

5. I/O Modules

5.1 IM2E1

The IM2E1 provides two CEPT/E1 interfaces for the V6M6 and V6M6HS PCI module carrier boards. Electrical interfaces for HDB3 networks are provided via RJ45 jacks, one for each E1 line. E1 frame encoding and decoding is performed on each E1 line by a Dallas Semiconductor DS2153Q CEPT Transceiver. Elastic frame buffers (DS2175) interface the outgoing E1 streams to the TDM subsystem. Incoming E1 data is passed through the internal elastic store of the DS2153Q framer chip.

Information in this manual is divided into the following sections:

- 5.2.1 A Quick Tour of the IM2E1
- 5.2.2 IM2E1 Hardware Description
 - 5.2.2.1 Module and E1 Framer Registers
 - 5.2.2.2 TDM Subsystem Interface
 - 5.2.2.3 Electrical Line Interface
 - 5.2.2.4 LED Indicator
 - 5.2.2.5 Specifications
- 5.2.3 Software Support
 - 5.2.3.1 Utility Programs
 - 5.2.3.2 Host Application Library
 - 5.2.3.3 MIPS Kernel Library
 - 5.2.3.4 Diagnostic programs

5.1.1 IM2E1 Quick Tour

Figure 4-1 is a block diagram of the IM2E1 showing the major components and signal paths. The primary components are the two Dallas Semiconductor DS2153Q E1/CEPT Transceivers. Please refer to the DS2153Q data sheet (provided in Appendix A: "Component Data Sheets") for details on their operation.

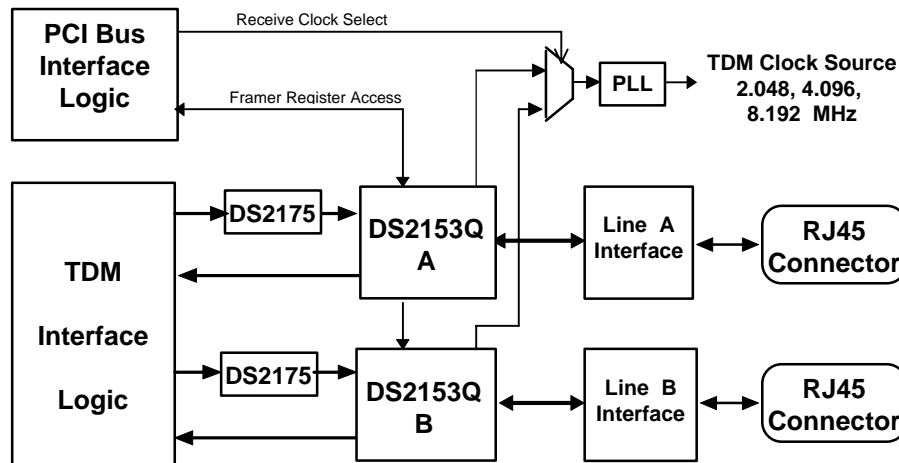


Figure 5-1: IM2E1 Module Block Diagram

The two E1 lines, designated A and B, connect to the smPCI board through RJ45 connectors. The E1 signals pass through the line interface components to and from the DS2153Q framers. The input impedance for each line is selectable to be 75Ω or 120Ω (see Section 5.1.2.3 Electrical Line Interface). The output impedance is controlled by the line build-out select in the DS2153Q's LICR registers (see table 12-2 in the Dallas DS2153Q data sheet). The output pulse transformer ratios are 1:1.15.

The DS2153Q E1 components provide clock and data recovery for the incoming E1 lines and E1 framing and pulse generation for the outgoing E1 lines. Access to the DS2153Q control, status and data registers is provided through the PCI interface.

Incoming and outgoing E1 data is accessed by other PCI modules via the TDM subsystem of the V6M6 board. Incoming multiframe and loss of signal status is also available via the TDM subsystem. The TDM interface is designed to be operated with 8-bit TDM time slots and TDM frames of 32 slots / 2.048 MBPS, 64 slots / 4.096 MPBS or 128 slots (8.192 MPBS). A phase-locked loop may be used to synchronize the TDM subsystem clock with one of the incoming receive clocks.

FIFO storage between incoming E1 data and the TDM subsystem is provided by the DS2153Q framers. The two Dallas Semiconductor 2175 elastic store buffers provide a FIFO interface for outgoing E1 data.

IM2E1 modules may be installed on V6M6 boards at module locations A and C for front panel access to the RJ45 connectors. A future option will allow IM2E1 modules to be installed at module locations B, D or F with its I/O connections made via the VME P2 connector.

5.1.2 IM2E1 Hardware Description

The IM2E1 and IM2T1 use the same FPGA configuration data stored in the Flash ROM on the V6M6. The flash object name is **intelco**. The PCI interface logic uses the same generic, configurable design as other IM modules for the V6M6. Configuration of this logic is performed by the host system using the V6M6 board initialization program, **pciinit**.

5.1.2.1 Module Control and E1 Framer Registers

The registers for control of the module and the E1 framers are accessed in PCI bus configuration space and i/o space. The table below shows the base addresses for the module locations that may be used to carry IM2E1 modules. The configuration space base addresses are determined by physical hardware connections. I/O space base addresses are determined by convention and are hard coded in various software modules. The values are loaded into the IM2E1 registers by the **pciinit** program.

PCI Base Addresses for IM2E1 Modules		
PCI Module	Config Space Base Address	I/O Space Base Address
A	0x01000000	0x30000000
B	0x00800000	0x31000000
C	0x00400000	0x32000000
D	0x00200000	0x33000000
F	0x00080000	0x35000000

The following two tables list the IM2E1 registers. The address or address range shown are the offsets from the base address of the module.

IM2E1 Configuration Space Registers			
Register Name	Address	Acc	Function / Description
PCI Stat/Cmd	0x04	rw	See PCI Spec for details
Mod Base	0x10	wo	PCI base address of module
Dev Cmd	0x40	rw	Generic I/O module controls and settings
Interrupt Stat/Mask	0x44	rw	I/O Interrupt stat and mask
Interrupt Ctl	0x48	rw	Controls interrupt polarity

IM2E1 I/O Space Registers			
Register Name	Address Range	Acc	Function / Description
E1 Framer 0	0x000 - 0x3FC	rw	Line A DS2153Q Registers (see Dallas Semi. data sheet)
E1 Framer 1	0x400 - 0x7FC	rw	Line B DS2153Q Registers (see Dallas Semi. data sheet)
TDM ID	0x800	rw	Stores module location ID
Clock Ctl	0x804	rw	Controls clock sources for TDM Subsystem and E1 Transmit
RSYNC Ctl	0x808	rw	Controls TDM SYNC drive to RSYNC to DS2153Q Framer
Slip Ctl	0x80C	rw	Controls slipped frame action and clears Fast RCLK Status
Clock Ctl2	0x810	rw	Additional TDM clock control
TDM Rcv Mode	0x817	rw	TDM receive mode

PCI Stat / Cmd Register (CFG: 0x04)

This register is used for configuring and obtaining status of the module's interface to the PCI bus. The control bits are all either hardwired or programmed by the **pciinit** program. The bits set by **pciinit** are:

Bit 0 = 1: I/O Space Enabled
Bit 6 = 1: PCI Parity Error Response Enabled

Two status bits, listed below, are available for PCI error checking. These are cleared by writing a 1.

Bit 30 - Module signaled a System Error
Bit 31 - Module detected a PCI Parity Error

Mod Base Register (CFG 0x10)

This register stores the module's I/O space base address on the PCI bus. It is programmed by the **pciinit** program and a copy of its contents is stored in the host's device driver and made available to host applications.

Dev Cmd Register (CFG: 0x40)

This register is used to control low-level module functions.

Bits 31:28 - Timing for the module's internal bus
Bit 23 - Module Reset (low active)

The timing bits are programmed by the **pciinit** program to meet timing requirements of the E1 Framer devices on the module. Module Reset is only used to center the DS2175 Elastic Storage FIFOs.

Interrupt Mask / Stat Register (CFG: 0x44)

This register controls the interrupt enable and reads interrupt status for the module. The IM2E1 supports the 2 interrupt sources from both of the DS2153Q Framers and a special interrupt indicating that one of the E1 line's RCLK is running at a faster rate than TDMCLK.

The interrupt mask bits allow any of the interrupt sources to be the source of PCI interrupt A or B. For each bit, a 1 passes the interrupt and a 0 masks it.

```
Bit 0 - Framer 0 Int 1 to PCI interrupt A
Bit 1 - Framer 0 Int 2 to PCI interrupt A
Bit 2 - Framer 1 Int 1 to PCI interrupt A
Bit 3 - Framer 1 Int 2 to PCI interrupt A
Bit 4 - Framer 0 Int 1 to PCI interrupt B
Bit 5 - Framer 0 Int 2 to PCI interrupt B
Bit 6 - Framer 1 Int 1 to PCI interrupt B
Bit 7 - Framer 1 Int 2 to PCI interrupt B
Bit 8 - RCLK > TDMCLK to PCI interrupt A
Bit 9 - RCLK > TDMCLK to PCI interrupt B
```

The interrupt status bits are valid regardless of the interrupt mask setting. A 1 indicates that the interrupt source is active.

```
Bit 24 - Framer 0 Int 1
Bit 25 - Framer 0 Int 2
Bit 26 - Framer 1 Int 1
Bit 27 - Framer 1 Int 2
Bit 28 - RCLK > TDMCLK
```

Interrupt Ctl Register (CFG: 0x48)

This register is written by the **pciinit** program to match the polarity of the E1 Framer interrupts to the polarity expected on the PCI interrupt lines.

E1 Framer Registers (IO: 0x000 – 0x3FC and 0x400 – 0x7FC)

The various registers of the DS2153Q Framers are accessed via PCI I/O space. Details of the registers' functions and contents are available in the Dallas Semiconductor DS2153Q data sheet.

Framer 0 is accessed starting at offset 0x000 from the module's I/O base. Framer 1 is accessed starting at offset 0x400 from the module's base. Each of the 8-bit registers are accessed on 32-bit boundaries with the register data in the LSB of the word. They may be accessed as 32-bit words (with only the lower 8-bits containing useful data) or as 8-bit words. In either case the DS2153Q register offset, as specified in the data sheet must be shifted left by two bits then added to the module base and framer base addresses. For byte access, the address must be further offset by 3.

32-bit access:

$$\text{PCIaddr} = \text{Mod_IObase} + \text{Framer_base} + (\text{Reg_offset} \ll 2)$$

8-bit access:

$$\text{PCIaddr} = \text{Mod_IObase} + \text{Framer_base} + (\text{Reg_offset} \ll 2) + 3$$

32-bit example: LICR register of Framer 1 on Module C

$$\text{PCIaddr} = 0x32000000 + 0x400 + (0x18 \ll 2) = 0x32000460$$

Certain control register bits must be set to the following values in order for the IM2E1 to operate properly and maintain synchronization with the TDM subsystem. These are shown in the table below. Other framer controls and parameters are application dependent.

<u>Register</u>	<u>Offset</u>	<u>Bit</u>	<u>Val</u>	<u>Comment</u>
RCR1	0x10	5	0	RSYNC pin is an output
RCR1	0x10	6	1	RSYNC pin in multiframe mode
RCR2	0x11	1	1	Rcv Elastic Store Enabled
RCR2	0x11	2	1	SCLK is 2.048 MHz
TCR1	0x12	0	0	TSYNC pin is an input
TCR2	0x13	0	0	RLOS/LOTC pin indicates RLOS
CCR3	0x1B	7	0	Xmt Elastic Store Disabled
TAF	0x20	6:0	0x1b	Frame alignment bits
TNAF	0x21	6	1	Frame alignment bit

TDM ID Register (IO: 0x800)

This register stores the module ID (location) of the module and is used for identifying the module for TDM data transfers. Bits 2:0 contain the ID value as programmed by the **pciinit** program.

Clock Ctl Register (IO: 0x804)

This register is used to control the relationship between the TDM subsystem clock and the E1 receive and transmit clocks.

The clock divider bits determine the relationship between the E1 clock rate and the TDM clock rate.

```
Bits 1:0 - Clock Divider
00 : TDM clock = E1 Clock * 1 (2.048 MHz)
01 : TDM clock = E1 Clock * 2 (4.096 MHz)
10 : TDM clock = E1 Clock * 4 (8.192 MHz)
```

The TDM clock may be derived from either of the two Framer's RCLK or not from the module at all. The IM2E1 (configured with version 13 or higher of the **intelco** FPGA configuration) includes logic to detect and select the faster of the two received clocks. This is useful in applications where the incoming E1 lines may have different time bases. The value of bits 1:0 multiply the selected RCLK to derive TDM clock.

```
Bits 3:2 - TDM Clock select bits
00 : TDM clock not sourced by the module
01 : TDM clock derived from RCLK A
10 : TDM clock derived from RCLK B
11 : TDM clock derived from the faster of RCLK A and RCLK B
    (see Clock Ctl2 Register)
```

E1 transmit clock sources may be derived from either Framer's RCLK or from the TDM subsystem clock. When TDM-derived TCLK is selected, the value of bits 1:0 divide TDM clock to derive TCLK.

```
Bits 5:4 - TCLK B Select
00 : TCLK B = RCLK B
01 : TCLK B = RCLK A
10 : TCLK B derived from TDM Clock
```

```
Bits 7:6 - TCLK A Select
00 : TCLK A = RCLK A
01 : TCLK A = RCLK B
10 : TCLK A derived from TDM Clock
```

Clock Ctl2 Register (IO: 0x810)

This register is included in revision 17 and higher of the IM2E1 FPGA configuration. It is used for additional control of the relationship between the TDM subsystem clock and the E1 receive clocks.

In situations where the relative clock rates of the two incoming E1 signals are very close and wavering, the faster-clock detection logic may cause occasional loss of data (slips occur in the wrong direction). To avoid this loss of data the logic in revision 17 or higher forces the resulting TDM clock to be approximately 3.8 PPM faster than the faster RCLK. This is the default mode and will result in occasional repeated frames from both framers. In case this mode is not desirable it may be disabled using the Clock Ctl2 register.

```
Bit 1 - TDM Clock Rate Increase Control
      0 : TDM clock = Faster RCLK w/ no adjustment
      1 : TDM clock = Faster RCLK + 3.8 PPM (default)
```

RSYNC Ctl Register (IO: 0x808)

This register controls whether or not the TDM SYNC signal is driven to the E1 framer's RSYN pin. It is used by the **im2e1_init_rsync** function to initialize synchronization between the E1 framer and the TDM subsystem.

SLIP Ctl Register (IO: 0x80C)

This register enables or disables the IM2E1 logic from invalidating TDM bus data for slipped frames. The **im2e1_slipinval** library function accesses this register to control this mode. Bits 0 and 1 of the register determine whether this mode is enabled for each E1 line:

```
Bit 0      0 : Slip invalidation mode disabled for line 0
           1 : Slip invalidation mode enabled for line 0

Bit 1      0 : Slip invalidation mode disabled for line 1
           1 : Slip invalidation mode enabled for line 1
```

TDM Rcv Mode Register (IO: 0x817)

This register controls the TDM receive mode for incoming IM2E1 time slots. The normal and default mode is to replace most of the bits of incoming time slot 0 with status information. Module FPGA configuration version 18 or higher provides an alternate mode in which both status information and the complete time slot 0 from both framers is available. See the TDM Framing section for more details. Bit 0 of the register determine whether this mode is enabled.

```
Bit 0      0 : Normal mode: Status + 31 time slots
           1 : Extended mode: Status + 32 time slots
```

5.1.2.2 TDM Interface

The primary method for transferring incoming and outgoing E1 data is via the TDM subsystem. The serial data input and output of the E1 framers are connected to the TDM data busses through elastic storage buffers, allowing the TDM clock rate to be a multiple of the E1 data rate.

Data is transferred between a TDM data bus and an elastic storage buffer on a slot-by-slot basis based on commands stored in the TDM subsystem MAP RAM. The MAP is built and loaded from the host system using the PCI library TDM functions described in **Section .XXXXX**. The control words, as stored in the TDM Map RAM, are described below.

The TDM interface logic also decodes the TDM control words generated from the TDM Map RAM on the base board. There are 8 control words for each time slot that determine the connections between the DSPs and the TDM subsystem busses. The TDM map is programmed using C functions provided in the V6M6 Host Application Library (see Section 5.1.3.1 V6M6 Host Application Library for IM2E1).

5.1.2.2.1 TDM Map Control Words

The TDM control words, as stored in the TDM Map RAM, are described below:

IM2E1 TDM MAP Control Words		
Control Word	Bits 7:4	Bits 3:0
0	TDMA-SRC	TDMB-SRC
1	TDMC-SRC	TDMD-SRC
2	MOD4-CTL	MOD5-CTL
3	MOD2-CTL	MOD3-CTL
4	MOD0-CTL	MOD1-CTL
5	MOD4-DST	MOD5-DST
6	MOD2-DST	MOD3-DST
7	MOD0-DST	MOD1-DST

TDM SRC The 4-bit code in each of the TDM Source selectors determines the device to source the TDM Bus.

0x0	Line A on Module A sources the TDM Bus
0x1	Line A on Module B sources the TDM Bus
0x2	Line A on Module C sources the TDM Bus
0x3	Line A on Module D sources the TDM Bus
0x4	Line A on Module E sources the TDM Bus
0x5	Line A on Module F sources the TDM Bus
0x6	Line B on Module A sources the TDM Bus
0x7	Line B on Module B sources the TDM Bus
0x8	Line B on Module C sources the TDM Bus
0x9	Line B on Module D sources the TDM Bus
0xA	Line B on Module E sources the TDM Bus
0xB	Line B on Module F sources the TDM Bus
0xF	No source for the TDM Bus

MOD CTL The 2-bit codes in these words control loading and unloading of the E1 data elastic buffers on the module.

Bits 1:0	Loading of outgoing buffers
11	No load
10	Load Line A buffer
01	Load Line B buffer
00	Load Line A and Line B buffers
Bits 3:2	Unloading of incoming buffers
11	No unload
10	Unload Line A buffer
01	Unload Line B buffer
00	Unload Line A and Line B buffers

MOD DST The 2-bit codes in these words select which TDM Bus is used for getting data to the outgoing E1 data elastic buffers on the module.

Bits 1:0 Bus for Data to Line A

00	TDM Bus A
01	TDM Bus B
10	TDM Bus C
11	TDM Bus D

Bits 3:2 Bus for Data to Line B

00	TDM Bus A
01	TDM Bus B
10	TDM Bus C
11	TDM Bus D

5.1.2.2.2 TDM Framing

TDM frames must always be set for 125 microseconds. They may be configured as 32 slots at 2.048 MHz, 64 slots at 4.096 MHz or 128 slots at 8.192 MHz.

TDM framing parameters are configured using host application functions described in section 4.3.1.2. When frames of 64 or 128 slots are used, it is recommended that the E1 slots be spread evenly across the available TDM time slots.

The E1 data frames, incoming and outgoing, are aligned with the TDM frames. Outgoing multiframes are synchronized with the multiframe sync generated by the TDM subsystem. Incoming multiframes are synchronized with a receive multiframe sync for each line.

It is possible to read and write the elastic buffers for both framers during the same time slot. It is also possible to unload a byte from an incoming buffer without driving the data onto a TDM bus. This allows unused data to be purged from the buffers without using up TDM bandwidth.

Exactly 32 bytes (time slots) per TDM frame must be transferred to each of the E1 line's elastic buffers. The first byte written to each buffer will be replaced when E1 time slot 0 is transmitted unless the framer is configured for FAS pass-through mode. In FAS pass through mode, the TDM data must provide the required FAS codes which include the Si and Sa bits.

Depending on the TDM receive mode selected, exactly 32 or 33 bytes (time slots) per TDM frame must be transferred from each of the E1 line's elastic buffers. The TDM receive mode is controlled by the TDM Rcv Mode register and the `im2e1_tdmrcvmode` library function.

In the normal TDM receive mode the IM2E1 provides 32 TDM time slots for each framer. The first incoming slot contains framer status information and some of the bits of the E1 alignment slot (slot 0), as described below. The other 31 E1 data slots are available as received from the RSER pin of the framers. The contents of the first TDM time slot in normal mode are shown below.

1st Rcv TDM Time Slot in Normal (STAT+31) Mode							
Bit 7	MSB first						Bit 0
Si	TDM MF	Alarm	Sa4	RLOS B	RLOS A	RSYNC B	RSYNC A

Version 18 or higher of the IM2E1 FPGA configuration provides an extended TDM receive mode with 33 receive TDM slots for each framer. In this mode the first

incoming TDM time slot contains framer status information and the second incoming TDM time slot contains the incoming E1 time slot 0, unchanged. The contents of these two tie slots are shown below.

1st Rcv TDM Time Slot in Extended (STAT+32) Mode							
Bit 7	MSB first						Bit 0
X	TDM MF	X	X	RLOS B	RLOS A	RSYNC B	RSYNC A

2nd Rcv TDM Time Slot in Extended (STAT+32) Mode (Align Frame)							
Bit 7	MSB first						Bit 0
Si	0	0	1	1	0	1	1

2nd Rcv TDM Time Slot in Extended (STAT+32) Mode (Non-Align Frame)							
Bit 7	MSB first						Bit 0
Si	1	Alarm	Sa4	Sa5	Sa6	Sa7	Sa8

For both TDM receive modes, the bit definitions are:

Si	International spare bit from slot 0 of selected line.
TDM MF	State of the MultiFrame Sync generated by the TDM Subsystem - indicates 1st frame of a transmit multiframe.
Alarm	Remote alarm bit from slot 0 of selected line.
Sa <i>n</i>	Sa bit(s) from incoming slot 0 of selected line.
RLOS A/B	Sampled output of framers' RLOS pin - indicates loss of receive sync.
RSYNC A/B	Sampled output of framers' RSYNC pin - indicates 1st frame of a receive multiframe.
X	undefined

5.1.2.2.3 TDM Clock Rates and Frame Slips

Most applications have the TDM subsystem clock, TDMCLK, derived from the clock received from an incoming E1 line. The E1 transmit clock, in turn, is usually derived from TDMCLK. The library functions, **im2e1_rcvclk**, **im2e1_xmtclk** and **im2e1_tdmclkdiv**, control the TDM and E1 clock timing. These functions are described in 5.1.3 IM2E1 Software Support.

For the majority of systems, multiple incoming E1 lines will be running at the same clock rate, being derived from a common timing source. In a few situations, however, the timing of multiple incoming E1 lines may be derived from different time bases; their clocks may be running at slightly different frequencies. Depending on which receive clock is faster, data from one of the incoming lines could be either lost or repeated as the elastic storage buffer in the framer chip under-flows or over-flows. If this condition occurs, this data loss or data repetition occurs by design at a frame boundary. In other words, an entire frame is either lost or repeated at any given time.

The IM2E1 has logic to compare the frequencies of its two receive clocks, to determine which is faster and to use the faster of the two clocks to derive the TDM subsystem clock. The **im2e1_rcvclk** function is used by host applications to select this mode of operation. With TDMCLK derived from the faster E1 receive clock, any frame slips occurring on the other E1 line will be due to elastic store under-flows and data will be repeated rather than lost. Should the frequencies of the two incoming E1 lines drift, the IM2E1 will automatically switch which of the receive clocks is used to derive TDMCLK. Furthermore, if one of the framers is not in sync, the IM2E1 will automatically select the other framer's receive clock to derive TDMCLK.

To insure that data is not lost in the case where the two incoming clock rates are very close and wavering, the derived TDM clock is adjusted to be approximately 3.8 PPM more than the faster received rate, which will cause repeated frames from both framers. This mode is available in IM2E1 FPGA revision 17 or higher. This adjustment is "on" by default but may be overridden. See the description of the Clock Ctl2 Register in section 5.2.2.1 and the **im2e1_rcvclk** library function in section 5.2.3.1.3.

Additional logic, available in Revision 2 of the IM2E1, provides a mode by which TDM data from a slipped frame is invalidated. This is accomplished using the TDM VALID signals that accompany each TDM data bus. For each TDM slot sourced by an E1 line during a slipped frame, the corresponding TDM VALID is pulsed low. Modules capable of interpreting the TDM Valid signals will be able to filter out data repeated due to slips. The slipped frame invalidation mode is enabled using the **im2e1_slipinval** function described in the IM2E1 Software Support section of this manual.

Slips are indicated to the IM2E1 logic by an interrupt from the DS2153 framer. The program controlling the IM2E1 (running either on the host or a processor on the base board) must enable and service this interrupt. To enable the interrupt, bit 4 of the framer's **Interrupt Mask Register 1** must be turned on. Servicing the interrupt requires the following standard sequence of accessing status from the framer:

Write a 1 to bit 4 of **Status Register 1**
Read **Status Register 1**
Write 1 to bit 4 of **Status Register 1**

Important notes

1. The slip interrupt must be serviced in time for the next frame slip to be recognized. The time between slips will vary depend on the frequency offset between the two incoming RCLKs. For example, with a frequency offset of 0.01%, a slip can be expected every 1.25 seconds.
2. Status Register 1 and the corresponding Interrupt Mask Register 1 contain other important status bits that an application is likely to be interested in and that may be used for other interrupt conditions.

Yet another situation for some applications is the use of more than one IM2E1 module. If the timing for all of the incoming E1 lines is from the same time base, all of the incoming E1 data will remain in sync. However, if the E1 lines are coming from different timing sources the possibility of slippage and lost data becomes an issue.

To aid in this situation, the IM2E1 compares its receive clock frequencies with the frequency of TDMCLK. When one of its two receive clocks is faster than TDMCLK, it sets a status flag which may be enabled to assert a PCI interrupt. The interrupt is enabled using the **im2e1_intenb** function. The **im2e1_intstat** function is used to read the status of this detection. The status condition is latched and remains active until cleared by calling the **im2e1_clrfastclk** function. When this interrupt is serviced, the application monitoring E1 performance may decide to switch which of the IM2E1 modules is used to source TDMCLK.

5.1.2.3 Electrical Line Interface

The IM2E1 interfaces to external E1 networks via two RJ45 connectors, one for each line. The electrical interface includes voltage clamping and current limiting devices to protect the smPCI module and base board from differential and common mode spikes and noise on the E1 signal wires.

The IM2E1 has selectable input impedance matching for the HDB3 interface. The options are 75 Ω , 120 Ω or Hi Z. The impedance is selected individually for each of the four E1 lines via jumper settings. Figure 4-2 shows the location of the impedance selection jumpers and the RJ45 connectors on the module.

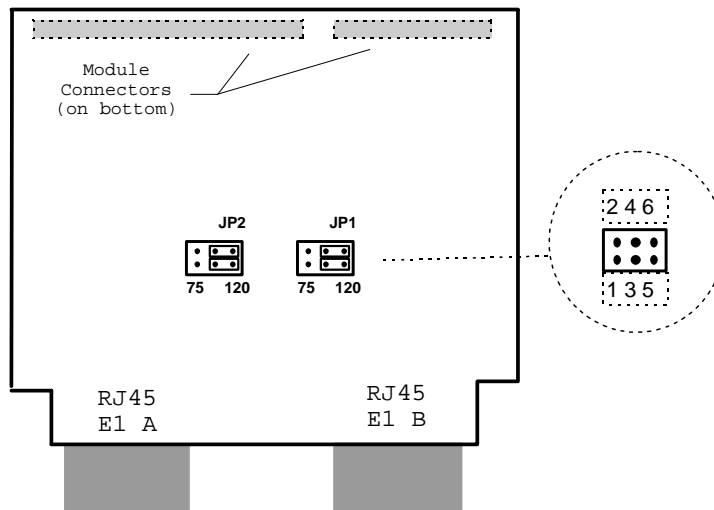
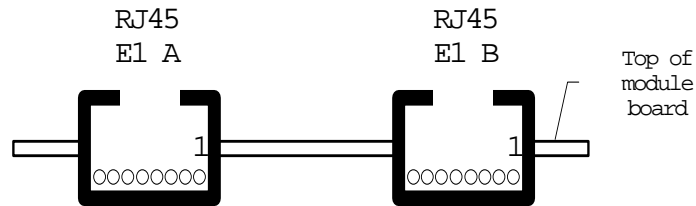


Figure 5-2: IM2E1 Line Impedance Selection

The HDB3 line input impedance is selected by positioning jumper pairs on JP1 and JP2 to connect the middle pins with the left (3-5 and 4-6) pins or right (3-1 and 4-2) pins. This will select the impedance, as indicated. Both jumpers in a pair should be positioned in the same way. The figure above shows the jumpers in their default (as shipped) positions, selecting 120 Ω impedance. High input impedance (for using the IM2E1 to monitor an E1 line without additional loading) may be selected by positioning the jumpers to connect pins 5-6 and pins 1-2.

Figure 4-3 shows the signal connections for the RJ45 connectors. In this view, pin 1 is at the left.



Pin Signal

1	Receive Ring
2	Receive Tip
4	Transmit Ring
5	Transmit Tip

Figure 5-3 IM2E1 RJ45 Pinout

5.1.2.4 LED Indicator

The IM2E1 uses the LED for its module location to indicate the following status:

RED	Module FPGAs are not configured
GREEN	Receive synchronization on one or both E1 lines

5.1.2.5 Specifications

The table below lists some of the significant performance characteristics of the IM2E1 module and its network interface components. For further details, consult the data sheets for the following components, some of which are available in Appendix A.

E1 Line Interface and Decoding and Framing:

Dallas Semiconductor DS2153Q CEPT Primary Rate Transceiver

E1 to TDM Interface:

Dallas Semiconductor DS2175 T1/CEPT Elastic Store

Parameter	Min	Typical	Max	Units
HDB3 Interface				
Line Impedance (selectable)				
Coax Cable		75		Ω
Twisted Pair		120		
Output transformer ratio		1:1.15		
Line Length			1500	meters
Output Pulse Amplitudes				V peak
75Ω coax		2.37		
120Ω twisted		3.00		
Input Surge Protection (breakover)	27		36	V
Power Requirements		TBD		Amps @ 5V
MTBF		TBD		1000 hrs

Notes:

1. Characteristics apply under ambient temperature of 0 to +55 °C.

5.1.3 IM2E1 Software Support

5.1.3.1 V6M6 Host Application Library for IM2E1

The V6M6 and V6M6HS Host Application Library includes several functions for control of the IM2E1 module, the DS2153Q framers and the TDM subsystem.

Functions operating on an IM2E1 module require a PCI_MOD structure pointer obtained by calling the **pciopen** function. The module type of the open module must be **PCI_IM2E1** (a macro defined in **<pciutil.h>**). Functions operating on the TDM subsystem require a PCI_TDM structure pointer obtained by calling the **pci_tdmopen** function.

Example:

```
#include <pciutil.h>
PCI_MOD *pci;
PCI_TDM *tdm;

pci = pciopen ("pci0a");
if (pci == NULL) {
    perror ("pci0a");
    exit (1);
}

if (pci->module_type != PCI_IM2E1) {
    printf ("IM2E1 not installed on pci0a\n");
    exit (1);
}

tdm = pci_tdmopen ("pci0");
if (tdm == NULL) {
    perror ("pci0t");
    exit (1);
}
```

5.1.3.1.1 IM2E1 Functions Quick Reference

These are the C-callable functions of the Host Application Library used for programming the IM2E1 module.

Full descriptions begin on page 5-24.

IM2E1 and Framer Control

<code>int im2e1_init(<i>pci</i>)</code>	Initializes the IM2E1 module.
<code>int im2e1_intenb(<i>pci, intmask</i>)</code>	Loads the IM2E1 interrupt mask register to specify the routing of interrupts from the IM2E1.
<code>int im2e1_intstat(<i>pci</i>)</code>	Reads the IM2E1 interrupt status register.
<code>int im2e1_clrfastclk(<i>pci</i>)</code>	Clears the status of the comparison of the receive clock frequencies and TDMCLK.
<code>int im2e1_outparams(<i>pci, line, params</i>)</code>	Loads control registers in a E1 framer from an array of values.
<code>int im2e1_lirst(<i>pci, line</i>)</code>	Momentarily activates the LIRST of the selected E1 framer.
<code>u_long im2e1_status(<i>pci, line, statmask</i>)</code>	Reads the four status registers of the selected E1 framer.
<code>int im2e1_xmtsig(<i>pci, line, data</i>)</code>	Writes data to the signaling registers in a E1 framer chip.
<code>int im2e1_rcvsig(<i>pci, line, data</i>)</code>	Reads data from the signaling registers in a E1 framer chip.
<code>u_char im2e1_inreg(<i>pci, line, reg</i>)</code>	Reads a single register from a E1 framer chip.
<code>int im2e1_outreg(<i>pci, line, reg, data</i>)</code>	Writes an 8-bit value to a single register in a E1 framer.

TDM Clock, Framing and Synchronization

<code>int im2e1_xmtclk(<i>pci, select</i>)</code>	Selects the source of the transmit clocks for the E1 lines on an IM2E1 module.
<code>int im2e1_rcvclk(<i>pci, select</i>)</code>	Selects whether or not one of the E1 receive clocks on the IM2E1 is used to derive the TDM subsystem clock.
<code>int im2e1_tdmclkdiv(<i>pci, clkdiv</i>)</code>	Sets the ratio between E1 clocks and TDM subsystem clocks.
<code>int im2e1_tdmrcvmode(<i>pci, rcvmode</i>)</code>	Sets the TDM receive framing mode.
<code>int im2e1_slipinval(<i>pci, select</i>)</code>	Enables or disables the slipped frame data invalidation mode.
<code>int pci_tdm_clksrc(<i>tdm, clksrc</i>)</code>	Selects the clock source for the TDM subsystem control logic.
<code>int pci_tdm_init(<i>tdm, clk, bits, slots, frames</i>)</code>	Initializes the parameters of the TDM subsystem controller.
<code>int pci_tdm_run(<i>tdm</i>)</code>	Starts the TDM subsystem timing logic.
<code>int im2e1_init_rsync(<i>pci, line</i>)</code>	Initializes synchronization of the incoming E1 frames to the TDM subsystem frames.

TDM Connection MAP Functions

<code>int pci_tdm_src_add(<i>tdm, slot, bus, devtyp, devnum</i>)</code>	Adds a device as a TDM data source for the specified time <i>slot</i> and TDM <i>bus</i> .
<code>int pci_tdm_src_del(<i>tdm, slot, bus</i>)</code>	Deletes a device as a TDM data source for the specified time <i>slot</i> and TDM <i>bus</i> .
<code>int pci_tdm_dst_add(<i>tdm, slot, bus, devtyp, devnum</i>)</code>	Adds a device as a TDM data destination for the specified time <i>slot</i> and TDM <i>bus</i> .
<code>int pci_tdm_dst_del(<i>tdm, slot, devtyp, devnum</i>)</code>	Deletes a device as a TDM data destination for the specified time <i>slot</i> and TDM <i>bus</i> .
<code>int pci_tdm_special_add(<i>tdm, slot, devtyp, devnum</i>)</code>	Adds reads of incoming E1 elastic store buffers which do not drive data onto a TDM bus.
<code>int pci_tdm_special_del(<i>tdm, slot, devtyp, devnum</i>)</code>	Deletes reads of incoming E1 elastic store buffers which do not drive data onto a TDM bus.

5.1.3.1.2 IM2E1 and Framer Control

The following library functions are used to control the IM2E1 module and its DS2153Q E1 framers.

```
function:  int im2e1_init(pci)
args:      PCI_MOD *pci
return:    1      IM2E1 module initialized.
           0      error indicated.
errors:    EINVAL  pci is NULL or doesn't reference an IM2E1.
           EIO     error in communicating with the module or
                   the framer chips.
```

The **im2e1_init** function initializes the IM2E1 module. This includes setting up the module's PCI interface and clearing the DS2153Q registers. It is called from the **pciinit** program. This function should not be used by an application that expects the IM2E1 to be previously configured and currently in operation.

```

function:  int im2e1_intenb(pci, intmask)

args:     PCI_MOD  *pci
          int      intmask

return:   1      interrupts enabled.
          0      error indicated.

errors:   EINVAL   pci is NULL or doesn't reference an IM2E1.
          EIO      error in communicating with the module.

```

The **im2e1_intenb** function loads the IM2E1 interrupt mask register bits to specify the routing of interrupts (from the DS2153Q framers and the comparison of receive clocks with TDMCLK) to the PCI bus interrupts. Bits turned on in *intmask* enable interrupts as follows:

im2e1_intstat mask bits			
<i>intmask</i> Bit	IM2E1 Interrupt	to	PCI Interrupt
0	Line 0 Int 1		A
1	Line 0 Int 2		A
2	Line 1 Int 1		A
3	Line 1 Int 2		A
4	Line 0 Int 1		B
5	Line 0 Int 2		B
6	Line 1 Int 1		B
7	Line 1 Int 2		B
8	RCLK > TDMCLK		A
9	RCLK > TDMCLK		B

```

function:  int im2e1_intstat(pci)
args:      PCI_MOD  *pci
return:    status
           -1      error indicated.
errors:    EINVAL   pci is NULL or doesn't reference an IM2E1.
           EIO      error in communicating with the module.

```

The **im2e1_intstat** function reads the IM2E1 interrupt status register bits which indicate the state of the interrupts from the DS2153Q framers and the comparison of receive clocks with TDMCLK. Bits turned on in the returned *intstat* indicate the interrupts as follows:

im2e1_intstat returned status	
<i>status</i> Bit	IM2E1 Interrupt
0	Line 0 Int 1
1	Line 0 Int 2
2	Line 1 Int 1
3	Line 1 Int 2
4	RCLK > TDMCLK

```

function:  int im2e1_clrfastclk(pci)
args:      PCI_MOD  *pci
return:    1      fast RCLK detection status cleared
           0      error indicated.
errors:    EINVAL   pci is NULL or doesn't reference an IM2E1.
           EIO      error in communicating with the module.

```

The **im2e1_clrfastclk** function clears the status of the comparison of the frequencies of the 2 receive clocks and TDMCLK. See Section 5.1.2.2.3 TDM Clock Rates and Frame Slips for further information.

function: int **im2e1_outparams**(*pci*, *line*, *params*)

args: PCI_MOD **pci*
int *line*
u_char **params*

return: 1 Framer registers loaded.
0 error indicated.

errors: EINVAL *pci* is NULL or doesn't reference an IM2E1,
line is invalid, *params* is NULL.
EIO error in communicating with the module.

Most applications are likely to use a fixed set of operational settings for the DS2153Q chips. The **im2e1_outparams** function provides a way to load most of control registers in the DS2153Q with an array of values.

The *line* argument (0 or 1) specifies which DS2153Q is to be loaded. The *params* argument points to an array of 48 unsigned characters containing the values to be loaded into registers 0x00 through 0x2F of the selected DS2153Q with the following exceptions and conditions:

- Values that correspond to read-only registers are ignored.
- The two Test registers are always loaded with zeros.
- Values that correspond to the Status and Receive Information registers should contain 1's for any status bits to be cleared.
- The bit corresponding to LIRST in CCR3 is ignored and replaced with a zero (use the **im2e1_lirst** function for line interface reset).

```
function:  int im2e1_lirst(pci, line)
args:      PCI_MOD  *pci
           int      line
return:    1      LIRST pulsed.
           0      error indicated.
errors:    EINVAL  pci is NULL or doesn't reference an IM2E1,
           line is invalid.
           EIO     error in communicating with the module.
```

The **im2e1_lirst** function momentarily activates the LIRST bit in Common Control Register 3 of the selected DS2153Q. This may be required during operation if a problem with the incoming signal is detected. The *line* argument (0 or 1) specifies which DS2153Q is to be reset.

```

function:  u_long  im2e1_status(pci, line, statmask)

args:     PCI_MOD  *pci
          int      line
          u_long   statmask

return:   status    framer status (see table below).
          0xffffffff error indicated.

errors:   EINVAL   pci is NULL or doesn't reference an IM2E1,
          line is invalid.
          EIO      error in communicating with the module.

```

The DS2153Q has four registers which provide status information to the host. These are Status Register 1, Status Register 2, the Receive Information register and the Synchronizer Status register. The first 3 registers require subsequent write accesses in order to clear the status information after it is read.

The **im2e1_status** function handles reading of all four status registers and clearing selectable active status bits. The *line* argument (0 or 1) specifies which DS2153Q is to be accessed.

The *statmask* argument specifies which bits of Status Registers 1 and 2 and the Receive Information register are to be read and cleared as shown in the table below. The *statmask* value does not affect the information from the Synchronizer Status register.

im2e1_status mask and returned bits		
DS2153Q Register	<i>statmask</i> bits	<i>status</i> bits
Status Register 1	0:7	0:7
Status Register 2	8:15	8:15
Receive Information	23:16	23:16
Synchronizer Status	n/a	31:24

```
function:  int im2e1_xmtsizg(pci, line, data)
           int im2e1_rcvsig(pci, line, data)

args:     PCI_MOD  *pci
           int      line
           u_char  *data

return:   1      data transferred to / from DS2153Q.
           0      error indicated.

errors:   EINVAL  pci is NULL or doesn't reference an IM2E1,
           line is invalid or data is NULL.
           EIO    error in communicating with the module.
```

The DS2153Q framers automatically extract and insert Channel Associated Signaling bits to and from registers. This operation is described in section 7 of the DS2153Q data sheet. The IM2E1 function library contains functions for reading and writing the signaling registers.

The Transmit Signaling registers (0x40 through 0x4F) are loaded using the **im2e1_xmtsizg** function. The Receive Signaling registers (0x30 through 0x3F) are read using the **im2e1_rcvsig** function.

For both functions the *line* argument specifies which DS2153Q is to be accessed. The *data* argument is a pointer to an array of 16 unsigned characters that contains the data to be written to or read from the framer.

function: u_char **im2e1_inreg**(*pci*, *line*, *reg*)

args: PCI_MOD **pci*
int *line*
int *reg*

return: *data* read from DS2153Q register.
0 error indicated.

errors: EINVAL *pci* is NULL or doesn't reference an IM2E1,
or *line* or *reg* invalid.
EIO error in communicating with the module.

The **im2e1_inreg** function may be used to read a single register from the DS2153Q specified by the *line* argument.

function: int **im2e1_outreg**(*pci*, *line*, *reg*, *data*)

args: PCI_MOD **pci*
int *line*
int *reg*
u_char *data*

return: 1 *data* transferred to DS2153Q register.
0 error indicated.

errors: EINVAL *pci* is NULL or doesn't reference an IM2E1,
or *line* or *reg* invalid.
EIO error in communicating with the module.

The **im2e1_outreg** function may be used to write an 8-bit value, *data*, to a single register in the DS2153Q specified by the *line* argument.

For both the **im2e1_inreg** and **im2e1_outreg** functions the *reg* argument may be the offset of the register in the DS2153Q chip or one of the following macros, defined in **<pciutil.h>**:

IM2E1REG_BCVC1	BVP or Code Violation Count 1
IM2E1REG_BCVC2	BVP or Code Violation Count 2
IM2E1REG_CRCE1	CRC4 Error Count 1 , FAS Error Count 1
IM2E1REG_CRCE2	CRC4 Error Count 2
IM2E1REG_EBIT1	E-Bit Count 1 , FAS Error Count 2
IM2E1REG_EBIT2	E-Bit Count 2
IM2E1REG_STAT1	Status Register 1
IM2E1REG_STAT2	Status Register 2
IM2E1REG_RECVI	Receive Information Register
IM2E1REG_RCTL1	Receive Control Register 1
IM2E1REG_RCTL2	Receive Control Register 2
IM2E1REG_TCTL1	Transmit Control Register 1
IM2E1REG_TCTL2	Transmit Control Register 2
IM2E1REG_CCTL1	Common Control Register 1
IM2E1REG_CCTL2	Common Control Register 2
IM2E1REG_CCTL3	Common Control Register 3
IM2E1REG_MASK1	Interrupt Mask 1
IM2E1REG_MASK2	Interrupt Mask 2
IM2E1REG_LICTL	Line Interface Control Register
IM2E1REG_SSTAT	Synchronizer Status Register
IM2E1REG_RCA F	Receive Align Frame
IM2E1REG_RCNAF	Receive Non-Align Frame
IM2E1REG_XTAF	Transmit Align Frame
IM2E1REG_XTNAF	Transmit Non-Align Frame
IM2E1REG_XBLK1	Transmit Channel Blocking 1
IM2E1REG_XBLK2	Transmit Channel Blocking 2
IM2E1REG_XBLK3	Transmit Channel Blocking 3
IM2E1REG_XBLK4	Transmit Channel Blocking 4
IM2E1REG_XIDL1	Transmit Idle Blocking 1
IM2E1REG_XIDL2	Transmit Idle Blocking 2
IM2E1REG_XIDL3	Transmit Idle Blocking 3
IM2E1REG_XIDL4	Transmit Idle Blocking 4
IM2E1REG_XIDLE	Transmit Idle Definition
IM2E1REG_RBLK1	Receive Channel Blocking 1
IM2E1REG_RBLK2	Receive Channel Blocking 2
IM2E1REG_RBLK3	Receive Channel Blocking 3
IM2E1REG_RBLK4	Receive Channel Blocking 4
IM2E1REG_RSIG	1st of 16 Receive Signaling Registers
IM2E1REG_TSIG	1st of 16 Transmit Signaling Registers

5.1.3.1.3 TDM Clock, Framing and Synchronization

The functions listed in this section control E1 clocks on an IM2E1 and set up the timing of the interface between IM2E1 modules and the TDM subsystem.

```
function:  int im2e1_xmtclk(pci, select)
args:      PCI_MOD  *pci
           int      select
return:    1      E1 transmit clocks selected.
           0      error indicated.
errors:    EINVAL   pci is NULL or doesn't reference an IM2E1.
           EIO      error in communicating with the module.
```

The **im2e1_xmtclk** function selects the E1 transmit clock for both E1 lines on an IM2E1 module. The *select* argument encodes the selection as shown below. When deriving E1 transmit clock from TDMCLK, the IM2E1 generates a 2.048 MHz clock, dividing the TDMCLK rate as specified with the **im2e1_tdmclkdiv** function.

im2e1_xmtclk select encoding			
Bits	controls	code	selection
3:2	TCLK A	00	RCLK A
		01	RCLK B
		10	TDM Derived
1:0	TCLK B	00	RCLK-B
		01	RCLK A
		10	TDM-Derived

```

function:  int im2e1_rcvclk(pci, select)

args:     PCI_MOD  *pci
          int      select

return:   1      RCLK to TDM clock selected.
          0      error indicated.

errors:   EINVAL   pci is NULL or doesn't reference an IM2E1.
          EIO      error in communicating with the module.

```

The **im2e1_rcvclk** function selects whether or not one of the E1 receive clocks on the IM2E1 is used to derive the TDM subsystem clock. The actual TDM clock rate can be a multiple of the selected E1 RCLK as set with the **im2e1_tdmclkdiv** function. In addition, the **pci_tdm_clksrc** function must be used to select the clock source for the TDM system control, itself.

When receiving E1 data, it is usually desirable to derive the TDM clock from the received E1 clock to ensure consistent capture of the incoming data. However, if more than one IM2E1 module is used or some other clock source must be used as the TDM master clock the module may be set to not drive the TDM clock.

When the two incoming E1 lines may be at different clock frequencies it is desirable to have the IM2E1 automatically select the faster of the two receive clocks as the source for deriving TDMCLK. See Section 5.1.2.2.3 TDM Clock Rates and Frame Slips for more information.

im2e1_rcvclk settings	
<i>select</i>	RCLK used for TDM Clk
0	none
1	RCLK A
2	RCLK B
3	Faster of RCLK A and B adjusted by +3.8 PPM
4	Faster of RCLK A and B with no adjustment

```

function:  int im2e1_tdmclkdiv(pci, clkdiv)

args:     PCI_MOD  *pci
          int      clkdiv

return:   1      Clock divider set.
          0      error indicated.

errors:   EINVAL   pci is NULL or doesn't reference an IM2E1,
              or clkdiv is invalid.
          EIO      error in communicating with the module.

```

The **im2e1_tdmclkdiv** function sets the ratio between E1 clocks and TDM subsystem clocks. For E1 transmit clocks derived from the TDM clock, it specifies the ratio between the TDM clock rate and the E1 TCLK rate. For TDM clock derived from the E1 receive clock, it determines the ratio between the selected E1 RCLK and the TDM clock.

The clock ratio is always based on an E1 clock of 2.048 MHz. Therefore the value of *clkdiv* actually specifies the TDM clock rate as:

im2e1_tdmclkdiv settings	
<i>clkdiv</i>	TDM Clock Rate
0	2.048 MHz
1	4.096 MHz
2	8.192 MHz

```

function:  int im2e1_tdmrcvmode(pci, rcvmode)

args:     PCI_MOD  *pci
          int      rcvmode

return:   mode    New or current mode.
          0       error indicated.

errors:   EINVAL  pci is NULL or doesn't reference an IM2E1,
          EIO     or clkmode is invalid.
               error in communicating with the module.

```

The **im2e1_tdmrcvmode** function sets or gets the current incoming TDM framing mode. The two possible modes are Normal (status + 31 slots) or Extended (status + 32 slots) as described in the TDM Framing section of this manual.

The *rcvmode* argument may be 0 or one of the macros defined in <pciutil.h>

```

0          return the current receive mode without changing it.
IM2E1_RCVSTAT_31  normal mode with status and 31 data slots.
IM2E1_RCVSTAT_32  extended mode with status and 32 data slots.

```

IM2E1_RCVSTAT_31 is the default mode (on power up and after calling **im2e1_init**). The extended mode (IM2E1_RCVSTAT_32) provides access to the complete data from incoming E1 time slot 0 on both framers. This mode is only available on modules configured with version 18 or higher of the IM2E1 FGPA configuration.

Unless an error occurs, the function returns the current mode setting value (IM2E1_RCVSTAT_31 or IM2E1_RCVSTAT_32). If the installed FPGA version does not support extended TDM receive mode it will return IM2E1_RCVSTAT_31 which should be considered an error if the requested mode was IM2E1_RCVSTAT_32.

```
function:  int im2e1_slipinval(pci, select)
args:      PCI_MOD  *pci
           int      select
return:    1      Slip invalidation mode set.
           0      error indicated.
errors:    EINVAL   pci is NULL or doesn't reference an IM2E1,
           EIO     or the module is prior to Revision 2
           error in communicating with the module.
```

The **im2e1_slipinval** function enables or disables the slipped frame data invalidation mode described in section 4.2.2.3. The two low bits of the *select* argument determine whether this mode is enabled or disabled for each of the two E1 lines. A 1 enables and a 0 disables the slipped frame data invalidation mode.

There is no practical reason why one line would have this mode enabled and the other would not. Therefore, normal values for *select* are 0 for no slip invalidation mode or 3 for enabling slip invalidation mode. The E1 framer chips must also be programmed to enable framer interrupt 1 on elastic store slip events as described in section 4.2.2.3.

This feature is supported on IM2E1 modules of Revision 2 or higher.

```
function:  int pci_tdm_clksrc(tdm, clksrc)  
  
args:     PCI_TDM  *tdm  
          int      clksrc  
  
return:   1      Clock divider set.  
          0      error indicated.  
  
errors:   EINVAL  tdm is NULL or clksrc is invalid.
```

The **pci_tdm_clksrc** function selects the clock source for the TDM subsystem control logic. The values for *clksrc* defined in <pciutil.h> are:

TDM_BASECLK	selects the clock generated within the controller or from the TDM expansion port.
TDM_MEZZCLK	selects a clock generated from a module on the board.

The application must be careful not to enable more than one clock driver at a time. When TDM_BASECLK is used, none of the modules should be set to drive TDM clock. When TDM_MEZZCLK is used, one and only one module must generate the TDM clock.

```

function:  int pci_tdm_init(tdm, clk, bits, slots, frames)

args:     PCI_TDM  *tdm
          int       clk, bits, slots, frames

return:   1       TDM controller initialized.
          0       error indicated.

errors:   EINVAL  tdm is NULL or other arguments are invalid.

```

The **pci_tdm_init** function initializes the parameters of the TDM subsystem controller.

With the IM2E1, use the following values for the arguments:

- clk* When an IM2E1 (or any module or TDM expansion port) is configured to generate the TDM subsystem clock this argument must be 0. To generate TDM clocks from the TDM subsystem controller a value of 2, 4, or 8 should be used. See the table below.
- bits* Bits per time slot should always be set to 8.
- slots* This value may be set to 32, 64 or 128, depending on the clock rate being used. The combination of clock rate and frame size must result in a 125 microsecond frame.
- frames* Frames per multiframe should be set to 16.

TDM clock and slots values for IM2E1	
TDM clock rate	TDM Slots per frame
2.048 MHz	32
4.096 MHz	64
8.192 MHz	128

function: int **pci_tdm_run**(*tdm*)

args: PCI_TDM **tdm*

return: 1 TDM subsystem started.
0 error indicated.

errors: EINVAL *tdm* is NULL or other arguments are invalid.
EIO failed to update the TDM MAP RAM, usually due to the absence of TDM clocks.

The **pci_tdm_run** function starts the TDM subsystem timing logic. The TDM map is updated, if necessary, and the TDM controller will begin generating the sync pulses as specified by the **pci_tdm_init** function.

function: int **im2e1_init_rsync**(*pci*, *line*)

args: PCI_MOD **pci*
int *line*

return: 1 E1 elastic store synchronized.
0 error indicated.

errors: EINVAL *pci* is NULL or doesn't reference an IM2E1,
or *line* is invalid.
EIO error in communicating with the module.

The **im2e1_init_rsync** function initializes synchronization of the incoming E1 frames to the TDM subsystem frames. It must be called after the source of TDM clocks is established and the TDM subsystem is running.

It is necessary to call **im2e1_init_rsync**, for each E1 line to be used for incoming E1 data, whenever the TDM subsystem is started or restarted.

5.1.3.1.4 TDM Connection MAP Functions

The following library functions provide the means for setting up TDM data transfer connections between IM2E1 modules and other modules.

Reading an incoming E1 data slot or status involves making one of the IM2E1 lines a TDM source during a TDM time slot. Sending outgoing E1 data involves making one of the IM2E1 lines a TDM destination during a TDM time slot.

All 32 data slots of each E1 line must be read and written during a TDM frame. One or both lines of an IM2E1 may be read or written during the same TDM time slot.

```
function:  int pci_tdm_src_add(tdm, slot, bus, devtyp, devnum)
           int pci_tdm_src_del(tdm, slot, bus)
```

```
args:     PCI_TDM  *tdm
           int      slot
           int      bus
           int      devtyp
           int      devnum
```

```
return:   1      Buffered TDM Map modified.
           0      error indicated.
```

```
errors:   EINVAL  tdm is NULL or does not reference the TDM
                subsystem of a V6M6, or other
                arguments are invalid.
           ENODEV  the specified devtyp (an IM2E1) is not
                installed at the module location specified
                in devnum.
```

These two functions add or delete a device as a TDM data source for the specified time *slot* and TDM *bus*. Only one device on a board may be the source for a TDM bus during a time slot so it is not necessary to specify the device type or unit number for the delete function.

The following arguments values are used for the IM2E1 module:

- slot* Values between 1 and the number of slots configured for the TDM subsystem frame (see **pci_tdm_init**).
- bus* One of the macros defined in **<pciutil.h>**, **TDM_BUSA**, **TDM_BUSB**, **TDM_BUSC** and **TDM_BUSD**, corresponding to the four TDM busses.
- devtyp* The device type for the IM2E1 the device type is **TDM_DEVIM2E1**. This is a macro defined in **<pciutil.h>**.
- devnum* The device unit number encodes the module location and which E1 line for the module as shown in the table below.

IM2E1 TDM device unit number	
Bit 4 = E1 Line	Bits 3:0 = Module
0 = Line A	0 = A
1 = Line B	1 = B
	2 = C
	3 = D
	5 = F

function: int **pci_tdm_dst_add**(*tdm*, *slot*, *bus*, *devtyp*, *devnum*)
int **pci_tdm_dst_del**(*tdm*, *slot*, *devtyp*, *devnum*)

args: PCI_TDM **tdm*
int *slot*
int *bus*
int *devtyp*
int *devnum*

return: 1 Buffered TDM Map modified.
0 error indicated.

errors: EINVAL *tdm* is NULL or does not reference the TDM
subsystem of a V6M6, or other
arguments are invalid.
ENODEV the specified *devtyp* (an IM2E1) is not
installed at the module location specified
in *devnum*.

These two functions add or delete a device as a TDM data destination for the specified time *slot* and TDM *bus*. A device may be a destination for only one TDM bus during a time slot so it is not necessary to specify the TDM bus for the delete function. The use of the arguments is the same as the **pci_tdm_src_add** function.

```
function:  int pci_tdm_special_add(tdm, slot, devtyp, devnum)
           int pci_tdm_special_del(tdm, slot, devtyp, devnum)

args:     PCI_TDM  *tdm
           int      slot
           int      devtyp
           int      devnum

return:   1      Buffered TDM Map modified.
           0      error indicated.

errors:   EINVAL  tdm is NULL or does not reference the TDM
              subsystem of a V6M6, or other
              arguments are invalid.
           ENODEV  the specified devtyp (an IM2E1) is not
              installed at the module location specified
              in devnum.
```

The TDM **special** functions are used to add or delete reads of incoming E1 elastic store buffers which do not drive data onto a TDM bus. These may be necessary when not all of the incoming data is being used and the TDM bandwidth is required for other data transfers. It is still necessary to specify unloading of the elastic stores to maintain slot and frame alignment. The use of the arguments is the same as the **pci_tdm_src_add** function.

5.1.3.2 MIPS Application Library

A library of functions provides access to the DS2153Q E1 framer registers and the IM2E1 interrupt mask and status by MIPS processor module applications run under the WMI MIPS Kernel.

The functions are ported from the host application library and have the same names, arguments and return values. The IM2E1 functions implemented for MIPS applications are:

```
im2e1_intenb
im2e1_intstat
im2e1_outparams
im2e1_lirst
im2e1_status
im2e1_xmtsiz
im2e1_rcvtsiz
im2e1_inreg
im2e1_outreg
```

5.1.3.3 Utility Programs

The following programs, in **\$CAC/bin**, are provided for the V6M6 PCI carrier boards. They are useful for initializing and maintaining the IM2E1 module. Detailed descriptions of these programs are available in Appendix B or online in UNIX man file format provided that you followed the setup procedures in Section 2.3.2 Programmer and User Setup.

- | | |
|-------------------|---|
| pciinit | initializes the base board and modules, instructs the on-board microprocessor to configure FPGAs on the module. It also sets up PCI address mappings. |
| pciinfo | examines the EEROMs on the base board and modules and the flash memory. It then displays all pertinent version and serial number information. |
| pciflashup | compares flash object (FPGA configurations and microprocessor program) versions in the flash memory with those residing in host files. Depending on the command line options it will either display the version numbers or update the flash memory with any new versions on disk. |

5.1.3.4 Diagnostic programs

The following diagnostic programs are being developed for the IM2E1 module. These programs will be distributed in the directory, **\$CAC/pci/diag**.

- pcie1loop** Runs a loop-back test with a single IM2E1 sending data to itself. The data is generated from, received and tested by a processor module on the same baseboard. The E1 signal may be looped back externally or internal to the DS2153Q framer.
- pcie1xmt** Transmits a known E1 data pattern generated from a processor module on the same baseboard.
- pcie1rcv** Receives E1 data and compares with the known pattern used by **pcie1xmt**.

The **pcie1loop** program is one of the components of the test and burn-in program, **pciburn**.