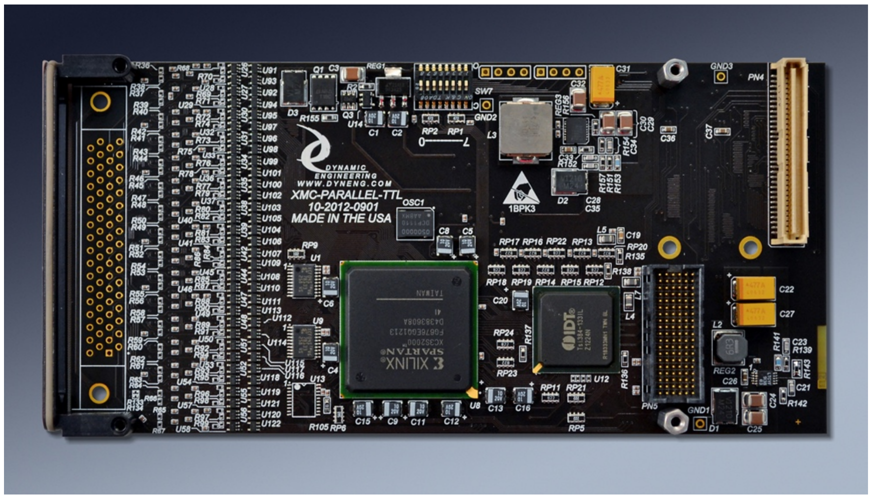
DYNAMIC ENGINEERING
150 DuBois St. Suite C, Santa Cruz, Ca 95060
831-457-8891   https://www.dyneng.com
sales@dyneng.com
Est. 1988

User Manual

XMC-PARALLEL-TTL-BA16

Digital Parallel Interface
XMC Module
2 Byte wide RX/TX ports with programmable rate



Rev01 shown

Revision 03P1
Corresponding Hardware: Revision 2
10-2012-0902
FLASH 0302

XMC-PARALLEL-TTL-BA16

Digital Parallel Interface
XMC Module
Dynamic Engineering
150 DuBois St. Suite C, Santa Cruz CA 95060
831-457-8891

Table of Contents

List of Figures

Product Description

In embedded systems many of the interconnections are made with single ended TTL or CMOS level signals. Depending on the system architecture an IP, PMC, or XMC will be the right choice to make the connection. With most architectures you have a choice as there are carriers for PCIe, PCI, cPCI, VPX, VME, PC/104p and other buses for XMC, PMC, and IP mezzanine modules.

Usually the choice is based on other system constraints as XMC, PMC, and IP can provide the IO you require. Dynamic Engineering would be happy to assist in your decision regarding architecture and other trade-offs with the XMC/ PMC / IP decision. Dynamic Engineering has carriers for XMC, IP, and PMC modules for most architectures, and is adding more as new solutions are requested and required by our customers.

The XMC compatible XMC-Parallel-TTL has 64 independent digital IO. The high density makes efficient use of XMC slot resources. The IO is available for system connection through the front panel, via the rear [Pn4] connector, or both. A high density 68 pin VHDCI front panel connector provides the front panel IO. The IO lines can be protected with optional transorbs. The rear panel IO has a PIM and PIM Carrier available for rear panel wiring options in some systems.

With the revision 02 PCB a larger industrial temperature FPGA is standard along with temperature sensor, 50 MHz PCI operation, and on-board FLASH reprogramming. See SW package for examples of how to use the new features.

XMC-Parallel-TTL-BA16 is a customerized version of the standard XMC-Parallel-TTL board. “BA16” is set to 3.3V, has front panel IO, and a specific state-machine implementation for two channels [two TX and two RX]. The transmit rate is programmable. The RX rate is expected to be 1 MHz from the customer equipment. The design has enough margin to support rates up to and beyond 8 MHz.

TX and RX FIFOs are 8Kx32 per port. This is an enhancement with the 0302 FLASH.

BA16 features are selectable. The standard register based and COS functions are available on unused [by BA16] pins and can be swapped with the BA16 functions under software control.

The HDEterm68 http://www.dyneng.com/HDEterm68.html
can be used as a breakout for the front or rear panel IO. The HDEcabl68 provides a convenient cable. http://www.dyneng.com/HDEcabl68.html VHDCI to SCSI adapter cables are available. A VHDCI based “Term68” is planned. Custom cables can be manufactured to your requirements.   Please contact Dynamic Engineering with your specifications.

Each channel is programmable to be input or output on a channel-by-channel basis. All 64 IO channels can be used as interrupt generators. Interrupts are programmable to be based on rising, falling and change of state [both] conditions. The interrupts are maskable to allow polled operation as well.

The inputs are available unfiltered and after the transition detection. The transition detection is programmable for clock rate. The local 50 MHz oscillator, PCI or external clocks can be selected as the reference to the clock divider. The clock divider is programmable to use the reference rate or to divide it to a lower frequency.

All of the IO are routed through the FPGA to allow for custom applications that require hardware intervention or specific timing- for example an automatic address or data strobe to be generated. The initial model was register based [FLASH 0101]. The design with revision 2 and later FLASH is DMA capable with a built in programmable parallel data output and input function. The new features are designed to default to "not used" to allow the new cards to be used with older customer software. The DMA function can be used with customer requirements too. Please contact Dynamic Engineering with your custom requirements.

The IO are driven with open-drain high current drivers. When enabled, the high side is driven with the device and augmented with pull-up resistors. When disabled the output is pulled high with the resistors unless another device on the line is driving that line low.

The driver can source & sink 24+ mA. The pull-up default value is for 470 ohms. The resistors are referenced to either 5V or 3.3V based on software selection. For custom models with alternate values please contact Dynamic Engineering.

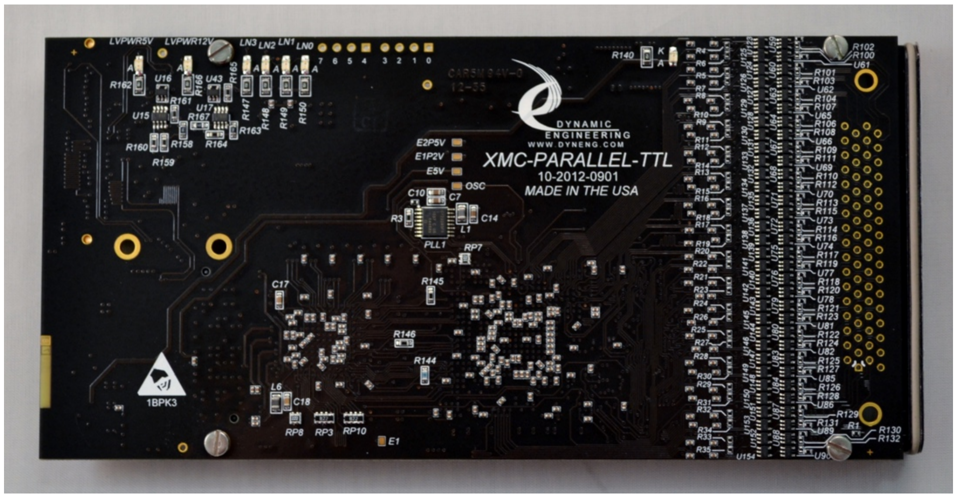


Figure 1 **XMC-PARALLEL-TTL REAR VIEW**

Each line driver is a separate Single Bus Buffer Gate – LVC125. By using separate gates all lines can be enabled or disabled without overheating a package with multiple drivers. The separated gates also allow for individual control for the enable signal.

Each signal also has a separate LVC17 receiver with hysteresis. Internal loop-back is supported and can be used for BIT purposes. Please note: external connections can affect the value read-back when performing internal loop-back.

The registers are mapped as 32 bit words. All registers are read-writeable. The Linux and Windows® compatible drivers are available to provide the system level interface for this design. 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. The Linux documentation is provided in-line with the source code.

The basic functions of parallel IO and COS capture are designed into the “base” model. Additional features will be added to the base model by using a mux on the output side to allow software to select the base or extended features. Data bit 0 was the first extended feature and is a programmable output for the COS reference clock. With software the output definition can be changed to drive the COS clock onto Data 0. The user can use a scope to check that their set-up is what they want it to be, and then likely return it to being a data bit. You can leave the output defined as a clock if desired.

A second update added FIFO’s to support output and input data streams. An additional register is added to allow the user to select on a bit-by-bit basis the programmable data output or the register based output. The input function does not preclude the use of the standard input functions. For example, COS can be run on the same inputs as the data capture uses.

XMC-PARALLEL-TTL is part of the XMC Module family of modular I/O components. XMC-PARALLEL-TTL conforms to the XMC standard. This guarantees compatibility with multiple XMC Carrier boards. Because the XMC may be mounted on different form factors, while maintaining plug and software compatibility, system prototyping may be done on one XMC Carrier board, with final system implementation on a different one.

For example, use PCIe8LXMCX1 to develop your SW in a standard PCIe slot and then port to your target HW.

Theory of Operation

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

The XMC-PARALLEL-TTL features a Xilinx Spartan 6 FPGA, high current LVC drivers, and LVC Schmidt trigger receivers with hysteresis. The design utilizes a bridge to convert between PCIe and PCI. 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 initial implementation provides an enhanced feature set based on the PMC-Parallel-TTL design. Designs can be ported between the PMC and XMC implementations. Additional FLASH updates will provide new features.

The drivers are initialized to the off state and pull-ups on board hold the IO lines in the ‘high’ state. The direction registers are used to program the channel to be a driver or not. The receivers are always enabled allowing local read-back of the transmitted data.

Data written to the IO registers can be placed on the bus. The master enable allows all 64 bits to be synchronized. The master enable can be programmed “on” to allow direct updates if 64 bit synchronization is not required.

For an IO with the direction bit set and master enabled: When a ‘0’ is written to any IO line register position the corresponding line is driven low. When a ‘1’ is written to any IO line register position that line is driven high by the local driver. The 470 resistor to 3.3/5 will provide additional “source current”, and level control when in “open drain” mode [programmed for receive].

If the direction bit is set to input, the level will be controlled by external devices and the attached pull-ups. The control register is read-writeable. The data register read corresponds to the IO side. The register read-back is at an alternate address offset. The register read-back is independent of the bus; the data read will always match the data written. The IO data read will reflect the state of the bus and not necessarily the state of the on-board drivers.

The read-back registers are clocked at a programmable rate with an internal clock generator. If desired the internal clock can be replaced with an external source and an enable. The basic option is available under SW control. If special programming is needed please contact Dynamic Engineering for a custom FPGA implementation.

All the IO control and registers are instantiated within the FPGA, only the drivers and receivers are separate devices. If desired, the IO lines can be specially programmed to create custom timing pulses etc. For example, if the interface is to put out an address and then an address qualifier to strobe the address into the receiving hardware one of the IO lines can be programmed to create a pulse after the address for the IO registers is written to. The custom pulse will be more accurate for delay and duration than a SW timing solution. The number of accesses to the card can be reduced as well having the effect of greater through-put. Please contact Dynamic Engineering with your requirements.

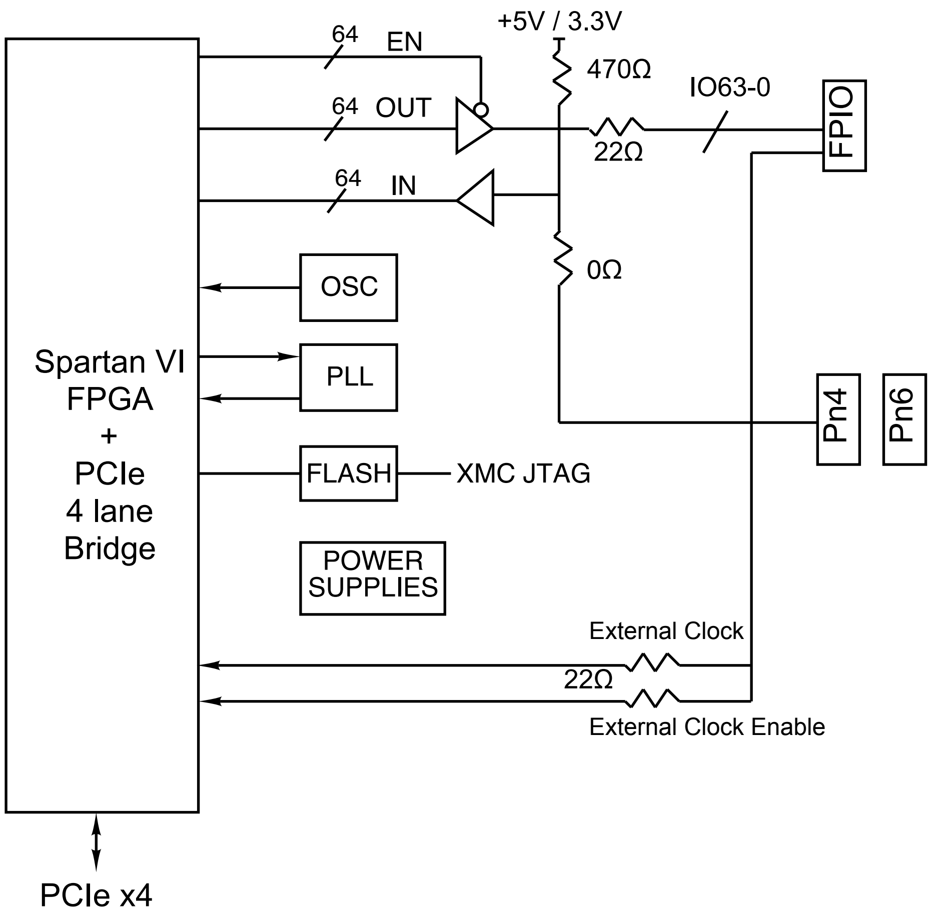


Figure 2 XMC-PARALLEL-TTL Block Diagram

XMC-Parallel-TTL-BA16 features a programmable data path with DMA support. The internal block RAM is configured to provide FIFOs for transmit and receive [8K x 32]. The PLL is used as a TX state-machine reference.   The FIFOs support a loop-back path between them for BIT [built in testing]. The IO is also selectable at the bit level to mask down to the user requirement. 8 bits can be masked down to 5 for example. A reference clock and data strobe are provided for synchronization. Data is valid on the rising edge of the clock. The clock is stopped when data is not being transmitted.

The hardware will pull data from the host memory and store into the transmit FIFO. The FIFO will be kept full with DMA operating at the PCI bus frequency. The output side will operate at the PLL programmed rate [1 MHz for example] and at the selected width [8 bits for example]. The state-machine will take care of reading from the FIFO, converting down or up in width, and outputting on the programmed frequency.

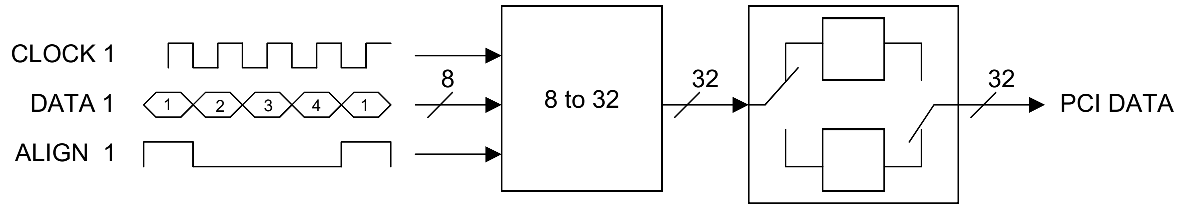
The state-machine supports “run until done” and “pause mode”. In Run until Done mode the transmission will continue until the FIFO is emptied. With DMA and lower frequencies of transmission and/or coupled with more narrow data widths this mode can work well. At our 1 MHz example, and byte wide data we have 50x on clock rate and 4x on width for a really large multiplier. In addition, the DMA FIFO is 8Kx32 for TX leaving a lot of “rubber band” in the memory chain to support the transmission. As the frequencies are increased and/or the width is increased the multiplier can be reduced to the point where the FIFO may go empty on occasion before the transmission is complete. OS delays are the main culprit. The Pause mode allows the hardware to “keep running” even when the FIFO goes empty. The state-machine interprets the empty FIFO as a pause not a stop. The state-machine cleans up and stops the reference clock along with the data and then restarts when data becomes available.

For longer runs or at smaller multiplier situations it is recommended to use the pause mode. At the end of the run when in pause mode the hardware will be idling waiting for more data to send. A new transmission can be started by loading the FIFO with the next data set or by disabling the transmitter and reinitializing. If the frequency is going to change it is recommended to reinitialize.

The DMA 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 slightly more efficient as there are potentially fewer initialization reads and less 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.

For reception the hardware samples the received clock and looks for the rising edge. The reference clock used can be the oscillator [50 MHz] or the PLL channel B. 6X oversampling is recommended as a minimum leading to an 8.3 MHz upper frequency limitation on the RX side with the oscillator or user defined if the PLL is used. The oscillator setting is recommended for most applications.

When the rising edge is found the Align bit is checked, and if valid, a reception started. 4 bytes are captured to build up a LW. At the 5th byte received the align signal is checked, and if not present an error flagged. The process repeats until SW stops reception. The LW data is moved to the FIFO as it becomes available. If the FIFO is full when it is time to write, an error is flagged. The DMA engine moves the data from the receive FIFO to the host memory.



The RX and TX DMA channels are separate and can run at the same time without software intervention. Full DMA engines are provided on each channel with internal arbitration between the channels to access the PCI bus to retrieve data or write data.

The clock rates for RX and TX are also independent using separate PLL frequencies or the external rate.

Address Map

```
Function                                Offset
// XMC Parallel TTL definitions
#define XmcParTtl _BASE          0x0000 // 0  XMC Parallel TTL base control register offset
#define XmcParTtl _ID            0x0004 // 1  XMC Parallel TTL ID Register offset
#define XmcParTtl _STATUS        0x0008 // 2  XMC Parallel TTL status Register offset
#define XmcParTtl _DirL          0x000c // 3  XMC Parallel TTL Direction lower Register offset
#define XmcParTtl _DirU          0x0010 // 4  XMC Parallel TTL Direction upper Register offset
#define XmcParTtl _DatL          0x0014 // 5  XMC Parallel TTL Data lower Register, line data read
#define XmcParTtl _DatU          0x0018 // 6  XMC Parallel TTL Data upper Register, line data read
#define XmcParTtl _DatLreg       0x001c // 7  XMC Parallel TTL Data lower Register read-back
#define XmcParTtl _DatUreg       0x0020 // 8  XMC Parallel TTL Data upper Register read-back
#define XmcParTtl _COSclk        0x0024 // 9  XMC Parallel TTL COS Clock definition Register
//#define spare                  0x0028 // 10  XMC Parallel TTL
#define XmcParTtl _RisLreg       0x002c // 11  XMC Parallel TTL Rising lower Register
#define XmcParTtl _RisUreg       0x0030 // 12  XMC Parallel TTL Rising upper Register
#define XmcParTtl _FallLreg      0x0034 // 13  XMC Parallel TTL Falling lower Register
#define XmcParTtl _FallUreg      0x0038 // 14  XMC Parallel TTL Falling upper Register
#define XmcParTtl _IntRisLreg    0x003c // 15  XMC Parallel TTL Interrupt Enable Rising lower Register
#define XmcParTtl _IntRisUreg    0x0040 // 16  XMC Parallel TTL Interrupt Enable Rising upper Register
#define XmcParTtl _IntFallLreg   0x0044 // 17  XMC Parallel TTL Interrupt Enable Falling lower Register
#define XmcParTtl _IntFallUreg   0x0048 // 18  XMC Parallel TTL Interrupt Enable Falling upper Register
#define XmcParTtl _IntRisLstat   0x004c // 19  XMC Par TTL Interrupt Rising LWR Stat Rd, write = clear
#define XmcParTtl _IntRisUstat   0x0050 // 20  XMC Par TTL Interrupt Rising UPR Stat Rd,  write = clear
#define XmcParTtl _IntFallLstat  0x0054 // 21  XMC Par TTL Interrupt Falling LWR Stat Rd, write = clear
#define XmcParTtl _IntFallUstat  0x0058 // 22  XMC Par TTL Interrupt Falling UPR Stat Rd, write = clear
#define XmcParTtl _DR_L          0x005C // 23  XMC Par TTL DMA - Register bit selection 31-0 R/W
#define XmcParTtl _DR_U          0x0060 // 24  XMC Par TTL DMA - Register bit selection 63-32 R/W

#define XmcParTtl_Temp           0x006C //27 XMC Parallel TTL Temperature interface

#define XmcParTtl _Chan0Cntl     0x0078 // 30  XMC Par TTL Chan 0 Control Register
#define XmcParTtl _ch0_st        0x007C// 31  XMC Par TTL Chan 0 status, interrupt clear, data count
#define XmcParTtl _ch0_brstin    0x0080// 32  XMC Par TTL channel 0 burst in control,
#define XmcParTtl _ch0_brstout   0x0084// 33  XMC Par TTL channel 0 burst out control,
#define XmcParTtl _ch0_swr       0x0088// 34  XMC Par TTL FIFO single read - RX, single write - TX
#define XmcParTtl _ch0_tx_aecnt 0x008C// 35  XMC Par TTL ch 0 almost empty count register and rd-bk
#define XmcParTtl _ch0_rx_afcnt  0x0090 // 36  XMC Par TTL ch 0 almost full count register and rd-bk
#define XmcParTtl _ch0_tx_ffcnt  0x0094 // 37  XMC Par TTL ch 0 tx fifo word count
#define XmcParTtl _ch0_rx_ffcnt  0x0098 // 38  XMC Par TTL ch 0 rx fifo word count

#define XmcParTtl _Chan1Cntl     0x00A0 // 40  XMC Par TTL Chan 1 Control Register
#define XmcParTtl _ch1_st        0x00A4// 41  XMC Par TTL Chan 1 status, interrupt clear, data count
#define XmcParTtl _ch1_brstin    0x00A8// 42  XMC Par TTL channel 1 burst in control,
#define XmcParTtl _ch1_brstout   0x00AC// 43  XMC Par TTL channel 1 burst out control,
#define XmcParTtl _ch1_swr       0x00B0// 44  XMC Par TTL FIFO single read - RX, single write - TX
#define XmcParTtl _ch1_tx_aecnt 0x00B4// 45  XMC Par TTL ch 1 almost empty count register and rd-bk
#define XmcParTtl _ch1_rx_afcnt  0x00B8 // 46  XMC Par TTL ch 1 almost full count register and rd-bk
#define XmcParTtl _ch1_tx_ffcnt  0x00BC // 47  XMC Par TTL ch 1 tx fifo word count
#define XmcParTtl _ch1_rx_ffcnt  0x00C0 // 48  XMC Par TTL ch 1 rx fifo word count
```

Figure 3 XMC-PARALLEL-TTL Internal Address Map Base Functions

The address map provided is for the local decoding performed within XMC-Parallel-TTL.

The addresses are all offsets from a base address. The upstream device connected to the XMC provides the 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 = 0xDCBA and the CardId = 0x006A for the XMC-Parallel-TTL-BA16.

Programming

Programming the XMC-PARALLEL-TTL-BA16 requires only the ability to read and write data in the host's XMC space.

Once the initialization process has occurred, and the system has assigned addresses to XMC-Parallel-TTL-BA16 card, software will need to determine what the address space is for the PCI interface [BAR0]. The offsets in the address table are relative to the system assigned BAR0 base address.

The next step is to initialize XMC-Parallel-TTL-BA16. If the basic mode of direct read and write operations is to be used then the default settings can be used except for setting the master output enable and the direction bits corresponding to the channels to transmit on.

If COS inputs are to be used the reference and divisor clocks may require programming. In many cases the default settings will work. In addition, the Rising, Falling, and Interrupt capabilities need to be programmed. Once the settings are in place it is recommended that the receive state registers are written to for clearing purposes as the programming steps may cause phantom events to be captured.

If the programmable data path is used the DMA_REG selection will be required plus the initialization of the DMA and channel control registers. The address map is defined with channel 0 and channel 1 definitions using the standard expandable architecture that Dynamic Engineering employs. Additional channels would be added at offsets with similar channel register definitions for ease of programming. One additional programming step will be to initialize the PLL to the user desired frequency.

For Windows™ and Linux systems the Dynamic Driver 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 BA16 specific functions the Channel Control, DMA/Reg and Direction registers 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 BA16 to start the DMA. Please refer to the Burst IN and Burst Out register discussions. The transmitter for the BA16 is a port from the standard data engine featured in the rev 02 FLASH with PMC-Parallel-TTL with the Align32 and Clock outputs added. The mode should be set to byte and the PLL programmed to 1 MHz to get the correct results if transmitting. Only byte wide data will be transmitted. The corresponding bits in the Direction and DMA/Reg control registers will need to be set to select output and DMA controlled function.

DMA should be set-up before starting the receiver. The receiver will automatically adjust to the incoming clock rate within the operational parameters. The data will be captured and extended to 32 bit wide words and then DMA'd to host memory. The first byte captured will be stored at the 7-0 location, the second at 15-8 and so forth.   Data will continue to be captured until the receiver is stopped with software. The 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 our any extra data captured. For on-the-fly processing multiple shorter DMA segments can be programmed, and at the interrupt restart DMA to point at the alternate segment to allow processing on the previous one.

Register Definitions

XmcParTtl_BASE

[$00 Base Control Register Port read/write]

DATA BIT	DESCRIPTION
31-24	Spare
23	TDO – Read Only
22	Spare
21	spare
20	bit 19 read-back of pll_dat register bit
19	pll_dat [write to PLL, read-back from PLL]
18	spare
17	pll_sclk
16	pll_en
15	VIOSEL
14-12	Spare
11	TMS_IP
10	TCK_IP
9	TDI_IP
8	JTAG_MUX_SEL
7-5	spare
4	Master Parallel Data Enable
3	M_TX_EN
2	M_RX_EN
1	Force Interrupt
0	Master Interrupt Enable

Figure 4 XMC-PARALLEL-TTL Control port 0 Bit Map

This is the base control register for the XMC Parallel TTL. 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.

Master Interrupt Enable when ‘1’ gates active interrupt requesting conditions onto Interrupt Request A. When set to ‘0’ the interrupting functions are available as status but no interrupt request is generated by the card to allow for polled operation.

Force Interrupt when ‘1’ and the master enabled will cause an interrupt request. The interrupt can be cleared by clearing this bit or disabling the master interrupt enable or both. Force Interrupt is used for test and software development purposes.

Master Parallel Data Enable is used to allow the upper and lower data to be synchronized. The upper 32 bits and the lower 32 bits are not accessed at the same time. If the user wants to have the upper and lower data change at the same time the Master enable can be cleared to ‘0’, both halves of the data written and then the enable set ‘1’. If synchronization is not an issue; program to ‘1’ as part of initialization.

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^{2}C 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.

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.

JTAG signals are used to reprogram, verify, erase etc. the FLASH used to load the FPGA. JTAG Mux Sel when set causes the JTAG mux to connect the FPGA control to the FLASH device. When ‘0’ the external programming header is selected. TDI, TMS, and CLK are used to control the actions of the state-machine within the FLASH and to transfer data. TDO is used to read the serial data returned from the FLASH. Our drivers provide utilities to program the FLASH. If writing your own please ask for a copy of ours for reference.

VIOSEL is used to set the reference for the TTL IO. Each IO has a pull-up resistor and the resistor can be tied to 3.3V or 5V. When cleared 3.3V is used as the reference. For 5V set this bit.

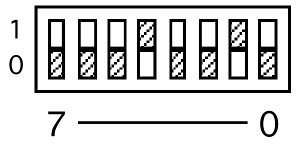
XmcParTtl _ID

[$04 Switch and Revision number port read only]

DATA BIT	DESCRIPTION
31-24	spare
23-16	Revision Major
15-8	Revision Minor
7-0	DIP switch

Figure 5 XMC-PARALLEL-TTL Revision, Switch

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 XMC Parallel TTL is being addressed 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.



RevisionMajor and RevisionMinor are stored and allow SW to determine the feature set of the installed card. Major revisions are updated when large changes occur – new feature etc. Minor Revisions are updated anytime a new FLASH is released. The BA16 Major revision is x03 and the minor revision is x01 currently.

Revision History.
0x0201 initial major release, add BA16 function 8/19/08
0x0202 Update internal loop-back 10/31/08
0x0203 Update Min GNT request for better DMA performance 12/3/09
0x0204 Register Status signals due to asynchronous updates 10/2/12
0x0205 Register Channel Status due to asynchronous updates 12/19/13
0x0301 Update Major and restart Minor revision ⇔ Port to Rev 02 PCB, Add Flash Reprogramming, Add Temperature sensor 3/3/20
0x0302 Increase PCI bus to 50 MHz, Upgrade FIFOs to 8K each. 3/4/20
Figure 6 XMC-PARALLEL-TTL Revision History

XmcParTtl_STATUS

[$08 Board level Status Port read only]

DATA BIT	DESCRIPTION
31	Interrupt Status
30-18	
17	int_stat1
16	int_stat0
15-6	spare
5	INTR Falling
4	INTR Rising
3-1	spare
0	local interrupt

Figure 7 XMC-PARALLEL-TTL Status Port Bit Map

Local Interrupt for the base design, this bit is the same as the Intforce bit – unmasked.

INTR Rising - This is the logical OR of the COS outputs for the Rising Edge condition. The RISING register will select which bits are enabled. If any of the enabled bits are active this bit is set. The status is captured before the master interrupt enable. If the master interrupt enable is set an interrupt will be generated if this condition is true.

INTR Falling - This is the logical OR of the COS outputs for the Falling Edge condition. The Falling register will select which bits can be active [enabled]. If any of the enabled bits capture a falling edge this bit will be set. The status is captured before the master interrupt enable. If the master interrupt enable is set an interrupt will be generated if this condition is true.

Int_Stat0/1 – This is the local masked not board level masked interrupt from channel 0. Int_Stat0/1 = DMA Write and DMA Write mask or DMA Read and DMA Read Mask or (IntForce or TX request) and Channel 0/1 mask. This bit will tell the SW if any channel 0/1 asset could be requesting an interrupt. If the master interrupt enable is set an interrupt will be generated if this condition is true.

Interrupt Status – Set if the PCI interrupt is asserted. This bit can be checked to determine if this card is causing an interrupt to the system. If set the other bits can be checked to see which feature(s) of the board need to be serviced. Secondary reads to the COS or Channel will determine the exact type of interrupt.

XmcParTtl_DirL

[$0C Direction Register bits 31-0 read – write]

DATA BIT	DESCRIPTION
31-0	DIR31-0

Figure 8 XMC-PARALLEL-TTL Direction Lower Bit Map

The lower 32 bits of the parallel port direction are controlled with this port. When reset this port is cleared 0x00000000. All IO are set to read [inputs]. To use one or more of the IO for outputs; program the corresponding direction bit(s) to ‘1’.

XmcParTtl_DirU

[$10 Direction Register bits 63-32 read – write]

DATA BIT	DESCRIPTION
31-0	DIR63-32

Figure 9 XMC-PARALLEL-TTL Direction Upper Bit Map

The upper 32 bits of the parallel port direction are controlled with this port. When reset this port is cleared 0x00000000. All IO are set to read [inputs]. To use one or more of the IO for outputs; program the corresponding direction bit(s) to ‘1’.

Once a Direction bit is set to output the data in the corresponding output holding register bit is broadcast on that IO line. The data in the holding register will match the data in the data output register if the master parallel enable bit is set. If initial states are important you may want to program the initial data and enable it before enabling the direction bits.

XmcParTtl_DatL

[$14 Data IO Port read/write]

DATA BIT	DESCRIPTION
31-0	Data IO 31-0

Figure 10 XMC-PARALLEL-TTL Data IO Lower Bit Map

XmcParTtl_DatU

[$18 Data IO Port read/write]

DATA BIT	DESCRIPTION
31-0	Data IO 63-32

Figure 11 XMC-PARALLEL-TTL Data IO Upper Bit Map

This port is really a combined Data Output port and a Data Input port. The data to be transmitted is written to the Data Output Port side of the Data Register. The data to be read from the IO are read from Data Input side of the Data register. Read back from the Data Output port is done though the separate "datareg" port.

The data read from the data register is a direct read of the state of the IO lines. The bits are not modified for level or transition etc. Some bits may be defined as outputs. The input will match the output definition in this case. Local loop-back can be performed for the bits where outputs are defined. The inputs will match the state of the system when external devices can drive the input lines. The input bits can be masked out of the data word to reduce the data to external inputs.

The output bits are driven onto the IO for the bits that are enabled with the direction control register, and when the master parallel enable is set. For bits without the direction register bit set there are no side effects. The direction register will act as a mask for the data register.

XmcParTtl_DatLreg

[$1C Data Reg Port read only]

DATA BIT	DESCRIPTION
31-0	Data IO 31-0

Figure 12 XMC-PARALLEL-TTL Data Reg Lower Bit Map

XmcParTtl_DatUreg

[$20 Data Reg Port read only]

DATA BIT	DESCRIPTION
31-0	Data IO 63-32

Figure 13 XMC-PARALLEL-TTL Data Reg Upper Bit Map

Data written to the Data IO registers can be read back through this port. The register is read back instead of the IO side when accessing this port. The data will match the state of the data output bits written to the output side of the Data IO register.

XmcParTtl_COSclk

[$24 COS clock definition port read -write]

DATA BIT	DESCRIPTION
15	Data Out 0 Enable
14-13	CLOCK PRE-SELECTOR
12	CLOCK POST-SELECTOR
11-0	DIVISOR

Figure 14 XMC-PARALLEL-TTL COS Clk Control Bit Map

Data Out 0 Enable when set and the corresponding Direction bit is set will drive the COS clock out on Data bit 0. An oscilloscope can be used to verify the frequency setting that is programmed with the COSclk register.

CLOCK PRE-SELECTOR

00	PCI Clock/2
01	Oscillator
10	External Clock
11	PCI Clock/2

The clock pre-selector is used to select which reference clock to use with the divisor hardware (clock source). The base design oscillator rate is 50 MHz. The external clock can be any TTL level source driven onto the External Clock input line. The clock should be free running to be used for this purpose. The PCI clock reference is 50 MHz/2 => 25 MHz.

POST-SELECTOR when '1' sets the output clock to the divided clock, when '0' sets the output clock to the pre-selector reference value (clock source).

DIVISOR[11-0] are the clock divisor select bits. The clock source is divided by a 12-bit counter. The output frequency is {reference / [2(n+1)]}, $n \geq 1$. The counter divides by N+1 due to counting from 0 to n before rolling over. The output is then divided by 2 to produce a square wave output.

The desired frequency of 1 MHz. is achieved by selecting Osc reference, divided clock and a factor of 50 with the standard 50 MHz oscillator. 2(N+1) = 50 => N = 24. 0x3018 would be the correct value to write to the COSclk.

XmcParTtl_RisLreg

$2C Rising Lower Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Rising 31-0

Figure 15 XMC-PARALLEL-TTL Rising Lower Bit Map

XmcParTtl_RisUreg

$30 Rising Upper Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Rising 63-32

Figure 16 XMC-PARALLEL-TTL Rising Upper Bit Map

The Rising control register bits correspond to the input data bits. All IO can be set-up for COS activity even if defined as an output. In most cases the output bits will be set to ‘0’ for the Rising register. When set ‘1’ and the corresponding input bit transitions from low to high the COS register of rising activity will be have the corresponding bit set. If the separate interrupt enable bit is also set then an interrupt can be generated. The Rising register is a control register. The COS data is read back separately.

XmcParTtl_FallLreg

$34 Rising Lower Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Falling 31-0

Figure 17 XMC-PARALLEL-TTL Falling Lower Bit Map

XmcParTtl_FallUreg

$38 Rising Upper Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Falling 63-32

Figure 18 XMC-PARALLEL-TTL Falling Upper Bit Map

The Falling control register bits correspond to the input data bits. All IO can be set-up for COS activity even if defined as an output. In most cases the output bits will be set to ‘0’ for the Falling register. When set ‘1’ and the corresponding input bit transitions from High to Low the COS register of falling activity will be have the corresponding bit set. If the separate interrupt enable bit is also set then an interrupt can be generated. The Falling register is a control register. The COS data is read back separately.

XmcParTtl_IntRisLreg

$3C Rising Interrupt Lower Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Rising Int En 31-0

Figure 19 XMC-PARALLEL-TTL Int rising Lower Bit Map

XmcParTtl_IntRisUreg

$40 Rising Interrupt Upper Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Rising Int En 63-32

Figure 20 XMC-PARALLEL-TTL int Rising Upper Bit Map

The Rising Interrupt Enable control register bits correspond to the input data bits. All IO can be set-up for COS activity even if defined as an output. In most cases the output bits will be set to ‘0’ for the Rising Interrupt Enable register. When set ‘1’ and the corresponding Rising bit is captured by the COS register an interrupt can be generated. Please note that the master interrupt enable will also need to be set for the interrupt to be requested.

XmcParTtl_IntFallLreg

$44 Falling Interrupt Lower Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Falling Int En 31-0

Figure 21 XMC-PARALLEL-TTL Int Falling Lower Bit Map

XmcParTtl_IntFallUreg

$48 Falling Interrupt Upper Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Falling Int En 63-32

Figure 22 XMC-PARALLEL-TTL int Falling Upper Bit Map

The Falling Interrupt Enable control register bits correspond to the input data bits. All IO can be set-up for COS activity even if defined as an output. In most cases the output bits will be set to ‘0’ for the Falling Interrupt Enable register. When set ‘1’ and the corresponding falling bit is captured by the COS register an interrupt can be generated. Please note that the master interrupt enable will also need to be set for the interrupt to be requested.

XmcParTtl_IntRisLstat

$4C Rising Status Lower Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Rising COS bits 31-0

Figure 23 XMC-PARALLEL-TTL Rising COS Status Lower

XmcParTtl_IntRisUstat

$50 Rising Status Upper Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Rising COS bits 63-32

Figure 24 XMC-PARALLEL-TTL Rising COS status upper

The COS captured for those bits enabled with the Rising register are held in this register. The bits are held until cleared. Bits are cleared by writing to the register with the corresponding bit or bits set. Writing to the register with the data read will clear the bits the software has read, and not clear the bits not set at the time of reading. This is the recommended practice to avoid conflicts. It is recommended to write to all bits [clear] after setting the COS Rising and Direction bits to clear any potential COS status generated by set-up.

XmcParTtl_IntFallLstat

$54 Falling Status Lower Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Falling COS Status bits 31-0

Figure 25 XMC-PARALLEL-TTL Falling COS Status Lower

XmcParTtl_IntFallUstat

$58 Falling Status Upper Control Register Port read/write

DATA BIT	DESCRIPTION
31-0	Falling COS Status bits 63-32

Figure 26 XMC-PARALLEL-TTL Falling COS status upper

The COS captured for those bits enabled with the Falling register are held in this register. The bits are held until cleared. Bits are cleared by writing to the register with the corresponding bit or bits set. Writing to the register with the data read will clear the bits the software has read, and not clear the bits not set at the time of reading. This is the recommended practice to avoid conflicts. It is recommended to write to all bits [clear] after setting the COS Falling and Direction bits to clear any potential COS status generated by set-up.

XmcParTtl_DR_L

[$0x5C DMA Register bits 31-0 read – write]

DATA BIT	DESCRIPTION
31-0	DMA or Register control 31-0

Figure 27 XMC-PARALLEL-TTL DMA Reg Lower Bit Map

The lower 32 bits of the DMA / Register selection are controlled with this port. When reset this port is cleared 0x00000000. All IO are set to register control. To use one or more of the IO for DMA controlled functions; program the corresponding direction bit(s) to ‘1’.

XmcParTtl_DR_U

[$60 DMA Register bits 63-32 read – write]

DATA BIT	DESCRIPTION
31-0	DMA or Register control 63-32

Figure 28 XMC-PARALLEL-TTL Direction Upper Bit Map

The upper 32 bits of the DMA / Register selection are controlled with this port. When reset this port is cleared 0x00000000. All IO are set to register control. To use one or more of the IO for DMA controlled functions; program the corresponding direction bit(s) to ‘1’.

To use the DMA function of programmed parallel data output, the direction register bits and DR register bits corresponding to those outputs must be set to ‘1’. The Direction bits enable the IO, and the DR bits select the State Machine output path instead of the Register path. Please note that the bits are selected on a bit by bit basis. Pick the closest larger size with the state-machine and the actual size with the DR and Direction bits. Any unused bits [by the state machine with masking] can be used as registered IO or COS inputs.

XmcParTtl_ch0,1_base

[0x78, A0] Channel Control Register (read/write)

Channel Control Register

Data Bit	Description
19	BI IDLE
18	BO IDLE
17	TX IDLE
16	RX IDLE
15	RX Endian
14	CLK RX Sel
13	CLK TX Sel
12	TX MT Mode
11	Spare
10	TX Endian
9	TX Mode 1
8	TX Mode 0
7	Enable RX
6	Enable TX
5	Force Interrupt
4	Channel Interrupt Enable
3	Read DMA Interrupt Enable
2	Write DMA Interrupt Enable
1	FIFO Bypass Enable
0	FIFO Reset

Figure 29 XMC-PARALLEL-TTL channel Control Register

FIFO Reset: When set to a one, the transmit and receive FIFOs will be reset. When these bits are zero, normal FIFO operation is enabled. In addition the TX and RX State Machine is also reset.

FIFO Bypass Enable: When this bit is set to a one, any data written to the transmit FIFO will be transferred to the receive FIFO. This allows for fully testing the data FIFOs without using the I/O. When this bit is zero, normal FIFO operation is enabled. The rate at which the data is transferred depends on the clock selection. It is recommended to operate with the clock set to the default oscillator selection when using the bypass mode. Bypass should be set before moving data into the TX FIFO. DMA can be used with Bypass mode.

Write/Read DMA Interrupt Enable: These two bits, when set to one, enable the interrupts for DMA writes and reads respectively. The DMA interrupts are not affected by the Master Interrupt Enable.

Channel Interrupt Enable: 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. An additional board level master interrupt enable is located in the Base register. The board level master must also be enabled to gate the interrupt through to the host.

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

Enable TX: When set '1' will start the TX function. The control bits for the TX operation should be selected first to guarantee correct operation as the TX output rate may be different from the PCI clock rate. The transmitter function will read data from the TX FIFO and output that data from the TTL outputs.

Enable TX will be cleared by the State-machine when the end conditions are met or can be cleared by software to stop transmission at the next LW boundary. For example if byte mode is selected and the enable is cleared by software mid word, the rest of the word will be sent and then the HW will stop.

Enable RX: When set '1' will start the RX function. The control bits for the RX operation should be selected first to guarantee correct operation as the RX reference rate may be different from the PCI clock rate. The receiver function will write data to the RX FIFO until software disables the data capture.

TX Mode 1:0 The size of the output data is selected with this 2 bit field.
00 Byte [8 bits]
01 Word [16 bits]
10 LW [32 bits]
11 QW [64 bits]

Please note Byte mode should be used with the BA16 design.

TX Endian is used to select the order that the data output is pulled from the FIFO.

The standard pattern for Byte Mode is to load the output pipeline from the bottom so that 7-0 appear on 7-0 first, 15-8 appear on 7-0 second, 23-16 appear on 7-0 third 31-24 last. Then repeat for the next 32 bit word. If Endian = '1' is selected the order is reversed to pull from the top. 31-24 appears on IO 7-0 first, 23-16 on 7-0 second and so forth.

TX Empty Mode when '0' causes the transmission process to stop when the FIFO becomes empty. For small transfers or when the rate of transfer is low compared to the PCI capability this mode is recommended. The data will burst out until the FIFO goes empty at which time the transmission will stop and the TX Enable will be cleared by the state machine.

When TX Empty Mode is set to '1' the state-machine uses the MT signal to determine that the hardware needs to pause rather than stop. The pipeline is paused with the last value remaining on the IO until data is present in the TX FIFO. The reference output clock is halted when data is not being transmitted to allow a receiver to determine when new data is available.   This mode is recommended when the FIFO might become empty prior to completion.

Clock TX Select when '1' selects the PLL A reference clock to be used for transmission. When '0' the oscillator [50 MHz] is selected. For most systems the PLL channel A will be used for speed control. Set the PLL channel A to the transmission rate desired. The PLL is controlled through the base register. The clock reference controls the TX FIFO read side reference clock and the state machine reference rate. Set to oscillator for Bypass mode.

Clock RX Select when '1' Selects PLL B reference clock to be used for reception. When '0' the oscillator [50 MHz] is selected. For most systems the oscillator will be used for speed control. Set the PLL channel B to the sampling rate desired for the receiver if frequencies higher than 8 MHz are expected. 6x the expected frequency [or more]. The PLL is controlled through the base register.   Set to oscillator for Bypass mode.

RX and TX IDLE are set when the respective state-machines are in the idle states. When clock rates other than PCI are used it may take a while to clean-up and return to the idle state – waiting for the next command. For example, if a 1 MHz output clock is used and the transmission is in byte mode then several clocks will be required to finish the transmission, clear the start bit and return to idle. If SW has cleared the start bit to finish the transmission off then the SW will not have a flag to determine when the clearing action has completed. When the IDLE bit is set 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 TX and burst data out of the card for Receive.

XmcParTtl_ch0,1_st

[0x7C,A4] Channel Status Read/Clear Latch Write Port

Channel Status Register

Data Bit	Description
31	Channel Interrupt Active
30-16	data count
15	Read DMA Interrupt Occurred
14	Write DMA Interrupt Occurred
13	Read DMA Error Occurred
12	Write DMA Error Occurred
11	spare
10	spare
9	Receive Done Interrupt Occurred
8	Transmit Done Interrupt Occurred
7	Receive Data Valid
6	Receive FIFO Full
5	Receive FIFO Almost Full
4	Receive FIFO Empty
3	spare
2	Transmit FIFO Full
1	Transmit FIFO Almost Empty
0	Transmit FIFO Empty

Figure 30 XMC-Parallel-TTL Channel STATUS PORT

Transmit FIFO Empty: When a one is read, the transmit data FIFO 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 is less than or equal to the value written to the TX_AMT_LVL register; when a zero is read, the FIFO level is more than that value.

Transmit 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.

Please note with the Receive 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 valid flag can be used to monitor the last few accesses. The data count is the combined FIFO and pipeline value and can also be used for read size control.

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

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

Receive 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.

Receive Data Valid: When a one is read, there is at least one valid receive data word left. This bit can be set even if the receive FIFO is empty, because as soon as the first four words are written into the FIFO, they are read out to fill the receive data pipe-line to be ready for a PCI read DMA or single word access. When this bit is a zero, it indicates that there is no valid receive data remaining.

Transmit Done Interrupt Occurred: When a one is read, it indicates that the transmit state-machine has completed. A zero indicates that a transmit message has not been completed. This bit is latched and can be cleared by writing back to the Status register with a one in this bit position.

Receive Done Interrupt Occurred: When a one is read, it indicates that the receive state-machine has received at least one complete message. At least one byte must have been received and then the receive data line must be idle for at least eight bit-periods for a message be considered completed. A zero indicates that a complete message has not been received. This bit is latched and can be cleared by writing back to the Status register with a one in this 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.

Channel Interrupt Active: When a one is read, it indicates that a system interrupt is potentially asserted caused by an enabled channel interrupt condition. A zero indicates that no system interrupt is pending from an enabled channel interrupt condition. The Board level master interrupt enable will also need to be asserted to allow the active channel interrupt to become an interrupt request.

XmcParTtl_ch0,1_brstin

[0x80,A8] Write DMA Pointer (write only)

DMA Pointer Address Register

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

Figure 31 XMC-Parallel-TTL 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.

XmcParTtl_ch0,1_brstout

[0x84,AC] Read DMA Pointer (write only)

DMA Pointer Address Register

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

Figure 32 XMC-Parallel-TTL 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.

XmcParTtl_ch0,1_swr

[0x88,B0] Write TX/Read RX FIFO Port

RX and TX FIFO Port

Data Bit	Description
31-0	FIFO data word

Figure 33 XMC-Parallel-TTL RX/TX FIFO Port

This port is used to make single-word accesses into the TX and out of the RX FIFO. Please note that reading is from the RX FIFO and writing is to the TX FIFO. Unless Bypass mode is established the data will not match.

XmcParTtl_ch0,1_tx_aecnt

[0x8C,B4] TX almost-empty level (read/write)

TX Almost-Empty Level Register

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

Figure 34 XMC-Parallel-TTL TX ALMOST EMPTY LEVEL register

This read/write port accesses the transmitter almost-empty level register. When the number of data words in the transmit data FIFO is equal or less 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. 2k x32 is the TX FIFO for an 11 bit valid count range [10-0].

XmcParTtl_ch0,1_rx_afcnt

[0x90,B8] RX almost-full level (read/write)

RX Almost-Full Level Register

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

Figure 35 XMC-Parallel-TTL RX ALMOST FULL LEVEL register

This read/write port accesses the receiver 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. 4k x32 is the RX FIFO for a 12 bit valid count range [11-0].

XmcParTtl_ch0,1_tx_ffcnt

[0x94,BC] TX FIFO data count (read only)

TX FIFO Data Count Port

Data Bit	Description
31-12	Spare
11-0	TX Data Words Stored

Figure 36 XMC-Parallel-TTL TX fifo data count Port

This read-only register port reports the number of 32-bit data words in the transmit FIFO. The TX FIFO has a maximum of 2047 locations.

XmcParTtl_ch0_rx_ffcnt

[0x98, C0] RX FIFO data count (read only)

RX FIFO Data Count Port

Data Bit	Description
31-12	Spare
11-0	RX Data Words Stored

Figure 37 XMC-Parallel-TTL RX fifo data count Port

This read-only register port reports the number of 32-bit data words in the receive FIFO. The channel status register contains the combined pipeline and FIFO count. The size depends on the FIFO size. This design has 4095 locations possible in the FIFO.

XmcParTtl_Temp

[0x006C] Temperature Register

Temperature Port

Data Bit	Description
31-16	Spare
15-14	“00”
13	Ready
12	Sign
11-0	Data

Figure 38 XMC-Parallel-TTL Temperature Port

Writing to the port starts a read of the TMP123 temperature Sensor. The temperature sensor is accurate to 1.5C, has 12 bit data plus sign. .0625 is the constant.

Reading from the port returns the current data captured with the write. The interface is serial and takes approximately 2 uS to return a value from Write to Read. No particular data pattern is required for the write 0x00 is suggested.

In addition, the TMP123 makes new conversions based on CS being high [not being written to] and does one every ½ second. CS must remain high for 320 mS [worst case] to have a conversion completed to read. Each time CS is taken low [new load command] the current conversion is terminated and the previous conversion transferred. Over sampling will stop new conversions.

An example of reading and converting is supplied in the reference SW packages.

Loop-back

The Engineering kit has reference software, which includes external loop-back tests. XMC-Parallel-TTL has a 68 pin VHDCI front panel connector. The tests require an external cable with the following pins connected. Rear IO connection options are also available.

For Parallel Port loop-back the full IO0-31 is cross connected with IO32-63.

External BA16 function Loop-Back

Signal RX	**From**	**To**	**Signal TX**
IO_0 [D00]	pin 33	pin 67	IO_32
IO_1 [D01]	pin 32	pin 66	IO_33
IO_2 [D02]	pin 31	pin 65	IO_34
IO_3 [D03]	pin 30	pin 64	IO_35
IO_4 [D04]	pin 29	pin 63	IO_36
IO_5 [D05]	pin 28	pin 62	IO_37
IO_6 [D06]	pin 27	pin 61	IO_38
IO_7 [D07]	pin 26	pin 60	IO_39
IO_8 [ALGN320]	pin 25	pin 59	IO_40
IO_9 [CLK0]	pin 24	pin 58	IO_41
IO_16 [D10]	pin 17	pin 51	IO_48
IO_17 [D11]	pin 16	pin 50	IO_49
IO_18 [D12]	pin 15	pin 49	IO_50
IO_19 [D13]	pin 14	pin 48	IO_51
IO_20 [D14]	pin 13	pin 47	IO_52
IO_21 [D15]	pin 12	pin 46	IO_53
IO_22 [D16]	pin 11	pin 45	IO_54
IO_23 [D17]	pin 10	pin 44	IO_55
IO_24 [ALGN321]	pin 9	pin 43	IO_56
IO_25 [CLK1]	pin 8	pin 42	IO_57

XMC Module Front Panel IO Interface Pin Assignment

The figure below gives the pin assignments for the XMC Module IO Interface on the XMC-Parallel-TTL. Installed for –FP and –FRP models. Also see the User Manual for your carrier board for more information.

EXT_CLK_EN	EXT_CLK	1	35
IO_31	IO_63	2	36
IO_30	IO_62	3	37
IO_29	IO_61	4	38
IO_28	IO_60	5	39
IO_27	IO_59	6	40
IO_26	IO_58	7	41
IO_25	IO_57	8	42
IO_24	IO_56	9	43
IO_23	IO_55	10	44
IO_22	IO_54	11	45
IO_21	IO_53	12	46
IO_20	IO_52	13	47
IO_19	IO_51	14	48
IO_18	IO_50	15	49
IO_17	IO_49	16	50
IO_16	IO_48	17	51
IO_15	IO_47	18	52
IO_14	IO_46	19	53
IO_13	IO_45	20	54
IO_12	IO_44	21	55
IO_11	IO_43	22	56
IO_10	IO_42	23	57
IO_9	IO_41	24	58
IO_8	IO_40	25	59
IO_7	IO_39	26	60
IO_6	IO_38	27	61
IO_5	IO_37	28	62
IO_4	IO_36	29	63
IO_3	IO_35	30	64
IO_2	IO_34	31	65
IO_1	IO_33	32	66
IO_0	IO_32	33	67
GND	GND	34	68

Figure 39 XMC-PARALLEL-TTL FRONT PANEL Interface

XMC Module Front Panel IO Interface Pin Assignment

The figure below gives the pin assignments for the XMC Module IO Interface on the XMC-Parallel-TTL-BA16. See the User Manual for your carrier board for more information.

RESERVED	RESERVED	1	35
IO_31	IO_63	2	36
IO_30	IO_62	3	37
IO_29	IO_61	4	38
IO_28	IO_60	5	39
IO_27	IO_59	6	40
IO_26	IO_58	7	41
CLK_IN1	CLK_OUT1	8	42
ALIGN32_IN1	ALIGN32_OUT1	9	43
DATA_IN_17	DATA_OUT_17	10	44
DATA_IN_16	DATA_OUT_16	11	45
DATA_IN_15	DATA_OUT_15	12	46
DATA_IN_14	DATA_OUT_14	13	47
DATA_IN_13	DATA_OUT_13	14	48
DATA_IN_12	DATA_OUT_12	15	49
DATA_IN_11	DATA_OUT_11	16	50
DATA_IN_10	DATA_OUT_10	17	51
IO_15	IO_47	18	52
IO_14	IO_46	19	53
IO_13	IO_45	20	54
IO_12	IO_44	21	55
IO_11	IO_43	22	56
IO_10	IO_42	23	57
CLK_IN0	CLK_OUT0	24	58
ALIGN32_IN0	ALIGN32_OUT0	25	59
DATA_IN_07	DATA_OUT_07	26	60
DATA_IN_06	DATA_OUT_06	27	61
DATA_IN_05	DATA_OUT_05	28	62
DATA_IN_04	DATA_OUT_04	29	63
DATA_IN_03	DATA_OUT_03	30	64
DATA_IN_02	DATA_OUT_02	31	65
DATA_IN_01	DATA_OUT_01	32	66
DATA_IN_00	DATA_OUT_00	33	67
GND	GND	34	68

Figure 40 PMC-PARALLEL-TTL BA16 FRONT PANEL Interface

_IN = RX, _OUT = TX, _0x = channel 0, *1x = channel 1*
IOxx are un-committed IO that can be used for registered IO or COS.

XMC Module Backplane IO Interface Pin Assignment

The figure below gives the pin assignments for the XMC Module IO Interface on the XMC-Parallel-TTL and routed to Pn4.   Pn4 installed for –RP and –FRP models. Option to install for BA16.   Also see the User Manual for your carrier board for more information. See also PN6 definitions

IO_0	IO_1	1	2
IO_2	IO_3	3	4
IO_4	IO_5	5	6
IO_6	IO_7	7	8
IO_8	IO_9	9	10
IO_10	IO_11	11	12
IO_12	IO_13	13	14
IO_14	IO_15	15	16
IO_16	IO_17	17	18
IO_18	IO_19	19	20
IO_20	IO_21	21	22
IO_22	IO_23	23	24
IO_24	IO_25	25	26
IO_26	IO_27	27	28
IO_28	IO_29	29	30
IO_30	IO_31	31	32
IO_32	IO_33	33	34
IO_34	IO_35	35	36
IO_36	IO_37	37	38
IO_38	IO_39	39	40
IO_40	IO_41	41	42
IO_42	IO_43	43	44
IO_44	IO_45	45	46
IO_46	IO_47	47	48
IO_48	IO_49	49	50
IO_50	IO_51	51	52
IO_52	IO_53	53	54
IO_54	IO_55	55	56
IO_56	IO_57	57	58
IO_58	IO_59	59	60
IO_60	IO_61	61	62
IO_62	IO_63	63	64

Figure 41 XMC-PARALLEL-TTL PN4 Interface

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. Open Drain 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. Series resistors are used and can be specified to be something other than the 22 ohm standard value. The connector is pinned out for a standard SCSI II/III cable to be used. It is suggested that this standard cable be used for most of the cable run.

Terminal Block. We offer a high quality 68 screw terminal block that directly connects to the SCSI II/III cable. The terminal block can mount on standard DIN rails. HDEterm68 [http://www.dyneng.com/HDEterm68.html] Use with a VHDCI to SCSI cable.

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

XMC Modules were conceived and engineered for rugged industrial environments. XMC-Parallel-TTL 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 options for front panel and rear panel are isolated with series resistor packs to eliminate bus stubs when one of the connectors is not in use.

Surface mounted components are used.

The XMC Module connectors are keyed and shrouded with Gold plated pins on both plugs and receptacles. They are rated at 1 Amp per pin, 50 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable.

The XMC is secured against the carrier with the connectors and front panel. If more security against vibration is required the stand-offs can be secured against the carrier.

The XMC Module provides a low temperature coefficient of 2.17 W/^{O}C for uniform heat. This is based upon the temperature coefficient of the base FR4 material of 0.31 W/m-^{O}C, and taking into account the thickness and area of the XMC. 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.

Thermal Considerations

XMC-PARALLEL-TTL 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; 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 St. Suite C
Santa Cruz, CA 95060
831-457-8891
support@dyneng.com

Specifications

Logic Interface:	XMC Logic Interface [PCIe ⇔ PCI] 32/50 at PCI.
Digital Parallel IO:	64 discrete IO channels. Each has a separate enable to control output. Inputs are maskable and always available. Two channels of “BA16” IO each with programmable rate TX. 8 bit parallel with reference clock and align.
CLK rates supported:	Osc, PLL, PCI, External reference rates coupled with 12 bit divider to allow user programmed sample rate for COS., Osc and PLL programmed TX rate for BA16 interface. Programmed bandwidth on RX [PLLB]
Software Interface:	Control Registers, IO registers, IO Read-Back registers
Initialization:	Programming procedure documented in this manual
Access Modes:	LW to registers, read-write to most registers
Access Time:	Frame to TRDY 4 PCI clocks or burst mode DMA – 1 word per PCI clock transferred. 50 MHz reference.
Interrupt:	All IO lines can be used as interrupt sources with programmable rising and or falling activity on IO line “COS”, DMA interrupts, TX function [BA16]
Onboard Options:	All Options are Software Programmable
Interface Options:	68 Pin VHDCI connector at front bezel User IO routed to Pn4/Pn6
Dimensions:	Standard Single XMC Module.
Construction:	Multi-Layer Printed Circuit, Through Hole and Surface Mount Components.
Temperature Coefficient:	2.17 W/^{o}C for uniform heat across PMC
Power:	TBD mA @ 5V outputs off Add 10 mA per active low output for pull-up current drivers support +/- 24 mA per IO line.

Order Information

standard temperature range -40-80ºC
XMC-Parallel-TTL-BA16 XMC Module with 64 IO channels, COS and direct IO, 2 ports of “BA16” parallel data interface – 8 bits parallel, reference clock, align [strobe], front panel IO, 3.3V reference voltage to pull-ups, DMA support, 4Kx32 FIFO RX, 2Kx32 FIFO TX per channel.
https://www.dyneng.com/XMC-Parallel-TTL.html

Order Options:

Pick One
–FP for front panel IO only [default if no selection made]
-RP for rear panel IO PN4 only
-**XIO** for rear panel both Pn4 and Pn6
-**XIOexc** for Pn6 only

Adding -RP, -XIO, -XIOexc will remove the default VHDCI connector and install the rear IO options.

Pick any combination to go with IO
-CC to add conformal coating

Related:

PCIe8LXMCX1: PCIe to XMC adapter to allow installation of XMC-Parallel-TTL into a PCIe system.
https://www.dyneng.com/PCIe8LXMCX1.html

HDEterm68: 68 position terminal block with two SCSI II/III connectors. XMC-Parallel-TTL compatible with VHDCI to SCSI cable.
https://www.dyneng.com/HDEterm68.html

XMC-UNIT-Test: PCIe to XMC adapter to allow installation of XMC-Parallel-TTL into a PCIe system. Vertical adapter for debugging etc.
https://www.dyneng.com/XMC-UNIV-TEST.html