HOLT INC.
INTEGRATED CIRCUITS

ADK-6130PCIe User's Guide –

MIL-STD-1553 PCIe Evaluation Card

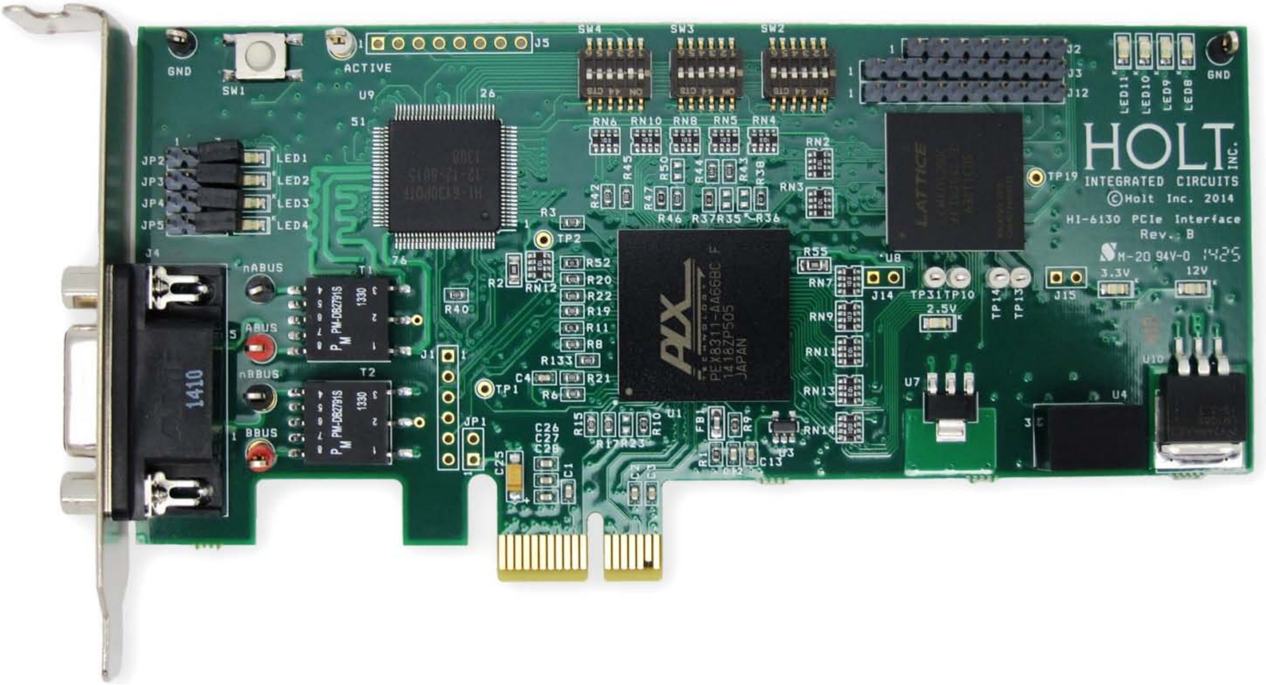
November 2017

REVISION HISTORY

Revision	Date	Description of Change
AN-6130PCIe, Rev. New	09/27/13	Initial Release.
Rev. A	06/30/14	Revise for API demo program changes.
Rev. B	03/04/15	Update board photo on page 1.
Rev. C	02/02/16	Updates for Holt API 2.x and PLX SDK 7.2.
Rev. D	11/02/17	Updates PLX driver links. Update document to new format.

Introduction:

The Holt HI-6130 MIL-STD-1553 PCIe Card is a PC interface card designed to interface the Holt HI-6130 MIL-STD-1553 multi-terminal to a single lane x1, x4/8 or x16 PCI Express (PCIe) 1.1 slot on a PC running Windows 7. The HI-6130 is a single supply 3.3V rail BC/MT/RT1/RT2 Multi-Terminal device for MIL-STD-1553 dual redundant bus communications. The card is bundled with the Holt high-level API software library and two demo programs. The two demo projects provided on the included CD-ROM demonstrate the basic features of the HI-6130 and the Holt API software library. The demo card and software can be used as starting point for any new custom design.



Evaluation Kit Contents

- This Application Note, AN-6130PCIe User's Guide.
- Holt API library software and user manual.
- ANSI C HI-6130PCIe test demo project.
- ANSI C HI-6130 PCIe API demo project.
- 9 Pin-D to MIL-STD-1553 BNC breakout cable.
- Full size PCI card optional bracket.
- CPLD Verilog project files.
- CD-ROM containing supporting documentation and software.

Topics:

- Introduction
- What's new in this release
- Board setup
- Quick Start Guide
- Hardware
- Programming Reference
- Software
- Customization
- Summary
- Schematics and BOM

New in this release:

- Updated PLX SDK from 7.11 to 7.20.
- Holt API library is updated to Rev. 02-1-2.
- Holt API Demo updated to Rev. 2.0.
- Holt API Demo updated to demonstrate APIs.

Board Default Setup:

Set SW2 position 6 set to Off (up) position**.** This sets the HI-6130 input pin AUTOEN low which is required for the demos to work properly. See the picture below.

JP6 and JP7 will be open. Optionally jumper these when necessary to have the negative side of the bus transformers (nXBUS) grounded for testing purposes only.

JP2-JP5 are not used or installed. See the PLX PEX8311 PCI Express documentation for usage.

Quick Start Guide

This board communicates with a PC using a PCIe bridge chip made by PLX Technology which is now owned by Broadcom (broadcom.com).

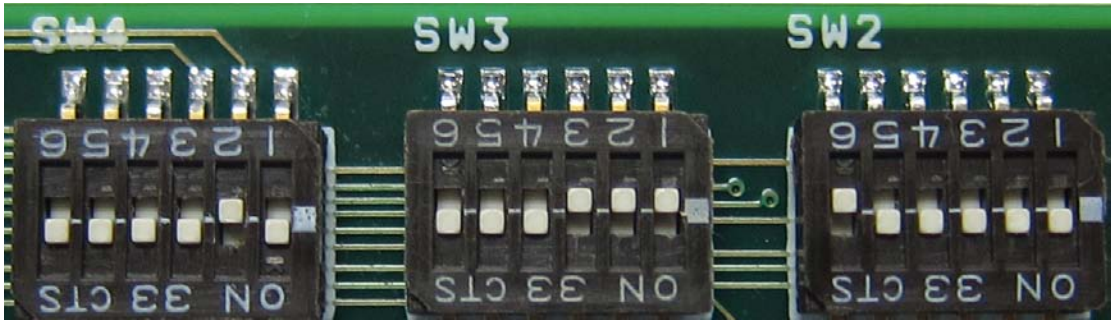
Install the PLX SDK by following the instructions in the Software section "PLX SDK installation" of this guide on page 13. A prebuilt Windows 7 compatible executable demo program is included on the CD-ROM in the folder "Holt HI-6130 PCIe Demo". Copy this folder to the desktop.

Confirm the factory default DIP switch settings: (logic-0 is down, logic-1 is up)

RT1 address = 3. SW3 1-3 set Off, 4-6 set On.
RT2 address = 1. SW4 all set On except position 2 which is set Off. (This was changed from previous release)

SW2 Positions 1-5 are user defined. These may be used by the demo program in future releases.

Default DIP switch settings

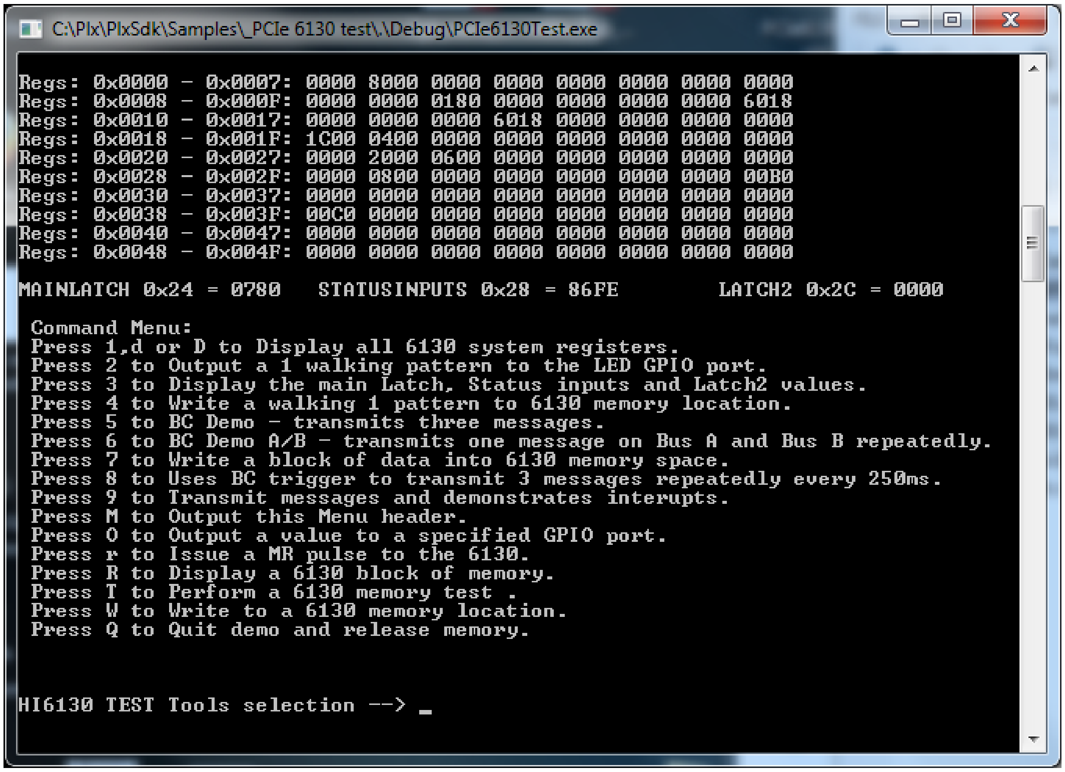


Metal brackets are provided for both full-height and low profile PCIe card slots. Use the correct bracket for your PC slot.

With the PC unpowered, plug the card into a PCIe x1, x4/8 or x16 card slot and fasten the card with the bracket screw so the card is secure. After powering up the PC, Windows automatically detects the new hardware and uses the driver installed by the PLX SDK installation. If issues are encountered with this demo proceed to page 13 “PLX SDK installation” and use Visual Studio to launch the program after installing the PLX SDK.

After the driver is installed, launch the PCIe6130Test application (.exe) by double clicking on the file located in the Holt HI-6130 Demo folder (on the desktop). A menu will be displayed showing sets of numbers 1 though x. The Holt card typically appears as the first item #1 with “9056 10b5” [b:xx s:xx f:xx]. Enter “1” and press Enter.

The main command menu displayed in the console window is shown below.


C:\Plx\PlxSdk\Samples_PCIe 6130 test\.\Debug\PCIe6130Test.exe

```
Regs: 0x0000 - 0x0007: 0000 8000 0000 0000 0000 0000 0000 0000
Regs: 0x0008 - 0x000F: 0000 0000 0180 0000 0000 0000 0000 6018
Regs: 0x0010 - 0x0017: 0000 0000 0000 6018 0000 0000 0000 0000
Regs: 0x0018 - 0x001F: 1C00 0400 0000 0000 0000 0000 0000 0000
Regs: 0x0020 - 0x0027: 0000 2000 0600 0000 0000 0000 0000 0000
Regs: 0x0028 - 0x002F: 0000 0800 0000 0000 0000 0000 0000 00B0
Regs: 0x0030 - 0x0037: 0000 0000 0000 0000 0000 0000 0000 0000
Regs: 0x0038 - 0x003F: 00C0 0000 0000 0000 0000 0000 0000 0000
Regs: 0x0040 - 0x0047: 0000 0000 0000 0000 0000 0000 0000 0000
Regs: 0x0048 - 0x004F: 0000 0000 0000 0000 0000 0000 0000 0000

MAINLATCH 0x24 = 0780   STATUSINPUTS 0x28 = 86FE        LATCH2 0x2C = 0000

 Command Menu:
 Press 1,d or D to Display all 6130 system registers.
 Press 2 to Output a 1 walking pattern to the LED GPIO port.
 Press 3 to Display the main Latch, Status inputs and Latch2 values.
 Press 4 to Write a walking 1 pattern to 6130 memory location.
 Press 5 to BC Demo - transmits three messages.
 Press 6 to BC Demo A/B - transmits one message on Bus A and Bus B repeatedly.
 Press 7 to Write a block of data into 6130 memory space.
 Press 8 to Uses BC trigger to transmit 3 messages repeatedly every 250ms.
 Press 9 to Transmit messages and demonstrates interupts.
 Press M to Output this Menu header.
 Press O to Output a value to a specified GPIO port.
 Press r to Issue a MR pulse to the 6130.
 Press R to Display a 6130 block of memory.
 Press T to Perform a 6130 memory test .
 Press W to Write to a 6130 memory location.
 Press Q to Quit demo and release memory.


HI6130 TEST Tools selection --> _
```

Press “1” to perform a HI-6130 register memory dump. This displays all the HI-6130 system registers from 0x0000 to 0x0047 which initialize to default values after a master reset. Default values are specified in the HI-6130 data sheet. Notice the word at address 0x0001 contains 0x8000. This is the Master Status and Reset Register (0x0001) in the HI-6130. The MSB bit-15 high indicates the HI-6130 READY is high which means the device is ready for the host to access the memory and registers in the device. See the HI-6130 data sheet for more details on registers and status bit definitions.

The HI-6130 bus controller is initialized with several predefined MIL-STD-1553 messages.

Press “5” to command the BC to transmit three (commands) messages to RT address 3. Message 1 is a Receive command with 32 data words to subaddress SA1 on Bus A, Message 2 is a Transmit command with 32 data words from SA1 on Bus B and Message 3 is a Receive command with 32 data words to SA1 on Bus A.

If a separate RT terminal or other RT-capable MIL STD 1553 test equipment is set for RT address 3 and connected to the bus through a suitable bus coupler, the message responses can be monitored. The program outputs transacted BC data blocks, BC instruction lists and HI-6130 system registers to the console after each transmission. To view the transmissions on an oscilloscope, trigger rising edge on the ACTIVE test point on the top left side of the card; put another scope probe on the ABBUS test point. Use a small clip lead to bring this signal up to make it accessible to the scope probe. If no external RT or test equipment is connected to the bus then use a 70 ohm termination resistor on the cable output or the signal will be distorted when viewed with the oscilloscope.

Press”8” to command the BC to transmit three messages repeatedly to RT address = 3. Messages 1 and 3 use Bus A and message 2 uses Bus B.

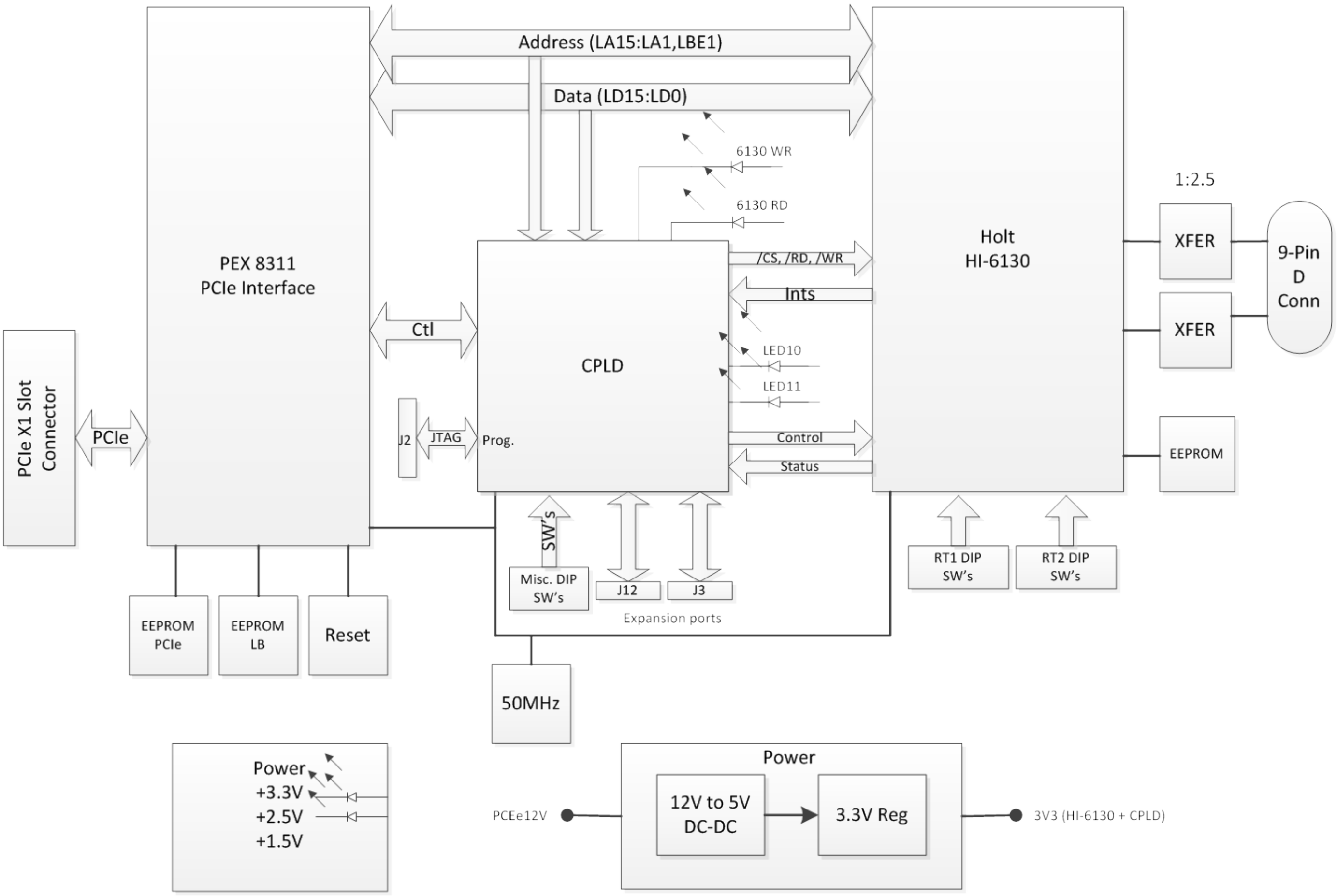
During transmissions LEDs 10 and 11 on the top edge of the card count in a binary fashion according to the message sent. Two other LEDs flash when the HI-6130 is read or written by the program to provide a visual aid during software development. LED 8 flashes when a read occurs and LED 9 flashes when a write occurs.

Press “q” to quit.

This is the end of the Quick Start Guide section.

Hardware

The Holt HI-6130 PCIe interface card consists of three main IC components shown below in the Block Diagram. The PEX 8311 is a PCI Express-to-Generic Local Bus bridge and provides the interface between the PCIe slot and the local bus (LB). A programmed CPLD translates the LB signals into CSn, RDn and WRn strobe signals for the HI-6130 timings. The CPLD also provides other GPIO and glue logic. A shared single 50MHz oscillator module provides the clock for the PEX8311 local bus, CPLD and the HI-6130. The HI-6130 must be clocked at 50MHz.



HI-6130 PCIe Card Block Diagram

The PEX8300 PCIe bridge is used in “EndPoint” mode which operates in slave mode relative to the host PC. A full SDK and RDK reference board design including drivers, documentation and demo software is available from PLX. This PLX RDK was used as the basis for the Holt card. An EEPROM (U6) contains configuration data which the PEX8311 latches in at power up to configure the LB for the target hardware base address space, data bus width, and the number of wait states. The data bus width is configured for 16-bits to match the width of the HI-6130. A second EEPROM is also connected to the PEX8311 for optional parameters to enable special configurations mainly for the PCIe side. A second EEPROM (U2) is installed for optional PCIe bus configuration but it may or may not be used. PLX recommends installing this EEPROM just in case of future needs. These EEPROMs can be examined and programmed using the PlxMon utility. The PLX utility program “PlxMon.exe” is located in the\Plx\ …\Bin folder after the SDK is installed. Information on how to use this utility is located in the PlxSdkUsersManual document in section 4 “PLX Debug Utilities”. Holt uses this utility to program the two EEPROMs.

The HI-6130 uses a 16-bit data bus, 16-bit address bus and three more bus control input signals during reads and writes. The CS0n, WRn and RDn signals are decoded by the CPLD and routed to the HI-6130. Only the upper address lines (LA31:LA26) of the LB are decoded by the CPLD for the chip selects. The LB address and data lines connect directly from the PEX8311 to the HI-6130. In addition to these lines are several dedicated inputs and outputs listed below going to the HI-6130 from the CPLD. For a complete description of the HI-6130, refer to the Holt data sheet and application notes that are included on the CD-ROM.

The functions provided by the CPLD:

- HI-6130 bus interface signals CSn, RDn and WRn meeting the HI-6130 timings.
- HI-6130 write and read access LEDs.
- Output latches with read-back for control signals to the HI-6130 and LEDs.
- HI-6130 status and DIP switch inputs.
- Interrupt source pin. The CPLD logically OR’s the HI-6130 IRQ, RT1MC8 and RT2MC8 signals into a single signal that is connected to the interrupt pin of the PLX8311. See the software section for information how PLX API’s handles interrupts. PLX includes a demo project “LocalToPciInt” to demonstrate how interrupts are handled. The technique used in “LocalToPciInt” was used in the Holt demos.

Programming Reference (LB decoded addresses)

HI-6130 chip select (R/W) – 0x2000-0000 – 0x2000-07FFF (32K Words).

Main Output Control Latches (R/W) – 0x2400-0000

BIT	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FIELD	**TEST**	**EECOPY**	**“K3”**	**LED2**	**LED1**	**RT2ENA**	**RT1ENA**	**BCENA**
RESET	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**

BIT	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FIELD	**MR**	**MTRUN**	**TXINHA**	**TXINHB**	**RAMDEC**	**TP31**	**ACKIRQ**	**BCTRING**
RESET	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**

DO	BCTRIG	HI-6130 input. Rising edge triggers the BC to execute next Opcode instruction. Usually used to start BC transmissions.
D1	ACKIRQ	HI-6130 input.
D2	TP31	CPLD spare pin.
D3	RAMEDC	HI-6130 Error detection/correction input. Set Low for this program.
D4	TXINHB	HI-6130 Bus B inhibit input.
D5	TXINHA	HI-6130 Bus A inhibit input.
D6	MTRUN	HI-6130 MT enable input.
D7	/MR	HI-6130 Master Reset input.
D8	BCENA	HI-6130 BC enable input.
D9	RT1ENA	HI-6130 RT1 enable input.
D10	RT2ENA	HI-6130 RT2 enable input.
D11	/LED1	General purpose LED (LED10 on board). On (low) at power up.
D12	/LED2	General purpose LED (LED11 on board). Off (high) at power up.
D13	“K3”	Not used but brought out to a pad on the PCB from the CPLD.
D14	EECOPY	HI-6130 EECOPY input.
D15	TEST	HI-6130 TEST input. Must be set Low for normal operation. See data sheet for Test Mode details.

Status Inputs (R only) - 0x2800-0000

BIT	15	14	13	12	11	10	9	8
FIELD	“A4”	SW2-5	SW2-4	SW2-3	SW2-2	SW2-1	“D5”	AUTOEN

BIT	7	6	5	4	3	2	1	0
FIELD	N/A	IRQ	RT2MC8	RT1MC8	WAIT	READY	MTPKTRDY	ACTIVE

DO	ACTIVE	HI-6130 outputs a high when the BC or RT is processing a 1553 message.
D1	MTPKTRDY	HI-6130 output. Monitor Packet active high when message complete.
D2	READY	HI-6130 output. Goes high after reset when the host can configure the device.
D3	WAIT	HI-6130 output. Bus control signal not used by this design.
D4	/RT1MC8	HI-6130 output. Outputs a pulse when RT1 receives a Mode Code 8 is reset.
D5	/RT2MC8	HI-6130 output. Outputs a pulse when RT2 receives a Mode Code 8 is reset.
D6	/IRQ (6130)	HI-6130 interrupt output.
D7	N/A	Not defined.
D8	AUTOEN	Set by the SW2 DIP switch 6. Input to HI-6130 for auto initialization from EEPROM.
D9	“D5”	Not used by connected to a pad on the PCB from “D5” to the CPLD.
D10	SW2-1	DIP SWITCH user defined.
D11	SW2-2	DIP SWITCH user defined.
D12	SW2-3	DIP SWITCH user defined.
D13	SW2-4	DIP SWITCH user defined.
D14	SW2-5	DIP SWITCH user defined.
D15	“A4”	Not used but is connected to a pad from “A4” to the CPLD.

Secondary Output Latches R/W – 0x2C00-0000

BIT	15-D5	4	3	2	1	0
FIELD	N/A	RT2LOCK	RT2SSF	MTSTOFF	RT1LOCK	RT1SSF
RESET	N/A	0	0	0	0	0

DO	RT1SSF	RT1 Subsystem Fail input.
D1	RT1LOCK	RT1 RT address input lock input.
D2	MTSTOFF	HI-6130 memory test disable. Set low by internal pull-down resistor.
D3	RT2SSF	RT2 Subsystem Fail input.
D4	RT2LOCK	RT2 RT address input lock input.
D5:D15	N/A	Not defined.

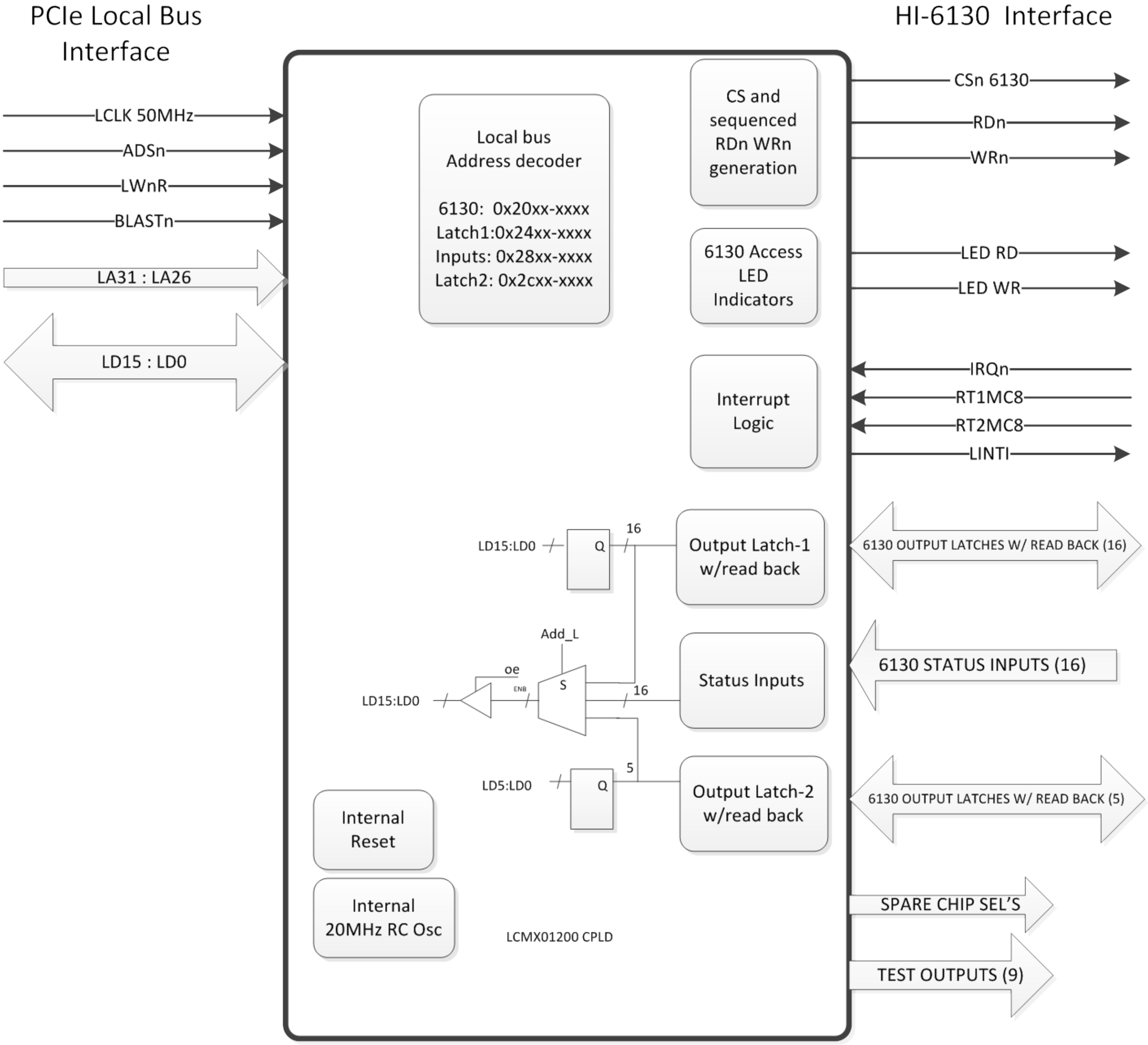
The CPLD is a Lattice LCMX01200 256-pin BGA device and is programmed through a JTAG port on J2. The Verilog source code listing is included in this document and the actual Verilog source and constraint files are included in the Lattice "Diamond" tool project included on the CD-ROM. The CPLD block diagram is shown below. The CPLD has an internal reset and RC clock generator which is used in Verilog design.

Up to 16 wait states can be programmed in the LB timing controlled by the PEX8311 for the HI-6130, output latches and input buffers in the CPLD. The value of 14 (0xD) is programmed in the LB EEPROM (U6). At 50MHz the access time is 1+14 or 1/50MHz * 15 = 300ns. This meets the worst case 240ns timing requirement of the HI-6130 for non-sequential read cycles with 60ns of margin. The LB BLASTn signal is used by the CPLD to time when to de-assert the CSn, RWn or WRn signals to the HI-6130 and the internal latches and input buffers. The ADSn signal from the LB is used by the CPLD to start the bus cycle. A faster access time could have been used for the GPIO but was kept the same to simplify the design. See the PEX8311 data book for the LB signal descriptions and timings.

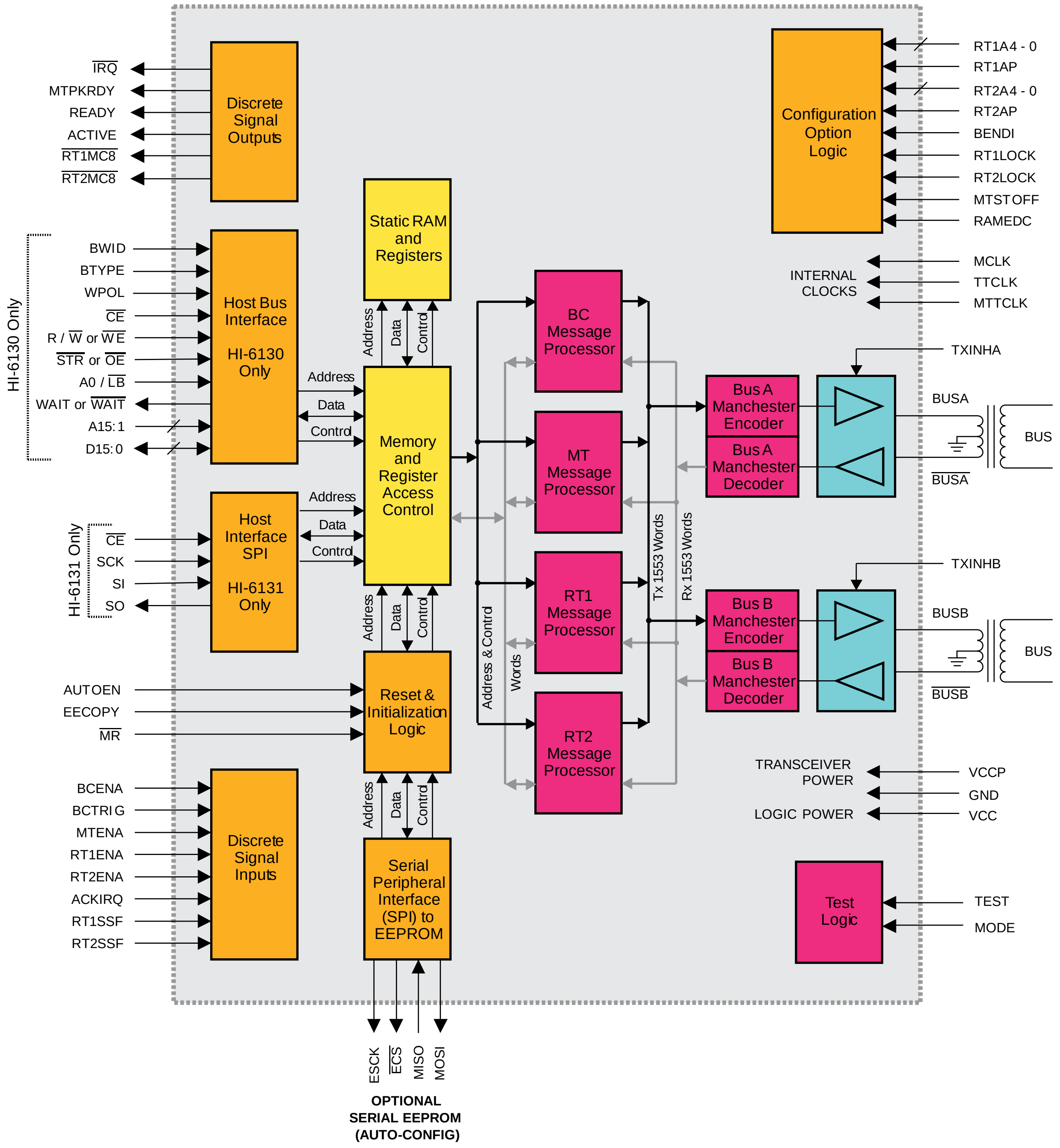
A large portion of the CPLD is unused with sufficient room for custom expansion. A Lattice USB programming cable PN: PN-USBN-2A is required to reprogram the CPLD but is NOT provided by Holt. This is only needed if the end user wishes to alter the Verilog code and reprogram the CPLD. Using the Lattice Diamond CPLD development software is beyond the scope of this document but many tutorials are built into Lattice Diamond software which is available for download from their website. When the board powers up, only LED10 is On. This is a convenient way to determine if the CPLD has been programmed.

The next page shows a block diagram of the CPLD

CPLD Functional Block Diagram



HI-6130 Functional Block Diagram



Software

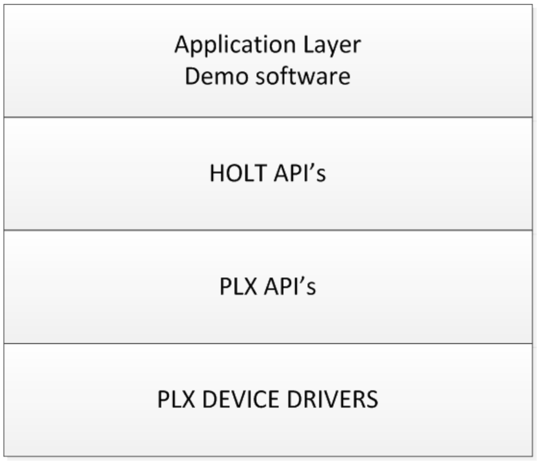
Two Holt demo programs are provided on the CD-ROM. PCIe6130 Test contains a menu of commands to demonstrate reading and writing to the latched I/O's, read and write to HI-6130 memory space and initialize the HI-6130 for BC transmissions. The simple BC demo periodically transmits 3 types of messages. This program was demonstrated in the Quick Start Guide section.

The HI-6130 API demo is more complex. The main purpose of this demo is to demonstrate Holt's API library. This demo program demonstrates how to use Holt's API library to initialize the HI-6130 for BC, RT or MT operations. Both demo projects are built using Microsoft Visual Studio 2012. For other compilers, the user needs to port the software project. All the API's and low-level drivers supporting the HI-6130 and GPIOs are contained in module HI6130.c with accompanying header file HI6130.h.

To rebuild these projects the following three items are needed:

- Holt demo projects contained on the CD-ROM.
- Microsoft Visual Studio 2012. (Not Provided)
- PLX SDK 7.20. (On broadcom.com website).

The 4 layer demo software is shown below.



PLX SDK installation

To modify the demo programs the PLX SDK must first be installed so that the Holt projects can be added to the SDK samples folder. To develop custom software applications for this card, the SDK from PLX is required. Download the SDK from the link below. PLX requires user registration to download their SDK by filling out their online registration and obtaining a login and password. Holt is not authorized to provide the PLX SDK package directly to customers. At the time this document was written, Windows 7 is supported and future OS versions including Linux are planned. After the SDK is installed, the PC may require a few moments or a reboot for the new drivers to take effect which are also installed by the SDK installation.

https://www.broadcom.com/products/pcie-switches-bridges/software-dev-kit
(If hyper-link does not work go directly to Broadcom.com)

The SDK installs a “Plx” folder on the root drive with several sub folders. The “PlxSdkUserManual” is located in the C:\Plx\PlxSdk\Documentation folder. This is an important document for information on installation, drivers, utilities and PLX API’s for the PEX8311 PCIe interface.

Follow these steps to install the demo project into the PLX SDK Samples folder:

Install the PLX SDK.

Locate the zipped Holt demo project “PCIe 6130 Test.zip” on the CD-ROM and unzip this project folder into the Samples folder shown.

```
C:\Plx\PlxSdk\Samples\_PCIe 6130 Test
```

Use this directory structure because some PLX files reference other files in these directories. The Holt API project details are provided later in this document.

Launch Microsoft *Visual Studio* and open the PCIe6130Test project using Open Project from the File menu, or from the Open Project short cut that may appear on the Start Page. Alternately, double-click on the PCIe6130Test project file in the PCIe6130 Test project folder.

1. The Solution Explorer with the source files is shown on the left side. If this is not seen, then open the Solution Explorer from the View menu at the top.

2. To run the program with the *Visual Studio* debugger, verify that the “Debug” configuration is selected at the top. Build the solution from the Build menu using Build Solution or just press F7. There should be no errors produced in the Output window.

3. Press the green arrow labeled “Local Windows Debugger” to run the program. The console output should be displayed like the one shown on page 4 in the Quick Start Guide.

4. To build an executable version of the demo code, select the “Release” configuration instead of “Debug” and rebuild the project. The executable file is put into the “Release” sub folder of the project folder. Use the “Debug” configuration for software development so that source level debugging features are operational.

   Note: When the SDK is installed a driver is installed for the PEX8311.

To understand the software and operation of the card for any development work or modifications, it is important to read the following documents.

- PLX SDK Users Manual which is installed when the SDK is installed.
- PLX PEX8311 RDK Hardware Reference Manual and the PLX PEX8311 data book. Latest versions are available from the broadcom.com website. These are not included in the SDK.
- Holt HI-6130 data sheet. Provided on the CD-ROM.
- Holt high-level API software users manual. Provided on the CD-ROM.

Other useful application notes from Holt:

- AN-6130_x.pdf
- AN-6130DG_x.pdf
- AN-550.pdf (for IC capacitor decoupling and transformer PCB routing)

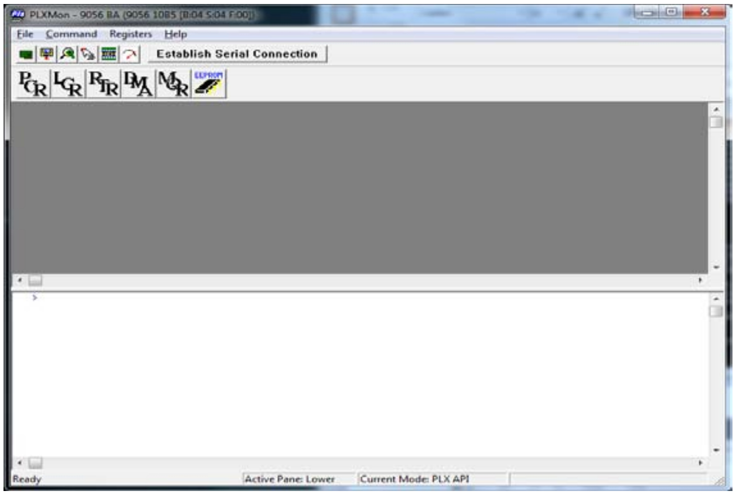
PLX API’s

PLX API’s use a unique input parameter (BAR) to specify which LB memory space to read and write from. BAR stands for Base Address Register. BAR 0 and BAR 1 are reserved for the upstream PCIe. BAR 2 and BAR 3 are used to provide separate memory spaces on the LB side. Bar 2 uses Space 0 and BAR 3 uses Space 1. The Space 0 and Space 1 PEX8311 registers are loaded with the desired starting BAR addresses at power up from the contents of the U6 EEPROM. Space 0 is reserved exclusively for the HI-6130 64KB memory space and is set to starting address 0x2000-0000. Space 1 is shared and modified on-the-fly in the API function calls to read/write to the output latches and reading the input status buffers. The latches are used to set the state of various inputs to the HI-6130 and the inputs are used to read HI-6130 status and DIP switches. The Holt demo code includes several functions to read and write to the HI-6130 memory space, latches and input status buffers. The API functions are located in HI6130.c. These functions use a PLX API to access the LB with either “PlxPci_PciBarSpaceRead(…)” or “PlxPci_PciBarSpaceWrite(…). One of the input parameters to these API’s is “bOffsetAsLocalAddr” this parameter controls how the API uses the U32 offset address. If “OffsetAsLocalAddr” is FALSE the input address is an offset address from the Space x value. If this parameter is TRUE the input address is the full address. See the PLX API user’s manual for descriptions and usage of the API’s, BAR spaces and other input parameters associated with the PEX8311 and SDK API’s.

Device	BAR	Space	Address	Notes
HI-6130	2	0	0x2000-0000	32K word range
Output Latches	3	1	0x2400-0000	First location used
Inputs	3	1	0x2800-0000	First location used
Output Latches-2	3	1	0x2C00-0000	First location used

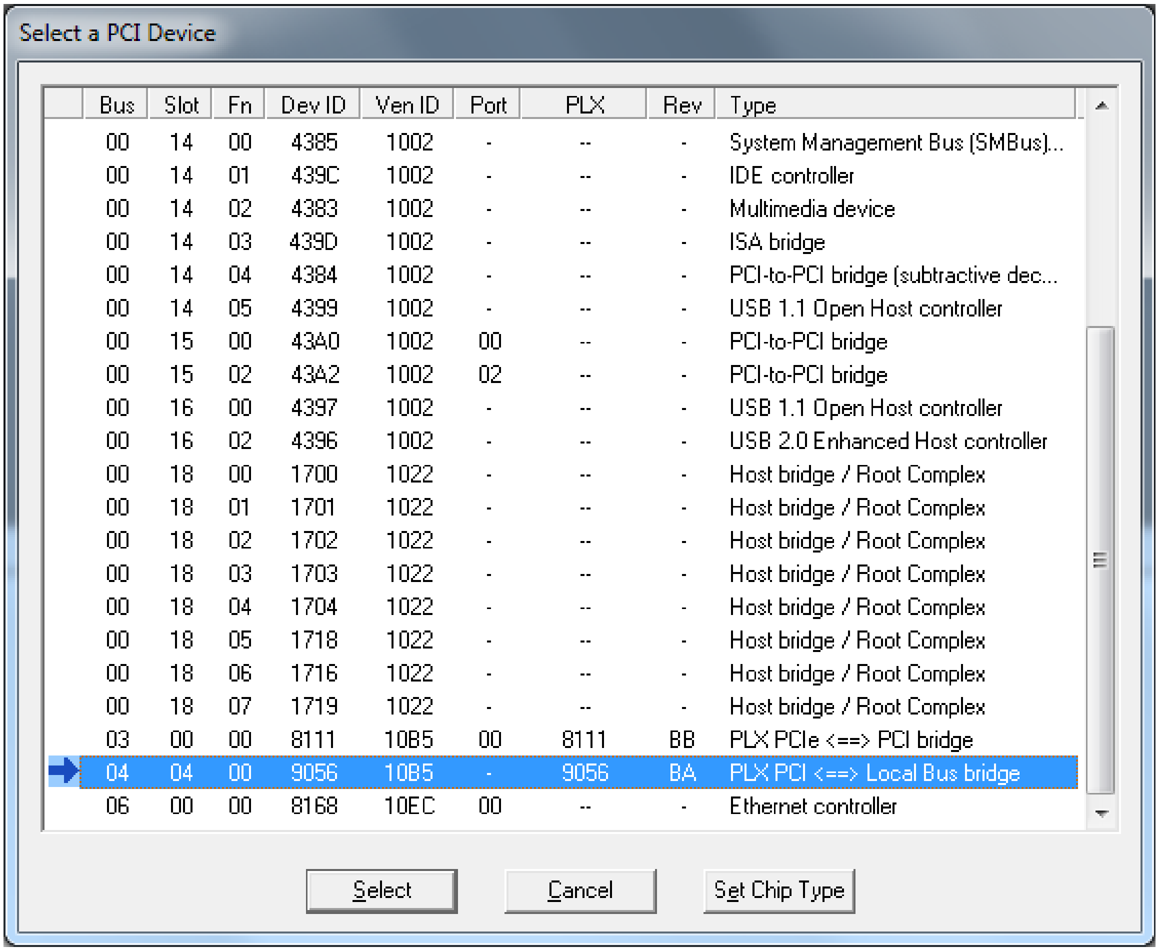
The U6 EEPROM contains configuration data the PEX8311 uses for LB configuration at power up. The LB is configured for a 16-bit data bus, 14 wait states, Space0 and Space1 starting addresses/ranges, and disables the TA/Ready input. The PCI PLX sub-ID 3566 is also programmed into this EEPROM. This sub-ID is assigned by PLX exclusively for the Holt demo board.

The PlxMon utility is launched from the Windows start menu or by double-clicking on the application found in the /Plx/…/Bin folder. The PlxMon utility opening screen is shown below:



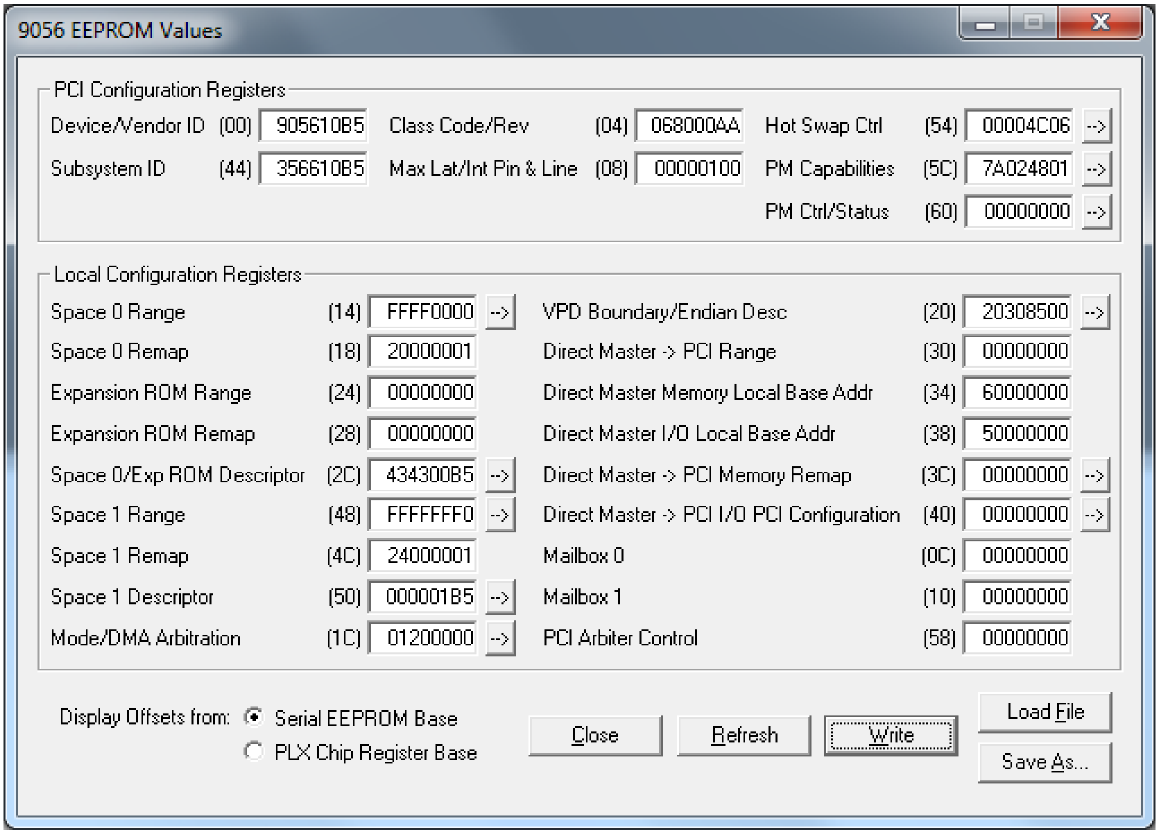
Special Note: Broadcom now owns PLX but the SDK version 7.2 still uses the “PLX” in the folder names.

Select the Holt PCIe card from the Command menu (or press the green icon button on the left) and select the device with Dev ID = 9056 and Ven ID = 10B5. The PEX8311 consists internally of a PEX8111 and a PCI9056.



	Bus	Slot	Fn	Dev ID	Ven ID	Port	PLX	Rev	Type
	00	14	00	4385	1002	-	--	-	System Management Bus (SMBus)...
	00	14	01	439C	1002	-	--	-	IDE controller
	00	14	02	4383	1002	-	--	-	Multimedia device
	00	14	03	439D	1002	-	--	-	ISA bridge
	00	14	04	4384	1002	-	--	-	PCI-to-PCI bridge (subtractive dec...
	00	14	05	4399	1002	-	--	-	USB 1.1 Open Host controller
	00	15	00	43A0	1002	00	--	-	PCI-to-PCI bridge
	00	15	02	43A2	1002	02	--	-	PCI-to-PCI bridge
	00	16	00	4397	1002	-	--	-	USB 1.1 Open Host controller
	00	16	02	4396	1002	-	--	-	USB 2.0 Enhanced Host controller
	00	18	00	1700	1022	-	--	-	Host bridge / Root Complex
	00	18	01	1701	1022	-	--	-	Host bridge / Root Complex
	00	18	02	1702	1022	-	--	-	Host bridge / Root Complex
	00	18	03	1703	1022	-	--	-	Host bridge / Root Complex
	00	18	04	1704	1022	-	--	-	Host bridge / Root Complex
	00	18	05	1718	1022	-	--	-	Host bridge / Root Complex
	00	18	06	1716	1022	-	--	-	Host bridge / Root Complex
	00	18	07	1719	1022	-	--	-	Host bridge / Root Complex
	03	00	00	8111	10B5	00	8111	BB	PLX PCIe <==> PCI bridge
➡	04	04	00	9056	10B5	-	9056	BA	PLX PCI <==> Local Bus bridge
	06	00	00	8168	10EC	00	--	-	Ethernet controller

After selecting the Holt card, select the LCR button (top left) to see the Main Control Registers. Changes to the LB configuration can be dynamically changed by entering new values from this menu. Alternately, press the EEPROM button to see the 9056 EEPROM values shown below. Notice the Vendor ID is 10B5. This is PLX’s vendor ID and must be used with the sub-ID 3566 assigned to Holt. A new end product would require a unique Vendor ID from PCI-SIG (obtained by becoming a PCI-SIG member) or a sub-ID obtained from PLX. For detailed information on these parameters and the PLX API’s refer to the PLX SDK user’s guide and data sheet on the PEX8311. For the Holt card, there is no need to alter any of these values.



PCI Configuration Registers

Device/Vendor ID	(00)	905610B5
Subsystem ID	(44)	356610B5
Class Code/Rev	(04)	068000AA
Max Lat/Int Pin & Line	(08)	00000100
Hot Swap Ctrl	(54)	00004C06
PM Capabilities	(5C)	7A024801
PM Ctrl/Status	(60)	00000000

Local Configuration Registers

Space 0 Range	(14)	FFFF0000
Space 0 Remap	(18)	20000001
Expansion ROM Range	(24)	00000000
Expansion ROM Remap	(28)	00000000
Space 0/Exp ROM Descriptor	(2C)	434300B5
Space 1 Range	(48)	FFFFFFF0
Space 1 Remap	(4C)	24000001
Space 1 Descriptor	(50)	000001B5
Mode/DMA Arbitration	(1C)	01200000
VPD Boundary/Endian Desc	(20)	20308500
Direct Master -> PCI Range	(30)	00000000
Direct Master Memory Local Base Addr	(34)	60000000
Direct Master I/O Local Base Addr	(38)	50000000
Direct Master -> PCI Memory Remap	(3C)	00000000
Direct Master -> PCI I/O PCI Configuration	(40)	00000000
Mailbox 0	(0C)	00000000
Mailbox 1	(10)	00000000
PCI Arbiter Control	(58)	00000000

Display Offsets from: ● Serial EEPROM Base ○ PLX Chip Register Base

On the bottom of the PlxMon screen is a console window where a command menu allows reading and writing to LB memory spaces. These are useful during initial hardware checkout to confirm LB configurations. Pressing “?” followed by a Return in this window lists a Help page of the commands. The commands are also documented in the PLX API user manual and some examples are provided in the PLX PEX8311 RDK Hardware Reference Manual.

Holt HI-6130 PCIe API Demo using Holt's high-level API library.

The HI-6130 PCIe API demo program demonstrates Holt's high-level API software library. In this version of the API demo the API library files are contained in a separate folder "_HI-613x API LIB" and must be unzipped and installed in the same directory folder where the "_HI-6130 PCIe API Demo" project folder resides. Copy both zipped project folders from the CD-ROM to the desktop and unzip them before copying both project folders to the PLX SDK samples folder:

```
C:\Plx\PlxSdk\Samples\_HI-613x API LIB
C:\Plx\PlxSdk\Samples\_HI-6130 PCIe API Demo
```

The Holt API library file "apiLib.lib" is prebuilt and inside _HI-613x API LIB/lib folder. This project cannot be rebuilt using *Visual Studio* because certain C source files are not included in this version. Contact Holt sales for a software license agreement (SLA).

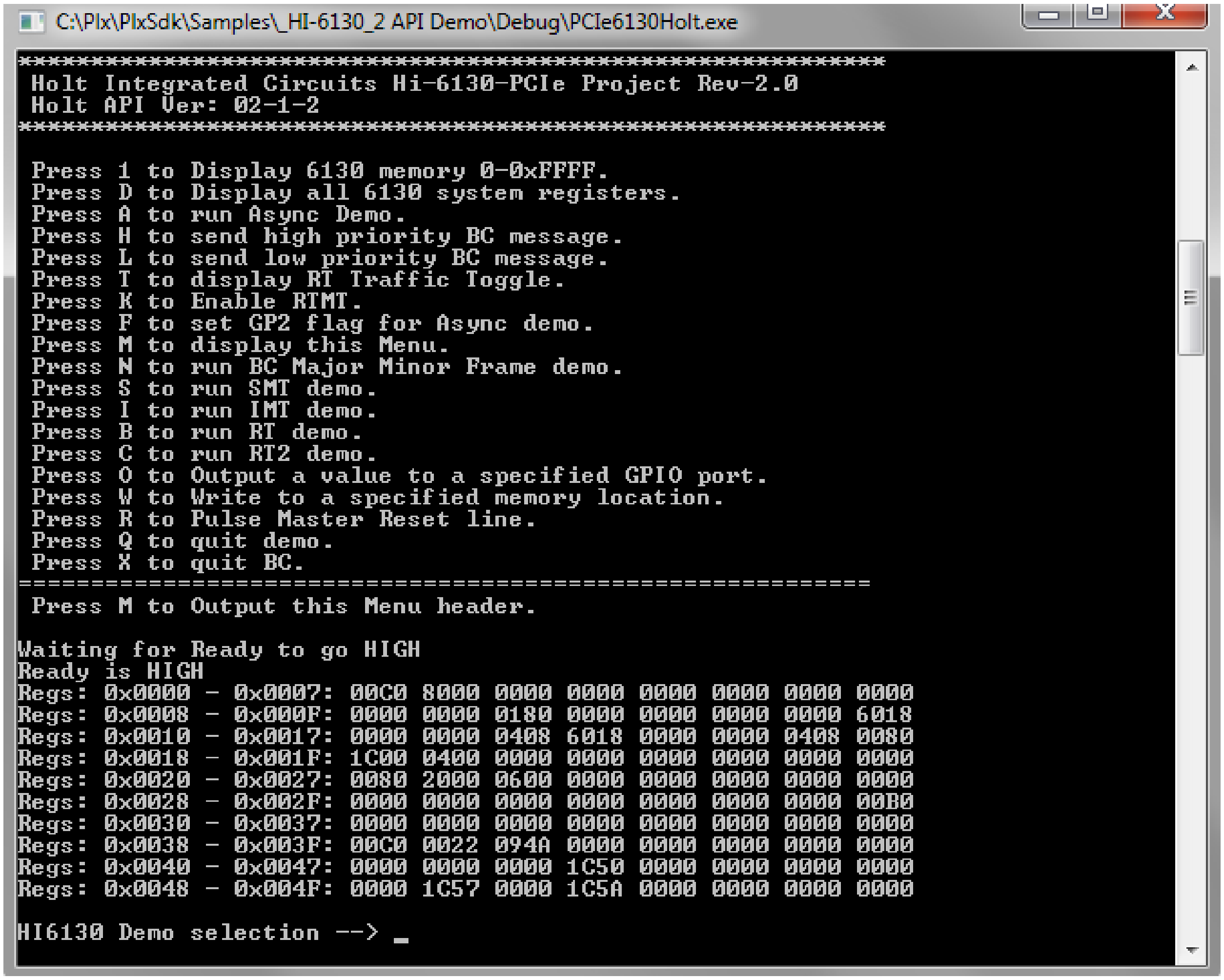
The demo consists of menu commands prompted on the console for demonstrating the BC, RT1, RT2, SMT and IMT.

Many of the PLX API examples used in Dslave and LocalToPciInt were used in the Holt projects. It's a good idea to review these two projects when first becoming familiar with the PLX API library. These PLX projects do not run on the Holt PCIe card because the LB memory spaces are defined differently.

Launch the program by either double-clicking the executable application "PCIeHolt" contained in the *Visual Studio* project "Release" sub folder or launch Visual Studio and open the project to run the program in debug mode. To run the project in debug mode make sure the debug configuration is selected at the top. Sometimes it may be necessary to perform a project clean by selecting "Clean PCIe6130Holt" from the BUILD pull down menu. After rebuilding the project, some warnings may appear in the Output window which can be ignored but there should be no critical errors preventing the debugger from running the project.

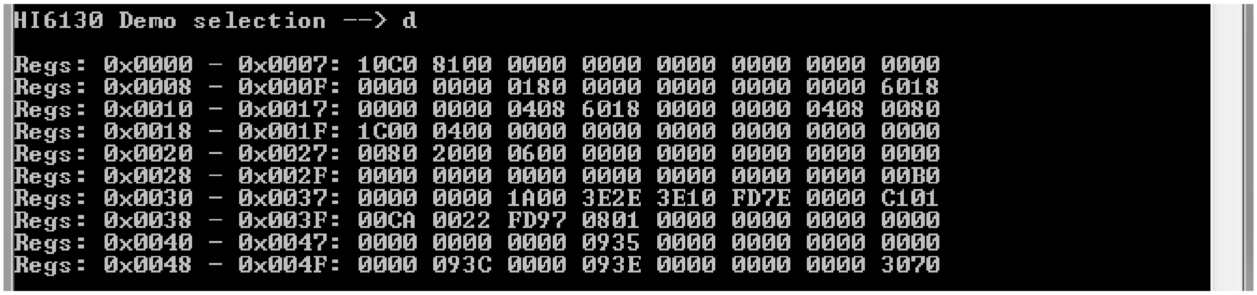
A menu will be displayed showing sets of numbers 1 though x. The Holt card typically appears as the first item, #1 with "9056 10b5" [b:xx s:xx f:xx]. Enter "1" and press Enter.

The console main menu shown below is displayed. Program revisions shown below might be different.


C:\Plx\PlxSdk\Samples_HI-6130_2 API Demo\Debug\PCIe6130Holt.exe

```
*************************************************************
 Holt Integrated Circuits Hi-6130-PCIe Project Rev-2.0
 Holt API Ver: 02-1-2
*************************************************************

 Press 1 to Display 6130 memory 0-0xFFFF.
 Press D to Display all 6130 system registers.
 Press A to run Async Demo.
 Press H to send high priority BC message.
 Press L to send low priority BC message.
 Press T to display RT Traffic Toggle.
 Press K to Enable RTMT.
 Press F to set GP2 flag for Async demo.
 Press M to display this Menu.
 Press N to run BC Major Minor Frame demo.
 Press S to run SMT demo.
 Press I to run IMT demo.
 Press B to run RT demo.
 Press C to run RT2 demo.
 Press O to Output a value to a specified GPIO port.
 Press W to Write to a specified memory location.
 Press R to Pulse Master Reset line.
 Press Q to quit demo.
 Press X to quit BC.
============================================================
 Press M to Output this Menu header.

Waiting for Ready to go HIGH
Ready is HIGH
Regs: 0x0000 - 0x0007: 00C0 8000 0000 0000 0000 0000 0000 0000
Regs: 0x0008 - 0x000F: 0000 0000 0180 0000 0000 0000 0000 6018
Regs: 0x0010 - 0x0017: 0000 0000 0408 6018 0000 0000 0408 0080
Regs: 0x0018 - 0x001F: 1C00 0400 0000 0000 0000 0000 0000 0000
Regs: 0x0020 - 0x0027: 0080 2000 0600 0000 0000 0000 0000 0000
Regs: 0x0028 - 0x002F: 0000 0000 0000 0000 0000 0000 0000 00B0
Regs: 0x0030 - 0x0037: 0000 0000 0000 0000 0000 0000 0000 0000
Regs: 0x0038 - 0x003F: 00C0 0022 094A 0000 0000 0000 0000 0000
Regs: 0x0040 - 0x0047: 0000 0000 0000 1C50 0000 0000 0000 0000
Regs: 0x0048 - 0x004F: 0000 1C57 0000 1C5A 0000 0000 0000 0000

HI6130 Demo selection --> _
```

Press D to display the HI-6130 system registers with labels followed by the same registers values formatted by beginning and end addressed rows followed by eight register values.



```
HI6130 Demo selection --> d

Regs: 0x0000 - 0x0007: 10C0 8100 0000 0000 0000 0000 0000 0000
Regs: 0x0008 - 0x000F: 0000 0000 0180 0000 0000 0000 0000 6018
Regs: 0x0010 - 0x0017: 0000 0000 0408 6018 0000 0000 0408 0080
Regs: 0x0018 - 0x001F: 1C00 0400 0000 0000 0000 0000 0000 0000
Regs: 0x0020 - 0x0027: 0080 2000 0600 0000 0000 0000 0000 0000
Regs: 0x0028 - 0x002F: 0000 0000 0000 0000 0000 0000 0000 00B0
Regs: 0x0030 - 0x0037: 0000 0000 1A00 3E2E 3E10 FD7E 0000 C101
Regs: 0x0038 - 0x003F: 00CA 0022 FD97 0801 0000 0000 0000 0000
Regs: 0x0040 - 0x0047: 0000 0000 0000 0935 0000 0000 0000 0000
Regs: 0x0048 - 0x004F: 0000 093C 0000 093E 0000 0000 0000 3070
```

Register 0 (0000) is the Master Control register and Register 1 is the Master Status and Reset Register. Notice the MSB is high (8) in word address 0x0001, this is the READY bit indicating the HI-6130 is ready for host access. Widen the Tera-Term window enough to view the full 16 registers across for best viewing.

General structure of demo functions

The Holt API demonstration program is contained in module "demos.c". The API libraries are contained in the library file " apiLib.lib " as executable object code. "Demos.c" contains the demo initialization API function calls supporting demonstrations executed from the console menu to initialize the BC, RT(s) and monitor terminals. Key presses are detected in "console.c" which is called from the main loop in "main.c" and executes demo functions in "demos.c".

Commands ‘A’ and ‘N’ transmit BC commands and can be viewed on an oscilloscope and optionally display decoded message traffic data on the console using the ‘K’ and ‘T’ command sequence. These demos demonstrate how the Holt API library is used to generate BC Asynchronous messages, Major/Minor frames, low priority and high priority asynchronous (unscheduled) messages. View these messages with external MIL-STD-1553 test equipment or view them with an oscilloscope.

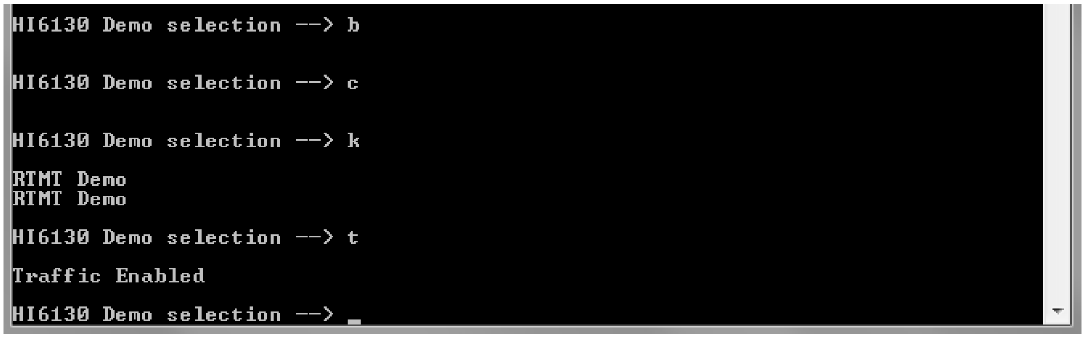
This exercise uses the internal BC to transmit messages, so message traffic data is displayed on the console. Since the internal BC, RTs and SMT share the same bus pins, the RT and SMT monitor terminals receive the BC messages. If an external BC is already connected to the 9-pin to MIL-STD 1553 Tri-Axial cable though a bus coupler, it is okay to leave it connected, but disable any external BC transmissions that will conflict with the on-chip BC transmissions. Without an external BC connected, connect an external 75Ω 1/2 Watt resistor across the BUSA and nBUSA test points to terminate the bus. These are red and black test points adjacent to the 9-pin D-Sub connector and transformers.

Internal BC demo with RT Traffic on console

1. Press command ‘B’ and ‘C’ to enable both RTs.

2. Press command ‘K’ to enable the RTMT demo.

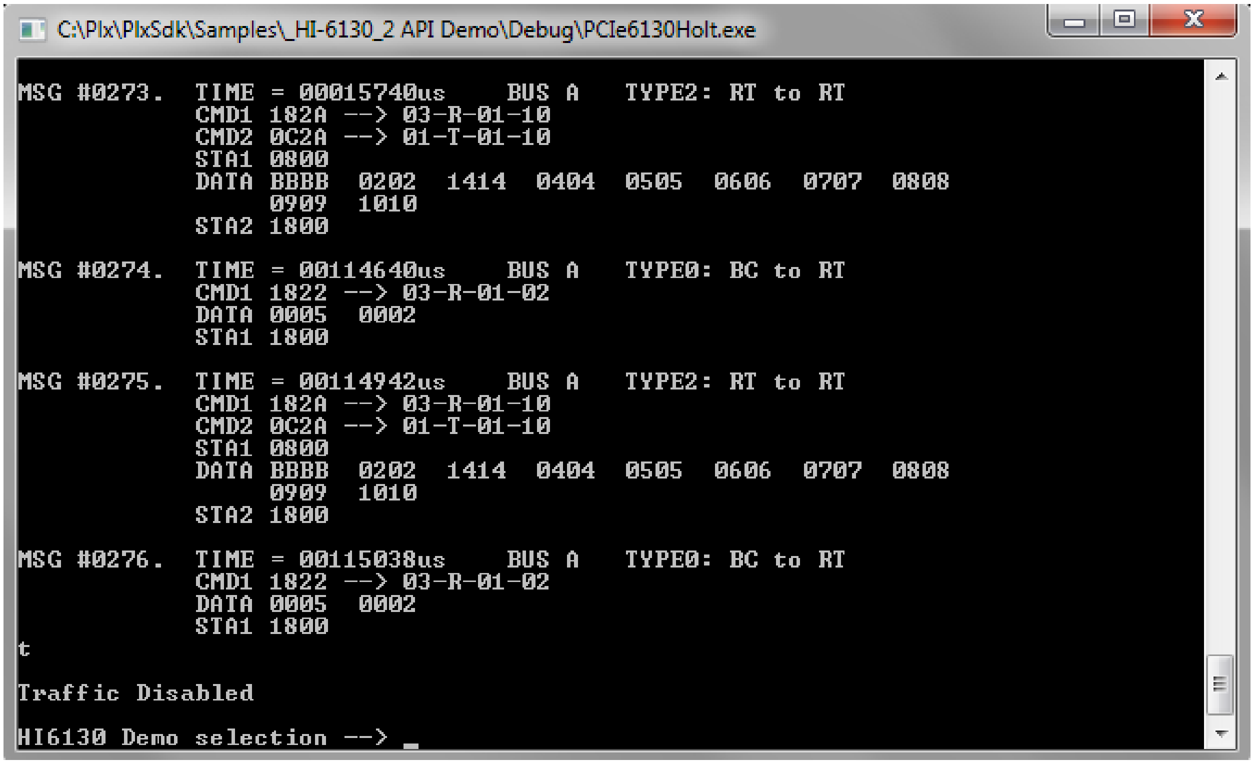
3. Press command ‘T’ (or spacebar) to display RT traffic on the console. Command ‘T’ toggles on and off alternately to enable or disable the RT traffic shown on the console. Using the ‘T’ command relies on prior execution of command ‘K’.



```
HI6130 Demo selection --> b

HI6130 Demo selection --> c

HI6130 Demo selection --> k

RTMT Demo
RTMT Demo

HI6130 Demo selection --> t

Traffic Enabled

HI6130 Demo selection --> _
```

4. Press command ‘A’ to start the BC transmitting messages.

Messages will display rapidly on the screen, Press 'T' to stop the console output. The console should freeze and look similar to the screen below. Press 'T' again to restart the console output. Using the 'T' command does not stop BC transmission or prevent RT or MT message reception; it only stops console output.

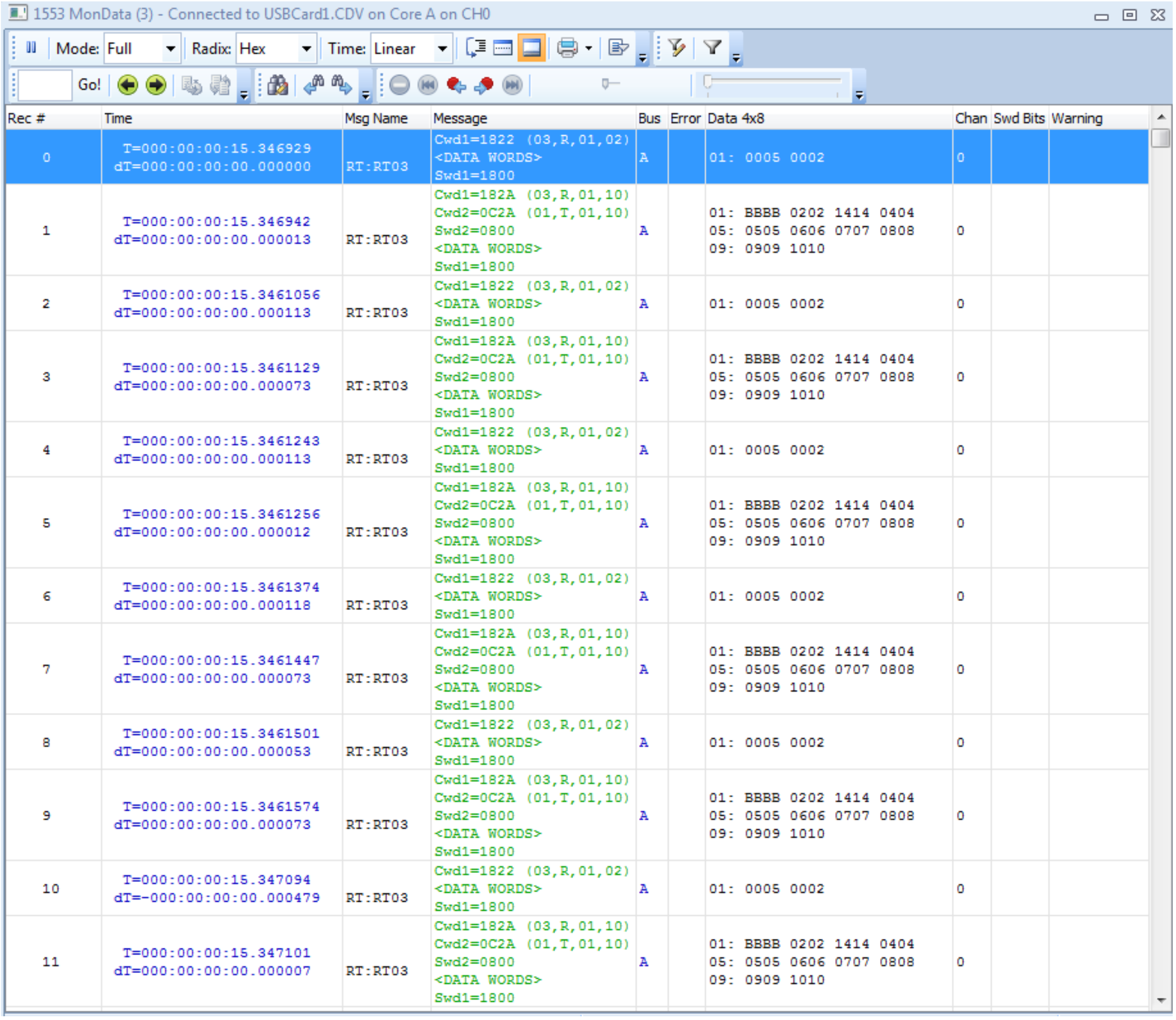


C:\Plx\PlxSdk\Samples_HI-6130_2 API Demo\Debug\PCIe6130Holt.exe

```
MSG #0273.   TIME = 00015740us    BUS A   TYPE2: RT to RT
             CMD1 182A --> 03-R-01-10
             CMD2 0C2A --> 01-T-01-10
             STA1 0800
             DATA BBBB  0202  1414  0404  0505  0606  0707  0808
                  0909  1010
             STA2 1800

MSG #0274.   TIME = 00114640us    BUS A   TYPE0: BC to RT
             CMD1 1822 --> 03-R-01-02
             DATA 0005  0002
             STA1 1800

MSG #0275.   TIME = 00114942us    BUS A   TYPE2: RT to RT
             CMD1 182A --> 03-R-01-10
             CMD2 0C2A --> 01-T-01-10
             STA1 0800
             DATA BBBB  0202  1414  0404  0505  0606  0707  0808
                  0909  1010
             STA2 1800

MSG #0276.   TIME = 00115038us    BUS A   TYPE0: BC to RT
             CMD1 1822 --> 03-R-01-02
             DATA 0005  0002
             STA1 1800
t

Traffic Disabled

HI6130 Demo selection --> _
```

The following screen shot (on next page) shows a partial listing of messages captured while running the BC Async demo. The transmissions shown on the next page were captured using a Ballard USB 1553 monitor.

5. From program launch, if the BC is started before enabling the 'K' and 'T' sequence to display message traffic, the first message may contain an error. This is normal; this occurs because the RT and MT are enabled midstream of a message in progress.



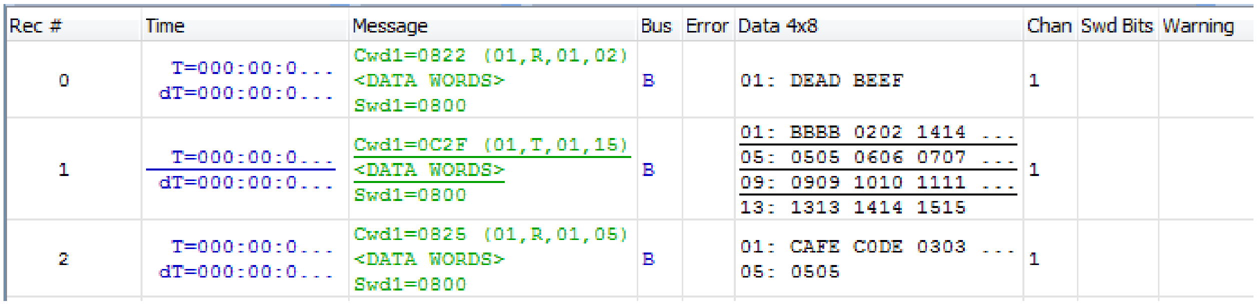
1553 MonData (3) - Connected to USBCard1.CDV on Core A on CH0

Mode: Full Radix: Hex Time: Linear

Rec #	Time	Msg Name	Message	Bus	Error	Data 4x8	Chan	Swd Bits	Warning
0	T=000:00:00:15.346929 dT=000:00:00:00.000000	RT:RT03	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: 0005 0002	0		
1	T=000:00:00:15.346942 dT=000:00:00:00.000013	RT:RT03	Cwd1=182A (03,R,01,10) Cwd2=0C2A (01,T,01,10) Swd2=0800 <DATA WORDS> Swd1=1800	A		01: BBBB 0202 1414 0404 05: 0505 0606 0707 0808 09: 0909 1010	0		
2	T=000:00:00:15.3461056 dT=000:00:00:00.000113	RT:RT03	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: 0005 0002	0		
3	T=000:00:00:15.3461129 dT=000:00:00:00.000073	RT:RT03	Cwd1=182A (03,R,01,10) Cwd2=0C2A (01,T,01,10) Swd2=0800 <DATA WORDS> Swd1=1800	A		01: BBBB 0202 1414 0404 05: 0505 0606 0707 0808 09: 0909 1010	0		
4	T=000:00:00:15.3461243 dT=000:00:00:00.000113	RT:RT03	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: 0005 0002	0		
5	T=000:00:00:15.3461256 dT=000:00:00:00.000012	RT:RT03	Cwd1=182A (03,R,01,10) Cwd2=0C2A (01,T,01,10) Swd2=0800 <DATA WORDS> Swd1=1800	A		01: BBBB 0202 1414 0404 05: 0505 0606 0707 0808 09: 0909 1010	0		
6	T=000:00:00:15.3461374 dT=000:00:00:00.000118	RT:RT03	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: 0005 0002	0		
7	T=000:00:00:15.3461447 dT=000:00:00:00.000073	RT:RT03	Cwd1=182A (03,R,01,10) Cwd2=0C2A (01,T,01,10) Swd2=0800 <DATA WORDS> Swd1=1800	A		01: BBBB 0202 1414 0404 05: 0505 0606 0707 0808 09: 0909 1010	0		
8	T=000:00:00:15.3461501 dT=000:00:00:00.000053	RT:RT03	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: 0005 0002	0		
9	T=000:00:00:15.3461574 dT=000:00:00:00.000073	RT:RT03	Cwd1=182A (03,R,01,10) Cwd2=0C2A (01,T,01,10) Swd2=0800 <DATA WORDS> Swd1=1800	A		01: BBBB 0202 1414 0404 05: 0505 0606 0707 0808 09: 0909 1010	0		
10	T=000:00:00:15.347094 dT=-000:00:00:00.000479	RT:RT03	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: 0005 0002	0		
11	T=000:00:00:15.347101 dT=000:00:00:00.000007	RT:RT03	Cwd1=182A (03,R,01,10) Cwd2=0C2A (01,T,01,10) Swd2=0800 <DATA WORDS> Swd1=1800	A		01: BBBB 0202 1414 0404 05: 0505 0606 0707 0808 09: 0909 1010	0		

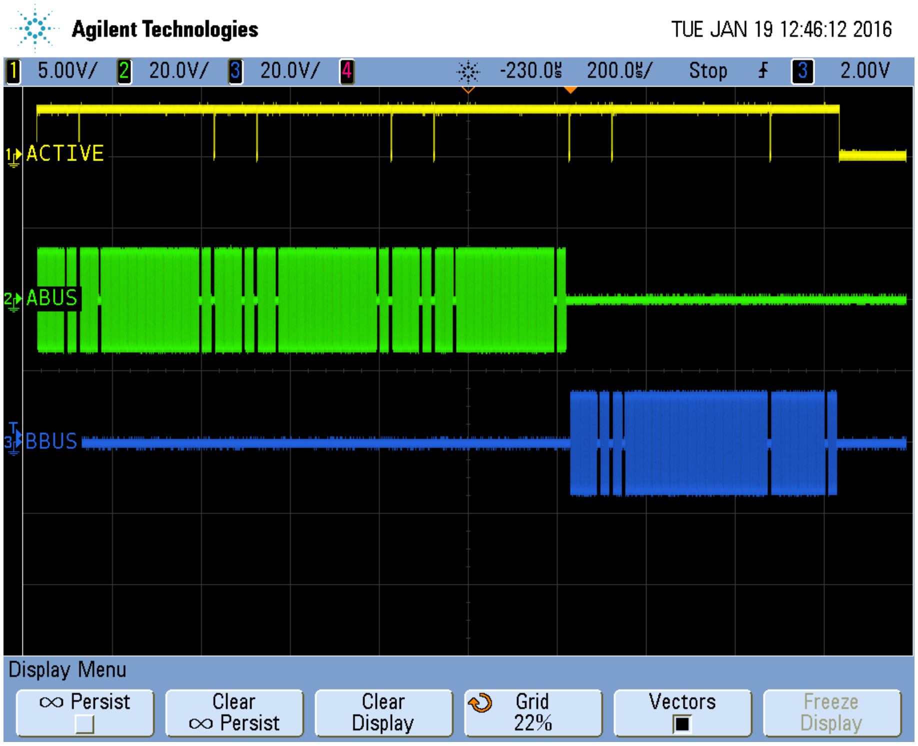
6. BC low priority asynchronous message insertion.

Command ‘L’ inserts a low priority message into the BC transmissions at end of frame. First, enable both RTs by pressing ‘B’ and ‘C’ then Press ‘A’ to enable the BC transmissions. Press ‘L’ to transmit three extra messages on Bus B. Bus B is used to make it easier to see on the scope. If the RTs were not enabled, retry messages would appear on bus B making it more difficult to see the three inserted messages. This will only work once after a power up or RESET.



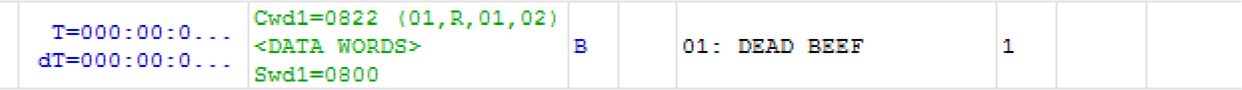
Rec #	Time	Message	Bus	Error	Data 4x8	Chan	Swd Bits	Warning
0	T=000:00:0... dT=000:00:0...	Cwd1=0822 (01,R,01,02) <DATA WORDS> Swd1=0800	B		01: DEAD BEEF	1		
1	T=000:00:0... dT=000:00:0...	Cwd1=0C2F (01,T,01,15) <DATA WORDS> Swd1=0800	B		01: BBBB 0202 1414 ... 05: 0505 0606 0707 ... 09: 0909 1010 1111 ... 13: 1313 1414 1515	1		
2	T=000:00:0... dT=000:00:0...	Cwd1=0825 (01,R,01,05) <DATA WORDS> Swd1=0800	B		01: CAFE CODE 0303 ... 05: 0505	1		

This shows BusA and BusB signals at the instant the low priority messages are sent in blue color.

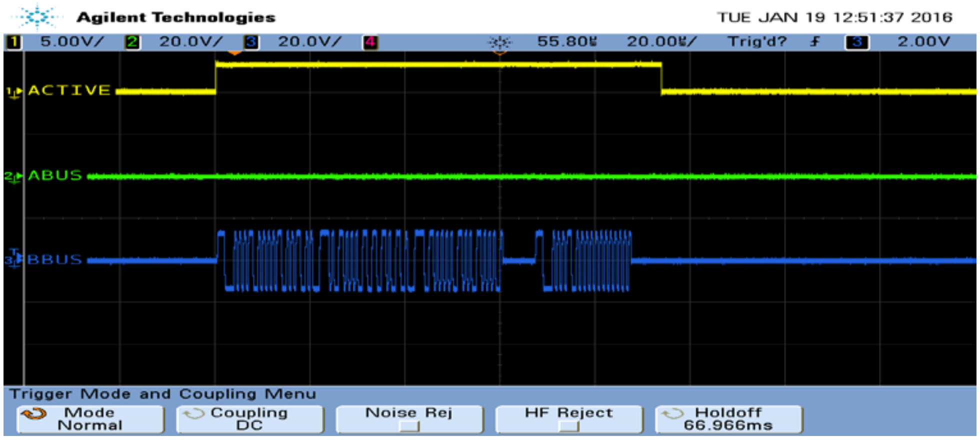


7. BC High priority asynchronous message insertion.

Follow the same steps as the previous BC low priority message example but this time Press ‘H’ to insert a single high priority after completion of the current scheduled message. This command is repeatable.



T=000:00:0... dT=000:00:0...	Cwd1=0822 (01,R,01,02) <DATA WORDS> Swd1=0800	B		01: DEAD BEEF	1		



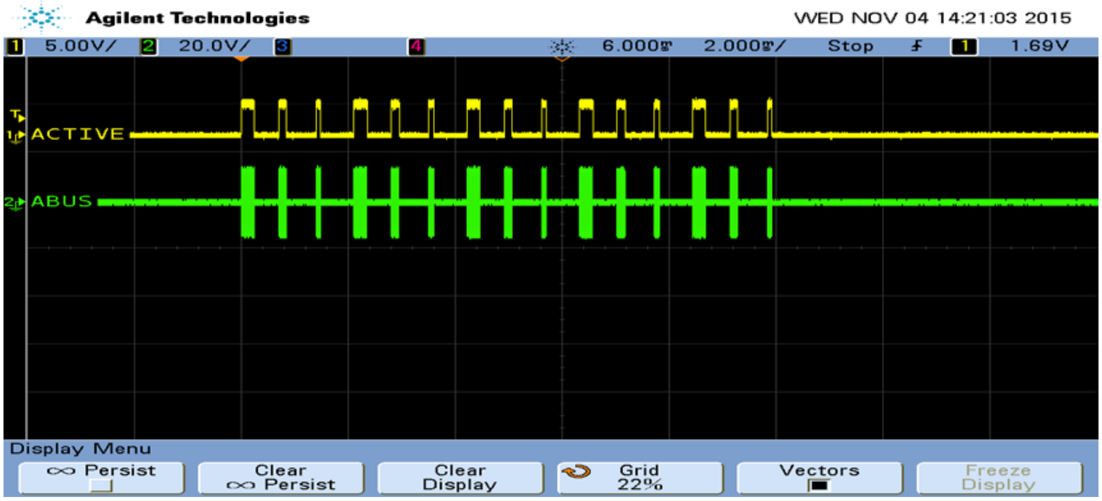
8. The ‘N’ command transmits fifteen commands to RT address 3. Press ‘B’ to enable the RT.

The RT naming might be confusing and needs further explanation: There are two RT terminals in the HI-6130 device. The two RT terminals have their own RT address DIP switches on the board to allow setting the RT address for each.

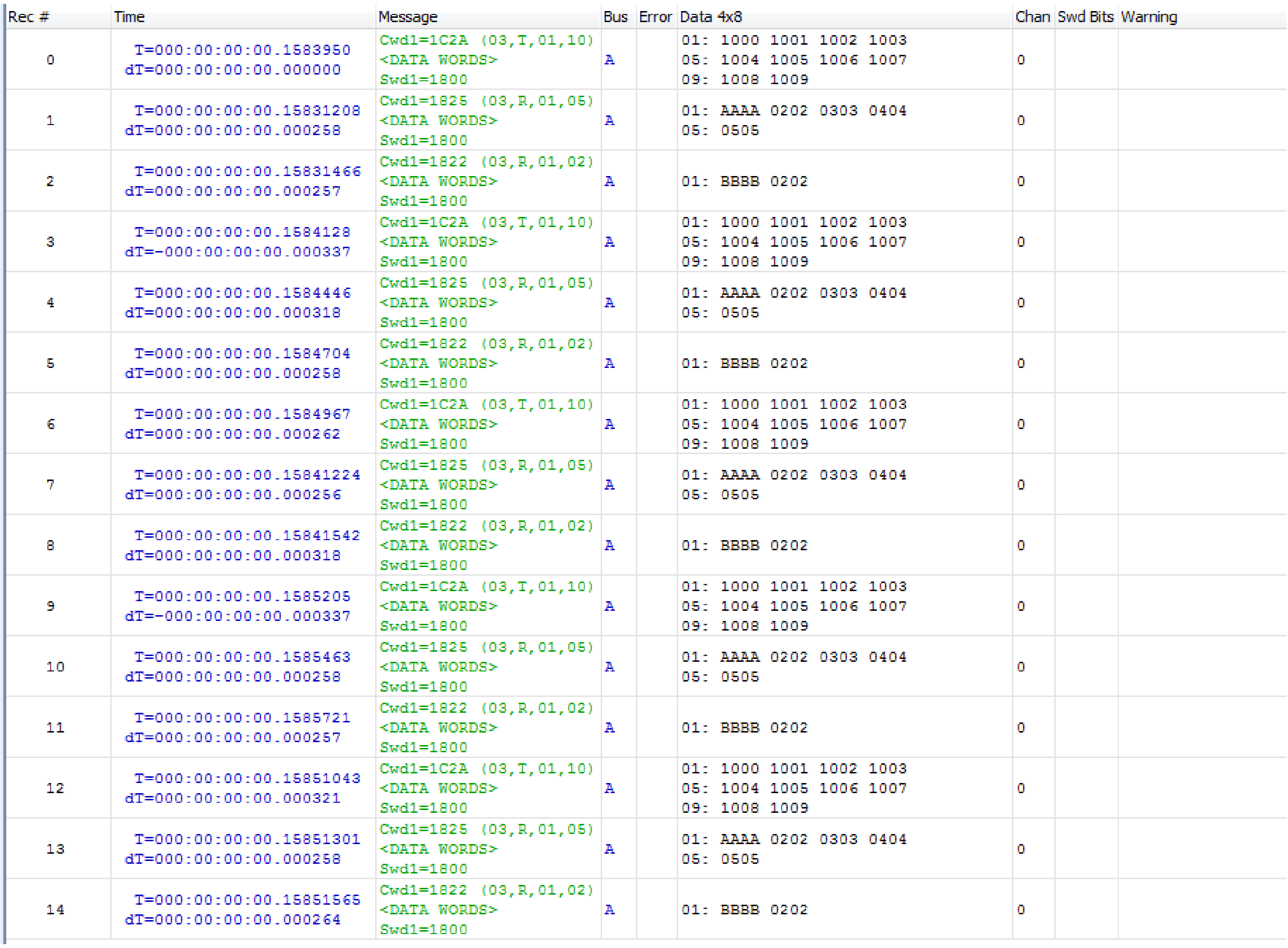
The first RT (referred to as RT or RT1) is set for RT address 3.

The second RT (RT2) is set for RT address 1.

Press ‘N’ to execute the BC transmissions (15 messages are transmitted) which will appear on the bus as shown below. To optionally see the message traffic on the console, enable the RT message traffic by pressing ‘T’ if it hasn’t already been enabled.



Command ‘N’ (15 message) Traffic capture using a Ballard USB 1553 monitor. Use the ‘T’ command to see the same messages on the console.



Rec #	Time	Message	Bus	Error	Data 4x8	Chan	Swd Bits	Warning
0	T=000:00:00:00.1583950 dT=000:00:00:00.000000	Cwd1=1C2A (03,T,01,10) <DATA WORDS> Swd1=1800	A		01: 1000 1001 1002 1003 05: 1004 1005 1006 1007 09: 1008 1009	0		
1	T=000:00:00:00.15831208 dT=000:00:00:00.000258	Cwd1=1825 (03,R,01,05) <DATA WORDS> Swd1=1800	A		01: AAAA 0202 0303 0404 05: 0505	0		
2	T=000:00:00:00.15831466 dT=000:00:00:00.000257	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: BBBB 0202	0		
3	T=000:00:00:00.1584128 dT=-000:00:00:00.000337	Cwd1=1C2A (03,T,01,10) <DATA WORDS> Swd1=1800	A		01: 1000 1001 1002 1003 05: 1004 1005 1006 1007 09: 1008 1009	0		
4	T=000:00:00:00.1584446 dT=000:00:00:00.000318	Cwd1=1825 (03,R,01,05) <DATA WORDS> Swd1=1800	A		01: AAAA 0202 0303 0404 05: 0505	0		
5	T=000:00:00:00.1584704 dT=000:00:00:00.000258	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: BBBB 0202	0		
6	T=000:00:00:00.1584967 dT=000:00:00:00.000262	Cwd1=1C2A (03,T,01,10) <DATA WORDS> Swd1=1800	A		01: 1000 1001 1002 1003 05: 1004 1005 1006 1007 09: 1008 1009	0		
7	T=000:00:00:00.15841224 dT=000:00:00:00.000256	Cwd1=1825 (03,R,01,05) <DATA WORDS> Swd1=1800	A		01: AAAA 0202 0303 0404 05: 0505	0		
8	T=000:00:00:00.15841542 dT=000:00:00:00.000318	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: BBBB 0202	0		
9	T=000:00:00:00.1585205 dT=-000:00:00:00.000337	Cwd1=1C2A (03,T,01,10) <DATA WORDS> Swd1=1800	A		01: 1000 1001 1002 1003 05: 1004 1005 1006 1007 09: 1008 1009	0		
10	T=000:00:00:00.1585463 dT=000:00:00:00.000258	Cwd1=1825 (03,R,01,05) <DATA WORDS> Swd1=1800	A		01: AAAA 0202 0303 0404 05: 0505	0		
11	T=000:00:00:00.1585721 dT=000:00:00:00.000257	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: BBBB 0202	0		
12	T=000:00:00:00.15851043 dT=000:00:00:00.000321	Cwd1=1C2A (03,T,01,10) <DATA WORDS> Swd1=1800	A		01: 1000 1001 1002 1003 05: 1004 1005 1006 1007 09: 1008 1009	0		
13	T=000:00:00:00.15851301 dT=000:00:00:00.000258	Cwd1=1825 (03,R,01,05) <DATA WORDS> Swd1=1800	A		01: AAAA 0202 0303 0404 05: 0505	0		
14	T=000:00:00:00.15851565 dT=000:00:00:00.000264	Cwd1=1822 (03,R,01,02) <DATA WORDS> Swd1=1800	A		01: BBBB 0202	0		

When a BC message is transmitted to an unassigned RT address, “RT no response” (NORES) error is indicated.

```
MSG #0170.  TIME = 00086918us    BUS A   TYPE2: RT to RT
            CMD1 182A --> 03-R-01-10
            CMD2 0C2A --> 01-T-01-10
 ERROR: NORES (not in red)
```

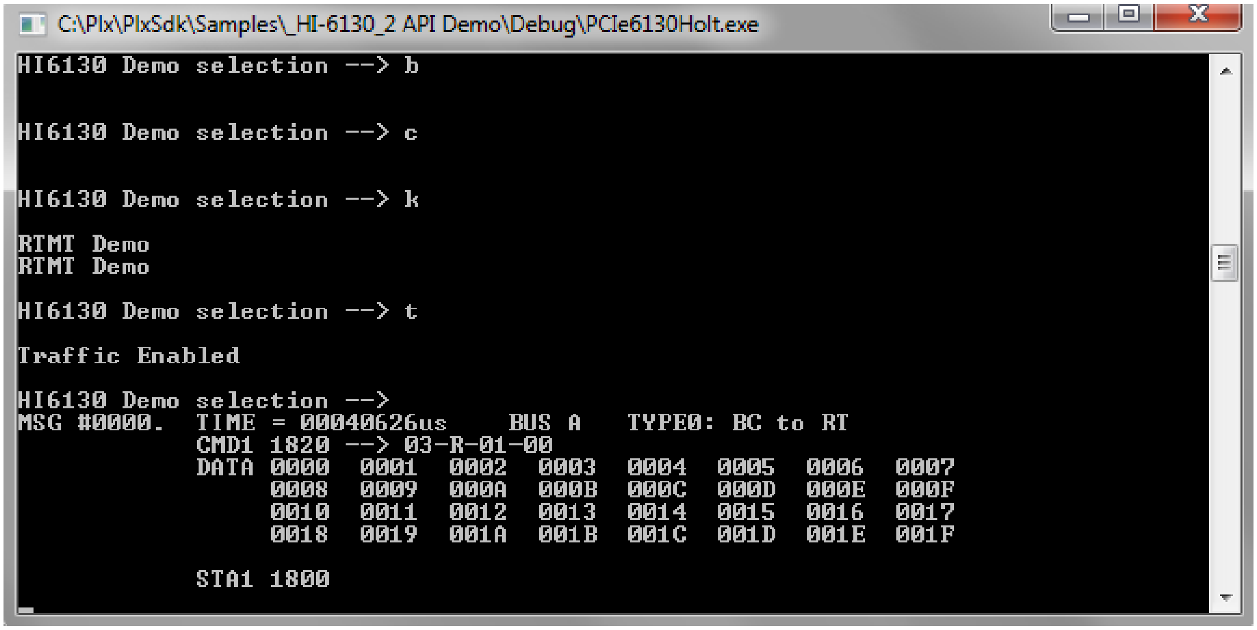
Commands ‘S’ and ‘I’

These demo commands initialize the SMT or IMT monitor features in the HI-6130 and return to the main menu, they do not display any information on the console. The SMT is also initialized and used in the ‘T’ RT Traffic command.

Using an external BC (such as Ballard tester) to transmit messages to the demo board.

When using a BC connected using a properly-terminated external bus to transmit commands to the RT or SMT, remove external 75 Ω resistors (if present) and connect the BC test equipment to the demo board circular tri-axial bus jacks using MIL-STD-1553 cables and bus couplers. If a bus coupler is not readily available, connect BC tester directly to the demo board cable tri-axial jack, but in this case use the 75 Ω termination resistor.

Restart the program and Press ‘B’ or ‘C” to enable both RTs then Press ‘K’ and ‘T’ to activate the RT traffic on the console. Compose a BC to RT message with SA=1 and 32 data words similar to the message shown below. The console should show the message transmitted by the BC, after the transaction.


C:\Plx\PlxSdk\Samples_HI-6130_2 API Demo\Debug\PCIe6130Holt.exe

```
HI6130 Demo selection --> b


HI6130 Demo selection --> c


HI6130 Demo selection --> k

RTMT Demo
RTMT Demo

HI6130 Demo selection --> t

Traffic Enabled

HI6130 Demo selection -->
MSG #0000.  TIME = 00040626us    BUS A   TYPE0: BC to RT
            CMD1 1820 --> 03-R-01-00
            DATA 0000  0001  0002  0003  0004  0005  0006  0007
                 0008  0009  000A  000B  000C  000D  000E  000F
                 0010  0011  0012  0013  0014  0015  0016  0017
                 0018  0019  001A  001B  001C  001D  001E  001F

            STA1 1800
```

When transmitting repeating messages at a high rate typical of MIL-STD 1553, the RT Traffic shown on the console may not keep pace due to limitations of the Windows PC OS. Depending on the message content and repetition rate, some messages may not show on the console. All messages are transacted properly and captured by enabled RT and MT, some messages simply will not be shown on the console.

Holt API function calls are fully described in the Holt API User’s Guide HI-6130-API_xx.pdf included in the ADK CD-ROM.

Appendix – A CPLD Verilog Source

A current copy of this source file is included in the Verilog source file on the CD-ROM.

```
// Holt PCI_6130 interface


module count_osc (rstn, osc_clk, LED, clk,   // clk and reset pins
        testpoint,                                   // test output
        add, decoderOutput,                          // address decoder
        lclk,blast,blast_q,ads,lwr,RDn,WRn,          // Read Write strobes
        LEDRD, LEDWR,                                // LED flashers
        Latch,                                       // 16 latched outputs
        DataBus,                                     // Data bus In/Out (16)
        InputBuffer,                                 // Inputs (16)
        nLINTi,                                      // Interrupt output pin
        Latch2,
        TP13
        );

// Misc. Signals
input   rstn ;
output  osc_clk ;
output  [7:0]LED ;
output  clk ;
output  testpoint ;                   // Test OE
output TP13;                          // Test ADSn

// Inputs
input lclk;                           // LCLK 50 MHZ input
input blast;                          // blast input
input [5:0]add;                       // inputs LA31-LA26 (6) for decoder
input [15:0]InputBuffer;              // 16 status input pins
input ads;                            // ADS input
input lwr;                            // LW/R input
//input RT1MC8n, RT2MC8n, IRQn;       // Interrupt inputs

// InOuts
inout [15:0]DataBus;                  // 16 In/out Data Bus pins

// Outputs
output blast_q;                       // blast output
output [4:0]decoderOutput;            // address decoder outputs
output reg RDn;                       // /RD output strobe
output reg WRn;                       // /WR output strobe
output reg LEDRD, LEDWR;              // LED flashers for 6130 indication
output reg [15:0]Latch;               // 16 latched outputs
output reg [15:0]Latch2;              // 2nd set of latches
output nLINTi;                        // Interrupt output to PLX

// Wires


// Constants
`define H6130                 8'b11111110    // HI6130 address decode address
`define LatchAddress  8'b11111101            // Latches output decode address
`define InputsAddress 8'b11111011            // Inputs address decode address
`define Latch2Address 8'b11110111            // Inputs address decode address

// Registers
reg            [26:0]c_delay ;
reg            [7:0] add_L;
reg            [0:0] blastq;
reg            [3:0] counter;
reg            [3:0] counter2;
reg     oe;                           // output enable for buffer reads
reg     [15:0] mux;                   // internal bus for muxing readback bus'
```

```
// Get IC clk and reset
GSR GSR_INST (.GSR(rstn));           // Reset occurs when argument is active low.
OSCC OSCC_1 (.OSC(osc_clk)) ;

// Logic


assign LED[3:0] = add_L;                            // copy decoder outputs to LEDs for test
assign LED[4] = blast_q;                            // copy of blast output
assign LED[5] = WRn;                                // copy WRn to output
assign LED[6] = RDn;                                // copy WRn to output
assign LED[7] = lclk;                               // copy clock to output
// End of test

assign decoderOutput = add_L;

assign TP13 = ads;


/* 3-8 Address Decoders */
always@(*)
       case (add)

               6'b001000 : add_L = `H6130;          // led1 CS20 - 6130
               6'b001001 : add_L = `LatchAddress;   // led2 CS24 - Output latches
               6'b001010 : add_L = `InputsAddress;  // led3 CS28 - Inputs
               6'b001011 : add_L = `Latch2Address;  // led4 CS2C - Latch2 output latches
               6'b001100 : add_L = 8'b11101111;     // led5 Spare CS
               default   : add_L = 8'b11111111;     // default all OFF
       endcase

/* 16 bit 3 to 1 multiplexer */
always@(*)
       begin
       case(add_L)
               `LatchAddress : mux = Latch;                 // First latches routed to mux
               `InputsAddress : mux = InputBuffer;          // Status inputs (6130) routed to mux
               `Latch2Address : mux = Latch2;               //Second set of latches routed to mux
               default   : mux = 16'hFFFF;                  // default
       endcase
       end


/* RD, RW, 6130CS Stobe generation */
// generate delayed blast signal used to clear RD, WR.
always @(posedge lclk)             begin
        if (blast)
           blastq <= 1'b1;           // set high during reset
        else
           blastq <= 1'b0;           // set high during reset
      end
assign  blast_q = blastq;            // leave as output pin for possible debugging later

// generate bracketed RDn stobe
always @(posedge lclk or negedge blastq)
     begin
               if(~blastq)
                      RDn <= 1'b1;          // set rd high
          else begin
                  if(~lwr && ~ads)
                      RDn <= 1'b0;          // set rd low
                  end

                        end

// generate bracketed WRn stobe
always @(posedge lclk or negedge blastq)
     begin
        if (~blastq)
                      WRn <= 1'b1 ;         // set wr high if blast_q =0
               else begin
```

```
                       if(lwr && ~ads)
                             WRn <= 1'b0;   // set wr low
                    end
      end


// -------------- 6130 RD & WR LED flashers ---------------
always @(posedge c_delay[18] or negedge RDn)

     begin
        if (~RDn)
                      begin
                             if(add_L == `H6130) // only if 6130
                                    begin
                                    LEDRD <= 1'b0 ;        // Turn on led
                                    counter = 6'b0;        // reset timer
                                    end
                      end
          else
                        if (counter==3)
                             LEDRD <= 1'b1;         // turn off led
                        else counter = counter +1;
      end


always @(posedge c_delay[18] or negedge WRn)
     begin
        if (~WRn)
                      begin
                             if(add_L == `H6130)   // only if 6130
                                    begin
                                    LEDWR <= 1'b0 ;        // Turn on led
                                    counter2 = 6'b0;       // reset timer
                                    end
                      end
          else
                        if (counter2==3)
                             LEDWR <= 1'b1; // turn off led
                        else counter2 = counter2 +1;
      end


/*  ----------- Latched 16 GPIO's -----------*/
// For Latch outputs
always @(posedge WRn or negedge rstn)
       begin
          if (~rstn)
               Latch <= 16'b0001000000110000; // Defaults: LED1 On (low), TXINHA, TXINHB off
          else if (add_L == `LatchAddress) // latches decode
                 Latch <= DataBus; //dbus;
       end

// For Latch2 outputs
always @(posedge WRn or negedge rstn)
       begin
          if (~rstn)
                 Latch2 <= 16'h002A; // Defaults: all low (00A5 for testing only)
          else if (add_L == `Latch2Address) // latches decode
                 Latch2 <= DataBus; //dbus;
       end

/* -------------  Read Input Buffers, or Latches 16 GPIO's ------------*/

always @(*)
       begin
               if(~RDn & (add_L==`InputsAddress))
                      oe=1'b1;       // turn on buffer
               else if(~RDn & (add_L==`LatchAddress))
                      oe=1'b1;       // turn on buffer
               else if(~RDn & (add_L==`Latch2Address))
                      oe=1'b1;       // turn on buffer
               else
                      oe=1'b0;       // turn off buffer
       end

assign testpoint = oe;               // leave as pin for debugging, later change to reg
//assign DataBus = oe ? InputBuffer : 16'hZ;        // Read the 16 inputs
assign DataBus = oe ? mux : 16'hZ;   // Read the 16 inputs


/* Misc Logic, Interrupt, MR  */

assign nLINTi =       InputBuffer[4] & InputBuffer[5] & InputBuffer[6];   // Interrupt pins

// End of Misc.


//  The c_delay counter is used to slow down the internal oscillator (OSC) output
//  to a rate of approximately 0.5 Hz
always @(posedge osc_clk or negedge rstn)
     begin
        if (~rstn)
           c_delay <= 32'h0000 ;
        else
           c_delay <= c_delay + 1;

      end

assign  clk = c_delay[18] ;

endmodule
```

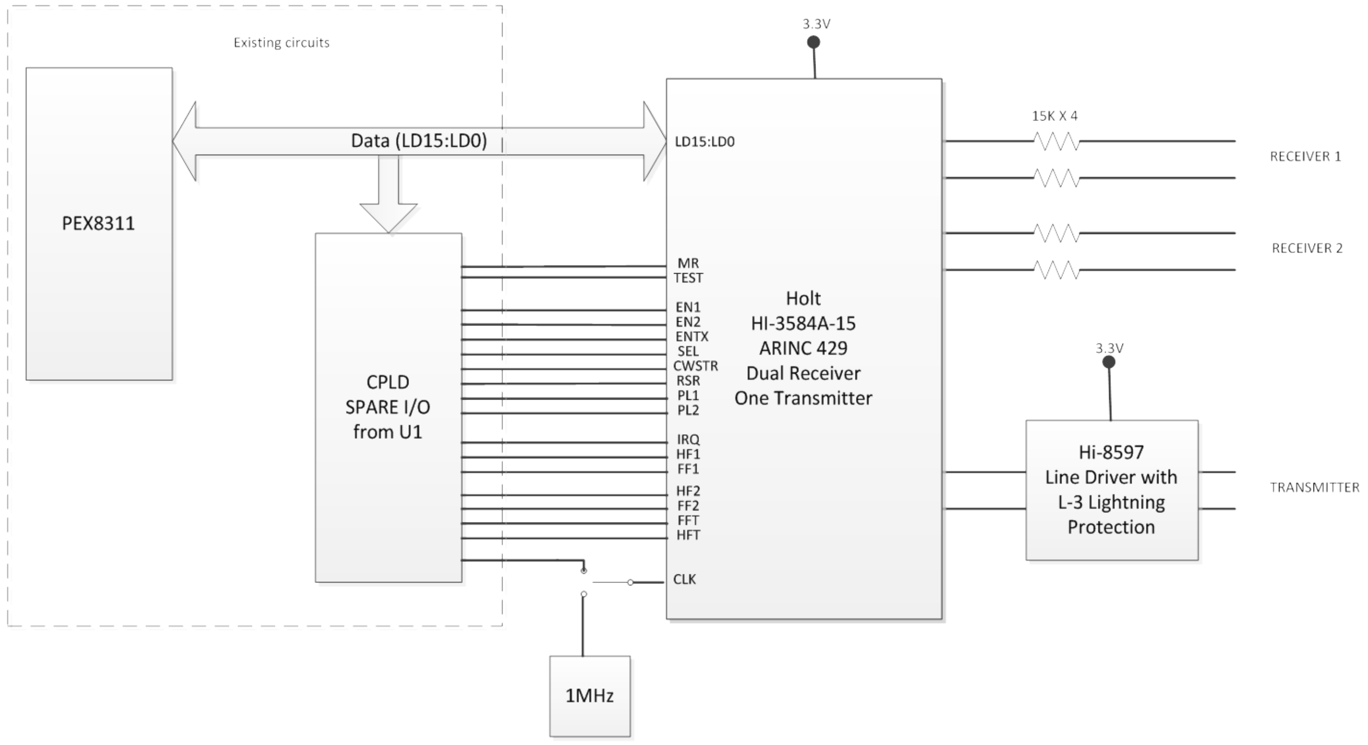
Customization

This section provides guidelines how to enhance the design. This would most likely require a new board design but limited prototyping could be achieved on a small add-on board, using the fourteen CPLD I/O's on J12 and J3 header connectors. PCIe12V, 3V3 voltages and ground connections are provided on these connectors. When adding circuits using these connections, power supply adequacy should be carefully reevaluated especially when using MIL-STD-1553.

ARINC 429

Holt has several ARINC 429 protocol receivers and transmitters that could be interfaced to the PCIe local bus (LB) and CPLD on this design. All the same hardware and software techniques apply. The devices that have a parallel interface would be the easiest to interface on the LB. Some suggested ARINC 429 16-bit parallel parts are the HI-3582A, HI-3583A and the HI-3584A. For a 3.3V only solution, use the HI-3584A with a 3.3V single supply rail line driver such as the HI-8596 or the HI-8597 (which includes built-in Level 3 lightning protection).

The following block diagram shows an interface for two ARINC 429 receivers and one transmitter powered by a single 3.3V supply meeting DO-160 Level 3 lightning protection. The 1MHz CLK input to the HI-3584A can be supplied directly by the CPLD or optionally by an external oscillator module. Refer to the HI-3584A, HI-8597 data sheets and AN-3582A application note for more information on these ARINC 429 devices. Contact Holt sales for other recommendations.



Discrete-to-Digital I/O Sensing

Discrete-to-Digital sensing capability can easily be added to this design using the Holt HI-8425, HI-8426, HI-8430 or HI-8431 parts. These also include either four high-side or low-side output drivers. The devices feature adjustable voltage thresholds and wetting currents on the inputs and over-current fault protection on the outputs. These devices use parallel I/O for input and outputs signals which can be driven by some of the spare CPLD pins. The HI-8425 and HI-8430 are the newest members of the Discrete-to-digital family, others are also available. Contact Holt sales for other recommendations.

EBR-1553 BC/MT/RT Terminals

The HI-6140 could replace the HI-6130 to achieve 10 Mbit/sec Extended Bit Rate MIL-STD-1553B terminal communication. The HI-6140 uses the same 16-bit parallel bus and 50MHz clock input as the HI-6130 and has a very similar register set and architecture. Holt HI-4853 slew-rate controlled RS-485 transceivers are used to drive the EBR bus. This change would require a revised board design, not an add-on board. See the HI-6140 product page on Holt's website for data sheets and application notes or contact Holt sales for recommendations.

MIL-STD-1553 Terminal Options

In addition to the HI-6130, the HI-6120 is a good candidate if only a single RT is required. The HI-6120 shares the same 16-bit parallel bus interface and 50MHz clock input as the HI-6130 with a similar pin-out. The RT operation of the HI-6120 is nearly the same as the HI-6130 but does have some differences in the registers and pin-outs. The HI-6120 is simpler to use and costs less than the HI-6130. This change would require a revised board design, not an add-on board.

Power supplies

PCIe computer add-in card slots have limitations which should be considered when designing a custom card. At the PC add-in card sockets, the PCI Express specification says the 3.3V supply can vary by ± 9% and the 12V supply can vary ± 8%. The HI-6130 is designed to work within a ± 5% voltage tolerance, so powering the device directly from the slot 3.3 volt supply will not meet MIL-STD-1553 performance characteristics during transmit; power supply regulation is inadequate and transient response will most likely not be suitable. Actual power rail characteristics will vary widely from computer to computer. MIL-STD-1553 devices like the HI-6130 and HI-6120 draw up to 0.9A during 1553 transmission. A two step power conversion is needed. The Holt card uses a 12V to 5V DC-DC converter to power a 3.3V linear regulator. The tightly regulated output of the 3.3V regulator powers the HI-6130 and CPLD.

PCIe current draw from the 12V rail cannot exceed 0.5A for x1 cards, 2.1A for x4/x8 cards or 4.4A for x16 cards. For a PCIe x1 card (like the Holt example) the worst-case available power from the slot 12V rail is (12V minus 8%) x .5A = 5.52W. The DC-DC conversion efficiency is approximately 90%, reducing available load capacity from 5.52 to 4.97W. Since the final 5V to 3.3V stage uses a linear regulator, output current equals regulator input current. Thus the maximum output load current is 4.97W / 5V = .99A. The 3.3V regulator can supply 3.27 Watts. Use x1 slots for smaller designs and use either x4/x8 or x16 slots for larger designs that require higher power.

For new custom designs, the user should be aware of the PCIe limitations on the maximum allowable current and power dissipation, which vary depending on card size (full height vs. low profile) and slot type used (x1 through x16).

Clocks

A single 50MHz oscillator module is shared between the PEX8311 LB, CPLD and HI-6130. The LB can be clocked up to 66MHz to achieve faster bus performance which might be desirable in a more complex design. The HI-6130 must always be clocked at 50MHz so an additional oscillator would be needed. If running the LB at 66MHz the number of wait states in the PLX LB EPROM registers would need to be increased to compensate for the faster clock. The LB and the CPLD must use the same clock to generate the synchronized strobe signals internal to the CPLD and the CSn, RDn and WRn strobe signals to the HI-6130. For a faster LB, use up to 66MHz for the LB clock input of the PEX8311 and CPLD and use a separate 50MHz clock for the HI-6130.

Additional memory

Adding additional on-board memory is possible with some moderate design effort. Connect the address and data bus directly to the LB signals of the PEX8311 and assign the required number of control signals for the memory by using some spare CPLD pins and modify the Verilog accordingly. Configure the LB EEPROM Space registers for the desired memory space, number of wait state and bus width type.

PCB layout considerations

The PLX data book and hardware checklist should be closely followed for the PCIe high speed bus signals. Review the Holt AN-550 for PCB layout guidelines for the transformers and decoupling capacitors for the HI-6120, HI-6130 and HI-6140.

Summary

The Holt HI-6130 PCIe low profile card reference design demonstrates how to interface the HI-6130 MIL-STD-1553 multi-terminal to a 1 lane (x1) PCIe bus. A Holt high-level API software library is provided and demonstrated in the demo software. All the design files are included on the CD-ROM, including the OrCAD schematics, two software demo projects and Verilog source for the CPLD with other related documents to enable rapid custom development. Some guidance how to enhance and customize the design with additional MIL-STD terminals, ARINC 429 protocol ICs, Discrete-to-Digital devices and memory was provided. For questions regarding this design contact Holt and for support on the PLX PEX8311 and PLX API software and drivers it is recommended to contact Broadcom directly using the technical support page on their website.

References:

http://www.holtic.com/category/351-mil-std-1553.aspx

http://www.holtic.com/category/352-arinc-429.aspx

www.holtic.com/category/420-discrete-to-digital-components.aspx

http://www.pcisig.com/home

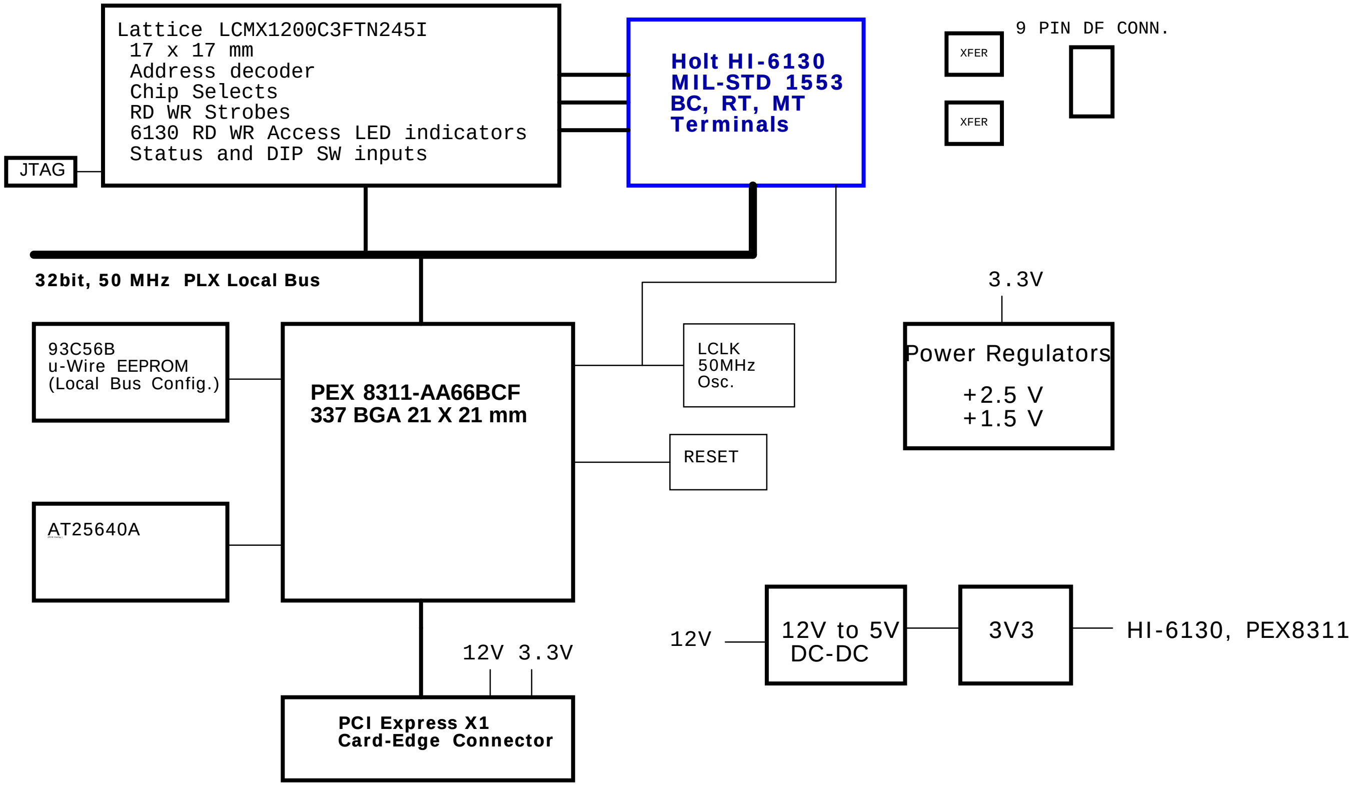
http://www.microsoft.com/visualstudio/eng/downloads

https://www.broadcom.com/products/pcie-switches-bridges/software-dev-kit

Note: Sometimes internet links become out of date so it might be necessary to navigate the main website and look for the desired information.

Table Of Contents

Block Diagram



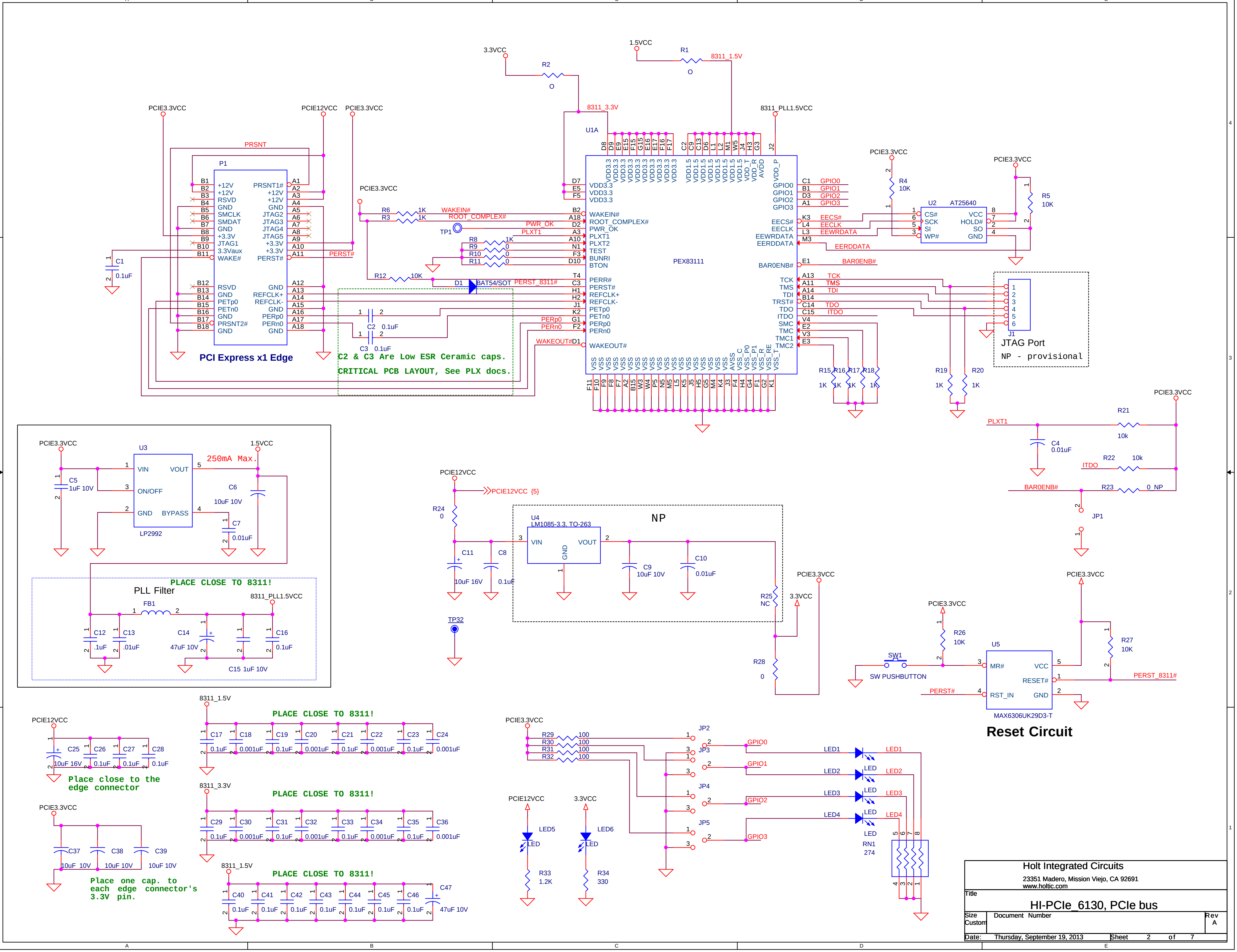
Date	Changes
9/19/2013	Rev A

Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title
HI-PCIe_6130

Size B | Document Number | Rev A

Date: Thursday, September 19, 2013 | Sheet 1 of 7



C2 & C3 Are Low ESR Ceramic caps.

CRITICAL PCB LAYOUT, See PLX docs.

PLACE CLOSE TO 8311!

Place close to the
edge connector

PLACE CLOSE TO 8311!

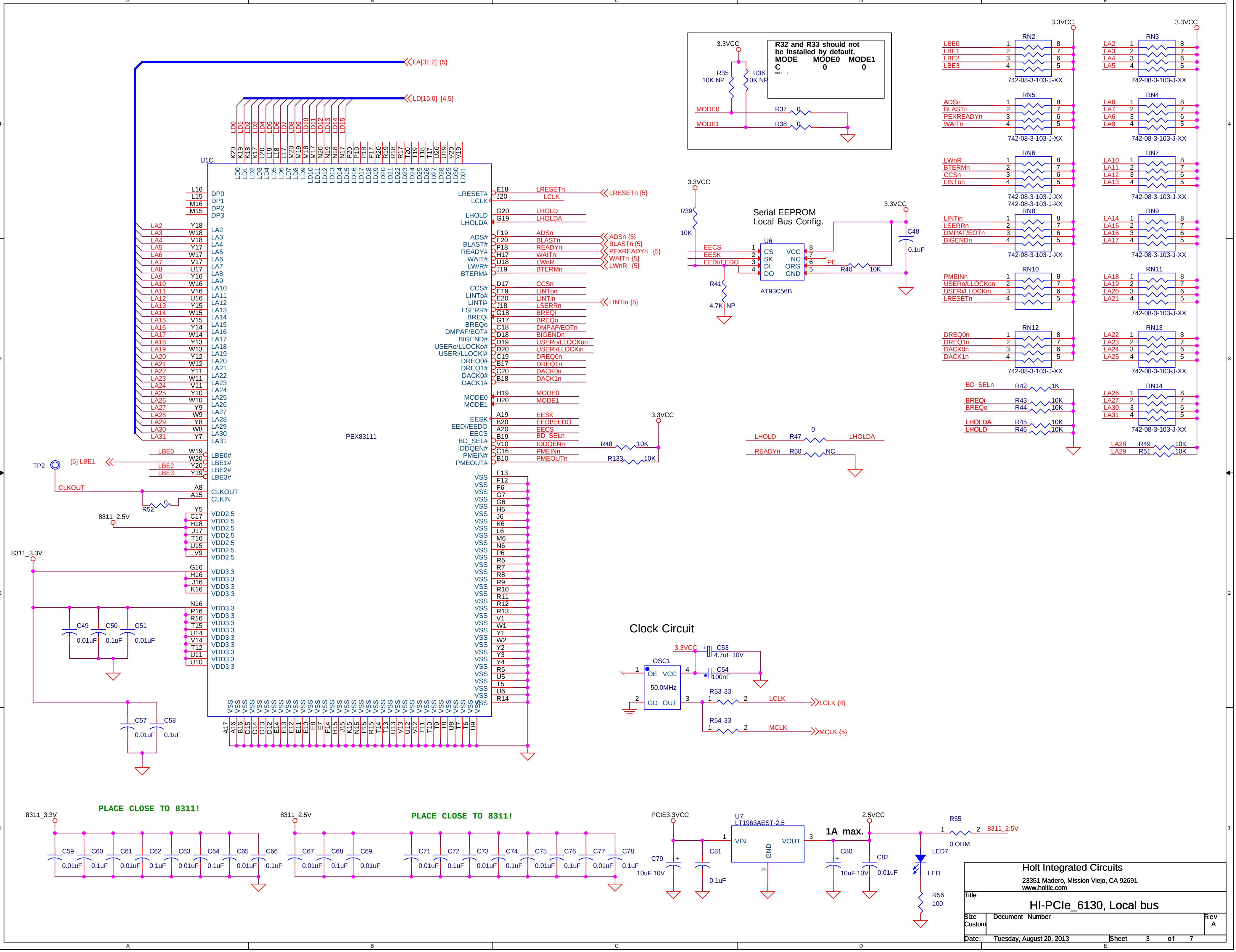
Place one cap. to
each edge connector's
3.3V pin.

PLACE CLOSE TO 8311!

PLACE CLOSE TO 8311!

Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title		
HI-PCIe_6130, PCIe bus		
Size Custom	Document Number	Rev A
Date: Thursday, September 19, 2013	Sheet 2 of 7	

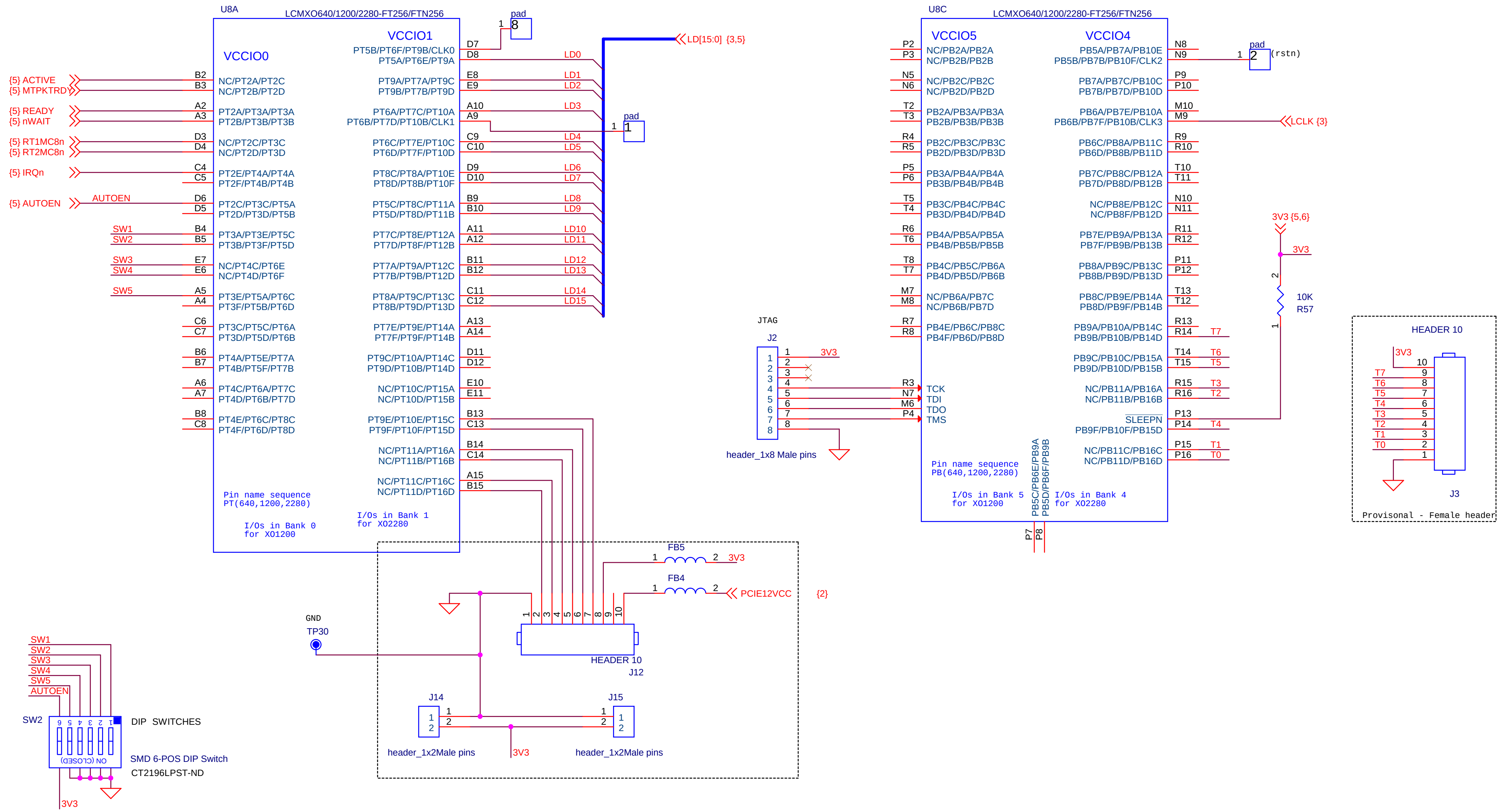


Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title
HI-PCIe_6130, Local bus

Size Custom | Document Number | Rev A

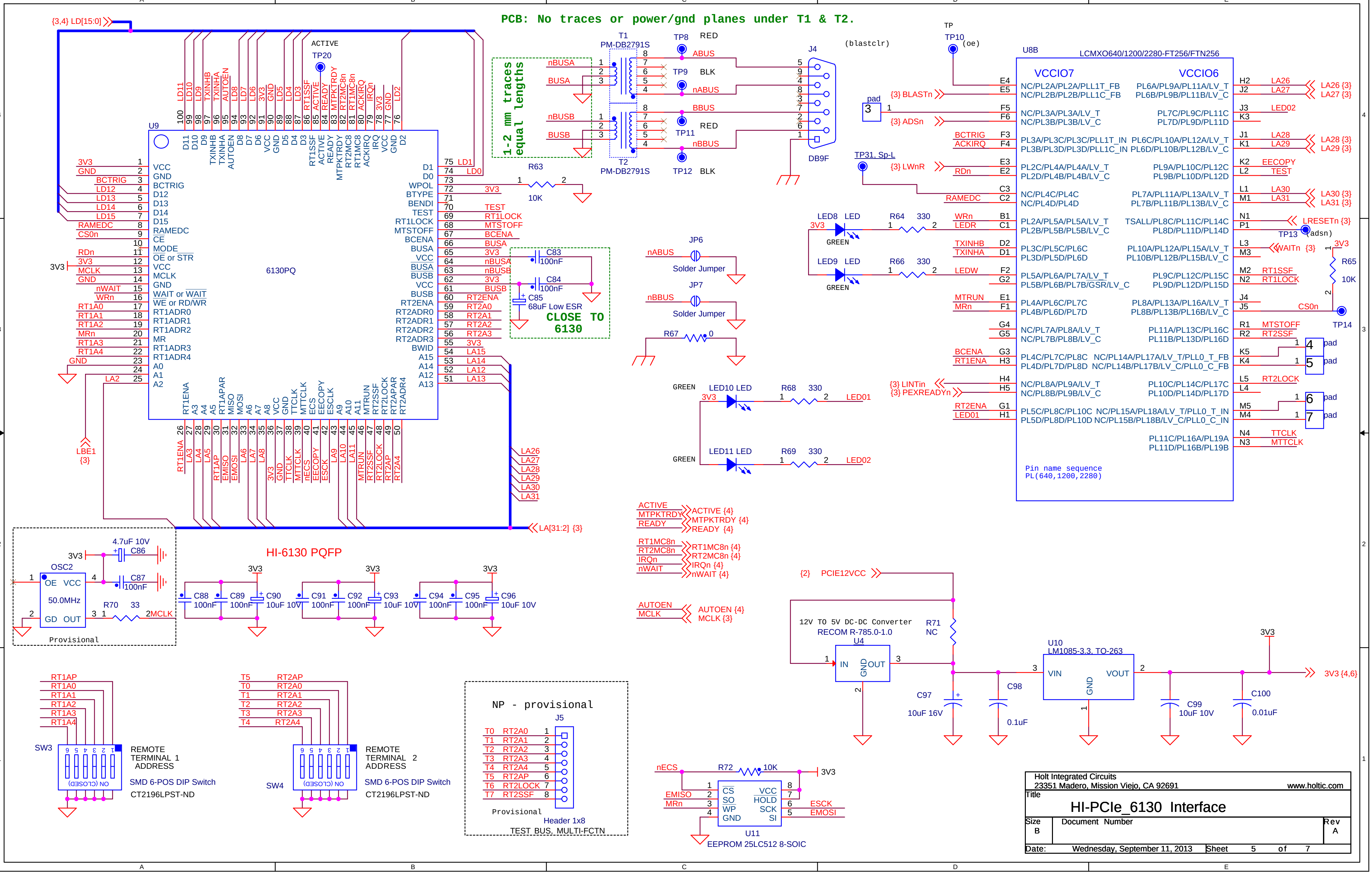
Date: Tuesday, August 20, 2013 | Sheet 3 of 7



Default: SW2 position 6 (AUTOEN)
should be Open (UP).

Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title	HI-6130 Status Inputs, JTAG	
Size B	Document Number	Rev A
Date:	Thursday, September 19, 2013	Sheet 4 of 7

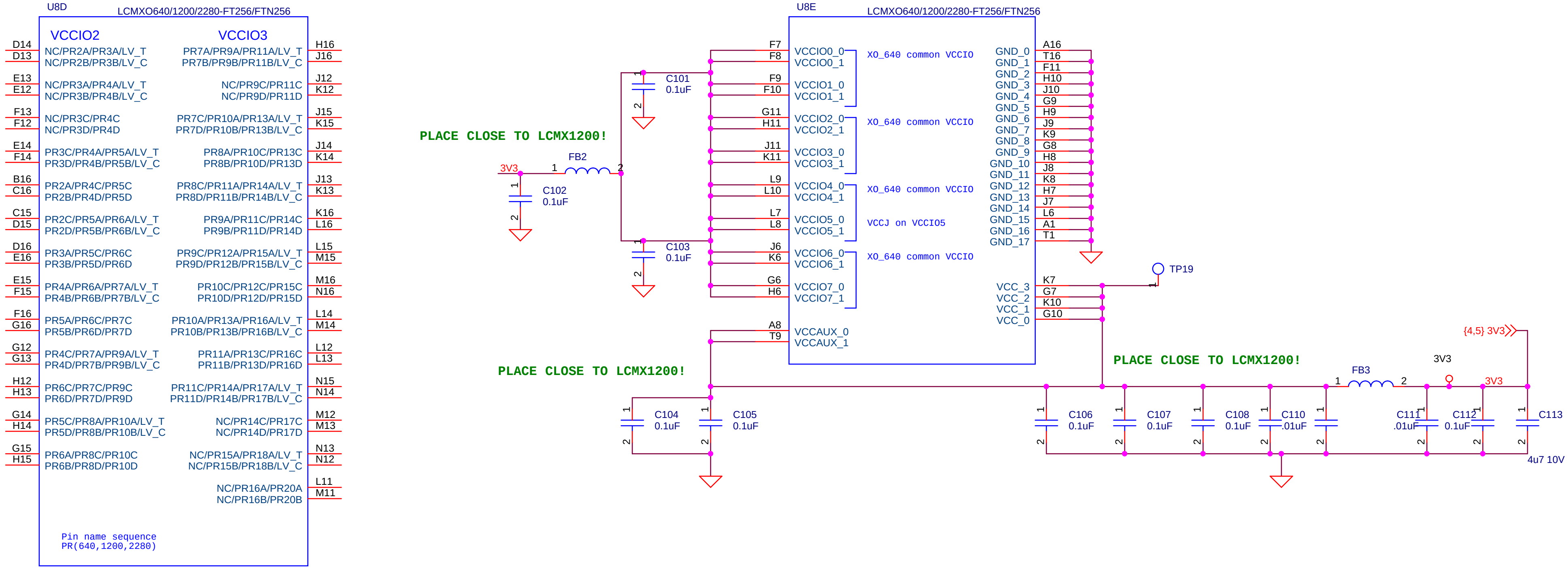


Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title
HI-PCIe_6130 Interface

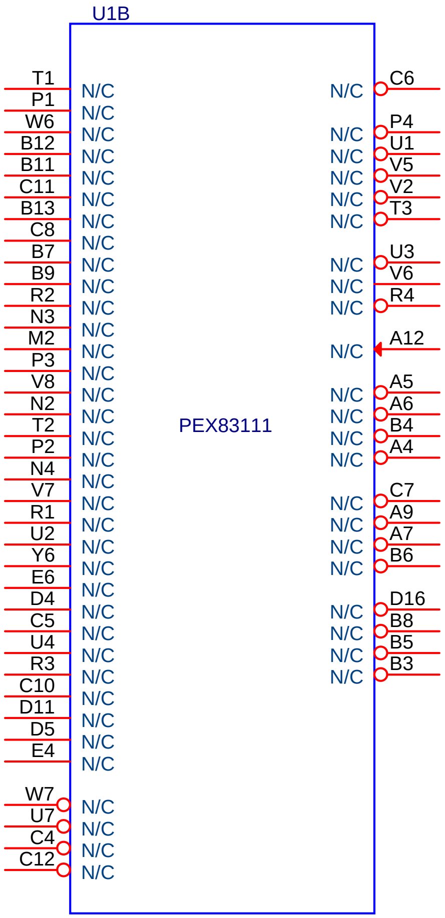
Size	Document Number	Rev
B		A

Date: Wednesday, September 11, 2013 Sheet 5 of 7



Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title	CPLD power	
Size B	Document Number	Rev
Date: Wednesday, August 14, 2013	Sheet 6 of 7	



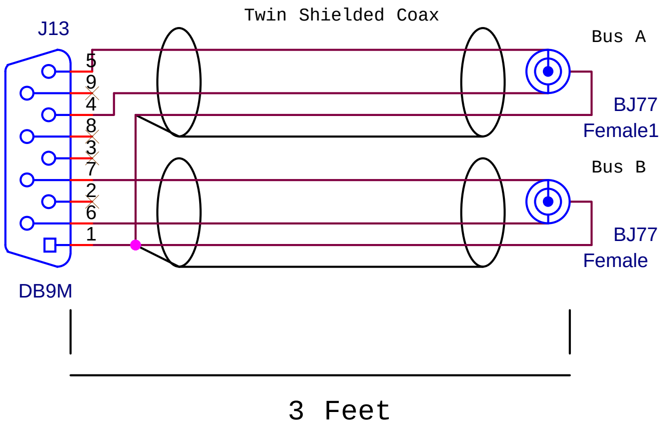
Holt Integrated Circuits
23351 Madero, Mission Viejo, CA 92691 www.holtic.com

Title
PEX 8311, NC BALLS

Size A
Document Number <Doc>
Rev

Date: Tuesday, August 13, 2013 Sheet 7 of 7

Bill of Materials		PCIe_HI-6130 Interface Board Rev. A			19-Sep-13
Item	***Qty***	***Description***	***Reference***	***DigiKey***	***Mfr P/N***
1	1	PCB, Bare, Evaluation Board	N/A	--------	Advanced Circuits 3568129-MQ
2	2	Ferrite Bead, 330 Ohm @ 100MHz 0805	FB4,FB5	490-5988-1-ND	BLM21PG331SN1D
3	3	Ferrite Bead, 390 Ohm @ 100MHz 500mA DC 0603	FB1,FB2,FB3	490-5980-1-ND	BLM18EG391TN1D
4	8	Capacitor, Ceramic 0.001uF 50V 10% X7R 0603	C18,C20,C22,C24,C30,C32,C34,C36	399-1082-1-ND	Kemet C0603C102K5RACTU
5	20	Capacitor, Ceramic 10nF 10% 50V X7R 0603	C4,C7,C13,C49,C51,C57,C59,C61,C63,C65,C67,C69,C71,C73,C75,C77,C82,C100,C110, C111	399-1091-1-ND	Kemet C0603C103K5RACTU
6	58	Capacitor, Ceramic 100nF 10% 50V X7R 0603	C1, C2, C3, C8, C12, C16, C17, C19, C21, C23, C26, C27, C28, C29, C31, C33, C35, C40, C41, C42, C43, C44, C45, C46, C48, C50, C54, C58, C60, C62, C64, C66, C68, C72, C74, C76, C78, C81, C83, C85, C87, C88, C89, C91, C92, C94, C95, C98, C101, C102, C103, C104, C105, C106,	399-5089-1-ND	Kemet C0603C104K5RACTU
7	2	Capacitor, Ceramic 1uF 10V 10% X7R 0603	C5,C15	399-9449-1-ND	Kemet T491A105K010AT
8	1	Capacitor, Ceramic 4.7uF 10V 10% X5R 0603	C113	399-5503-1-ND	Kemet C0603C475K8PACTU
9	2	Capacitor, Tantalum 4.7uF 10% 10V Size A	C53,C86	399-3699-1-ND	Kemet T491A475M016AT
10	12	Capacitor, Tantalum 10uF 10V 10% Size A	C6,C10,C14,C37,C38,C39,C79,C80,C90,C93,C96,C99	399-3684-1-ND	Kemet T491A106K010AT
11	3	Capacitor, Tantalum 10uF 16V 10% Size A	C11,C25,C97	399-3687-1-ND	Kemet T491A106M016AT
12	2	Capacitor 47uF 10% 6.3V Tantalum Low ESR SMD C	C14,C47	495-1543-1-ND	Kemet T495C476K016ZTE300
13	1	Capacitor 68uF 10% 6.3V Tantalum Low ESR SMD C	C84	495-1507-1-ND	Kemet T495C107K006ZTE150
14	1	Resistor Array 270 Ohm 4 Res 1206	RN1	742C083271JPCT-ND	CTS 742C083271JP
15	13	Resistor Array 10K Ohm 4 Res 1206	RN2,RN3,RN4,RN5,RN6,RN7,RN8,RN9,RN10,RN11,RN12,RN13,RN14	742C083103JPCT-ND	CTS 742C083103JP
16	1	Resistor, 0.0 Ohm 1/10W 0603	R23	DNP	Panasonic ERJ-3GEY0R00V
17	1	Resistor, Provision, 5% 1/10W 0603	R50	DNP	
18	1	Resistor, 4.7K ohm 5% 1/10W 0603	R41	DNP	Panasonic ERJ-3GEYJ472V
19	2	Resistor, 10K Ohm 5% 1/10W 0603	R35,R36	DNP	Panasonic ERJ-3GEYJ103V
20	2	Resistor 0.0 Ohms, 1/8W, 0805	R25,R71	DNP	Panasonic ERJ-6GEY0R00V
21	5	Resistor 0.0 Ohms, 1/8W, 0805	R2,R24,R28,R55,R67	P0.0ACT-ND	Panasonic ERJ-6GEY0R00V
22	8	Resistor, 0.0 Ohm, 1/10W, 0603	R1,R9,R10,R11,R37,R38,R47,R52	P0.0GCT-ND	Panasonic ERJ-3GEY0R00V
23	3	Resistor, 33 Ohm 5% 1/10W 0603	R53,R54,R70	P33GCT-ND	Panasonic ERJ-3GEYJ330V
24	5	Resistor, 100 Ohm 5% 1/10W 0603	R29,R30,R31,R32,R56	P100GCT-ND	Panasonic ERJ-3GEYJ101V
25	5	Resistor, 330 ohm 5% 1/10W 0603	R34,R64,R66,R68,R69	P330GCT-ND	Panasonic ERJ-3GEYJ331V
26	10	Resistor, 1K Ohm 5% 1/10W 0603	R3,R6,R8,R15,R16,R17,R18,R19,R20,R42	P1.0KGCT-ND	Pansonic ERJ-3GEYJ102V
27	1	Resistor, 1.2K Ohm 5% 1/10W 0603	R33	P1.2KGCT-ND	Pansonic ERJ-3GEYJ122V
28	21	Resistor, 10K Ohm 5% 1/10W 0603	R4,R5,R12,R21,R22,R26,R27,R39,R40,R43,R44,R45,R46,R48,R49,R51,R57,R63,R65,R72,R133	P10KGCT-ND	Panasonic ERJ-3GEYJ103V
29	4	Header 1x3, Male, 0.1" Pitch	JP2,JP3,JP4,JP5	S1012E-03-ND	Sullins PEC36SAAN
30	4	Shunt 1x2, 0.1" Pitch	JP2, JP3, JP4,JP5	A26227-ND	TE 382811-6
31	3	Header 1x2, Male, 0.1" Pitch	JP1,J14,J15	S1012E-02-ND	Sullins PEC36SAAN
32	1	Header 1x6, Male, 0.1" Pitch	J1	S1012E-06-ND	Sullins PEC36SAAN
33	2	Header 1x8, Male, 0.1" Pitch,	J2,J5	S1012E-08-ND	Sullins PEC36SAAN
34	2	Header 1x10, Male, 0.1" Pitch,	J3,J12	S1012E-10-ND	Sullins PEC36SAAN
35	1	Osc 50MHz, 25ppm, 3.3V SMD 5x7mm	Osc1	535-10087-1-ND	Abracon ASV-50.000MHZ-E-T
36	11	LED Green 0805	LED1-LED11	160-1179-1-ND	Lite-On LTST-C170GKT
37	1	Diode Schottky 40V 0.3A SOT23	D1	497-12131-1-ND	ST BAT54FILMY
38	1	Switch Tactile SPST-NO 0.02A 15V	SW1	P12943SCT-ND	Panasonic EVQ-Q2K03W
39	3	DIP Switch 6-Pos Half Pitch SMD	SW2,SW3,SW4	CT2186LPST-ND	CTS 218-6LPST
40	2	Test Point, Red, 0.062" hole	TP8,TP11	5010K-ND	Keystone 5010
41	3	Test Point, Black, 0.062" hole	TP9,TP12,TP32	5011K-ND	Keystone 5011
42	1	Test Point, White, 0.062" hole	TP20 - Active	5012K-ND	Keystone 5012
43	4	Test Point, White, 0.040" hole	TP10,TP13,TP14,TP31	5002K-ND	Keystone 5002
44	2	Test Point, 0.040" hole	TP2,TP19	N/A	
45	7	Test Point Pad	1,2,3,4,5,6,7	N/A	
46	1	Connector DB9F, R/A PCB Short Body, Board Lock	J4	A35107-ND	TE 1734354-1
47	1	PEX 8311 PCIe IC - 337BGA	U1	862-PEX8311-AA66BC-F (Mouser PN)	PLX PEX-8311-AA66BCF
48	1	IC PLD 1200LUTs 211 I/O 256FTBGA	U8	220-1064-ND	Lattice LCMXO1200C-3FTN256I
49	1	IC EEPROM 512Kbit 20 MHz 8-SOIC	U11	25LC512T-I/SNCT-ND	Micro Chip 25LC512T-I/SN
50	1	IC, EEPROM 64K 20MHz 8-SOIC	U2	AT25640B-SSHL-T-ND	Atmel AT25640B-SSHL-T
51	1	IC HI-6130 100-PQFP	U9	HI-6130	Holt IC
52	2	Transformer PM-DB2791S 2.5T	T1,T2	PM-DB2791S	Holt IC
53	1	Regulator LDO 1.5V 0.25A SOT23-5	U3	LP2992IM5-1.5/NOPBCT-ND	LP2992IM5-1.5/NOPB
54	1	Regulator 3.3V 3A, DDPAK TO-263-4	U10	LM1085IS-3.3/NOPB-ND	TI LM1085ISX-3.3/NOPB
55	1	Converter DC/DC 1A 5V Out SIP Vertical	U4	945-1038-ND	Recom Power R-785.0-1.0
56	1	IC Reg LDO 2.5V 1.5A SOT223	U7	LT1963AEST-2.5#PBF-ND	Linear LT1963AEST-2.5#PBF
57	1	Reset IC, 2.93V, SOT-23-5L	U5	MAX6306UK29D3+TCT-ND	Maxim MAX6306UK29D3+T
58	1	IC,3-wire EEPROM 2KBb, 8-SOIC	U6	AT93C56B-SSHM-TCT-ND	Atmel AT93C56B-SSHM-T
59	1	Low Profile PCIe Bracket w/ DB 9 Opening	N/A	Low Bracket Star Mfg	Star Manufacturing
60	1	High Profile PCIe Bracket w/ DB 9 Opening	N/A	High Bracket Star Mfg	Star Manufacturing
61	2	Triax Connector w/ Bend Relief Spring Plug Crimp	N/A	Mouser 530-PL75-29	Trompeter/Emerson PL75-29
63	2	Two 3ft in Length Cable	N/A	Mouser: 530-TWC-78-1/500ft	Twinax TWC-78-1 - 500ft Reel
64	1	Conn DB9 Male Solder Cup Nickel w/ Gold Plated Pins	N/A	209ME-ND	Norcomp 171-009-103L001
65	1	Backshell DB9 Die Cast Black Chrmt	N/A	970-09BCA-ND	Norcomp 970-009-040R011



Holt Integrated Circuits

23351 Madero, Mission Viejo, CA 92691
www.holtic.com

Title HI-6130 PCIe MIL-STD-1553 Cable		
Size A	Document Number	Rev NEW
Date: September 2013	Sheet 1 of 1	