TCPU Rev C Engineering and User's Manual

Version 6.0a (01.13.2010)

Lloyd Bridges
William Burton
Jo Schambach

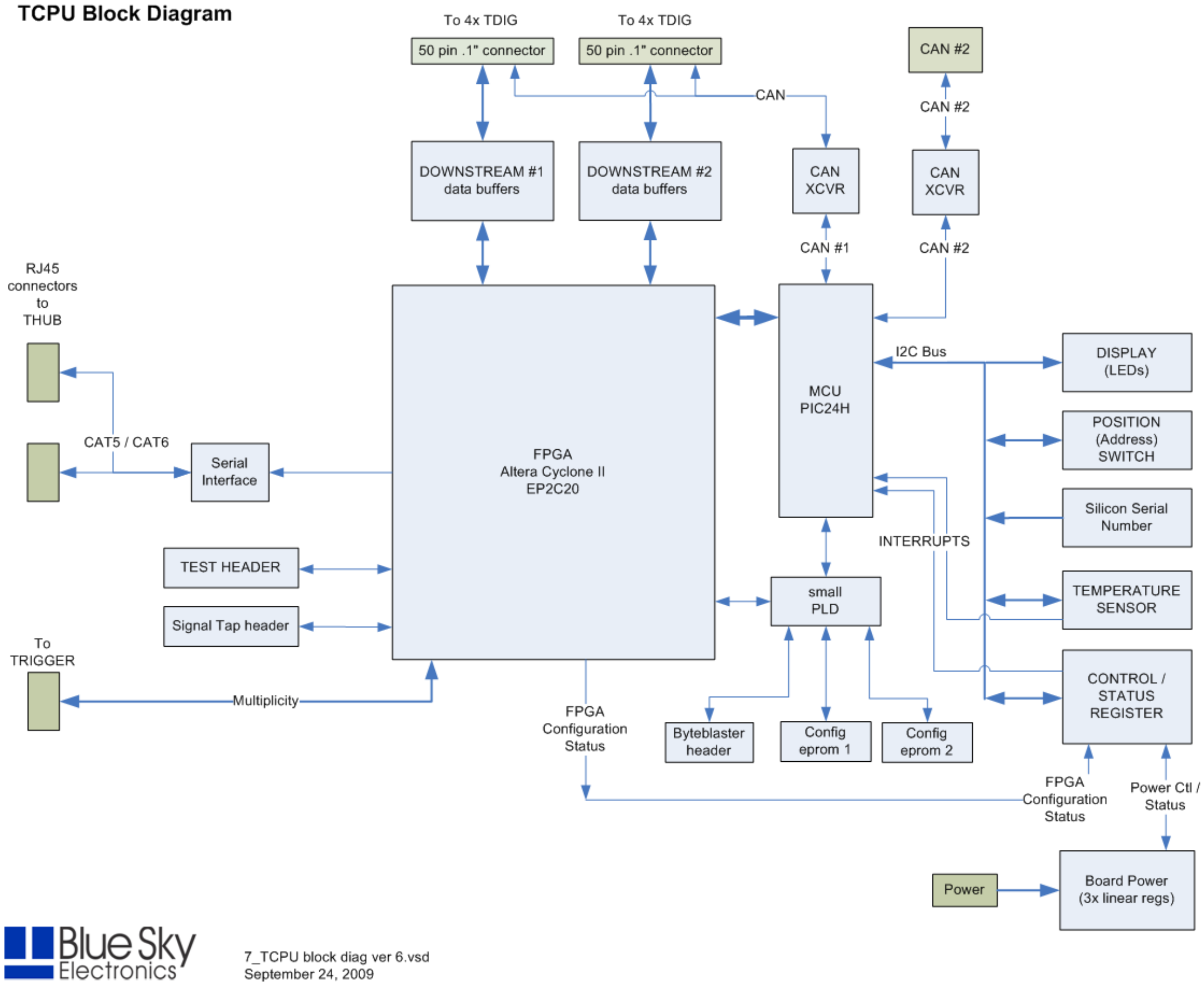
TCPU Engineering and User Manual

Version 6_x: September 2009

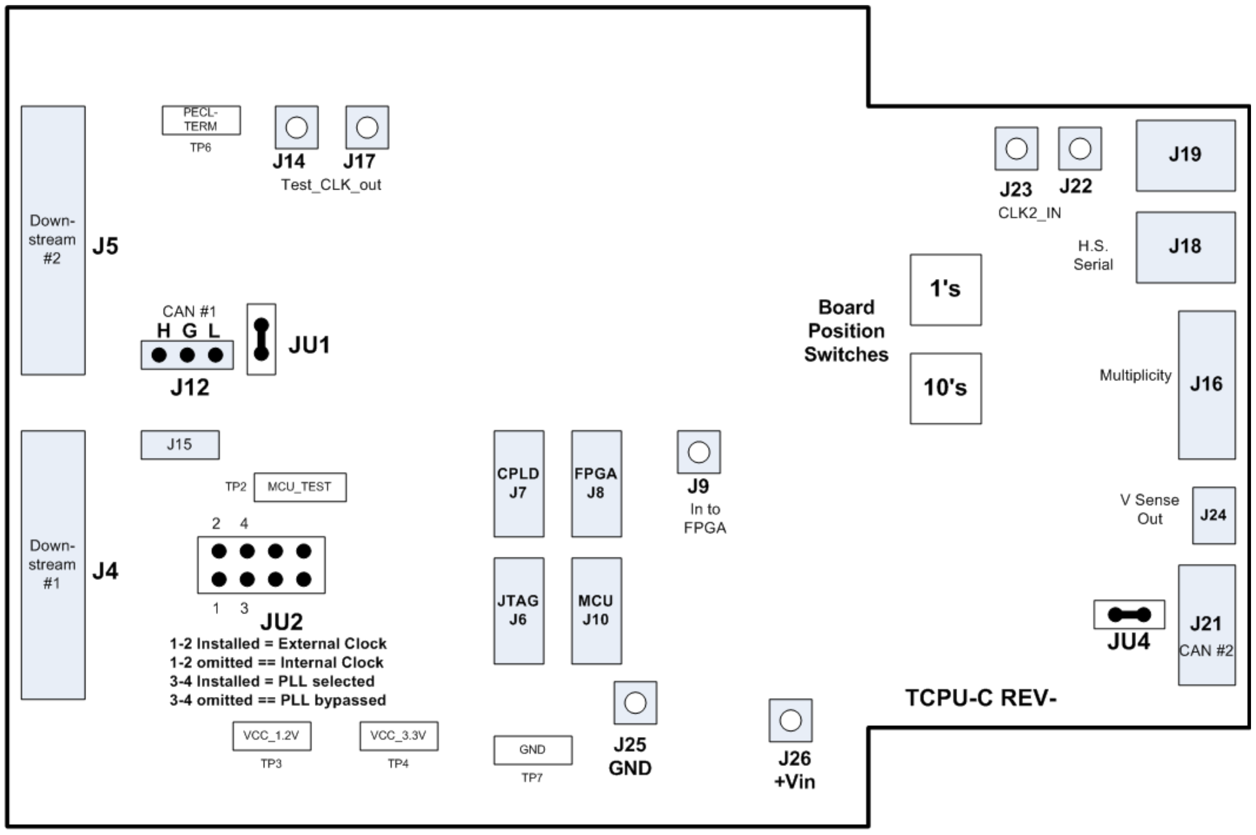
Table of Contents

1. Block Diagram

TCPU Block Diagram



2. Board Outline and Connector Layout



3. Major Components and Interfaces

3.1. Microcontroller (U19):

The Microcontroller (MCU) is a Microchip PIC24HJ256GP610-I/PT. This chip is programmed to perform the supervisory functions for the TCPU board; respond to commands arriving via the CAN Bus; and to pass CAN Bus messages from the Internal (Tray) CAN Bus to the External (THUB) CAN Bus. The microcontroller oversees clock and PLL selection, (re)-configuration of the FPGA, and downloading of new code into itself (second-image code) and the second EEPROM for the FPGA. The MCU communicates using a parallel data /control path with registers internal to the FPGA to perform various control and status functions.

3.1.1. I2C Bus (U19):

The Microcontroller has an I^{2}C (Inter-Integrated-Circuit) Bus used for accessing additional functions of the TCPU Board. The functions, bits, and addresses for the chips attached to the bus are defined in firmware source file: <TCPU-C_board.h>

Chip Designator	Function	Chip Type	I2C Address (hex)	Used For
U34	Parallel I/O	MCP23008	0x40x	LEDs, one LED per bit[7..0], the LED for bit 7 is amber.
U35	Parallel I/O	MCP23008	0x42x	Board ID (Location) switch SW4 and SW5 inputs.
U36	Parallel I/O	MCP23008	0x44x	“Extended Control and Status Register” ECSR. Bit 0: PLD_CONFIG_DONE input Bit 1: PLD_INIT_DONE input Bit 2: PLD_CRC_ERROR input Bit 3: PLD_nSTATUS input Bit 4: Button press input; 1==pressed Bit 5: Jumper 5-6; input; 1==installed Bit 6: Jumper 3-4; input; 1==installed Bit 7: Jumper 1-2; input; 1==installed
U37	Temperature	MCP9801	0x94x	Temperature monitoring and alert
U60	Serial Number	DS28CM00	0xA0x	Hardware serial number

3.1.2. MCU Port Usage (U19):

The MCU contains a number of ports and bits used either as byte-wide or bit-oriented input and/or output lines to other elements of the TCPU. Symbolic definitions for these signals are made in the TCPU firmware source file: TCPU-C_Board.h

MCU Port/bits	Direction (IN or OUT)	Signal Name	Used For
D.0	Out	PLL_RESET	Causes PLL to re-calibrate/re-initialize
D.1	In	PLL_LOS	PLL Status – Loss of Input Signal
D.2	In	DH_ACTV	PLL Status
D.3	In	CAL_ACTV	PLL Status – Calibration (locking) in progress
D.4	In	MCU_EE_DATA	EEPROM #2 data
D.5	Out	MCU_EE_DCLK	
D.6	Out	MCU_EE_ASDO	
D.7	Out	MCU_EE_NCS	
D.8	Out	MCU_SEL_EE2	
D.9	Out	MCU_CONFIG_PLD	

3.2. CAN Busses

The TCPU has 2 independent CAN bus controllers within the MCU, associated transceivers and cable connectors. The **External CAN Bus (HOST)** connects between trays and THUB. The **Internal CAN Bus (TRAY)** connects between TCPU and all TDIG boards within the same tray. The MCU monitors both CAN Busses and either interprets and processes commands received or passes messages from one CAN Bus to the other. Termination (nominally 120 ohm) of the CAN Busses is controlled by the presence/absence of jumpers JU4 for the External (HOST) and JU1 for the Internal (TRAY) CAN Bus. See the document "CAN Bus High Level Protocol" for a description of messages. The External CAN Bus operates at a nominal bit rate of 500 Kbits/second. The Internal CAN Bus to the TDIGs operates at a nominal bit rate of 1 Mbits/second. For TCPU MCU code after August 2009, the CANBus baud rate is jumper-selectable for 500 Kbits/sec or 1 Mbit/sec.

3.3. Tray geographic ID (SW4, SW5):

Each TCPU has two rotary switches for manual setting of the tray geographic ID: SW5 and SW4. Each switch has 10 positions, 0 to 9, which are used to set the tray geographic ID as a 2 digit decimal number – SW5 is the 10's digit and SW4 is the 1's digit. The MCU reads these switches using the I2C bus at power-up and after reset; limits the value modulo-32 for compliance with the CAN Bus communication protocol. This value becomes part of the TCPU (Tray) "address".

3.4. FPGA (U1), EEPROMs (U15, U18), CPLD (U10):

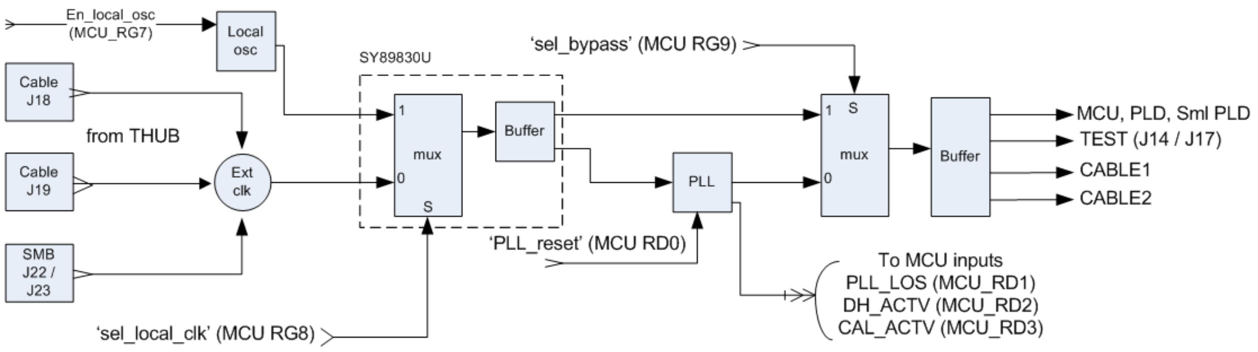
The FPGA is an Altera EP2C20F484. This chip is programmed to perform high speed data communication using the serial interface link to the THUB and provides status and control ports for access by the MCU. FPGA configurations can come from either of two EEPROMs (Altera EPCS4 and ST Micro). The first EEPROM can be programmed only via the Altera Byte-Blaster connection (J6). The second EEPROM can be reprogrammed using commands and data received over the External CAN Bus. (See section ?? for details). Under control of the MCU, the FPGA can be (re)configured from either EEPROM #1 or EEPROM #2. The CPLD (Altera EPM3064ATC44-10) is programmed (only via programming header J7) to manage configuration signals between the MCU, the EEPROMs, the FPGA programming connector (J6) and the FPGA. The MCU communicates with the FPGA and CPLD

3.5. Serial Link to-and-from THUB (U66):

The TCPU has a high-speed serial communication link for transmission and reception of high-speed data to THUB. This communication path uses the FPGA and a National Semiconductor DS92LV18T (Serializer/deserializer). See 11 for a description of the communication protocol.

3.6. Clock Distribution

The TCPU can use clocks from one of four sources and can (optionally) process that clock through a Phase Locked-Loop. The four sources are: a) On-board 40 MHz oscillator; b) External clock through Cable J18; c) External clock through J22 and/or J23; or d) External clock through Cable J19. Only one external clock source may be driven at any time. Driving multiple clock inputs simultaneously will lead to unstable operation.



3.6.1. Local Oscillator

In order to use the on-board (Local) oscillator, the MCU must both enable the oscillator chip using bit MCU_RG7, and select the local clock through the multiplexor using MCU_RG8. Clock input from external sources is thus disconnected from clock-processing circuits.

3.6.2. External Oscillator

To use the External Oscillator, one of the three External Oscillator sources is driven externally. The three inputs are capacitively coupled before being applied to the multiplexor (in effect creating a "wired-or"). The MCU must select the local clock through the multiplexor using MCU_RG8 set low.

3.6.3. Phase-Locked-Loop

The previously-selected Internal or External clock signal is applied to the PLL input. After the PLL_Reset signal has been toggled and the PLL has become locked, the PLL Output signal can be used as the board and cable clock source by selecting "sel_bypass, MCU_RG9 set low. A high-level on this signal sets the PLL Bypassed condition.

3.7. Tray Interface (J4, J5):

J4 and J5 are each 50 pin ribbon cable connectors which carry signals between the TCPU and 2 sets of TDIG cards. Each connector/cable communicates with 4 TDIG boards. The order and labeling of boards within a detector tray is as follows:

TDIG0	TDIG1	TDIG2	TDIG3	TDIG4	TDIG5	TDIG6	TDIG7	TCPU

Daisy-chained tray cabling is arranged physically and logically as follows:

TDIG0	←→	TDIG1	←→	TDIG2	←→	TDIG3	←→	J5 / TCPU

TDIG4	←→	TDIG5	←→	TDIG6	←→	TDIG7	←→	J4 / TCPU

See the TDIG Engineering/User Manual for further descriptions of board-to-board connections (Upstream and Downstream connectors).

4. Connector Summary

4.1. Low Voltage Power Input (J25, J26):

Two L.V. "Screw Terminals" Keystone # 1202 or 8195
Mating wire lugs = MNG10-10-FB

Connector	Signal
J25	GND
J26	+4.5Vdc

4.2. Tray voltage Sense (J??)

3 Pin "Micro Mate-N- Lock" AMP # 1445098-3
Mating cable plug = AMP # 1445022-3 plus appropriate pins.
Pinout: 2 leads + shield. To be connected before the fuse on TCPU.
This "output" is current limited by a (140 milliamp) self-resetting fuse

Pin	Signal
1	
2	
3	

4.3. Downstream Cable # 1 (J4):

Connector: AMP 1-1761607-5, Header Assembly, Low Profile, RA, TH 50 pins, right angle with ejectors.
Mating connector: 50-position dual row ribbon-cable receptacle

CABLE SIGNAL	CABLE PINS	BOARD SIGNAL	DIRECTION
C1_DCLK_40MHZ	3,4	C1_DCLK_40MHZ	DOWNSTREAM
TRAY_CAN	13,14	TRAY_CAN	BIDIRECTIONAL
C1_DLV_0	7,8	C1_SER_OUT	UPSTREAM
C1_DLV_1	9,10	C1_TOKEN_OUT	UPSTREAM
C1_DLV_2	11,12	C1_STROBE_OUT	UPSTREAM
C1_DLV_3	15,16	C1_DDAISY_DATA	UPSTREAM
C1_DLV_4	17,18	C1_DDAISY_TOK_OUT	UPSTREAM
C1_DLV_5	19,20	C1_DSTATUS0	UPSTREAM
C1_DLV_6	21,22	C1_DSTATUS1	UPSTREAM
C1_DLV_7	23,24	C1_DDAISY_CLK	DOWNSTREAM
C1_DLV_8	25,26	C1_DDAISY_TOK_IN	DOWNSTREAM
C1_DLV_9	27,28	C1_MCU_RESETB	DOWNSTREAM
C1_DLV_10	29,30	C1_FLEX_RESET_OUT	DOWNSTREAM
C1_DLV_11	31,32	C1_TRAY_TOKEN	DOWNSTREAM
C1_DLV_12	33,34	C1_DSPARE_OUT1	DOWNSTREAM
C1_DLV_13	35,36	C1_BUNCH_RST	DOWNSTREAM
C1_DLV_14	37,38	DCONFIG_OUT	DOWNSTREAM
C1_DLV_15	39,40	C1_TRIGGER	DOWNSTREAM
C1_DLV_16	41,42	C1_CLK_10MHZ	DOWNSTREAM
C1_DLV_17	43,44	C1_DMULT0	UPSTREAM
C1_DLV_18	45,46	C1_DMULT1	UPSTREAM
C1_DLV_19	47,48	C1_DMULT2	UPSTREAM
C1_DLV_20	49,50	C1_DMULT3	UPSTREAM

Table 1: J4 Signal Assignments

4.4. J5 Downstream Cable # 2 (J5):

Connector: AMP 1-1761607-5, Header Assembly, Low Profile, RA, TH, 50 pins, right angle with ejectors.
Mating connector: 50-position dual row ribbon-cable receptacle

CABLE SIGNAL	CABLE PINS	BOARD SIGNAL	DIRECTION
C2_DCLK_40MHZ	3,4	C2_DCLK_40MHZ	DOWNSTREAM
TRAY_CAN	13,14	TRAY_CAN	BIDIRECTIONAL
C2_DLV_0	7,8	C2_SER_OUT	UPSTREAM
C2_DLV_1	9,10	C2_TOKEN_OUT	UPSTREAM
C2_DLV_2	11,12	C2_STROBE_OUT	UPSTREAM
C2_DLV_3	15,16	C2_DDAISY_DATA	UPSTREAM
C2_DLV_4	17,18	C2_DDAISY_TOK_OUT	UPSTREAM
C2_DLV_5	19,20	C2_DSTATUS0	UPSTREAM
C2_DLV_6	21,22	C2_DSTATUS1	UPSTREAM
C2_DLV_7	23,24	C2_DDAISY_CLK	DOWNSTREAM
C2_DLV_8	25,26	C2_DDAISY_TOK_IN	DOWNSTREAM
C2_DLV_9	27,28	C2_MCU_RESETB	DOWNSTREAM
C2_DLV_10	29,30	C2_FLEX_RESET_OUT	DOWNSTREAM
C2_DLV_11	31,32	C2_TRAY_TOKEN	DOWNSTREAM
C2_DLV_12	33,34	C2_DSPARE_OUT1	DOWNSTREAM
C2_DLV_13	35,36	C2_BUNCH_RST	DOWNSTREAM
C2_DLV_14	37,38	DCONFIG_OUT	DOWNSTREAM
C2_DLV_15	39,40	C2_TRIGGER	DOWNSTREAM
C2_DLV_16	41,42	C2_CLK_10MHZ	DOWNSTREAM
C2_DLV_17	43,44	C2_DMULT0	UPSTREAM
C2_DLV_18	45,46	C2_DMULT1	UPSTREAM
C2_DLV_19	47,48	C2_DMULT2	UPSTREAM
C2_DLV_20	49,50	C2_DMULT3	UPSTREAM

Table 1: J5 Signal Assignments

4.5. J18 Serial link to THUB (J18):

RJ45 connector part #AMP 5555153-1

Pin	**Signal**
1	RIN+ (receiver input)
2	RIN- (receiver input)
3	DO+ (transmitter output)
4	No connect
5	No connect
6	DO- (transmitter output)
7	Clock + (LVPECL) input
8	Clock – (LVPECL) input
9	GND (shield)
10	GND (shield)

4.6. Auxiliary Serial Link to THUB (J19)

RJ45 connector part #AMP 5555153-1

Pin	Signal
1	ext test 1
2	ext test 2
3	ext test 3
4	ext test 4
5	No connect
6	No connect
7	Clock + (LVPECL) "CLK3/CLK3B" signal pair
8	Clock – (LVPECL) "CLK3/CLK3B" signal pair
9	GND (shield)
10	GND (shield)

Ext test [4:1] signals are routed directly to the FPGA.

The SERDES chip used will be National DS92LV18.

Receiver needs 100 Ohm termination across the receiver lines

Clock used will be 20 MHz. Use PLL to clean up and multiply to 40 MHz on RCLK.

Can possibly be used then as input clock to TCPU.

Rev 2: Tests with the THUB prototype showed that the clock is not constant in phase between power-cycles, so it can't be used.

So we added a 40MHz LVPECL clock on pins 7 and 8 instead for this revision to test as possible candidate of TCPU master clock.

Separate clock inputs:

4.7. Clock Inputs (J23, J22)

SMB connectors for paired LVPECL-level clock input signals.

Connector	Signal
J22	Clock- (CLK2B, LVPECL)
J23	Clock+(CLK2, LVPECL)

Only 1 of the 3 clock signal pairs can be active (driven) at a given time. Input pairs can be permanently disabled by removing the AC coupling capacitors (C? and C?) from their signal path.

4.8. Programming headers

4.8.1. **J7**: Altera Byteblaster JTAG programming of CPLD

4.8.2. **J8**: Altera Byteblaster active serial programming of FPGA (EEPROM #1)

4.8.3. **J6**: Altera Byteblaster JTAG debugging of FPGA

4.8.4. **J10**: Microchip MPLAB ICD2 programming of MCU

4.9. Multiplicity

"Right angle shrouded pin header" AMP # 499913-2
Ribbon cable plug = 3M #3385-6000 (14 pin)
Voltage level: LVPECL
Chipset: Texas Instruments TB5D1M (Quad Differential PECL Driver)

PIN	SIGNAL
1	Bit1+
2	Bit1-
3	Bit2+
4	Bit2-
5	Bit3+
6	Bit3-
7	Bit4+
8	Bit4-
9	Bit5+
10	Bit5-
11	RHIC Clock +
12	RHIC Clock -
13	GND
14	GND

4.10. External CAN network connection (J21)

Connector: 3 pin "Mate N Lock" AMP # 1-350943
Location: bottom side of board
Cable plug AMP # 350766
Termination: Placing the shorting jumper at JU4 applies 120 ohm termination.

PIN	SIGNAL
1	CANL
2	GND
3	CANH

4.11. Internal (Tray) CAN network connection (J12, J4, J5)

Normally, these are part of the **downstream cable connections** via ribbon connectors J4, and J5 (see section 4.3 and section 4.4: “Tray interface”)
J4/ pin 14 (CANH), pin 13 (CANL)
J5/ pin 14 (CANH), pin 13 (CANL)
The 3-pin straight header J12 is available for testing (not connected when trays are installed)

J12 PIN	SIGNAL
1	CANL
2	GND
3	CANH

Termination: Placing the shorting jumper at JU1 applies 120 ohm termination.

4.12. Aux L.V. Out (J1)

This connector is for the Alternate Multiplicity "TTRIG" board.

Samtec: HLE-106-02-SM-DV-BE-A. Placed on top side of TCPU and accepts pins from below.

PIN	SIGNAL
1, 2, 3, 4	GND
5, 6, 7, 8	+V Tray
9, 10, 11, 12	GND

5. Jumper Settings:

Jumpers J11, J12, JU1, and JU4 control various configuration options which are expected to be non-dynamic.

5.1. J11: Always jumper 1-2

5.2. JU2:

5.2.1. Short 1-2 = external oscillator
5.2.2. Open 1-2 = internal oscillator

6. MCU to FPGA Communication

6.1. Registers 0 thru 15 within the FPGA:

6.1.1. MCU writes to:

Byte address (hex)	Register Name	Bit Position	Description (Active high unless indicated)	Default Value
0	**CONFIG_0**		**Configuration 0**	
		0	SELECT TEST INPUT TO MCU FIFO	0 (not in BNL)
		1	SELECT CABLE_2 (J5) INPUT TO MCU FIFO;1 = cable2 (J5) input	0 = cable1 (J4) (not in BNL boards)
		2	Select test mode for serial readout: FPGA uses Strobe 0 as signal to issue token and do serial readout from TDCs on both downstream cables.	0 (not in BNL boards)
		3	SEL TEST COUNTER DATA ON DOWNSTREAM CABLES	0 (not in BNL boards)
		4	ENABLE CABLE TEST MODE for TDC Trigger. FPGA uses STROBE_2 as signal to issue one trigger pulse to both downstream cables.	0 (not in BNL boards)
		5	Select test mode for Bunch Reset. Bunch reset is routed from J9 input.	0 (not in BNL boards)
		6	Select test mode for Event Reset. Event reset is issued when MCU issues strobe_8	0 (not in BNL boards)
		7		0
1	**CONFIG_1**		**Configuration 1**	
		0	RESET FPGA STATE MACHINES	0
		1	CLEAR FIFOs	0
		2		
		3		
		4		
		5		
		6		
		7		

Byte address (hex)	Register Name	Bit Position	Description (Active high unless indicated)	Default Value
2	**CONFIG_2**			
		0	ENABLE READOUT	0
		1	ENABLE SERDES TRIGGERS (0 = test pulse from J9 or from internal ctr – selected by config_14.3; 1 = decoded thub trigger)	0
		2	TURN-OFF CANBUS DATA PACKETS (0 = tdc data writes to mcu fifo are enabled)	0
		3	TURN ON SERDES LINK	0
3	**CONFIG_3**		**Configuration 3**	
4	**STROBE_0**	n/a	**Generate Test Token**	(not in BNL boards)
			Upon receipt of this signal, the serial readout controller will issue token and do serial readout from TDCs. Enabled by CONFIG_0.2.	
5	**STROBE_1**	n/a		
6	**STROBE_2**	n/a	**Generate Test Trigger**	(not in BNL boards)
			FPGA will generate 1 trigger pulse for each TDC for each STROBE_2 input. Enabled by CONFIG_0.4.	
7	**STROBE_3**	n/a	**Generate Test Bunch Reset**	(not in BNL boards)
			FPGA will generate 1 bunch reset pulse for each TDC for each STROBE_3 input. Enabled by CONFIG_0.5	
8	**STROBE_4**		**Multiplicity clock delays**	
		3:0	Clock phase for data register to DSMI	0 (1/16th RHICstr)
		7:4	Clock phase to trays for multiplicity	0 (1/16th RHICstr)
9	**STROBE_5**	n/a	**Reset Readout**	(not in BNL boards)
			Resets local serial readout state machine.	
9	**CONFIG_9**			
		7	Multiplicity (0) or Test/Ramp (1) data	0
A	**STROBE_6**	n/a	**Reset MCU FIFO**	(not in BNL boards)
B	**STROBE_11**	n/a	**Increment Test Counter**	(not in BNL boards)
			Increments test data counter to MCU FIFO	
C			**Switch Position**	(not in BNL boards)
			Switch position 6:0 right justified	
D				

Byte address (hex)	Register Name	Bit Position	Description (Active high unless indicated)	Default Value
E				
		2:0	Internal test pulser frequency	0
		3	Test pulse: J9 (0) or internal pulser (1)	0
		4	Bunch reset	0
		7	TDC readout through FPGA (1) or Aux path(0)	0
F				

6.1.2. MCU READS FROM

Byte address (hex)	Register Name	Bit Position	Description	Default Value
0	**CONFIG_0**		**Configuration 0 readback**	
1	**CONFIG_1**		**Configuration 1 readback**	
2	**CONFIG_2**		**Configuration 2 readback**	
3	**STATUS**		**Configuration 3 readback**	
		0	Serdes_lock_n	Active low
		1	Serdes ready	
4			TEST CABLE 1, BYTE 0	(not in BNL boards)
5			TEST CABLE 2, BYTE 0	(not in BNL boards)
6				
7	**VERSION_ID**		**TCPU FPGA CODE VERSION**	
			Constant value can be used to ID FPGA code version	
8				
9				
A			**CLK TEST COUNTER**	(not in BNL boards)
B			**FIFO (7:0) LS byte**	
C			**FIFO (15:8)**	
D			**FIFO (23:16)**	
E			**FIFO (31:24) MS byte**	
			Reads should be from LS byte (read first) to MS byte. Read from MS byte generates read_clock_enable signal to FIFO.	
F			**FIFO STATUS**	(not in BN boards)
		4:0	MCU FIFO Occupancy (# of data words in FIFO	
		5	MCU FIFO PARITY	
		6	MCU FIFO FULL	
		7	MCU FIFO EMPTY	

6.2. Useful FPGA configuration bits and Operations

6.2.1. Bunch reset

Register.bit: CONFIG_14.4 This bit is mapped directly onto the output signals C1_BUNCH_RST and C2_BUNCH_RST. To issue a bunch reset signal (pulse), send a CAN Bus message that will toggle the contents of this bit low-high-low. The default value is low. CONFIG_0.3 must also be LOW (default), otherwise test counter data will be sent to these signals.

6.2.2. Select cable

Register.Bit: CONFIG_0.1 This bit selects between readout from Cable1 and Cable2. It will be superseded when default operation reads from both cables with a timeout so that the system won't hang if no boards are on one of the cables. Value = 0 is power-on default and selects Cable1 (J4). Value = 1 selects Cable2 (J5).

6.2.3. Select Auxiliary TDIG Readout

Register.bit: CONFIG_14.7 This bit selects between NORMAL (FPGA involvement) and AUXiliary data readout paths on the TDIG boards. Value = 0 is default and selects NORMAL. Value = 1 selects AUXiliary.

6.2.3.1. Aux data Readout (TDIG)

To use the AUX readout path, you must:

1. Set jumper J17 properly on each TDIG in the readout chain for Aux data path token routing – Jumper 2-3 if board is farthest from TCPU; otherwise jumper 1-2.
2. Set HPTDC configuration bit #44 "SELECT BYPASS INPUTS" to "1" in all TDCs in the readout chain.
3. Set TCPU FPGA configuration register bit CONFIG_14.7 = "1"
4. Unless power is cycled, a reset to the TCPU state machines, and possibly the HPTDCs, will probably be necessary when switching between modes (toggle CONFIG_1.0 HI then LO ("002 5 0E 01 00 01 00").

7. Serial Communication: TCPU(FPGA)←→THUB

TCPU high speed serial I/O uses the FPGA and DS92LV18T (U66, serializer/deserializer device)
TCPU outputs (to DIN[17:0] on U66) are labeled as "TX" for transmit.
TCPU inputs (from ROUT[17:0] on U66) are labeled as "RX" for receive.

All operations on U66 ser/deser data words (DIN / TX [17:0] and ROUT / RX [17:0]) assume the following bit ordering conventions:

LOGICAL BIT NAME on TCPU	BIT #	PIN #	BIT ORDER
TX_D0	0	21	LSB
TX_D1	1	22	
TX_D2	2	23	
TX_D3	3	24	
TX_D4	4	25	
TX_D5	5	26	
TX_D6	6	27	
TX_D7	7	28	
TX_D8	8	33	
TX_D9	9	34	
TX_D10	10	35	
TX_D11	11	36	
TX_D12	12	37	
TX_D13	13	38	
TX_D14	14	39	
TX_D15	15	40	
TX_D16	16	18	
TX_D17	17	3	MSB

LOGICAL BIT NAME on TCPU	BIT #	PIN #	NOTES
RX_D0	0	45	LSB
RX_D1	1	46	
RX_D2	2	47	
RX_D3	3	48	
RX_D4	4	54	
RX_D5	5	55	
RX_D6	6	56	
RX_D7	7	57	
RX_D8	8	64	
RX_D9	9	65	
RX_D10	10	66	
RX_D11	11	67	
RX_D12	12	70	
RX_D13	13	71	
RX_D14	14	72	
RX_D15	15	73	
RX_D16	16	62	
RX_D17	17	80	MSB

Refer to Appendix A – High Speed Serial Messages for details of data message formats.

8. Option Reference

8.1. Board Firmware Identifiers

Beginning at TDIG firmware version 11D CANBus messages are available which enable reading the “Firmware Identifiers” for the MCU and FPGA components of boards.

Method:

1. Issue the CANBus command: “Read board”, length 1, message 0xB1.
2. The response will be a “Read reply”, length 4, message 0xB1 <2-byte MCU firmware ID><1-byte FPGA ID>.

Example:
1. Send: 0x104, length 1, 0xB1
2. Receive: 0x105, length 4, 0xB1 0x0D 0x11 0x55

Details: IDs are built into the firmware components as follows:
1. The MCU firmware ID is defined through bytes FIRMWARE_ID_0 and FIRMWARE_ID_1 in the main program source file “TDIG-D_VER...C”.
2. The FPGA firmware ID is set through definition of the contents returned to “data7x” when the MCU reads FPGA register 7 (currently “CONSTANT five_five_byte...”). This is defined in module TDIG_pins1.vhd in the TDIG Altera source directory.

8.2. Board Programming using Pre-Compiled Files

8.2.1. CPLD Programming

Note that the CPLD must be programmed prior to attempting to program the FPGA.

a. Start Quartus; select “run with existing license” (Or start Quartus Programmer stand-alone program and skip to step c.)
b. Select “Tools”; select “Programmer”, a blank programming screen opens titled “QuartusII – [chain1.cdf].
c. Select “File”; select “Open”; select “Files of type: Programming Files...”
d. Browse to the location containing “SmallCPLD.CDF”; click “Open”.
e. Select “Open a new Programmer window listing this file”; click “OK”.
f. The chain will open.
g. The display will show “Mode JTAG”; “Progress 0%”.
h. The release programming filename; “Device EPM3064AL44”; Checksum as documented in the release document; “Usercode” all F’s should display.
i. The two boxes labeled “Program/Configure” and “Verify” must be checked.
j. Connect ByteBlaster cable to board header J7; apply power to board.
k. Click the button labeled “Start”. The program/verify sequence will run and programming progress will be indicated in the “Progress” bar.

8.2.2. FPGA Programming

Note that the CPLD must be programmed prior to attempting to program the FPGA.

a. Start Quartus; select “run with existing license” (Or start Quartus Programmer stand-alone program and skip to step c.)
b. Select “Tools”; select “Programmer”, a blank programming screen opens titled “QuartusII – [chain1.cdf].
c. Select “File”; select “Open”; select “Files of type: Programming Files...”
d. Browse to the location containing “TDIG_pins1.CDF”; click “Open”.
e. Select “Open a new Programmer window listing this file”; click “OK”.

f. The chain will open.
g. The display will show “Mode Active Serial Programming”; “Progress 0%”.
h. The release programming filename; “Device EPCS4”; Checksum as documented in the release document; “Usercode” all 0’s should display.
i. The boxes labeled “Program/Configure” and “Verify” must be checked for both the EPCS4 and the sub-device “Page 0” (4 check-marks total).
j. Connect ByteBlaster cable to board header J8; apply power to board.
k. Click the button labeled “Start”. The program/verify sequence will run and programming progress will be indicated in the “Progress” bar.

8.2.3. MCU Programming

a. Start MPLAB
b. Select “Programmer”;
c. Select “MPLAB ICD2”. If the MPLAB module is connected to the board several messages will appear, ending with “MPLAB ICD2 Ready”.
d. Select “File”; select “Import”.
e. Browse to the location containing the release filename.hex
f. Select the filename and click “open”. The message “Loaded should appear in the MPLAB “Output” window.
g. Connect Blue Sky Electronics programming cable from MPLAB ICD2 to board header J10; apply power to board.
h. Select “Programmer”; then select “Program”. MPLAB will erase, program, and verify the MCU code image. The message “Programming succeeded will appear in the “Output” window. (The yellow “Program target device” icon performs the same action as the “programmer/program” selection sequence).

8.3. MCU Reprogramming via CANBus

0. Introduction:
Some considerations in section 2 relating to the alternate vector table are subject to change when alternate download code is developed.
This note describes programming and linking considerations and the download procedure for programming a second code image into the PIC24HJxxx processors used on the Blue Sky Electronics TDIG-D or later and TCPU-B or later boards.

1. Initial programming:
Presuming the MCU has been programmed (using MPLAB or via some other means) with MCU code version 11d or later for TDIG, version 1F or later for TCPU.

2. Considerations for the design and coding of the “second image”:
The PIC24HJxxx series processors have the ability to re-program while running. The download procedure takes advantage of this ability and the presence of an “alternate” interrupt vector table. The ability to return to the “original image” code necessitates some explicit coding which might not otherwise be needed.

a. Any setup of CPU registers and configurations must be done explicitly in the second-image code.

b. It is not possible, except through explicit reprogramming, to alter the CPU identification fields. ID fields, Clock select fields, and other special “one-time” fields are not generally reprogrammed by the download procedure.
c. The second-image is entered from the first image via a Jump to location 0x4000 with interrupts disabled and the Alternate Interrupt Vectors selected in the CPU INTCON register. The user should readjust interrupt priorities during start-up of the second-image code but should NOT select “standard” interrupt vector mapping.
d. The first- and second- images share the DMA and RAM space.
e. Upon startup, the RAM space is cleared by the C-Language-provided startup routines CRT0() which is entered automatically at startup and reset.

3. Considerations for linking “second image” code:
Stand-alone (“First” images) will use the normal Microchip provided linker script file for the processor. Downloadable “Second” images MUST be linked using the modified linker script file (download-P24HJ....GLD). The special linking control file provides the following:

a. Reassignment of the “origin” to a location above the end of the first-image code. Currently location 0x4000 is defined as the start location for second-image code.
b. Reduction in the amount of available program space to allow for the first image to be retained.
c. Remapping the “Standard” interrupt vector table into the space normally occupied by the “Alternate” interrupt vector table. The user does NOT use the “Alt...” form of the definition for interrupt service routines. This simplifies the development and debugging of code through MPLAB as though it were the “first image”. It is possible to use the Alt... form of the interrupt vectors; in which case the remapping/reassignment provided by the .GLD file is not needed and should be removed.
d. The remainder of the usual link options remain; however certain of the fields will not be downloaded, specifically:
    1) The RESET vector at location 0
    2) The Oscillator select/control registers FOSCSEL, FOSC locations 0xF80006 and F80008.
    3) The Watchdog and Power-On Reset registers FWDT and FPOR at locations 0xF8000A and 0xF8000C
    4) The Processor ID fields (FUID0...3) at locations 0xf80010 through 0xF80016.
e. After linking, examine (using “Notepad” or other text editor) the linker output map (filename.MAP). It is important to verify that the “__resetPRI” external symbol references location 0x4000. This insures that the correct startup code will be executed (as though a hardware RESET had occurred) when control transfers after the download.

4. Downloading the second image to the MCU:

The result of the linking procedure is a file with the .hex extension. This file describes the in-memory image of the completed program. This image is parsed and downloaded using program MCU2.EXE via a series of CANBus block-transfer messages to the "first-image" code running in the MCU. The MCU transfers the image to MCU program (flash) memory. Once the download is complete, a final CANBus message triggers transfer of control to the new image. The download protocol as implemented by the program MCU2.exe proceeds as follows (MCU2.cpp):

a. MCU2 is invoked from the command line with arguments describing the board-id and hex filename to be downloaded. (e.g. MCU2 0x10 download-test.hex) The program proceeds as:
    1) The CANBus support .DLLs are loaded.
    2) The HEX file is opened.
    3) The HEX file is parsed and records interpreted:
        a) "Address" type records are not transmitted; they update address variables within the MCU2 program controlling the block-target-MCU protocol messages.
        b) "Data" type records are transmitted using the CANBus HLP3 block transfer protocol if they are destined for "allowed" addresses. In order to preserve the original interrupt vector table and first-image code, the memory areas allocated to those are NOT written. Only addresses 0x100 through 0x200 (Alternate vector table) and 0x4000 through 0x8FFF are allowed for download. Other addresses are skipped by MCU2.
        c) As blocks are completed, they are "written" by the MCU to the flash memory. Because the first-image low-memory area (interrupt vector tables) needs to be preserved, a special type of "erase" command, ERASE_PRESERVE is used in this memory region.
b. When the MCU2 program completes, files and .DLLs are closed.
c. The MCU2 program reports its operation to the .DOS window which may be saved for examination.

5. Starting the second image:
After downloading the second image, the first image is still operating. To begin execution of the second image, a CANBus HLP message sequence is issued:

a. board|WRITE; length 5; payload: 0x8D 0x69 0x96 0xa5 0x5a is issued.
b. This causes the MCU code to reply with: board|WRITE_REPLY; length 2; payload: 0x8D 0x0.
c. The MCU locks-out interrupts, sets the interrupt vector table selector to "alternate" and does a Jump to address 0x4000. The startup code CRT0 within the second image begins executing to initialize the C-language environment, followed by "main()" being invoked.

6. Restarting the first image:

a. From the second image: A call to the C-routines exit(); or reset(); or execution of the assembly instruction “reset” will cause a jump through the original reset vector (which was not changed during download) and cause the first image code to be restarted.

b. From within the first image: If the second image was not started: then issuing the CANBus HLP command board|WRITE length 5; payload: 0x8F 0x69 0x98 0xa5 0x5a will cause the equivalent of a power-on reset to take place which restarts the first image.

8.4. FPGA Reprogramming via CANBus

Refer to the *CANBus High-Level-Protocol* document for more details on the proper sequence of CANBus download commands.
The Altera FPGA (U1) on boards TDIG-D, TDIG-E, and TDIG-F can be reprogrammed and reconfigured via CANBus using a download procedure and FPGA reconfiguration commands as follows assuming the CANBus has been configured and operates properly:

1. The TDIG-D, TDIG-E, and TDIG-F board(s) must be running MCU code version 11C or later
2. Using Altera Quartus software, compile the new FPGA code. Be sure to use “compressed bitstreams” and “Generate RBF” file options in Quartus as shown below. The resulting .RBF file will be used to download to the auxiliary configuration EEPROM.

```
In Quartus under:
<Assignments>
  <Device>
    <Device & Pin Options>
      <Configuration tab>
        Enable (check) the box "Generate compressed bitstreams".
      <Programming Files tab>
        Enable (check) the box "Raw Binary File (.RBF)”
      <General tab>
        Enable (check) the boxes "Enable INIT_DONE output" and
          "Enable device-wide reset DEV_CLRn"
```

3. Close the PcanView utility. (Only one attachment to the PEAK CANBus DLL files is allowed).

4. From a command line interface (“DOS box”), invoke the download utility eeprom2.exe using the command line format:

```
eeprom2  <brd#> <filename.rbf>
```

Where: `eeprom2` is the name of the utility eeprom2.exe
`<brd#>` is the board number of the target board, set by the “board position switch(es)”.
`<filename.rbf>` is the name of the .RBF file generated in step 2.

This download procedure may take several minutes and generates numerous diagnostic and progress messages. If an abnormal condition occurs, the download process will stop and display an "error" message. When a successful download is complete a success message will appear. At that point it is safe to close the DOS window and return to the PcanView (or other CAN monitor/control) program.

5. At the conclusion of step 4, the FPGA is still running from the original (unmodifyable) configuration. The final step is to cause the FPGA to reconfigure using the newly downloaded image. This may be done using the PcanView program by issuing a Reconfiguration message (Write Command) to the MCU:

```
Message ID  Length  Payload Contents (hex):
 <brd#>2     5    8A   69   96   a5   5a
```

e.g. "`012 5 8A 69 96 a5 5a`" will cause reconfiguration from EEPROM #2 of the board ID'd 1.

The message reply will consist of the following message: `<brd#>3  3  8A  00 r7`

Where: `<brd#>3` indicates a "Write Reply" from <brd#>
`3` is the length of the reply
`8A` is the target of the write (this value will be `89` for reconfiguration from EEPROM#1)
`00` is the status of the write (`00` is normal return)
`r7` is the contents of the "ID Register-7" following reconfiguration.

6. Following reconfiguration, the FPGA is reset and the default register contents are loaded.

7. Operation from the EEPROM #1 configuration can be restored by sending a message with contents:

```
Message ID  Length  Payload Contents (hex):
 <brd#>2     5    8A   69   96   a5   5a
```

9. Quick-Start Guide

Power :

J26= +4.5V
J25 = GND

Jumper settings:

Jumper JU2: Short 1-2 = external oscillator (From ONE of J18; J19; J22/J23)

Open 1-2 = internal oscillator
Short 3-4 = PLL in the clock path
Open 3-4 = PLL Bypassed (not in clock path)

Programming headers and locations:

J7: Byteblaster JTAG programming of CPLD
J8: Byteblaster active serial programming of FPGA (EEPROM #1)
J6: Byteblaster JTAG debugging of FPGA
J10: ICD2 programming of MCU

J5 : Downstream cable 2
J4: Downstream cable 1

Firmware Identifiers:
In response to a CANBus command `204 1 B1`
The response will be `205 4 B1 41 02 78`

9.1. Can bus commands for initialization

NOTE: These commands are always useful if the TCPU readout is "stuck". These commands are issued automatically by the TCPU's MCU on power up, but the operational environment for an entire tray could cause problems. Command #1 gives a hardware reset to the FPGA, resetting all flip flops in the device. Command #2 resets the state machines, in particular the serial readout state machine. **Any debug process should always start by trying these commands!**
(assuming TCPU set to board # 0, and TDIG set to board # 4)

1. FPGA DEV_CLRn on TCPU (sends active low pulse to pin B3)

```
002        5  0C  69 96 a5 5a
```

2. FPGA soft reset signal to stage machines (this command shows how to send a pulse to a configuration bit: first a "01" value is sent to config register #1 on TCPU, then a "00" to the same register).

```
002    5    0e    01    01    01    00
```

(3.) Switch to cable 2 input routed to MCU CAN (Cable 1 is default).

```
002    3    0e    00    02
```

4. Reset TDCs (hardware pulse to "reset" pin on all 3 TDCs)

```
142    5    0e    02    01    02    00
```

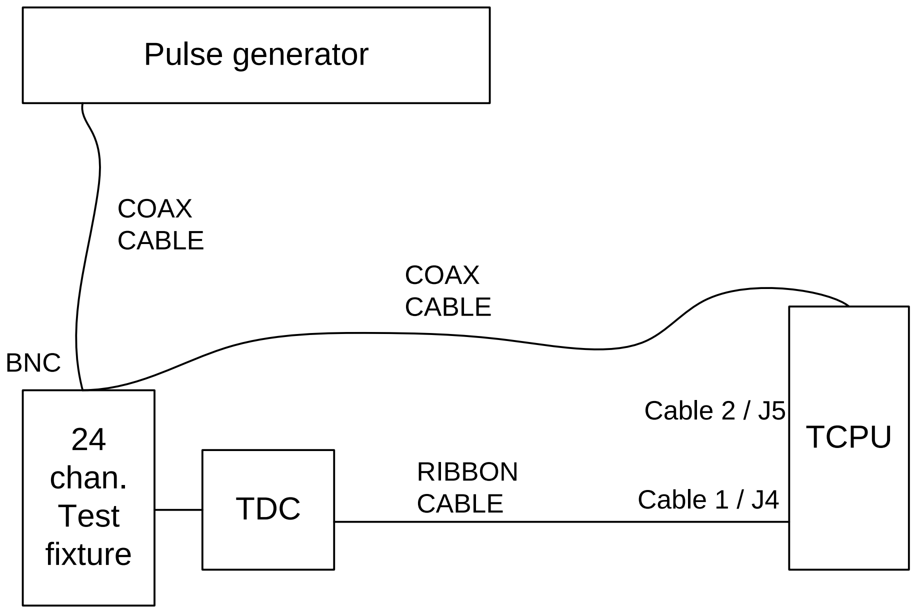
9.2. Reading Data to a File

1) stop pcan view program
2) open a dos window
3) run getdata.exe from /can drivers directory:
4) getdata node (1 to 8) #words filename

e.g.: “getdata 1 1000 data.dat”
Note: data source must be active or program will timeout

9.3. Bench Test #1

The following setup uses a pulse generator to send data pulses to Ted’s 24 channel fixture, and at the same time a “readout data” signal to TCPU:



The pulse generator is set up as follows:

Pulser ouput level is set to LVTTL (3.3V max amplitude, 0V min amplitude), 50 OHM termination.
Trigger rate X #TDC channels must be less than ~20K/sec to stay below 1 Mb/sec data rate over CAN bus. Also, loss of data can occur if the PC or the CAN bus

readout module experience buffer overflow, and this depends on how fast and how busy the PC is. **Tests show that a safe pulser rate = 100 Hz for reading out 1 TDC card, 50 Hz for 2 cards, etc. As part of the data analysis, use the periodic data pattern to make sure there is no loss of data.**

Operation:

1 CAN message from each board should be received on power up.

The initial reset commands are generated automatically by the MCU, so data messages should be visible on the CAN bus whenever the pulser begins operation.

The signal to TCPU initiates readout of one of the 2 ribbon cables (selected by configuration register bis config0.1).

During a readout cycle the TCPU issues a token to the first TDC on the first board in the readout chain, then reads data words until the token is returned from the last TDC on the last board in the readout chain.

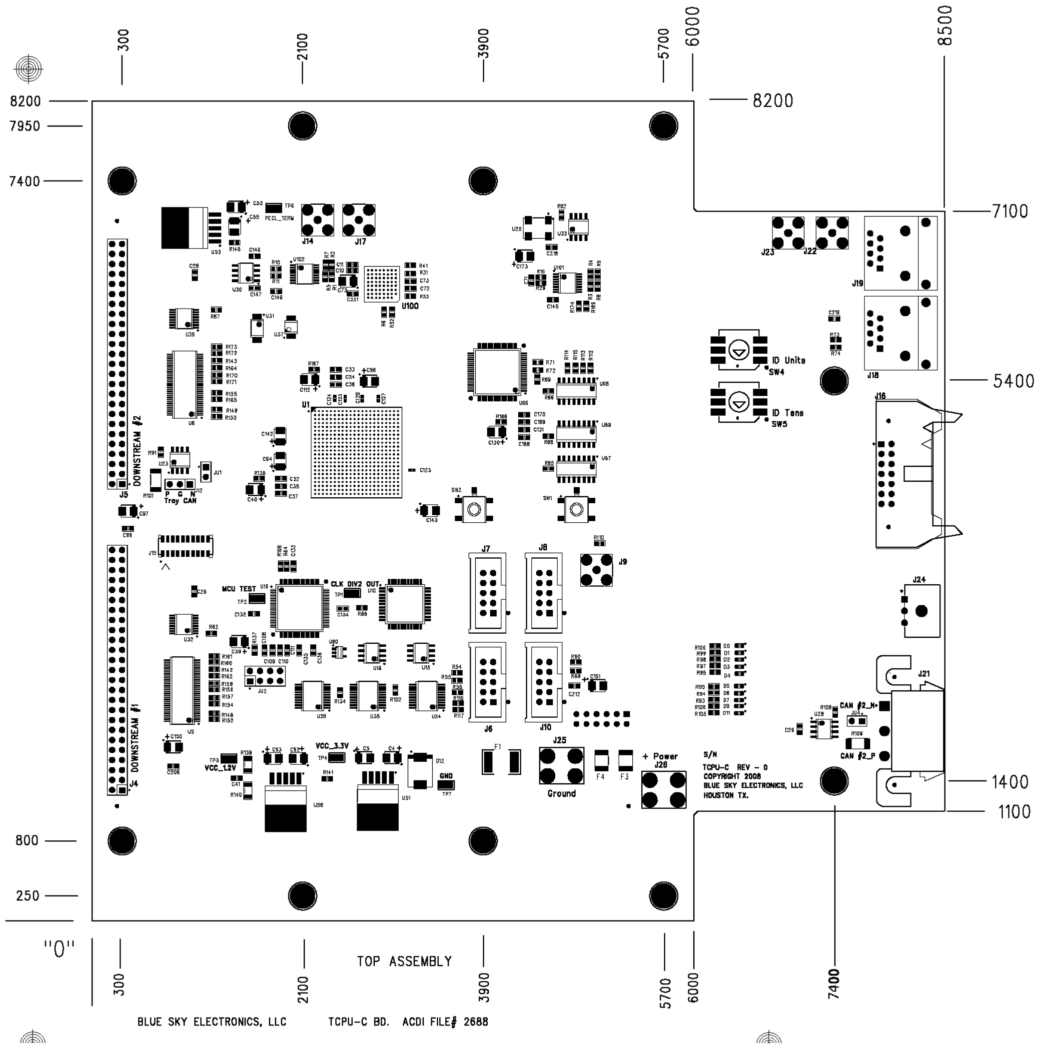
10. Mechanical Dimensions

10.1. PCB Layer Stackup and Specifications

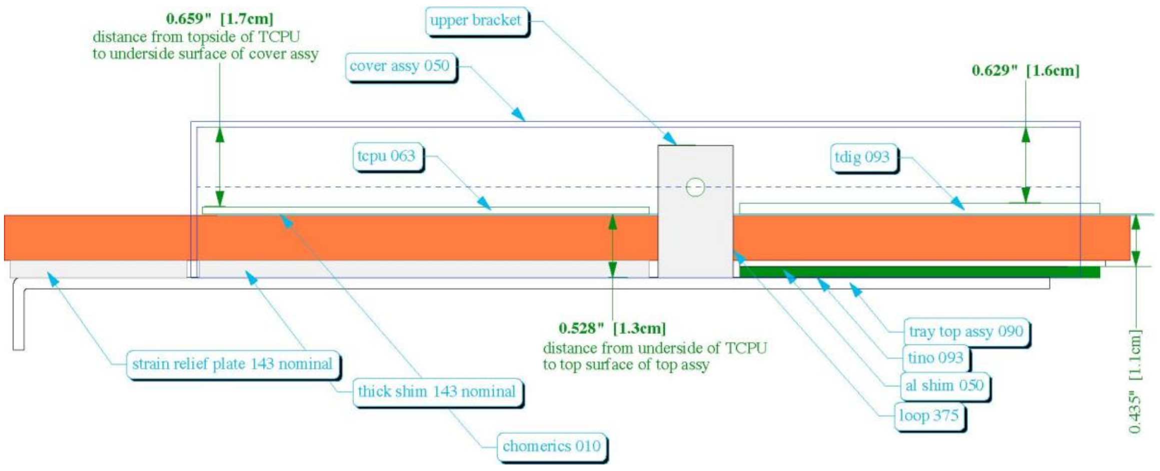
Overall board size: 8.2 x inches (Connectors J4, J5 overhang edge of board)
Board material: FR4 nominal thickness 0.093 inches 8 conductive layers.
Surface finish: Immersion Silver.
Layer Stackup (Primary-to-Secondary):

Conductive Layer		Laminate Layer
0.5 oz Copper primary	Over	3.45 mils FR-406
1 oz Copper	Over	6.0 mils FR-406
1 oz Copper	Over	7.55 mils FR-406
1 oz Copper	Over	47 mils FR-406
1 oz Copper	Over	7.55 mils FR-406
1 oz Copper	Over	6.0 mils FR-406
1 oz Copper	Over	3.45 mils FR-406
0.5 oz Copper secondary		

10.2. Tray integration dimensions:



Clearance from top of TCPU to underside surface of cover assembly = 0.629 inches.



Appendix A – High Speed Serial Messages

1. Messages From THUB transmitter to TCPU receiver

Messages from THUB to TCPU shall consist of 18-bit words.
The format for these THUB command words is shown in the table below:

Bit position	Field
17	Data type 1
16	Data type 0
15	Data ID 3
14	Data ID 2
13	Data ID 1
12	Type ID 0
11	Data 11
10	Data 10
9	Data 9
8	Data 8
7	Data 7
6	Data 6
5	Data 5
4	Data 4
3	Data 3
2	Data 2
1	Data 1
0	Data 0

Field Definitions:

1.1. Data Type [1:0]

Values may be:

[11] Valid data, odd parity
[10] Valid data, even parity
[01] Control word
[00] invalid data

Data ID [3:0] value definitions depend on Data type values:

1.2. Data Word (Data type = [11] or [10])

Data ID values may be:

[0000] = trigger token data, phase A (and issue trigger)
[0001] = trigger token data, phase B (and issue trigger)
[1100] = test data 0
[1101] = test data 1

1.3. Control (Command) Word (Data Type = [01])

Data ID values may be:

[0000] = go to test mode 0
[0001] = go to test mode 1
[0010] = issue bunch reset
[0011] = reset
[0100] = abort upload

1.4. Data [11:0]

This field is valid for data words only (Data ID = [11] or [10]) and otherwise should be set to 0.

2. Messages from TCPU Transmitter to THUB receiver

Messages from TCPU to THUB shall consist of groups of at least 3 18-bit words.

The first word(s) shall function as a synchronization word (invalid data) and may be repeated. This word shall have [00] in bit positions 17 and 16 to indicate invalid data.

Following the synchronization word(s) shall always be exactly 2 TCPU data words.

The format for these TCPU data words is shown in the table below:

First TCPU Data Word	
Bit position	Field
17	Data type 1
16	Data type 0
15	Data ID 3
14	Data ID 2
13	Data ID 1
12	Data ID 0
11	Data 27
10	Data 26
9	Data 25
8	Data 24
7	Data 23
6	Data 22
5	Data 21
4	Data 20

3	Data 19
2	Data 18
1	Data 17
0	Data 16

Second TCPU Data Word	
Bit position	Field
17	Data Type 1
16	Data Type 0
15	Data 15
14	Data 14
13	Data 13
12	Data 12
11	Data 11
10	Data 10
9	Data 9
8	Data 8
7	Data 7
6	Data 6
5	Data 5
4	Data 4
3	Data 3
2	Data 2
1	Data 1
0	Data 0

Field Definitions:

2.1. Data type [1:0]

Values may be

[11] valid data, odd parity
[10] valid data, even parity
[01] control word – TBD (to preserve symmetry)
[00] synchronizer word – data ID and data values are ignored

2.2. Data ID [3:0] (in First TCPU Data Word)

Values may be

[0000] Thru
[0111] = HPTDC data types

[1000] = status/error data

[1001] = multiplicity data
[1010] = trigger token return
[1011] = test data
[1100] = geographic data
[1101] = tag data (not used in TCPU-THUB communications)
[1110] = separator data

2.3. Data [23:0]

Values are formatted according to Data ID.

2.3.1. HPTDC Data for Data ID from 0 to 7

data bits are defined according to the HPTDC reference manual.

[Jo – please define the mapping of trigger token, geographic, tag, and separator data into the field DATA[27:0]]

2.3.2. Trigger token return (Data ID [1010])

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	1	0																	token											

Empty fields are reserved for higher level processing in the THUB and should be set to all 0's.

2.3.3. Geographic Data (Data ID [1100])

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0																					tray ID							H

where

"*tray ID"* is a number between 1 and 120, uniquely identifying each tray according to the document "STAR Geometry" (STAR Note 229),
"*H"* is either 0 or 1, identifying a *half tray* corresponding to 4 TDIG boards; TDIG boards 0 – 3 correspond to *H = 0,* while TDIG boards 4 – 7 correspond to *H = 1.*
Empty fields are reserved and should be set to all 0's

2.3.4. Separator Word (Data ID [1110])

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	0													number of items								trigger counter							

where:
"number of items" is the number of preceding HPTDC words from a readout chain.
"trigger counter" are the 8 least significant bits of an internal counter on TCPU that keeps track of how many trigger commands were received from THUB.

3. Example Communications Sequence

At the beginning, some time after startup, the first message will be a bunch reset message from THUB to all TCPUs:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	C	C	C	C	X	X	X	X	X	X	X	X	X	X	X	X

where:
DATA TYPE = 01 = CONTROL DATA
CCCC = DATA ID = bunch reset command
XXxx = 0000 0000 0000

The run can then start, and whenever THUB receives a trigger, it sends a trigger word to TCPU as follows:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DT1 = 1	DT0 = PARITY	C	C	C	C	T	T	T	T	T	T	T	T	T	T	T	T

where:
DATA TYPE = 10 | 11 = [11] valid data, odd parity OR [10] valid data, even parity

CCCC = 0000 | 0001 indicating phase
TTTT = 12 bit TCD Token

The TCPU then issues a trigger to all TDIG boards either in the 40Mhz phase that coincides with the incoming 20Mhz rising clock edge, or in the 40Mhz phase that coincides with the incoming 20Mhz trailing edge, according to the phase information contained in this packet.

[JO : PUT A TIMING DIAGRAM HERE – THIS IS UNCLEAR]

After issuing the trigger, the TCPU will attempt to readout the two serial readout chains from the tray, and then start sending data to THUB according to a sequence defined below, two 18 bit words at a time:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p	D31	D30	D29	D28	D27	D26	D25	D24	D23	D22	D21	D20	D19	D18	D17	D16

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	p	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

wh
ere:

1p = DATA TYPE
D[31:0] = data word bit
XX.. = don't care
00 = invalid data

[JO PLEASE SIMPLIFY THIS TO EXCLUDE MULTIPLICITY DATA, OR TO BE EXPLICIT ON PLACEHOLDER CONSTANTS TO BE SENT –

THEN, IF YOU LIKE, ADD A SPEC FOR THE FINAL MULTIPLICITY REQUIREMENT, PREFACED BY THE STATEMENT: THE IMPLEMENTATION OF SENDING LEVEL 2 MULTIPLICITY DATA IS ANTICIPATED FUTURE DEVELOPMENT AND IS NOT INCLUDED IN THE CURRENT DEVELOPMENT EFFORT WHICH WILL END IN THE FIRST HALF OF CALENDAR YEAR 2007]

The sequence of (32-bit) data words sent from TCPU to THUB is then defined as follows:

Word #:	Content:
1	Header Trigger Data (containing TCD Token)
2	Geographical Data for first half tray
3	HPTDC data words from 12 HPTDCs (TDIG 0 – 3)
…	…
3+n	HPTDC data words from 12 HPTDCs (TDIG 0 – 3)
3+n+1	TDIG separator word containing # of preceding HPTDC data
3+n+2	Geographical Data for second half tray
3+n+3	HPTDC data words from 12 HPTDCs (TDIG 4 – 7)
…	…
m	HPTDC data words from 12 HPTDCs (TDIG 4 -7)
m+1	TDIG separator word containing # of preceding HPTDC data
m+2	Multiplicity data bits 0 – 27

Word #:	Content:
m+3	Multiplicity data bits 28 – 56
m+4	Multiplicity data bits 56 – 85
m+5	Multiplicity data bits 86 – 113
m+6	Multiplicity data bits 114 – 141
m+7	Multiplicity data bits 142 – 169
m+8	Multiplicity data bits 170 – 192, left justified with 0's

The format of the Multiplicity data is simple: one bit per hit MRPC channel, as inferred from the HPTDC data. In version 0 of the firmware, the multiplicity data will not be sent by the TCPU and the data transmission terminates after word "m+1" in the above table. The sequence above shows where future versions of the firmware will insert this multiplicity information.

In case of errors or timeouts we will use "Control" packets (which I haven't defined, yet) instead of the data words shown above. Those "Control" packets could again be single 18 bit transmissions (instead of two 18 bit transmissions, as is the case for "Data"), where the upper 2 bits are "01". We can then use the same definition that we use for the THUB-TCPU communication, i.e.:

17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	C	C	C	C	A	A	A	A	A	A	A	A	A	A	A	A

where:
01 = "Control word"
CCCC = "Status" or "Error" code (e.g. "timeout", "no data", "PLD SEU", etc)
AAaaa = Whatever argument is needed to further specify the "Status" or "Error".

We could discuss what we do in case there is no data (but no error condition either) from a whole half tray. Do we still send the Geographical data and TDIG separator words? Or maybe just one of those? Or both with the # of preceding HPTDC data in the separator word set to 0?
Let's decide here that version 0 of the firmware sends all of the framing packets no matter if there are data or not for debugging purposes. We can then decide in the future to omit the framing information when there is no data, if so desired.

[JO – PLEASE INCLUDE SOME SORT OF STATE DIAGRAM OR OTHER WAY TO DESCRIBE THE HIGH LEVEL OPERATIONAL SCENARIO DEALING WITH UPLOAD ABORT COMMANDS, MULTIPLE TRIGGERS ETC. IT'S OK TO DESCRIBE MORE THAN ONE SCENARIO OR STUB FUTURE DEVELOPMENT SOMEHOW, BUT BE VERY CLEAR ABOUT WHAT THE FIRST VERSION WILL BE.]
Version 0 of the firmware will ignore any further THUB-TCPU communication while processing the initial trigger command from THUB, until all data for the corresponding event has been transferred to THUB according to the above indicated sequence. In the

future, we will define “Abort” control packets that will cause the TCPU to abort the current event and get ready for receipt of the next trigger. In future versions of the firmware we will also allow additional triggers to occur while the current event is being processed.

Appendix B: Example CAN bus commands

1. TCPU: Send DEV_CLRn pulse from MCU to FPGA (Pin B3)

Send hardware reset to the FPGA from the MCU

```
002     5   0C 69  96  A5 5A
```

2. TCPU: reset readout state machines in FPGA

Write twice to FPGA register 01, toggling bit 0 to reset readout state machine

Note that the default value for this register is 00, and the reset signal from TCPU FPGA configruation register 0, bit 0 "config_0.0" is active high.

```
002     5   0E 01  01  01  00
```

(reply is 003 0E 00)

3. Read firmware versions:

TCPU #0:

send 004 1 B1

reply 005 B1 47 01 71 (low byte MCU ver | high byte MCU ver | FPGA ver)

TDIG #0:

send 104 1 B1

reply 105 B1 0F 11 61

TDIG #1

send 114 1 B1

reply 115 B1 0F 11 61

4.TCPU: switch to reading from Aux data path

(set config_14.7 = high)

```
002     3  0E 0E 80
```

Appendix C: ATP notes

1. Multiplicity output test:

Select test mode for data mux: config_0.1 = 1
Write to config_9.[4:0] with multiplicity value
Look at multiplicity value on differential pairs at J16, using Ted's fixture

2. Trigger clock input:

Stanford ab inverted output to Ted's fixture with white BNC cable
Terminated 50 ohm, amplitude = +0.5V, offset = -0.5V

Appendix D: Programming and Testing TCPU Boards

Jun 26, 2009

Note: **Bold** lines are repeated operations once programs are set-up.

1. **Apply serial number label to board.**

2. **Connect power to board (4.5VDC nominal).**

3. Use Quartus programmer (one instance) to program CPLD.
    - **a.** From within Quartus Programmer open chain file for the CPLD; use <file><open> navigate to directory *C:/WDB/TCPU_B_C/TCPU_C/CPLD-FPGA_Code;* select file *cpld.cdf*; click <open>. This sets up the byteblaster for:
        - i. "JTAG" programming
        - ii. File *C:/WDB/TDIG_D_E_F/Altera/SmallPLD/TDIG-D_U10_ver1b.pof* (CPLD files for TDIG and TCPU are the same)
        - iii. Checksum 000E0A85
        - iv. Verify that options "Program/Configure" and "Verify" are checked.
    - **b. Connect the Byteblaster cable to TCPU board J7 (upper left connector).**
    - **c. Click <Start>; this will program the CPLD (takes only a few seconds).**

4. Use Quartus programmer (second instance) to program FPGA.
    - **a.** From within Quartus Programmer open chain file for the FPGA; use <file><open> navigate to directory *C:/WDB/TCPU_B_C/TCPU_C/CPLD-FPGA_Code;* select file fpga*.cdf*; click <open>. This sets up the byteblaster for:
        - i. "Active Serial" programming
        - ii. File *C:/WDB/TCPU_B_C/ TCPU_C/CPLD_FPGA_Code/TestCode/TCPU_C_TOP.pof*
        - iii. Checksum 05D0F294
        - iv. Verify that options "Program/Configure" and "Verify" are checked in two places each.
    - **b. Connect the Byteblaster cable to TCPU board J8 (upper right connector).**
    - **c. Click <Start>; this will program the FPGA (Initially it does not *appear* to be working for a few seconds during the initial "erase" time, this is normal, the whole programming process takes quite a while).**

5. Use Microchip MPLAB-IDE to program the MCU.
    - **a.** Start the MPLAB IDE.
    - **b.** Select <Programmer> and <MPLAB-ICD2>. A warning message will appear if the ICD2 is not connected to the board or if the board is not powered; this is OK and the warning can be dismissed.
    - **c.** Select <File> <import> and navigate to file: *C:/WDB/TCPU_B_C/TCPU_C/MCU-C_Code/TCPU_Ver2/TCPU-C_2.hex* (use version _2H 02/02/09)
    - **d.** Select <open>
    - **e.** Object file will be loaded. Observe checksum 0xCC88
    - **f. Connect MPLAB programming cable (custom cable for TDIG and TCPU) connected to TCPU J10 connector (Lower right).**
    - **g. Select <programmer> <program> or click the yellow page->chip icon. This programs MCU code.**

6. **Turn off power to board and remove programming connectors.** Board should be ready for testing.

7. **Make test connections as shown in the diagram:**

- **a.** Hewlett-Packard Frequency Counter input connected to TDIG Board 0 test module CLKI pin.
- **b. Ribbon cable from J4 (lower connector) to TDIG Board 0 (bottom of stack).**
- **c. Ribbon cable from J5 (upper connector) to TDIG Board 4 (top of stack).**
- **d. CANBus#2 (white connector) from J21 to Peak CAN Module.**
- **e. 2-wire voltage sense connector to J24.**
- **f. 16-pin ribbon connector to J16 multiplicity connector from "TCPU-C J16" test module and National Instruments NI-USB-6008 module. NI-USB-6008 module must be connected to USB port on computer**
- **g. Insure that jumpers are present at JU1 and JU4 (CANBus terminators).**
- **h. Jumpers JU2 1-2 and 3-4 should be OPEN (no jumper).**
- **i. Insure TCPU Board Position Switches are set to "0".**
- **j. Connect and apply power to board (4.5VDC nominal).**

8. Start National Instruments LabView and open the LabView file: *BillJ16Test.vi* but do not run it yet.

9. Start National Instruments Measurement & Automation Explorer (DAQMx).
   - **a.** Select <My System/Data Neighborhood> <NI DAQMx Tasks> and <Diff4Chans> but do not run it yet.
   - **b.** Arrange NI windows so that both the LabView Front Panel and DAQMx windows are visible.

10. Open the directory containing TestTCPU.exe, resize the window if necessary.

11. Open a DOSbox window in the TestTCPU directory.

12. **Run TestTCPU.exe** from the command line by typing **TestTCPU <sn>** where <sn> is the serial number of the board being tested.

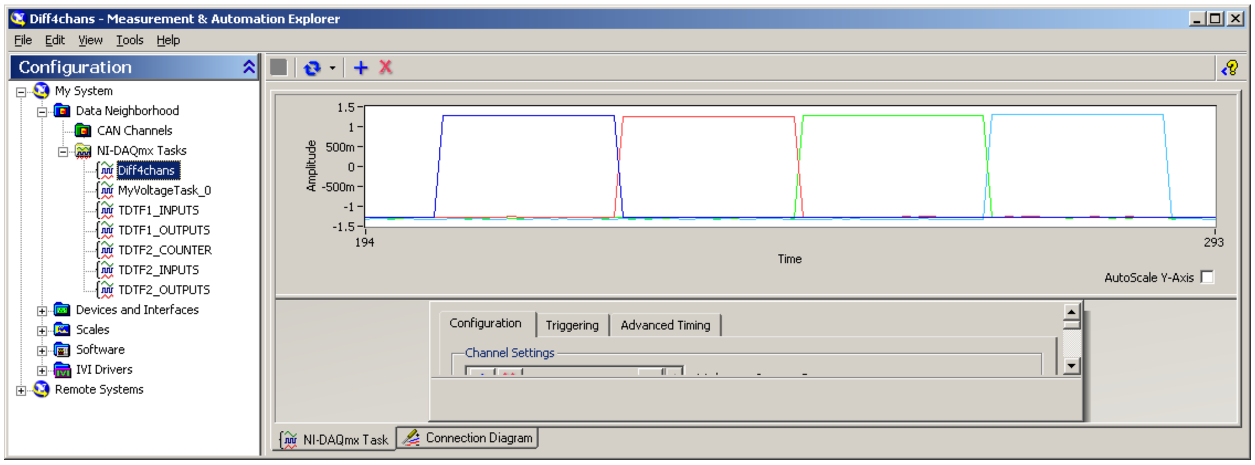
    - **a.** TestTCPU performs various tests of the TCPU board under test by sending a series of CANBus messages and observing CANBus replies.
    - **b.** The status of the tests is displayed in the DOS window and is placed in a text file *TCPUxxxx.LOG* where xxxx is the 4 digit board serial number.
    - **c.** The board serial number and "hardware" serial number are also appended to a "master" serial number file TCPU_HDW_SERIAL.LOG

13. **Print** a copy of the *TCPUxxxx.LOG* file by right-clicking the file-name and selecting <print>. The hard-copy will be used for manual notation of the results of the following tests.

14. **Run the Automation Explorer** by clicking the blue circular arrows. Observe data scrolling in chart window.

15. **Run the LabView application *BillJ16Test*** by clicking the white right-pointing arrow. The Test application sends a series of CANBus commands via the NI CAN USB Module which set and clear bits on the multiplicity connector.

    - **a.** The DAQMx task reads and displays 4 differential output voltages (the 5th bit was tested during execution of TestTCPU).
    - **b.** UNcheck the "Autoscale Y-Axis" while the differential values are being acquired.
    - **c.** When the LabView task *BillJ16test.vi* finishes, **Stop the DAQMx task** by clicking the red X.
    - **d. The DAQMx chart should look like the following** ("Time" values may be different" but the overall shape of the waveforms should look similar:



i. Four successive “pulses” should appear (Blue, Red, Green, and Cyan), one for each of the tested bits.

ii. Transitions should be from -1.3VDC to +1.3VDC (nominal). Low amplitude, mis-shapen, or missing pulses are indication of faulty Multiplicity outputs. Slight “noise” appearing on the high and/or low levels is acceptable.

e. **Note** satisfactory/failed completion of this test on the hardcopy .LOG file on line 18a “J16 Out Levels:”

16. **Voltage Tests.**
   a. **Connect** voltmeter ground to convenient grounding point (TP7, mounting hole, J24 pin 2, etc).
   b. **Observe and note** on log sheet the DC voltage present at the right-hand terminal of fuse F1 (Vin).
   c. **Observe and note** on log sheet the DC voltage present at testpoint TP6 (PECL-TERM).
   d. **Observe and note** on log sheet the DC voltage present at the yellow wire connected to J24 (J24 pin 1).

17. **Internal/External/PLL clocking tests.** The TCPU clock is passed through the ribbon cable to the downstream connectors J4 and J5 and is received by the HP Frequency Counter.
   a. With **JU2 pins 1-2 and 3-4 left open**, the TCPU is configured to operate on its Internal clock without PLL processing. **Observe 40MHz** (nominal) displayed on the Frequency Counter and **note success/failure** on Log sheet item 13c “Int.Osc+Bypass”.
   b. **Turn power to TCPU OFF.**
   c. **Place jumper on JU2 pins 3-4**, This configures TCPU for Internal clock ***with PLL*** processing.
   d. **Turn power to TCPU ON**.
   e. **Observe 40MHz** displayed on the Frequency Counter (it may take a few seconds for the PLL to stabilize). **Note success/failure** on Log sheet item 13c “Int.Osc+PLL”.
   f. **Turn power to TCPU OFF.**
   g. **Place jumper on JU2 pins 1-2**, This configures TCPU for ***External*** clock *with PLL* processing.
   h. **Connect External Clock** from TDIG Board 4 (J14) to TCPU board J23.
   i. **Turn power to TCPU ON**.
   j. **Observe 40MHz** displayed on the Frequency Counter (it may take a few seconds for the PLL to stabilize). **Note success/failure** on Log sheet item 13c “Ext.Osc+PLL”.
   k. **Turn power to TCPU OFF.**
   l. **REMOVE jumper From JU2 pins 3-4**, This configures TCPU for *External* clock ***Without*** *PLL* processing. Leave External clock input connected.

m. **Turn power to TCPU ON**.
n. **Observe 40MHz** displayed on the Frequency Counter. **Note success/failure** on Log sheet item 13c "Ext.Osc+Bypass".
o. **Turn power to TCPU OFF.**

18. Tests are complete. **Note overall success/failure** on "Result" line.

Change history

Version 6_0a: Added modifications for BNL boards FPGA registers (JS)
Version 6_x : September 2009 Major restructuring and additions for update to TCPU-C rev 1 hardware.
Version 5_0 : May 19, 2008

Version 4_4
- Changed TDC numbering

Version 4_2 : Sept 29, 2007
- New information on Aux data path usage.
- Additional examples of CAN bus messages as Appendix C.
- Included formerly separate firmware programming instructions (using programming headers) as Appendix B.

Version 4_1: Sept 24, 2007
- Updated configuration register table.
- Added "useful FPGA configuration bits" section, with an initial description of bunch reset signal.