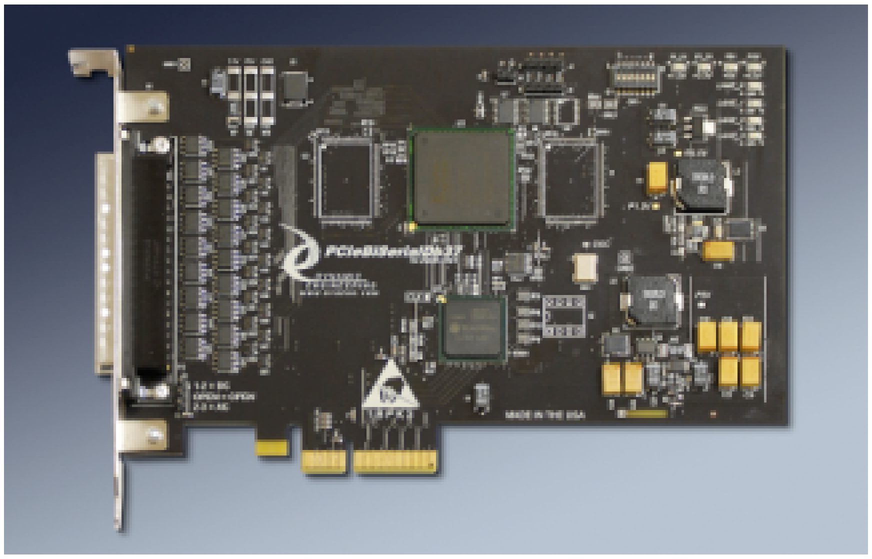
DYNAMIC ENGINEERING
150 DuBois St. Suite C, Santa Cruz, CA 95060
831-457-8891 Fax 831-457-4793
http://www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

PCIeBiSerialDb37-LM9

ARC210 IF / Parallel Port
PCIe 4 lane Module
Transmit and Receive Interface Protocols
RS485/422



Revision A1
Corresponding Hardware: Revision 1
10-2009-0401
FLASH 0101

PCIeBiSerialDb37LM9

ARC-210 and Digital Parallel Interface
PCIe Module

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

Manual Revision A. Revised 10/9/09

Table of Contents

List of Figures

Product Description

In embedded systems many of the interconnections are made with differential [RS-422/485 or LVDS] signals. Depending on the system architecture an IP, PMC or native bus card will be the right choice to make the connection. You have choices with carriers for cPCI, PCI, PCIe, VME, PC/104p and other buses for both PMC and IP mezzanine modules.

The BiSerial family includes IP, PMC and PCIe versions, each with multiple “clientized” design implementations.

Usually the choice of format is based on other system constraints. Dynamic Engineering is happy to assist in your decision regarding architecture and other trade-offs with the form factor decision. Dynamic Engineering has carriers for IP and PMC modules for most systems, and is adding more as new solutions are requested by our clients.

The PCIe compatible PCIeBiSerialDb37 has 18 independent differential IO available. A DB-37 connector is mounted through the bezel to carry the signals. Each of the IO have independent direction and termination controls. Each of the IO are matched length and routed with 100 Ω differential impedance.

The IO are buffered from the FPGA with differential transceivers. The transceivers can be populated with LVDS or RS-485 compatible devices. The power plane for the transceivers is isolated to allow selectable 3.3 or 5V references for the IO. The LVDS IO requires 3.3, and 40 MHz capable RS-485 requires 5V. When mixed LVDS and RS485 are used the reference is set to 3.3 and lower speed RS-485 parts are used that are compatible with the 3.3V.

Each IO has pull-up and pull-down options to allow half duplex lines to be set to a “marking” state when no device is on the line. The P is is ganged and the M side is too. Each side can be set to gnd or vcc to allow a ‘1’ or a ‘0’ to be set on the lines. The resistors are in resistor packs and can be implemented with many values.

The terminations utilize analog switches to selectively parallel terminate the differential pair with approximately 100 ohms. It is recommended that the receiver side provide the termination.

The analog switches are protected with a DIODE on the input side of the power supply. The switches can back-feed voltage into the rest of the circuit when powered down and the system connected to it is not. The DIODE’s allow for more flexible operation and power sequencing.

PcieBiSerialDb37Lm9 is a “clientized” version of the standard PCIeBiSerialDb37 board.

"LM9" is set to use the RS485/RS422 standard, and supports one Transmit and one Receive channel plus a parallel port. The transmitter and receiver are designed to interface with the ARC-210 radio.

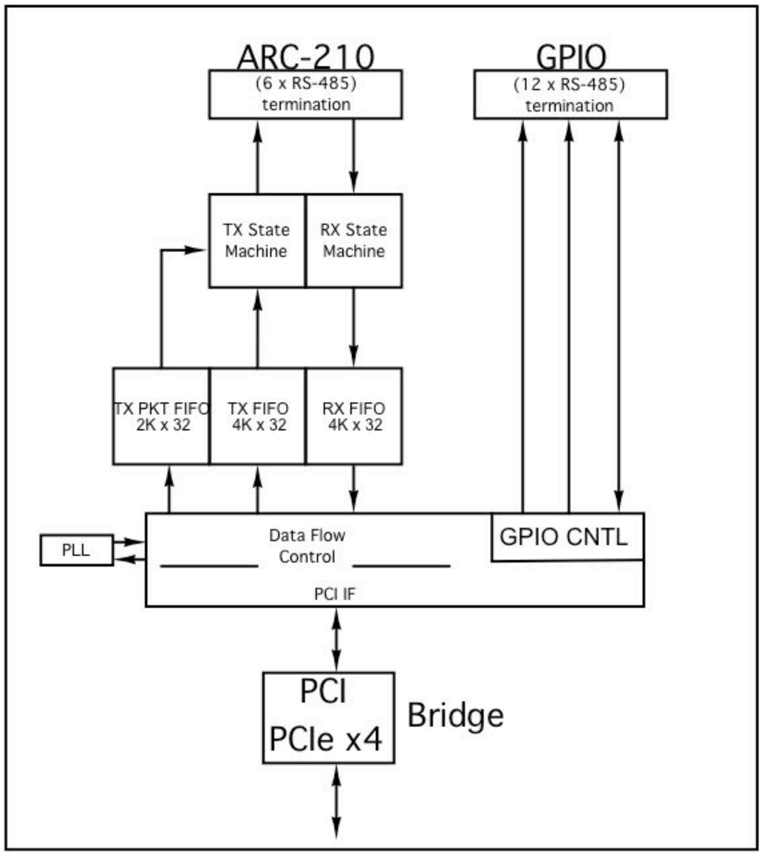


Figure 1 PCIEBISERIALDB37LM9 Block Diagram

The ARC-210 will supply the "SendTiming" and "ReceiveTiming" clock reference signals. The PLL can be used to create a Tx clock reference to allow for loop-back testing. The control is via SW. The PLL is referenced to 50 MHz. and can be programmed with new .JED files using the driver.

The ARC-210 interface is serial with an interface similar to a UART. The Start and Marking bits have opposite polarity with the Start bit being active high '1' and the marking state being a "0" on the lines. The hardware is programmable to allow the Start bit to be active low and the marking state to be active high. Data is transferred as byte wide with a reference clock. The reference clock is received from the radio. The default is to use the rising edge of the reference clock with data stable on the falling edge. The clock polarity is programmable to allow the active and stable edges to be reversed. Parity is added with a programmable Odd or Even definition. The transmitter will assert RTS and wait for CTS before transmitting data.

The number of bytes to send is programmable. The number can be stored into a secondary FIFO to allow multiple packets to be transmitted from a single larger DMA transfer. The number of bytes can also be stored into a register to cover cases where the size of each packet is consistent.

The hardware waits until there is a definition of the byte count, and data in the transmit FIFO before starting. Once started if the data FIFO is empty when the transmitter is ready to read the next data set an error for underflow is flagged. The error can cause an interrupt if desired.

The receiver is designed to be always ready when the radio transmits data to the LM9 interface. The receiver channel when enabled can accept data from the radio independent of the transmitter. This allows for loop-back or other applications. The ARC-210 can only transmit or receive and not both at the same time. The receiver will only receive data when the transmitter is not operating – in a system situation.

The receiver uses a 33 MHz clock to sample the received data and clock and to control the data flow internally. The expected ReceiveTiming rate is around 1 MHz. leaving plenty of headroom for the sampling and data control.

The received timing input is converted to a series of 33 MHz. pulses which are used to enable the receive shift register to capture the synchronized Receive Data. As the data is captured the marking state is checked to allow for initial start-up [10 bits+ of marking bits in a row]. The state-machines check if in the marking state and then if a start bit has been received. Data bits are counted and when a complete Start Byte Parity sequence is received the data is parallel loaded to a holding register. Parity is calculated and tested against what was received. The data is moved to a secondary holding register to build up to a 32 bit word. When 4 bytes are received or at the end of a programmed length the data is moved to the output FIFO.

Both the transmitter and receiver allow for bit and byte reversal. The data is stored as 32 bit words into the transmit FIFO from the system or the receive FIFO from the interface. The data is used with little Endian conventions as the default – 0,1,2,3 for the byte order where 0 = D7-0 [data on AD7-0] first and D31-24 last. The bits are sent LSB first so D0 is first on the line and D31 is last if all 4 bytes are to be sent. Similarly the receiver loads 0,1,2,3 so the first bit in goes into D0 and the last into D31 for each long word. When the bytes are reversed the order becomes 3,2,1,0 which which would make the IO 24 first, 7 last since it is still lsb first. The bit reversal swaps D7 with D0 etc on each byte read so the order becomes D31 first and D0 last if both reversal options are selected. For systems using Windows Little Endian is consistent with the driver and memory mapping. With Linux and some RISC based systems the reversal may be necessary.

The transmitter and receiver have 4K x 32 FIFO’s with direct access, DMA support, and loop-back capabilities. The driver/reference software includes examples of all three modes.

DMA when used will keep the transmitter FIFO full and the receiver FIFO empty automatically and without overflow etc. The hardware has the necessary flow control designed in.

In addition the transmitter side has a 2K x 32 Packet FIFO where each entry stored represents the number of bytes to send in a given transmission. RTS is held on during a packet transmission. RTS is released between transmissions and then reasserted and checked. CTS is not checked during the transmission since the radio is unidirectional and has no way to receive a “pause” from the “other end”. Smaller packets can be used for more frequent checking.

For situations where more consistent and likely shorter transmissions are to happen the registered byte count can be used. The hardware will repeat the same size packet with new data from the data FIFO as long as there is data in the FIFO.

Please note that the Data in the FIFO’s are packed on a Packet basis. The last LW is padded as necessary to reach 32 bits. On the receive side the storage register is cleared between each long word to pad with 0’s in each location not used.

Similarly the RX Packet FIFO is used to store the number if bytes in each packet received. The RX Packet FIFO is also 2K x 32.

The receive Packet size can be determined by programming the local Packet Length and selection of the Packet length mode or by using the timeout function and having a variable length packet automatically received.

Custom cables can be manufactured to your requirements. The loop-back IO definitions are toward the end of this manual. Please contact Dynamic Engineering with your specifications.

In the “LM9” design the Termination and Direction controls are set in the VHDL for the ARC-210 IO and programmable with software for the GPIO. The received signals are terminated and the transmitted signals are not.

All of the IO are routed through the FPGA to allow for custom applications. Larger external and internal FIFO’s and Dual Ported memories can be implemented with different FPGA selections and adding the 128K x 32 FIFO’s to the board.

The registers are mapped as 32 bit words and support 32 bit access. Most registers are read-writeable. The Windows® and Linux compatible drivers are available to provide the system level interface for this version of the Biserial. Use standard C/C++ to control your hardware or use the Hardware manual to make your own software interface. The software manuals are also available on-line.

PcieBiserialDb37 can be used for multiple purposes with applications in telecommunications, control, sensors, IO, test; anywhere multiple independent or coordinated IO are useful.

PcieBiserialDb37 features a Xilinx FPGA, and high speed differential devices. The FPGA contains the PCI interface and control required for the parallel interface.

The Xilinx design incorporates the "PCI Core" and additional modules for DMA in parallel with a direct register decoded programming model. The design model has a "base" level with the basic board level functions and "channels" which contain IO oriented functions. In the LM9 design the ARC-210 functions are designed into the channel and the PLL programming, switch, GPIO and other common or basic functions are in the base design.

From a software perspective the design can be treated as "Flat" or as a hierarchy. The Dynamic Engineering Windows® driver uses the hierarchical approach to allow for more consistent software with common bit maps and offsets. This implentation has only one channel. The channel function was kept to allow for future expansion with more than 1 ARC-210 interface or a secondary function in added channels. The user software can control the Channels with the same calls and use the channel number to distinguish. This makes for consistent and easier to implement user level software.

The hardware is designed with each of the channels on a common address map – each channel has the same memory allocated to it and as much as possible the offsets within each space are defined in the same way or similar way. Again this make understanding each port easier to accomplish and less likely to have errors.

The transceivers are initialized to the receive state. Once a channel is defined via software to be a transmitter the IO are enabled and driven to the appropriate levels. Terminations are activated for ports defined to be receivers.

PcieBiserialDb37 is part of the PCIe Module family of modular I/O components. The PcieBiserialDb37 conforms to the PCIe standard. This guarantees compatibility with your PCIe system. The base is 4 lane operation. The design can handle 1-4 lanes being available. LED's are provided to show the active PCIe lanes.

Designs implemented on PC104p, PMC, IP and PCIe versions of the BiSerial family can in large part be ported between platforms. If you see what you need in one version and prefer it on another please contact Dynamic Engineering about porting the design. In most cases it will require a recompile of the VHDL and not much more. We do a lot of "just like but different " adaptations for our clients. Please contact us to help you with a successful special adaptation of off- the-shelf hardware.

The DMA programmable length is 32 bits => longer than most computer OS will allow in one segment of memory. The DMA is scatter gather capable for longer lengths than the OS max and for OS situations where the memory is not contiguous. With Windows® lengths of 4K are common while Linux can provide much larger spaces. Larger spaces are more efficient as there are fewer initialization reads and reduced overhead on the bus. A single interrupt can control the entire transfer. Head to tail operation can also be programmed with two memory spaces with two interrupts per loop.

The hardware is organized with the IO function in channel 0 and the card level functions in the "base". The driver provides the ability to find the hardware and to allocate resources to use the base and channel functions.

The basic use of the interface is to facilitate data transfer via ARC-210 radio between the host and the remote target.

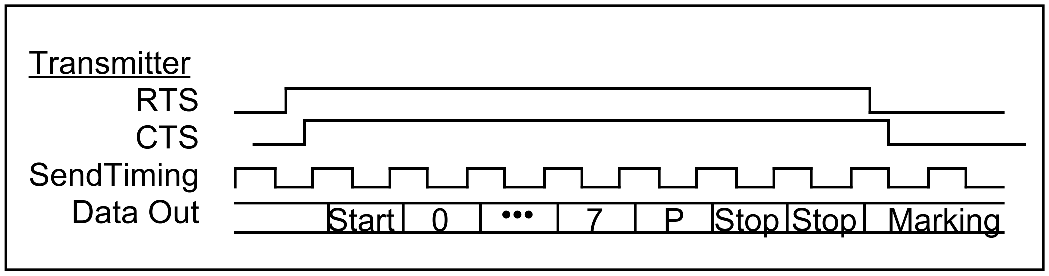


Figure 2 PCIEBISERIALDB37LM9 Timing Diagram

Transmitter side timing is shown. The SendTiming signal is the reference clock and is supplied by the Radio [ARC-210] and is free running. RTS is asserted when the host has data to transmit via radio. The radio responds with CTS [or could already be asserted]. Data is sent with a start bit, byte of data, parity and then 2 stop bits. If more bytes are available to send, the next byte would start after the two stop bits. If no more data then the stop bits are continued [and called marking].

Address Map

```
Function                                         Offset
// PCIeBiSerialDb37LM9 BASE definitions
#define LM9_BASE_BASE              0x0000 // 0 LM9Base Base control register
#define LM9_BASE_PLL_WRITE         0x0000 // 0 LM9Base Base control register
#define LM9_BASE_PLL_READ          0x0000 // 0 LM9Base base control register
#define LM9_BASE_USER_SWITCH       0x0004 // 1 LM9Base User DIP switch read
#define LM9_BASE_XILINX_REV        0x0004 // 1 LM9Base Xilinx revision read port
#define LM9_BASE_XILINX_DES        0x0004 // 1 LM9Base Xilinx design read port

#define LM9_BASE_STATUS            0x0008 // 2 LM9Base status Register offset

// 17-6 IO numbering, 11-0 register numbering for a 12 bit GPIO interface with an IO offset
#define LM9_BASE_GPIO_TERM         0x0040 // 16 LM9Base GPIO Termination Register offset
#define LM9_BASE_GPIO_DIR          0x0044 // 17 LM9Base GPIO Direction Register offset
#define LM9_BASE_GPIO_DATA         0x0048 // 18 LM9Base GPIO Data Register offset --
register data R/W
#define LM9_BASE_GPIO_IO           0x004C // 19 LM9Base GPIO Data IO offset (read only)
```

Figure 3 PCIeBiSerialDb37LM9 Internal Address Map Base Functions

The address map provided is for the local decoding performed within PcieBiserialDb37Lm9. The addresses are all offsets from a base address. Dynamic Engineering prefers a long-word oriented approach because it is more consistent across platforms.

The map is presented with the #define style to allow cutting and pasting into many compilers “include” files.

The host system will search the PCI bus to find the assets installed during power-on initialization. The VendorId = 0x10EE and the CardId = 0x003E for the PcieBiSerialDb37Lm9.

The LM9 design has 1 channel implemented at this time. The BASE contains the common elements of the design, while the Channels have the IO specific interfaces. The BASE starts at the card offset. Channel 0 starts at register 20

Section	Register Address Range (starting Hex address)	COM name
Base	0-19 (0x0000)	PLL, Switch, Status
Channel 0	20-39 (0x0050)	ARC-210 Transmitter & Receiver

Function	**Offset from Channel Base Address**
// **PCIeBiSerialDb37LM9 Channel definitions**	
#define LM9_CHAN_CNTRL	0x00000000 //0 LM9Chan General control register
#define LM9_CHAN_STATUS	0x00000004 //1 LM9Chan Interrupt status port
#define LM9_CHAN_INT_CLEAR	0x00000004 //1 LM9Chan Interrupt clear port
#define LM9_CHAN_WR_DMA_PNTR	0x00000008 //2 LM9Chan Write DMA dpr physical PCI address register
#define LM9_CHAN_TX_FIFO_COUNT	0x00000008 //2 LM9Chan TX FIFO count read port
#define LM9_CHAN_RD_DMA_PNTR	0x0000000C //3 LM9Chan Read DMA physical PCI address register
#define LM9_CHAN_RX_FIFO_COUNT	0x0000000C //3 LM9Chan RX FIFO count read port including pipeline
#define LM9_CHAN_FIFO	0x00000010 //4 LM9Chan FIFO for single word RW
#define LM9_CHAN_TX_AMT_LVL	0x00000014 //5 LM9Chan TX almost empty level RW
#define LM9_CHAN_RX_AFL_LVL	0x00000018 //6 LM9Chan RX almost full level register RW, used for HW control
#define LM9_CHAN_TX	0x0000001C //7 LM9Chan TX control register
#define LM9_CHAN_TX_PACKET_LEN_FIFO	0x00000020 //8 LM9Chan Packet Length Tx FIFO
#define LM9_CHAN_TX_PACKET_LEN_REG	0x00000024 //9 LM9Chan Packet Length Tx Register
#define LM9_CHAN_RX	0x00000034 //13 LM9Chan RX control register
#define LM9_CHAN_RX_PACKET_LEN_FIFO	0x00000038 //14 LM9Chan Packet Length Rx FIFO
#define LM9_CHAN_RX_TIMEOUT_LEN	0x0000003C //15 LM9Chan Time out between packets
#define LM9_CHAN_RX_BYTECOUNT_LEN	0x00000040 //16 LM9Chan Byte Count length expected

Figure 4 PcieBiSerialDb37Lm9 Channel Address Map

Programming

Programming the PcieBiSerialDb37Lm9 requires only the ability to read and write data in the host's PCIe space.

Once the initialization process has occurred, and the system has assigned addresses to the PcieBiSerialDb37Lm9 card the software will need to determine what the address space is for the PCI interface [BAR0]. The offsets in the address tables are relative to the system assigned BAR0 base address.

The next step is to initialize the PcieBiSerialDb37Lm9. The PLL will need to be programmed to use the loop-back function. The Cypress CyberClocks software can be used to create new .JED files if desired. PLLA should be set to the transmit reference frequency output by the transmitter.

The driver comes with a .JED file prepared. The driver has a utility to load the PLL and read back. The reference application software has an example of the use of PLL programming. The reference application software also includes XLATE.c which converts the .JED file from the CyberClocks tool to an array that can be programmed into the PLL.

The IO for the ARC-210 direction and termination are hardwired in this design. The ports are unidirectional and initialization is simplified with this approach. The GPIO ports can be programmed if used. The termination should be set for any input bits [unless the termination is located in your cable] and any transmitters will need to be enabled.

The control bits for the ARC-210 will select how the data is transmitted – Byte ordering, size of transfer etc.

For Windows™ and Linux systems the Dynamic Drivers can be used. The driver will take care of finding the hardware and provide an easy to use mechanism to program the hardware. The Driver comes with reference software showing how to use the card and reference frequency files to allow the user to duplicate the test set-up used in manufacturing at Dynamic Engineering. Using simple, known to work routines is a good way to get acquainted with new hardware.

To use the LM9 specific functions the Channel Control, and PLL interface plus DMA will need to be programmed. To use DMA, memory space from the system should be allocated and the link list stored into memory. The location of the link list is written to the LM9 to start the DMA. Please refer to the Burst IN and Burst Out register discussions.

DMA should be set-up before starting the channel port function. For transmission this will result in the FIFO being full or close to it when the transfer is started or at least the Packet loaded if shorter than the FIFO size. For reception it means that the FIFO is under HW control and the delay from starting reception to starting DMA won't cause an overflow condition.

DMA can be programmed with a specific length. The length can be as long as you want within standard memory limitations. At the end of the DMA transfer the Host will receive an interrupt. The receiver can be stopped and the FIFO reset to clear out any extra data captured. For on-the-fly processing multiple shorter DMA segments can be programmed; at the interrupt restart DMA to point at the alternate segment to allow processing on the previous one. This technique is sometimes referred to as "ping-pong".

The Receive Byte Count register can be used to control the input Packet size to work with your DMA scheme. Alternately the Byte Count register can be programmed to the size of the Packet if known. For situations where the size is unknown the timeout and Almost Full interrupt options can be used.

Please see the channel control register bit maps for more information.

Base Register Definitions

LM9_BASE_BASE

[$00 Base Control Register Port read/write]

DATA BIT	DESCRIPTION
31-21	spare
20	bit 19 read-back of pll_dat register bit
19	pll_dat [write to PLL, read-back from PLL]
18	pll_s2
17	pll_sclk
16	pll_en
15-0	spare

Figure 5 PcieBiSerialDb37Lm9 Control Base Register Bit Map

This is the base control register for the LM9. The features common to all channels are controlled from this port. Unused bits are reserved for additional new features. Unused bits should be programmed '0' to allow for future commonality.

pll_en: When this bit is set to a one, the signals used to program and read the PLL are enabled.

pll_sclk/pll_dat : These signals are used to program the PLL over the I^2C serial interface. Sclk is always an output whereas Sdata is bi-directional. This register is where the Sdata output value is specified or read-back.

pll_s2: This is an additional control line to the PLL that can be used to select additional pre-programmed frequencies. Set to '0' for most applications.

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 50 MHz.

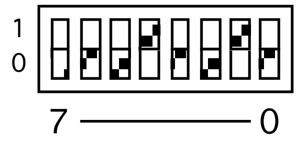
LM9_BASE_ID

[$04 Switch and Design number port read only]

DATA BIT	DESCRIPTION
31-24	spare
23-8	Design ID and Revision
7-0	DIP switch

Figure 6 PcieBiSerialDb37Lm9 ID and Switch Bit Map

The DIP Switch is labeled for bit number and ‘1’ ‘0’ in the silk screen. The DIP Switch can be read from this port and used to determine which PcieBiserialDb37Lm9 physical card matches each PCI address assigned in a system with multiple cards installed. The DIPswitch can also be used for other purposes – software revision etc. The switch shown would read back 0x12.



The Design ID and Revision are defined by a 16 bit field allowing for 256 designs and 256 revisions of each. The LM9 design is 0x01 the current revision is 0x01.

The PCI revision is updated in HW to match the design revision. The board ID will be updated for major changes to allow drivers to differentiate between revisions and applications.

LM9_BASE_STATUS

[$08 Board level Status Port read only]

DATA BIT	DESCRIPTION
31-10	set to ‘0
9	undefined
8	undefined
7-1	set to ‘0’, reserved for additional channels
0	Unmasked Ch0 Interrupt

Figure 7 PcieBiSerialDb37Lm9 Status Port Bit Map

Channel Interrupt – The local interrupt status from the channel. Each channel can have different interrupt sources. DMA Write or DMA Read or IntForce or TX/RX request are typical sources. Polling can be accomplished using the channel status register and leaving the channel interrupt disabled.

Channel Bit Maps

The LM9 design has 1 channel. The basic control signals are the same for the channel base, channel status, FIFO and DMA interfaces across multiple designs.

Notes:
The offsets shown are relative to the channel base address not the card base address.

LM9_CHAN_CNTRL

[0x0] Channel Control Register (read/write)

Channel Control Register

Data Bit	Description
31-7	spare
12	*reserved this design* FIFO External Reset
11-9	Spare
8	OutUrgent
7	InUrgent
6	Read DMA Interrupt Enable
5	Write DMA Interrupt Enable
4	Force Interrupt
3	Channel Interrupt Enable
2	Bypass
1	Receive FIFO Reset
0	Transmit FIFO Reset

Figure 8 PcieBiSerialDb37Lm9 channel Control Register

<ins>FIFO Transmitter/Receiver Reset</ins>: When set to a one, the transmit and/or receive FIFOs will be reset. When these bits are zero, normal FIFO operation is enabled. In addition the Transmit and Receive State Machines are also reset.

<ins>Write/Read DMA Interrupt Enable</ins>: These two bits, when set to one, enable the interrupts for DMA writes and reads respectively.

<ins>Channel Interrupt Enable</ins>: When this bit is set to a one, all enabled interrupts (except the DMA interrupts) will be gated through to the PCI interface level of the design; when this bit is a zero, the interrupts can be used for status without interrupting the host. The channel interrupt enable is for the channel level interrupt sources only.

Force Interrupt: When this bit is set to a one, a system interrupt will occur provided the Channel Interrupt enable is set. This is useful for interrupt testing.

InUrgent / OutUrgent when set causes the DMA request to have higher priority under certain circumstances. Basically when the TX FIFO is almost empty and InUrgent is set the TX DMA will have higher priority than it would otherwise get. Similarly if the RX FIFO is almost full and OutUrgent is set the read DMA will have higher priority. The purpose is to allow software some control over how DMA requests are processed and to allow for a higher rate channel to have a higher priority over other lower rate channels.

ByPass when set allows the FIFO to be used in a loop-back mode internal to the device. A separate state-machine is enabled when ByPass is set and the TX and RX are not enabled. The state-machine checks the TX and RX FIFO's and when not empty on the TX side and not Full on the RX side moves data between them. Writing to the TX FIFO allows reading back from the RX side. An example of this is included in the Driver reference software.

FIFO External Reset: When cleared to a zero, the External FIFOs will be reset. When set the External FIFO is enabled. Please note that the state of the Load pin in the transmit control register affects how the part comes out of reset – what the default Almost Full and Almost Empty offsets are.

This function is reserved since the LM9 design does not use external FIFO's.

LM9_CHAN_STATUS

[0x4] Channel Status Read/Clear Latch Write Port

Channel Status Register

Data Bit	Description
31	Interrupt Status
30	LocalInt
29-28	Spare
27	RxPacketFull
26	RxPacketMt
25	TxPacketFull
24	TxPacketMt
23	BurstInIdle
22	BurstOutIdle
21	TxIdleState
20	RxIdleState
19	TxFifoUnFlLat
18	RxFifoOvFlLat
17	RxPacketCompletedLat
16	TxPacketCompletedLat
15	Read DMA Interrupt Occurred
14	Write DMA Interrupt Occurred
13	Read DMA Error Occurred
12	Write DMA Error Occurred
11	RxAFLvlIntLat
10	TxAELvlIntLat
9	RxParityErrorLat
8	spare
7	spare
6	Rx FIFO Full
5	Rx FIFO Almost Full
4	Rx FIFO Empty
3	Spare
2	Tx FIFO Full
1	Tx FIFO Almost Empty
0	Tx FIFO Empty

Figure 9 PcieBiSerialDb37Lm9 Channel STATUS PORT

LM9 FIFO: Two 4K x 32 FIFO's are used to create the internal Tx and Rx memory. The status for the Tx FIFO and Rx FIFO refer to these FIFO's. The status is active high. 0x13 would correspond to empty Rx and empty Tx internal FIFO's.

Please note with the Rx side status; the status reflects the state of the FIFO and does not take the 4 deep pipeline into account. For example the FIFO may be empty and there may be valid data within the pipeline. The data count with the combined FIFO and pipeline value and can also be used for read size control. [see later in register descriptions]

In addition there are two Packet FIFO's to store the Packet Lengths. The empty and full status for each FIFO is provided. When neither is set, there is data in the FIFO and room for more.

Rx FIFO Empty: When a one is read, the FIFO contains no data; when a zero is read, there is at least one data word in the FIFO.

Rx FIFO Almost Full: When a one is read, the number of data words in the data FIFO is greater than the value written to the corresponding RX_AFL_LVL register; when a zero is read, the FIFO level is less than that value.

Rx FIFO Full: When a one is read, the receive data FIFO is full; when a zero is read, there is room for at least one more data-word in the FIFO. If the FIFO is full when time to write received data to the FIFO an overflow error is declared.

Tx FIFO Empty: When a one is read, the FIFO contains no data; when a zero is read, there is at least one data word in the FIFO. If the FIFO is empty when time to read transmitted data from the FIFO an underflow error is declared.

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

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

RxFifoOvFlLat: When a one is read, an error has been detected. This will occur if FIFO is full when the loader function tries to write to it. A zero indicates that no error has occurred. This bit is latched and can be cleared by writing back to the Status register with a one in the appropriate bit position.

RxParityErrorLat: When a one is read, an error has been detected. This will occur if incorrect parity is received. The exact location of the error is not known. It is recommended that the error bits are cleared per packet to allow the system to retransmit the packet with the error. This bit is latched and can be cleared by writing back to the Status register with a one in the appropriate bit position.

TxFifoUnFlLat: When a one is read, an error has been detected. This will occur if FIFO is empty when the state machine tries to read from it. A zero indicates that no error has occurred. This bit is latched and can be cleared by writing back to the Status register with a one in the appropriate bit position.

Write/Read DMA Error Occurred: When a one is read, a write or read DMA error has been detected. This will occur if there is a target or master abort or if the direction bit in the next pointer of one of the chaining descriptors is incorrect. A zero indicates that no write or read DMA error has occurred. These bits are latched and can be cleared by writing back to the Status register with a one in the appropriate bit position.

Write/Read DMA Interrupt Occurred: When a one is read, a write/read DMA interrupt is latched. This indicates that the scatter-gather list for the current write or read DMA has completed, but the associated interrupt has yet to be processed. A zero indicates that no write or read DMA interrupt is pending.

Tx IDLE is set when the state-machine is in the idle state. When lower clock rates are used it may take a while to clean-up and return to the idle state. If SW has cleared the start bit to terminate the data transfer; SW can use the IDLE bit to determine when the HW has completed its task and returned.

Rx IDLE is set when the state-machine is in the idle state. When lower clock rates are used it may take a while to clean-up and return to the idle state. If SW has cleared the start bit to terminate the transfer; SW can use the IDLE bit to determine when the HW has completed its task and returned.

BO and BI Idle are Burst Out and Burst In IDLE state status for the Receive and Transmit DMA actions. The bits will be 1 when in the IDLE state and 0 when processing a DMA. A new DMA should not be launched until the State machine is back in the IDLE state. Please note that the direction implied in the name has to do with the DMA direction – Burst data into the card for Transmit and burst data out of the card for Receive.

Local Interrupt is the masked combined interrupt status for the channel not including DMA. The status is before the master interrupt enable for the channel.

Interrupt Status is the combined Local Interrupt with DMA and the master interrupt enable. If this bit is set this channel has a pending interrupt request.

Rx Packet Empty: When a one is read, the FIFO contains no Packet Definitions; when a zero is read, there is at least one Packet definition in the FIFO.

Rx Packet Full: When a one is read, the receive Packet FIFO is full; when a zero is read, there is room for at least one more Packet Definition in the FIFO.

Tx Packet Empty: When a one is read, the FIFO contains no Packet Definitions; when a zero is read, there is at least one Packet definition in the FIFO.

Tx Packet Full: When a one is read, the transmit Packet FIFO is full; when a zero is read, there is room for at least one more Packet Definition in the FIFO.

Rx Packet Completed Lat: When a one is read, the receiver has processed and stored at least one packet. The packet FIFO can be read to retrieve the size of the stored packet. The signal is latched and can be cleared via write back with this bit set.

Tx Packet Completed Lat: When a one is read, the transmitter has transmitted at least one of the stored packets. The signal is latched and can be cleared via write back with this bit set.

RxAFLvlIntLat: When set the Rx Data FIFO has become almost Full based on the programmed count. The software can do a looped read or use DMA to retrieve the programmed count amount of data from the storage FIFO. The signal is latched and can be cleared via write back with this bit set. The signal can be used to generate an interrupt if desired.

TxAELvlIntLat: When set the Tx Data FIFO has become almost Empty based on the programmed count. The software can do a looped write or use DMA to load the programmed count amount of data to the storage FIFO. The signal is latched and can be cleared via write back with this bit set. The signal can be used to generate an interrupt if desired.

LM9_CHAN_WR_DMA_PNTR
[0x8] Write DMA Pointer (write only)

BurstIn DMA Pointer Address Register

Data Bit	Description
31-2	First Chaining Descriptor Physical Address
1	direction [0]
0	end of chain

Figure 10 PcieBiSerialDb37Lm9 Write DMA pointer register

This write-only 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.

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

LM9_CHAN_TX_FIFO_COUNT

[0x8] TX [Target] FIFO data count (read only)

TX FIFO Data Count Port

Data Bit	Description
31-16	Spare
15-0	TX Data Words Stored

Figure 11 PcieBiSerialDb37Lm9 TX FIFO data count Port

This read-only register port reports the number of 32-bit data words in the Transmit FIFO. This design has 4095 locations possible. The FIFO count is padded with '0' to a word boundary.

LM9_CHAN_RD_DMA_PNTR

[0xC] Read DMA Pointer (write only)

BurstIn DMA Pointer Address Register

Data Bit	Description
31-2	First Chaining Descriptor Physical Address
1	direction [1]
0	end of chain

Figure 12 PcieBiSerialDb37Lm9 Read DMA pointer register

This write-only 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.

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.

LM9_CHAN_RX_FIFO_COUNT

[0xC] RX [Master] FIFO data count (read only)

RX FIFO Data Count Port

Data Bit	Description
31-16	Spare
15-0	RX Data Words Stored

Figure 13 PcieBiSerialDb37Lm9 RX FIFO data count Port

This read-only register port reports the number of 32-bit data words in the Receive FIFO plus pipeline. The maximum count is the FIFO size plus 4 = 0x1003. The FIFO count is padded with '0' to a word boundary.

LM9_CHAN_FIFO

[0x10] Write TX/Read RX FIFO Port

RX and TX FIFO Port

Data Bit	Description
31-0	FIFO data word

Figure 14 PcieBiSerialDb37Lm9 RX/TX FIFO Port

This port is used to make single-word accesses to and from the FIFO. Data read from this port will no longer be available for DMA transfers. Writing to the port loads the Tx FIFO, Reading unloads the Rx FIFO.

LM9_CHAN_TX_AMT_LVL

[0x14] Tx almost-empty level (read/write)

Tx Almost-Full Level Register

Data Bit	Description
31-16	Spare
15-0	Tx FIFO Almost-Empty Level

Figure 15 PcieBiSerialDb37Lm9 TX ALMOST EMPTY LEVEL register

This read/write port accesses the almost-empty level register. When the number of data words in the transmit data FIFO is less than than this value, the almost-empty status bit will be set. The register is R/W for 16 bits. The mask is valid for a size matching the depth of the FIFO.

LM9_CHAN_RX_AFL_LVL

[0x18] Rx almost-full (read/write)

Rx Almost-Full Level Register

Data Bit	Description
31-16	Spare
15-0	Rx FIFO Almost-Full Level

Figure 16 PcieBiSerialDb37Lm9 RX ALMOST FULL LEVEL register

This read/write port accesses the almost-full level register. When the number of data words in the receive data FIFO is equal or greater than this value, the almost-full status bit will be set. The register is R/W for 16 bits. The mask is valid for a size matching the depth of the FIFO. The level includes the pipeline for an additional 4 locations.

LM9_CHAN_TX

[0x1C] Channel Transmit Control Register (read/write)

Channel TX Control Register

Data Bit	Description
27-16	ClkDiv
15	TxMarkingBit
14	TxStartBit
13	TxClockSrc
12	TxClockDir
9	TxParitySel
8	TxRegPacket
7	TxClkPolarity
6	TxDataBitOrder
5	TxDataByteOrder
4	TxUnderFlowEn
3	TxAEIntEn
2	TxIntEn
1	spare
0	TxEn

Figure 17 PcieBiSerialDb37Lm9 Channel Transmit Control Register

TxEn when set causes the Transmit State Machine to begin operation. When the Transmitter has determined that Data is in the Transmit FIFO, and a packet definition has been read, and CTS is received, data will be transmitted. Clearing TxEn will return the State Machine to the idle state, generally at the end of the current transmitted byte.

Disabling TxEn while transmitting is considered a “panic stop” situation. The FIFO’s should be cleared and the process restarted from scratch when this is done. Restarting without clearing will allow the data to be completed but with the incorrect remaining packet count likely leading to error(s) being detected.

TxIntEn when set enables the transmitter to generate an interrupt for each packet sent. When not enabled the status is still available to allow for polling situations. TxPacketCompleted is the corresponding status bit.

TxAEIntEn when set enables the interrupt based on the TX FIFO Almost Empty flag. When the interrupt occurs a programmable amount of data can be stored into the FIFO making for an efficient DMA or burst of writes to load the FIFO.

TxUnderFlowEn when set allows the UnderFlow error status to cause an interrupt to the host. When cleared the UnderFlow status is still available for polling and will not cause an interrupt.

TxDataByteOrder when cleared retrieves the first byte received from the D7-0 lane in the FIFO and PCI bus. D15-8 is second and D31-24 last. If the control bit is set the order is reversed causing D31-24 to be used for the first byte transmitted and D7-0 for the 4th byte sent per longword. Cleared is standard in PC systems. Set may be required for BigEndian systems.

TxDataBitOrder when cleared transmits the data D0-D7 from each byte as read from FIFO based on the TxDataByteOrder setting. With both set to ‘0’ the data will transmit D0 first D31 last. With the TxDataBitOrder set the data is transmitted D7-D0. With both order bits set the data will be sent D31 first D0 last.

Please note the start bit, parity bit, and stop/marking states are not affected.

TxClkPolarity When set inverts the clock used by the transmitter. The default ‘0’ setting uses the rising edge to change and the falling edge stable condition. When set the data changes on the falling edge and is stable on the rising edge.

TxRegPacket when set causes the transmitter to use the packet definition in the Packet Register instead of the Packet FIFO. Use this setting with consistent packet lengths – for example in command and control situations.

TxParitySel is used to control whether odd or even parity is added to each byte transmitted. 0 = even, 1 = odd.

TxClockDir is used to allow the tranmitter to generate the SendTiming signal instead of receiving it. 1 = generate SendTiming 0 = use SendTiming. The Arc-210 provides the clock in normal situations. For loop-back testing or interconnection with alternative HW the clock can be enabled. PLLA is the clock in use.

TxClockSrc is used to select the PLLA or a divided version of PLLA. Set to ‘0’ for use with PLLA as programmed. Set to ‘1’ to use with secondary divided PLLA clock signal. Set to ‘0’ for normal operation.

TxStartBit defines the sense of the Start Bit. When set the start bit will be active high. When cleared the start bit will be active low.

TxMarkingBit defines the sense of the Stop Bits and Marking State. When cleared the bits following the parity bit leading up to the start bit will be ‘0’.

Please note that TxStartBit and TxMarkingBit should have opposite definitions. The default for the Arc-210 is Start = ‘1’ and Marking = ‘0’.

ClkDiv is used to define the secondary divisor option. The counter uses the programmed value to compare against before changing sense. The counter advances from 0 to the programmed count for N+1. The signal changes sense 2x per period for a factor of 2. The division is therefore 2*(n+1).

Until more channels are added to this design the ClkDiv and TxClkSrc should be treated as reserved. Extremely low frequencies below the PLL capability would be the exception. In this case a 1 MHz or similar frequency can be programmed into the PLL and the divisor used to reach lower frequencies.

LM9_CHAN_TX_PACKET_LEN_FIFO

[0x20] TX Packet Size FIFO

TX Data Count Port

Data Bit	Description
31-0	TX Data Bytes per Packet

Figure 18 PcieBiSerialDb37Lm9 TX Packet Size FIFO

The TX Packet FIFO can store multiple Packet size definitions. Packet Data can be transferred via DMA from host memory to the TX Data FIFO and the corresponding definitions stored into this FIFO. 2K x 32 is the FIFO size.

Larger DMA transfers can be used for grouped packets and individual packets sent if the data needs to be parsed. See status register for information on underflow. Please see the REG version if the packet size is consistent.

This port is read-write. Reading back will remove the data from the FIFO. The read path is provided for self test purposes.

LM9_CHAN_TX_PACKET_LEN_REG

[0x24] TX Packet Size Reg

TX Data Count Port

Data Bit	Description
31-0	TX Data Bytes per Packet

Figure 19 PcieBiSerialDb37Lm9 TX Packet Size Register

This read-write register port holds the number of bytes to transmit per packet. Data is repeatedly sent until the TX FIFO is empty. See status register for information on underflow.

LM9_CHAN_RX

[0x34] Channel RX Control Register (read/write)

Channel Control Register

Data Bit	Description
15	RxMarkingBit
14	RxStartBit
13-11	spare
10	RxTimeOutEn
9	RxParitySel
8	spare
7	RxClockPol
6	RxDataBitOrder
5	RxDataByteOrder
4	RxFifoOvFlIntEn
3	RxFifoAFIntEn
2	RxIntEn
1	RxParityErrorIntEn
0	RxEn

Figure 20 PcieBiSerialDb37Lm9 Channel Rx Control Register

RxEn when set causes the Rx State Machine to begin operation. The ReceiveData line is tested and when the marking state is found [10+ marking bits] the state-machine will look for the start bit. When the start bit detected, bits are counted and the byte is stored. Once synchronized the receiver requires 1 stop bit between words.

RxIntEn when set enables the RxPacketCompleted status to cause an interrupt to the host. The status is set when a packet has been received and stored and the length FIFO updated.

RxFifoAFIntEn when set enables the interrupt based on the Rx FIFO Almost Full flag. When the interrupt occurs a programmable amount of data can be read from the FIFO making for an efficient DMA read or burst of reads to unload the FIFO.

RxFifoOvFlIEn when set enables the interrupt based on the Rx FIFO OverFlow condition. When the State-machine writes to the FIFO the status is tested. If the FIFO is full when time to write the OverFlow status is set. If the interrupt is enabled the status is gated out to drive the interrupt request to the host.

RxParityErrorIntEn when set enables the interrupt based on the Rx Parity Error condition. When the State-machine moves data to the secondary holding latches, parity is generated based on the “Odd/Even” selection and tested against the stored [received] value. If the parity generated does not match the parity received an error is declared. If the interrupt is enabled the status is gated out to drive the interrupt request to the host.

RxDataByteOrder when cleared loads the first byte received to the D7-0 lane in the FIFO. D15-8 is second and D31-24 last. If the control bit is set the order is reversed causing D31-24 to be loaded for the first byte received and D7-0 for the 4th byte received per longword. Cleared is standard in PC systems. Set may be required for BigEndian systems.

RxDataBitOrder when cleared treats the received data as D0-D7. With the RxDataBitOrder set the data is received and then loaded into the FIFO as D7-D0. With both order bits set the data will be received D31 first D0 last.

Please note the start bit, parity bit, and stop/marking states are not affected.

RxClkPolarity When set inverts the clock used by the received. The default ‘0’ setting uses the falling edge to sample the received data. When set the rising edge is used.

RxParitySel is used to control whether odd or even parity is tested against the parity received on each byte. 0 = even, 1 = odd.

RxTimeOutEn when set enables the HW to terminate a received Packet based on the stored TimeOut count. At the end of each byte the clocks are counted to determine if the time before the next byte is long enough to declare the end of the packet. The first byte is not counted against the time. The clock is the PCI clock usually 33 MHz. If using this method the time should be set to something longer than the number of expected marking states between bytes usually transmitted.

When cleared the programmed byte count is used without a timer. For systems with a defined size of packet this is the preferred method of operation.

RxStartBit defines the sense of the Start Bit expected from the tranmitter. When set the start bit will be active high. When cleared the start bit will be active low. This bit needs to match the definition in use by the tranmitter. The standard is ‘1’ for the ARC-210.

RxMarkingBit defines the sense of the Stop Bits and Marking State. When cleared the bits following the parity bit leading up to the start bit will be ‘0’. This is the expected state of the line. This must be programmed to match the transmitter for proper reception to occur.

Please note that RxStartBit and RxMarkingBit should have opposite definitions. The default for the Arc-210 is Start = ‘1’ and Marking = ‘0’.

LM9_CHAN_RX_PACKET_LEN_FIFO

[0x38] RX Packet size

RX Packet Data Count Port	
Data Bit	Description
31-0	RX Data Bytes per Packet Received

Figure 21 PcieBiSerialDb37Lm9 RX Packet Size

This read only port holds the packet size definitions. Each word loaded into the FIFO represents the size of a Packet loaded into the data FIFO. Please note that this is a FIFO. The Packet data read will match the Packet Definitions – same order. The data, once read by the system is no longer available to be read from the FIFO.

LM9_CHAN_RX_TIMEOUT_LEN

[0x3C] RX TimeOut Length

RX TimeOut Port	
Data Bit	Description
31-0	Max Clocks before new Packet

Figure 22 PcieBiSerialDb37Lm9 RX TimeOut Length

This read-write only port holds the packet timeout length definition. If the gap between received bytes is longer than the TimeOut length * 30nS the end of the packet will be declared [in HW]. When the end of the packet is declared the HW will load the last data into the FIFO [if needed and resume looking for more data. The packet FIFO will be updated with the size of the packet received.

The clock used is the PCI clock.

If the Transmitter is running at 1 MHz and has a typical 2 Marking states between bytes sent then there are 3 uS [Stop, marking, marking]. If there is a defined gap between packets for a packet break that time can be used. If there is no definition then the observed termination sequence [3 clocks] plus some margin can be used to set the time out. Remember to scale for the period.

5 us ~ 165 clocks at 33 MHz.

The counter counts from 0 leading to an N+1 TimeOut length. The count is checked with a Greater Than function effectively making it an N based count. Having said all of that you will likely have to "play" with this variable to achieve the system performance you need with a non packet defined interface. The shorter the time the quicker the response and the more likely a single packet is broken into two or more pieces.

LM9_CHAN_RX_BYTECOUNT_LEN

[0x40] RX ByteCount Length

RX ByteCount Port

Data Bit	Description
31-0	Expected Bytes per Packet

Figure 23 PcieBiSerialDb37Lm9 RX ByteCount Length

The RxByteCount Length is used to specify the number of bytes in a packet.

The length can be used in several ways. The length can be used to match the known packet size expected from the receiver. The length can be used to create smaller SW defined packets with a received amorphous stream.

A larger DMA thread can be defined and smaller packet interrupts based on the ByteCount allowing the SW to "chase" the reception with known step-sizes.

For example set-up a multi-megabyte DMA transfer to system memory. The DMA will operate based on the FIFO state and have moved the Packet to memory before the RxPacketCompleted interrupt is processed. The DMA will have only one interrupt per large transfer. The packet interrupt will allow the host software to trail the DMA process operating on the data stored from the Packets received. The Packets are a consistent size making for an efficient retrieval of data from system memory to application.

Alternatively the data can be moved directly to disk using system utilities and the known data stored into the FIFO. In this case the DMA operation would be programmed to be called each time the Packet data arrived.

Loop-back

The Engineering kit includes reference software, utilizing external loop-back tests.

The test set-up included PcieBiSerialDb37Lm9 and loop-back plug. The Pin numbers are for the interconnections on the Loop-back plug. The IO names can be used to accommodate a different set-up. The loop-back plug is a DB37 connector with the interconnections protected with a connector shell.

Signal	From	To	Signal
GPIO_0+	7	13	GPIO_6+
GPIO_0-	8	14	GPIO_6-
GPIO_1+	26	32	GPIO_7+
GPIO_1-	27	33	GPIO_7-
GPIO_2+	9	15	GPIO_8+
GPIO_2-	10	16	GPIO_8-
GPIO_3+	28	34	GPIO_9+
GPIO_3-	29	35	GPIO_9-
GPIO_4+	11	17	GPIO_10+
GPIO_4-	12	18	GPIO_10-
GPIO_5+	30	36	GPIO_11+
GPIO_5-	31	37	GPIO_11-
RTS+	1	20	CTS+
RST-	2	21	CTS-
SENDDATA+	3	22	RXDATA+
SENDDATA-	4	23	RXDATA-
SENDTIMING+	5	24	RXTIMING+
SENDTIMING-	6	25	RXTIMING-

PCIe Module Front Panel IO Interface Pin Assignment

The figure below gives the pin assignments for the IO Interface on the PcieBiSerialDb37Lm9.

IO_0p (RTS+)	IO_0m (RTS-)	1	2
IO_1p (CTS+)	IO_1m (CTS-)	20	21
IO_2p (SENDDATA+)	IO_2m (SENDDATA-)	3	4
IO_3p (RXDATA+)	IO_3m (RXDATA-)	22	23
IO_4p (SENDTIMING+)	IO_4m (SENDTIMING-)	5	6
IO_5p (RXTIMING+)	IO_5m (RXTIMING-)	24	25
IO_6p (GPIO_0+)	IO_6m (GPIO_0-)	7	8
IO_7p (GPIO_1+)	IO_7m (GPIO_1-)	26	27
IO_8p (GPIO_2+)	IO_8m (GPIO_2-)	9	10
IO_9p (GPIO_3+)	IO_9m (GPIO_3-)	27	29
IO_10p (GPIO_4+)	IO_10m (GPIO_4-)	11	12
IO_11p (GPIO_5+)	IO_11m (GPIO_5-)	30	31
IO_12p (GPIO_6+)	IO_12m (GPIO_6-)	13	14
IO_13p (GPIO_7+)	IO_13m (GPIO_7-)	32	33
IO_14p (GPIO_8+)	IO_14m (GPIO_8-)	15	16
IO_15p (GPIO_9+)	IO_15m (GPIO_9-)	34	35
IO_16p (GPIO_10+)	IO_16m (GPIO_10-)	17	18
IO_17p (GPIO_11+)	IO_17m (GPIO_11-)	36	37
GND*		19	

Figure 24 PcieBiSerialDb37Lm9 FRONT PANEL Interface

GND is shunt selectable for DC, AC and open configurations. Jumper is located near lower edge of DB37 connector. Options labeled in silk-screen.

Applications Guide

Interfacing

The pin-out tables are displayed with the pins in the same relative order as the actual connectors. 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. Differential interface devices provide some immunity from, and allow operation when part of the circuit is powered on and part is not. It is better to avoid the issue of going past the safe operating areas by powering the equipment together and by having a good ground reference.

Keep cables short. Flat cables, even with alternate ground lines, are not suitable for long distances. In addition series resistors are used and can be specified to be something other than the 0 ohm standard value. The connector is pinned out for a standard DB37 cable to be used. It is suggested that this standard cable be used for most of the cable run or an equivalent with proper twisted pairs and shielding.

Coming Soon. Terminal Block. We offer a high quality 37 screw terminal block that directly connects to the DB37. The terminal block can mount on standard DIN rails. DBterm37 has an associated twisted pair cable compatible with the PCIeBiSerialDB37. [http://www.dyneng.com/DBterm68.html]

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 particular device's rated voltages.

Construction and Reliability

PCIe Modules were conceived and engineered for rugged industrial environments. The PcieBiSerialDb37Lm9 is constructed out of 0.062 inch thick high temperature ROHS compliant material.

The traces are matched length from the FPGA ball to the IO pin. The analog switches and termination resistors are located directly under the transceivers and connected with “zero stub” routing to eliminate unwanted effects from unused options.

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 PCIe 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 PCIeBiSerialDb37 has a wider keep out than required by PCIe specification 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 PcieBiSerialDb37Lm9 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:	PCIe 1-4 lanes. 4 lanes recommended
Digital Parallel IO:	RS485 IO [LVDS option]
Digital Serial IO:	ARC-210 serial interface with RTS/CTS SENDDATA/RECEIVEDATA, SENDTIMING/RECEIVETIMING
DIP Switch:	DipSwitch supplied for board identification and other user purposes.
CLK rates supported:	PLLA is programmed to select Transmit Clock rate. For loop-back and alternate HW implementations. PLLB,C,D reserved for new applications.
Software Interface:	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
Access Time:	Frame to TRDY 121 nS [4 PCI clocks] or burst mode DMA – 1 word per PCI clock transferred.
Interrupt:	Each port has independently programmable interrupt sources, DMA interrupts included.
Onboard Options:	All Options are Software Programmable
Interface Options:	37 Pin DB connector at front bezel.
Dimensions:	Standard 1/2 length PCIe module.
Construction:	Multi-Layer Printed Circuit, Through Hole and Surface Mount Components.
Power:	+12 and +3.3 used from PCIe interface. No secondary power supply connections required. 1.2, 2.5 and 5V developed locally.
Weight:	TBD oz

Order Information

standard temperature range Industrial
PcieBiSerialDb37Lm9 PMC Module with 22 IO channels. One Transmitter and one Receiver port implemented.
http://www.dyneng.com/pciebiserialdb37.html

Order Options:

Pick any combination to go with IO
-CC to add conformal coating
-ET to change to industrial Temp [-40 - +85C] Standard this design
-COM to change to commercial temp parts [0-70]

Dbterm37: 37 position terminal block with DB37 connector.
http://www.dyneng.com/DBterm37.html

Dbcabl37: DB37 cable compatible with PCIeBiSerialDB37. Twisted pairs on correct pin pairs. **http://www.dyneng.com/DBcabl37.html**

PCIe BiSerial DB37 LM9 Eng Kit : Windows or Linux Driver software, Loop-Back Plug, reference schematics. Recommended for first time purchases.
http://www.dyneng.com/pciebiserialdb37.html