The Australia Telescope
National Facility
PCI Correlator Interface.
Version 1.
2-Oct-2003

1.0 Description.

The PCI Correlator Interface is a plug-in card that allows the connection between the PCI bus and the dedicated Correlator 24/32 bit output bus. As the key logic on the card is handled by a Xilinx FPGA and a highly configurable PCI interface IC, many alternate functions not included here, can be easily implemented.

Notable features are:

- Maximum PCI data rate of 66MB/second (single control/data mapping).
- All I/O lines on Correlator Bus, and Auxillary Bus balanced TTL.
- Programmable external data transfer length from 1 to 16384 data words (32 bit).
- Supports simultaneous transfer from correlator bus to buffer memory and buffer memory to PCI bus.
- Occupies 256KB block of memory space on PCI bus.
- Control and Status as 5 X 32 bit registers.
- Interrupts on Data Transfer Completion, Timeout error, External Signal and/or any/all of the 4 Auxillary input lines..
- 4 auxillary input and 4 auxillary output lines.
- Transaction timeout limiter set to 256 system (usually PCI) clock periods.
- Selectable system clock rates: PCI, or onboard crystal oscillator.
- Activity light for PCI transactions, and for Correlator Bus transactions.
- Supports burst mode transfers but not bus mastering.

2.0 Operation.

2.1 Memory model.

The interface appears as a 256K byte block of memory on the PCI bus. Locations 0 through to word address 0x7FFF are the 5 X 32 bit Control and Status registers, and locations 0x8000 through to word address 0xFFFF is the buffer memory. The Control and Status registers are:

	Address Offsets	
Name	Word address	Byte Address
Control / Status Register	0x0000	0x00000000
Interrupt Control Register	0x0001	0x00000004
Interrupt Status Register	0x0002	0x00000008
Transfer Length Register	0x0003	0x0000000C
Start Address Register	0x0004	0x00000010

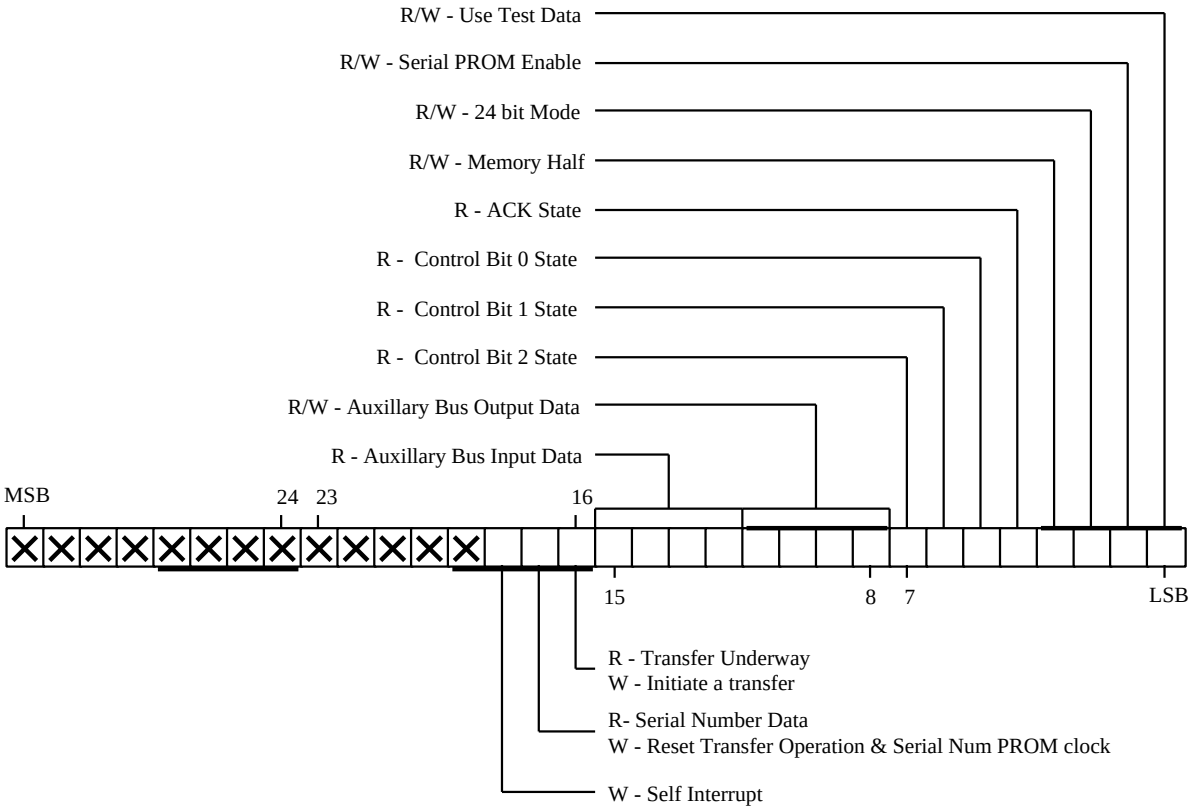
To aid with speedy access, only 5 bits of the address lines are used for decoding. Thus the register's image appear every 32 words (128 bytes) of memory.

2.2 Register Description.

2.2.1 The Control / Status Register.

This contains various bits that control and/or report the status of the interface and any pending operation. This is a limited read/write register.

CONTROL / STATUS REGISTER (Offset 0x0)



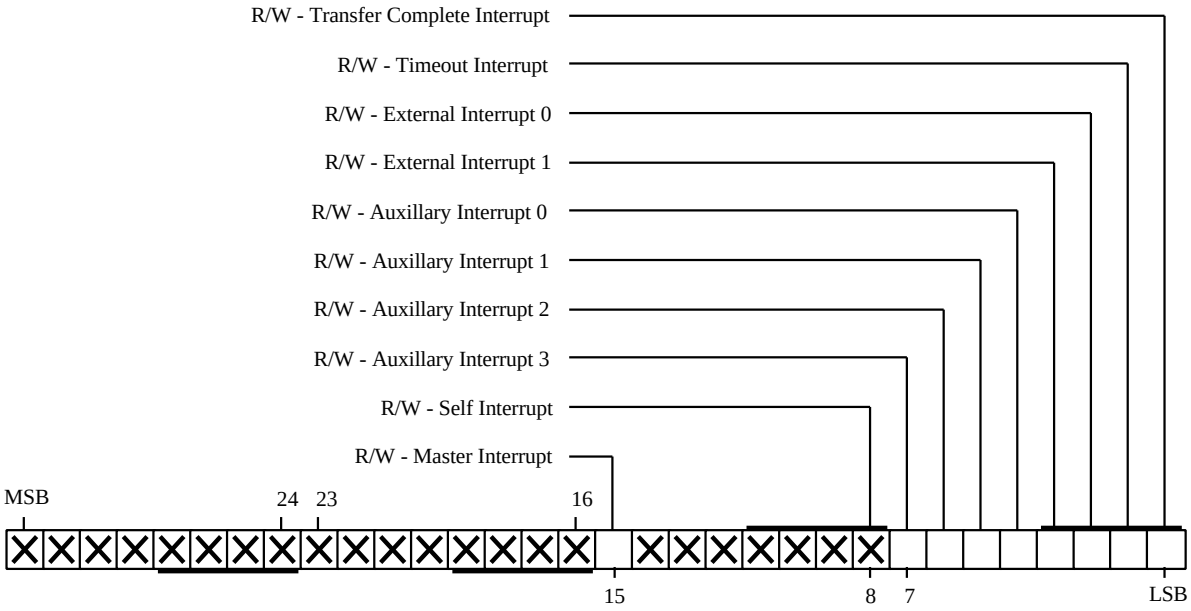
Mask	Type	Description
00000001	R/W	Use Test Data. When set, data place into the buffer memory is internally generated. When cleared, data is taken from the Correlator Bus. This is useful for internal testing.
00000002	R/W	Serial Number PROM Enable. Setting this bit enables the Serial Number PROM, and in doing so disables **all** access to the memory from the Correlator Bus. Accessing the Serial Number PROM should only be done during initialization or at a time when data is not being transferred from the Correlator Bus to the memory. Access form the memory to the host computer is not affected.
00000004	W	24 Bit Mode. When set, this bit configures the external bus to be 24 bits wide, making it compatable with the earlier ATNF Compact Array correlator. When cleared, this bit configures the external bus to be 32 bits wide. When configured as a 24 bit bus the most significant bit is extended to the top 8 bits. This saved the host computer the overhead of doing a sign extension to the data.
00000008	W	Memory Half. The half to start loading Correlator data into. This effectively drives the most significant bit of the buffer memory in the Correlator Data write

		circuitry (A14). When cleared (Lo) Correlator Data will be written into the lower half of the buffer memory starting from location 0x0. When set (Hi), Correlator Data will be written into the upper half of the buffer memory starting from location 0x4000. This allows the system to treat the buffer memory as 2 banks, so that a new transfer my be initiated before the previous data has been copied from the buffer. The lower half starts at PCI byte address offset of 0x20000, and the upper half starts at PCI byte address offset 0x30000.
00000010	R/W	ACK State. This is the ACK line as define on the external Correlator Bus cable.
00000020	R/W	Control Bit 0. This is the Control 0 line as define on the external Correlator Bus cable.
00000040	R/W	Control Bit 1. This is the Control 1 line as define on the external Correlator Bus cable.
00000080	R/W	Control Bit 2. This is the Control 2 line as define on the external Correlator Bus cable.
00000f00	R/W	Auxillary Output Data. These bits feed the 4 output bits of the Auxillary I/O bus.
0000f000	R	Auxillary Input Data. These bits are fed from the 4 input bits of the Auxillary I/O bus.
00010000	R	Transfer Underway. When set, this bit indicates that a transfer is presently underway.
	W	Initiate Transfer. Setting this bit initiates a transfer. The Transfer Underway bit is immediately set and remains set until the transfer is complete.
00020000	R	Serial Number PROM data bit. This is the data bit from the Serial Number PROM since the PROM's clock line was pulsed. Writing a Hi to this bit does the pulsing. A short software delay should be inserted between pulsing the Serial Number PROM clock line and reading this bit. The Serial Number PROM Enable bit in this register must be set for this to have any meaning.
	W	Reset Transfer Operation and Serial Number PROM clock. Writing a Hi to this bit immediately kills the current transfer operation as well as pulsing the Serial Number PROM clock line. After the Serial Number PROM has been clocked the next bit in the PROM can be read.
00040000	W	Self Interrupt. Writing a Hi to this bit initiates an interrupt. The Self Interrupt bit as well as the Master Interrupt enable bit must be set in the Interrupt Control Register for the interrupt to actually occur.

2.2.2 Interrupt Control Register.

This contains a series of bits that enable the source of interrupts. This is a read/write register.

INTERRUPT CONTROL REGISTER (Offset 0x4)

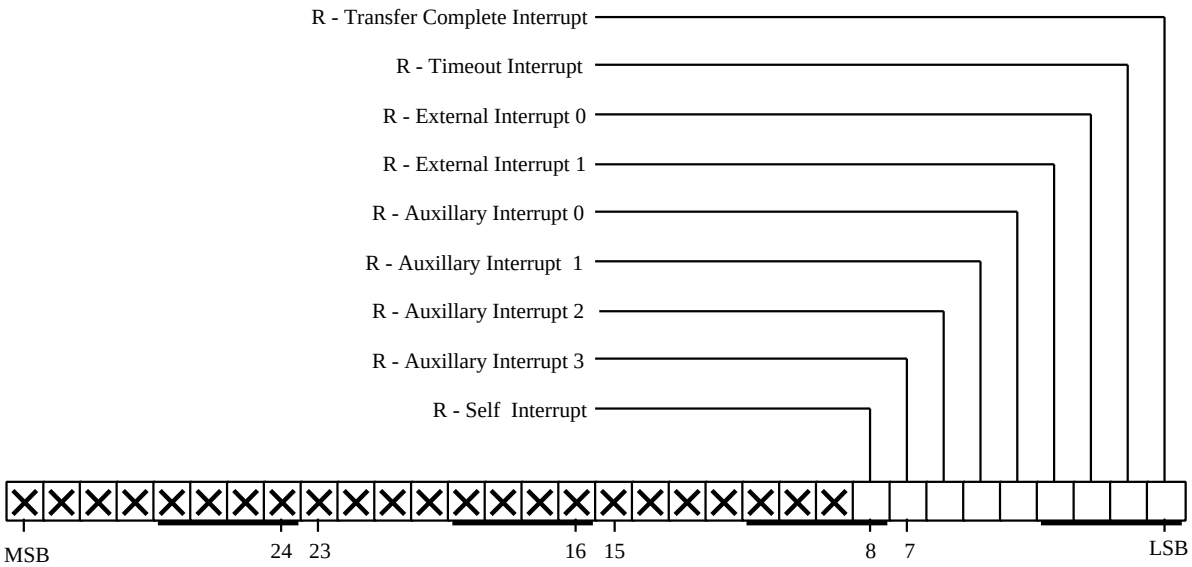


Mask	Type	Description
00000001	R/W	Transfer Complete Interrupt Enable. When set, an interrupt will be generated on completion of a transfer. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000002	R/W	Timeout Interrupt Enable. When set, an interrupt will be generated whenever a timeout occurs. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000004	R/W	External Interrupt #0 Enable. When set, an interrupt will be generated on a rising edge of the External Interrupt #1 line. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000008	R/W	External Interrupt #1 Enable. When set, an interrupt will be generated on a rising edge of the External Interrupt #1 line. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000010	R/W	Auxillary Interrupt #0 Enable. When set, an interrupt will be generated on a rising edge of the Auxillary Input #0 line. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000020	R/W	Auxillary Interrupt #1 Enable. When set, an interrupt will be generated on a rising edge of the Auxillary inp Input ut #1 line. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000040	R/W	Auxillary Interrupt #2 Enable. When set, an interrupt will be generated on a rising edge of the Auxillary Input #2 line. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000080	R/W	Auxillary Interrupt #3 Enable. When set, an interrupt will be generated on a rising edge of the Auxillary Input #3 line. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.
00000100	R/W	Self Interrupt Enable. When set, an interrupt will be whenever a Hi is written into the Self Interrupt bit in the Control/Status Register. The Master Interrupt Enable bit must also be set for the interrupt to actually get to the CPU.

00008000	R/W	Master Interrupt Enable. When set an interrupt is allowed to pass to the CPU. When cleared, no interrupts will be passed. If any of the above interrupts are enabled, their presents will still show in the Interrupt Status Register.

2.2.3 Interrupt Status Register.
This contains a series of bits that indicate the source of the interrupt. This is a read only register.

INTERRUPT STATUS REGISTER (Offset 0x8)



Mask	Type	Description
00000001	R	Transfer Complete Interrupt. When set, a transfer complete signal has been generated. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.
00000002	R	Timeout Interrupt. When set, a timeout signal occurred during a transfer. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.
00000004	R	External Interrupt #0. When set, a rising edge on the External Interrupt #0 line occurred. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.
00000008	R	External Interrupt #1. When set, a rising edge on the External Interrupt #1 line occurred. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.

00000010 R Auxillary Interrupt #0. When set, a rising edge on the Auxillary Input #0 line occurred. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.

00000020 R Auxillary Interrupt #1. When set, a rising edge on the Auxillary Input #1 line occurred. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.

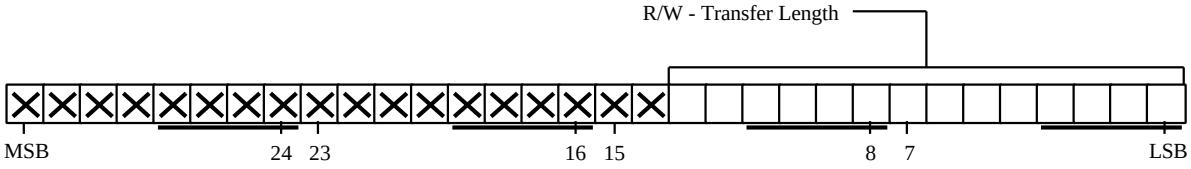
00000040 R Auxillary Interrupt #2. When set, a rising edge on the Auxillary Input #2 line occurred. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.

00000100 R Self Interrupt. When set, a self interrupt was initiated. The contents of this register should be ANDed with the contents of the Interrupt Control Register, to confirm that this interrupt is intended. The contents of this register is cleared upon a read.

2.2.4 Transfer Length Register.

This contains the number of words that are to be collected by the interface. The maximum limit is half the size of the on board memory on the interface card (32K words) which is 16K words, ie 14 bits. Once initiated, the interface will cycle until the number of words set in this register are collected, an error occurs, or terminated under programme control. This is a limited read/write register. The data is placed into the bottom or top half as controlled by the Memory Upper Half bit in the Control and Status Register.

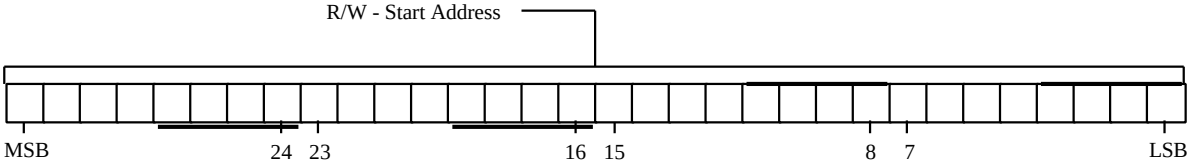
TRANSFER LENGTH REGISTER (Offset 0xC)



2.2.5 Start Address Register.

This is the start address on the Correlator bus from where to get the data. This 32 bit number will be broadcast on the bus at the start of a transfer. Any external device recognising this value will respond by driving the ACK line Hi. Thus for older 24 bit versions of the Correlator bus, only the lower 24 bit have any meaning, and for 32 bit versions, all 32 bits can be used. This is a full read/write register.

START ADDRESS REGISTER (Offset 0x10)



2.3 Use.

2.3.1 External synchronisation.

To synchronise the computer to external events perform the following.

i. Enable the relevant External Interrupt line (#0 or #1, or both) and/or any of the Auxillary input lines and the Master Interrupt Enable bit in the Control/Status register.

ii. On interrupt, the ISR reads the Interrupt Status Register to find out what caused the interrupt. This will reset the interrupt line. Process all set interrupt bits.

2.3.2 To perform a transfer the following is required:

i. Load the number of words to retrieve from the correlator, by loading the length field in the Transfer Length register. Normally this need only be configured once if all data transfers from the correlator are the same length.

ii. Enable the Transfer Completion interrupt, the Timeout interrupt (if required) and the Master Interrupt Enable bit (if not already set). The interface will cause an interrupt on the PCI bus on completion of the data transfer from the correlator to the PCI interface card or on timeout.

iii. Write a source address to the Address Register.

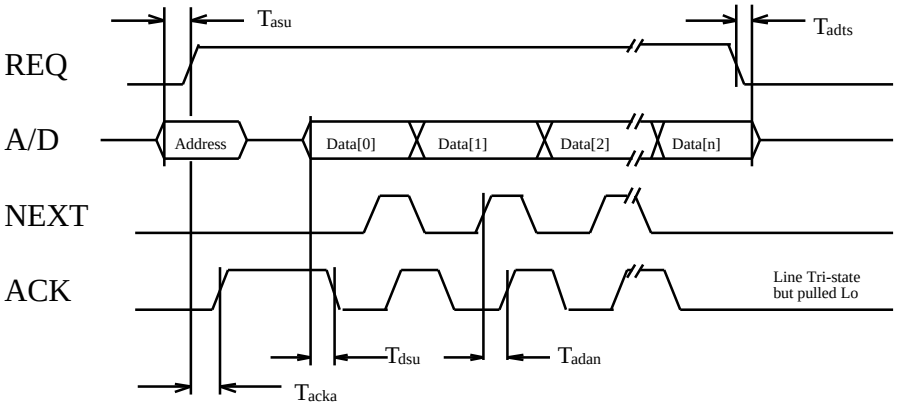
iv. Set the Initiate Transfer bit in the Control/Status Register This will trigger the interface to collect the requested number of data words from the correlator memory/accumulator starting from the address in the Address Register. On Transfer Complete or Timeout an interrupt will be issued.

iv. On interrupt, the ISR should read the Interrupt Status Register to find out what caused the interrupt. This will reset the Interrupt Register. Process all set interrupt bits, and flag the data collection process to collect the data.

v. Collect the data. This is performed by software on the host computer.

2.4 Correlator Bus Timing.

2.4.1. Typical Transaction.



DMA Bus transfer

At the start of the transfer cycle, the master places the source address onto the AD lines of the bus. After period Tasu, the REQ line is asserted Hi. This indicates to the targets that there is a

valid address on the cable. The host waits for a target to acknowledge the address by raising ACK Hi. On seeing ACK go Hi, the master immediately tri-states the AD lines, and waits for the target to lower the ACK line. The target, having recognised its address, fetches the first word of data starting at the address given, and places it onto the bus. After minimum period data setup time, Tdsu, the target lowers the ACK line indicating to the master that data is available. The master, seeing the ACK line go Lo, loads the data into its holding memory, and then signals the target to send the next data word by raising NEXT. The target, seeing NEXT go Hi, raises ACK hi, and commences to fetch the next word. The master seeing ACK go Hi immediately lowers NEXT. The target will not put any data onto the bus until NEXT has been lowered. The target will continue to place data onto the bus in response to the NEXT signal.
The process continues until the REQ line is deasserted. At any point, whenever the interface is waiting for a response from a target, a timer is initiated, to limit the duration of the wait. In the event of this time limit expiring, the entire transfer process is aborted, and a timeout error signal generated.

2.4.2. Timeout control.
All transactions on the Correlator Bus are timed. In the event that a target has not reponded within this time, a Timeout signal is issued internally, and the current transaction are aborted. If the Timeout Interrupt Enable bit is set in the Interrupt Control Register, then the Timeout Interrupt flag is set. If the Master Interrupt enable bit in the same register is also set, then an interrupt is generated. The Timeout value is currently set to 256 system clock periods. If the system clock is derived directly from the PCI clock, then this will be 256 PCI clock periods. For a PCI clock rate of 33MHz, the Timeout period will be approximately 7.757 μS. It should be noted that the PCI bus clock rate can (legally) vary between 0 and 33MHz.

3.0 Hardware.

3.1 User Configurable links.
There are 14 hardware links on the card. This section details their function. **Links denoted by "*" indicate the link is hard wired by the use of a shorting track and this track must be cut before setting alternate options.**

LK1. Enables/disables test mode of the PLX9050 PCI interface chip. See PLX Technology PLX9050 data sheets for more information.
Closed Places the PLX9050 PCI chip into 'test' mode.
Open Places the PLX9050 PCI chip into 'normal' mode.

LK2. This along with LK14 sets the source of the boards system clock. The user can select either the PCI clock, the onboard crystal oscillator, or from the Xilinx which would be a derivative of the former two (for example PCI-clock / 2).
1-2 Use system clock from an onboard source.
2-3 Use output from Xilinx chip for system clock.

LK3. This link (along with LK4), allows the user to set the 4^{th} output control line to either be a power source, or a logic signal generated from the Xilinx chip. When used with the ATNF correlators (including the Multibeam and the SEST correlators) this link would be set to be a power source.
1-2 Drive 4^{th} (+) output control line to GND.

2-3 Drive 4th (+) output control line from Xilinx.

LK4 (see also LK3).
1-2 Drive 4th (-) output control line to +5V.
2-3 Drive 4th (-) output control line from Xilinx.

LK5. This link, along with LK6, LK7, and LK8 allow the user to enable/disable line termination in the Auxillary input control lines.
Closed Terminate AUX-IN0 line.
Open No terminate AUX-IN0 line.

LK6 (see LK5).
Closed Terminate AUX-IN1 line.
Open No terminate AUX-IN1 line.

LK7 (see LK5).
Closed Terminate AUX-IN2 line.
Open No terminate AUX-IN2 line.

LK8 (see LK5).
Closed Terminate AUX-IN3 line.
Open No terminate AUX-IN3 line.

LK9. This link maps the onboard memory to either Chip Select region 0 or 1. The PLX9050 chip supports up to 4 separately mapped regions. The user can select which region to use.
1-2 Map onboard memory to CS0 region.
2-3 Map onboard memory to CS1 region.

LK10. This link is in place of a fixed pushbutton switch. A remote switch may be connected to the pins on this link. Shorting the link pins will reset the Xilinx IC.
Closed Forces a local reset.
Open Normal operation.

LK11. This link sets the configuration mode of the Xilinx IC. If configuring using a download cable, set the mode to Slave, and if configuring from the serial PROM, set the mode to Master.
Closed Sets Xilinx for Master serial mode.
Open Sets Xilinx for Slave serial mode.

LK12, LK13. These two links connect directly to the ~PRSNT1 and ~PRSNT2 lines on the PCI bus. Thes lines indicate to the host system whether or not a board is present, and if so the maximum power it will draw. As this board uses less than 15W the links have been set to reflect this.

Closed*	Closed*	Board present, requires max. 7.5W power.
Closed	Open	Board present, requires max. 25W power.
Open	Closed	Board present, requires max. 15W power.
Open	Open	Ignore board's present.

LK14 (see also LK2).

1-2* Set onboard clock source as PCI clock.
2-3 Set onboard clock source as crystal oscillator.

3.2 Connector Pin Description.
3.2.1 DMA bus. The DMA bus connector is made up of 32 data lines, 4 input control lines and 4 output control lines, all using balanced TTL signal transmission (RS-422 drivers / receivers). The drivers and receivers for these lines may be swapped for Low Voltage Differential Signal types. The 40 signal pairs are made available through two 40 way 0.1” pin spaced headers. The pinouts are listed below:
Connector 1:

Line	True	Complement	Description
D0	1	2	Data (I/O)
D1	3	4	Data (I/O)
D2	5	6	Data (I/O)
D3	7	8	Data (I/O)
D4	9	10	Data (I/O)
D5	11	12	Data (I/O)
D6	13	14	Data (I/O)
D7	15	16	Data (I/O)
D8	17	18	Data (I/O)
D9	19	20	Data (I/O)
D10	21	22	Data (I/O)
D11	23	24	Data (I/O)
D12	25	26	Data (I/O)
D13	27	28	Data (I/O)
D14	29	30	Data (I/O)
D15	31	32	Data (I/O)
D16	33	34	Data (I/O)
D17	35	36	Data (I/O)
D18	37	38	Data (I/O)
D19	39	40	Data (I/O)

Connector 2:

Line	True	Complement	
D20	1	2	Data (I/O)
D21	3	4	Data (I/O)
D22	5	6	Data (I/O)
D23	7	8	Data (I/O)
D24	9	10	Data (I/O)
D25	11	12	Data (I/O)
D26	13	14	Data (I/O)
D27	15	16	Data (I/O)
D28	17	18	Data (I/O)
D29	19	20	Data (I/O)
D30	21	22	Data (I/O)
D31	23	24	Data (I/O)
Cout0	25	26	Control (O)
Cout1	27	28	Control (O)
Cout2	29	30	Control (O)
Cout3	31	32	Control (O)
Cin0	33	34	Control (I)
Cin1	35	36	Control (I)
Cin2	37	38	Control (I)
Cin3	39	40	Control (I)

I/O – bidirectional signal.
I – Input to Interface card.
O – Output from Interface card.

4.0 Hardware.

4.1 PCI interface configuration.
The PLX PCI9050 interface chip may be configured for many different modes. This section details two modes possible for this particular application. For a full understanding of the PCI9050 chip consult the PCI9050 data sheet.

4.1.1 Single Region.
In this mode, both the memory and the control/status registers share the same PCI memory area. The control appears in the bottom half of the region and the memory, where data from the external correlator accululators is held, occupies the top half. This therefore requires a memory region of 256K of memory. The interface card should then be configured to use only “CS0” control line. The main drawbacks to this regime is there must be at least 1 wait state inserted into each local transfer when using to the slower Xilinx chips (-3 or slower), and there is much wasted PCI memory space. Insertion of wait states effectively at least halves the PCI data rate. (The local data rate does not effect the operation of the PCI bus). The advantage, on the other hand, is that it makes the driver very simple.

PROM data.

Addr	Contents	Description

0	9050	Device ID - PLX PCI9050
2	10B5	Vendor ID.
4	0680	Class Code.
6	0001	Class Code revision (not loadable - set to zero).
8	9050	Subsystem ID.
A	10B5	Subsystem Vendor ID.
C	0000	Max Latency and Min Grant (not loadable - set to zero).
E	0100	Interrupt Pin (not loadable - set to zero).
10	0FFC0000	Range for PCI to Local Address space 0 - set for 256KB & PCI memory mapped.
14	00000000	Range for PCI to Local Address space 1 - set to zero for none.
18	00000000	Range for PCI to Local Address space 2 - set to zero for none.
1C	00000000	Range for PCI to Local Address space 3 - set to zero for none.
20	00000000	Range for PCI to Local Expansion ROM - set to zero for none.
24	02000000	Local Base Addr for PCI to Local Addr space 0 - set for 256KB.
28	00000000	Local Base Addr for PCI to Local Addr space 1 - set to zero for none.
2C	00000000	Local Base Addr for PCI to Local Addr space 2 - set to zero for none.
30	00000000	Local Base Addr for PCI to Local Addr space 3 - set to zero for none.
34	00000000	Local Base Addr for PCI to Local Expansion Rom - set to zero for none.
38	10808045	Bus Region Descriptor for Local Addr Space 0 - set for Burst enables, Bterm input enabled, 1 wait state insertion in both read and write address to data timing, 32 bit bus. *
3C	00000000	Bus Region Descriptor for Local Addr Space 1 - set to zero.
40	00000000	Bus Region Descriptor for Local Addr Space 2 - set to zero.
44	00000000	Bus Region Descriptor for Local Addr Space 3 - set to zero.
48	00000000	Bus Region Descriptor for Expansion Rom - set to zero.
4C	01010001	Chip Select 0 Base and Range - set enabled and for 256K,
50	00000000	Chip Select 1 Base and Range - set disabled.
54	00000000	Chip Select 2 Base and Range - set disabled.
58	00000000	Chip Select 3 Base and Range - set disabled.
5C	00000043	Interrupt Control and Status register - enable local interrupt, active hi (same as ISA) and PCI interrupt.
60	18784B66	EEPROM Control and misc control - User I/O 0 output, set User I/O 0 Hi, User I/O 2 as CS2, User I/O 3 as CS3, PCI Read Mode - disconnect immediately, PCI retry Delay Clock set to 184 cycles, Read EPROM data bit, EPROM valid.

* For faster Xilinx chips (-2 or faster) consider using the following entry in lieu of that above:

38	00800005	Bus Region Descriptor for Local Addr Space 0 - set for Burst enables, Bterm input enabled, no wait state insertion, 32 bit bus.

4.1.2 Dual Regions.

In this mode the holding memory and the control/status registers occupy their own region in the PCI memory area. The memory claims a region of 128K bytes of memory and the control/status registers claim a region of 64bytes of memory. The interface card should then be configures so that the “CS0” line is set to control the memory, and the “CS1” line set to control the Xilinx operations.

7.0 Faults and Problems.

7.1. Design faults PCB version 1.0 5-97.

The following faults have been found.

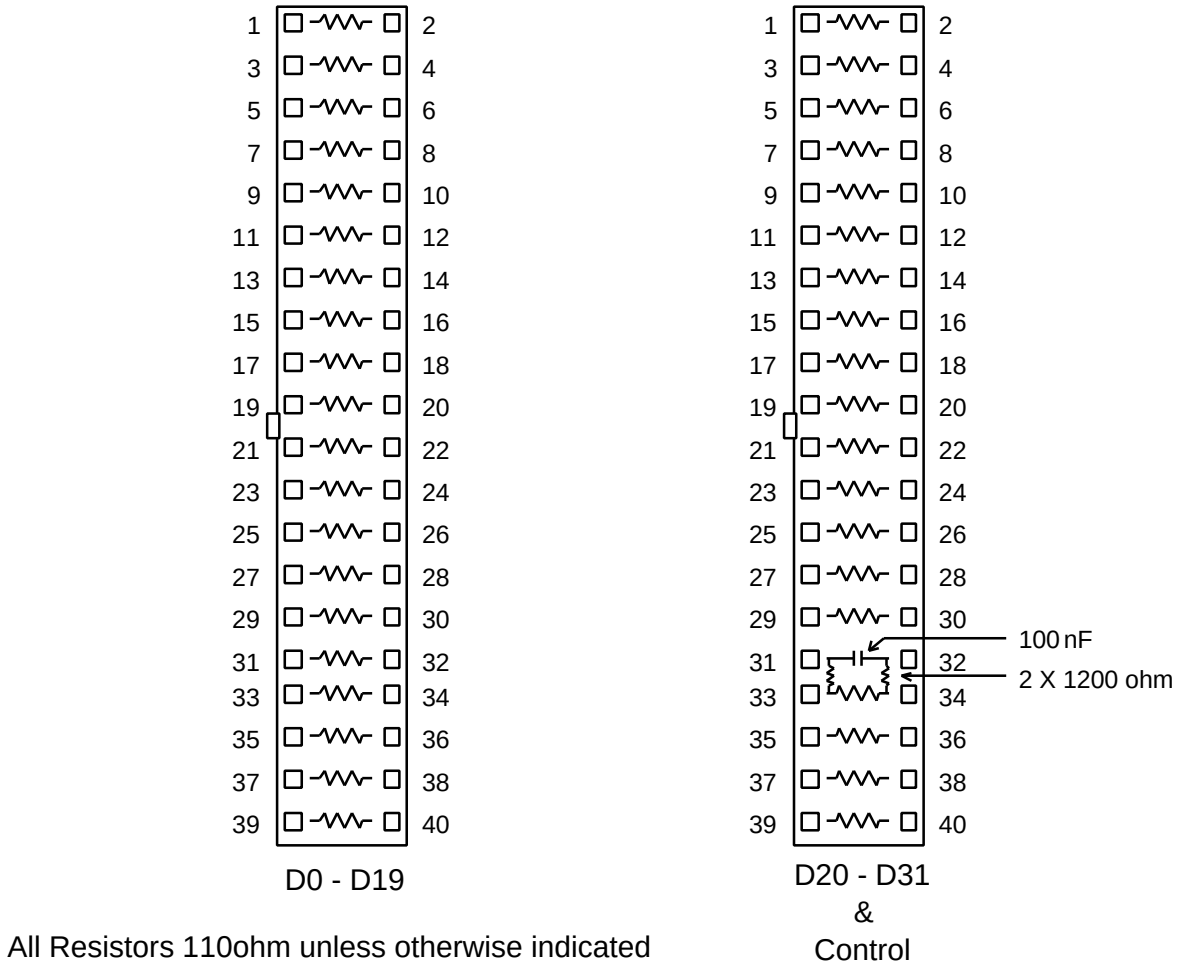
a) Short on PCI connector between pins B11 and B12. This appears to be a manufacturing fault. Cut the track with a sharp knife/blade. Confirm cut has been made by testing with an Ohm meter.
b) Track over-run on C134. Purely a cosmetic operation. Cut track with a sharp knife/blade.
c) Swap tracks to pin 1 and pin 2 on U2. Cut the track about midway between U2 and U1, and scrape of the solder resist on each of these tracks a short way either side of cut, tin exposed part of track with solder, and install changeover wires using #30G wirewrap wire. If necessary glue wires.
d) Various component and reference values not on correct side of board. Items fixed on next version.

8.0 Appendix.

8.1.1 Cable Termination.

The I/O Data Bus cable may require appropriate termination. When used with the AT correlators where multi-drop topology is used, the "~ACK" line must be pulled into a logic low state. This is done by providing a bias voltage onto an adjacent set of pins to the "~ACK" signal pair, and providing a set of pull resistors to the adjacent "~ACK" line. Links LK3 and LK4 should both be set to positions 1-2.

Correlator Bus Cable Terminators



8.2.1 Serial Number PROM data format.

The Serial Number PROM data format is the same as that used in the Event Generator, AT Distributed Clock and the BCC Interface card. Data is held as a preamble followed by a NULL terminated ASCII string (as per the C language). The preamble is any number of 1's, followed by a 0 and then seven other bits (don't cares). The next bit is the start of the data. Each ASCII character is 8 bits and is read from the PROM least significant bit first.

8.2.2. Code example.

```
/*****************************************************************
        Control and Status register.
*****************************************************************/
#define CS_REG                          0x0     /* master control/status    */
                                                /* register offset          */
#define   CS_TEST_DATA                  0x00001 /* Use test data            */
#define   CS_SERIAL_PROM_ENABLE         0x00002 /* Enable serial num prom   */
#define   CS_24_BIT_BUS                 0x00004 /* Operate in 24 bit mode   */
#define   CS_MEMORY_UPPER_HALF          0x00008 /* Use top half of memory   */
```

```
#define  CS_AUX_DATA_OUTPUT_MASK    0x00f00  /* Bits for aux output          */
#define  CS_AUX_DATA_INPUT_MASK     0x0f000  /* Bits for aux input           */
#define  CS_XFER_START              0x10000  /* w - Initiate xfer            */
#define  CS_XFER_UNDERWAY           0x10000  /* r - Xfer underway            */
#define  CS_RESET_XFER_PROM         0x20000  /* w - reset xfer/ Serial       */
                                             /*       number clock           */
#define  CS_SERIAL_NUM_DATA         0x20000  /* r - Serial number data       */
#define  CS_SELF_INTERRUPT          0x40000  /* w - Initiate self intrupt    */

#define  CS_AMBIQUITY_MASK       (CS_RESET_XFER_PROM |
                                  CS_XFER_START |
                                  CS_SELF_INTERRUPT) /* all the bits      */
                                             /* that are either dual         */
                                             /* function or write            */
                                             /* only.                        */

#define MAX_IDENT_LENGTH       80 /* max length of ident string */

/*****************************************************************************
*
* Get the firmware version/serial number string from the serial number PROM.
* NOTE- This function assumes exclusive use of the interface.
*
* RETURNS: 0 if OK, or an error code otherwise
*
*****************************************************************************/

int GetSerial
(
SysInfo_struct *sysinfo   /* info on the interface card */
)
{
int delay;
int n;
int bit;
volatile int *addr;

#define PROM_DELAY 5

addr = (int *) ((*sysinfo).BaseAddr + CS_REG);

   /* reset version/serial number PROM. This is done by disabling it  */
   *addr = 0;
   udelay(PROM_DELAY);

   /* enable  the PROM. */
   *addr = (*addr |
            CS_SERIAL_PROM_ENABLE) &
     (~(CS_RESET_XFER_PROM | CS_XFER_UNDERWAY));
```

```
    delay(PROM_DELAY); /* delay is a program that does what it says! */

    /* version/serial number preamble consists of any number of 1's,
       followed by a 0, followed by seven more bits, so first find the 0 bit
       NOTE- before reading each bit insert a delay because some
       PROMS are fairly slow. 5uS should be enough.
       This code only failed when there was no delay. */
#define MAXBITS 1000
    for(n = 0; n < MAXBITS; n++) {
      delay(PROM_DELAY);
      if((*addr & CS_SERIAL_NUM_DATA) == 0)
           break;

      /* increment PROM */
      *addr = (*addr | CS_RESET_XFER_PROM) & ~(CS_XFER_UNDERWAY);
    }

    /* insure 0 bit was found (not MAXBITS exceeded) */
    if(n >= MAXBITS) {
      /* disable the PROM */
      *addr = *addr &
           (~(CS_SERIAL_PROM_ENABLE |
             CS_RESET_XFER_PROM |
             CS_XFER_UNDERWAY));
      return(ER_PCIIFPreambleError);
    }

    /* skip remaining bits in preamble */
    for(n = 0; n < 8; n++) {
      delay(PROM_DELAY);
      *addr = (*addr | CS_RESET_XFER_PROM) & (~CS_XFER_UNDERWAY);
    }

    /* now read characters until null (indicating end-of-string) is found */
    for(n = 0; n < MAX_IDENT_LENGTH; n++) {
      /* get next character */
      (*sysinfo).PCIIFIdentString[n] = '\0';
      for(bit = 0; bit < 8; bit++) {
           delay(PROM_DELAY);
           if((*addr & CS_SERIAL_NUM_DATA) != 0)
             (*sysinfo).PCIIFIdentString[n] |= 1 << bit;
           *addr = (*addr | CS_RESET_XFER_PROM) & ~(CS_XFER_UNDERWAY);
      }
      /* printk("<1>Num chars = %d\n", n); */
      /* jump out if it's string terminator */
      if((*sysinfo).PCIIFIdentString[n] == '\0') break;
    }
```

```
   /* disable the PROM */
   *addr = *addr &
     (~(CS_SERIAL_PROM_ENABLE |
          CS_RESET_XFER_PROM |
          CS_XFER_UNDERWAY));

   return((n >= MAX_IDENT_LENGTH) ? ER_PCIIFSerNumTooLong : 0);
}
```

8.3. Data Transfer.

8.3.1. General

Data can be easily transferred from the Correlator to the on board memory. Depending upon the operating system being used, this memory may need to be mapped to the system virtual memory map.

The transfer itself requires 4 steps:

- load the Transfer Length Register,
- load the Start Address Register,
- enable the appropriate interrups (if used).
- set the Initiate Transfer bit in the Control/Status Register.
- wait for the transfer to complete. This may be interrupt driven or polled.

An important point to note when using transfer completion interrupts, is that for short transfer length, a race condition may occur between initiating the transfer and the operation to wait for the transfer completion interrupt.

8.3.2. Code Example.

```
/*************************************************************************
        Control and Status register.
*************************************************************************/
#define CS_REG                                  0x0       /* master control/status      */
                                                          /* register offset            */
#define   CS_TEST_DATA                          0x00001   /* Use test data              */
#define   CS_SERIAL_PROM_ENABLE                 0x00002   /* Enable serial num prom     */
#define   CS_24_BIT_BUS                         0x00004   /* Operate in 24 bit mode     */
#define   CS_MEMORY_UPPER_HALF                  0x00008   /* Use top half of memory     */
#define   CS_AUX_DATA_OUTPUT_MASK               0x00f00   /* Bits for aux output        */
#define   CS_AUX_DATA_INPUT_MASK                0x0f000   /* Bits for aux input         */
#define   CS_XFER_START                         0x10000   /* w - Initiate xfer          */
#define   CS_XFER_UNDERWAY                      0x10000   /* r - Xfer underway          */
#define   CS_RESET_XFER_PROM                    0x20000   /* w - reset xfer/ Serial     */
                                                          /*       number clock         */
#define   CS_SERIAL_NUM_DATA                    0x20000   /* r - Serial number data     */
#define   CS_SELF_INTERRUPT                     0x40000   /* w - Initiate self intrupt  */

#define   CS_AMBIQUITY_MASK         (CS_RESET_XFER_PROM |
                                     CS_XFER_START |
```

```
                                        CS_SELF_INTERRUPT) /* all the bits      */
                                                           /* that are either dual */
                                                           /* function or write    */
                                                           /* only.                */

/*********************************************************************
    Interrupt Control and Staus register.
    These registersshare many common bits.
*********************************************************************/
#define IC_REG                  0x4     /* Interrupt control register          */
#define IS_REG                  0x8     /* Interrupt status register           */
#define    ICS_XFER_COMPLETE    0x0001  /* Transfer Complete interrupt         */
#define    ICS_TIMEOUT          0x0002  /* Timeout interrupt                   */
#define    ICS_EXTERNAL_0       0x0004  /* External #0 interrupt               */
#define    ICS_EXTERNAL_1       0x0008  /* External #1 interrupt               */
#define    ICS_AUXILLARY_0      0x0010  /* Auxillary #0 interrupt              */
#define    ICS_AUXILLARY_1      0x0020  /* Auxillary #1 interrupt              */
#define    ICS_AUXILLARY_2      0x0040  /* Auxillary #2 interrupt              */
#define    ICS_AUXILLARY_3      0x0080  /* Auxillary #3 interrupt              */
#define    ICS_SELF             0x0100  /* Self interrupt                      */
#define    IC_ENABLE            0x8000  /* enable interrupts                   */

/*********************************************************************
    Length register.
*********************************************************************/
#define LENGTH_REG          0xc     /* length register                     */
#define    LENGTH_MASK          0x3fff  /* mask to indicate uses bits          */

/*********************************************************************
    Source address register.
*********************************************************************/
#define  ADDR_REG           0x10    /*  target start address register      */
#define    ADDR_MASK            0xffffffff  /* address register mask          */

/*********************************************************************
    Start of onboard memory.
*********************************************************************/
#define    BUFFER_START    0x20000         /* start of xfer buffer            */


int ReadCorrelator(
struct SysInfo *sysinfo,  /* contains all the details of the card including base address */
int StartAddr,            /* the correlator memory start address  */
int *UserBuffer,          /* pointer to the user's buffer */
int Length                /* the number of words to get. UserBuffer must be big enough */
                          /* to receive this number of words */
)
{
int err;      /* error value */
```

```
volatile int *addr;  /* a pointer to make reading the code easier. You may also need to */
                     /* make it 'volatile' to be sure that it is read each time it is accessed */

        /* setup start address register. (*sysinfo).BaseAddr is the remapped PCI address */
        addr = (int *) ((*sysinfo).BaseAddr + ADDR_REG);
        *addr = StartAddr;

        /* setup length register */
        addr = (int *) ((*sysinfo).BaseAddr + LENGTH_REG);
        *addr = Length;

        /* turn on the relevant interrupt lines. We won't use timeout interrupt. */
        /* Timeout can be incorporated in the WaitFor Interrupt() routine */
        addr = (int *) ((*sysinfo).BaseAddr + IC_REG);
        *addr = *addr | ICS_XFER_COMPLETE | IC_ENABLE;

        /* initiate the transfer. Don't forget to mask out the other ambiguous bits in */
        /* the CSR */
        *addr = (*addr & (~CS_RESET_XFER_PROM)) |
                                        CS_XFER_START;

        /* wait for the transfer complete interrupt with timeout for 1 second */
        err = WaitForInterrupt(1000);
        if(err != 0) return(err);  /* if return value is not 0 then error occurred */

        /* Now transfer data to users buffer */
        addr = (int *) ((*sysinfo).BaseAddr + BUFFER_START);
        memcpy(UsersBuffer,  addr, Length * sizeof(int));

        return(0);
}
```

%% END %%