Link frame data bits	Number of Bits	Frame Position
COMMAND (SRC -> FFO Command)	5	2-1 (7 frames)
XQT (Execute Immediately)	1	2-1 (7 frames)
AD_PIPE (Advance Pipeline)	1	2-1 (7 frames)
L1A (Level 1 Accept)	1	2-1 (7 frames)
BXING_NUM (Bunch Crossing Number)	8	2-1 (7 frames)
RDQ (Back-end State Information)	2	2-1 (7 frames)
PIPE_RD2 (Return Pipeline Capacitor)	1	2-1 (7 frames)
SYNC (132 ns Clock)	1	0-1 (2 frames)
Total Number of Bits	20 bits	

4.2.2.2 Error Detection

To ensure that no commands are miss-interpreted from the SRC G-Link, a “voting sample” is taken from the data bits (all 20 bits) over three separate G-Link frames. The samples are taken during frames 6-1. If there are any discrepancies an error is reported to the SRC, majority sample is then placed on the J3 backplane bus. The FRAME2-3 bit is used to signal the appearance of the new command and can be used to start the data sampling. The FRAME2-3 bit is purposefully 2 frames long to ensure that if frame 2 is corrupted that frame 3 can still be used to send the FRAME2-3 marker; in this case, frames 3-5 will be sampled.

4.2.3 Command Summary and Protocol

The commands that the SRC sends to the FIB controller are associated with the modes of operation of the SVX3 chips and the data acquisition. These commands are sent in parallel with the SYNC line. A zero-to-one transition in the SYNC line validates the commands. Table 6 lists the code assignment of these commands.

XQT	CODE	MEANING
1	0x4	Pre-amplifier Start Reset Interrupt
1	0x2	Pre-amplifier Stop Reset Interrupt
0	0x16	G-Links stop Fill Frames
0	0x15	G-Links send Fill Frames
0	0x14	Reset PC Controller
0	0x13	Reset FIB
0	0x12	Reset SVX3 chip
0	0x11	Latch status (reset by VME access)
0	0x9	Calibration inject
0	0x6	Digitize
0	0x5	Readout
0	0x4	Pre-amplifier Start Reset
0	0x2	Pre-amplifier Stop Reset
0	0x1	Abort Readout or digitization
0	0x0	No operation

Table 6. SRC Command Assignments

Commands 0x12, 0x13, and 0x14 set the SVX3 chips, the FIB, and the PC into known states. Command 0x12 does not re-configure the SVX3 chips but sets the SVX3 chips to a known state (specifically, to the data sampling mode, but no SVX-II acquisition clock is delivered). Command 0x15 forces the G-Link transmitter to send fill frames to allow the G-Link receiver on the VRB to regain lock. Command 0x11 is used for diagnostic purposes, and latches the last command processed by the FIB. The commands that are not listed have the same behavior as No Operation but may be used in future upgrades and should be considered RESERVED.

The SRC forwards these commands and signals every 132 ns, and all input data is validated by the SYNC pulse and changes after the duration of the SYNC pulses

.

The FIB-MS cannot execute more than one command at a time as the FIB-MS does not have pipeline capability. After the SRC sends a command, the SRC has to wait until the end of the execution of this command, which can take several multiples of 132 ns. Please note that there is an on board FIFO to queue requests.
However, this rule has exceptions that we will now describe:

a) The Latch Status command can be sent when another command is in execution.

b) The Pre-amplifier Start Reset and Pre-amplifier Stop Reset can processed as preemptive interrupts during the latter portion of Digitization and at any time during readout. The exception to this is the beginning of readout, when charge transfer is taking place. The SRC is responsible for not allowing this exception from happening.

The number of SYNC pulses to execute different SVX3 operations, including the one used to validate the command, are listed in Table 7. The operation Readout is not listed because the FIB-FC supplies the EOR control byte to inform when the operation is completed.

OPERATION	**NUMBER OF SYNC PULSES**
Reset TPC Controller	*pending (TFIB: 10)*
Reset TFIB	*pending (TFIB: 300 ms[1])*
Reset SVX-II chip	*pending (TFIB: 6)*
Latch status	*pending (TFIB: 1)*
Calibration inject (minimum)	*pending (TFIB: 6)*
Calibration inject (maximum)	*pending (TFIB: 8)*
Stop data sampling clocks	*pending (TFIB: 1)*
Preamplifier start reset	*pending (TFIB : 4)*
Enter in data sampling mode	*pending (TFIB: 7)*
Preamplifier stop reset	*pending (TFIB: 2)*
Digitize	*pending (TFIB: N/A)*
Return Capacitor to Pipeline	*pending (TFIB: N/A)*

Table 7. Number of SYNC pulse to execute different SVX3 operations

[1] This parameter does not use “number of SYNC pulses”. This time is given by the reset circuitry of the TFIB, formed by a MAX707 manufactured Maxim. It maximum reset pulse is 280 ms.

4.3 Port Card Interface

The PC receives encode control information from the FIB through the Control Bus (C[4:0]), Control Data (C_Data) and Control Clock (C_CLK) lines. The FIB Micro-sequencer issues a series of data “words” on the control bus in sequence when executing a specific operations on the SVX3 chips.

The electrical communication is accomplished by several sets of differential lines which are driven from the FIB/PC transition module. For further information regarding the transition module, see reference [12].

The table on the next page shows the P3 I/O pin assignments, that are used to connect the FIB to the FIB/Port Card transition module.

FIB J3 Connector Pin-out - (rev 2.1 - 10/29/96)

Pin #	Row 'a'	Row 'b'	Row 'c'	Row 'd'	Row 'e'
1	GND	GND	BUS1	GND	GND
2	**B_MCLK**	**HDI-A-D0**	BUS2	NC	**HDI-B-D0**
3	GND	**HDI-A-D1**	BUS3	GND	**HDI-B-D1**
4	**B_AD_PIPE**	**HDI-A-D2**	BUS4	NC	**HDI-B-D2**
5	GND	**HDI-A-D3**	BUS5	GND	**HDI-B-D3**
6	**B_SYNC**	**HDI-A-D4**	BUS6	NC	**HDI-B-D4**
7	GND	**HDI-A-D5**	BUS7	GND	**HDI-B-D5**
8	NC	**HDI-A-D6**	BUS8	NC	**HDI-B-D6**
9	GND	**HDI-A-D7**	BUS9	GND	**HDI-B-D7**
10	NC	**HDI-A-OBDV**	BUS10	NC	**HDI-B-OBDV**
11	GND	**HDI-C-D0**	BUS11	GND	**HDI-D-D0**
12	NC	**HDI-C-D1**	BUS12	NC	**HDI-D-D1**
13	GND	**HDI-C-D2**	BUS13	GND	**HDI-D-D2**
14	NC	**HDI-C-D3**	BUS14	NC	**HDI-D-D3**
15	GND	**HDI-C-D4**	BUS15	GND	**HDI-D-D4**
16	NC	**HDI-C-D5**	BUS16	NC	**HDI-D-D5**
17	GND	**HDI-C-D6**	BUS17	GND	**HDI-D-D6**
18	**C0**	**HDI-C-D7**	BUS18	NC	**HDI-D-D7**
19	GND	**HDI-C-DVAL**	BUS19	GND	**HDI-D-DVAL**
20	**C1**	**HDI-E-D0**	BUS20	NC	**HDI-F-D0**
21	GND	**HDI-E-D1**	BUS21	GND	**HDI-F-D1**
22	**C2**	**HDI-E-D2**	BUS22	NC	**HDI-F-D2**
23	GND	**HDI-E-D3**	BUS23	GND	**HDI-F-D3**
24	**C3**	**HDI-E-D4**	BUS24	NC	**HDI-F-D4**
25	GND	**HDI-E-D5**	BUS25	GND	**HDI-F-D5**
26	**C4**	**HDI-E-D6**	RESERVED	NC	**HDI-F-D6**
27	GND	**HDI-E-D7**	SYNC(n)	GND	**HDI-F-D7**
28	**C5**	**HDI-E-DVAL**	/SYNC(n)	NC	**HDI-F-DVAL**
29	GND	**HDI-G-D0**	RESERVED	GND	**HDI-H-D0**
30	**C_DATA**	**HDI-G-D1**	MCLK(n)	NC	**HDI-H-D1**
31	GND	**HDI-G-D2**	/MCLK(n	GND	**HDI-H-D2**
32	**C_CLK1**	**HDI-G-D3**	RESERVED	NC	**HDI-H-D3**
33	GND	**HDI-G-D4**	GND	GND	**HDI-H-D4**
34	**C_CLK2**	**HDI-G-D5**	**BNRB-A**	NC	**HDI-H-D5**
35	GND	**HDI-G-D6**	**BNRB-B**	GND	**HDI-H-D6**
36	**PIPE_RD2**	**HDI-G-D7**	**BNRB-C**	NC	**HDI-H-D7**
37	GND	**HDI-G-DVAL**	**BNRB-D**	GND	**HDI-H-DVAL**
38	**L1A**	**HDI-I-D0**	**BNRB-E**	NC	**HDI-J-D0**
39	GND	**HDI-I-D1**	**BNRB-F**	GND	**HDI-J-D1**
40	**FE_CLK1**	**HDI-I-D2**	**BNRB-G**	NC	**HDI-J-D2**
41	GND	**HDI-I-D3**	**BNRB-H**	GND	**HDI-J-D3**
42	**FE_CLK2**	**HDI-I-D4**	**BNRB-I**	NC	**HDI-J-D4**
43	GND	**HDI-I-D5**	**BNRB-J**	GND	**HDI-J-D5**
44	**BE_CLK1**	**HDI-I-D6**	GND	NC	**HDI-J-D6**
45	GND	**HDI-I-D7**	NC	GND	**HDI-J-D7**
46	**BE_CLK2**	**HDI-I-DVAL**	NC	NC	**HDI-J-DVAL**
47	GND	GND	NC	GND	GND

4.4 G-Link Interface

The electrical interface between the G-Link and the FIB is described in G-Link/Finisar Transmitter and Receiver Boards specification [ref. 9 and 10]. The data being read out from two and one-half hybrids are combined into one G-Link transfer as is shown in the example of Table 8.

HDI # A (8 bits)	**HDI # B (8 bits)**	**HDI # E (4 bits)**
HDI MSB ID	HDI MSB ID	HDI MSB ID
HDI LSN ID& FIB LADDER ID	HDI LSN ID & FIB LADDER ID	HDI LSB ID & FIB LADDDER ID
BXING NUM	BXING NUM	BXING NUM
BE STATE (2 bits)	BE STATE (2 bits)	BE STATE (2 bits)
CHIP ID	CHIP ID	CHIP ID
STATUS	STATUS	STATUS
CH #	CH #	CH #
DATA	DATA	DATA
CH #	CH #	CH #
:	:	:
CHIP ID	CH #	CH #
STATUS	DATA	DATA
CH #	CH 127	CH 127
:	DATA	DATA
CH 127	EORn	EORn
DATA	EORn	EORn
EORn	EORf	EORf
EORn	EORf	EORf
EORf	EORf	EORf
EORf	EORf	EORf

Table 8. G-Link or Data FIFO data format

The data of two and one-half HDIs are transferred at a rate of 53 MHz per word with a word being 20 bits. The data transmitted has appended a HDI identification word, the MSB ID, LSN ID and FIB LADDER ID. These two bytes correspond to two bytes used to identify the source of the data packet. EOR is flagged independently on each HDI data packet by asserting bits 6 and 7 of the data byte (see Table 1). The Bunch Crossing Number (8 bits) and Back End State Information is also appended. The Data Output Format section (2.4.6) provides further explanation on this topic.

The EOR is inserted by the FIB-Pipeline Processor as soon as it recognizes that the last SVX3 chip of the HDI is done transmitting its data. The readout data from one HDI can finish before the other, and then we have EOR Fill frames (EORf) being asserted in that HDI (EORf = 0xFF). The HDI data packet is formed by an even number of bytes.

The outputs of the Finisar LEDs are ST connectors and protrude through the front panel of the FIB.

Note: The Data FIFOs are organized in the following fashion, as shown in the below table:

G-Link Number	G-Link Data Bit Assignments D7-D0 (8)	D15-D8 (8)	D20-D16 (4)
0	HDI A[7-0] (L0-PA)	HDI B[7-0] (L1-PA)	HDI E[3-0] (L4-PA-LSN)
1	HDI C[7-0] (L2-PA)	HDI D[7-0] (L3-PA)	HDI E[7-4] (L4-PA-MSN)
2	HDI F[7-0] (L0-PB)	HDI G[7-0] (L1-PB)	HDI J[3-0] (L4-PB-LSN)
3	HDI H[7-0] (L2-PB)	HDI I[7-0] (L3-PB)	HDI J[7-4] (L4-PB-MSN)

Table N4. G-Link Data Organization (Layer #-Port Card A or B)

4.4.1 Output Data Skew

The following sections discuss output data skew in the readout for different locations of the detector.

4.4.1.1 Reading Data from a Single FIB or FIBs in a Single Subrack

The five layers of a wedge are read via two G-link cables, two and one half layers per G-link cable as illustrated in Table N4. Because the information for the fifth HDI/Ladder is distributed across two separate G-Links (denoted HDI E and HDI J above), the data from each group of five HDIs must be synchronized at the FIB before transmission. Therefore the data from G-links 0 and I are synchronized and the data from G-Links 2 and 3 are synchronized. All four will run off the same free-running crystal clock source within a single FIB but it is important to note that the start of transmission of data from G-links 0 and I most always will be different from that of G-links 2 and 3. This is due to the fact that data arrives from different layers at different times depending upon 'hit' channel patterns.

On a single synchronized pair of G-Links from event to event, there is no constant delay and/or constant clock phasing from, for example, the edge of a Level I Accept to the transmission of the first word of data from a FIB. Similarly, there is no constant delay and/or constant clock phasing between the transmission of the first word of data from one synchronized pair of G-links on a single FIB and the other synchronized pair of G-links. Therefore, FIFO buffering at the receiver of data from a single (or multiple FIBs for that matter) is required. How many levels of buffering are required is unknown and dependent upon several factors (sources of data skew) such as those listed in Table N5.

4.4.1.2 Reading Data from Multiple FIBs in Multiple Subracks

As in the case of data from a single subrack, free-running crystals, one per FIB, are used to transmit data multiple FIB subracks. One major difference which affects the timing and clock phasing of the receipt of data from more than one FIB subrack is the fact that separate timing links from the SRC go to each FIB subrack. Thus there are additional factors which affect delays and phasing between data transmitted from multiple subracks as listed in Table N5.

4.4.1.3 FIFO Buffering when receiving FIB data

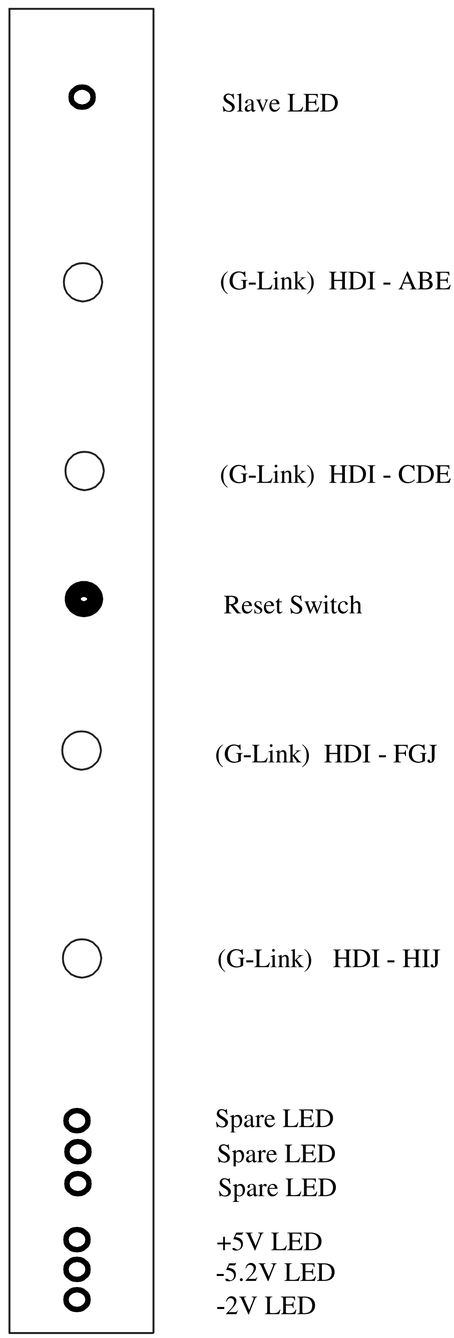
Taking into consideration variations in propagation delays within electronic components and (optical and copper) cables and those variations with temperature, humidity, etc. and given that cable lengths are matched as closely as possible where necessary, FEFO buffering to re-synchronize received data is still mandatory whether that data is from a single synchronized pair of G-links from a single FIB or many G-Links from several FIBs from more than one subrack. Given matched cable lengths where necessary, the level of FIFO buffering needed should be small and thus, for example, not significantly increase Level I processing time in the SVT.

Data connection	Sources of data skew
1 wedge from 1 FIB VRB or SVT	• Variation in electrical to optical propagation delay through the Finisar transmitter. • Variation in optical cable lengths. • Variation in optical to electrical propagation delay through the Finisar receiver. • Variation in the G-Link receiver PLL, including quantum phase error. • Variation in the input to output propagation delay through the G-Link receiver. • Variation in the propagation delay through the GRT receiver data buffers.
2 wedges from 1 FIB VRB or SVT	• All of the above, plus • Variation in control cable length down to the port card • Variation in propagation through port card clock buffering and DDR chip • Variation in HDI lengths • Variation in the input to output propagation delay through the SVX3 BE chip. • Variation in electrical to optical propagation delay through the DOIM transmitters • Variation in optical cable lengths. • Variation in optical to electrical propagation delay through the DOIM receivers. • Variation in the propagation delay through the FIB-TM receiver data buffers.
multiple wedges from 1 FIB sub-rack VRB or SVT	• All of the above, plus • Variation in clock crystal phase
multiple wedges from 2-4 FIB sub-racks SVT	• All of the above, plus • Variation in electrical to optical propagation delay through the SRC Finisar transmitter. • Variation in optical cable lengths. • Variation in optical to electrical propagation delay through the FIB Fan-out Finisar receiver. • Variation in the G-Link receiver PLL, including quantum phase error. • Variation in the input to output propagation delay through the G-Link receiver. • Variation in the propagation delay through the GRT receiver data buffers.

Table N5. Factors Affecting Data Skew at the Receiver of FIB data

4.5 Front Panel

Figure 7 shows the front panel of the FIB.



5. RESET, POWER UP RESET & INITIALIZATIONS

This section is pending

6. ERROR DIAGNOSIS

There is minimal diagnosis of PC and FIB errors which has been defined at this point; though a few are listed below:

SVX3 Configuration and Read-back

Failure of Read-back Chain (EOR channel missing)

Checking FIB to Port Card Connections

Analysis of Data Stream (not yet implemented)

6.1 SVX3 Configuration and Read-back

Reading back the SVX3 configuration data will be most helpful to:

A) Verify that there is basic communication down to the Port Card and the SVX3 chips, and

B) Diagnose if any configuration data is being lost due to radiation damage.

6.2 Failure of Read-Back Chain (EOR channel missing)

Failure of the read back chain, such that the EOR channel is not seen is dealt with in a very straight forward manner:

STEP 1 - EORt (EOR truncate) is transmitted after too many clocks are issued down to the PC for data.

STEP 2 - After the EORt frames are detected a system error is issued (i.e. SVX system goes into HALT)

STEP 3 - During RESET, Last Chip and Channel Registers of offending read-back chain are read to determine the last good channel that was read.

STEP 4 - Last Channel Register is loaded with new value and/or SVX3 chips are re-programmed to eliminate bad chip(s) from string.

6.3 Checking FIB to Port Card Connections

The primary check of the FIB to Port Card Connection is to monitor the output of the 4 control lines to the port card. These can be individually controlled by the FIB, and their status read back through the VMEbus. This provides for a good check of the I/O between the FIB and the Port Card, however, does not check 100% of the connection (example. Bit 4-7 cannot be driven by the Port Card/DDR chip.

6.4 Analysis of the Data Stream (not yet implemented)

During processing the Odd-byte Data valid is used to verify that the data frames are arriving at the proper phase. If there is an error in the data phase the FIB could be programmed to mark the bad data stream with an appropriate EOR marker.

6.5 Additional Error Diagnosis Information

As additional error detection information is available this section will be expanded.

7. ELECTRICAL & MECHANICAL SPECIFICATIONS

7.1 Headers

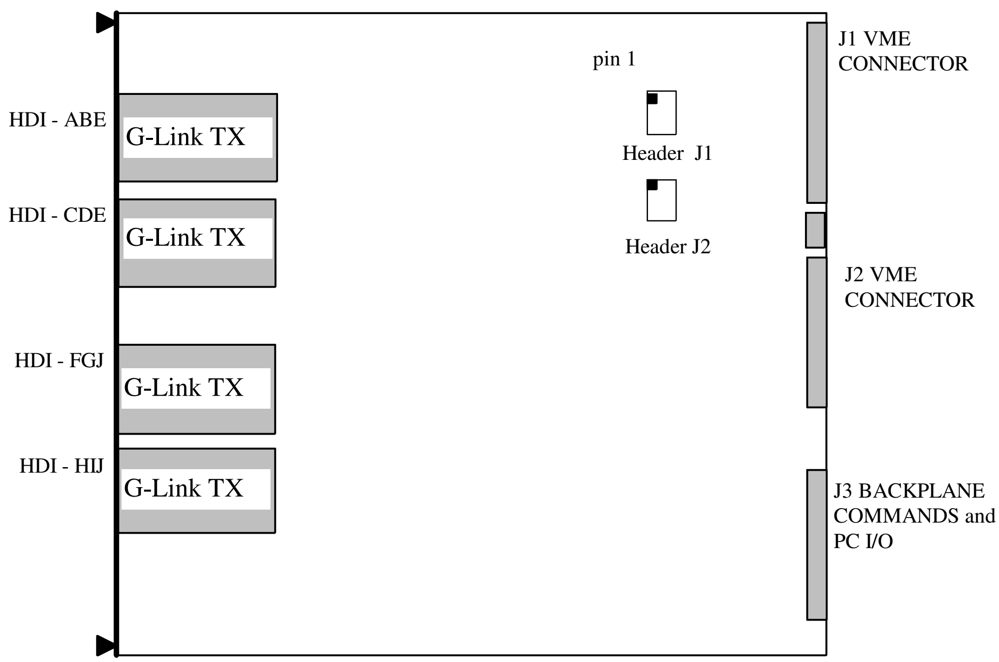
(Verify placement and bit assignment)

Figure 9 shows the location of the Board Headers.

There two headers on the board which are for programming the ALTERA 7128SQC100 components.

HEADER NUMBER	USEAGE
J2	*ISP - Altera (1) VMEbus interface only*
J1	*ISP - Altera (16) All other 7128 parts*

Table 9. Board Headers

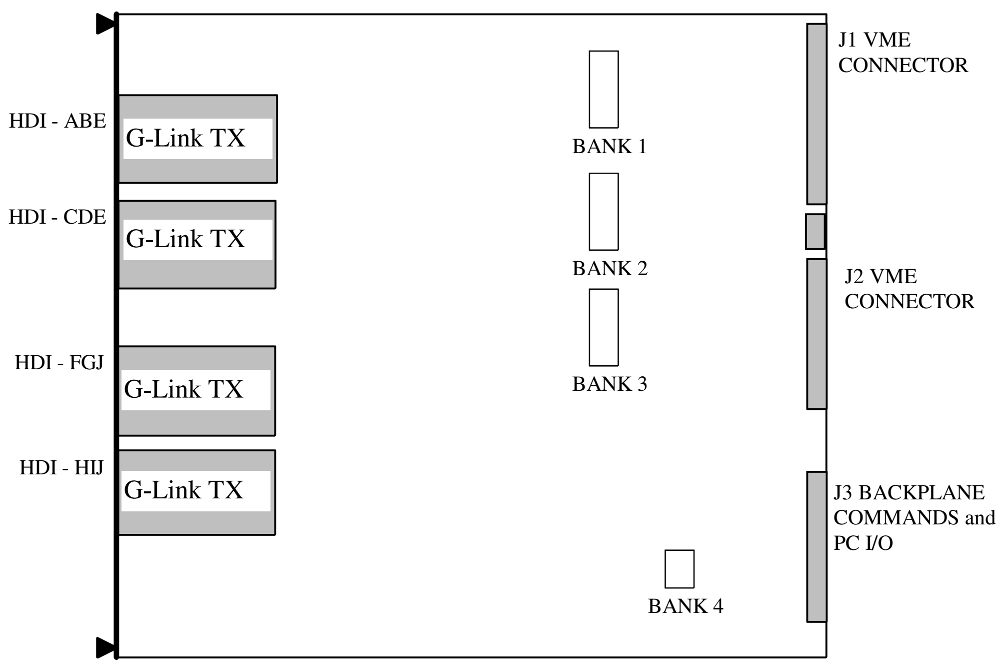


7.2 Jumper Banks

Figure 10 shows the location of the four jumper banks on the board. Three are for static status read-back primarily revision and serial number. The other is for FIB status back to the FFO (this is currently unused).

JUMPER BANK NUMBER	USEAGE
1	*Fixed Status*
2	*Fixed Status*
3	*Fixed Status*
4	*Open Collector Status Outputs (Reserved)*

Table 10. Board Headers



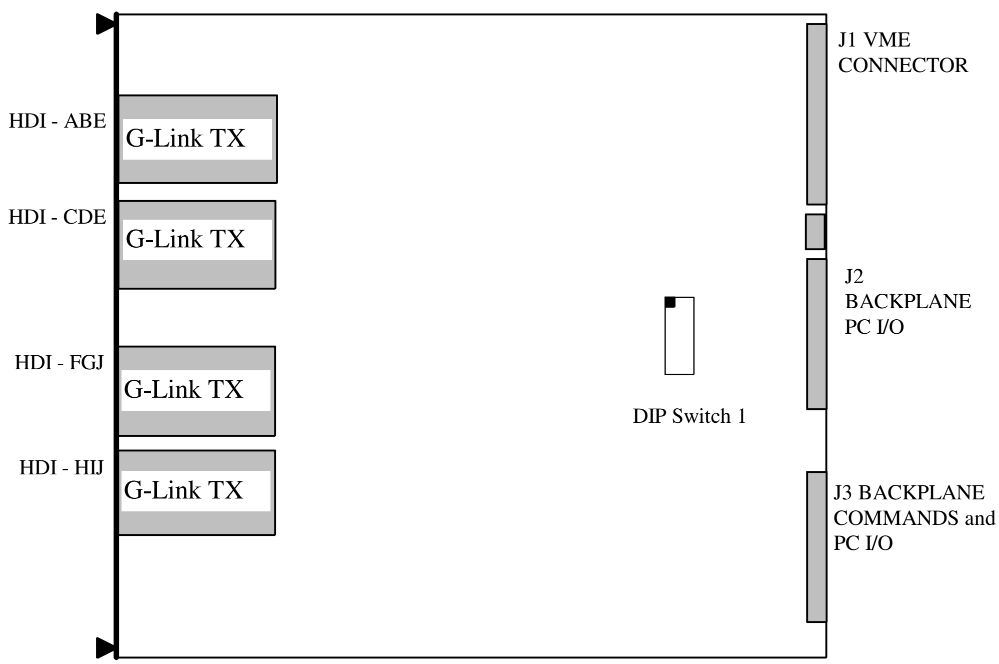
7.3 Dip Switches

Figure 8 shows the location of the address DIP switch.

DIP switch S1 is used to set the base address to the FIB if the Geographical Pins are not available (see Section 4.1.1). The table below shows the bit assignments of DIP switch S1. Bits are set to zero by having the switch in the on position. Bits are set to one by having the switch in the off position.

DIP SWITCH OPEN = 1, CLOSED = 0	**ADDRESS LINE**
8	*A27*
7	*A28*
6	*A29*
5	*A30*
4	*A31*
3	*NC*
2	*NC*
1	*NC*

Table 11. DIP Switch S2 Assignment



7.4 Packaging & Physical Size

The TFIB is a 9U x 400 card.

7.5 PC Board Construction

Signal Std. Trace Width:	8 mils
Signal Copper Thickness:	1 oz (0.015)
Plane Copper Thickness:	2 oz
Via size:	x od/ y id
Plane stacking:	see table below
Board Thickness:	0.093 +/-
Board Dimensions:	h/w +/-

Layer	Approximate Impedance	Spacing to next Layer
Signal 1	110	5mils
Signal 2	105	5 mils
Signal 3	100	20 mils
GND 1	N/A	5 mils
+5V	N/A	5 mils
-5.2V/-2V	N/A	5 mils
GND 2	N/A	20 mils
Signal 4	100	5 mils
Signal 5	105	5 mils
Signal 6	110	5 mils

7.6 Power Requirements

The TFIB requires +5.0V and -5.2V power supplies. -2V is supplied on board by a -2V regulator.

7.7 Cooling Requirements

Cooling for the TFIB will be supplied by fans mounted above or below the VMEbus subrack.

Additional cooling will be provided by required air-water heat exchangers.

8. SAFETY FEATURES & QUALITY ASSURANCE PROCEDURES

8.1 Module Fusing & Transient Suppression

The TFIB card is protected with fuses and transient absorbers.

8.2 Other Safety & Quality Assurance Subsections

9. APPENDICES

9.1 Schematics

See office file.

9.2 PAL, FPGA Equations

See office file.

9.3 Timing Diagrams

See office file.

9.4 Parts List

Inventory number	Manufacturer, Description	Part Number	Quantity
1	Cypress, 8K x 9 Synchronous FIFO	CY7C4251-15JC	12
2	IDT, 4Kx9 Parallel - Serial FIFO	IDT72104L50J	2
3	Motorola, 32Kx8 SRAM, 32 pin SOJ	MCM6206DJ12	10
4	Altera, 7128S EPLDs, -7 speed, 100 pin, QFP	EPM7128SQC100-7	17
5	Dallas, DS1708 Power Monitor Chip, 8-pin SOIC	DS1708ESA	1
8	SIMTEK, 8Kx8 Non-volatile SRAM	STK11C68-S25	4
12	Motorola, Transient absorber	Newark # ICTE-5	2
13	Dale, 3.3K 10 pin SIPs, Type number: CSC10A-01-332G	Newark # 81F4857	13
19	IDT, Octal D transparent latch, QSOP, IDT74FCT573ATQ	IDT74FCT573ATQ	7
20	IDT, Octa latched transceiver, QSOP, IDT74FCT543ATQ	IDT74FCT543ATQ	5
21	IDT, Octal buffer w/ tristate, QSOP, IDT74FCT541ATQ	IDT74FCT541ATQ	75
22	IDT, Octal buffer inverting w/ tristate, QSOP, IDT74FCT540ATQ	IDT74FCT540ATQ	1
23	IDT, Octal bus transceiver, QSOP, IDT74FCT645ATQ	IDT74FCT645ATQ	24
24	TI, Quad two input NAND buffers, open collector, SN74F38D	SN74F38D	2
25	Panasonic, 0.1 uF 1206 surface mount capacitors, X7R type	Digikey # PCC104BCT-ND	363
26	Panasonic, 1K Ohm 1206 surface mount resistor	Digikey # P1.00KFCT-ND	41
27	Panasonic, 10K Ohm 1206 surface mount resistor	Digikey # P10.0KFCT-ND	10
29	Sprague, 47 uF 2213 surface mount cap, #293D476X9010D2T	Newark # 93F2698	4
30	Finisar, Transmitter Boards **(WITHOUT OFC CONTROL)**	FTC-1101-1	4
31	AMP, Board-to-Board Connector, System 50	104078-6	4
32	AMP, Eurocard type C, 96 pin, right angle PCB mount connector	650473-5	2
33	AMP, discrete socket, flat bottom	2-331272-2	4
34	AMP, Z-PACK 2mm HM PCB Mount, Type B, 125 positions	100145-1	1
35	AMP, Z-PACK 2mm HM PCB Mount, Type B, 110 positions	95-10586-1	1
36	ERN, 2mm HM PCB Mount, Type B, 90 positions	914794	1
37	Front Panel Ejector Kits	Front Panel Kit	1
38	Transition Module Ejector Kits	Transition Panel Kit	1
43	Alcoswitch, 8 position DIP switch	ADE08	1
44	Rittal, Double Wide Front Panel Ejector Kits	Double-wide Front Panel Kit	1
45	Motorola, ECL to TTL clock distribution chips (37 package)	MC10H643FN	5
46	Motorola, registered hex TTL/ECL translator	MC100H604FN	16
48	IDT, 10 bit buffer with dual output enable	IDT74FCT827ATQ	20
49	Cypress, 8Kx9 high speed Synchronous FIFOs (100MHz)	CYC4251-10J	10
50	Synergy, low-power hex TTL to ECL converter	100S324	4
51	Synergy, low-power hex ECL to TTL converter	100S325	1
52	Dale, 510 Ohm 10 pin SIPs, Type number: CSC10A-01-511G	Newark # 81F4857	12
53	Panasonic, 50 Ohm 1206 surface mount resistor	Digikey # P50FCT-ND	7
54	Panasonic, 100 Ohm 1206 surface mount resistor	Digikey # P100FCT-ND	4
55	Motorola, variable negative voltage regulator	LM337	1
57	Motorola, 9-Bit TTL/ECL Translator	MC10H600FN	1
58	Motorola, 4-Bit D Flip-Flop	MC10E131FN	2
59	Motorola, Quint 2-Input AND/NAND Gate	MC10E104FN	1
60	Motorola, Quad 4-Input OR/NOR Gate	MC10E101FN	1
61	ELMEC, SIP fixed delay line, 10ns delay line, FDD type	FDD10010	1
62	Signetics, Surface mount, general purpose CMOS timer	ICM7555CD	1
63	Signetics, Surface mount, one of 10 decoder, with 3-state outputs	N74F537D	3
64	Signetics, Surface mount, 8 bit magnitude comparitor	74HCT688D	2
65	Newark , Dale, 50MHz crystal oscilliator, XO-43B 50M	XO-43B 50M	1
67	Motorola, Quint Differential Line Reciever	MC10E116FN	1
68	IDI, RED LED Circuit Board Indicator, Digikey L20471-ND	5309K1	1
69	IDI, GREEN LED Circuit Board Indicator, Digikey L20475-ND	5309K5	3
70	IDI, YELLOW LED Circuit Board Indicator, Digikey L20477-ND	5309K7	3
71	Tusonix, EMI filter	4100-06X	2
72	Rittal, PCB holder	3685619	1
73	Rittal, M2.5x10 screw	3606610	1
74	Rittal, M2.5x8 screw	3654320	1
75	Rittal, Keys Grey	3684325	12
76	Rittal, Keys Red	3684326	12
77	Rittal, Solder Side Covers, 9U x 400mm	3685806	1
78	Tracor Littelfuse, 7 Amp,	Digikey # F834-ND	1
79	Tracor Littelfuse, 10 Amp	Digikey # F835-ND	1
80	Bourns, 100 Ohm sips	Digikey # 4310R-1-101-ND	4
81	Dallas, Silicon Delay Line with Logic	DS1012Z-V50	10
82	SAMTEC, 10 pin header	TSW-105-07-G-D	2
83	Panasonic, 1M Ohm 1206 surface mount resistor	Digikey # P1.00MFCT-ND	4

9.5 Configuration of the PC and SVX3 chips

9.5.1 Configuring the PC

The VMEbus motherboard also has to program two (2) digital to analog converters (DAC) and HDI control settings on the PC. The DACs are use for calibration. Downloading this information is done by first activating a specific port card (The specific implementation of this has not been determined, however it will consist of enabling the control clock and/or control signals to one or both of the port cards). Second a series of command are issued to the FIB-CT for the FIB Micro-sequencer to download the configuration data to the PC. This includes enabling a specific HDI and also and DAC configuration settings. This command is described in more detail in the section below (Configuration Details).

9.5.2 Configuration of the SVX3 chips

For configuration of the SVX3 chips a similar procedure is followed. First a Port Card is selected (as described above). Second, an HDI is selected by issue a PC initialization as outlined above. Third, data is downloaded by issuing a Download HDI command. Unlike the other commands in the FIB Micro-sequencer, the data for this command is stored in an auxiliary shift register (Data Shift Register). There is also a companion shift register (Pointer Shift Register) which has the end of data stream marker. These are described in more detail in the below section (Configuration details). Since the SVX chips require a large amount of data, this last step needs to be repeated a number of times until the entire chain is downloaded.

9.5.3 Configuration details

The FIB allows the VMEbus CPU board to download and upload the configuration of the SVX3 chips. And connected to two Port Cards, and the DACs and HDI enable registers on those Port Cards. These operations are performed through the following VME registers:

- Command FIFO Register.
- FIB Micro-sequencer.
- Data Shift Register and Pointer Shift Register

The VMEbus CPU board sees configuration of the SVX3 chips and PC as multiple operations. Here is an outline of the configuration steps for both procedures:

9.5.3.1 Configuring the PC

1) SELECT PC. A command is issued to enable encode control to only one Port Card. This command essentially enables the control clock to one of the port cards.

2) ENABLE HDI COMMAND. A command is issued to select a specific HDI. There is a specific micro-sequence for each HDI enable. Note: there is also an enable all micro-sequence which is not used here.

4) CONFIGURE DAC. After an HDI is enabled, the DAC associated with that HDI can be configured. Each data bit is set or reset with a different micro-sequence.

9.5.3.2 Configuring the SVX3 chips

Downloading the SVX3 chips operates in approximately the same fashion. Here are the steps:

1) SELECT PC. A command is issued to enable encode control to only one Port Card.

2) ENABLE HDI COMMAND. A command is issued to select a specific HDI. There is a specific micro-sequence for each HDI enable. Note: there is also an enable all micro-sequence which is not used here.

3) LOAD DOWNLOAD FIFO WITH PROPER CONFIGURATION STRING. All of the download bits are loaded into the FIFO in parallel.

4) LOAD DOWNLOAD COUNTER WITH NUMBER OF BITS IN CONFIGURATION STRING.

5) ISSUE DOWNLOAD SVX3 COMMAND. After the FIB-MS receives the command for downloading the HDI, the FIB-MS transfers all of the data in the Data Shift Register until the End of Data Marker is reached. After this step is completed, the data from the end of the SVX3 configuration chain will have been pushed into the Data Shift Register. This step includes transferring the configuration data into the secondary registers in the SVX3 chips.

6) Steps 1) through 5) are repeated until all data is transferred into the SVX3 chips.

7) SEND RESET PORT CARDS COMMAND. This command resets all of the pipelines and puts the SVX3 chips into data acquisition mode.

REFERENCES

[1] "VME64 Extensions for Physics and Other Applications", ANSI/VITA 23-1998
[2] "FIB Subrack J3 Backplane", John Chramowicz, Stefano Rapisarda, Ken Treptow, ESE-SVX-990704
[3] "Device Specifications for the SVX3 Readout Chip Set", R. Ströhmer and W. Knopf, ESE-SVX-950703.
[4] "A Beginners Guide to the SVX-II", R. Yarema, FERMILAB-TM-1892.
[5] "FIB Fanout Module (FFO)", John Anderson, Kerry Woodbury, John Chramowicz, Stefano Rapisarda, ESE-SVX-951128
[6] (add reference to PC document - pending)
[7] "Silicon Readout Controller", C. Gay and J. Oliver, 1995.
[8] "Low Cost Gigabit Rate Transmitter and Receiver", Hewlett Packard, November 1993.
[9] "VME Readout Buffer", Mark Bowden, Ted Zmuda, Thinh Pham, ESE-SVX-950719.
[10] (add reference to SVT)
[11] (add reference to HDI)
[12] K. Woodbury, "FIB/PC Transition Module Specification", *pending*.
[13] VMEbus Specifications, ANSI/IEEE STD1014.
[14] J. Andresen, "G-Link/Finisar Transmitter and Receiver Boards", October 1995.