

DYNAMIC ENGINEERING

150 DuBois St., Suite C, Santa Cruz, CA. 95060

831-457-8891, Fax 831-457-4793

sales@dyneng.com

www.dyneng.com

Est. 1988

User Manual

PCI ASCB Bus Versions D/E Avionics Bus Interface

**2-channel Bi-directional
Transformer coupled ASCB interface
PCI Board and PMC Module Carrier**

Revision E1

Corresponding Hardware: Revision A, B, C, D

Current Fab: 10-2008-120(1-4)

Corresponding Firmware: Revision D

**PCI-ASCB-DE Avionics Bus Tester
2-Channel Bi-Directional
PCI Board and PMC Module Carrier**

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

©2008-2012 by Dynamic
Engineering.
Other trademarks and
registered trademarks are
owned by their respective
manufactures.
Manual Revision E1 Revised
March 31, 2012

This document contains information of proprietary interest to Dynamic Engineering. It has been supplied in confidence and the recipient, by accepting this material, agrees that the subject matter will not be copied or reproduced, in whole or in part, nor its contents revealed in any manner or to any person except to meet the purpose for which it was delivered.

Dynamic Engineering has made every effort to ensure that this manual is accurate and complete. Still, the company reserves the right to make improvements or changes in the product described in this document at any time and without notice. Furthermore, Dynamic Engineering assumes no liability arising out of the application or use of the device described herein.

The electronic equipment described herein generates, uses, and can radiate radio frequency energy. Operation of this equipment in a residential area is likely to cause radio interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Dynamic Engineering's products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Dynamic Engineering.

This product has been designed to operate with PMC Modules and compatible user-provided equipment. Connection of incompatible hardware is likely to cause serious damage.



Table of Contents

PRODUCT DESCRIPTION	6
THEORY OF OPERATION	7
ADDRESS MAP	10
PROGRAMMING	11
REGISTER DEFINITIONS	12
BASE_CONTROL	12
BASE_STATUS	14
BASE_FRAME_END	15
BASE_FRAME_INT	15
BASE_FRAME_CNT	15
CHAN_CONTROL	16
CHAN_STATUS	18
CHAN_TX_DMA	19
CHAN_RX_DMA	20
CHAN_TX_INDX	20
CHAN_RX_INDX	21
TX_SM_MEM	22
RX_SM_MEM	23
PCI-ASCB IO PIN ASSIGNMENT	24
APPLICATIONS GUIDE	25
Interfacing	25
Construction and Reliability	26
Thermal Considerations	26
Warranty and Repair	27
Service Policy	27
Out of Warranty Repairs	27
SPECIFICATIONS	28





List of Figures

FIGURE 1	PCI-ASCB BLOCK DIAGRAM	6
FIGURE 2	PCI-ASCB MANCHESTER TIMING DIAGRAM	7
FIGURE 3	PCI-ASCB 8B10B DIAGRAM	8
FIGURE 4	PCI-ASCB INTERNAL ADDRESS MAP	10
FIGURE 5	PCI-ASCB BASE CONTROL REGISTER BIT MAP	12
FIGURE 6	PCI-ASCB STATUS REGISTER	14
FIGURE 7	PCI-ASCB FRAME END COUNT REGISTER	15
FIGURE 8	PCI-ASCB FRAME INTERRUPT COUNT REGISTER	15
FIGURE 9	PCI-ASCB CURRENT FRAME COUNT REGISTER	15
FIGURE 10	PCI-ASCB CHANNEL CONTROL REGISTER	16
FIGURE 11	PCI-ASCB CHANNEL STATUS REGISTER	18
FIGURE 12	PCI-ASCB CHANNEL WRITE DMA POINTER PORT	19
FIGURE 13	PCI-ASCB CHANNEL READ DMA POINTER PORT	20
FIGURE 14	PCI-ASCB CHANNEL TRANSMIT INDEX REGISTER	20
FIGURE 15	PCI-ASCB CHANNEL RECEIVE INDEX REGISTER	21
FIGURE 16	ASCB CONNECTOR PINOUT	24

Product Description

PCI-ASCB-DE is a special-purpose PCI board designed to interface with the Avionics Standard Communication Bus versions D and E. The interface can be used for a variety of applications including the testing avionics components.

PCI-ASCB-DE has an on-board PCI bridge that creates a local two-slot PCI bus. Slot 0 has a XC3S2000 Xilinx FPGA that implements a two-channel ASCB-D/E interface complete with a programmable PLL based frame-count timing standard. Two nine-pin D-connectors on the PCI front panel connect to the two ASCB channels, each with primary and back-up transformer-coupled bus I/O with fail-safe signal disconnect shunts and individual 16 KByte frame data memory for the transmit and receive functions.

Slot 1 is a PMC slot for an optional PrPMC (Processor PCI Mezzanine Card) processor module. The Pn4 PMC connector connects to an RJ45 ethernet connector on the PCI front panel. The ASCB interrupt can be configured to be serviced by either the local processor or the PCI host computer.

Along with the primary and back-up differential data busses, each nine-pin connector has two active-low bus disables; one for each of the primary and back-up busses and a frame sync pulse output that indicates the start of a data-frame for bus test analysis purposes.

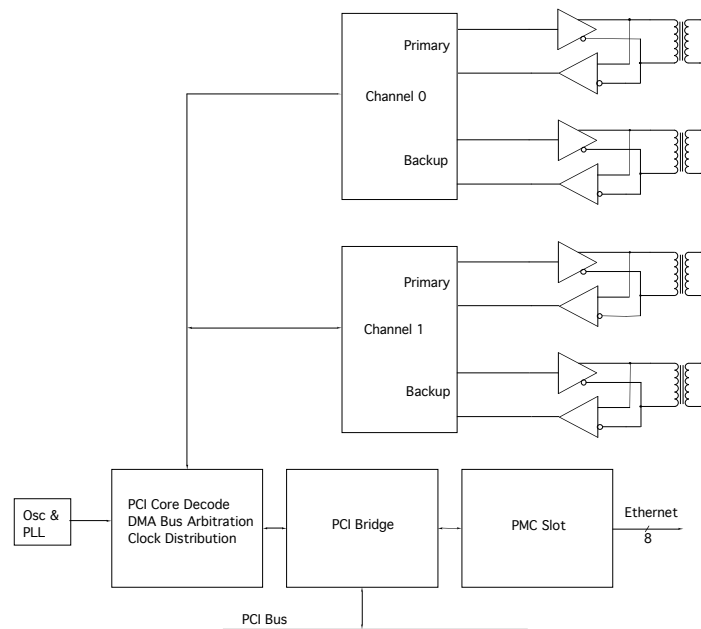


FIGURE 1

PCI-ASCB BLOCK DIAGRAM

Theory of Operation

PCI-ASCB is designed for transferring data from one point to another with a serial protocol. Currently two of the ASCB protocols are implemented. Type “D” and type “E”.

PCI-ASCB features a Xilinx FPGA. The FPGA contains all of the registers and protocol controlling elements of the design. Only the transformers, transceivers, and switches are external to the Xilinx device.

The PCI interface to the host CPU is controlled by a logic block within the Xilinx. The design requires one wait-state for read or write cycles to any address.

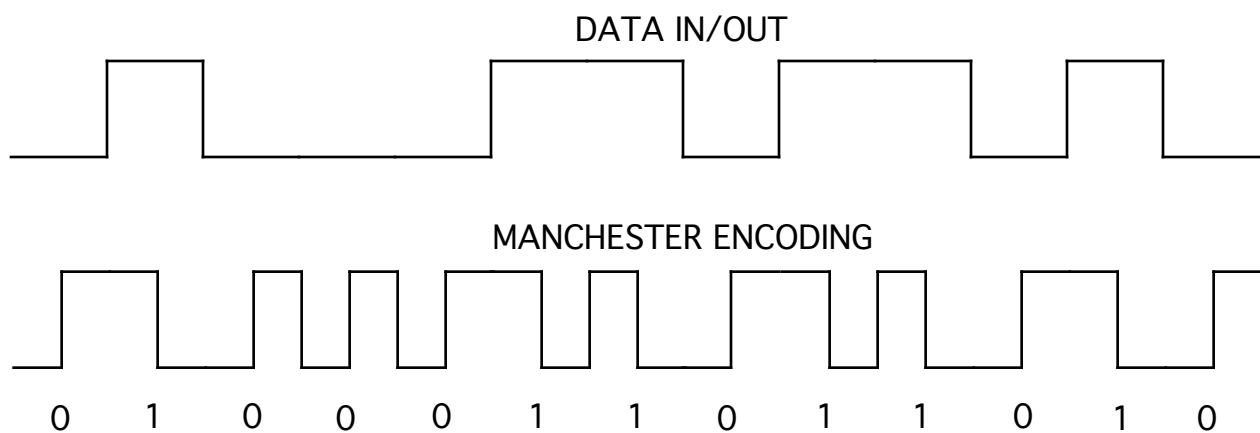


FIGURE 2

PCI-ASCB MANCHESTER TIMING DIAGRAM

PCI-ASCB can support many protocols. The D Type uses Manchester serial encoded data and clock. Data is sent in 16-bit words concatenated for multiple word transfers. The Manchester timing is shown in Figure 2. The E Type uses 8B10B encoding and operates at 20 MHz. See Figure 3 for the message format.

State machines within the FPGA control all transfers between the internal DPR and FPGA logic, and the FPGA and the data buffers. The TX state machine reads from the transmit memory and loads the shift-register before sending the data. The RX state machine receives data from the data buffers and takes care of moving data from the shift-register into the RX memory.

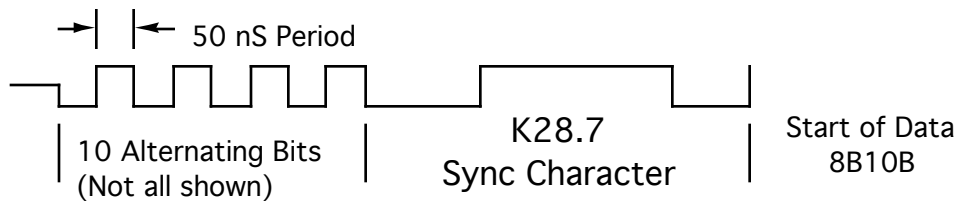


FIGURE 3

PCI-ASCB 8B10B DIAGRAM

The E type interface starts with the transmitter coming out of tri-state and driving the bus with an alternating pattern of 1's and 0's for 20 bit periods. The preamble is followed by the K character (K28.7) which is encoded as shown. Data follows the sync character and continues until the next to last position. The data is assumed to be on 32 bit boundaries with the last 32 bits being the CRC for the message. Data is encoded and decoded based on the 8b10b standard. No control characters are contained in the message body. The CRC is calculated using the same method as the D type and applied to all of the data in the body of the message.

The hardware [when receiving] determines the lack of new data arriving and terminates the current message reception.

The K character has two encodings, and since the new message always starts with the same running disparity, the encoding for the initial position is always the same. The Hardware resets the disparity at the start of each message to enforce this behavior.

The control interface and memory operating procedure is the same for both versions. A single bit is switched to change from D to E and vice-versa. In addition this version allows for the primary and redundant data to be captured in parallel. The previous model required the choice of primary or redundant. The single channel larger memory allocation is also under software control.

Custom interfaces are available. We will redesign the state machines and create a version for your specific requirements. That protocol will then be offered as a "standard" special order product. Please see our web page for current protocols offered. Please contact Dynamic Engineering with your custom application.

The ASCB-D/E implementations have 4 – Dual-Port RAMs (DPR) using the internal block RAM of the Xilinx. Each channel has two associated DPR's. Each DPR is configured to have a 32-bit port on the PCI side, and a 32-bit port on the IO side.

All configuration registers support read and write operations for maximum software convenience, and all addresses are long word aligned.



Interrupts are supported by PCI-ASCB. An interrupt can be configured to occur at the end of a transmitted packet or message. An interrupt can be set at the end of a received packet. All interrupts are individually maskable, and a master interrupt enable is also provided to disable all interrupts simultaneously.

PCI-ASCB-DE is interconnected to other equipment via cable. The cable can have multiple taps and is terminated at the end points. The terminations are 124 ohms each providing a 62 ohm parallel impedance. Ideally the cable will match this as well. PCI-ASCB-DE uses transformer coupling with a 1:1 transformer. The cable side has 30.1 ohm stub resistors to conform to other fielded standard equipment and approximate the impedance on the cable. The FPGA is isolated from the transformer with RS485 transceivers. The transceivers are terminated with 30.1 ohm series resistors to match impedances. In addition a low pass filter is installed into each leg to reduce the edge rate of the signal presented to the cable. The filters are set to approximately 80 MHz. on each side to allow the 20 MHz signal to pass without much distortion but reduce the edge rate to help with reflected noise on long cables.

PCI-ASCB-DE has been tested using flight cables of approximately 100 Ft.

Address Map

BAR 0:	<u>Offset</u>	<u>Reg num</u>	<u>Function</u>
BASE_CNTRL	0x0000	0	Base control register offset
BASE_STATUS	0x0004	1	Base status register offset
BASE_FRAME_END	0x0008	2	Base frame clock end count register offset
BASE_FRAME_INT	0x000C	3	Base frame clock interrupt count register offset
BASE_FRAME_CNT	0x0010	4	Base current frame count read port offset
CHAN_0_CNTRL	0x0018	6	Channel 0 control register offset
CHAN_0_STATUS	0x001C	7	Channel 0 status register offset
CHAN_0_TX_DMA	0x0020	8	Channel 0 write DMA start register offset
CHAN_0_RX_DMA	0x0024	9	Channel 0 read DMA start register offset
CHAN_0_TX_INDXX	0x0028	10	Channel 0 Transmitter index read port offset
CHAN_0_RX_INDXX	0x002C	11	Channel 0 Receiver index read port offset
CHAN_1_CNTRL	0x0034	13	Channel 1 control register offset
CHAN_1_STATUS	0x0038	14	Channel 1 status register offset
CHAN_1_TX_DMA	0x003C	15	Channel 1 write DMA start register offset
CHAN_1_RX_DMA	0x0040	16	Channel 1 read DMA start register offset
CHAN_1_TX_INDXX	0x0044	17	Channel 1 Transmitter index read port offset
CHAN_1_RX_INDXX	0x0048	18	Channel 1 Receiver index read port offset

BAR 1: I/O channel RAM blocks are mapped to this space for single-word access

CHAN_0_TX_RAM	0x0000 – 0x3FFC
CHAN_0_RX_RAM	0x4000 – 0x7FFC
CHAN_1_TX_RAM	0x8000 – 0xBFFC
CHAN_1_RX_RAM	0xC000 – 0xFFFF

FIGURE 4

PCI-ASCB INTERNAL ADDRESS MAP

The address map provided is for the local decoding performed within the PCI-ASCB. The addresses are all offsets from a base address, which is assigned by the system when the PCI bus is configured.

Programming

Programming the PCI-ASCB requires only the ability to read and write data from the host. The base address is determined during system configuration of the PCI bus. The base address refers to the first user address for the slot in which the PMC is installed.

Depending on the software environment, it may be necessary to set-up the system software with the PCI "registration" data. For example in WindowsNT there is a system registry, which is used to identify the resident hardware.

In order to receive data, the software is only required to enable the Rx channel and set the frequency parameters. To transmit the software will need to load the message into the appropriate channel Dual Port RAM, set the frequency and mode and enable the transmitter.

The interrupt service routine should be loaded and the interrupt mask set. The interrupt service routine can be configured to respond to the channel interrupts on an individual basis. After the interrupt is received, the data can be retrieved. An efficient loop can then be implemented to fetch the data. New messages can be received even as the current one is read from the Dual Port RAM.

The TX interrupt indicates to the software that a message has been sent and that the message has completed. If more than one interrupt is enabled, then the SW needs to read the status to see which source caused the interrupt. The status bits are latched, and are explicitly cleared by writing a one to the corresponding bit. It is a good idea to read the status register and write that value back to clear all the latched interrupt status bits before starting a transfer. This will insure that the interrupt status values read by the interrupt service routine came from the current transfer.

Refer to the Theory of Operation section above and the Interrupts section below for more information regarding the exact sequencing and interrupt definitions.

The VendorId = 0x10EE. The CardId = 0x0037. Current FLASH revision = 0x0D



Register Definitions

BASE_CONTROL

[0x00] ASCB-D Base Control Register (read/write)

Control Register	
DATA BIT	DESCRIPTION
31-24	Spare
23	PLL sdat output
22	PLL s2
21	PLL sclk
20	PLL enable
19-8	Spare
7	Channel 1 Rx Interrupt Enable
6	Channel 1 Tx Interrupt Enable
5	Channel 0 Rx Interrupt Enable
4	Channel 0 Tx Interrupt Enable
3	Frame Count Loop Enable
2	Frame Interrupt Enable
1	Frame Count Clear
0	Frame Count Enable

FIGURE 5

PCI-ASCB BASE CONTROL REGISTER BIT MAP

All bits are active high and are reset on power-up or reset command except the PLL enable bit which resets to a high state for PLL initialization.

PLL sclk/sdata output: These signals are used to program the PLL over the I2C serial interface. SCLK is always an output whereas sdata is bi-directional.

PLL s2 output: This is an additional control line to the PLL that can be used to select additional pre-programmed frequencies.

PLL enable: When this bit is set to a one, the sdat output signal is enabled. When set to a zero the sdat signal is tri-stated by the Xilinx.

The register bits for PLL enable, PLL s2, PLL sclk are unidirectional from the Xilinx to the PLL – always driven. PLL sdat is a bi-directional signal (open drain). The sdat register bit, when written low and enabled will be reflected with a low on the sdat signal to the PLL. When sdat is taken high or disabled the sdat signal will be tri-stated by the Xilinx, and can be driven by the PLL. The sdat register bit when read reflects the state of the internal sdat bit but may not be in the same state as the signal between the Xilinx and PLL. To determine the state of the external sdat signal, read the bit from the

The PLL is a separate device controlled by the Xilinx. The PLL has a fairly complex programming requirement which is simplified by using the Cypress® frequency descriptor software <http://www.dyneng.com/CyberClocks.zip>, and then programming the resulting control words into the PLL using the PLL Control ports. The interface can be further simplified by using the Dynamic Engineering Driver to take care of the programming requirements.

If you are writing your own driver, contact Dynamic Engineering and we can send you a file with code excerpts from our driver and test software that cover each step of the process from parsing the .jed file to the low-level bit manipulation of the I²C bus.

Channel 1 Rx Interrupt Enable:

Channel 1 Tx Interrupt Enable:

Channel 0 Rx Interrupt Enable:

Channel 0 Tx Interrupt Enable:

Frame Count Loop Enable:

Frame Interrupt Enable:

Frame Count Clear:

Frame Count Enable:



BASE_STATUS

[0x04] ASCB-D Base Status Register (read status/write clear)

Status Register	
DATA BIT	DESCRIPTION
31	Interrupt Active (read only)
30-24	Spare
23	PLL sdat input (read only)
22-21	Spare
20	Frame Interrupt Status (read/write clear)
19	Channel 1 Rx Interrupt Status (read/write clear)
18	Channel 1 Tx Interrupt Status (read/write clear)
17	Channel 0 Rx Interrupt Status (read/write clear)
16	Channel 0 Tx Interrupt Status (read/write clear)
15-8	Xilinx Flash Revision (read only)
7-0	User Switch 7-0 (read only)

FIGURE 6

PCI-ASCB STATUS REGISTER

The Flash revision for the ASCB-DE project is 0x0E. The FLASH ID will be updated as features are added or revisions made. The revision is also readable from the configuration space revision register.

The Switch Read Port has the user bits. The user bits are connected to the eight dip-switch positions. The switches allow custom configurations to be defined by the user and for the software to identify a particular board by its switch settings and to configure it accordingly.

The Dip-switch is marked on the silk-screen with the positions of the digits and the '1' and '0' definitions. The numbers are hex coded. The example shown would produce 0x12 when read [and shifted down].

Channel interrupts are duplicated here for user convenience. The interrupts can be operated from either location.

BASE_FRAME_END

[0x08] ASCB-D Frame End Count Register (read/write)

Frame End Count Register	
DATA BIT	DESCRIPTION
31-18	Spare
17-0	Frame End Count

FIGURE 7

PCI-ASCB FRAME END COUNT REGISTER

BASE_FRAME_INT

[0x0C] ASCB-D Frame Interrupt Count Register (read/write)

Frame Interrupt Count Register	
DATA BIT	DESCRIPTION
31-18	Spare
17-0	Frame Interrupt Count

FIGURE 8

PCI-ASCB FRAME INTERRUPT COUNT REGISTER

BASE_FRAME_CNT

[0x10] ASCB-D Frame Count Register (read only)

Current Frame Count Register	
DATA BIT	DESCRIPTION
31-18	Spare
17-0	Current Frame Count

FIGURE 9

PCI-ASCB CURRENT FRAME COUNT REGISTER

CHAN_CONTROL

[0x18, 0x34] ASCB-D Channel 0, 1 Control Register (read/write)

Channel Control Register	
DATA BIT	DESCRIPTION
17	Encoding
16	RxIntEn1
15	RxSources
14	RxStart1
13	InternalCrc
12	Init
11-10	Spare
9	Channel Force Interrupt
8	Channel Master Interrupt Enable
7	Receive DMA Enable
6	Transmit DMA Enable
5	RxIntEn0
4	TxIntEn
3	Manchester Data Invert
2	Spare
1	RxStart0
0	TxEn

FIGURE 10

PCI-ASCB CHANNEL CONTROL REGISTER

[TxEn](#) when set '1' enables the Transmitter to operate in accordance with the programmed options contained in the DPR and this register.

[RxStart0](#), [RxStart1](#) when set '1' enable the two receive ports per channel separately. Both the Primary and the Secondary data streams can be directed to the 0 port using the [RxSources](#) and [RxSourceSelect](#) bits. Alternatively both the Primary(0) and Secondary(1) can be received in parallel.

[RxSourceSelect](#) is used to select which channel [Primary '0' Secondary '1'] are received by the lower memory section. This bit only has meaning when the channel is in single port mode. This bit has been removed with the update to the "DE" level design.

[RxSources](#) is used to select between 1 receive channel and 2 receive channels. The full receive DPR for the channel is used for the single port when cleared = LargeMap model. When set '1' the small map version is used where the DPR is split in half with [RxStart0](#) – primary used to load the lower half of the DPR and the secondary input plus [RxStart1](#) used to load the upper half of the DPR allocated to the channel.

When in LargeMap mode use only one RxStart bit at a time. Which ever is enabled will determine which data is captured. RxStart0 will capture the primary and RxStart1 will capture the secondary channel. Both data sets will be loaded to the full DPR range.



Enabling both channels will cause data overwrite with “interesting” results.

Manchester Data Invert is used to optionally invert the transmit and receive data to be compatible with non-standard equipment. The default is ‘0’ and usually is not needed. If the other equipment is not receiving what this board is sending and vice-versa, try this bit. No effect on 8B10B data.

TxIntEn, RxIntEn0, RxIntEn1 are enables to allow interrupts from the transmitter, and each of the receivers. Set to enable interrupts.

Transmit DMA Enable, Receive DMA Enable are set to allow DMA operation into and out of the DPR associated with the transmit and receive functions in the channel. The driver will normally manage these bits.

Channel Master Interrupt Enable must be set to allow the user interrupts to be asserted. This bit allows for a quick on/off control for the interrupts without individually disabling them.

Channel Force Interrupt when set causes an interrupt from the channel. Clear to remove the interrupt. Used for test purposes and SW development. You can set to simulate an interrupt from the channel to test your ISR etc.

Init acts as a local reset for the channel. Active when set. Normal operation when cleared. Returns State Machines etc. to their Idle or default states.

InternalCrc is used to control the insertion of the internally calculated CRC into the receive data stream. We used this option to help debug the calculated versus received CRC when doing the initial integration. When the CRC error bit is not set in the message status the received CRC matches the internally generated CRC.

Encoding selects between the Manchester “D” type operation and the 8B10B “E” type. The default is D type (0). When set ‘1’ the “E” type is enabled.



CHAN_STATUS

[0x1C, 0x38] ASCB-D Channel 0, 1 Status Register (read status/write clear)

Channel Status Register	
DATA BIT	DESCRIPTION
31	Interrupt Active
30-14	Spare
13	RxPacket1
12	RxIntLat1
11	Receive DMA Ready
10	Transmit DMA Ready
9	Receive DMA Error
8	Transmit DMA Error
7	Receive DMA Interrupt
6	Transmit DMA Interrupt
5	RxIntLat0 (read/write clear)
4	TxIntLat (read/write clear)
3	Spare
2	RxPacket0
1	Receiver Active
0	Spare

FIGURE 11

PCI-ASCB CHANNEL STATUS REGISTER

Spare bits are tied to '0'.

[Receiver Active](#) is an additional read-back location for the RxStart0 bit. When '1' in the control register it will be '1' in this status as well and vice-versa.

[RxPacket0](#), [RxPacket1](#) indicate that a packet receive is in progress on that channel.

[TxIntLat](#), [RxIntLat0](#), [RxIntLat1](#) when set indicate that an interrupt is active for the respective port. Clear by writing back to the bit with a '1'. Enabled in the control register. The enable is anded after the status latch allowing for polled operation using the status instead of being interrupt based.

[Transmit DMA Interrupt](#), [Receive DMA Interrupt](#) are set when the respective DMA operation completes. Usually handled by the driver.

[Transmit DMA Error](#), [Receive DMA Error](#) when set indicate that an error occurred during interrupt processing. Usually only seen during initial integration when pointers or commands are not quite right.

Transmit DMA Ready, Receive DMA Ready are status signals from the DMA state machines indicating that the DMA engines are back to the IDLE state and ready for a new command. Used when a DMA action is terminated and the next command should not be written until the DMA system “unwinds”.

Interrupt Active is a quick read status bit to indicate that no active user interrupts are present or conversely that something in the channel is interrupting. Software can check this bit to make a quick decision when processing interrupts and determining the source.

CHAN_TX_DMA

[0x20, 0x3C] ASCB-D Channel 0, 1 Tx DMA Register (write only)

Input DMA Pointer Address Port	
Data Bit	Description
31-0	First Chaining Descriptor Physical Address

FIGURE 12

PCI-ASCB CHANNEL WRITE DMA POINTER PORT

This write-only port is used to initiate a scatter-gather input DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads four successive long words beginning at that address. The first is the address of the first memory block of the DMA buffer, the second is the local RAM address to start writing that memory block into, the third is the length in bytes of that block [only the lower 22 bits are valid], the second is the local RAM address to start writing that memory block into, and the fourth is the address of the next chaining descriptor in the list of buffer memory blocks. This process is continued until the end-of-chain bit is set in one of the next pointer values.

CHAN_RX_DMA

[0x24, 0x40] ASCB-D Channel 0, 1 Rx DMA Register (write only)

Output DMA Pointer Address Port	
Data Bit	Description
31-0	First Chaining Descriptor Physical Address

FIGURE 13

PCI-ASCB CHANNEL READ DMA POINTER PORT

This write-only port is used to initiate a scatter-gather output DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads four successive long words beginning at that address. The first is the address of the first memory block of the DMA buffer, the second is the local RAM address to start reading that memory block from, the third is the length in bytes of that block [only the lower 22 bits are valid], and the fourth is the address of the next chaining descriptor in the list of buffer memory blocks. This process is continued until the end-of-chain bit is set in one of the next pointer values.

CHAN_TX_INDX

[0x28, 0x44] ASCB-D Channel 0, 1 Tx Index Register

Channel Transmit Index Register	
DATA BIT	DESCRIPTION
31-28	Spare
27-16	Starting RAM Address of Next Transmit DMA
15-12	Spare
11-0	Current Transmit Access RAM Address

FIGURE 14

PCI-ASCB CHANNEL TRANSMIT INDEX REGISTER

This register can be accessed to see where to write to and where data is currently being transmitted from. Since the memory is Dual Ported, the Next Data can be written before the current data is completed. Since the memory is circular in nature, the new data should not be written if it will overwrite the current block of data. Some pointer arithmetic is required for proper management of the TX side.

The transmit side uses the read DMA – the names are relative to the ASCB and for transmit the data is read from the system memory and written to local memory.

CHAN_RX_INDx

[0x2C, 0x48] ASCB-D Channel 0, 1 Rx DMA Register (read only)

Channel Receive Index Register	
DATA BIT	DESCRIPTION
31-28	Spare
27-16	Starting RAM Address of Next Output DMA
15-12	Spare
11-0	Current Receive Access RAM Address

FIGURE 15

PCI-ASCB CHANNEL RECEIVE INDEX REGISTER

This register can be accessed to see where to read from and where data is currently being received into. Since the memory is Dual Ported, the Previous Data can be read before the current data is completed. Since the memory is circular in nature, the new data should be read before new data will overwrite it. Some pointer arithmetic is required for proper management of the RX side.

TX_SM_MEM

In transmit the first two 32-bit words are the control word. The packet follows: Data, CRC. The CRC position should be set to zero as this is part of the CRC calculation. The hardware will calculate and insert the CRC.

```
nextstrt    <= control(11 downto 0);  
// pointer to the next transmitted block start location  
tpri_dis    <= control(12); // primary disable  
tbkp_dis    <= control(13); // secondary disable  
crc_dis     <= control(14); // CRC generation disable – use SW defined  
start_en    <= control(15); // Set to require use of start time  
stop_en     <= control(16); // set to require use of stop time  
strtcnt     <= control(34 downto 17); // time to start transmission  
stopcnt     <= control(52 downto 35); // stop on or before
```

The start time is required for the first block of data to be sent. After that block is sent as long as the transmitter is enabled and the next block is ready to be sent before the prior one is completed, the start_en can be set to '0'. In this case the second block is sent immediately after the first block based on hardware delays.

Memory configuration per block of data.

Location	IO	Value (32-bits)
0		lower command
1		upper command
4		First data
...		
N		last data
N+1		CRC or zero data

RX_SM_MEM

Each received data block has the first two 32-bit words used for status/control. The packet follows: Data, CRC. The CRC is the last 32 bit word received. The hardware will calculate the CRC on the received data and compare with the received value setting [or not] the CRC Error bit in the status word with the results.

Receiver Status (first two 32-bit words in packet)

```
status(11 downto 0)    <= rxend; // pointer to the start of the next block
status(16 downto 12)  <= num_bits; //
status(17)            <= bus_sel; // which port was received
status(18)            <= man_err; // Manchester error was detected
status(19)            <= crc_err; // CRC error was detected
status(37 downto 20)  <= strt_cnt; // start time of message received
status(55 downto 38)  <= stop_cnt; // end time of message received
status(63 downto 56)  <= x"AA"; // fixed termination pattern
```

PCI-ASCB IO Pin Assignment

The figure below gives the pin assignments for the IO Interface the ASCB connectors. J2 is the Channel 0 or Left port, and J3 is the Channel 1 or Right port.

PrimaryP	1
PrimaryN	6
PTXDIS	2
STXDIS	3
SecondaryP	4
SecondaryN	8
Frame	5
Grounds	7,9

FIGURE 16 ASCB CONNECTOR PINOUT

	J1(RJ45)	Pn4(Std)	Pn4(ALT)
TRD0P	1	1	45
TRD0N	2	3	47
TRD1P	3	7	46
TRD2P	4	2	49
TRD2N	5	4	51
TRD1N	6	9	48
TRD3P	7	8	50
TRD3N	8	10	52

Figure 17 Ethernet pinout

	J4(1x3 HDR)	Pn4(Std)	Pn4(ALT)
RXD	1	34	4
TXD	2	28	3
GND	3		

Figure 18 SERIAL PORT pinout

The RJ45 and Serial ports are included to support an installed PrPMC. Two pinouts are supplied to allow for PrPMC's which follow the PICMG standard and an alternate set based on our initial client's preferred PrPMC.

Applications Guide

Interfacing

The pin-out tables are displayed with the pins in the same relative order as the actual connectors. The pin definitions are defined with noise immunity in mind. The pairs are chosen to match the ASCB or Ethernet standard cable.

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

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 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.

Cables. The ASCB specification calls out a specific cable for use in ASCB systems. The cable is for a half duplex system with nodes interconnected linearly and terminations at the ends of the cable. The D type interface operates with Manchester encoding at 10 MHz bit rate. The E type operates with 8B10B encoding at 20 MHz. The interface is transformer isolated RS-485. The terminations are set to 124 ohms each or 62 ohm parallel impedance.

We provide the components. You provide the system. Safety and reliability can be achieved only by careful planning and practice. Inputs can be damaged by static discharge, or by applying voltage outside of the devices rated voltage.



Construction and Reliability

PCI Modules were conceived and engineered for rugged industrial environments. The PCI-ASCB is constructed out of 0.062 inch thick high temperature ROHS compliant material.

Surface mounted components are used. The components are available with commercial and Industrial temperature ranges. Please order the “ET” version for more demanding environments. Conformal coating is an option for condensing environments or for another measure of board protection. Please order the “CC” version.

The PCI is secured against the chassis with the connectors and front panel. If more security against vibration is required a chassis with top side support can be used. The PCI-ASCB has a wider keep out than required by PCI specification for most of the top edge to allow use in industrial chassis and horizontal mount situations.

The power and ground planes are implemented with relatively heavy copper to help with heat spreading in chassis with limited air flow. The components are spaced to allow for efficient cooling and power dispersion.

Thermal Considerations

The PCI-ASCB design consists of CMOS and similar circuits. The power dissipation due to internal circuitry is very low. It is possible to create higher power dissipation with the externally connected logic. If more than one Watt is required to be dissipated due to external loading; cooling with forced air is recommended. With the one degree differential temperature to the solder side of the board external cooling is easily accomplished.

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 \$125. 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
support@dyneng.com



Specifications

Logic Interface:	PCI 32/33 ↔ 64/66 lanes.
Digital Serial IO:	ASCB protocol versions D & E.
PMC Position	1 PMC position suitable for PrPMC or standard PMC operation.
DIP Switch:	DipSwitch supplied for board identification and other user purposes.
CLK rates supported:	PLL is programmed to select the frame clock rate. 40 Mhz local oscillator used as reference for RX and TX of Manchester and 8b10b data. PLLB reserved for new applications.
Software Interface:	ASCB Control Registers, IO registers, IO Read-Back registers, FIFO. R/W, 32 bit boundaries.
Initialization:	Programming procedure documented in this manual
Access Modes:	LW to registers, read-write to most registers
Interrupt:	Each port has independently programmable interrupt sources, DMA interrupts included.
Onboard Options:	All Options are Software Programmable
Interface Options:	2 DB9 Pin connectors at front bezel for ASCB channels. Additional RJ45 to support Ethernet or other interface from PrPMC. Serial port header supplied on board.
Dimensions:	Standard 1/2 length PCI module.
Construction:	Multi-Layer Printed Circuit, Through Hole and Surface Mount Components.
Power:	PMC supplied with standard voltages. Rest of ASCB function uses 5V, 3.3V and locally generated voltages for the FPGA.
Weight:	TBD oz



Order Information

PCI-ASCB PCI Module with 2 redundant ASCB channels. ASCB-D and ASCB-E supported. Software control over mode of operation and operational parameters.

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

Order Options:

Pick any combination to go with IO

-CC to add conformal coating

-ET to change to industrial Temp [-40 - +85C]

-ROHS to add ROHS processing

-ALT to change to alternate pinout for Ethernet and Serial Ports

-FAN to add a Zero Slot 5.2 CFM fan to the assembly

-FANR to add a rear mount 8CFM fan to the assembly – will take two slots to install

-JTAG to add the JTAG header to connect to the PMC position.

Win32 driver software available with board purchase. Additional OS porting available upon request.

All information provided is Copyright Dynamic Engineering

