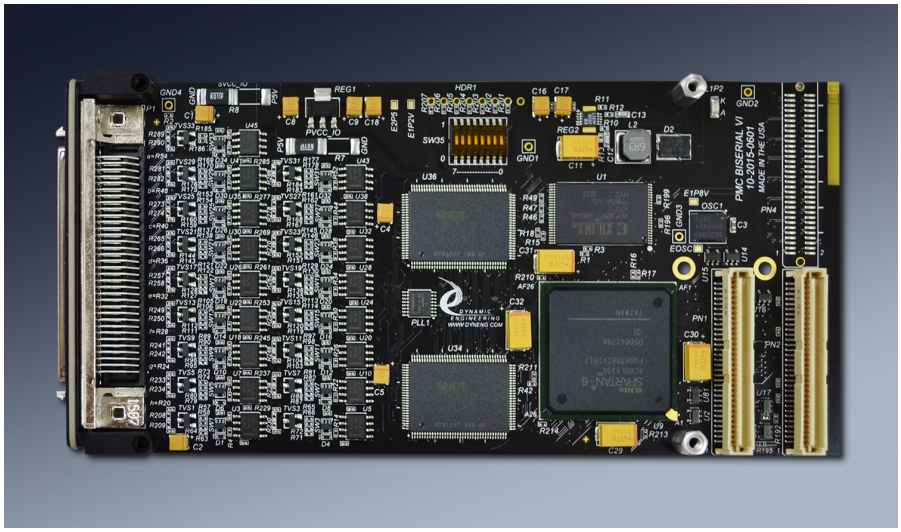
DYNAMIC ENGINEERING
150 DuBois St., Suite C Santa Cruz, CA 95060
(831) 457-889
https://www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PMC-BiSerial-VI-UART Hardware Manual

8-Channel UART Interface

Manual Revision 01p12



Corresponding Hardware: 10-2015-0601/2/3/4/5

PMC-Biserial-VI-UART

8-Channel UART Interface

Dynamic Engineering
150 DuBois St., Suite C
Santa Cruz, CA 95060
(831) 457-8891

Manual Revised 10/25/21

Table of Contents

List of Figures

Product Description

PMC-BISERIAL-VI-UART is part of the Dynamic Engineering family of modular I/O. PMC-BISERIAL-VI-UART is a PMC with options for bezel and rear IO, up to 2 MHz signaling, multiple modes of operation, 1K byte of storage per Tx or Rx node. 8 Full Duplex Ports. Added features with the BiSerial-VI version include DMA and additional transmit modes to support forced errors and unusual protocols.

PMC-BISERIAL-VI-UART uses a 10 mm inter-board spacing for the front panel, standoffs, and PMC connectors. The 10 mm height is the "standard" height and will work in most systems with most carriers. If your carrier has non-standard connectors (height) to mate with PMC-BISERIAL-VI-UART, please let us know. We may be able to do a special build with a different height connector to compensate.

Feature Table:
1. 255x32 FIFO's for Rx and Tx data storage per channel
2. 255x16 FIFO's for Packet definitions Rx and Tx
3. 4 operating modes, 32 bit packed, 8 bit unpacked, packetized, and Test(Tx only)
4. 8 position Switch
5. Windows driver and reference software. Linux and VxWorks by request.
6. Industrial temperature components [-40 ⇔ +85C]
7. Standard baud rates and non-standard baud rates programmable based on a 32 MHz reference. 2M Tx and Rx max rate. PLL with user defined frequency up to 100 MHz also available. With PLL max rate is 6.25 Mbits.

The following diagram shows the PMC-BISERIAL-VI-UART configuration:

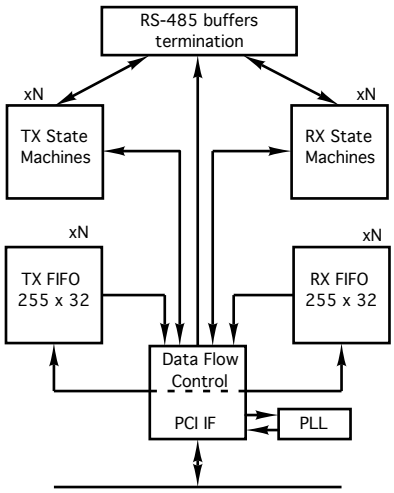


FIGURE 1 PMC-BISERIAL-VI-UART BLOCK DIAGRAM

Please note: The Packet FIFO's provide an additional 256 x 16 per channel per direction [2xN] to store packet sizes for transmission or definitions from reception.

If you can use the BiSerial hardware but need an alternate protocol please contact Dynamic Engineering. We will redesign the state machines and create a custom interface protocol. 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 UART protocol implemented provides RS422 data inputs and outputs. The transceivers have supporting programmable terminations to allow for in cable and on-board termination situations. The receivers are open cable safe – marking state is detected when undriven.

Baud rates are programmed for each transmitter and receiver separately. The design uses a distributed enable concept to allow all channels to be referenced to the master 32 MHz clock and be programmed to unique counts.

The transmitter has a pulse generator that puts out 1 clock period per programmed count. The state-machine is referenced to the master clock and sequences when the pulsed enable is present. This allows all transmit UART's to use the same reference clock and results in much better timing within the FPGA with limited clock resources.

Rx data is asynchronous and potentially noisy. Rx data is synchronized and filtered

with the master reference clock before being presented to the UART decoder. Within the UART, data is sampled and checked for being in the marking state before looking for the first start bit.

Transitions are detected and used to update the reference count. When transitions are not detected; the reference count and programmed baud rate [expected count] are used to determine when to capture bits. The receiver uses the programmed count to determine when to sample the data received. The transition detections are filtered to only be applicable within 1/8th of the expected transition. The receiver can handle quite a lot of jitter in this manner. Depending on the data [number of transitions] up to +/- 1/8th bit period per bit cell (with a transition).

Each PMC-BISERIAL-VI-UART channel is supported by two 255 by 32-bit FIFOs. The TX FIFO supports long-word writes, and the RX FIFO supports long-word reads. A FIFO test bit in each channel control register enables the data to be routed from the TX to the RX FIFO for loop-back testing of the FIFO's. The FIFO's are used for packed, unpacked, packetized, Alternate Packetized and Test modes of operation.

In packed mode 32 bit data is assumed, 4 bytes per LW to transmit or receive. Bytes are sent/received 0,1, 2, 3 with byte 0 being the data bits 7-0 on the PCI bus. 1/4 of the reads and/or writes are needed in this mode compared to unpacked.

Unpacked mode operates more like a traditional byte wide UART. Only Byte 0 is used for each LW read / written to the FIFO's. Effectively a 255 byte FIFO for TX and RX in this mode compared to 1020 bytes possible in packed mode.

With both packed and unpacked modes, if the UART is enabled the data is sent and received on demand. As soon as there is data in the output FIFO it is transmitted. If the FIFO becomes empty the transmitter waits in the marking state until more data is ready to send. Similarly the receiver writes data as it comes in without any concept of a frame or packet.

In packetized mode the transmitter waits for the packet descriptor FIFO [255x16] to have at least one descriptor loaded. As data for the packet becomes available it is transmitted. Any number of bytes can be sent in this mode. Data is packed with the possible exception of the last LW in a packet. 1, 2, 3, or 4 bytes can be sent from the last LW read for a particular packet. The next packet will start on the next LW boundary. Packets can be stacked in memory and unloaded as described [just multiple times]. In addition the inter-packet timer can be utilized to add delay between consecutive packets.

Alternate Packetized mode is similar to Packetized with the ability to send packed data

and a final LW with a smaller number of bytes for any length. The difference is the data is packed 3 bytes per LW with the upper byte used for control. The last LW in a packet has the MSbit set to indicate it is the last in the packet, the next two bits provide the count – number of bytes to send. The transmitter can be programmed to go to tristate after the packet completes in this mode as well. The advantange is a single DMA transfer can load the packet control. The disadvantage is the loss of ¼ of the data transfer available.

Depending on your system requirements, and the number of bytes to send per message the best choice of the 4 standard modes can be made. Test mode is used to create errors and for system test and development.

Test Mode allows the user complete control over the data sent on a word by word basis. The lower 16 bits of the FIFO determine what is sent with the SW providing all of the formatting – including start bits and so forth. A separate field provides the number of bits to send out of the 16.

The receiver uses a programmable timeout to determine the end of the packet. It is suggested to use the equivalent time to 2 characters modified as needed for the inter-character gap you expect in your system. Data being received is stored locally and built into a LW to write to the Rx FIFO. When an inter-character gap exceeds the programmed delay the accumulation stops and the data captured is written to the FIFO. In addition, data is written to the FIFO when a complete LW is available. When the end of packet is detected the packet length and packet status are written to the Rx Packet FIFO. The accumulated status is written along with the length to allow multiple packets to be stored and accurate status per packet to be available.

Interrupts can be programmed from a variety of sources. The FIFO's have counts and comparators to allow almost full and almost empty situations to cause interrupts. In addition an interrupt is available for packet transmitted, packet received, and various error conditions. All interrupts are individually maskable, and a channel master interrupt enable is provided to disable all interrupts on a channel simultaneously. The current real-time status is also available from the FIFO's making it possible to operate in a polled mode.

When using internal loop-back the Almost Full and Almost Empty counts should be set to x10 or more from the end of the FIFO.

More on byte alignment: Transmit bytes are read from byte positions 0->3 byte lane wise [7-0] first, [15-8] second, [23-16] third and [31-24] last and the bytes are transmitted in this order. For message byte-counts not divisible by four, the last long-word is read as described. Any unused bytes are considered padding with the next

message starting with the next FIFO long-word. For example, with 7 bytes to send, a word of 4 bytes will be read, then the lower 3 bytes will be read and sent and the 8th byte will be dropped.

In the receive direction the action is similar. Bytes are written as long-words to the RX FIFO. The first byte received is loaded into long-word byte 0 [7-0], then byte 1 [15-8], byte 2 [23-16] and byte 3 [31-24]. Whenever a message does not have a complete long-word to load and the end-of-packet character is received, zero-padding of the unused upper-bytes will occur before the long-word is written to the FIFO.

Dynamic Engineering offers drivers and reference software for Windows®, Linux, and VxWorks. Drivers and reference SW are available AS-IS to clients of the PMC-BISERIAL-VI-UART. Support contracts are encouraged to help with integration and enhancements. https://www.dyneng.com/TechnicalSupportFromDE.pdf

Theory of Operation

PMC-BISERIAL-VI-UART provides UARTs for transferring data from one point to another using the standard UART transfer protocol.

While UART's are mature devices, enhancements will necessitate updates over time. PMC-BISERIAL-VI-UART features the ability to reprogram the FPGA storage FLASH to allow updates via software. A programming adapter is required to use this feature on this HW set.

A logic block within the Xilinx controls the PCI interface to the host CPU. PMC-BISERIAL-VI-UART design requires one wait state for read or writes cycles to any address. The wait states refer to the number of clocks after the PCI-core decode before the "terminate with data" state is reached. Two additional clock periods account for the 1 clock delay to decode the signals from the PCI bus and to convert the terminate-with-data state into the TRDY signal.

There are multiple UART's each with separate Receiver and Transmitter. Each pair is organized into a Channel within the FPGA. Frequency of operation [Baud rate], mode of operation, Parity, Stop bits, interrupt conditions are all programmable on a channel basis. In addition the mode of operation can be selected for each receiver and transmitter.

Each channel has separate state-machines to control the Transmit and Receive operation. The Tx state-machine uses the programmed values to regulate the transfer of data from the transmit storage FIFO and transmit packet FIFO to the Tx line. The Rx state-machine uses the programmed values to regulate the transfer of data from the line to the receive storage FIFO and to store descriptors into the Rx packet FIFO.

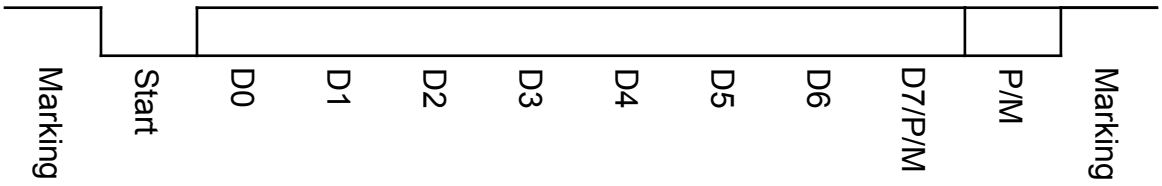


FIGURE 2 UART TRANSFER ENCODING

The Transmit state-machine will transmit a high level followed by the first falling edge of the transmission. The falling edge is the leading edge of the start bit. The start bit is 1 period wide and followed by the first data bit [LSB] of the byte being transmitted. D1-D6 follow. If the UART is programmed for 8 bit data the next period is D7. If programmed for 7 bit data the next position can be Parity if that is enabled or the marking state. The shortest transfer of a byte is 7 bit data, no parity and 1 stop bit for a total of 1[start]+7[data]+1[stop] = 9 bits. If 8 bit data is selected and parity is enabled the length becomes 1+8+1+1 = 11 bits. If 2 stop bits are selected an extra clock period is inserted between byte transfers.

The receiver does not have a clock to work with and uses over-sampling to detect the transitions and the programmed expected transfer rate to count into the bit periods to determine the bit value. The receiver also checks the expected termination values are present – for example a framing error is detected if the received signal is low when marking is expected.

Parity can be programmed to be odd, even or level. When odd the parity bit is set/cleared to make the number of 1's odd. For example if the data is "AA" an even number of bits are set in the data so the parity would be set "0 01010101 1 1" would be the string with start, data, parity and stop shown. Please note the lsb first nature of the data. The spaces are added for clarity. For even parity the reverse is true, with parity set/cleared to make the total of the data and parity fields an even count.

In addition to the framing and parity errors, FIFO over-run is flagged. When the Rx FIFO is full and a write is attempted the error is captured. A full FIFO will not accept the new write so that data is lost.

Break characters are detected by the RX state-machine and prioritized in terms of status. Status is determined Break, Frame, Parity with only one type of error or condition reported per incident. Interrupts can be generated from the occurrence.

When Break or Frame is detected the receiver resynchronizes before looking for new characters. With parity errors the error is flagged and processing continues without resynchronization.

Over reading the Rx FIFO is not an error condition. The FIFO will continue to provide the last read data multiple times. The FIFO count should be read prior to doing read multiple commands to prevent under-run.

On the Tx side an empty FIFO causes the transmitter to go to the marking state once the last word read has been transmitted. When more data is available that data will be transmitted. No under-run error is generated for this situation.

Programming

Programming PMC-BISERIAL-VI-UART board 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 board is installed. The VendorId = 0xDCBA. The CardId = 0x0061.

In order to transfer data to another UART, several steps must be performed. First a physical connection must be established with the appropriate interface cable. Then the Channel of interest must be programmed with the appropriate UART parameters for transmit and or receive operations. Each channel has a separate register set for control bits, baud rate and other parameters. Once programmed you can load data into the Tx buffer for transmission or read from the Rx when data becomes available.

Be sure to select the correct mode of operation, and note the Rx and Tx do not need to be the same.

The hardware supports several modes of operation. Choose the right mode based on your environment. For example if you are operating with a console program and need to remain compatible with other standard UART's the 8bit [unpacked] mode will be the right choice. 3/4 of the FIFO is lost and still provides 512 bytes for both Rx and Tx.

If you need more performance and can do some adaptation the Packet, Alt Packet, or Packed modes are very useful. The Packed mode is easy to use but requires byte counts that are multiples of 4. Packet mode is the best of both with almost complete FIFO utilization and the ability to send non-LW boundary message lengths. Packet mode Packets can be stored ahead and transmitted based on the packet descriptor being written or pre-loaded and sent out as the HW becomes ready. Packet received and packet transmitted interrupts are available to help optimize operation.

The reference software has examples of using all modes of operation.

The baud rate is programmable, and should be set to a value close to the value expected. The jitter tolerance will allow slightly off frequencies to work, but will effectively have no jitter tolerance when operating in this manner. The baud rate is programmable directly based on the reference frequency allowing 1 part in 32 $x10^6$. With the RS485 IO the maximum rate tested is 4M+. Using the PLL reference higher rates are programmable.

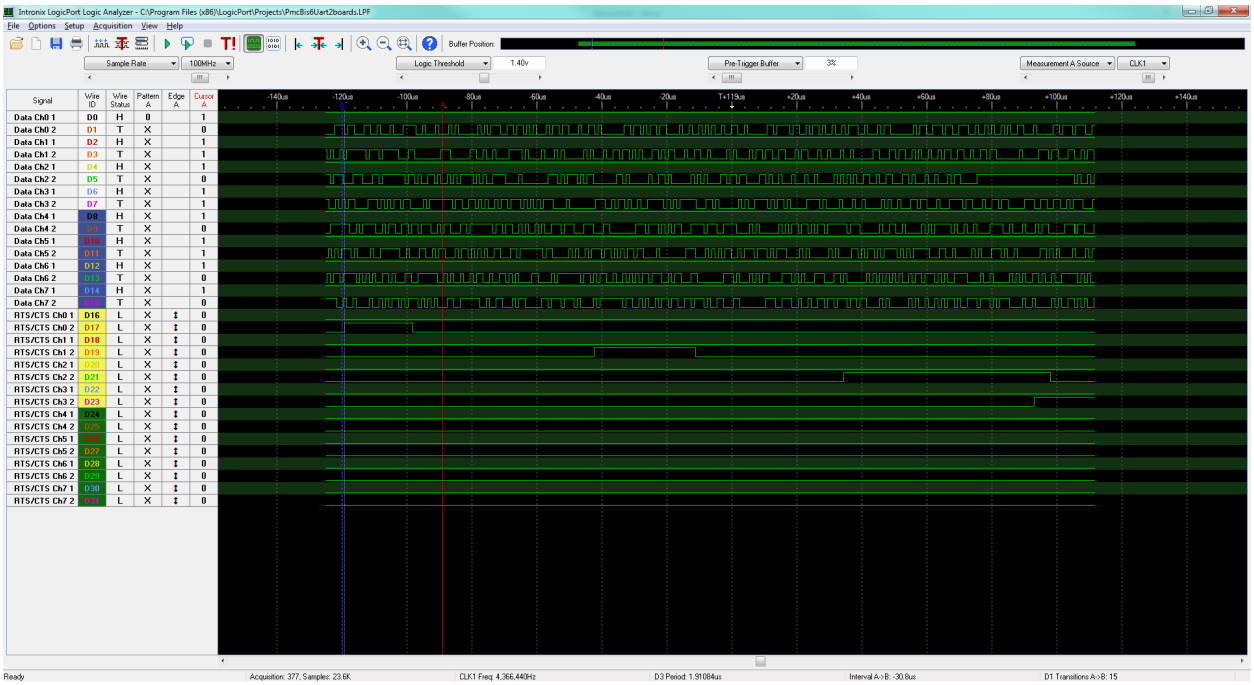


FIGURE 3 UART TRANSFER SCREEN SHOT

The diagram above depicts all 8 UART channels running with data at 2 MBits/sec and using RTS/CTS flow control. This is a board-to-board test where the 8 transmiters are on one card and the 8 receivers on a second card. Data is loaded and unloaded using DMA. The test SW is available for the Windows reference package. The SW is reversible meaning the current receiving board can become the transmitter and vice-versa.

Firmware Updates

Revision 2.1: First release on “VI”. DMA added
Revision 2.2 Test and Alternate Packetized modes added
Revision 2.3 Add back to back test mode transmission
Revision 2.4 Add clock option to use PLL or 32 MHz for each channels reference
Revision 2.5 Add RTS/CTS flow control to design.
Revision 2.6 Add clearing of accumulated status errors between packets for Alt Packet Mode [already in place for standard Packet mode]
Revision 2.7 Add Big Endian support
Revision 2.8 misc clean-up, mark and space parity clean-up, allow for PLL rates up to 64 MHz.
Revision 2.9 Update DMA for better sharing between ports for access to the PCI bus.
Revision 2.10 Update DMA for better performance, recompile for 100 MHz PLL operation.

Base Address Map

Register Name		Offset	Description
#define	BASE_REG	0x0000	//0 JTAG programming control
#define	BASE_GP	0x0004	//1 Switch, revision
#define	BASE_INT	0x0008	//2 Interrupt Status
#define	BASE_PllData	0x0010	//4 PLL Data
#define	BASE_PllStatus	0x0014	//5 PLL Status

FIGURE 4 PMC-BISERIAL-VI-UART BASE ADDRESS MAP

UART Port Address Map

Register Name		Offset	Description
#define	CHAN_CNTL	0x0000	//0 UART Port Control Bits R/W
#define	CHAN_CNTLB	0x0004	//1 Expanded UART control bits & Tx Packet delay
#define	CHAN_STAT	0x0008	//2 UART Port Status Bits Read /write to clear
#define	CHAN_TX_FIFO_CNT	0x000C	//3 UART Port TX Packet and Data FIFO's
#define	CHAN_RX_FIFO_CNT	0x0010	//4 UART Port RX Packet and Data FIFO's
#define	CHAN_TX_DMA_PTR	0x0014	//5 UART Port Write TX DMA Pointer
#define	CHAN_RX_DMA_PTR	0x0018	//6 UART Port Write RX DMA Pointer
#define	CHAN_RX_UART_FIFO	0x001C	//7 UART Port Read from RX UART FIFO
#define	CHAN_TX_UART_FIFO	0x001C	//7 UART Port Write to TX UART FIFO
#define	CHAN_TXFIFO_LVL	0x0020	//8 UART Port Tx Almost Empty 15-0 =
#define	CHAN_RXFIFO_LVL	0x0024	//9 UART Port Rx Almost Full 15-0
#define	CHAN_FRAME_TIME	0x0028	//10 UART Port End of Frame Time 23-0
#define	CHAN_BAUD_RATE	0x002C	//11 UART Port Frequency 15-0 = Tx, 31-16 = Rx
#define	CHAN_TX_PKT_FIFO	0x0030	//12 UART Port Write to TX Packet FIFO
#define	CHAN_RX_PKT_FIFO	0x0030	//12 UART Port Read from RX Packet FIFO
#define	CHAN_TX_MODULUS	0x0034	//13 UART R/W Modulus definition Port
#define	CHAN_TX_CURRENT	0x0038	//14 UART RO Timer Curent Count
			//15-19 Spare decodes per port

FIGURE 5 PMC-BISERIAL-VI-UART UART CHANNEL ADDRESS MAP

There are 8 UART Ports. Each port/channel has a separate set of control registers as shown in Figure 4. The offset for each of the channels is shown in Figure 5.

Channel Offsets

```
#define    CH_0    0x0050 //20 address pointer for channel 0
#define    CH_1    0x00A0 //40 address pointer for channel 1
#define    CH_2    0x00F0 //60 address pointer for channel 2
#define    CH_3    0x0140 //80 address pointer for channel 3
#define    CH_4    0x0190 //100 address pointer for channel 4
#define    CH_5    0x01E0 //120 address pointer for channel 5
#define    CH_6    0x0230 //140 address pointer for channel 6
#define    CH_7    0x0280 //160 address pointer for channel 7
```

FIGURE 6 PMC-BISERIAL-VI-UART CHANNEL OFFSETS

The base address for PMC-BISERIAL-VI-UART is set by the system. For Base features the base address is added to the base feature offset. For Channel features the base address is added to the Channel Offset and to the Channel Feature. Address = Base + Channel Offset+Channel Feature. All addresses are on LW boundaries and all accesses affect the entire LW. Writing a byte still affects the other three bytes.

Register Definitions

BASE_REG

Base Control Register (read/write)

Base Control	
Data Bit	**Description**
31	BigEndianDMA
30-5	Spare
4	PllAltAddress
3	CheckPll
2	ReadPll
1	ClrPll
0	PllProgEn

FIGURE 7 PMC-BISERIAL-VI-UART BASE CONTROL REGISTER

All bits are active high and are reset on system power-up or reset.

PllProgEn: When this bit is set to a one, the state-machine used to program the PLL is enabled to operate.

ClrPll: when set the PLL and associated memories are cleared. Must be returned to cleared for normal operation.

ReadPll: when set the PLL is read and the data returned. Must be returned to cleared for normal operation.

CheckPll: when set the PLL address is checked the data returned. Must be returned to cleared for normal operation.

PllAltAddress: when set the alternate PLL address is used for programming and reading operations.

The PLL is programmed with the output file generated by the Cypress PLL programming tool. [CY3672 R3.01 Programming Kit or CyberClocks R3.20.00 Cypress may update the revision from time to time.] The .JED file is used by the Dynamic Driver to program the PLL. Programming the PLL is fairly involved and beyond the scope of this manual. For clients writing their own drivers it is suggested to get the Engineering Kit for this board including software, and to use the translation and programming files ported to your environment. This procedure will save you a lot of time. For those who want to do it themselves the Cypress PLL in use is the 22393. The output file from the Cypress tool can be passed directly to the Dynamic Driver [Linux or Windows] and used

to program the PLL without user intervention.

The reference frequency for the PLL is 16 MHz.

BigEndianDma : ‘0’ disables this option. ‘1’ enables this option. When operating with a Big Endian platform and using PCI accesses DMA can have challenges. The register accesses directly over the PCI bus are usually taken care of automatically with byte swapping within the CPU or PCI interface on the CPU. DMA data is written to or read from the local memory and is not swapped. The direct read/write from memory ends up with scrambled data [relative to UART little endian definitions]. Setting this bit will byte reverse the data for the DMA path into the Tx and out of the Rx FIFO’s only. Register accesses are not affected.

31-24, 23-16, 15-8, 7-0 ⇔ 7-0, 15-8, 23-16, 31-24 byte swapping pattern implemented.

BASE_GP

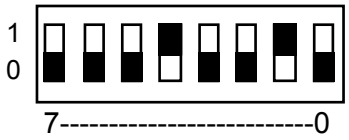
Base General Purpose Register (read)

Base General Purpose Register

```
#define   BASE_STAT_SW_MASK   0x000000FF // 7-0 are switch bit when installed
#define   BASE_STAT_REV_MAJ   0x0000FF00 // Design major revision
#define   BASE_STAT_REV_MIN   0x00FF0000 // Design minor revision
#define   BASE_STAT_XIL_TYP   0xFF000000 // Design number -- static per
                                         implementation
```

FIGURE 8 PMC-BISERIAL-VI-UART BASE GP REGISTER

Switch 7-0: The user switch is read through this port. The bits are read as the lowest byte. Access the read-only port as a long word and mask off the undefined bits. The dip-switch positions are defined in the silkscreen. For example the switch figure below indicates a 0x12. The switch is an optional item. Bits have no meaning if not installed.



The Major Revision is used to track FLASH releases to the client. The revision will be updated when official releases to clients occur to allow the client to tell if a board has been updated. Currently 2.

The Minor Revision is used to track FLASH updates during development and for unofficial releases to clients. This revision may roll over depending on the number of iterations needed. Currently 10.

The Xilinx Type is the design number for a particular version of the board. A new type will be assigned for each new design implemented. In addition the CardID will also be updated. UART is type x15. -LM model is type x1A.

BASE_INT

Base Interrupt Status Register (read/write)

Base Interrupt Status

```
#define   BASE_INT_CH_0    0x00000001 // Set if interrupt active channel 0
#define   BASE_INT_CH_1    0x00000002 // Set if interrupt active channel 1
#define   BASE_INT_CH_2    0x00000004 // Set if interrupt active channel 2
#define   BASE_INT_CH_3    0x00000008 // Set if interrupt active channel 3
#define   BASE_INT_CH_4    0x00000010 // Set if interrupt active channel 4
#define   BASE_INT_CH_5    0x00000020 // Set if interrupt active channel 5
#define   BASE_INT_CH_6    0x00000040 // Set if interrupt active channel 6
#define   BASE_INT_CH_7    0x00000080 // Set if interrupt active channel 7
```

FIGURE 9 PMC-BISERIAL-VI-UART BASE INTERRUPT STATUS

Each UART channel has a multitude of interrupt options. Those possible interrupts are combined into one for the port and used to generate a board level interrupt, and to provide the status in the register as shown. Clear the interrupt by servicing the source channel. Multiple interrupts can be detected in one read.

BASE_PllData

[0x0010] PLL Data FIFO (read/write)

PLL Data FIFO

Data Bit	Description
31-0	Data to PLL or Data From PLL

FIGURE 10 PMC-BISERIAL-VI-UART BASE PLL DATA FIFO

A hardware I2C interface for programming the PLL is provided. Dynamic Engineering driver support packages include utilities to take the .jed file from the Cypress CyberClocks program, parse and load into the FIFO with the proper sequence of controls via Base Control Register. Please see the reference code for the sequence. Linux, VxWorks, Win7 packages. The PLL clock reference is 16 MHz.

The data to program the PLL is written to this address. The hardware has a state-machine to read the data from the FIFO and load into the PLL. Similarly the state-machine can read the data from the PLL and write it to the read side FIFO.

BASE_PllStatus

[0x0014] PLL Status (read/write)

Time Control Register

Data Bit	Description
31-10	Spare
10	PLL Error Latched
9	PLL Done Latched
8	PLL Ready
7	Spare
6	PLL FIFO RX Data Valid
5	PLL FIFO RX FULL
4	PLL FIFO RX EMPTY
3	Spare
2	PLL FIFO TX Data Valid
1	PLL FIFO TX FULL
0	PLL FIFO TX MT

FIGURE 11 PMC-BISERIAL-VI-UART BASE PLL STATUS

The PLL Status bits are used to as feed-back to control the transfer of data to and from the PLL FIFO. TX refers to programming the PLL and RX refers to reading back from the PLL.

The Latched Bits {10,9} are held until cleared by writing back with the bit position(s) set. Usually these bits are cleared before starting an operation.

PLL FIFO TX MT is set when the programming FIFO for the PLL is empty.

PLL FIFO TX FULL is set when the programming FIFO for the PLL is full.

PLL FIFO TX Data Valid is set when data is valid in the pipeline between the FIFO and the State –Machine. The bit is cleared each time the data is read. During operation this bit will toggle to provide some indication that the transfer is occurring.

PLL FIFO RX MT is set when the read-back FIFO for the PLL is empty.

PLL FIFO RX FULL is set when the read-back FIFO for the PLL is full.

PLL FIFO RX Data Valid is set when Data is valid in the output port for the PLL read

path. Data is pre-read from the FIFO and held in the FIFO holding register. The FIFO can be Empty and still have 1 word left in the holding register if Valid is still set.

PLL Done Latch is set when the transfer is completed. This bit can be polled to know when the PLL has been programmed or when the PLL has been read. Please note: The PLL settling time is in addition to the transfer time. Several mS should be delayed after programming the PLL to make sure the specified frequencies are within range. 10 mS is recommended.

PLL Error Latched is set when an error is detected in the I2C transfer. The main purpose for this bit is in discovery for the address of the PLL. The Address can be x6A or x69. Once the correct address is known this bit should be checked but not set. Sticky bit, write with bit position set to clear.

PLL mapping to ports is defined in the channel section under Control Register B.

UART_CHAN_CONT

UART CHANNEL CONTROL

```
#define   ChRstA          0x0001 // set to reset channel Tx side
#define   LoopBackA       0x0002// set to loop-back FIFO data
#define   TxEnable        0x0004// set to enable Tx operation
#define   RxEnable        0x0008// set to enable Rx operation

#define   RxErrIen        0x0010// set to enable interrupt on Error
#define   TxFfAmtIen      0x0020// set to enable Transmit almost empty interrupt
#define   RxFfAfIen       0x0040// set to enable Receiver almost full interrupt
#define   DmaRdIEn        0x0080// set to enable DMA Interrupt Read

#define   DmaWrIEn        0x0100// Set to enable DMA Interrupt Write
#define   ForceInt        0x0200// set to force an interrupt from this channel
#define   RxOverFlowIen   0x0400// set to enable Rx Data FIFO overflow interrupt
#define   RxPckLvlIen     0x0800// set to enable Packet FIFO not empty interrupt

#define   ChRstB          0x1000 // set to reset channel Rx side
#define   TxBreak         0x2000 // set to cause Tx break – Space on TXD
#define   spare           0x4000 // set to
#define   MastIntEn       0x8000// set to allow any interrupts from this channel

#define   TxParityOn      0x00010000// set to use parity on Tx
#define   TxParityOdd     0x00020000// set to generate odd parity when Parity is On
#define   TxStopBits      0x00040000// set to transmit 2 or more stop bits
#define   TxLength        0x00080000// set to transmit 8 bits cleared = 7 bit data

#define   RxParityOn      0x00100000// set to use parity on Rx
#define   RxParityOdd     0x00200000// set to expect odd parity when Parity is On
#define   RxStopBits      0x00400000// set to expect 2 or more stop bits for framing
#define   RxLength        0x00800000// set to expect 8 bits, cleared = 7 bit data

#define   TxMode(0)       0x01000000// Encoded transmit type
#define   TxMode(1)       0x02000000//
#define   TxMode(2)       0x04000000//
#define   TxParityLvl     0x08000000// Set to use level parity

#define   RxMode(0)       0x10000000// Encoded receive type
#define   RxMode(1)       0x20000000//
#define   RxMode(2)       0x40000000//
#define   RxParityLvl     0x80000000// Set to use level parity
```

FIGURE 12 PMC-BISERIAL-VI-UART UART CHAN CONTROL

ChRstA, ChRstB : When bit(s) is/are set to one, most functions within the channel are reset. Holding registers are not reset. Memories, state-machines etc. are reset. Clear for normal operation. The “A/B” indicates this signal is Or’d with the RST signal to make the channel reset based on local or global resets. A for Tx Functions, B for Rx. Software timed – leave asserted for at least one UART reference clock period.

Loop-BackA: When this bit is set to a one, any data written to the transmit FIFO will be immediately transferred to the receive FIFO. This allows for fully testing the data FIFO’s without connecting externally. When this bit is zero, normal operation is enabled. The “A” indicates HW protection to require both Tx and Rx enables to be disabled to do loop-back testing.

TxEnable when set allows the Transmit state-machine to operate. Depending on the mode other conditions will also need to be met before transmission will begin. TxEnable can also be set and cleared via HW. In Alternate Packet mode if the TxTimerMode is set to affect TxEnable, the enable will be cleared at the end packet and enabled when the timer expires. Please see those sections for more detail.

RxEnable when set allows the Receive state-machine to operate. This bit should be set after the other pertinent parameters are programmed.

pertinent parameters: Baud Rate, FIFO levels, character level controls [parity, number of bits etc.] When switching modes the enable should be disabled and then re-enabled to allow the state-machine to return to idle before resuming processing. Allow several clock periods.

RxErrIen is set to allow the error conditions of Parity, Framing, Packet FIFO overrun to cause an interrupt to the host. When cleared the status is available but the interrupt is not.

TxFfAmtIen is set to allow the Transmit FIFO Almost Empty condition to cause an interrupt. When cleared the status is available but the interrupt is not. An interrupt will be generated when the transmit FIFO level becomes equal or less than the value specified in the TX_AMT register, provided the channel master interrupt enable is asserted.

RxFfAftIen is set to allow the Receive FIFO Almost Full condition to cause an interrupt. When cleared the status is available but the interrupt is not. An interrupt will be generated when the receive FIFO level becomes equal or greater to the value specified in the RX_AFL register, provided the channel master interrupt enable is asserted.

DmaRdIEn/ DmaWrIEn DMA Interrupt Enable: These two bits, when set to one, enable the interrupts for DMA write and read completion for the referenced channel. These two interrupts cannot be disabled by the master interrupt enable.

ForceInt is set to cause an interrupt to occur. Used for SW development and test purposes.

RxOverFlowIen is set to allow the Rx FIFO overflow condition to cause an interrupt. When cleared the status is available but the interrupt is not.

RxPckLvlien is set to allow the Rx Packet Received interrupt. If enabled and a Packet Descriptor is in the Packet FIFO the interrupt is set. This is a level based interrupt. Clear by reading the descriptors in the packet FIFO.

TxBreak when set forces the TXD line low which creates a “Break” condition on the transmit line – forced into the spacing state. Software timed.

MasterIntEn when set allows any of the programmable interrupt conditions to be passed to the host. When cleared no interrupts are generated by this channel.

TxParityOn when set causes the transmitted data to have parity inserted. When cleared parity is not added.

TxParityOdd when set causes odd parity when Parity is enabled and Level is not enabled. When cleared even parity is inserted if enabled.

TxStopBits when set causes the HW to add a wait state – an extra marking state between characters sent. The minimum is 1 stop bit [sent when TxStopBits is not set]. If another character is not ready when the current one is completed additional marking bits will also be inserted.

TxLength when set causes 8 bit characters [considered standard] and when cleared 7 bits per byte are transmitted. The Msb is trimmed when in the 7 bit mode.

RxParityOn when set causes the receiver to expect data with parity inserted. Parity is checked in this mode and parity errors reported. When cleared, parity is not expected and potential framing errors captured if parity is received.

RxParityOdd when set causes odd parity to be checked when Parity is enabled and not in level mode. When cleared even parity is checked if enabled.

RxStopBits when set causes the HW to expect a wait state – an extra marking state between characters sent. The minimum is 1 stop bit [sent when RxStopBits is not set]. If a start bit is received when a second stop bit is expected a framing error will result.

RxLength when set causes 8 bit characters to be expected in the data stream[considered standard] and when cleared 7 bits per byte are received. The data is LSB aligned when received in 7 bit mode. Framing errors can result if 8 bit data is received when 7 is expected and vice-versa.

TxMode 2:0
TxOneByte (TxMode "001") when selected causes data to be transmitted based on using only the LS byte from the FIFO [unpacked mode – standard low speed UART operation and use with console operation].

TxPacked (TxMode "010") When selected all 4 bytes are transmitted per LW [packed mode – higher bandwidth but requires LW based data transfers – divisible by 4 data frames]

TxPacketized (TxMode "011") when selected, enables operation in Packet Mode [Packetized]. Programming note: Packetized mode is a hybrid of the packed and unpacked modes allowing for higher bandwidth operation via lower overhead for medium to larger messages. Please see the packet FIFO description for more details of using this mode.

TxAltPacketized (TxMode "100") when selected, enables operation in the Alternate Packet Mode. The data and control for the packet are both in the data stream in this mode. In this mode the packet control information is embedded in the transmit data. The advantage is the control can be DMA transferred along with the data. The disadvantage is losing 1 byte per LW transferred to the control information.

Bytes are transferred in the same order as the other modes. 0, 1, 2.
Upper Byte Definition:
31 = last data set in packet. Set for last data set within packet, cleared otherwise.
30-29 = byte count in last data set. 01, 10, 11 are valid.
28-25 = spare
24 = Transceiver Tristate, After Packet complete [all bits sent] disable Transceiver Enable/Tristate Transmitter [either, neither, both]. Either SW enable or Start of new Packet [Timer expired] will re-enable.

TxTest (TxMode "101") when selected, enables operation in the Test Mode. The raw data and control are included in the same LW. The lower 16 bits contain the data. The upper nibble is the length. The bits are sent as programmed without adding formatting

other than the marking state between characters.

When the TxTest mode is selected the transmission is governed by the FIFO Empty status. As characters are available to transmit they are read and sent. The character is sent LSB first. The bits are parallel loaded into a shift register and transmitted. No HW modification in the sense of adding Start, parity and so forth.

“1111 1110 0100 0010” for example would transmit x21 with the start bit prepended and the marking state for the remaining bits. If parity is needed it would be added after the “2” and before the ‘1’s used for padding. The character count could be set to anything larger than the total bit count needed. The count starts with 0 to allow F to be all 16 locations. In the example the count could be 8 or more. If the exact length is used the HW will not insert added bits between characters. If the count is larger than the size of the character, additional stop bits will be added [assuming a new character is available].

If the FIFO is empty when the terminal count is reached for the current character, the transmission is terminated after several ‘1’s are clocked out. If the FIFO is not empty when the bit count reaches 3 the FIFO is read and the next character and length stored for use by the shift register and state machine. This allows rapid character transmission when multiple characters are stored. The cost is a minimum count since the pre-read of the next character needs to happen after the current character has been loaded and started to prevent overwriting unsent data. The minimum count is 5 which corresponds to 6 bits sent including a start bit.

The purpose of this mode is to allow SW to create any sort of error desired – missing start bit, missing parity, wrong type of parity, incorrect data bit in any location, not enough stop bits etc.

RxMode 2:0

RxOneByte (RxMode “001”) when selected causes the received data to be loaded one byte per LW in the Rx Data FIFO.

RxPacked (RxMode “010”) When selected all 4 bytes are loaded per LW stored [packed mode –requires LW based data transfers – divisible by 4 data frames]

RxPacketized (RxMode “011”) when selected causes the Rx state-machine to group received data into packets and to load packet descriptors into the Rx Packet FIFO. Packet lengths are automatically determined based on the programmed FrameTime. Be sure to program this time-out if in Packet Mode for Rx.

RxAltPacketized (RxMode “100”) when selected, enables operation in the Alternate Packet Mode. The data and control for the packet are both in the data stream in this mode. In this mode the packet control information is embedded in the received data. The advantage is the control can be DMA transferred along with the data. The disadvantage is losing 1 byte per LW transferred to the control information.

Bytes are received in the same order as the other modes. 0, 1, 2.
Upper Byte Definition:
31 = last data set in packet. Set for last data set within packet, cleared otherwise.
30-29 = byte count in last data set. 00, 01, 10 are valid.
28-27 = spare
26 = Data FIFO overflow Error occurred in this packet
25 = Framing Error occurred in this packet
24 = Parity Error occurred in this packet

Notes: 1) Byte Count, when 00 means no bytes stored, message was divisible by 3, written before end of packet detected leaving a remainder of 0. Status is set in the last word.
2) Error bits are accumulated through the packet, and when the packet is complete; stored into the status word. “000” for these bits would be no error. These are the latched status bits from the status register. In this mode the status is cleared before each new packet is received for independent reported on each packet. Similar to Packet mode.
3) When last data set bit is not set, all three bytes have data and the count is not loaded.

TxParityLvl when set and parity enabled causes the inserted parity to be a level with the ODD/EVEN control determining the level. ODD forces a ‘1’ and Even forces a ‘0’.

RxParityLvl when set and parity enabled checks the inserted parity to be a level with the ODD/EVEN control determining the level. ODD expects a ‘1’ and Even expects a ‘0’.

UART_CHAN_CONTB

UART CHANNEL CONTROL

```
#define   BreakRiseIen        0x00000001 //0 set to enable capture of Break Detection
#define   BreakFallIen        0x00000002 //1 set to enable capture of Break removal
#define   BreakIen            0x00000004 //2 set to enable Break Interrupt
#define   TxPckDoneIen        0x00000008 //3 set to enable Tx Packet Done Interrupt

#define   DirTx               0x00000010 //4 set to enable Tx Buffers
#define   TermRx              0x00000020 //5 set to enable Rx Termination
#define   TermTx              0x00000040 //6 set to enable Tx Termination
#define   RxPckDoneIen        0x00000080 //7 set to enable Rx Packet Done Interrupt

#define   TxPckDelayMask      0x0000FF00 //15-8 8 bits to define delay for TX packets

#define   TxTimerEn           0x00010000 //16 set to enable TxTimer32 Function
#define   TxTimerIen          0x00020000 //17 set to enable TxTimer32 Interrupt
#define   TxTimerIen          0x00020000 //17 set to enable TxTimer32 Interrupt
#define   TxTimerEMsk         0x00040000 //18 TxTimer32 Enable Mask Control

#define   TxTimerMask         0x00300000 //21-20 set to control behavior of
                                         TxTimer/Tristate control

#define   DirRTS              0x01000000 //24 0 = Tristate, 1 = RTS signal driven
#define   ForceRTS            0x02000000 //25 0 = normal, 1 = Force RTS to block
#define   InvertFlowCntl      0x04000000 //26 0 = normal, 1 = invert RTS/CTS
#define   UseCTS              0x08000000 //27 0 = ignore CTS, 1 = Use Flow Control

#define   TermRTS             0x10000000 //28 0 = unterminated, 1 = terminated
#define   TermCTS             0x20000000 //29 0 = unterminated, 1 = terminated

#define   ReferenceSel        0x80000000 //31 0 = 32 MHz, 1 = PLL Reference
```

FIGURE 13 PMC-BISERIAL-VI-UART UART CHANB CONTROL

Note: All bits R/W. Undefined bits will return programmed value.

BreakRiseIen and **BreakFallIen** are used to select which edges of the Break detection status are used to generate latched status. Rising is associated with Break being asserted. Falling is associated with Break being removed.

BreakIen when set allows the captured [latched] status to generate an interrupt from the a change in state of Break. Clear the interrupt by writing with the corresponding bit set.

TxPckDoneIen when set ‘1’ gates the Tx Packet Done latched status through to generate an interrupt. Clear the interrupt by clearing the latched status or disabling this bit.

DirTx when set enables the external and internal buffers to transmit. Normally set to ‘1’. When set to ‘0’ the line level will tristate.

Note: the equivalent Rx control bit is set to receive in HW.

TermRx and **TermTx** when set cause the RS485 connection to have a 100 ohm resistor switched in. Analog switches are controlled to allow the parallel termination to be applied or not. Normal is Rx enabled ‘1’ and Tx not enabled ‘0’. If terminations are in the cable both maybe off. Under some system conditions both may need to be enabled.

RxPckDoneIen when set ‘1’ gates the Rx Packet Done latched status through to generate an interrupt. Clear the interrupt by clearing the latched status or disabling this bit.

TxPckDelayMask defines the field used to determine the number of bit periods to delay between packets when transmitting in packet mode. When set to x00 no additional delay is added. When set to x01, 1 bit time is added. Please note the HW requires several bit times of marking state to start a new packet when one completes. The programmed times are in addition to this HW defined delay. The delay is applied to the start of a packet to insure adeqate gap time when initially started. [Alt Packet Mode HW delay = 9, Standard Packet Mode HW delay = 11]

TxTimerEn when set enables the Timer32 function to count down using the stored Modulus. When the timer reaches 0x00 the counter reloads and repeats until disabled. At the zero crossing a pulse is generated which is latched for status/interrupt generation and optionally for setting the TxEnable [either the line enable, the function enable, both or neither]

TxTimerIen when set allows the captured [latched] status to generate an interrupt from the TxTimer32 function. See the Status register detail for the Latched Status Bit.

TxTimerMask when set ‘1’ TxTimer Strobe is masked with the Tx Data FIFO Empty status. If the FIFO is Empty when the strobe is asserted, TxEnable is not set. When this control bit is ‘0’ the FIFO status is not used, TxEnable is set at the end of the TxTimer count independent of the FIFO status.

Programming note: When TxTimer32 is selected to start transmission, TxEnable is set at the end of the programmed countdown. If not masked by the FIFO status, TxEnable can be set without data present leading to immeditate transmission of data when data is loaded. If the Timing strobe is required for the start of a burst of data the Mask should be used to make sure data is only transmitted immediately after the strobe from the timer function.

TxTimerMask defines the field used to determine the behavior of the Timer and Alternate Packet TX Enable/Disable bit.
x00 = no affect on TriState or TxEnable
x01= Use Alternate Packet Mode to disable TxEnable and TxTimer32 to Enable TxEnable
x10= Use Alternate Packet Mode to Tristate IO lines and enable IO lines, TxEnable not affected
x11= Alternate Packet Mode to Tristate IO and Disable TxEnable. Timer32 to enable TxEnable and Alternate Packet Mode to enable IO.

DirRTS when set '1' causes the external tristate driver to be enabled. Set this bit to use flow control. '0' can be selected when flow control is not used to save power.

ForceRTS when set '1' causes the RTS signal to be disabled logically. In a standard system RTS = '0' on the line to enable data transfer. If ForceRTS set is applied the level will be forced to '1'. Please note: InvertFlowCntl affects the definition. ForceRTS always causes no data transfer when asserted regardless of the standard/inverted selection.

The line state referenced is the P side. The N side will be in the opposite state.

InvertFlowCntl when set '1' causes the definition of RTS and CTS to be inverted – active high on the line to transfer and active low to block instead of the standard definition of high to block and low to transfer.

UseCTS when set '1' causes the transmitter [not test mode] to use the CTS signal for flow control. This affects unpacked, packed, packetized, and Alternate Packetized modes. '0' will cause the transmitter to ignore the state of the CTS input.

About RTS and CTS HW implementation. RTS when in non-forced normal mode is asserted low to allow data to transfer anytime the data level in the RX FIFO has 16 bytes or more. In unpacked mode this is 16 positions, in packed mode it is 4 since each LW has 4 bytes – HW automatically adjusts based on the mode selected. Most HW can stop transmitting within 12 bytes of RTS deassertion. PMC-BiSerial-VI-UART transmit function when CTS transitions to disabled will complete the current data word

and then stop. This means 1-2 bytes in unpacked mode and up to 4 in packed, packetized and alternate packetized modes depending on when CTS changes state.

TermRTS, TermCTS when set ‘1’ cause the terminations to be applied to the signals. Normally the TermCTS control is set to ‘1’ and TermRTS is cleared ‘0’.

ReferenceSel when ‘0’ selects the 32 MHz oscillator for the UART clock reference. When set ‘1’ the PLL associated with the port is used instead.
PLL Clock A is the alternate reference for ports 0,1.
PLL Clock B is the alternate reference for ports 2,3
PLL Clock C is the alternate reference for ports 4,5
PLL Clock D is the alternate reference for ports 6,7

The baud rate definitions use the selected clock to determine the frequency for transmit and expected frequency for receive.

The design is compiled with the max PLL clock set to 64 MHz. This corresponds to 4 Mbits/sec max guaranteed on the line. We have tested at greater than 6 MBits with success showing margin in the timing.

UART_CHAN_STAT

UART CHANNEL STATUS

```
#define     TxFfMt          0x00000001 //0 Transmit FIFO Empty
#define     TxFfAmt         0x00000002 //1 Almost Empty
#define     TxFfFl          0x00000004 //2 Full
#define     TxTimer32Lat    0x00000008 //3 Set when Timer32 function cycles

#define     RxFfMt          0x00000010 //4 Receive FIFO Empty
#define     RxFfAfl         0x00000020 //5 Almost Full
#define     RxFfFl          0x00000040 //6 Full
#define     RTSstatus       0x00000080 //7 current RTS level

#define     RxParErrLat     0x00000100 //8 -- status bits in each packet descriptor and
                                       latched here for non packet mode operation
#define     RxFrameErrLat   0x00000200 //9 -- status bits in each packet descriptor and
                                       latched here for non packet mode operation
#define     RxDataOvFlLt    0x00000400 //10
#define     RxPckOvFlLt     0x00000800 //11

#define     DmaWrErr        0x00001000 //12 Write (Tx) DMA Error
#define     DmaRdErr        0x00002000 //13 Read (Rx) DMA Error
#define     DmaWrDn         0x00004000 //14 Write DMA List Complete
#define     DmaRdDn         0x00008000 //15 Read DMA List Complete

#define     RxPckFifoMt     0x00010000 //16 Receive Packet FIFO Empty
#define     RxPckFifoFull   0x00020000 //17 Receive Packet FIFO Full
#define     TxPckFifoMt     0x00040000 //18 Transmit Packet FIFO Empty
#define     TxPckFifoFull   0x00080000 //19 Transmit Packet FIFO Full

#define     LocalInt        0x00100000 //20 Non DMA interrupt status
#define     IntStat         0x00200000 //21 All Interrupts status
#define     RxPckDoneLat    0x00400000 //22 Rx Packet Done Latched
#define     TxPckDoneLat    0x00800000 //23 Tx Packet Done Latched

#define     TxIdle          0x01000000 //24 Tx SM Idle State
#define     RxIdle          0x02000000 //25 Rx SM in Idle State
#define     BurstInIdle     0x04000000 //26 Tx DMA engine in Idle State
#define     BurstOutIdle    0x08000000 //27 Rx DMA engine in Idle State

#define     BreakStatLat    0x10000000 //28 -- Latched COS edge of Break Condition
#define     BreakStat       0x20000000 //29 -- Current Rx Break Status
#define     TxAmtLt         0x40000000 //30 -- Tx Almost Empty latched status
#define     RxAflLt         0x80000000 //31 -- Rx Almost Full latched status
```

FIGURE 14 PMC-BISERIAL-VI-UART UART STATUS

Transmit FIFO Empty: When a one is read, the transmit data FIFO for the corresponding channel contains no data; when a zero is read, there is at least one data-word in the FIFO.

Transmit FIFO Almost Empty: When a one is read, the number of data-words in the transmit data FIFO for the corresponding channel is less than or equal to the value written to the CHAN_FIFO_LVL register for that channel; when a zero is read, the level is more than that value.

Transmit FIFO Full: When a one is read, the transmit data FIFO for the corresponding channel is full; when a zero is read, there is room for at least one more data-word in the FIFO.

TxTimer32Lat: When a one is read the TxTimer32 function has downcounted to 0x00 and set this bit. If the interrupt enable is set the associated interrupt will also be set. Clear this bit by writing back to the status register with this bit set.

Receive FIFO Empty: When a one is read, the receive data FIFO for the corresponding channel contains no data; when a zero is read, there is at least one data-word in the FIFO. Please note: the count includes the DMA pipeline and can have up to 4 words available with an empty FIFO.

Receive FIFO Almost Full: When a one is read, the number of data-words in the receive data FIFO for the corresponding channel is greater or equal to the value written to the CHAN_FIFO_LVL register for that channel; when a zero is read, the level is less than that value.

Receive FIFO Full: When a one is read, the receive data FIFO for the corresponding channel is full; when a zero is read, there is room for at least one more data-word in the FIFO.

RTSstatus : reflects the state of the RTS signal prior to direction control potentially tri-stating. Affected by SW direct control, Rx Enable, and FIFO level.

Parity Error Detected: When a one is read, it indicates that a parity error has occurred since the status was last cleared. This bit is latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no parity error has occurred. Parity can be programmed to be odd, even, level or not implemented. An error indicates the received encoding does not match the programmed encoding.

Frame Error Detected: When a one is read, it indicates that a frame error has occurred since the status was last cleared. This bit is latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no frame error has occurred. A frame error occurs when the size of the received character including packaging does not match the programmed size.
Start bit is always 1 period wide
Data is 7 or 8 periods wide
Parity is 0 or 1 period wide
Stop Bits are either 1 or 2 minimum periods wide

Leading to the minimum character of 1+7+1 = 9 bits and the max of 1+8+1+2 = 12 bits. The Hardware automatically determines the expected size based on the parameters.

RxDataOvFlLt when set the Rx Data FIFO has had an overflow condition – FIFO is full when time to write the next data word. When cleared no error has occurred. This is a latched bit and is cleared by writing back with this bit position set.

RxDataOvFlLt: when set the Rx Packet FIFO has had an overflow condition – FIFO is full when time to write the next packet descriptor. When cleared no error has occurred. This is a latched bit and is cleared by writing back with this bit position set.

RxPckFifoMt : When a one is read, the receive Packet FIFO for the corresponding channel contains no data; when a zero is read, there is at least one descriptor in the FIFO.

RxPckFifoFl: When a one is read, the receive Packet FIFO for the corresponding channel is full; when a zero is read, there is room for at least one more descriptor in the FIFO.

TxPckFifoMt : When a one is read, the transmit Packet FIFO for the corresponding channel contains no data; when a zero is read, there is at least one descriptor in the FIFO.

TxPckFifoFl: When a one is read, the transmit Packet FIFO for the corresponding channel is full; when a zero is read, there is room for at least one more descriptor in the FIFO.

Write/Read DMA List Complete: When a one is read, it indicates that the corresponding DMA has completed. These bits are latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that the corresponding DMA has not completed.

Write/Read DMA Error: When a one is read, it indicates that an error has occurred while the corresponding DMA was in progress. This could be a target or master abort or an incorrect direction bit in one of the DMA descriptors. These bits are latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no DMA error has occurred.

Tx/Rx Idle: When a '1' is read, the corresponding function is in the Idle state. For changes of mode it is best if the State Machine is in the Idle state to make sure the mode is processed properly. Not all modes return to Idle as part of normal processing. The unpacked and packed modes in particular do not return to Idle unles the enable is cleared.

Write/Read DMA Idle: When a one is read the corresponding DMA State Machine is in the IDLE state. When '0' the DMA state machine is busy processing.

BreakStat is the synchronized line level of the Rx Break Status. Reading this value returns the current state of Break Status for this channel. When set a Break is currently in effect. When '0' break is not being received. Only has meaning when receiver is enabled and has made it through synchronization.

BreakStatLat is set when a programmed edge is captured based on the Break Status. If the rising edge is enabled, when a Break is detected the latch is set. If the falling edge is enabled the status is set when the status transitions low meaning the break is turned off. This is a sticky bit, cleared by writing back with the same bit position set.

TxAmtLt is set when the Transmit Data FIFO level <= the programmed Almost Empty number of words [set with CHAN_FIFO_LVL]. TxAmtLt is a sticky bit and is cleared by writing back with the bit position set.

RxAflLt is set when the Receive Data FIFO level >= the programmed Almost Full number of words [set with CHAN_FIFO_LVL]. RxAflLt is a sticky bit and is cleared by writing back with the bit position set.

RxPckDoneLat is a sticky bit set when a packet has been received. Cleared by writing back to the status register with this bit set. This signal can be enabled to generate an interrupt.

TxPckDoneLat is a sticky bit set when a packet has been transmitted. Cleared by writing back to the status register with this bit set. This signal can be enabled to generate an interrupt.

LocalInt when set indicates one of the non DMA interrupt requests is active. This is

after the individual interrupt masks and before the channel master interrupt enable.

IntStatus when set indicates this channel has a pending interrupt request. DMA and local Interrupts [after the master enable].

CHAN_TX_FIFO_CNT

TX FIFO Counts

```
#define    CHAN_PKT_CNT_MASK_TX      00FF0000 //
#define    CHAN_DATA_CNT_MASK_TX     0000FFFF //
```

FIGURE 15 PMC-BISERIAL-VI-UART TX FIFO COUNTS

Reading from this port returns the Packet and Data FIFO counts. The FIFO's are 255 deep. The counts are zero extended. It is recommended to program for a 16 bit field to allow for an increased FIFO size count without needing to change the driver.

CHAN_RX_FIFO_CNT

RX FIFO Counts

```
#define    CHAN_PKT_CNT_MASK_RX      00FF0000 //
#define    CHAN_DATA_CNT_MASK_RX     0000FFFF //
```

FIGURE 16 PMC-BISERIAL-VI-UART RX FIFO COUNTS

Reading from this port returns the Packet and Data FIFO counts. The FIFO's are 255 deep. There are an additional 4 locations in the DMA pipeline leading to a total of x103 possible locations. It is recommended to program for a 16 bit field to allow for an increased FIFO size count without needing to change the driver.

CHAN_TX_DMA_PNTR

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

FIGURE 17 CHANNEL WRITE DMA POINTER PORT

This read-write port is used to initiate a scatter-gather write [TX] DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads three successive long words beginning at that address. Essentially this data acts like a chaining descriptor value pointing to the next value in the chain. When read the current address is returned. Please note: this is the updated physical address where the HW is reading data from.

The first is the address of the first memory block of the DMA buffer containing the data to read into the device, the second is the length in bytes of that block, and the third 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 in one of the next pointer values read indicates that it is the last chaining descriptor in the list.

All three values are on LW boundaries and are LW in size. Addresses for successive parameters are incremented. The addresses are physical addresses the HW will use on the PCI bus to access the Host memory for the next descriptor or to read the data to be transmitted. In most OS you will need to convert from virtual to physical. The length parameter is a number of bytes, and must be on a LW divisible number of bytes.

Status for the DMA activity can be found in the channel control register and channel status register.

Notes:

1. Writing a zero to this port will abort a write DMA in progress.
2. End of chain should not be set for the address written to the DMA Pointer Address Register. End of chain should be set when the descriptor follows the last length parameter.

The Direction should be set to ‘0’ for Burst In DMA in all chaining descriptor locations.

CHAN_RX_DMA_PNTR

Write only

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

FIGURE 18 CHANNEL READ DMA POINTER PORT

This write-read port is used to initiate a scatter-gather read [RX] DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads three successive long words beginning at that address. Essentially this data acts like a chaining descriptor value pointing to the next value in the chain. When read the current physical address where the HW is writing data is returned.

The first is the address of the first memory block of the DMA buffer to write data from the device to, the second is the length in bytes of that block, and the third 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 in one of the next pointer values read indicates that it is the last chaining descriptor in the list.

All three values are on LW boundaries and are LW in size. Addresses for successive parameters are incremented. The addresses are physical addresses the HW will use on the PCI bus to access the Host memory for the next descriptor or to read the data to be transmitted. In most OS you will need to convert from virtual to physical. The length parameter is a number of bytes, and must be on a LW divisible number of bytes.

Status for the DMA activity can be found in the channel control register and channel status register.

Notes:

1. Writing a zero to this port will abort a write DMA in progress.
2. End of chain should not be set for the address written to the DMA Pointer Address Register. End of chain should be set when the descriptor follows the last length parameter.
3. The Direction should be set to ‘1’ for Burst Out DMA in all chaining descriptor locations.

NOTE: The direction bit (bit 1) must be set when the physical address of the first chaining descriptor is written to this register or a read DMA error will result.

CHAN_UART_FIFO

UART FIFO
#define CHAN_UART_FIFO_MASK_PACKED 0xFFFFFFFF // #define CHAN_UART_FIFO_MASK_UNPACKED 0x000000FF //

FIGURE 19 PMC-BISERIAL-VI-UART UART FIFO

Writing to the Chan Data FIFO or UART FIFO will load data for the transmitter to utilize. Data can be written in Packed, Unpacked, or Packetized formats.

Packed data has 4 bytes per LW loaded as shown with the corresponding Mask.

UnPacked data has 1 byte per LW loaded as shown with the corresponding Mask.

Packetized is a hybrid where Packed data is used for the data format with the exception of the last word which has 1, 2, 3, or 4 bytes loaded. The Packet FIFO is used to control the number of bytes sent per packet loaded.

Alternate Packetized uses up to 3 bytes per LW and the last LW has the descriptor control built in.

Packed is the most efficient data structure in terms of bytes loaded per LW used.
Packetized comes in second and as the total number of bytes in a packet increases becomes close to the efficiency of the Packed mode but with the flexibility of odd byte counts.
Alternate Packetized removes the need to write to the Packet FIFO. For medium size messages using DMA this may prove more effient than the Packetized mode.
UnPacked is the least efficient and the most flexible.

When reading from the CHAN_UART_FIFO address the data from the Rx Data FIFO is presented. The data is packed in the same manner as described above. Packed mode provides 32 bits per LW read, UnPacked returns data in the lower byte only, and Packetized/ Alternate Packetized a combination of Packed(3/4) and an odd length word depending on the size of the packet.

For non-Packed modes the non-loaded bytes are set to zero.

CHAN_TXFIFO_LVL

TX & RX FIFO Level		
#define	CHAN_TXAMT_FIFO_MASK	0x0000FFFF //

FIGURE 20 PMC-BISERIAL-VI-UART AMT LEVEL

CHAN_RXFIFO_LVL

TX & RX FIFO Level		
#define	CHAN_ RXAFL _FIFO_MASK	0x0000FFFF //

FIGURE 21 PMC-BISERIAL-VI-UART AFL LEVEL

The FIFO's are 255 deep. Unused bits should be set to zero when programming.

The TX mask is used to set the threshold for the Almost Empty condition. When the Count for the number of words in the FIFO is less than the programmed level the Almost Empty status becomes true.

The Rx mask is used to set the threshold for the Almost Full condition. When the count for the number of words in the Rx FIFO is equal or greater than the programmed level the Status is set.

For internal loop-back the Tx threshold should be set to at least 0x10 and Rx threshold set to xEF or less. The transfer engine for internal loop-back uses the almost full and almost empty status to determine if burst mode can be used. If the threshold is too small the transfer engine will not operate properly and attempt to do burst transfers when the FIFO's don't have enough room [RX or enough data TX].

CHAN_FRAME_TIME

Programmable Time Out		
#define	CHAN_FRAME_TIME_MASK	0x00FFFFFF //

FIGURE 22 PMC-BISERIAL-VI-UART FRAME TIME

CHAN_FRAME_TIME is a programmable count to determine how long to wait without a new character arriving for the receiver to declare “end of packet”. The count is based on the master clock [32 MHz or PLL as programmed in each channel]. The objective is to have a time long enough to be sure all characters belonging to a packet are captured into the same packet and short enough to complete the packet in a timely fashion. If the transmitter is capable of back-to-back character transmission a 2 character period would be sufficient. If the data is not so densely packed larger delays may be desired.

CHAN_BAUD_RATE

TX & RX Frequency		
#define	CHAN_TX_BAUD_MASK	0x0000FFFF //
#define	CHAN_RX_BAUD_MASK	0xFFFF0000 //

FIGURE 23 PMC-BISERIAL-VI-UART BAUD RATE

CHAN_BAUD_RATE is a programmable count to determine the frequency of operation. The master clock is the reference which can be 32 MHz or the user programmed PLL rate associated with this port. See Channel Control Register B. The count programmed [N-1] determines the frequency of transmission or reception plus adjusts some of the filtering aspects of the receiver.

Rate(based on 32 MHz)	Recommended Setting [N-1 shown]
2M	15
1M	31
500K	63
250K	127
125K	255
62.5K	511
31.25K	1023
9600	3332 (9600.96 actual frequency)

Using the PLL reference can provide more exact frequencies in some cases. Setting to 1.8432 MHz and using a divisor of 191 (192) will yield 9600 exactly.

CHAN_PACKET_FIFO

PACKET FIFO		
#define	CHAN_PKT_FIFO_MASK_TX	FFFF //
#define	CHAN_PKT_FIFO_MASK_RX	0FFF //

FIGURE 24 PMC-BISERIAL-VI-UART PACKET FIFO

Writing to the Chan Packet FIFO will load a descriptor into the TX Packet FIFO. The descriptor is the number of bytes to send from the TX Data FIFO. The transmitter will wait for additional data if the Data FIFO is empty when time to read more data to complete a packet allowing packet sizes larger than the FIFO. Since the FIFO can be loaded during transmission the Almost Empty Status can be used to trigger adding more data to extend a packet. If a zero value is read the packet descriptor is ignored. 1⇔FFFF bytes.

When reading from the Channel Packet FIFO the descriptors for the data in the Rx FIFO are read plus the status for the packet. The lower bits 11-0 are the size of the data in bytes and the upper bits 15-12 are the status captured for that packet.

15 RxParErrLat
14 RxFrameErrLat
13 RxDataOvFlLt
12 RxPckOvFlLt

The definitions are found in the Channel Status register description.

Packets on the receive side are limited to the size of 1⇔FFF bytes.

Programming notes: When in Packet Mode the Channel Packet FIFO interrupt can be used to detect when new descriptors have been written to the FIFO. If larger Packets are anticipated, the AFL Data FIFO interrupt can be used to read the data in as it is received and then parsed based on the descriptor when it is ready. The MT status or count can be used if polling is preferred; to determine when the descriptor is ready.

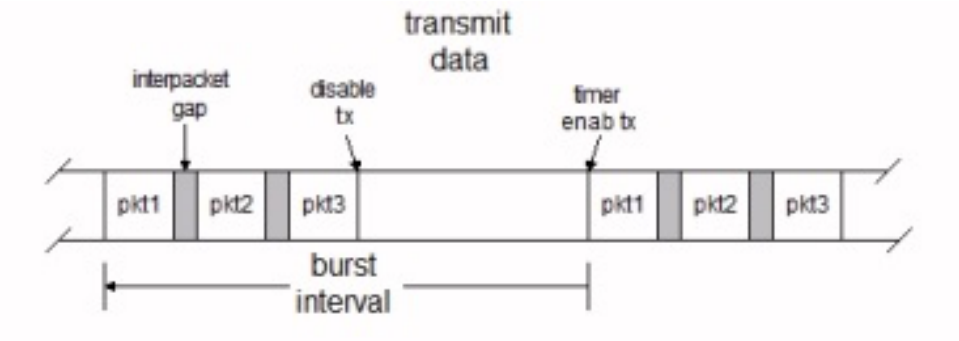
The Frame Timer should be programmed to determine the conditions for the end of frame. If left at the default setting packets will not be properly detected resulting in non-optimal behavior.

CHAN_TX_TIMER_MOD

Tx Timer Modulus Reg		
#define	CHAN_TX_TIMER_MOD_MASK	xFFFFFFFF //

FIGURE 25 PMC-BISERIAL-VI-UART TX MODULUS

This 32 bit R/W port stores the modulus used by TxTimer32 to define the range to count through. The reference clock is the selected channel clock of 32 MHz or PLL. There is a minimum count requirement of x5.



As shown in the diagram, the idea is to program the TxTimer for an interval longer than the combination of packets and inter-packet delays. The Packets can disable TxEnable as shown [end of Pkt3 in this case] and then be enabled by the TxTimer32 function.

Once the TxTimer32 function is enabled and running, writing to this register will cause a reload of the counter to the new value.

Please note: the timer can also be used as a system timer if TxMode is programmed to neither or Tristate control.

CHAN_TX_TIMER_CNT

Tx Timer Timer Reg		
#define	CHAN_TX_TIMER_CNT_MASK	xFFFFFFFF //

FIGURE 26 PMC-BISERIAL-VI-UART TX TIMER CNT

This 32 bit Read Only port allows the user to monitor the current count in the TxTimer32 function. The counter operates at the PLL or 32 MHz rate as programmed for the channel. The output is synchronized to the system reference clock.

LOOP-BACK & IO Connection Definitions - STD

PMC-BISERIAL-VI-UART can be used with direct end point cabling or with an interface. Dynamic Engineering uses HDEterm68 along with loop-back connections to accomplish loop-back. The standard version is compatible with UARTcable8 ⇔ SCSI to DB9 adapter. The DB9 pinout is designed for standard RS422 cabling.

The following table shows the connections the HDEterm68 used in the loop-back test. PMC-BiSerial-VI uses 32 Differential IO. Each UART uses 4 IO to create the TX, RX, RTS, and CTS connections. The reference SW uses loop-back within the same channel as a test mechanism. IO0, IO1, IO16, IO17 form UART 1.

Twisted Pair: Pins shown for P1 SCSI connector and match on HDEterm68 Numbers shown P1/Pn4 . For loop-back connections with rear IO use this table plus the Rear IO mapping table from the PMC carrier.

UART1_TXP	1/1	UART1_RXP	2/2
UART1_TXN	35/3	UART1_RXN	36/4
UART1_CTSP	17/33	UART1_RTSP	18/34
UART1_CTSN	51/35	UART1_RTSN	52/36
UART2_TXP	3/5	UART2_RXP	4/6
UART2_TXN	37/7	UART2_RXN	38/8
UART2_CTSP	19/37	UART2_RTSP	20/38
UART2_CTSN	53/39	UART2_RTSN	54/40
UART3_TXP	5/9	UART3_RXP	6/10
UART3_TXN	39/11	UART3_RXN	40/12
UART3_CTSP	21/41	UART3_RTSP	22/42
UART3_CTSN	55/43	UART3_RTSN	56/44
UART4_TXP	7/13	UART4_RXP	8/14
UART4_TXN	41/15	UART4_RXN	42/16
UART4_CTSP	23/45	UART4_RTSP	24/46
UART4_CTSN	57/47	UART4_RTSN	58/48

UART5_TXP	9/17	UART5_RXP	10/18
UART5_TXN	43/19	UART5_RXN	44/20
UART5_CTSP	25/49	UART5_RTSP	26/50
UART5_CTSN	59/51	UART5_RTSN	60/52
UART6_TXP	11/21	UART6_RXP	12/22
UART6_TXN	45/23	UART6_RXN	46/24
UART6_CTSP	27/53	UART6_RTSP	28/54
UART6_CTSN	61/55	UART6_RTSN	62/56
UART7_TXP	13/25	UART7_RXP	14/26
UART7_TXN	47/27	UART7_RXN	48/28
UART7_CTSP	29/57	UART7_RTSP	30/58
UART7_CTSN	63/59	UART7_RTSN	64/60
UART8_TXP	15/29	UART8_RXP	16/30
UART8_TXN	49/31	UART8_RXN	50/32
UART8_CTSP	31/61	UART8_RTSP	32/62
UART8_CTSN	65/63	UART8_RTSN	66/64

Dynamic Engineering Drivers and Reference SW include loop-back tests using the above connections.

LOOP-BACK & IO Connection Definitions – LM12

The LM12 version of PMC-BiSerial-VI is modified to incorporate IO definitions compatible with a third party design taken EOL. [Pinout to match Abaco / Radstone PMC-Q1F] Please note: additional signals not on the Q1F are present on the undefined IO. The following loopback table shows the connection changes from the base model.

The design number is also updated to allow SW to determine which model is connected.

Twisted Pair: Pins shown for P1 SCSI connector and match on HDEterm68. The alternate color pins are the ones modifed to match the Q1F design. To make room for UART1-4 other definitions changed too. All are marked.

UART1_TXP	2	UART1_RXP	4
UART1_TXN	36	UART1_RXN	38
UART1_CTSP	17	UART1_RTSP	18
UART1_CTSN	51	UART1_RTSN	52
UART2_TXP	8	UART2_RXP	10
UART2_TXN	42	UART2_RXN	44
UART2_CTSP	6	UART2_RTSP	20
UART2_CTSN	40	UART2_RTSN	54
UART3_TXP	13	UART3_RXP	15
UART3_TXN	47	UART3_RXN	49
UART3_CTSP	7	UART3_RTSP	22
UART3_CTSN	41	UART3_RTSN	56
UART4_TXP	19	UART4_RXP	21
UART4_TXN	53	UART4_RXN	55
UART4_CTSP	23	UART4_RTSP	24
UART4_CTSN	57	UART4_RTSN	58

UART5_TXP	9	UART5_RXP	5
UART5_TXN	43	UART5_RXN	39
UART5_CTSP	25	UART5_RTSP	26
UART5_CTSN	59	UART5_RTSN	60
UART6_TXP	11	UART6_RXP	12
UART6_TXN	45	UART6_RXN	46
UART6_CTSP	27	UART6_RTSP	28
UART6_CTSN	61	UART6_RTSN	62
UART7_TXP	1	UART7_RXP	14
UART7_TXN	35	UART7_RXN	48
UART7_CTSP	29	UART7_RTSP	30
UART7_CTSN	63	UART7_RTSN	64
UART8_TXP	3	UART8_RXP	16
UART8_TXN	37	UART8_RXN	50
UART8_CTSP	31	UART8_RTSP	32
UART8_CTSN	65	UART8_RTSN	66

End of LM12 connection table.

PMC PCI Pn1 Interface Pin Assignment

The figure below gives the pin assignments for the PMC Module PCI Pn1 Interface. See the User Manual for your carrier board for more information. Unused pins may be assigned by the specification and not needed by this design.

TCK	-12V	1	2
GND	INTA#	3	4
		5	6
BUSMODE1#	+5V	7	8
		9	10
GND		11	12
CLK	GND	13	14
GND		15	16
	+5V	17	18
	AD31	19	20
AD28	AD27	21	22
AD25	GND	23	24
GND	C/BE3#	25	26
AD22	AD21	27	28
AD19	+5V	29	30
	AD17	31	32
FRAME#	GND	33	34
GND	IRDY#	35	36
DEVSEL#	+5V	37	38
GND	LOCK#	39	40
		41	42
PAR	GND	43	44
	AD15	45	46
AD12	AD11	47	48
AD9	+5V	49	50
GND	C/BE0#	51	52
AD6	AD5	53	54
AD4	GND	55	56
	AD3	57	58
AD2	AD1	59	60
	+5V	61	62
GND		63	64

FIGURE 27 PMC-BISERIAL-VI-UART PN1 INTERFACE

PMC PCI Pn2 Interface Pin Assignment

The figure below gives the pin assignments for the PMC Module PCI Pn2 Interface. See the User Manual for your carrier board for more information. Unused pins may be assigned by the specification and not needed by this design.

+12V		1	2
TMS	TDO	3	4
TDI	GND	5	6
GND		7	8
		9	10
	+3.3V	11	12
RST#	BUSMODE3#	13	14
+3.3V	BUSMODE4#	15	16
	GND	17	18
AD30	AD29	19	20
GND	AD26	21	22
AD24	+3.3V	23	24
IDSEL	AD23	25	26
+3.3V	AD20	27	28
AD18		29	30
AD16	C/BE2#	31	32
GND		33	34
TRDY#	+3.3V	35	36
GND	STOP#	37	38
PERR#	GND	39	40
+3.3V	SERR#	41	42
C/BE1#	GND	43	44
AD14	AD13	45	46
GND	AD10	47	48
AD8	+3.3V	49	50
AD7		51	52
+3.3V		53	54
	GND	55	56
		57	58
GND		59	60
	+3.3V	61	62
GND		63	64

FIGURE 28 PMC-BISERIAL-VI-UART PN2 INTERFACE

Applications Guide

Interfacing

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

ESD
Proper ESD handling procedures must be followed when handling the PMC-BISERIAL-VI-UART. The card is shipped in an anti-static, shielded bag. The card should remain in the bag until ready for use. When installing the card the installer must be properly grounded and the hardware should be on an anti-static workstation.

Start-up
Make sure that the "system" can see your hardware before trying to access it. Many BIOS will display the PCI devices found at boot up on a "splash screen" with the VendorID and CardId and an interrupt level. Look quickly, if the information is not available from the BIOS then a third party PCI device cataloging tool will be helpful.

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.

We provide the components. You provide the system. Only careful planning and practice can achieve safety and reliability. Inputs can be damaged by static discharge, or by applying voltage outside of the device rated voltages.

Construction and Reliability

Dynamic Engineering Modules are conceived and engineered for rugged industrial environments. PMC-BISERIAL-VI-UART is constructed out of 0.062-inch thick High-Temp ROHS compliant FR4 material.

ROHS and standard processing are available options.

Through-hole and surface-mount components are used. PMC connectors are rated at 1 Amp per pin, 100 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable.

PMCs are secured against the carrier with four screws attached to the 2 stand-offs and 2 locations on the front panel. The four screws provide significant protection against shock, vibration, and incomplete insertion.

The PCB provides a (typical based on PMC) low temperature coefficient of 2.17 W/°C for uniform heat. This is based upon the temperature coefficient of the base FR4 material of 0.31 W/m-°C, and taking into account the thickness and area of the board. The coefficient means that if 2.17 Watts are applied uniformly on the component side, then the temperature difference between the component side and solder side is one degree Celsius.

PMC-BISERIAL-VI-UART has internal thermal planes made up of heavy copper power and ground planes. The planes will spread the thermal load over the entire board to minimize hotspots and increase the “coolability”. The components are Industrial temperature rated or better. Thermal vias are added under components to tie in with the thermal plane directly. Where possible devices with thermal ties were chosen to allow direct connection to the ground plane.

Thermal Considerations

The PMC-BISERIAL-VI-UART design consists of CMOS 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, then forced-air cooling 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. 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 Street, Suite C
Santa Cruz, CA 95060
831-457-8891
support@dyneng.com

Specifications

Host Interface (PCI):	PCI Interface 33 MHz. 32-bit
Serial Interfaces:	8 UART channels each with Rx,Tx, RTS, CTS signals
TX Bit-rates generated:	user programmable for each UART channel with the standard baud rates up to 2M and custom programmed rates.
Software Interface:	Control Registers, FIFO's, and Status Ports
Initialization:	Hardware reset forces all registers to 0 except as noted
Access Modes:	Long-word boundary space (see memory map)
Wait States:	One for all addresses
Interrupt:	Multiple programmable interrupts per channel for flow control and error recognition.
DMA:	Indendent controllers for each port TX and each port RX [16 total]
Onboard Options:	All Options are Software Programmable
Interface Options :	Front or Rear IO. Front IO via P1 SCSI connector. Rear IO through Pn4.
Dimensions:	Standard Single PMC.
Construction:	High Temp ROHS compliant FR4 Multi-Layer Printed Circuit, Through-Hole and Surface-Mount Components
Temperature Coefficient:	2.17 W/^{o}C for uniform heat across PMC [similar for other formats]
Power	TBD

Order Information

Please refer to our PMC-BISERIAL-VI-UART webpage for the most up to date information:
https://www.dyneng.com/pmc_biserial_VI.html

PMC-BISERIAL-VI-UART	Standard version with 8 UARTs, each with Rx, Tx, RTS, & CTS RS-422/485 signals supported. Programmable for any baud rate 3.125M ⇔ 150, programmable character length[7,8], stop bits[1,2], parity[odd, even, level, none]. 255x32 FIFO per Tx and Rx. Test, Packet, Alt Packet, Packed, and UnPacked operation supported. 32 bit system timer per port. Programmable delay to start/restart transmission. non-ROHS assembly. Industrial temperature components standard. 32 MHz reference plus PLL[user frequency]
-LM12	Same as above with alternate Pinout to match Abaco / Radstone PMC-Q1F
-LVDS	Change to LVDS IO instead of RS485.
-RIO	Change to Pn4 IO instead of SCSI connector.
-CC	Add conformal coating option. Recommended for condensing or near condensing environments
-ROHS	Leaded solder is standard on this product. Add -ROHS for ROHS processing.
HDEterm68	https://www.dyneng.com/HDEterm68.html is available as a breakout or for loop-back purposes. Available with several options including connector orientation, DIN rails, Terminal Block, header strip.
HDEcabl68	SCSI cable suitable to interconnect PMC BiSerial III and HDEterm68. Available in various lengths. Twisted shielded construction.

Glossary

Acronyms and other specialized names and their meaning:

PMC	PCI Mezzanine Card - establishes common connectors, connections, size and other mechanical features.
PCI	Peripheral Component Interconnect – parallel bus from host to this device.
VendorID	Manufacturers number for PCI/PCIe boards. DCBA is Dynamic Engineering's ID.
CardID	Unique number assigned to design to distinguish between all designs of a particular vendor.
UART	Universal Asynchronous Receiver Transmitter. Common serialized data transfer with start bit, stop bit, optional parity, optional 7/8 bit data. Can be over any electrical interface. RS232 and RS422 are most common.
Baud	Used as the bit period for this document. Not strictly correct but is the common usage when talking about UART's.
FIFO	First In First Out Memory
JTAG	Joint Test Action Group – a standard used to control serial data transfer for test and programming operations.
TAP	Test Access Port – basically a multi-state port that can be controlled with JTAG [TMS, TDI, TDO, TCK]. The TAP States are the states in the State machine controlled by the commands received over the JTAG link.

TMS	Test Mode State – this serial line provides the state switching controls. '1' indicates to move to the next state, '0' means stay put in cases where delays can happen, otherwise 0,1 are used to choose which branch to take. Due to complexity of state manipulation the instructions are usually precompiled. Rising edge of TCK valid.
TDI	Test Data In - this serial line provides the data input to the device controlled by the TMS commands. For example the data to program the FLASH comes on the TDI line while the commands to the state-machine to move through the necessary states comes over TMS. Rising edge of TCK valid.
TCK	Test Clock provides the synchronization for the TDI, TDO and TMS signals
TDO	Test Data Out is the shifted data out. Valid on the falling edge of TCK. Not all states output data.
Packet	Group of characters transferred. When the characteristics of a group of characters is known the data can be stored in packets, transferred as such and the system optimized as a result. Any number of characters can be sent.
Packed	When UART characters are always sent/received in groups of 4 allowing full use of host bus / FIFO bandwidth.
UnPacked	When UART characters are sent on an unknown basis requiring single character storage and transfer over the host bus.
MUX	Multiplexor – multiple signals multiplexed to one with a selection mechanism to control which path is active.
Flash	Non-volatile memory used on Dynamic Engineering boards to store FPGA configurations or BIOS.