DYNAMIC ENGINEERING
150 DuBois St. Suite C Santa Cruz CA 95060
831-457-8891 **Fax** 831-457-4793
http://www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

CPCI Receiver Controller Model CSS1

cPCI 3U 4HP

Revision B
Corresponding Hardware: Revision A
Fab number10-2008-0101

cPCI Receiver Controller
3U 4HP

Dynamic Engineering
150 DuBois St Suite C
Santa Cruz, CA 95060
831-457-8891
831-457-4793 FAX

Table of Contents

PRODUCT DESCRIPTION 6

Special features: 6

ADDRESS MAP 12

BIT MAPS 13

cPCI_rcvrcntl_BASE 13

cPCI_rcvrcntl_ID 16

cPCI_rcvrcntl_DAC 16

cPCI_rcvrcntl_CHK 18

cPCI_rcvrcntl_CHKref 19

cPCI_rcvrcntl_CHKtst 19

cPCI_rcvrcntl_RSL 20

cPCI_rcvrcntl_RSU 20

cPCI_rcvrcntl_RSC 20

cPCI_rcvrcntl_CSL 22

cPCI_rcvrcntl_CSU 22

cPCI_rcvrcntl_CSC 22

cPCI_rcvrcntl_TMP 23

cPCI_rcvrcntl_ADCC 25

cPCI_rcvrcntl_ADCD 25

cPCI_rcvrcntl_UARTC 27

cPCI_rcvrcntl_UARTrx 29

cPCI_rcvrcntl_UARTtx 29

cPCI_rcvrcntl_UARTst 29

cPCI_rcvrcntl_UARTaf 30

cPCI_rcvrcntl_UARTae 31

cPCI_rcvrcntl_TOAL 32

cPCI_rcvrcntl_TOAU 32

cPCI_rcvrcntl_TOAC 33

cPCI_rcvrcntl_TOAC1 33

cPCI_rcvrcntl_TOAC2 33

cPCI_rcvrcntl_TOAC3 33

cPCI_rcvrcntl_AASC 34

cPCI_rcvrcntl_AASLD 35

cPCI_rcvrcntl_COS 35

cPCI_rcvrcntl_TOAFL 37

cPCI_rcvrcntl_TOAFU 37

APPLICATIONS GUIDE 42

Interfacing 42

Construction and Reliability 43

Thermal Considerations 43

Warranty and Repair 44

Service Policy 44
Out of Warranty Repairs 44

For Service Contact: 44

SPECIFICATIONS 45

ORDER INFORMATION 47

List of Figures

FIGURE 1	CPCI RECEIVER CONTROLLER RS BLOCK DIAGRAM	7
FIGURE 2	CPCI RECEIVER CONTROLLER RS J5 PIN ASSIGNMENT	38
FIGURE 3	CPCI CALIBRATION SOURCE CS J6 PIN ASSIGNMENT	39
FIGURE 4	CPCI RC RF/IF J7,J8,J9,J10 PIN ASSIGNMENT	40
FIGURE 5	CPCI RC FP CLK J3 PIN ASSIGNMENT	40
FIGURE 6	CPCI RC RP J2 PIN ASSIGNMENT	41

Product Description

cPCI Receiver Controller is part of the Dynamic Engineering cPCI compatible family of modular I/O components.

The cPCI Receiver Controller is a 3U 4HP design optimized for the control of a specific digital receiver. Multiple SPI buses, A/D, D/A, clock references and special purpose IO are provided to support the receiver.

The PCI interface is implemented within a Spartan 3 FPGA. The control logic for the receiver and related interface hardware is also done within the FPGA.

Special features:

- Universal cPCI 3U 4HP[1].
- SPI for Receiver Synthesizer “RS”
- SPI for Calibration Source “CS”
- SPI for A/D – 8 channel A/D on board
- SPI for D/A – 8 channel D/A on board plus inverting buffers for attenuation control, 2 per RF/IF Module connector
- SPI for TMP123 temperature sensor located on board
- 5V power from cPCI used to generated 1.2, 3.3, 2.5, -5, +15, -15, +24
- Fused ±15, ±5, +24 supplied to RS and CS connectors
- 125 MHz clock reference LVDS
- 10 MHz clock reference LVTTL SMA x2
- UART function with programmable baud, parity, stop bits , 1023 deep FIFO, Differential IO.
- DIP Switch for positive board identification
- PLL – 22393 programmed with local serial bus, two clocks connected to FPGA
- 7 – RS485 transceivers used for differential IO functions
- 1 – LVDS transceiver used for 125 MHz clock reference out.
- 18 – LVTTL buffered signals – (24 total some duplicated)

[1] Most connectors oriented for internal cabling between the Receiver Controller and the next slice of the Digital Receiver. When cables attached the effective height will be more than 4 HP.

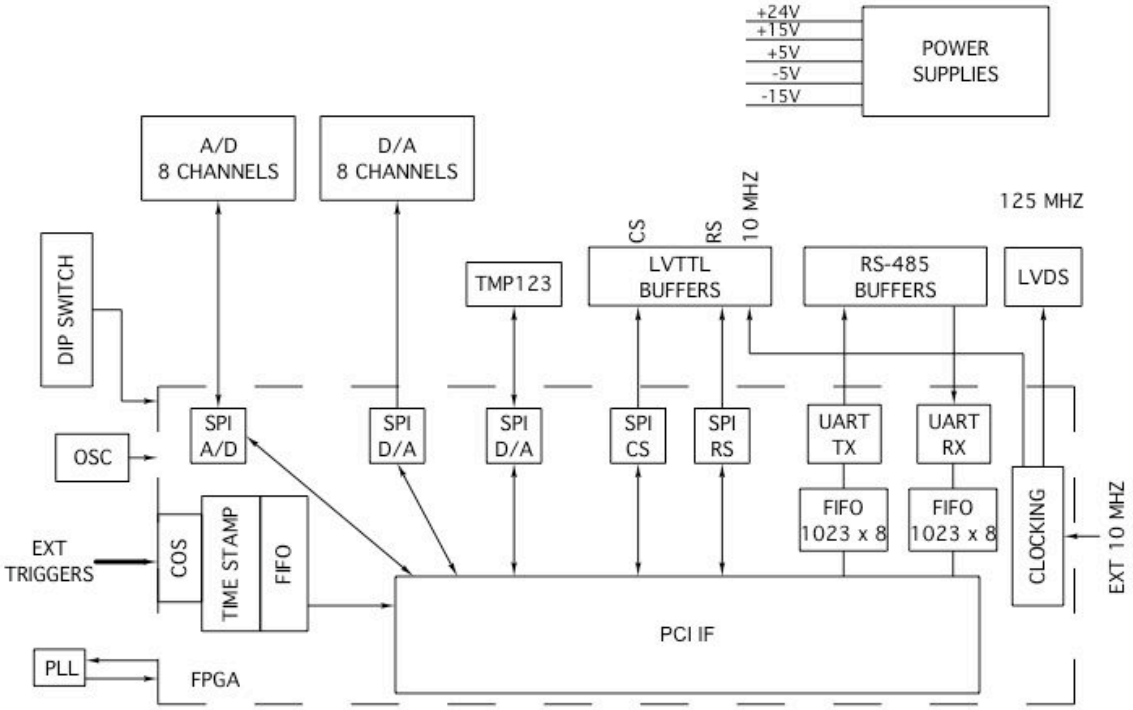

FIGURE 1 CPCI RECEIVER CONTROLLER RS BLOCK DIAGRAM

The Receiver Controller has many features and many details to cover. An outline of the features and basic capabilities is provided in this section of the manual. The detailed programming and use information is in the next section.

The diagram is simplified. Most of the interfaces shown have control and data registers. The register map is direct.

In addition to the information within this manual the PLL programming guide for the Cypress 22393 may be helpful. The reference software has a routine to program the PLL and read back the programmed results. The software is provided as C source code as part of the engineering kit. The C can be converted to other OS. The Cypress PLL programming pre-processor software is used to determine the programming stream to send to the PLL for the desired frequency. The Cypress program is available as a free download from the Cypress site.

A table is provided later in this manual with the data to send for the “standard”

programming of 18.432 MHz. The output of the PLL is used to run the UART channels for the base design. The second PLL channel is spare for the base design. If you are changing the 18.432 MHz output to a new frequency it will be important to know that the reference rate is the same as the Oscillator rate of 50 MHz. By shifting the 18.432 to a new value additional “non standard” baud rates can be created.

The UARTs are run using the 18.432 MHz clock divided down to the baud rate or 16X the Baud rate for the receiver side. The RS-485 transceivers are rated to 40 MHz. The upper limit on the PLL is 200 MHz. 200/16 => 12.5 MHz would be the upper limit for the RX side baud rate. The FPGA was not constrained for this frequency. On a different design with the same VHDL implementation [for the UART] 10 MHz was tested. If your design requires rates approaching the maximum please contact us to test the higher rates for you.

Each UART has an associated FIFO – 1023 x 8 – to store characters for transmission or reception. The Transmit UART will send characters when enabled as long as data is in the FIFO. The baud rate, parity, stop bits are programmable. The RX UART shares the definitions for parity etc. and has a separate enable. When enabled and characters are received the data is stored into the FIFO.

The design supports DMA within the PCI interface. DMA is not implemented at this time. With a relatively small amount of work the FIFO’s can be tied into the DMA arbitration engine to move data automatically within the programmed DMA transfer.

The Receiver Controller has an 8 position DIP Switch installed. The DIP Switch is not used for any feature selection. The DIP Switch is available for software defined purposes. A suggested use is for differentiation of multiple Receiver Controller boards within the system.

In addition to the PLL and oscillator several DCM’s are used within the FPGA to create the base frequencies needed to support the SPI buses and other clock requirements. The FPGA receives 50 MHz from the oscillator. The clock input is doubled to 100 MHz and then divided to 20 MHz. The 20 MHz is further divided to create 10 MHz and 2 MHz. The 10 MHz is mux’d with the external 10 MHz under software control. The selected 10 MHz is used to generate 125 MHz and the two 10 MHz references driven to the SMA connectors.

The 125 MHz signal is used internally to run the Time Stamp function and is converted to LVDS for external use.

The internally generated 10 MHz is supplied to a counter in parallel with a second counter referenced to the external 10 MHz. The software can reset, start, and set the end count for the internally referenced counter. For example the software can set a count corresponding to 1 mS of time counting. At the end of the time the status will indicate that the test is complete. The reference counter can be read back and compared with the external clock version. The two should match within a few counts if the external clock is valid. The software can then choose to use the external reference for the 125 and external 10 MHz clocks.

The 125 MHz is used to run an interesting counter. The counter is 33 bits wide and made up of a lower and an upper section. The counter is used to track Time of Arrival. The lowest 5 bits are set to count 0-24 for a divide by 25 function. The upper bits [28] count when the lower bits reach 24 also using the 125 MHz clock. The upper and lower halves can be preset, started and stopped.

The count is captured into a FIFO when 1 or more of 8 programmable conditions occur. There are three external signals [blanking1, blanking 2, 1 PPS] and 1 internal signal [Antenna Array Strobe (AAS)] which can be programmed to capture on the rising, falling or both edges [Change of State (COS)]. The source of the trigger is appended to the TOA data and stored into the FIFO. The FIFO is larger than the PCI bus width making two transfers required. The data is read from the FIFO and stored into two registers before being buffered onto the PCI bus. Data integrity is protected with a state-machine that moves the data from the FIFO to the registers based on the order the reads. Please see the SW section for more information.

An octal A/D is provided with an SPI bus to manage it. The 14 bit A/D uses a 4.096 V reference to measure the received signals. The A/D is programmed and read serially. The A/D function is started by writing to a register with the A/D programming information. The hardware converts the stored data to the SPI format and sends it to the A/D at a 2 MHz rate. The hardware then recovers the converted data and stores that into another register. The status is changed from busy to valid.

Channel 0 and 2 are set with voltage dividers to allow higher voltage inputs to be used. The divider is approximately 4.45 to allow 18V signals to be measured. Channel 1 and 3 are the return references for channels 0 and 2. Differential measurements can be made using 0,1 and 2,3 as pairs. Alternatively single ended signals can be measured and channels 1 and 3 used for additional inputs.

Channels 4,5,6,7 are also set-up to be used as differential measurements. Channels 4 and 6 are pulled to 5V via 10K ohms. Channels 5 and 7 are tied

through 10KΩ to ground. The system will add a third 10KΩ resistor in parallel to each differential pair. If the system cables are in place the 5V will be divided in 3 and roughly 1/3 will be measured across the pair. If the system cable is not in place a 5V reading will happen. If the system cable is in place and the tested condition [flooding] happens the pair will be shorted together creating a “0” reading on the differential pair. The on-board resistors can be changed for value or installation to provide other options. The lines can be used as single ended if desired.

The Maxim 5593 10 bit DAC is used for the 8 channels of D/A conversion. An SPI interface converts the data written to the control register to SPI format and sends it to the DAC. The DAC outputs a voltage corresponding to the programmed value. The system requires a voltage range of 0 to –3V. Inverting buffers are used to change the range of the signal. Each of the outputs is coupled through a 100Ω resistor. Two outputs are tied to each of the 4 RF/IF module connectors.

In addition to the data bits there are 4 control bits to select the mode of the programming. Several features built into the 5593 are interesting. Output on one channel, output to multiple channels, etc.

TMP123 is a 1.5/2C accurate temperature sensor in a small surface mount package. TMP123 is mounted near the FPGA toward the outside edge of the Receiver Controller. The hardware reads the temperature value from the sensor and stores into a register when tasked by the CPU. The TMP123 is tasked by sending a command with the SPI bus. After the conversion delay the value is read back. The temperature is returned as a 12bit plus sign value. Hardware strips the 0 padding and shifts down to a standard LSB aligned value. Conversions can be repeated every .5S if desired.

The Calibration Source and Receiver Synthesizer have identical and separate SPI interfaces. The SPI interface is a 2 bit parallel programmable length interface with a 20 MHz bit rate. Data is stored into two registers for the upper and lower values. Data is sent 1-32 bits at a time from each register [2-64 overall]. The data is buffered with ±24 mA LVTTL 3.3V buffers. Clock is low between transfers. Data stream 1 is valid on the rising edge of the clock and Data stream 2 is valid on the falling edge of the clock. The enable is asserted before the first clock and held after the last clock.

The system interfaces need small amounts of power supplied at voltages not supported by cPCI. The Receiver Controller board generates 24, ±15, and –5V to support these requirements plus provides them fuse protected as well as fused

5V. “Self healing” fuses are used. The FPGA also requires some non cPCI voltages to operate – 1.2 and 2.5. Switching power supplies are used for the generated voltages with the exception of the 3.3V rail that is supported with a linear regulator also from 5V.

Address Map

Name	Offset	Function
cPCI_rcvrcntl_BASE	0x0000 // 0	base control register offset
cPCI_rcvrcntl_ID	0x0004 // 1	ID Register offset
cPCI_rcvrcntl_DAC	0x0008 // 2	DAC Register offset
cPCI_rcvrcntl_CHK	0x000c // 3	control for 10 MHz clock check
cPCI_rcvrcntl_RSL	0x0010 // 4 - RS data 31-0	
cPCI_rcvrcntl_RSU	0x0014 // 5 - RS data 63-32	
cPCI_rcvrcntl_RSC	0x0018 // 6 - RS control	
cPCI_rcvrcntl_CSL	0x0020 // 8 - CS data 31-0	
cPCI_rcvrcntl_CSU	0x0024 // 9 - CS data 63-32	
cPCI_rcvrcntl_CSC	0x0028 // 10- CS control	
cPCI_rcvrcntl_TMP	0x0030 // 12	TMP123 port
cPCI_rcvrcntl_ADCC	0x0038 // 14	ADC Control Register offset
cPCI_rcvrcntl_ADCD	0x003C // 15	ADC Data and Status offset
cPCI_rcvrcntl_UARTC	0x0040 // 16 - UART Control register	
cPCI_rcvrcntl_UARTrx	0x0044 // 17 - UART FIFO RX read	
cPCI_rcvrcntl_UARTtx	0x0048 // 18 - UART FIFO TX Write	
cPCI_rcvrcntl_UARTst	0x004C // 19 - UART FIFO Status	
cPCI_rcvrcntl_CHKref	0x0050 // 20	reference clock count read-back
cPCI_rcvrcntl_TOAL	0x005C // 23 -	Low Side[4-0]
cPCI_rcvrcntl_TOAU	0x0060 // 24 -	high side 27-0
cPCI_rcvrcntl_TOAC	0x0064 // 25 -	toa cnt 27-0 [upper side]
cPCI_rcvrcntl_TOAC1	0x0068 // 26 -	toa cnt1 27-0 [upper side]
cPCI_rcvrcntl_TOAC2	0x006C // 27 -	toa cnt2 27-0 [upper side]
cPCI_rcvrcntl_TOAC3	0x0070 // 28 -	toa cnt3 27-0 [upper side]
cPCI_rcvrcntl_AASC	0x007C // 31 -	aas 15-0 control port
cPCI_rcvrcntl_AASLD	0x0080 // 32 -	aas 31-0 down count reference
cPCI_rcvrcntl_COS	0x0088 // 34 -	cos cntl 15-0
cPCI_rcvrcntl_TOAFL	0x008C // 35 -	toa FIFO lower [31-0]
cPCI_rcvrcntl_TOAFU	0x0090 // 36 -	toa FIFO Upper [40-32 + FIFO status MT, full, Valid]
cPCI_rcvrcntl_UARTaf	0x00A0 // 40	UART FIFO AFL level
cPCI_rcvrcntl_UARTae	0x00A4 // 41	UART FIFO AMT level
cPCI_rcvrcntl_CHKtst	0x00A8 // 42	clock under test count read-back

Bit Maps

cPCI_rcvrcntl_BASE

0x0000 // 0 base control register offset

Bits	Function
19	SDAT
18	S2
17	SCLK
16	PLL_EN
15–6	unused
5	F_INT
4	M_INT_EN
3-2	unused
1	SPARE2
0	SPARE1

SDAT, S2, SCLK, PLL_EN are used to program the PLL with the bit pattern generated with the Cypress PLL programming tool. Please note that the read-back of the SDAT bit is the data from the PLL not this register. Address offset 1 has the register data.

Programming the PLL requires a sequence of simple steps that seem complex when taken all at once. Taken separately it is reasonable. The engineering kit has C code for the programming of a file into the PLL.

Step 1: use the Cypress PLL programming tool to create a file set. 50 MHz is the reference.
Step 2: massage the data within the file to look like the following.

File for “standard 18.432 frequency output on PLL clock A
```
UCHAR  dat0[]   = {0x08,
                  0x12, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x50,
                  0x55, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
                  0x40,
                  0x69, 0x60, 0x61, 0x69, 0x60, 0x61, 0x61, 0x60,
                  0x61, 0x69, 0x60, 0x61, 0x00, 0x00, 0x00, 0x00,
                  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

The file produced by the Cypress compiler looks like:

Generated by CyClocksRT R3.10.00

Modification Date: 3/5/2008
Comments:
Customer:
FAE:
License #: bm0kd1gd1-ns6ps0cc2

Don't modify file contents after this line:
< Checksum: 20B37826
s CY22393

f 50.000000 ;External Ref
f   0.000000 ;Pll3
f   0.000000 ;Pll2
f 331.776000 ;Pll1, 0x
f 331.776000 ;Pll1, 1
f 331.776000 ;Pll1, 2
f 331.776000 ;Pll1, 3
f   0.000000 ;Pll1, 4
f   0.000000 ;Pll1, 5
f   0.000000 ;Pll1, 6
f   0.000000 ;Pll1, 7 >

*
QP0008* QF2048* G0*
L00064
0001 0010 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0001 0101 0000 0101 0101 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0001 0000 0000 0000 0000 1110 1001 0000 1000*

L00512
0110 1001 0110 0000 0110 0001
0110 1001 0110 0000 0110 0001
0110 1001 0110 0000 0110 0001
0110 1001 0110 0000 0110 0001
0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000*

C072B*
0000

To convert the .JED file from the Cypress tool to something compatible with programming the PLL use the 18.432 converted file as a model. Keep the “08” and “40” replace the second and third line with hex converted data fields. “0001 0010” become “12” etc. The first 16 bytes are used. Replace the next data block using the data from the second block after the “L00512”. 24 bytes are used this time.

Our software communicates with the PLL using a software clock the enable and data lines. Data is read from the array and shifted out to the device. The data is later read back to compare against the array and make sure proper communications are established. Details for programming the CY22393 can be found in the Cypress data sheet or in the reference software in the engineering kit.

SPARE1 and SPARE2 are LVTTL outputs tied to the J2 [rear cPCI] connector. The bits are uncommitted in the CCS1 version. The output follows the input definition.

M_INT_EN is the board level Master Interrupt Enable. When ‘1’ interrupts can be generated by the board. When ‘0’ interrupts are masked off.

F_INT is the Force Interrupt bit. When set and the master is enabled the design will cause an interrupt request to the system. Clear by setting to ‘0’. Default is ‘0’. Main purpose is to allow software programmed interrupts for testing purposes.

cPCI_rcvrcntl_ID

0x0004 // 1 ID Register offset

Bits	Function
16	Interrupt
15-8	Xilinx Revision
7-0	DIP Switch

The Xilinx Revision is currently set to 0x03. The revision will be updated as new features are added to the CCS1 version to allow software differentiation. The Xilinx revision is also visible from the configuration space within the PCI status.

The DIP Switch is read directly from this port. The value read matches the settings on the switch. The Switch is defined on the silkscreen for bit and 1,0.

Interrupt is the combination of potential masked interrupt requests from the card. When set '1' an interrupt request is pending from the card. When '0' no interrupt is pending. Interrupt is used for determining the source of a request in a shared interrupt environment.

cPCI_rcvrcntl_DAC

0x0008 // 2 DAC Register offset

Bits	Function
16	DAC STAT
15-0	DAC Data

DAC STAT is high when busy and low when ready to receive a new command.

DAC DATA is written to a register and then transmitted to the DAC device. Reading back from the address retrieves the DAC data sent. The act of writing to the register causes the SPI controller to begin sending data to the DAC.

The DAC used is a 10 bit device. The command structure is designed to work with several models including a 12 bit version. The upper 4 bits are control and the lower 12 are data. Data is MSB aligned and zero's are added to the LSB side where shorter control words are used.

Bit 15-12 = Control C3-C0
Bit 11-0 = Data, in the 10 Bit case D11-D2 are valid and D1,D0 are set to 0.

C3-C0
0000 outputs unchanged, load register A
0001 outputs unchanged, load register B
0010 outputs unchanged, load register C
0011 outputs unchanged, load register D
0100 outputs unchanged, load register E
0101 outputs unchanged, load register F
0110 outputs unchanged, load register G
0111 outputs unchanged, load register H

1000 plus DATA select output register(s) to update from stored shift register data
1001 load all input registers from shift register output unchanged - parallel value load
1010 load all input and output registers from common input.

Other control options exist to program slew rates and so forth. Please refer to the Maxim 5590-5595 data sheet for more information.

The DAC serial interface utilizes a 20 MHz reference frequency. The commands are 16 bits long. Discounting the overhead within the state-machine and rate matching, the transmit delay is .8 uS allowing an update rate of 1.25 MHz. Accounting for overhead and SW latency the maximum rate will be closer to 1 MHz. If you are running at the maximum rate the skew rate may come into play. The default for the DAC is 1.6 V/uS. If you will be exceeding this rate the Fast mode will need to be programmed in using the control codes to yield 3.6 V/uS.

The outputs are buffered with inverting op-amps. The range is converted to provide a 0-M3.3V range for the attenuators. The 10 bit range of the DAC corresponds to the 0-(-3.3) range on the output.
Vout at the DAC = .5*3.3V * Code/4096 for the forced sense configuration. The buffer opamps employ a gain of -2 to offset the factor of 1/2 and invert.
VOUT[connector] = -3.3*code/4096 = -.00081* code

Upon reset the DAC can be programmed [with resistor options] to default to 0 or mid scale. The CSS1 setting is “0”.

cPCI_rcvrcntl_CHK

0x000c // 3 counter check on 10 MHz clock

Bits	Function
31-28	spare
27	CLOCK_10_SEL
26	cnt_match_int_en
25	CNT CLR
24	CNT EN
23-0	CNT END

To test the availability and frequency of the external 10 MHz clock a dual counter with common controls is supplied. The reference and test counters are cleared. The terminal count set and then counting enabled. When the terminal count is detected both counters are stopped. The test count is compared to the reference count and if similar the 10 MHz external is valid. If grossly different the signal is either un-reliable, not present, or at a different frequency.

Load the terminal count into the lower 24 bits with the enable off ‘0’ and the clear on ‘1’. The base frequency is 10 MHz based on the local oscillator. The count for a given time can be calculated based on the time and the period of the 10 MHz clock. Reload the count with the clear off ‘0’ and the enable off ‘0’. Reload again with the enable set ‘1’. This sequence of three writes will clear the count and then enable with the terminal count loaded into the check register.

When the Count Match changes to a set condition, the results can be read back.

The reference count will be the programmed count +1 due to the match logic feed-back delay. The test count can be compared to the reference count and should be within a couple of counts either way. There is some ambiguity due to phase differences.

cnt_match_int_en when ‘1’ and Count match is set will cause an interrupt request. Please note the master interrupt enable is also required. When ‘0’ Count Match can still be polled, but no interrupt will be generated from this function.

cPCI_rcvrcntl_CHKref

0x0050 // 20 counter check read-back

Bits	Function
31-25	spare
24	Count Match
23-0	Count Internal reference

cPCI_rcvrcntl_CHKtst

0x00A8 // 42 counter check read-back

Bits	Function
31-25	spare
24	Count Match
23-0	Count Tested clock

Count Match status. When ‘1’ the reference count matched the programmed count. The status is repeated on both ports for convenience.

The values from the counters controlled with the counter check register are read back here. The match bit should be monitored to find the end of the test. The values can be read and compared. Due to the asynchronous nature of the external clock the counts may differ by a few lsb’s. If more than a few lsb’s and still a value then a frequency issue exists with the external clock. If the count is near or at zero then the clock is not present.

cPCI_rcvrcntl_RSL

0x0010 // 4 - RS data 31-0

cPCI_rcvrcntl_RSU

0x0014 // 5 - RS data 63-32

Data to be sent from the SPI port to the Receiver Synthesizer is stored into the RSL and RSU registers. The transmission is 2 bit parallel. The bits are transferred from each register in parallel. Data1 is loaded from the RSL register and Data2 is loaded from the RSU register. Data is sent MSB first. Data is MSB aligned.

RSU when written causes the transfer to start.

cPCI_rcvrcntl_RSC

0x0018 // 6 - RS control

Bits	Function
17	RS_LOCK – read only
16	RS_STAT – read only
15-14	spare
13-8	length
7-6	spare
5	RS_10MHZSEL
4	RS_DACONL
3	Channel Select
2-0	ABC Bits

Prior to writing to the RSU and RSL registers with the data to be sent to the RS SPI port the port should be initialized. The RSC port controls the length and ancillary bits associated with the SPI port.

The length is set with 1-32 valid options [written as hex]. The length is for the number of clocks. 2X data will be transferred [2,4,6..64 total bits sent].

Data1 is sent with the data valid on the rising edge of the clock. Data2 is sent with the data valid on the falling edge of the clock. The transitions are on the opposite edge for both Data streams. The enable signal is asserted low prior to the first clock and held asserted until after the last complete clock. The clock is

low between transmissions.

Once a transmission is started with the RSL port the RS_STAT can be read to check when the transmission is complete. When RS_STAT is '1' the port is busy.

RS_LOCK reflects the state of the RS_LOCK signal on the RS connector.

RS_10MHZSEL and RS_DACONL are register bits driven with LVTTL buffers to the RS connector.

The Channel Select and ABC bits are registered and held until a transfer occurs on the SPI bus. The bits can be changed at any time. The outputs will not change until a transfer has occurred.

cPCI_rcvrcntl_CSL

0x0020 // 8 - CS data 31-0

cPCI_rcvrcntl_CSU

0x0024 // 9 - CS data 63-32

Data to be sent from the SPI port to the Calibration Source is stored into the CSL and CSU registers. The transmission is 2 bit parallel. The bits are transferred from each register in parallel. Data1 is loaded from the CSL register and Data2 is loaded from the CSU register. Data is sent MSB first. Data is MSB aligned.

CSU when written causes the transfer to start.

cPCI_rcvrcntl_CSC

0x0028 // 10- CS control

Bits	Function
17	CS_LOCK – read only
16	CS_STAT – read only
15-14	spare
13-8	length
7-6	spare
5	CS_10MHZSEL
4	CS_DACONL
3	CS_CHANNEL Select
2-0	spare

Prior to writing to the CSU and CSL registers with the data to be sent to the CS SPI port the port should be initialized. The CSC port controls the length and ancillary bits associated with the SPI port.

The length is set with 1-32 valid options [written as hex]. The length is for the number of clocks. 2X data will be transferred [2,4,6..64 total bits sent]

Data1 is sent with the data valid on the rising edge of the clock. Data2 is sent with the data valid on the falling edge of the clock. The transitions are on the opposite edge for both Data streams. The enable signal is asserted low prior to

the first clock and held asserted until after the last complete clock. The clock is low between transmissions.

Once a transmission is started with the RSL port the CS_STAT can be read to check when the transmission is complete. When CS_STAT is '1' the port is busy.

CS_LOCK reflects the state of the CS_LOCK signal on the CS connector.

CS_10MHZSEL and CS_DACONL are register bits driven with LVTTL buffers to the CS connector.

CS Channel Select is a register bit. The bit is driven to the output at the end of the transmission. The register bit can be changed, and the output will not change until the transmission occurs.

cPCI_rcvrcntl_TMP

0x0030 // 12 TMP123 port

Bits	Function
16	TMP_STAT
15-0	TMP_DATA

Writing to this port causes the hardware to access the TMP123 device and to update the value held in the read-back register. The data written to the port can be anything. TMP_STAT will be '1' when busy and '0' when the data has been stored into the holding register.

Reading from the TMP123 device retrieves the previous conversion value and starts a new conversion. The conversion takes about 500 mS. The initial value read will not be a valid temperature value as there are none stored within the TMP123 to read [after power on reset]. The first access requires two write, wait for valid, read cycles to get the temperature. A 500 mS delay should be inserted between the writes to allow the device to complete the requested conversion.

If the temperature is read continuously – every 500 mS or so; the pipeline within the write – read cycle will not affect the validity of the readings. If the temperature is checked on a less frequent basis you may want to write, wait for

valid, wait 500 mS]. write, wait for valid and then read to get a value that is current.

The TMP_DATA is 12 bits plus a sign bit. Bit 15 is the sign bit. The LSB is .0625C. Read the value, right shift 3 places to be LSB aligned. The range of the device is –25 to + 85 with 1.5C accuracy, and –40 to 125C with ±2C. The data is in binary 2's complement form.

cPCI_rcvrcntl_ADCC

0x0038 // 14 ADC Control Register offset

Bits	Function
7	Start bit
6	SEL2
5	SEL1
4	SEL0
3	SGL/DIF
2	UNI/BIP
1	PD1
0	PD0

cPCI_rcvrcntl_ADCD

0x003C // 15 ADC Data and Status offset

Bits	Function
17	ADC SSTRB
16	ADC Status
15-0	ADC Data

Writing to the ADCC will cause an ADC cycle to be performed. The hardware will take the 8 bits stored in the register and write those to the ADC. The ADC will respond by converting one of the 8 channels as identified within the 8 bit data stream and send back 14 bits of data. The upper two bits are set to 0 within the FPGA. The data is available in the ADCD register.

The Start bit should be set for any command. SEL2..0 select the channel or channel pair to be converted. SGL/DIF when ‘1’ selects single ended. When ‘0’ selects differential. SEL in this case selects the channel pair and polarity. UNI/BIP when ‘1’ selects Unipolar operation. When ‘0’ selects BiPolar operation. PD1,0 select the clock and power down modes. Normally set to “11”.

The table on the next page shows the SEL and channel relationship. Please note that the Single Ended table has an interesting correlation between SEL and the channels. The Differential table is more linear.

With SGL/DIF = ‘1’

SEL2	SEL1	SEL0	0	1	2	3	4	5	6	7
0	0	0	+							
1	0	0		+						
0	0	1			+					
1	0	1				+				
0	1	0					+			
1	1	0						+		
0	1	1							+	
1	1	1								+

With SGL/DIF = ‘0’

SEL2	SEL1	SEL0	0	1	2	3	4	5	6	7
0	0	0	+	-						
0	0	1			+	-				
0	1	0					+	-		
0	1	1							+	-
1	0	0	-	+						
1	0	1			-	+				
1	1	0					-	+		
1	1	1							-	+

While the ADC conversion is taking place the Status is set to busy ‘1’, and when the data is ready to be read, cleared to ‘0’.

SSTRB is a status bit directly from the ADC and can be polled. The status is of little use to SW and normally not used. It is easy to miss as it is a short pulse.

cPCI_rcvrcntl_UARTC

0x0040 // 16 - UART Control register

Bits	Function
19	rt_tx1_amt_intlvl_en <
18	rt_rx1_afl_intlvl_en <
17	rt_tx1_amt_int_en <
16	rt_rx1_afl_int_en <
15	UART TX FIFO CLR
14	spare
13	UART TX EN
12	UART RX EN
11	spare
10	UART STOP
9	UART PARITY SEL
8	UART PARITY EN
7-0	UART BAUD RATE

If the PLL is programmed to generate 18.432 MHz the dividers for the UART will produce 1.152 MHz as a reference. 18.432 divided by the baud rate divider will produce the RX clock and 1.152 divided by the baud rate divider will produce the TX reference. The TX rate matches the data rate. The RX rate is 16x the data rate.

The Baud rate divider is 2(n+1). For 38.4K take 1.152/38.4 => 30. 30/2 – 1 = 14 => 0x0e for the baud rate factor. The RX side will have 16x this rate to properly process the received characters.

The Parity Enable when set ‘1’ will add parity to the transmitted character and check for parity on the RX side. When ‘0’ parity is not added on the TX or checked on the RX.

Parity Select when set ‘1’ adds odd parity to the character and looks for odd parity on the Rx side. When ‘0’even parity is used. Only has meaning when Parity Enable is set.

UART STOP when set adds 2 stop bits to the character. When ‘0’ 1 stop bit is added. If data remains in the FIFO after a character is sent the remaining data can be sent immediately after the end of the previous character. Adding an extra Stop Bit forces 1 extra bit time between characters. If there is no more data to send the marking state is used.

RX EN and TX EN when set ‘1’ allow the state-machines to transmit or receive data. When ‘0’ data will not be received or sent respectively.

To clear the RX FIFO disable the receiver and read from the FIFO until empty.

To clear the TX FIFO set the TX FIFO CLR bit. The state-machine will continuously read from the TX FIFO until it is empty.

rt_rx1_afl_int_en <, rt_tx1_amt_int_en < when set ‘1’ enable the latched level interrupts associated with the UART.

Uart_rx1_afl_int_en when ‘1’ will enable the “Almost Full” interrupt for the UART receiver. When the level within the UART RX FIFO goes from the state of not almost full to almost full the interrupt will be triggered. The interrupt is latched and the state remembered – the interrupt will stay asserted even if the level becomes not almost full. The interrupt is cleared by writing to the status register. The concept is to use the programmed Almost full level to “know” how much can be read. When the interrupt is asserted, the software can do quick loop without needing status reads to empty the Almost Full amount of data.

rt_rx1_afl_intlvl_en < is similar to uart_rx1_afl_int_en. Both are triggered by the Almost full condition. The difference is that the rt_rx1_afl_intlvl_en < version is not registered and conditioned to require the transition from not almost full to almost full. There is no clear bit associated with this interrupt enable. The mask or the master interrupt enable can be used to keep this condition from generating an interrupt.

rt_tx1_amt_int_en < when ‘1’ will enable the “Almost Empty” interrupt for the UART transmitter. When the level within the UART TX FIFO goes from the state of not almost empty to almost empty the interrupt will be triggered. The interrupt is latched and the state remembered – the interrupt will stay asserted even if the level becomes not almost empty. The interrupt is cleared by writing to the status register. The concept is to use the programmed Almost empty level to “know” how much can be written. When the interrupt is asserted, the software can do quick loop without needing status reads to fill the FIFO with data.

rt_tx1_amt_intlvl_en < is similar to rt_tx1_amt_int_en <. Both are triggered by the Almost Empty condition. The difference is that the rt_tx1_amt_intlvl_en < version is not registered and conditioned to require the transition from not almost empty to almost empty. There is no clear bit associated with this interrupt enable. The

mask or the master interrupt enable can be used to keep this condition from generating an interrupt.

cPCI_rcvrcntl_UARTrx

0x0044 // 17 - UART FIFO RX read

Read from this FIFO to retrieve data stored from the RX UART. Bits 7-0 contain the data. LW access.

cPCI_rcvrcntl_UARTtx

0x0048 // 18 - UART FIFO TX Write

Write to this address to store data for the TX UART to send. 7-0 contain the data bits. LW access. 31-24 are don't care bits.

cPCI_rcvrcntl_UARTst

0x004C // 19 - UART FIFO TX Write

Read from this port for UART status

Bits	Function
16	TX FIFO FULL
15	TX FIFO MT
14	spare
13	spare
12	RX FIFO FULL
11	RX FIFO MT
10	RX DN
9-4	Spare
3	tx1ffamt
2	rx1ffafl
1	tx1ffamt_int_stat
0	rx1ffafl_int_stat

TX FIFO FULL when '1' indicates that the TX FIFO has reached capacity and no new characters will be accepted. When '0' more data can be written; at least 1 open location.

TX FIFO MT when '1' indicates that the TX FIFO is empty and can have the FIFO

depth written. When ‘0’ indicates that the FIFO is not empty – has at least one location filled.

RX FIFO FULL when ‘1’ indicates that the receiver FIFO has all locations filled. Any new data received prior to data being read will cause an over flow condition. When ‘0’ at least one location is available.

RX FIFO MT when ‘1’ indicates that no data is stored in the RX FIFO. When ‘0’ at least one location is filled.

RX DN when ‘1’ indicates that the receiver state machine is not working on a character. Real time status which will be hard to use with software.

Tx1ffamt when ‘1’ indicates that the UART TX FIFO is almost empty. When ‘0’ more than the almost empty level is present.

Rx1FFAFL when ‘1’ indicates that the UART RX FIFO is almost full. When ‘1’ less than the almost full level are stored in the FIFO.

Tx1ffamt_int_stat is the latched status bit for the Almost Empty condition. When ‘1’ the level within the TX FIFO has gone from above or equal to below the Almost Empty threshold. Clear by writing to this port with this bit set.

Rx1ffafl_int_stat is the latched status bit for the Almost Full condition. When ‘1’ the level within the RX FIFO has gone from less than to equal or greater than the Almost Empty threshold. Clear by writing to this port with this bit set.

cPCI_rcvrcntl_UARTaf

cPCI_rcvrcntl_UARTaf	0x00A0 // 40 UART FIFO AFL level
15-0	Programmable Almost Full level

The PAF field is set to 16 bits for read-write. Bits 9-0 are used in this design for the Programmable Almost Full level. The level written to this register is used to compare against the count from the RX FIFO to determine if the FIFO is Almost Full or not. When the PAF value written is less than or equal to the count from the FIFO; the PAF bit is set in the status register and an interrupt can be generated.

cPCI_rcvrcntl_UARTae

cPCI_rcvrcntl_UARTae 0x00A4 // 41 UART FIFO AMT level

15-0 Programmable Almost Empty level

The PAE field is set to 16 bits for read-write. Bits 9-0 are used in this design for the Programmable Almost Empty level. The level written to this register is used to compare against the count from the TX FIFO to determine if the FIFO is Almost Empty or not. When the PAE value written is greater than the count from the FIFO; the PAE bit is set in the status register and an interrupt can be generated.

cPCI_rcvrcntl_TOAL

0x005C // 23 - Low Side[4-0]

Bits	Function
16	Count Enable
15	TOA Reset Enable
4-0	Lower Counter Pre-load bits

cPCI_rcvrcntl_TOAU

0x0060 // 24 - high side 27-0

Bits	Function
27-0	Upper Counter Pre-load bits

TOA Reset Enable when set '1' allows a pulse generator to send a 10 clock wide active high pulse on the TOA RESET Line. The 125 MHz clock is the reference for an 80 nS wide pulse. The pulse is created when the TOAU register is accessed.

Count Enable when set '1' enables the TOA counter to operate. When '0' the counter is stopped and the count is held.

When TOAU is accessed the values loaded into the TOAU and TOAL registers are loaded into the counter.

The values in the registers are offsets for the initial loop and have no affect once the counter rolls over.

The lower 5 bit portion of the counter counts from 0-24. The counter when it rolls over will enable the upper portion of the counter to count. This creates a divide by 25 on the 125 MHz for a counting rate of 5 MHz on the upper bits, still referenced to the 125 MHz clock.

The output from the counters is used for the TOA data stored onto a FIFO. See TOAFL and TOAFU registers.

cPCI_rcvrcntl_TOAC

0x0064 // 25 - toa cnt 27-0 [upper side]

Bits	Function
27-0	TOA Count

The upper counter data can be read back through this port.

cPCI_rcvrcntl_TOAC1

0x0068 // 26 - toa cnt1 27-0 [upper side]

Bits	Function
27-0	TOA CNT1

cPCI_rcvrcntl_TOAC2

0x006C // 27 - toa cnt2 27-0 [upper side]

Bits	Function
27-0	TOA CNT2

cPCI_rcvrcntl_TOAC3

0x0070 // 28 - toa cnt3 27-0 [upper side]

Bits	Function
31-28	Load Count
27-0	TOA CNT3

TOA CNT[1..3] is a pipelined capture of the counter data. The counter data has been re-referenced to the PCI clock so there is some ambiguity between the exact count that was on the counter and the value stored. The counts are stored based on a special counter. The counter counts to create a 1mS delay between pipeline captures. The counter runs whenever the counter is enabled. The initial time to capture is not known exactly as the counter may have been stopped and started. With three stages the counts between stages can be checked against the expected to make sure the counter is advancing properly. The ATP test software does this.

The Load Count provides a reference count to the number of times the pipeline

has been stored. The count clears when the counters are disabled. It is suggested that this value be polled until the count is 3. The counter disabled and the pipeline read.

The data is stored into the pipeline 1 => 2 => 3 so the comparison should be 2-3 and 1-2. .001/(1/5000000) => 5000 => 0x1388 for the expected delta. A 5% margin is more than enough to compensate for the frequency domain change.

This feature is available mainly as an engineering test feature. It is not necessary to use in normal operation.

cPCI_rcvrcntl_AASC

0x007C // 31 - aas 15-0 control port

Bits	Function
4	AAS CNTL LD
3	spare
2	AAS CNTL EN
1	AAS CNTL S
0	AAS CNTL R

The AAS toggle output can be used with several different modes. Each interacts with the other. For example if the AAS bit is forced low and then put into automatic mode the bit will stay low until the next automatic mode change and toggle to high.

AAS CNTL R when set ‘1’ acts as a RESET input and forces the output bit low. When ‘0’ the control is in the inactive state.

AAS CNTL S when set ‘1’ acts as a SET input and forces the output bit high. When ‘0’ the control is in the inactive state.

AAS CNTL EN when set ‘1’ enables the counter. The counter is held when not enabled. Enable when set also enables automatic mode. The S and R controls can override the EN for the output. When set and R and S are ‘0’ the output will toggle based on when the counter underflows. The rate is programmable based on the count loaded.

The counter also requires the upper counter enable to be set [from the TOA counter] for the counter to decrease by 1. The reference clock is 125 Mhz. Essentially the count is the number of times the lower bits have sequenced through on the TOA counter [0-24] for a basically 5 Mhz rate.

AAS CNTL LD when set forces the counter to the initial value. The counter counts down and then reloads the value again. The Enable is not required to load. The LD signal should be released before going to the run state.

cPCI_rcvrcntl_AASLD

0x0080 // 32 - aas 31-0 down count reference

Bits	Function
31-0	AAS Count Initial Value

The count is loaded when the LD signal is set or when the counter underflows.

cPCI_rcvrcntl_COS

0x0088 // 34 - cos cntl 15-0

Bits	Function
16	gnd
15	FIFO Load Signal
14	toafifo_valid_int_en
13-12	spare
11	TOA FIFO read enable
10-8	spare
7	AAS RISing
6	AAS FALLing
5	BLANK2 RISing
4	BLANK2 FALLing
3	BLANK1 RISing
2	BLANK1 FALLing
1	OnePPS RISing
0	OnePPS FALLing

Each of the bits can be enabled ‘1’ or disabled ‘0’ independently. The 33 MHz clock is used to check for a Change of State on each bit [1PPS, BLANK1, BLANK2 and AAS]. When the change is from a ‘0’ state to a ‘1’ state and the RISing bit is set the TOA is captured and stored into a FIFO. Similarly if the signal of interest changes from ‘1’ to ‘0’ and the FALLing bit is set the TOA is captured into the same FIFO. If both are set the TOA is captured for any change in state,

The FIFO is 32 deep and 41 wide to capture the TOA [33 bits] plus the COS output status bits.

toafifo_valid_int_en when ‘1’ enables the interrupt based on the valid data being available in the output register.

If FIFO LOAD Signal is enabled ‘1’ the FIFO LOAD signal is generated when data is written to the TOA FIFO. The pulse is 10 clocks wide and referenced to the 125 MHz for an 80 nS wide pulse. The signal is buffered to LVTTL levels and located on J2.

TOA READ Enable when set ‘1’ enables the FIFO read state-machine to auto move data from the FIFO to a pair of holding registers for unified reading of the data. The state-machine senses when data is available to be read and when data has been read. The MT signal from the FIFO is used for the available side and the lower FIFO being read for the room in the register signal. TOA FIFO Valid is set when data is in the output registers. The status is available in the TOAFU register.

cPCI_rcvrcntl_TOAFL

0x008C // 35 - toa FIFO lower [31-0]

Bits	Function
31-0	TOA FIFO data

cPCI_rcvrcntl_TOAFU

0x0090 // 36 - toa FIFO Upper [40-32 + FIFO status MT, full, Valid]

Bits	Function
11	TOA FIFO VALID
10	TOA FIFO FULL
9	TOA FIFO MT
8	aas rising
7	aas falling
6	blank2 rising
5	blank2 falling
4	blank1 rising
3	blank1 falling
2	onePPS rising
1	onePPS Falling
0	TOA Count MSB

The TOA FIFO VALID bit can be polled along with the FIFO FULL and FIFO MT status bits without affecting the FIFO data. When the VALID bit is set the Count and COS bits are valid and stored into the two FIFO holding registers. The upper register should be read first. The lower register triggers the next move of data into the holding registers.

Receiver / Synthesizer Interface Pin Assignment

Name	Pin [J5]
+5V	1
+5V	14
-5V	2
GND	15
+15V	3
GND	16
-15V	4
V_TUNE_OUTP	17
+24V	5
V_TUNE_OUTN	18
RS_LOCK	6
GND	19
RS_DACONL	7
	20
RS_SPID2	8
	21
RS_10MHZSEL	9
	22
CHAN_SELRS	10
	23
RS_ENABLE	11
	24
RS_SPID1	12
	25
RS_CLOCK	13

FIGURE 2 CPCI RECEIVER CONTROLLER RS J5 PIN ASSIGNMENT

Calibration Source Interface Pin Assignment

Name	Pin [J6]
+5V	1
+5V	14
-5V	2
GND	15
+15V	3
GND	16
-15V	4
V_TUNE_OUTP	17
+24V	5
V_TUNE_OUTN	18
CS_LOCK	6
GND	19
CS_DACONL	7
	20
CS_SPID2	8
	21
CS_10MHZSEL	9
	22
CHAN_SELCS	10
	23
CS_ENABLE	11
	24
CS_SPID1	12
	25
CS_CLOCK	13

FIGURE 3 CPCI CALIBRATION SOURCE CS J6 PIN ASSIGNMENT

RF/IF Interface Pin Assignment

Name	Pin [J7,8,9,10]
+5V	1
-5V	6
+5V	2
ATTENUATION B	7
GND	3
	8
GND	4
CHAN SEL	9
ATTENUATION A	5

FIGURE 4 CPCI RC RF/IF J7,J8,J9,J10 PIN ASSIGNMENT

Front Panel Clock Interface Pin Assignment

Name	Pin [J7,8,9,10]
125 MHZ P	1
125 MHz N	6
	2
	7
	3
	8
	4
	9
	5

FIGURE 5 CPCI RC FP CLK J3 PIN ASSIGNMENT

Rear Panel cPCI J2 Interface Pin Assignment

Name	Pin [J2]
TMP FLD A P	C2
TMP FLD A N	C1
TMP FLD B P	B2
TMP FLD B N	B1
BLANK1	E3
BLANK2	E4
1PPS	E5
CLOCK_10_IN	D6
CLOCK_10_RTN	D7
ABC_0P	C8
ABC_0N	C9
ABC_1P	D8
ABC_1N	D9
ABC_2P	E8
ABC_2N	E9
AAS_P	C10
AAS_N	C11
SDI_P	D10
SDI_N	D11
SDO_P	E10
SDO_N	E11
TOA_RESET	E12
SPARE_1	C13
SPARE_2	D13
FIFO_LOAD	E13

FIGURE 6 CPCI RC RP J2 PIN ASSIGNMENT

J13 and J14 are vertical SMA connectors with 10 MHz available. The signals are driven with TTL levels.

Applications Guide

Interfacing

Some general interfacing guidelines are presented below. Do not hesitate to contact the factory if you need more assistance.

Installation

The PMC is mounted to the cPCI Receiver Controller prior to installation within the chassis. For best results: with the cPCI bracket installed, install the PMC at an angle so that the PMC front panel bezel penetrates the cPCI bracket then rotate down to mate with the PMC [PnX] connectors.

There are four mounting locations per PMC. Two into the PMC mounting bezel, and two for the standoffs near the PMC bus connectors.

Start-up

A third party PCI device cataloging tool will be helpful to check that the VendorID and CardID are “seen” by the OS.

Watch the system grounds. All electrically connected equipment should have a fail-safe common ground that is large enough to handle all current loads without affecting noise immunity. Power supplies and power consuming loads should all have their own ground wires back to a common point.

Power all system power supplies from one switch. Connecting external voltage to the cPCI Receiver Controller when it is not powered can damage it, as well as the rest of the host system. This problem may be avoided by turning all power supplies on and off at the same time.

Construction and Reliability

The cPCI Receiver Controller is constructed out of 0.062 high temp ROHS compliant material. Gold has been used for plating rather than Tin for improved performance over time. “leaded or unleaded” components can be used along with solder choices. Dynamic Engineering can support both processes.

Surface mounted components are used. The connectors are through hole soldered for the cables and compression fit for the cPCI.

Thermal Considerations

The cPCI Receiver Controller is built with “commercial” parts. The parts can be upgraded to provide an Extended Temperature “ET” version of the design. The connectors and other components are ET rated in either case.

The base design is fairly low powered and will not require a lot of cooling. External draw on the power supplies or IO signals can add a significant power load on the Receiver Controller. Forced air cooling is recommended in this case.

During T&I it is recommended to read the temperature sensor and see what temperature is registering on the board. If anywhere close to 70C forced air should be implemented. The MTBF will be longer at cooler temperatures [up to a point]. Remember that the TMP123 is measuring the temperature on the FAB near the FPGA. Other parts may be warmer and likely are hotter than the surface temperature of the PCB. Getting the temperature reading below 50C would provide quite a bit of margin and add to the MTBF. If this is not possible then the ET version is recommended as that adds 15C to the top end.

Warranty and Repair

Please refer to the warranty page on our website for the current warranty offered and options.

http://www.dyneng.com/warranty.html

Service Policy

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering Products not purchased directly from Dynamic Engineering contact your reseller. Products returned to Dynamic Engineering for repair by other than the original customer will be treated as out-of-warranty.

Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. The current minimum repair charge is $100. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois St. Suite C
Santa Cruz, CA 95060
831-457-8891
831-457-4793 fax
InterNet Address support@dyneng.com

Specifications

Logic Interfaces:	PCI Interface 33/32
Access types:	PCI target accesses. DMA can be added.
CLK rates supported:	125 LVDS based on 10 MHz from rear panel or local osc, 10 MHz [LVTTL] on SMA(2), Programmable UART Baud rates
Power	Fused 24V, ±15V, ±5V supplied to connectors from on-board power supplies. +5 used from cPCI bus.
SPI	Internal SPI used for A/D, D/A, Temperature sensor. HW implemented with simple register access. External SPI supplied for two buses. 2 bit parallel implementation with programmable length [1-32 bits], 20 MHz.
IO	Additional miscellaneous and program specific IO are supported with a combination of RS-485, LVTTL, and LVDS.
A/D	8 channels of A/D are supported. The A/D is programmable for single ended or differential operation, Uni or Bipolar encoding, Easy to use register based access with conversion status.
D/A	8 channels of D/A are supported. Each channel is separately programmable. The output is buffered with an inverting opamp to provide a 0-M3.3V range. Register based programming with conversion status. Multi and single channel access.
Temperature	The TMP123 temperature sensor is mounted near the FPGA and accessed with an SPI bus. Hardware takes care of the conversion. Software requests data, hardware retrieves the data and provides status for completion.
UART	TX and RX UART functions with RS-485 IO are provided. The PLL is used to generate the 18.432 reference clock and a software programmable baud rate generator used to select transmit and receive rates. 1023 x 8 buffer for TX and RX.

TOA	Time of Arrival counter provided using 125 MHz reference clock. COS on 4 signals to trigger TOA storage into a 32 x 41 FIFO. 33 bits of time tag plus status from COS generator.
DipSwitch	8 position switch supplied with register access. Switch can be used to differentiate between boards when more than one Receiver Controller is in the same system or for SW defined purposes.
FLASH	FPGA program is stored into FLASH memory with a JTAG header to allow for field updates and new programs.
Revision	The FPGA VHDL revision is programmed into a register to allow for SW detection of different revisions. When the base design changes a new CardID will be assigned.
Software Interface:	register mapped IO
Initialization:	registers are initialized to 0.
Interface:	Registers are R/W with the exception of Status bits.
Dimensions:	3U 4HP [minus internal cabling]
Construction:	High Temp ROHS compliant Multi-Layer Printed Circuit board, Through Hole and Surface Mount Components. Standard processing with leaded components and solder. –ROHS option for ROHS compliant components and solder.

Order Information

standard temperature range –0 ⇔ +70ºC
extended temperature range –40 ⇔ +85C

cPCI-Rcvr-Cntl-CSS1	3U 4HP cPCI card A/D, D/A, Temperature, UART, Two SPI buses, Clock references, Fused Power http://www.dyneng.com/cpci_rcvr_cntl.html
-CC	Conformal Coating is available as an option.
-ROHS	Add for ROHS processing.
-ET	Add Extended Temperature