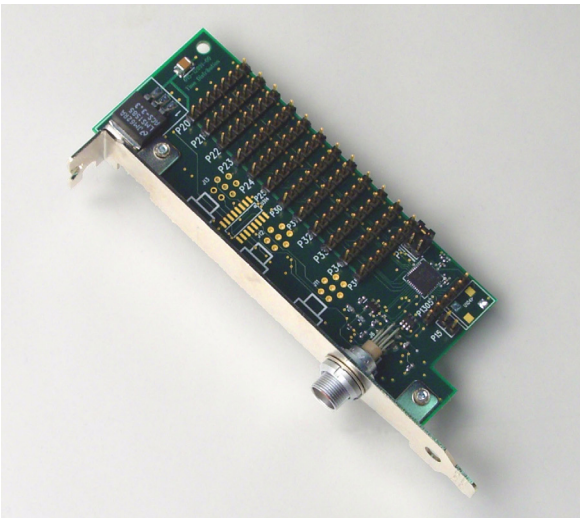
PCI GS or PCIe8 LX Time Distribution Board

for use with PCI GS or PCIe8 LX Main Board



August 28, 2008
008-02783-01

edt
a HEICO company

.

Contents

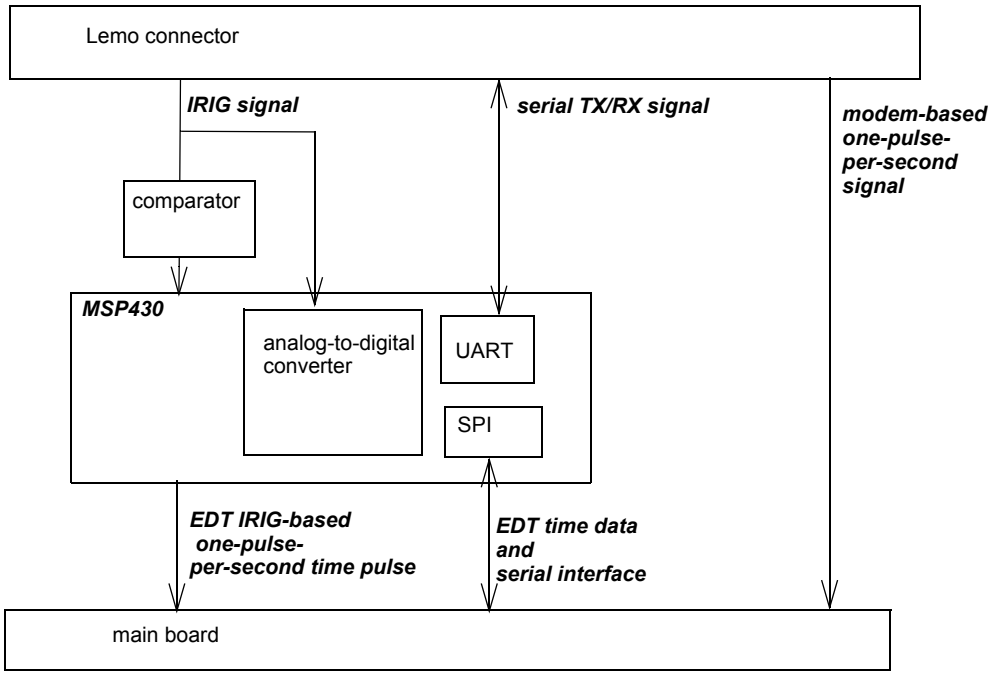
Using theTime Distribution Board

The Time Distribution board is an auxiliary board for use with the PCI GS or PCIe8 LX main board, and their compatible mezzanine boards. It enables you to receive a GPS or IRIG time code and process it in any manner you wish to implement, including sending time-stamped data to the host computer from the main board DMA.

The Time Distribution board connects to a satellite modem or other time code signal source. From there, the signal goes to a Texas Instruments MSP430F2272 microcontroller, which computes the time in either UNIX or TAI format, then sends it to the main board through an eight-pin ribbon cable.

The Time Distribution board has twelve headers for ribbon cables, allowing you to send the signal to as many as twelve PCI GS or PCIe8 LX boards. One header also permits communication back from the main board to the Time Distribution board; from there, a UART in the microcontroller enables control of the satellite modem or other signal source. Figure 1 shows a block diagram of the board.

Figure 1. Time Distribution Board Block Diagram



Installation and Setup

The Time Distribution board mounts to the attachment flange of a PCI Bus slot, but does not plug into the bus connector. Instead, it gets power through the ribbon cable and uses the main board's PCI Bus connection.

To install the Time Distribution board:

1. Install the Pcd software as specified on the software disk jacket, if you have not already done so.
2. Power down the host computer.
3. Attach the Time Distribution board to the main PCI GS or PCIe8 LX board(s) using the eight-pin ribbon cable(s) provided. One Time Distribution board can send time code signals to as many as twelve main boards.

   Connect the cable to the Time Distribution board so that the brown wire is closest to the back panel. Connect it to the main board so that the brown wire is next to the board's edge. See Figure 3.

NOTE The Time Distribution board includes one ribbon cable. If you need additional ribbon cables, order EDT part number 016-02886-00. For details, see Ribbon Cable Pinout on page 11.

4. Connect the Time Distribution board to your time signal source using the provided adapter cable: TD/Lemo/RS232/TTL/IRIGB/DB9, EDT part number 016-03005-00. For details, see Lemo-to-DB9 Cable Pinout on page 11.

NOTE For adapter cables that convert signals having different electrical characteristics, contact EDT.

5. Connect any required mezzanine boards as specified on the software CD jacket.
6. Fit the main / mezzanine board assembly into the PCI Bus connector in the host computer as specified on the software CD jacket.
7. Fit the Time Distribution board to a different PCI slot mounting flange. (It does not connect to the PCI Bus connector.)
8. Power the host back on.

NOTE To use the Time Distribution board, you must ensure that your (main and mezzanine) board pair is configured with the appropriate firmware (using `initpcd` or your own configuration utility), and that you are using a driver that is version 4.1.7.6 or later to support the board's firmware.

Included Files

The Time Distribution board ships with the board-specific software below.

`msp430_serial.vhd`
: VHDL source that shows how to receive the time code and the pulse-per-second, transmit and receive over the SPI interface, and program the MSP430 flash PROM.

`ssd16io_time.bit`
: An example of VHDL firmware that incorporates the above example code.

`msp430_load`
: The application that loads the `time_msp430_fw.txt` firmware to program the MSP430 flash PROM. This updates firmware flash only; it does not load the firmware required to operate the Time Distribution board, because that loads from flash memory on power-up or reset.

`msp430_load.c`
: The C source for the application above.

`time_msp430_fw.txt`
: The firmware that `msp430_load` loads into the flash PROM in the MSP430 microcontroller.

`time_msp430_fw.c`
: The C source for the application above.

`timing_test`
: An example application to test the board assembly and print the time to *stdout*.

`timing_test.c`
: The C source for the application above.

`libedt_timing.c`
: C source for routines that allow you to poll for the time, set and respond to interrupts, program the MSP430 flash PROM, and communicate with the modem over the UART interface. These routines in turn call routines in `libedt.c`, the EDT DMA Library. They are also documented in that file.

`libedt_timing.h`
: A header file for the C source above.

`README.time`
: A readme file listing the most up-to-date combination of main and mezzanine boards supported, as well as the appropriate driver versions.

Updating the Firmware

To update the flash PROM on the MSP430, enter:

```
msp430_load -f filename
```

For example:

```
msp430_load -f time_msp430_fw.txt
```

When prompted, press `Enter` to confirm the operation.

When loaded with `ssd16io_time.bit`, the Time Distribution board uses a base register address of 0x60; other bitfiles can use different base addresses. Also, some mezzanine boards include both the timing circuitry and an interface to the timing board. By default, `msp430_load` uses a base address of 0x60. To program the flash in alternate timing circuits, use the `-b` option.

For example, to update the MSP430 flash PROM on an MSDV mezzanine board with a base register address of 0xA4, enter:

```
msp430_load -b a4 -f filename
```

As an argument to the `-b` option, the 0x hexadecimal indicator is optional, and the argument is not case-sensitive.

To see all usage options, enter `msp430_load -help`.

NOTE You need not run `msp430_load` for the Time Distribution board unless you need to update the firmware on the MSP430 microcontroller.

Building Applications

Executable files and Pcd source files are at the top level of the EDT PCD distribution directory. To rebuild any programs, run `make` in this top-level directory.

Source code for the MSP430 microcontroller was compiled using IAR Workbench. Source code compiled using TI_CCE Workbench is available on request; contact EDT.

The example VHDL bitfile was compiled with Xilinx Project Navigator Release 9.2, service pack 0.2i.

To make use of the example VHDL in your own firmware:

- Insert the VHDL in `msp430_serial.vhd` into your own design.
- Connect the signals in the VHDL as needed, depending on the mezzanine board you're using.
- Implement the necessary registers (described in Registers on page 7) at free addresses, also depending on the mezzanine board you're using.

Testing

The example application `timing_test` provides a way to test the entire board assembly and make sure it's working correctly. It prints the time to *stdout*.

To run the example application, at the Pcd Utilities or shell prompt, enter:

```
timing_test
```

An interactive command menu is printed with each prompt.

When loaded with `ssd16io_time.bit`, the timing board uses a base register address of 0x60; other bitfiles can use different base addresses. Also, some mezzanine boards include both the timing circuitry and an interface to the timing board. By default, `timing_test` uses a base address of 0x60. To test alternate timing circuits, use the `-b` option.

For example, to test the timing circuit on an MSDV mezzanine board with a base register address of 0xA4, enter:

```
timing_test -b a4
```

As an argument to the `-b` option, the 0x hexadecimal indicator is optional, and the argument is not case-sensitive.

To aid your development efforts, C source code for the example application is included.

The Time Distribution board has no board ID.

Time Code Formats

The Time Distribution board can accept time code inputs in IRIG-B (Inter Range Instrumentation Group mod B), and outputs a 64-bit integer specifying the number of seconds since midnight, January 1, 1970, based on the 32-bit UNIX time format.

NOTE Time code signals from GPS receivers, or in other IRIG formats, can also be accommodated; contact EDT if this is of interest.

By default, IRIG-B time coding is a binary-coded decimal number representing the date in days, hours, minutes and seconds. It's based on GMT time, which is synchronized to the Earth's rotation, and therefore includes calendar adjustments and leap seconds.

The microcontroller on the Time Distribution board translates the incoming time code into a 64-bit integer specifying the number of seconds since midnight, January 1, 1970 — the same time base used by UNIX.. Your application can therefore easily convert this to UNIX time, which is a signed 32-bit integer respresenting the number of seconds since the same time base.

Serial Peripheral Interface (SPI)

The Time Distribution board provides synchronous serial data input and output using the Serial Peripheral Interface (SPI). Input is available on all headers; output is available on the master header only. This interface enables you to:

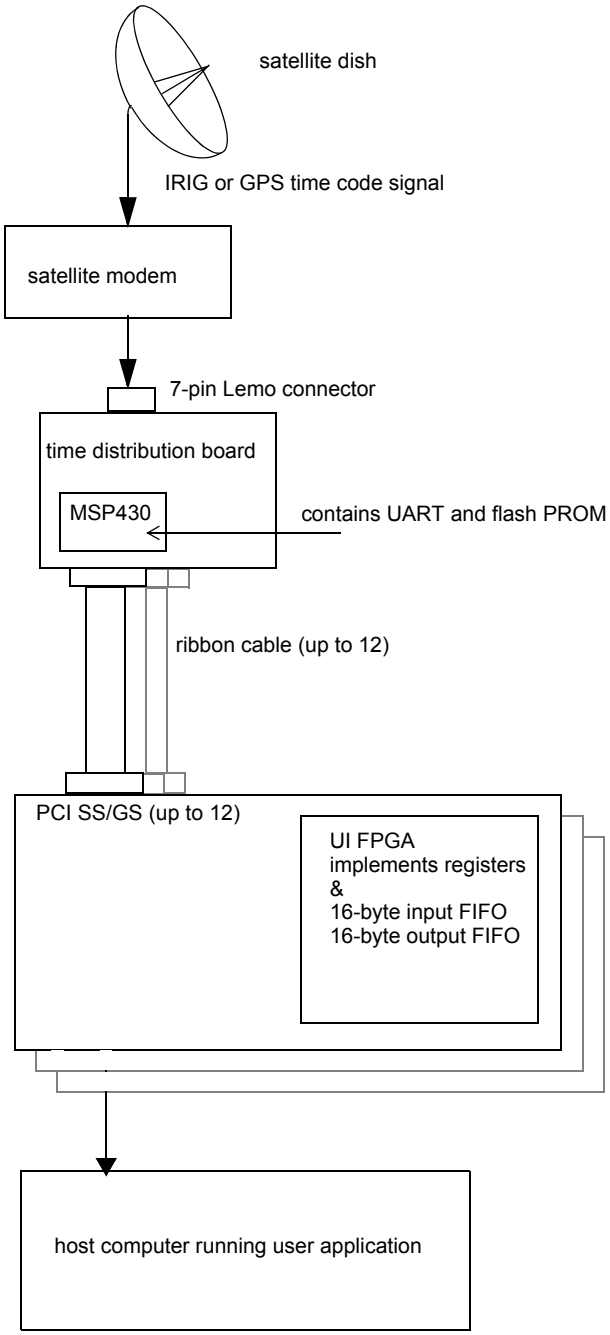
- receive time data;
- control the modem or other signal source by means of the MSP430 UART (see Connector Pinouts on page 10);
- query its status;
- change time code signal formats; and
- program the flash PROM in the MSP430.

The serial data path is implemented using two 16-byte FIFOs, one for input and one for output. The input FiFO must be serviced often enough to prevent overflow and subsequent loss of data. However, the device driver and software library buffers incoming data to prevent such overflow.

The application `timing_test.c` contains an example of using the serial interface.

Figure 2 shows the SPI signal path.

Figure 2. SPI Signal Path



SPI Packet Communications

Beginning with MSP430 firmware version 3, the MSP430 microcontroller supports CRC-checked packet communication, either with host application software or with the user interface FPGA.

Packets to and from the microcontroller over the SPI bus always start with:

```
#define SPI_PKT_START 0x82                                  // Start of packet byte.
```

followed by a command buffer, followed by a two-byte command buffer CRC check.

The command buffer consists of the following:

Byte 0 is the command byte.

Byte 1 is the number of data bytes 0 - 60.

Bytes 2-62 are command-specific data bytes.

Currently supported commands are:

```
#define SPI_PACKET_LOOP 0
```

Loop back packet; send back the packet. Data size 0-60 bytes; no ACK/NAK response.

```
#define SPI_TIMECODE_EN 1
```

Enable (1) or disable (0) the 1 Hz timestamp packet. Data size 1 byte. ACK/NAK response.

```
#define SPI_CLK_SELECT 2
```

Select MSP430 clock source as internal DCO (0) or external XIN (1) plus clock rate. ACK/NAK response.

Data size 5 bytes; first byte selects internal/external, last 4 bytes contain an integer specifying the clock rate in Hz; LS byte first; MS byte last.

Currently supported clock rates for DCO (0) are 1, 8, 12 and 16 Mhz.

Currently supported clock rate for XIN (1) is 10 Mhz.

```
#define SPI_TIMECODE_PKT 3
```

Sent from MSP430 to FPGA. Data consists of 4-8 bytes of timecode LSByte first, extended UNIX seconds time format.

The CRC check bytes apply to the command buffer (command, length and data bytes), and are computed as follows where CKL is the low CRC byte, CKH is the high CRC byte, B1 is the Command byte and Bn is the last data byte (derived from Texas Instruments Application Report SLAA089D "Features of the MSP430 Bootstrap Loader"):

CKL = INV [B1 XOR B3 XOR ... XOR Bn–1]

CKH = INV [B2 XOR B4 XOR ... XOR Bn]

For code examples, see *timing_test.c*, *libedt_timing.c*, and *libedt_timing.h* in this driver distribution.

Registers

The Time Distribution board has a 16-byte input/output FIFO for receiving time data from the time code microcontroller, and communicating with the microcontroller or the modem, or reprogramming the flash PROM in the main board UI FPGA.

To enable SPI-based synchronous serial communication, your VHDL must implement the registers below. For an example of such an implementation, see `ssd16io.vhd`.

SPI Data Register

Size	8-bit
I/O	read-write
Address	0x60
Access	SPI_DATA
Comment	When read, bits read from the input FIFO. When written, bits to write to the output FIFO.

SPI Status and Control Register

Size	8-bit
I/O	read-write
Address	0x61
Access	SPI_STAT_CTRL

Bit	Name	Description
7	CHIP_ENABLE	Detects which main board is connected to the Time Distribution board master header. A value of one indicates the master header; otherwise reads zero.
6	IRIGB_PULSE_IN	When set, indicates that the one-pulse-per-second signal has been detected from the incoming IRIG signal, and it has been processed by, and passed through, the MSP430.
5	MODEM_PULSE_IN	When set, indicates that the one-pulse-per-second signal has been detected from the satellite modem and passed directly through, without any processing by the MSP430.
4		not used; reads as zero
3	FIFO_OUT_EMPTY	When set, indicates the output FIFO is empty.
2	FIFO_OUT_FULL	When set, indicates the output FIFO is full.
1	FIFO_IN_EMPTY	When set, indicates the input FIFO is empty.
0	When written: RESET	On write: toggle this bit to reset the SPI data path.
	When read: FIFO_IN_OV	On read: when set, indicates the input FIFO has overflowed. Data may be lost.

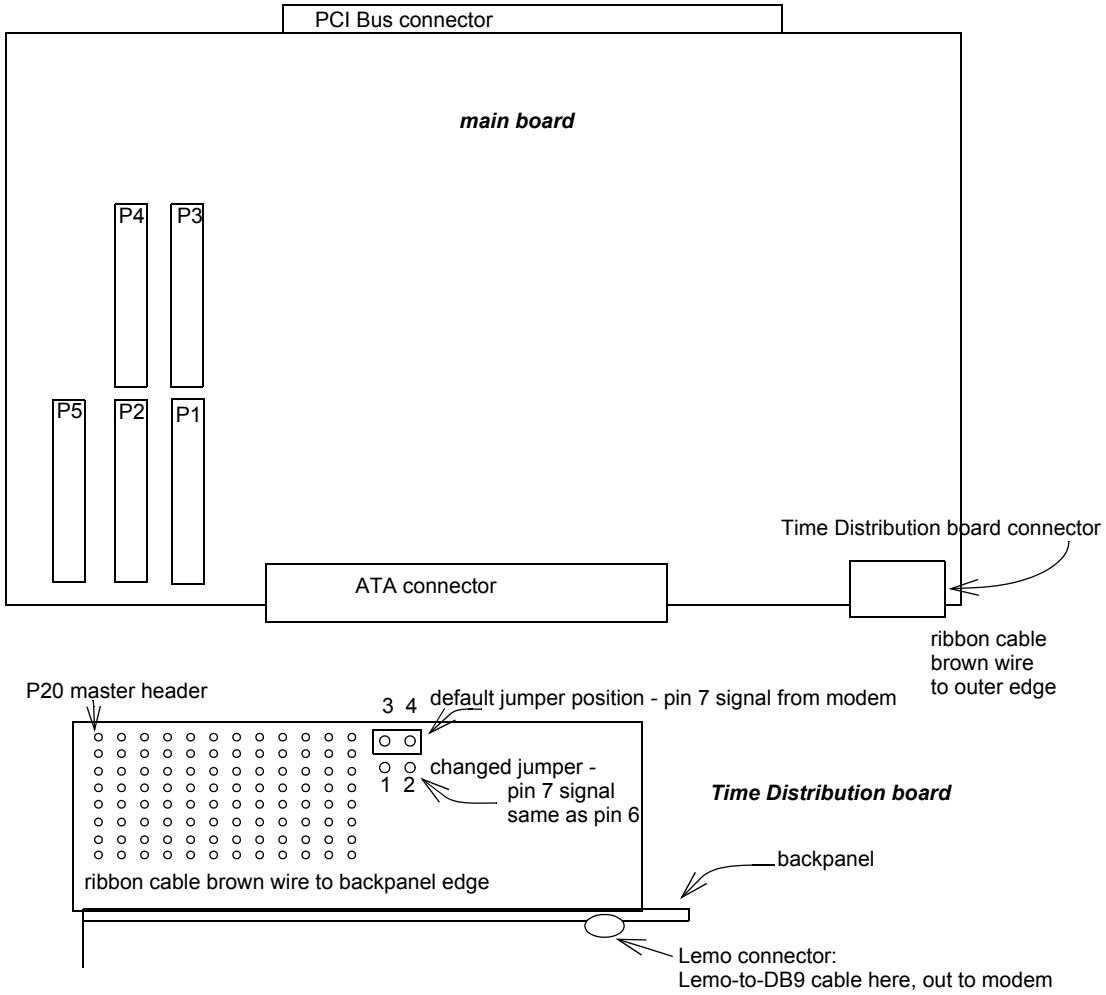
SPI Strobe Register

Size	8-bit
I/O	write only
Address	0x62
Access	SPI_STROBE
Comment	Write (any value) to this register to advance the input FIFO.

Connector Pinouts

The Time Distribution Board connects to the main board using 8-pin headers and ribbon cables, and to the signal source using a 7-pin Lemo connector and a Lemo-to-DB9 cable. Figure 3 shows the connectors and the jumper that enables changing from GPS to IRIG pulses on header pin 7.

Figure 3. Connector Locations



In the following subsections:

Table 1 shows the signals from the Time Distribution board to the header connectors, and from there to the user interface FPGA on the PCI GS or PCIe8 LX main board.

Table 2 shows the signals received through the 7-pin Lemo connector.

Table 3 shows the signals received through the 9-pin DB9 connector from the satellite receiver.

Ribbon Cable Pinout

In the ribbon cable, EDT part number 016-02886-00, pins 3 and 4 — the clock/data pair — give the time. Pin 6 or 7 is the time pulse, indicating when the time given is valid.

Table 1. Time Distribution Board Header to Main Board UI FPGA Pinout

PCIe8 LX UI FPGA Pin	Header Pin	PCI GS UI FPGA Pin	Signal
	1		+5 V
J25	2	G14	not used
K25	3	H14	SPI clock from MSP430
L26	4	G15	SPI data from MSP430
K27	5	H15	***P20 (master header):*** SPI data into MSP430 ***other headers:*** not connected
J27	6	F18	1 Hz IRIG pulse from MSP430
J28	7	G18	***default jumper position (jumper connects 3 and 4):*** 1 Hz GPS pulse from straight through from modem j***umper connects 1 and 2:*** 1 Hz IRIG pulse from MSP430
	8		ground

Lemo-to-DB9 Cable Pinout

Lemo-to-DB9 cable, EDT part number 016-03005-00, has active circuitry to convert the electrical characteristics of the one-pulse-per-second single-ended 50-Ω signal to LVDS; it converts the RS-232 serial signals to low-voltage TTL and sends the IRIG signal straight through.

Satellite modems vary; if you need a cable that accepts different signal level inputs, contact EDT.

Lemo pin numbers map to physical locations as shown in Figure 4 (looking into the connector).

Figure 4. Lemo Pin Layout.

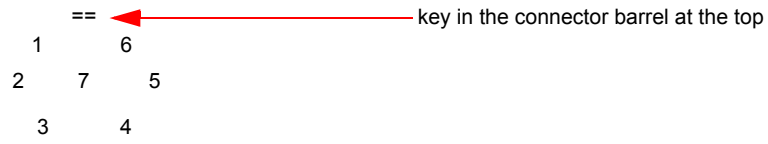


Table 2. Lemo Pinout

Lemo Pin	Signal
1	GPS transmit — low-voltage TTL
2	GPS receive — low-voltage TTL
3	1 pulse per second, positive — LVDS
4	1 pulse per second, negative — LVDS
5	IRIG
6	+4 V
7	ground

Table 3. DB9 Pinout

DB9 Pin	Signal
1	not used
2	RS-232 receive
3	RS-232 transmit
4	1 pulse per second, TTL
5	ground
6	IRIG signal
7	not used
8	not used
9	not used

References

Time code microcontroller
Texas Instruments MSP430F2272 Ultra-Low Power Microcontroller general info at:
http://www.msp430.com

User Guide at:
http://www.ti.com

IRIG-B
General description of IRIG-B standard at:
http://irigb.com

Description of IRIG-B serial time code formats at:
http://handle.dtic.mil/100.2/ADA346250