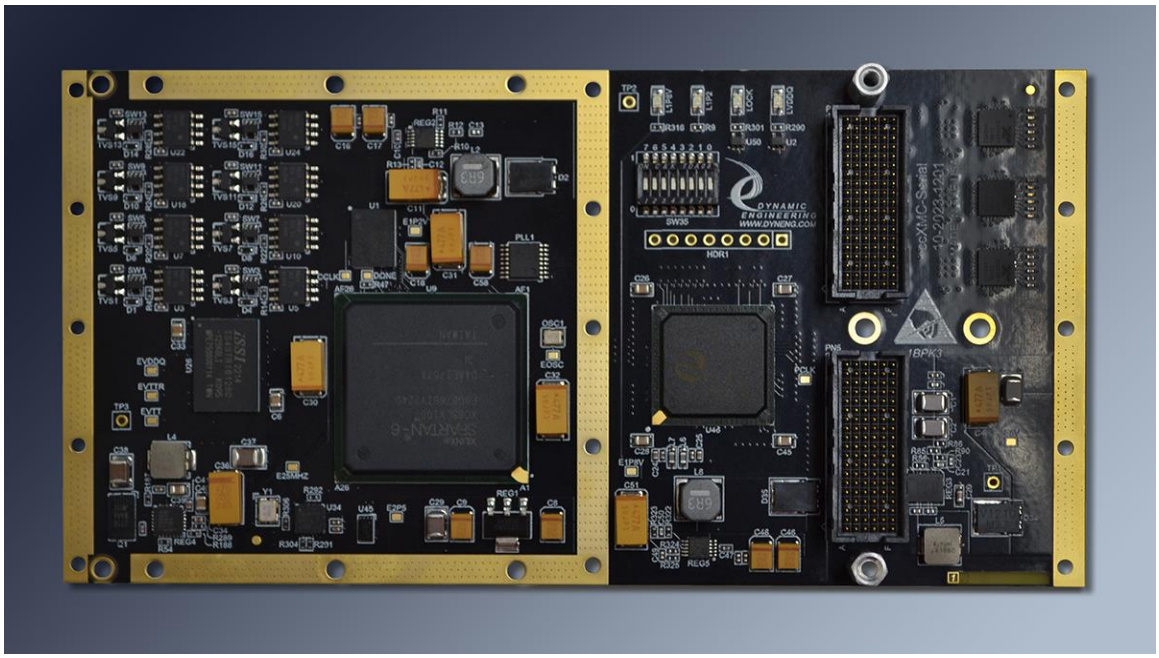
DYNAMIC ENGINEERING
150 DuBois, Suite B&C Santa Cruz, CA 95060
(831) 457-8891
https://www.dyneng.com sales@dyneng.com
Est. 1988

User Manual

ccXMC-Serial
HDLC, NRZ-L, UART

2 Ports HDLC, 2 Ports NRZ-L, 3 Ports UART

Conduction Cooled XMC Module



Manual Revision 1p2 8/16/24
Corresponding PCB : 10-2023-1201

ccXMC-Serial
PMC Module

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

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

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

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

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

Table of Contents

List of Figures

Product Description

ccXMC-Serial is part of the XMC Module family of components by Dynamic Engineering. ccXMC-Serial is capable of providing multiple protocols. The base version implements includes 2 ports of HDLC, 2 ports of NRZ-L and 3 UARTs. Alternate designs will have a "-#" specifying the model. The UARTs have programmable 422 or 232 operation. The HDLC and NRZL are implemented with RS-485 and support a build option for LVDS.

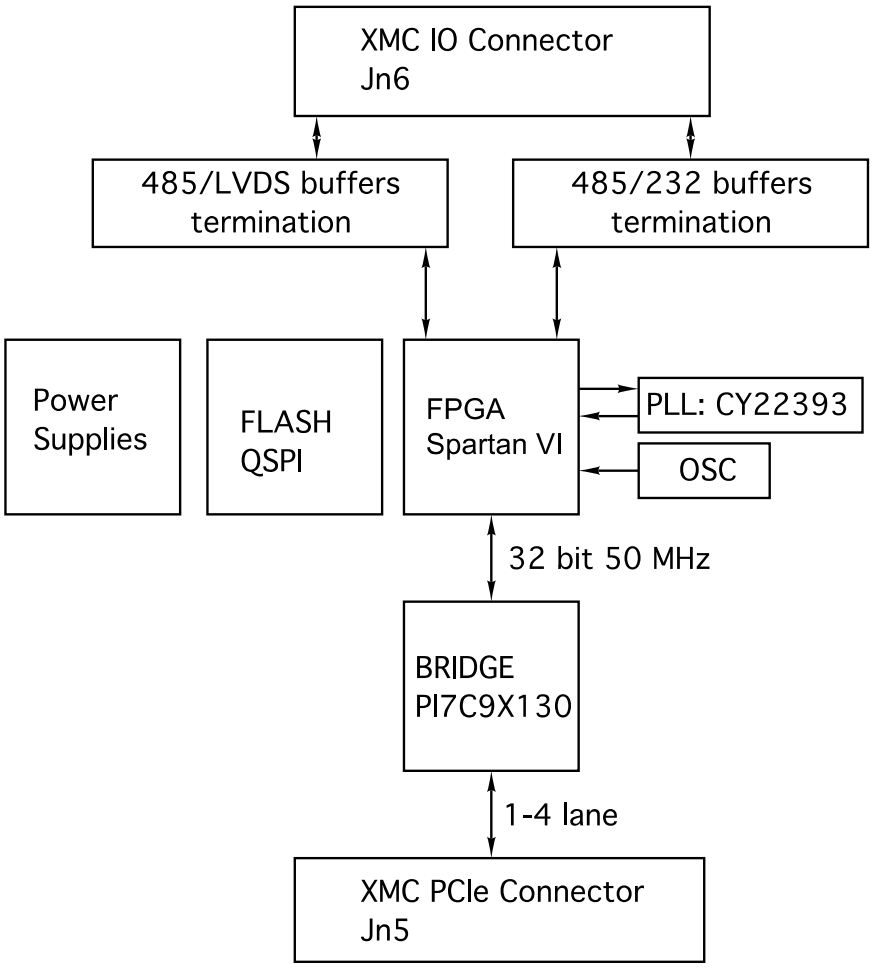


FIGURE 1 CCXMC-SERIAL TOP LEVEL BLOCK DIAGRAM

ccXMC-Serial is a conduction cooled XMC device with a mix of IO types. Software control originates with the host computer and accesses the device over the PCIe bus. The bridge converts between a local PCI bus and the host PCIe bus. The base design of the FPGA implements a target PCI interface. The memory map is linear and separated by port. All addresses are directly accessible.

The FPGA requires voltages not supplied by an XMC requiring power supplies for 2.5V, 1.8V, and 1.2V. VPWR can be 12V or 5V. A Buck-Boost supply converts VPWR to 5V. Several of the other power supplies are referenced to the 5V reference for sequencing purposes. Several LEDs are provided to demonstrate the proper operation of the supplies.

QSPI Flash is used to store the configuration program for the FPGA. The JTAG interface on the XMC connector is used to program the Flash.

A local oscillator provides 32 MHz to the FPGA to use an internal reference and to provide a known reference to the PLL.

The PLL has 4 programmable outputs based on the reference clock and parameters loaded over an I2C bus. The I2C bus is under user control to allow new files to be loaded, changing the A-D outputs of the PLL. The HDLC ports have separate references to allow transmit frequency control and a reference to sample the received clock for the Rx function. NRZL uses selectors to support each port with a programmable counter to provide separate clocks to those ports. The UART ports have access to PLL D and the 32 MHz reference. Local dividers are used to control each of the three ports. See the base register to make reference selections for each port. PLLA, PLLB, or PLLC can be selected [software] to be the reference as shown.

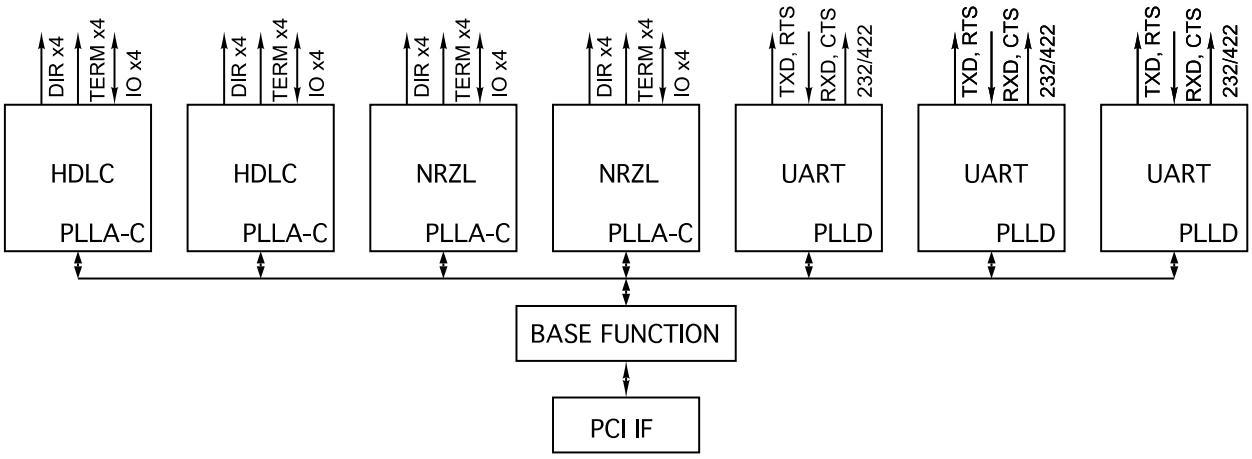


FIGURE 2 CCXMC-SERIAL FPGA BLOCK DIAGRAM

ccXMC-Serial conforms to the XMC and CMC standards. 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 uses a different one. Contact Dynamic Engineering for a copy of this specification. It is assumed that the reader is at least casually familiar with these documents.

Rear IO definitions are SOSA [Sensor Open Systems Architecture] compliant. See tables are end of document for signal assignments. Initial carrier in use is Concurrent Technologies TR L98 3U VPX with processor and local XMC site.

In standard configuration, ccXMC-Serial is a Type 1 mechanical with only low-profile components on the back of the board and one position wide, with 10 mm inter-board height. The 10 mm height is the "standard" height and will work in most systems with most carriers. If your carrier has non-standard connectors (height) to mate with ccXMC-Serial, please let us know. We may be able to do a special build with a different height connector to compensate. ccXMC-Serial is conduction cooled and uses the shorted conduction cooled length as defined in the VITA specification.

The block diagram shown in Figure 1 highlights the main features of the design. The FPGA is a Spartan-VI in the 676 package. The 100 size is typically used and other sizes can be supported. Industrial temperature parts are used throughout.

The PLL provides 4 clock references, in addition to the PCI and oscillator inputs. The PLL is referenced to 40 MHz. The 40 MHz reference comes from a 32 MHz oscillator.

16 differential IO are provided each with separate termination and direction control. The IO transceivers can be installed with RS-485 or LVDS devices. Each differential pair is protected with a TVS device [400W]. The termination, and direction for each transceiver are programmable through the Xilinx device to provide flexibility in the mix of outputs and inputs for a specific protocol implementation.

The **HDLC** implementation has two 4 Kbyte Dual Port RAM (DPR) blocks implemented using the Xilinx internal block RAM per port. Each DPR is configured to have a 32-bit port on the PCI side, and a 16-bit port on the I/O side. See Figure 3 for a representation of the HDLC circuit.

The HDLC interface uses a fixed 200 MHz clock as a reference frequency to sample the internal or external transmitter clock. One of the PLL inputs is selected for the Tx clock when using internal clock mode. Clock and data, in and out, comprise the four I/O lines of each port. The two DPRs are partitioned into one block each for transmit and receive. The RAM blocks are used as circular buffers that have independently specified start and stop addresses and separate transmit and receive interrupts. Programmable for active edge, standard or modified Poly and CRC processing.

NRZ-L is implemented in two ports. Each port has Transmit, and Receive capabilities. The Transmitter is supported with a Data FIFO (0x3FFF - 32 bit words), and a Packet FIFO with room for x3FF descriptors. The receive side has the same configuration.

Separate controls for Tx and Rx allow user selection of Clock Sense, Data Sense, MSB / LSB order, and interrupt enables. With the various options one can interface with NRZ or NRZL. In addition, the Tx side has a register to select the bit rate transmitted. The receiver auto detects the clock and does not require frequency selection. Both Transmitter and Receiver have programable values to control the time between transmissions or how much time to wait to detect the end of packet.

Packet descriptors are used by the transmitter to determine how many bits to send in one packet. The receiver generates packet descriptors to document the number of bits stored in a received packet.

Status is provided for Overrun and Underrun. Interrupts are available for transmission or reception of a packet. Programmable Almost Full [Rx Data FIFO] and Almost Empty [Tx Data FIFO] are provided. Additional counts for Data and Packet FIFOs, Full and Empty status plus state machine Idle condition provided.

Built in test is provided with R/W registers plus loop-back between the Data FIFOs.

Most of the data **I/O** lines are programmable to be register controlled or state-machine controlled. Any or all of the bits can be used as a parallel port instead of being dedicated to a specific I/O protocol. 16 differential I/O are provided at the rear IO connector. The drivers and receivers conform to the RS-485 specification (exceeds RS-422 specification). The RS-485 input signals are selectively terminated with 100Ω. The termination resistors are in separate packages to allow flexible termination options for custom formats and protocols.

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

Interrupts are supported by ccXMC-Serial. All interrupts can be individually masked, and a master interrupt enable is also provided to disable all interrupts simultaneously. The current interrupt status is available whether an individual interrupt is enabled or not making it possible to operate in polled mode.

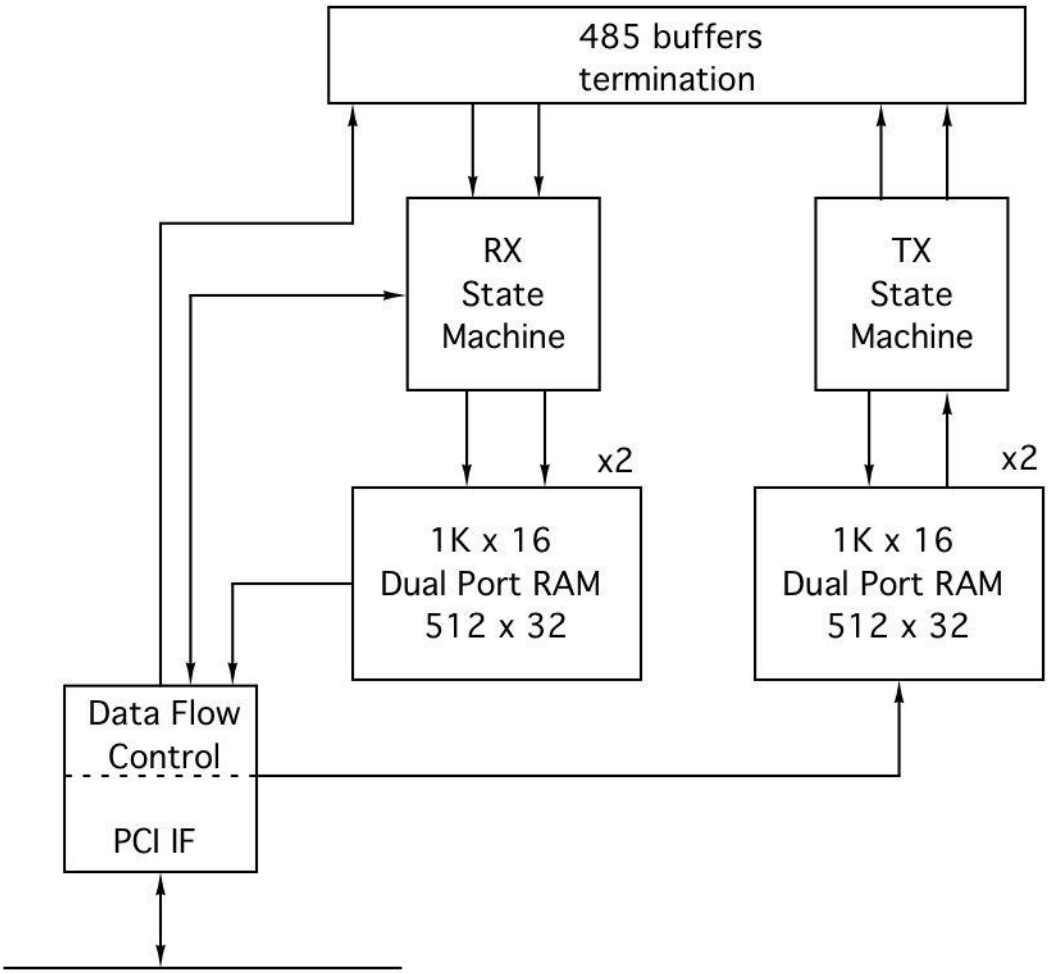


FIGURE 3 CCXMC-SERIAL HDLC BLOCK DIAGRAM

The following diagram shows the ccXMC-Serial UART configuration:

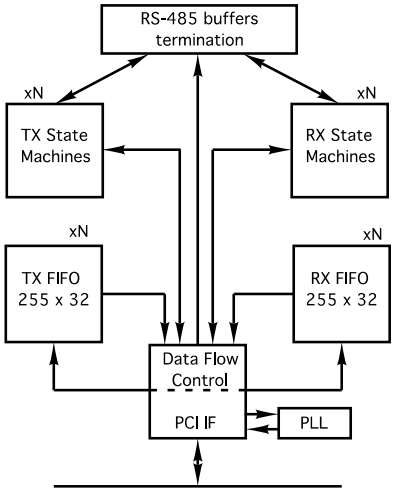


FIGURE 4 CCXMC-SERIAL UART BLOCK DIAGRAM

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

The UART protocol implemented provides RS422 data inputs and outputs. The transceivers have supporting programmable terminations to allow for in cable and on-board termination situations. The receivers are open cable safe – marking state is detected when undriven.

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

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

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

Transitions are detected and used to update the reference count. When transitions are not detected; the reference count and programmed baud rate [expected count] are used to determine when to capture bits. The receiver uses the programmed count to determine when to sample the data received. The transition detections are filtered to

only be applicable within 1/8th of the expected transition. The receiver can handle quite a lot of jitter in this manner. Depending on the data [number of transitions] up to +/- 1/8th bit period per bit cell (with a transition).

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

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

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

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

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

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

Depending on your system requirements, select the best choice of the 4 standard modes. Test mode is used to create errors and for system test and development.

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

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

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

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

More on byte alignment: Transmit bytes are read from byte positions 0->3 byte lane wise [7-0] first, [15-8] second, [23-16] third and [31-24] last and the bytes are transmitted in this order. For message byte-counts not divisible by four, the last long-word is read as described. Any unused bytes are considered padding with the next message starting with the next FIFO long-word. For example, with 7 bytes to send, a word of 4 bytes will be read, then the lower 3 bytes will be read and sent and the 8th byte will be dropped.

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

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

Theory of Operation

ccXMC-Serial features a Xilinx FPGA. The FPGA contains all of the registers and protocol controlling elements of the Serial design.

ccXMC-Serial can support many protocols. The initial implementation of ccXMC-Serial supports two ports of full-duplex HDLC, two full-duplex ports of NRZL, and 3 ports of UART.

SDLC/HDLC Description:
SDLC is a subset of HDLC. To interface with SDLC hardware make sure the message size is compliant with byte boundaries. This implementation generates and appends the CRC to the transmit message, and checks the received CRC for received messages. Some SDLC systems do not use the CRC.

HDLC is implemented as a synchronous interface with separate clock and data inputs and outputs. Each message is delimited by eight-bit flag characters. The beginning flag and the ending flag enclose the frame. Both beginning and ending flags have the binary format 01111110. Hardware automatically adds and strips the start and end flags.

The ending flag for one frame may serve as the beginning flag for the next frame. Alternatively, the ending zero of an ending flag may serve as the beginning zero of a beginning flag, thus forming the pattern '011111101111110'.

The transmitter may insert multiple flags between frames to maintain the active state of the link if a gap between message frames is required.

In order to avoid false flag detection from the data pattern, the HDLC interface uses zero insertion. If five consecutive ones appear anywhere in the data stream, a zero is inserted to avoid having six consecutive one bits. On the receive side, when five ones are received the sixth bit is monitored. If it is a zero, it is removed from the data stream, if it is a one then either a start/stop flag or an abort character (0xFE) has been detected.

Any ending flag may be followed by a frame, by another flag, or by an idle condition. The idle condition is signaled by a minimum of 15 consecutive one bits. As long as 1's are transmitted, the link remains in the idle state.

To send a message, write the message data to the transmit memory, specify the start and stop addresses and configuration control bits, then enable the transmitter. The state-machine will load the start address, send the beginning flag character and send the data sequentially LSB first until the end address is reached. The CRC is generated from the full message not including the start and end flags or the CRC itself. When the end address is reached the CRC is transmitted and then the ending flag.

The data written from the host is expected to be little endian with the bytes ordered 3210 corresponding to 31-0. This is consistent with most PCI and PCIe systems. For a big endian platform you may need to change the byte order written to compensate. The bytes are transmitted 0 1 2 3 with the lsb of each byte sent first.

For example: 04030201 written to memory will look like "01111110100000000100000011000000000100000CRC01111110" Data is valid

As soon as the beginning flag is sent, the transmitting status bit will be asserted. SW can poll this bit if desired. The ending address will be latched in the transmitter and new addresses can be written for the next message to be sent. This message will be sent as soon as the current message completes. If a new transmit starting address is not written, the transmitter will continue reading data with the next address after the stop address of the current frame. A new transmit end address must be written to trigger sending an additional message-frame.

If the TX Clear control bit is enabled, when no more message frames are left to transmit; automatically disable the transmitter and the TX interrupt will be asserted.

If the TX clear control bit is not enabled, the transmitter will remain enabled after the last message, and the TX interrupt will still be asserted. When multiple frames are being sent, the frame done interrupt will be asserted at the end of each message-frame.

The TX interrupt will only occur after the last frame and the transmitter will wait, pointing at the next address after the end address. If additional data has been or is later written to the DPR, a new message can be started by entering a new end address (and optionally a new start address). The transmit state-machine will start the new message and continue sending data until the new end address has been reached. If the end address of the message is less than the start address of the message, when the end of memory is reached the transmitter will wrap around ⇔ to the start of memory until the end address is reached.

To receive a message the receiver must be enabled. Only the starting address of the receive buffer is specified. Data will be stored sequentially in the next address after the starting address, incrementing until the closing flag is detected. This will latch a receiver done interrupt status and can cause an interrupt if enabled. Data is written into memory as the message is received. The start address is skipped with data starting at the start address + 1 word. When the message is completed the end address plus status is written to the starting address. Status includes CRC valid and number of bits in the last word.

This allows any received message to be quickly accessed in the received data by reading the address pointer in the message start location, which points to the end

address of the first message-frame. The memory location following the end of the first message-frame contains the end address of the second message-frame. This process can be repeated as many times as needed to find the message of interest.

At the end of each frame, the end address is also latched and can be read from the control register as a read-only field, but this will be overwritten as subsequent frames complete. The transmit interrupt is mapped to the first interrupt line of the selected port, the transmit frame done interrupt is mapped to the second interrupt line, the receive interrupt is mapped to the third interrupt line and the abort received interrupt is mapped to the fourth interrupt line of the selected port.

When a frame completes and no more message-frames are pending, the bus can stay active by continually sending flags or it can go idle by sending ones. The HDLC Idle After Frame Done control bit determines this behavior for the transmitter. If this bit is not set and the bus remains active by sending multiple flags, the Repeated Flags Share Zero control bit determines whether the transmitter sends a '0111111001111110' or a '01111110111110' pattern while waiting for a new message-frame to be requested. When the transmitter is disabled the bus defaults to a high state, which is equivalent to the idle condition.

The Rx Clock reference selected is also used to sample the the transmit reference clock to detect transitions. These transitions are used to determine when to drive the next data bit onto the transmit data I/O line. The transmitter clock reference can be supplied by an external source or an internal clock reference selected from 3 of the PLL outputs.

For test purposes, a substitute external clock is created by muxing a test clock output onto I/O configured as outputs. These clocks may be connected externally to any or all selected ports for loopback testing. A control bit in each port's control register is used to select between these two options. When the internal clock mode is selected the transmit clock line is configured as an output. When the external clock mode is selected the transmit clock line is configured as an input.

The transmit data line is always an output and the receive clock and data lines are always inputs. Due to IO limitations the Output clock for the NRZL function is used to provide the Tx reference when in external loop-back with the external clock source selected. With a normal system the clock will come from the internal source or the system and not require the NRZL clock as a reference.

Per the ISO/IEC specification the CRC is calculated on the data between the FLAGs not including the CRC itself. X16 + X12 + X5 + 1 is the form of the calculation. Since the data can be non-16 bit lengths the calculation is done with a shift register to allow partial bit shifting on the last word. The preset value is xffff. The final value is inverted [bitwise].

NRZL is a common interface. Clock and Data on a differential IO standard. Several programmable features are implemented to create an adaptable interface.

1. MSB or LSB first selection
2. Active Clock Edge selection – 50/50 data period with rising or falling edge centered for transmission
3. Standard or Inverted Data
4. Number of bits to transmit
5. Frequency of transmission
6. Auto Frequency Rx
7. Automatic programmable time between packets transmitted
8. Programmable end of packet detect for Rx
9. Interrupts and status to control operation
10. Full duplex support
11. Clk Idle option to transmit the clock without data
12. Burst clock option – send a specific number of bits per frame.

To transmit, the HW is programmed for the mode used in your system. Select MSB or LSB first transmission, Active edge of the transmitted clock, Standard or inverted data, and if you want to enable an interrupt request at the end of each transfer. These values are written to the Tx Control Register. If receiving, the same choices are available with the Rx Control Register. The choices do not need to match. If performing loop-back you will want to have them match.

If sending multiple packets with HW control of the timing program the Tx Gap register with the requested delay.

Program the Tx Rate register to set the clock period when transmitting.

Load data into the Data FIFO. Write a Descriptor to the Packet FIFO.

The last step is repeated for each new transmission. No need to repeat the others.

In addition, make sure the Parallel Port Mux and Termination registers are programmed to support the NRZL port operation.

For reception the operation is symmetrical with 2 key differences.
1. Data and Descriptor are read from the port.
2. Meaning of the Rx GAP. For the Tx GAP, if left programmed to 0x00 [reset default] the HW will ignore the gap since it is zero. For the Rx side the Gap time is based on the reference clock rather than the Tx 2X rate and is used to specify the time to wait to determine end of packet has been reached. Program to 2X the normal gap between words to determine a frame / packet is completed.

UART

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

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

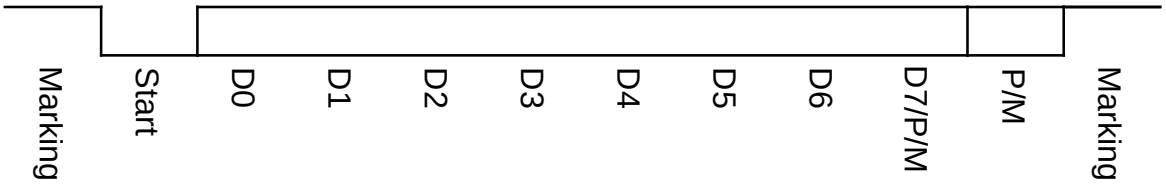


FIGURE 5 UART TRANSFER ENCODING

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

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

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

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

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

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

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

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

Address Map

BASE Map

Base	0x0000	Pointer to Base memory space	
BASE	0x0000	0	Base control register
BASE1	0x0004	1	Master Interrupt Enable
STATUS	0x0008	2	Interrupt and other Status
ID	0x000C	3	Switch, Revision register
IO_DATA	0x0010	4	Data register 31 - 0
IO_DIR	0x0014	5	Direction register 31 - 0
IO_TERM	0x0018	6	Termination register 31 - 0
IO_MUX	0x001C	7	Mux register 31 – 0
RES	0x0020	8	Spare
RES	0x0024	9	Spare
PLL_DATA	0x0028	10	PLL R/W port for FIFOs
PLL_STATUS	0x002C	11	PLL programming status
TEMP	0x0030	12	Temperature port
IO_RDBK	0x0034	13	External I/O read register

HDLC Map

Port 0	0x1000	Pointer to base address for Port 0	
Port 1	0x4000	Pointer to base address for Port 1	
HDLC_TX_MEM	0x0000	Dual-port TX RAM read/write port	
HDLC_RX_MEM	0x1000	Dual-port RX RAM read/write port	
HDLC_CNTL	0x2000	0	HDLC control port
HDLC_STAT	0x2000	0	HDLC control read-back with added status
HDLC_CNTLB	0x2004	1	HDLC Master Interrupt enable port
HDLC_STAT2	0x2008	2	HDLC Interrupt Status port
HDLC_RXCLKCNT	0x200C	3	HDLC Received Clock Count
HDLC_CRCC_0	0x2010	4	HDLC extended CRC control lower
HDLC_CRCC_1	0x2014	5	HDLC extended CRC control upper

NRZ-L Map

Port 2	0x7000	Pointer to base address for Port 2
Port 3	0x7080	Pointer to base address for Port 3
NRZL_CNTL	0x0000	NRZL Control port
NRZL_CNTLB	0x0004	NRZL Master Interrupt Port
NRZL_TXREG	0x0008	NRZL Tx Function Control Port
NRZL_RXREG	0x000C	NRZL Rx Function Control Port
NRZL_TXAMT	0x0010	NRZL Tx Almost Empty FIFO level
NRZL_RXAFL	0x0014	NRZL Rx Almost Full FIFO level
NRZL_TXCLK2X	0x0018	NRZL Tx transmit rate 2x reg
NRZL_FIFO	0x001C	NRZL FIFO port – wr to Tx, rd from Rx
NRZL_STAT	0x0020	NRZL status register [FIFO]
NRZL_STAT2	0x0024	NRZL Interrupt Status port
NRZL_RXCNTS	0x0028	NRZL Rx Pkt Cnt : Rx FIFO Cnt
NRZL_TXCNTS	0x002C	NRZL Tx Pkt Cnt: Tx FIFO Cnt
NRZL_PKT	0x0030	NRZL Packet FIFO wr to Tx, rd from Rx
NRZL_TXGAP	0x0034	NRZL Tx Gap Timer definition
NRZL_RXGAP	0x0038	NRZL Tx End of packet time definition
NRZL_BITSON	0x0040	NRZL Number of bits per frame to send
NRZL_BITSOFF	0x0044	NRZL Gap between groups of BitsOn

UART Port Address Map

Port 4	0x7100 Pointer to base address for Port 4
Port 5	0x7180 Pointer to base address for Port 5
Port 6	0x7200 Pointer to base address for Port 6
UART_CNTL	0x0000 //0 UART Port Control Bits R/W
UART_CNTLB	0x0004 //1 Expanded UART control bits & Tx Packet delay
UART_STAT	0x0008 //2 UART Port Status Bits Read /write to clear
UART_TX_FIFO_CNT	0x000C //3 UART Port TX Packet and Data FIFO's
UART_RX_FIFO_CNT	0x0010 //4 UART Port RX Packet and Data FIFO's
UART_TX_DMA_PTR	0x0014 //5 UART Port Write TX DMA Pointer Res/Unused
UART_RX_DMA_PTR	0x0018 //6 UART Port Write RX DMA Pointer Res/Unused
UART_RX_UART_FIFO	0x001C //7 UART Port Read from RX UART FIFO
UART_TX_UART_FIFO	0x001C //7 UART Port Write to TX UART FIFO
UART_TXFIFO_LVL	0x0020 //8 UART Port Tx Almost Empty 15-0 =
UART_RXFIFO_LVL	0x0024 //9 UART Port Rx Almost Full 15-0
UART_FRAME_TIME	0x0028 //10 UART Port End of Frame Time 23-0
UART_BAUD_RATE	0x002C //11 UART Port Frequency 15-0 = Tx, 31-16 = Rx
UART_TX_PKT_FIFO	0x0030 //12 UART Port Write to TX Packet FIFO
UART_RX_PKT_FIFO	0x0030 //12 UART Port Read from RX Packet FIFO
UART_TX_MODULUS	0x0034 //13 UART R/W Modulus definition Port
UART_X_CURRENT	0x0038 //14 UART RO Timer Current Count
	//15-19 Spare decodes per port

FIGURE 6 CCXMC-SERIAL INTERNAL ADDRESS MAP

The address map provided is for the local decoding performed within ccXMC-Serial. The addresses are all offsets from a base address, assigned by the system when the PCI bus is configured. The Base and Port offsets relative to the system assigned address are shown. The register and memory offsets are relative to the base and port offsets.

VendorId = 0xDCBA, CardId = 0x0078
Flash design ID = 0x0001

FLASH Revision:
1p0 : original release TBD
2p0 : updated with Clock Selector for each port 0-3 and added sub burst capability to NRZL function.

Programming

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

Depending on the software environment it may be necessary to set-up the system software with the ccXMC-Serial "registration" data.

For HDLC, In order to receive data the software is only required to initialize the receiver buffer start address and enable the Rx port. To transmit the software will need to load the message into the appropriate memory, set the transmitter buffer start and end address and any configuration parameters and enable the transmitter.

When a received message completes, the end address of the message will be written to the receiver buffer start address with the received data stored starting with the next address. The next message will be stored starting with the following address unless a new starting address has been written after the first message has begun. The end address of each received message can also be read from the address field of the channel control register, but this will be over-written when the next message completes.

Once the transmitter starts sending a message, a new end address (and optionally a new start address) can be written to send subsequent messages. Multiple messages can be loaded into the transmitter RAM and sent in any order desired.

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

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

HDLC is the primary function of Ports 0 and 1. If not in use the IO allocated to the ports can be used as a parallel port by selection in the Mux register. Similarly, the NRZ-L ports can be used for the primary function or swapped out for parallel port operation. The UART ports are also available for parallel port operation. Be sure to set the type

and pay attention to the directionality as those assignments are not as flexible as the 485/LVDS IO group.

Base Definitions

BASE

[$00] Base Control Register Port read/write

Base Control Register

DATA BIT	DESCRIPTION
31	TestClockSel
30	Spare
29-28	Nrz1Clk
27-26	Nrz0Clk
25-24	HdlcTx1Clk
23-22	HdlcRx1Clk
21-20	HdlcTx0Clk
19-18	HdlcRx0Clk
17	PLL Use Alternate ID
16	PLL Check ID
15	PLL Read Enable
14	PLL Reset
13	PLL Enable
12-5	reserved
4	ForceInt
3-0	reserved

FIGURE 7 BASE CONTROL REGISTER BIT MAP

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

PLL Enable: When this bit is set to a one, the PLL programmer module, used to program and read the PLL, is enabled. When this bit is zero, the PLL programmer is disabled.

PLL Reset: When this bit is set to a one, the PLL programmer will stop processing, if not stopped already, and return to its initial state. When this bit is zero, the PLL programmer is ready to accept control inputs.

PLL Read Enable: When this bit is set to a one and the PLL programmer is enabled, the programmer will perform a read of the PLL device internal registers. The 40 bytes of data obtained will be written into the PLL read FIFO as ten long-words. When this bit is zero and the PLL programmer is enabled, the programmer will write data into the PLL device or simply check for a response to the selected ID value depending on the PLL Check ID control bit.

PLL Check ID: When this bit is set to a one and the PLL programmer is enabled, the programmer will begin a write operation, but will stop after the device ID has been sent. If the ID was acknowledged successfully, the done status will be set and the error status will be cleared. If the ID was not acknowledged successfully, the done status will be cleared and the error status will be set. When this bit is zero and the PLL programmer is enabled, the PLL programmer will perform a write or read operation depending on the PLL Read Enable control bit.

PLL Use Alternate ID: When this bit is set to a one, the device ID sent will be the alternate ID: 0x6A. When this bit is zero, the normal ID: 0x69 will be sent to the PLL device.

TestClockSel when '1' selects having a test clock output on the IO. The Mux register and Direction registers will also need to be set to transmit and non-state-machine use to have the clock output. Approximately 5 MHz. For IO and termination testing. Not used in normal operation. When '0' standard operation with parallel port or programmed IO is applied to the IO based on the Mux settings.

ForceInt: When '1' this bit forces an interrupt request. This feature is useful for testing and software development. Note: requires the Master Interrupt Enable to be set [enabled].

HdlcRx0Clk, HdlcRx1Clk <= 00, 01, 10, 11 selects 48MHz, PLLA, PLLB, PLLC
HdlcTx0Clk, HdlcTx1Clk <= 00, 01, 10, 11 selects PLLA, PLLA, PLLB, PLLC
Nrz0Clk, Nrz1Clk <= 00, 01, 10, 11 selects PLLA, PLLA, PLLB, PLLC

PLLA is selected 2 times to provide full decoding, nothing is implied.
Use the selections to use the same or different references to each port. The 48MHz is internally derived. PLLA-D require programming. The driver initializes with PLLA = 1 MHz, PLLB = 100 MHz, PLLC = 100 MHz, and PLLD = 3.6864 MHz. Driver also selects 48 MHz for HDLC Rx ports, PLLA for HDLC TX ports, PLLB for NRZ0 and PLLC for NRZ1. PLLD is hardware selected to UART ports.

The user can change the .JED file and use the driver utilities to load the new file into the PLL.

BASE1

[$04] Base Control Expansion Port read/write

Base1 Control Register	
DATA BIT	**DESCRIPTION**
31-1	Spare
0	Interrupt Enable Master

FIGURE 8 BASE CONTROL REGISTER BIT MAP

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

Interrupt Enable Master: When '1' allows interrupts generated by ccXMC-Serial to be driven onto the carrier (INTA). When '0' the interrupts can be individually enabled and used for status without driving the backplane. Polled operation can be performed in this mode. Please note: additional enables may be present in each port.

STATUS

[$08] Status Port read only

Design Number / FLASH Revision

DATA BIT	DESCRIPTION
31	InterruptNoMask
30-15	reserved
14	PortInt6
13	PortInt5
12	PortInt4
11	PortInt3
10	PortInt2
9	PortInt1
8	PortInt0
7-5	reserved
4	ForceInt
3-1	reserved
0	InterruptMasked

FIGURE 9 DESIGN ID REGISTER BIT MAP

InterruptNoMask is set when a Port Interrupt or Force Interrupt is active. This bit can be used for polling if the Base Level Master Interrupt Enable is disabled.

InterruptMasked is set when any of the interrupt sources is active and the Master Interrupt Enable is also enabled.

ForceInt is set when the ForceInt bit in the Base Control register is set.

PortInt0 is set when Port 0 SDLC is requesting an interrupt.
PortInt1 is set when Port 1 SDLC is requesting an interrupt.
PortInt2 is set when Port 2 NRZL is requesting an interrupt.
PortInt3 is set when Port 3 NRZL is requesting an interrupt.
PortInt4 is set when Port 4 UART is requesting an interrupt.
PortInt5 is set when Port 5 UART is requesting an interrupt.
PortInt6 is set when Port 6 UART is requesting an interrupt.

With any of these bits the associated port Interrupt Status should be read to determine the cause of the interrupt.

With the Windows driver this port is read when the system detects an interrupt potentially from this device. Based on the Port and Force interrupt status the secondary

port interrupt or force interrupt routine handlers are launched.

ID

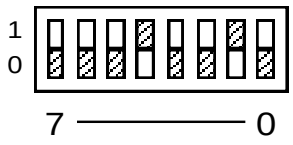
[$0C] Revision & Switch Port read only

User Switch Port

DATA BIT	DESCRIPTION
31-24	spare
23-16	FLASH Major Revision
15-8	FLASH Minor Revision
7-0	SW7-0

FIGURE 10 REVISION AND SWITCH PORT

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



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

The major and minor revisions are used to track the development and updates of the design.

Major.Minor – see revision table near Address Map.

IO_DATA

[$10] Parallel Data Output Register read/write

Parallel Data Output Register	
DATA BIT	**DESCRIPTION**
31-0	parallel output data

FIGURE 11 PARALLEL OUTPUT DATA BIT MAP

There are 16+ potential output bits in the parallel port. (31-16 unused) The Direction, Termination, and Mux Control registers are also involved. When the direction is set to output, and the Mux control set to parallel port the bit definitions from this register are driven onto the corresponding parallel port lines.

This port is direct read/write of the register. The I/O side is read-back from the IO_RDBK port. It is possible that the output data does not match the I/O data in the case of the Direction bits being set to input or the Mux control set to state-machine.

IO_DIR

[$14] Direction Port read/write

Direction Control Port	
DATA BIT	**DESCRIPTION**
31-0	Parallel Port Direction Control bits

FIGURE 12 DIRECTION CONTROL PORT

When set (‘1’) the corresponding bit in the parallel port is a transmitter. When cleared (‘0’) the corresponding bit is a receiver. The corresponding Mux control bits must also be configured for parallel port. Only applies to 485/LVDS bits. (31-16 unused)

IO_TERM

[$18] Termination Port read/write

Termination Control Port	
DATA BIT	**DESCRIPTION**
31-0	Parallel Port Termination Control bits

FIGURE 13 TERMINATION CONTROL PORT

When set (‘1’) the corresponding I/O line will be terminated. When cleared (‘0’) the corresponding I/O line is not terminated. These bits are independent of the Mux control definitions. When a bit is set to be terminated; the analog switch associated with that bit is closed to create a parallel termination of approximately 100 Ω. In most systems the receiving side is terminated, and the transmitting side is not. These bits are not controlled by the state machines as some systems terminate in the cable. Only applies to 485/LVDS bits. (31-16 unused)

IO_MUX

[$1C] Mux Port read/write

Multiplexor Control Port	
DATA BIT	**DESCRIPTION**
31-0	Parallel Port Mux Control bits

FIGURE 14 MUX CONTROL PORT

When set (‘1’) the corresponding bit is set to State-Machine control. When cleared (‘0’) the corresponding bit is set to parallel port operation. The Mux control definition along with the Data, Direction and Termination registers allows for a bit-by-bit selection of operation under software control. To use the defined ports [SDLC, NRZL] set to 0xFFFF. (31-16 unused with the UART ports direct connected to the top layer.

PLL_DATA

[0x28] PLL Output FIFO Write/ PLL Input FIFO Read

PLL Output/Input FIFO Ports	
Data Bit	**Description**
31-0	FIFO data word

FIGURE 15 PLL FIFO PORT

Writes to this port load PLL programming data into the PLL TX FIFO. This data is used to configure the PLL device. Reads from this port return data from the PLL RX FIFO. This data is the PLL device's internal register data that was read by the PLL programmer. Both FIFOs are 32 words deep and 32 bits wide.

The PLL is a separate device controlled by the Xilinx. The PLL has a fairly complex programming requirement which is simplified by using the Cypress® CyberClocks utility, and programming the resulting control words into the PLL using this PLL Control port. The interface can be further simplified by using the Dynamic Engineering driver to take care of the low-level bit manipulation requirements.

PLL_STATUS

[0x2C] Base Interrupt Status – (read only)

Base Interrupt Status Register

Data Bit	Description
31-11	Spare
10	PLL Error
9	PLL Done
8	PLL Ready
7	Spare
6	PLL Read FIFO Data Valid
5	PLL Read FIFO Full
4	PLL Read FIFO Empty
3	Spare
2	PLL Write FIFO Data Valid
1	PLL Write FIFO Full
0	PLL Write FIFO Empty

FIGURE 16 PLL STATUS PORT

PLL Write/Read FIFO Empty: When a one is read, it indicates that the corresponding FIFO contains no data; when a zero is read, there is at least one word in the FIFO. Although the FIFO is empty, there may still be one valid data word in the pipeline. The FIFO data valid bit indicates whether this is the case.

PLL Write/Read FIFO Full: When a one is read, it indicates that the corresponding FIFO is full; when a zero is read, there is room for at least one word in the FIFO.

PLL Write/Read FIFO Data Valid: When a one is read, there is valid data available; when a zero is read, there is no valid data available.

PLL Ready: When a one is read, the PLL programmer is idle and ready to accept a new command; when a zero is read, the programmer is actively sending data or reading data to/from the PLL device.

PLL Done: When a one is read, the programmer has successfully completed an input or output request; when a zero is read, this is not the case. This bit is latched and must be cleared by writing the PLL done bit back to this register.

PLL Error: When a one is read, an error occurred while processing a read or write request; when a zero is read, no error occurred. This bit is latched and must be cleared by writing the PLL error bit back to this register.

TEMP

[$30] Temperature Port

Temperature Port

DATA BIT	DESCRIPTION
31-24	Spare
23-8	Lm75WriteData
7-5	Lm75Pointer [x80 = PTR only]
4	Lm75Read
3-1	spare
0	Lm75Write

FIGURE 17 LM75 CONTROL

LM75B is a 400 KHz. I2C device. The 32 MHz reference is used to create an 80x reference to the controller. Write a null pointer to initialize [x81]. The Write and Read bits are auto cleared when the operation is complete. After the initial write completes do a dummy read of the data [x10]. Once the read bit is cleared repeat for data. See reference SW for an example. We use a flag to go through the initialization cycle once. Data is read back in the field shown. After shifting down the data is byte swapped to be in the proper order. Test the sign bit to see if a negative number.

0.125 C is the bit value.

IO_RDBK

[$34] Read-Back Port read only

I/O Read-Back Port

DATA BIT	DESCRIPTION
31-0	I/O Data 31-0

FIGURE 18 I/O READBACK PORT

The I/O lines can be read at any time. The value is not filtered in any way. If the transceivers are set to TX by the parallel port or state-machine the read-back value will be the transmitted value. If the transceivers are set to receive the port values will be those received by the transceivers from the external I/O. The upper 16 bits are set to 0.

HDLC_CNTL

[$2000] HDLC Control/Status Register

HDLC Control/Status Register

DATA BIT	DESCRIPTION
31	Idle Detected/Clear (see note after description)
30	Abort Detected/Clear (see note after description)
29	Reset Rx Clk Count
28-25	Bits – number of bits in last word to Tx
24	HDLC Internal Clock Select
23	Send an Abort (write only)
22	Load Transmit End Address (write only)
21	Load Transmit Start Address/HDLC Done
20	Load Receive Start Address/HDLC Sending Data
19	HDLC Idle After Frame Done
18-8	Address Input/ Receive End Address
7	Repeated Flags Share Zero
6	Received Abort Interrupt Enable
5	Receive Interrupt Enable
4	Transmit Frame Done Interrupt Enable
3	Transmit Interrupt Enable
2	Transmit Clear Enable
1	Receive Enable
0	Transmit Enable

FIGURE 19 HDLC CONTROL/STATUS REGISTER

Transmit Enable: When this bit is a one the transmitter is enabled to send data starting with the address stored in the transmitter start-address register and continuing until the data at the address in the transmitter end-address register has been sent. When this bit is a zero the transmitter is disabled.

Receive Enable: When this bit is a one the receiver is enabled to receive data and store it in the dual-port RAM starting with the address stored in the receiver start-address register if it is the first message since the receiver was enabled, or in the next 16-bit address after the end-address of the last message if it is not. When this bit is a zero the receiver is disabled.

Transmit Clear Enable: When this bit is a one the transmit enable bit will be cleared when the transmitted message completes and there is not another message pending. When this bit is a zero the transmitter will remain enabled, but no more data will be sent until a new end address is loaded.

Transmit Interrupt Enable: When this bit is a one the transmitter interrupt is enabled. The interrupt will occur at when the transmit state-machine reaches the end address stored in the transmitter end-address register and there is not another message pending. When this bit is a zero the interrupt status will still be latched, but will not cause an interrupt to occur. The transmit interrupt is mapped to the first interrupt line in its channel block.

Transmit Frame Done Interrupt Enable: When this bit is a one the transmit frame done interrupt is enabled. This interrupt will occur when each message frame completes regardless of whether another message is pending. When this bit is a zero the interrupt status will still be latched, but will not cause an interrupt to occur. The transmit frame done interrupt is mapped to the second interrupt line in its channel block.

Receive Interrupt Enable: When this bit is a one the receiver interrupt is enabled. The interrupt will occur at the end of a message transmission, which is determined by the detection of a HDLC flag character (0x7e) after the message has started. When this bit is a zero the interrupt status will still be latched, but will not cause an interrupt to occur. The receive interrupt is mapped to the third interrupt line in its channel block.

Received Abort Interrupt Enable: When this bit is a one, the received abort interrupt is enabled. This interrupt will occur when an HDLC abort character (0x7f) is received. When this bit is a zero the abort interrupt status will still be latched, but will not cause an interrupt to occur. The received abort interrupt is mapped to the fourth interrupt line in its channel block.

Repeated Flags Share Zero: When this bit is a one and the transmitter is sending repeated flag characters, the last zero in each flag will also serve as the first zero in the next flag. This is only true for two successive flags, the last flag before data is sent will be sent entirely. When this bit is a zero, all eight bits of each flag will be sent regardless of adjacent characters.

Address Input/Receive End Address: This field is used with the three load address bits to specify address boundaries for the transmitter and receiver data buffers. When this field is read, it represents the address in which the last received data word from the last message-frame is stored. Note that this is a 16-bit address, bit 0 indicates which half of the appropriate long-word the last 16-bit word was stored (0 -> lower half, 1 -> upper half).

HDLC Idle After Frame Done: When this bit is a one, the HDLC link will go to the idle state (minimum of 15 consecutive ones) when message transmission completes. The link will remain high until a new message is requested. When this bit is zero and the transmitter remains enabled, the transmitter will send repeated flags until a new message is requested.

Load Receive Start Address/HDLC Sending Data: When this bit is a one the value in the address input field is loaded into the receiver start-address register. When this bit is a zero no action is taken. When this bit is read as a one, the transmitter is actively sending data. At this time new addresses can be written for the next message-frame to be sent. A new transmitter end address is required to queue a new message-frame. New transmit or receive start addresses are optional. If new start addresses are not written, the transmitter and/or receiver will continue reading/storing data at the next address after the end address of the last message frame. When this bit is a zero, the link is either idle, aborted or sending repeated flags.

Load Transmit Start Address/HDLC Frame Done: When this bit is a one the value in the address input field is loaded into the transmitter start-address register. When this bit is a zero no action is taken. When this bit is read as a one, it indicates that the last message has completed. This bit is latched and will be cleared by any write to this control register. An interrupt can be configured to occur when this bit goes high by asserting the transmit frame done interrupt enable. When this bit is read as a zero, a message-frame has not completed since the last write to the HDLC control register.

Load Transmit End Address (write only): When this bit is a one the value in the address input field is loaded into the transmitter end-address register. When this bit is a zero no action is taken. This bit also loads the BITs field which specifies how many bits to send out of the last word. “0” = 16 bits, 1-F correspond to 1-15 bits sent.

Send an Abort (write only): When this bit is set to a one the transmit state-machine will send an abort character (0xfe) provided a transmission is currently in progress. When this bit is a zero normal operation will continue.

HDLC Internal Clock Select: When this bit is a one, the transmitter will use the internal transmit clock as a reference for sending HDLC data. When this bit is a zero, an external received clock will be used as the reference for data transmission. *If using external clock with loop-back be sure to connect the reference clock IO and set the control Mux to enable the reference clock outputs.*

Abort Detected/Clear: When an abort character is detected by the receiver, this status bit will be latched and can be cleared by writing a one back in this bit position. When this bit is a zero, no abort has been detected since the latch was last cleared.

Idle Detected/Clear: When an idle bus state is detected by the receiver, this status bit will be latched and can be cleared by writing a one back in this bit position. When this bit is a zero, the bus has not idled since the latch was last cleared.

Note: Writing Abort Clear or Idle Clear disables updating the remaining bits. Previously programmed values are retained. Clearing the Latched bits must be done separately.

Tx Bits: This field when set to 0 selects all 16 bits in the last word to be transmitted. Other values, 8-F specify the number of bits to send. For example, x8 will send the lower byte of the last word. Bits are sent LSB first.

Reset Rx Clk Count when set '1' will reset the counter checking the Rx Clk frequency. The bit is AND with the enable – not stored - no need reset the bit '0' after use.

HDLC_CNTLB

[$2004] Control Register Expansion

HDLC Master Interrupt Control

DATA BIT	DESCRIPTION
31-2	Spare
1	ForceInt
0	Master Interrupt Enable

FIGURE 20 HDLC CONTROL EXPANSION REGISTER

Master Interrupt Enable when set gates the interrupts from the associated SDLC port to the device level [base] interrupt processing. When '0' the status bits can be used to poll and no interrupt will be generated. Please note: the individual interrupt enables also need to be set.

ForceInt when set will cause an interrupt request to be generated. This bit is masked by the Master Interrupt Enable for the port and device.

HDLC_STAT2

[$2008] SDLC Interrupt Status Register

Interrupt Status and Clear Register

DATA BIT	DESCRIPTION
31	InterruptActive
30-5	Spare
4	ForceInt
3	Received Abort Interrupt Latched
2	Rx Interrupt Latched
1	Transmit Frame Done Latched
0	Tx Interrupt Latched

FIGURE 21 HDLC INTERRUPT STATUS REGISTER

Please refer to the Control register definitions for the Latched interrupts. ForceInt is set when set in the HDLC_CNTLB. Latched bits are cleared by writing to this port with the corresponding bit set.

HDLC_RXCLKCNT

[$0200C] Rx Clk Count Port

HDLC Rx Count Port

DATA BIT	DESCRIPTION
31-0	Count

FIGURE 22 HDLC RX CLK CNT

The Rx clock counts are captured by counting the received clock and counting once per period. The Host clock is used to sample and generate pulses to count. The intended purpose is to measure the frequency based on the number of counts per unit time. The counters can be cleared by setting the associated reset in the Port control register.

HDLC_CRCC

[$02010, $2014] CRC Extended Control Port

HDLC CRC Extended Control Port

DATA BIT	DESCRIPTION
63	RisingFalling
62-49	Spare
48	Invert Final Value
47-32	CRC Initial Value
31-16	CRC Check Value
15-0	Poly Divisor

FIGURE 23 HDLC CRC EXTENDED CONTROL PORT

Two 32 bit registers create this port. The values in the port default to the ISO/IEC standard method of calculating and checking the CRC.

x16 + x12 + x5 + 1. 0x1020 in the Poly Divisor field yields this algorithm. D5 set corresponds to x5 and D12 set corresponds to x12. Set any bit to include that bit in the calculation. xn <= xn-1 XOR feedback when selected and Xn <= Xn-1 when not selected. Feedback is the XOR of the CRC current MSB and Data being processed MSB. The “1” and x16 are always included. Affects both transmitter and receiver.

0x1d0f is the default value for the Rx side calculation. The received data has the CRC calculated including the CRC value received. If 0x1d0f is the remainder the CRC matches the received data. If the algorithm is changed this target value may need to change. If not changed the status for the message may show CRC error incorrectly.

0xFFFF is the default initialization value for the CRC calculation. You can program other values based on your specific interface.

Invert final value defaults to “yes” and the CRC is bitwise inverted prior to transmitting.

RisingFalling when True causes the transmission to change on the falling edge and be stable on the Rising edge. When False, the data changes on the rising edge and is stable on the falling edge. Affects both transmitter and receiver.

HDLC_MEM

Each HDLC port has 2 Dual Port RAM implementations. The host side of the memory is 32 bits wide and 1K deep – 4K bytes. The HDLC base address is the starting address for the TX memory. The Rx memory is the same size and starts offset by the 4K bytes x1000.

The Tx HDLC function reads from the internal port of the TX memory. The Rx HDLC function writes to the RX internal port.

The memory is accessible via target read and write operations from both ports. It is permissible to write and then read from the Tx memory or Rx memory spaces. The ATP uses this feature as part of testing the hardware.

See the Description for HDLC operation for a discussion of how the memory operates and interacts with the HDLC function.

See the HDLC base register to set the TX and RX operational addresses. The same field is used for both with control bits selecting which address is being loaded.

The HDLC transfer is in terms of 16-bit words. In cases of non-LW aligned data lengths the Start and End addresses are encoded with the last word to send. The programmed addresses are word addresses. For example, x7FE would be the full memory of words. For non-16 bit data lengths the BITs field is programmed to encode the number of bits in the last word to send. 0x00 = 16, 8=8 … xF= 15

For received data the memory structure is larger with the Start location filled with the message status. The data follows and finally the CRC received. 2 locations longer than the data size. Locations are based on words. The last data position may have fewer than 16 bits. The received data is zero extended [padded]. The number of bits received is part of the status.

Status = CRC Status, BITs, Address. The BITs field has 4 bit positions. The CRC Status will be set when the received CRC and the calculated CRC match. The received CRC is processed along with the message and the resultant should be a constant. If x1D0F is the resultant CRC the value is deemed correct. The Address field is 11 bits and is the end address of the received message = CRC value received.

HDLC Memory Configuration

Address	DESCRIPTION
0	Status
1-(N-1)	Data
N	CRC
Bit Definition	Status
15	CRC Good
14-12	Bits received [last word]
11-0	End Address (N)

note: bit positions shown after alignment – 32 bit data read from interface and aligned to 16 bit boundaries

FIGURE 24 HDLC RECEIVE MEMORY CONFIGURATION

The driver supports multiple reads and writes by passing a structure with the offset, array and length to either write or read. The driver also supports single word accesses. See the driver manual for more information.

NRZL_CNTL

[$0000] Control Register

NRZL Control Register

DATA BIT	DESCRIPTION
31-4	Spare
3	Reset Rx Clk Count
2	Reset Tx Clk Count
1	FIFO Loopback
0	NRZL Reset

FIGURE 25 NRZL CONTROL REGISTER

Set NRZL Reset to reset the FIFOs [Data and Packet] plus reset the state-machines. Control registers are not reset. Clear for normal operation.

Set FIFO Loopback to cause data in the TX data FIFO to loop to the RX data FIFO. Flow control is employed. TX and RX state-machines should be disabled to prevent conflicts when using this mode. Clear for normal operation. Data transferred to the RX FIFO is no longer available to transmit.

Reset Rx Clk Count and Reset Tx Clk Count bits are set to clear the counters accumulating the Rx received clock and Tx transmitted clock respectively. This bit requires clearing after use to enable the counters.

NRZL_CNTLB

[$0004] Control Register Expansion

NRZL Master Interrupt Control

DATA BIT	DESCRIPTION
31-2	Spare
1	ForceInt
0	Master Interrupt Enable

FIGURE 26 NRZL CONTROL EXPANSION REGISTER

Master Interrupt Enable when set gates the interrupts from the associated NRZL port to the device level [base] interrupt processing. When '0' the status bits can be used to poll and no interrupt will be generated. Please note: the individual interrupt enables also need to be set.

ForceInt when set will cause an interrupt request to be generated. This bit is masked by the Master Interrupt Enable for the port and device.

NRZL_TXCNTL

[$0008] Tx Control Register

NRZL Tx Specific Control

DATA BIT	DESCRIPTION
31-6	Spare
5	Tx Idle Clock Mode
4	Tx Interrupt Enable
3	Tx Clk Inv
2	Tx Data Inv
1	Tx MsbLsb
0	Tx Enable

FIGURE 27 NRZL TX CONTROL REGISTER

Tx Enable when set ‘1’ enables the transmitter to send data. The state machine will wait for the Data FIFO to be not empty and for the Tx Packet FIFO to have a descriptor before sending data. Set to ‘0’ when using FIFO loop-back mode.

Tx MsbLsb determines the order of the data transmitted. For MSB first operation set this bit. For LSB first leave cleared. See Data and Packet FIFO definitions for more on the order of the bits – how to load and how to define the length.

Tx Data Inv when set inverts the stored data bit by bit for transmission. When cleared data is transmitted as stored [not inverted].

Tx Clk Inv when set changes the sense of the clock to be falling edge stable. When Cleared the rising edge is used by the receiver. Transmitted data is very close to 50-50 duty cycle. Change on the falling edge, stable on the rising or the opposite based on this bit.

Tx Interrupt Enable when set gates the latched status for the end of transmission to be gated through to the master interrupt enable stage of interrupt generation. When cleared the status can still be read for polling but does not generate an interrupt request. Set each time a packet is completed [last bit sent].

Tx Idle Clock Mode is normally set to ‘0’ to have a burst clock along with data transmitted. If set ‘1’ the clock will be transmitted without data – Tx En should be off. If the BitsOn and BitsOff parameters are non-zero the clock can be sent for On periods and disabled for Off periods. See the On and Off registers for additional information.

NRZL_RXCNTL

[$000C] Rx Control Register

NRZL Rx Specific Control	
DATA BIT	**DESCRIPTION**
31-5	Spare
4	Rx Interrupt Enable
3	Rx Clk Inv
2	Rx Data Inv
1	Rx MsbLsb
0	Rx Enable

FIGURE 28 NRZL RX CONTROL REGISTER

Rx Enable when set ‘1’ enables the receive to capture data. The state machine will wait edges to be detected, capture bits and store words. When a packet is completed based on the programmed gap time the current descriptor is stored. Set to ‘0’ when using FIFO loop-back mode.

Rx MsbLsb determines the order of the data received. For MSB first operation set this bit. For LSB first leave cleared. See Data and Packet FIFO definitions for more on the order of the bits – how to read and interprete.

Rx Data Inv when set inverts the received data bit by bit. When cleared data is stored as received [not inverted].

Rx Clk Inv when set changes the sense of the clock to be falling edge stable. When cleared the rising edge is used by the receiver. A reference clock based sampling state-machine is used to detect edges and store the received bits. This bit selects the edge the detector is searching for.

Rx Interrupt Enable when set gates the latched status for the end of reception to be gated through to the master interrupt enable stage of interrupt generation. When cleared the status can still be read for polling but does not generate an interrupt request. Set each time a packet is completed [last word stored].

NRZL_TXAMT

[$0010] Tx Almost Empty

NRZL Tx Almost Empty	
DATA BIT	**DESCRIPTION**
31-16	Spare
15-0	Tx Almost Empty

FIGURE 29 NRZL TX ALMOST EMPTY

Tx Almost Empty sets the count for the Tx Data FIFO Almost Empty comparison. The count is used as a “less than” comparison. If set to x10 when the FIFO is x0F and lower the bit will be set. The status is in the Status Register. Full width register with 16 bits assigned to the Almost Empty function to match the depth of the FIFO.

NRZL_RXAFL

[$0014] Rx Almost Full

NRZL Rx Almost Full	
DATA BIT	**DESCRIPTION**
31-16	Spare
15-0	Rx Almost Full

FIGURE 30 NRZL RX ALMOST FULL

Rx Almost Full sets the count for the Rx Data FIFO Almost Full comparison. The count is used as a “Greater than” comparison. If set to x400 when the FIFO is x401 and higher the bit will be set. The status is in the Status Register. Full width register with 16 bits assigned to the Almost Full function to match the depth of the FIFO.

NRZL_TXCLK2X

[$0018] Tx Clock Rate x2

NRZL Tx Clock Rate	
DATA BIT	**DESCRIPTION**
31-16	Spare
15-0	Tx Clock Rate x2, Program with N

FIGURE 31 NRZL TX CLOCK RATE

TX Clock Rate is used to set the state machine rate for the transmitter. It is set to 2x the desired clock rate – for example setting to 10 MHz provides 5 MHz clock at the transmitter. The PLLC rate is counted from 0 to N in a free running manner. A pulse is generated for each loop. The pulse is used to control shifting, loading, and clock output generation. Ref Freq / N+1 = 2x the desired frequency. With a 100 MHz reference on PLLC and a divisor of 9 the reference pulse rate will be 10 MHz and the output clock rate 5 MHz. Output clock is burst mode – only present when data is being transferred.

A second example. 9.6 MHz Tx desired. 19.2 MHz reference required.

NRZL_FIFO

[$001C] Data FIFO

NRZL FIFO	
DATA BIT	**DESCRIPTION**
31-0	FIFO DATA

FIGURE 32 NRZL DATA FIFO

Writing to the Data FIFO address stores data to transmit. Reading from the address retrieves data stored from reception.

Transmitted and Received data are both stored based on the packet descriptor. In the Tx case, the host provides the descriptor. In the Rx case the HW generates the descriptor and stores into the Rx Packet FIFO to let the host know how much data to read.

Descriptors are bit lengths. Data is stored LWs first Remainder last. The remainder is

LSB aligned for both Tx and Rx whether LSB first or MSB first operation, The HW automatically picks off the apparent MSB for the remainder and sends the last bits starting from there[MSB mode]. The receiver automatically shifts the last word down to be LSB aligned [LSB mode].

LW
LW
LW
Remainder LSB aligned

Any number of bits can be sent within the bit count provided by the descriptor. See the packet FIFO discussion for more on this topic.

The Tx and Rx FIFOs are separate and 16K-1 x 32 bits each.

NRZL_STATUS

[$0020] NRZL STATUS Register

NRZL Status Register

DATA BIT	DESCRIPTION
31-14	'0'
13	RX SM IDLE
12	TX SM IDLE
11	RX PKT FIFO FULL
10	RX PKT FIFO MT
9	TX PKT FIFO FULL
8	TX PKT FIFO MT
7	'0'
6	RX FIFO FULL
5	RX FIFO AFL
4	RX FIFO MT
3	'0'
2	TX FIFO FULL
1	TX FIFO AMT
0	TX FIFO MT

FIGURE 33 NRZL DATA FIFO

Bits marked '0' will return '0' when read. These bits are spare for future additions.

TX FIFO MT = '1' when the TX Data FIFO is Empty. Otherwise '0'.
TX FIFO AMT = '1' when the TX Data FIFO count is below the programmed TX AMT

count. Otherwise ‘0’.
TX FIFO FULL = ‘1’ when the TX Data FIFO is FULL. Otherwise ‘0’.

RX FIFO MT = ‘1’ when the RX Data FIFO is Empty. Otherwise ‘0’.
RX FIFO AFL = ‘1’ when the RX Data FIFO count is above the programmed RX AFL count. Otherwise ‘0’.
RX FIFO FULL = ‘1’ when the RX Data FIFO is FULL. Otherwise ‘0’.

TX PKT FIFO MT = ‘1’ when the TX Packet FIFO is Empty. Otherwise ‘0’.
TX PKT FIFO FULL = ‘1’ when the TX Packet FIFO is FULL. Otherwise ‘0’.

RX PKT FIFO MT = ‘1’ when the RX Packet FIFO is Empty. Otherwise ‘0’.
RX PKT FIFO FULL = ‘1’ when the RX Packet FIFO is FULL. Otherwise ‘0’.

TX and RX SM IDLE bits are set when the respective state machines are in the IDLE state. Can be polled to know when a disabled SM is back to the idle state for further processing. Alternatively, disable and then reset the port to force back to the idle state.

NRZL_STAT2

[$0024] NRZL Interrupt Status Register

Interrupt Status and Clear Register

DATA BIT	DESCRIPTION
31	InterruptActive
30-6	Spare
5	ForceInt
4	Rx Packet Over Flow Latched
3	Rx Data Over Flow Latched
2	Tx Under Run Latched
1	Rx Interrupt Latched
0	Tx Interrupt Latched

FIGURE 34 NRZL INTERRUPT STATUS REGISTER

Latched bits are cleared by writing to this port with the corresponding bit set.

Tx Interrupt is set each time the Tx state-machine finishes processing a descriptor. The bit is latched.

Rx Interrupt is set after loading the descriptor for a reception. The end of the packet is

determined by the lack of a clock transition within the programmed Rx Gap time.

Tx Under Run is set if the transmitter does not have enough data to complete a descriptor. Data can be loaded on the fly – however it must be in the FIFO when time to read that word to send.

Rx Data Over Flow is set if the Rx Data FIFO is full when it is time to write more data.

Rx Packet Over Flow is set if there is no room to write the new descriptor when ready to load.

The UnderRun and OverFlow error bits should be monitored and corrected if they occur. If under running – preload more data before starting [loading the descriptor]. If over flowing read more often or make use of the burst read utility provided in the SW package.

Force Int is set when the Force Int control bit is set to allow a single read to determine the cause(s) of an interrupt from the port. Clear this bit in the control register.

NRZL_RXCNTS

[$0028] Rx FIFO Counts

NRZL Rx FIFO Counts

DATA BIT	DESCRIPTION
31-16	Rx Packet FIFO Count
15-0	Rx Data FIFO Count

FIGURE 35 NRZL RX FIFO COUNTS

Rx Data FIFO Count field returns the number of LW in the Rx data FIFO. Use associated Packet FIFO Descriptor to crack the packed data into complete LW and remainder for non-longword aligned transfers. 0x3FFF is the current max count.

Rx Packet FIFO Count is the number of descriptors stored into the Rx Packet FIFO. 0x3FF is the current max count.

NRZL_TXCNTS

[$002C] Tx FIFO Counts

NRZL Tx FIFO Counts

DATA BIT	DESCRIPTION
31-16	Tx Packet FIFO Count
15-0	Tx Data FIFO Count

FIGURE 36 NRZL TX FIFO COUNTS

Tx Data FIFO Count field returns the number of LW in the Tx data FIFO. 0x3FFF is the current max count.

Tx Packet FIFO Count is the number of descriptors stored into the Tx Packet FIFO. 0x3FF is the current max count.

NRZL_PKT

[$0030] Packet FIFO

NRZL Packet FIFO

DATA BIT	DESCRIPTION
31-29	"0"
28-5	Descriptor LW count
4-0	Descriptor Remainder

FIGURE 37 NRZL PACKET FIFO

Writing to the Packet FIFO address stores the Descriptor to program the length to transmit. Reading from the address retrieves the descriptor for the data stored in the Data FIFO.

If the transmitter is enabled and data is stored the act of writing the descriptor will also launch the transmit state-machine.

Descriptors are bit lengths. The D4-0 represent the remainder – the part of the message sent that is not on a LW boundary. 0x20 would be 1 LW or 32 bits sent. 1 LW loaded into the Data FIFO.

LW D31-0

For LSB or MSB data shorter than 1 LW the descriptor would be 0x1F – 0x1. The data will be LSB aligned. The upper portion is not sent and can be anything. Padding with '0' may provide some benefits when tracing.

For more than 1 LW in length the data is stored with at least 2 LW with the complete LW sent first and the remainder sent last.
LW
…
LW
Remainder LSB aligned

The Tx and Rx Packet FIFOs are separate and 1K-1 x 32 bits each.

Example: 3 LW loaded, LSB first
76543210
FEDCBA98
13121110

The Descriptor would be 3 x 32 bits = x60
Data would be transmitted ‘b 0000 1000 0100 1100 0010 1010 0110 1110…
Gaps are for ease of reading and not present in the data stream

Example: 16 bits to send MSB

0x0000ABCD loaded into Data FIFO

Descriptor = 16 bits = 0x10
Data transmitted 1010 1011 1100 1101

Gaps are for ease of reading and not present in the data stream

Reading a Descriptor tells the host what to do with the data. Divide the descriptor by 32 and read the integer value in LW. The remainder is the portion of the next LW with valid data. Alternatively, the Host can read the FIFO count and move data to host memory using the descriptors to break into separate messages after. With multiple small packets this can be a more efficient method.

Note: Data in the Data FIFO is separated by packet. Messages are always read [Tx] stored [Rx] starting on a LW boundary [bit wise].

NRZL_TXGAP

[$0034] Tx Gap Register

NRZL Tx Gap Register

DATA BIT	DESCRIPTION
31-24	“0”
23-0	Programmed time between packets

FIGURE 38 NRZL TX GAP

The Tx Gap register is used to program the time between packets sent. If 0x00 the state-machine skips the gap time. The time is counted by the clock enables from the 2X clock. If programmed to 10 MHz as in the previous example the delay would be 100 nS per count.

The full register is present and can be read back. This register can be changed at any time as the value is synchronized to the reference clock [PLLC] before comparing against the local count. If the count is decreased during while a gap is being counted

the counter may have already gone past the count and have to loop around 1 time before the following gaps are operating with the programmed time. Best to change when Tx is in the Idle state.

The Tx GAP time is best used when multiple packets of data are loaded and multiple packet descriptors are to be loaded with the Tx GAP time between them.

NRZL_RXGAP

[$0038] Rx Gap Register

NRZL Rx Gap Register

DATA BIT	DESCRIPTION
31-24	"0"
23-0	Programmed Packet Timeout

FIGURE 39 NRZL RX GAP

The Rx Gap register is used to program the time to wait before declaring end of packet. The active edge of the received clock is used to capture data and to reset a timer. If the time gets to the programmed time "end of Packet" is declared. The bits shifted into position [if LSB first] and the remainder if any loaded [if any] into the Data FIFO.

The timer is using the PLLC reference – 100 MHz or the programmed rate if changed. If using a 5 MHz data stream the period of the clock received is 200 nS. The gap timer has an infinite wait for the 1st bit so it does not timeout before any data comes. Once the clocks are received the timer runs being reset with each bit. The time should be longer than the period of the expected data and shorter than the time between packets if multiple packets are expected. The time will also delay writing the descriptor allowing the host to process the stored data. Something like 2x the expected period allows some jitter on the received clock and not an excessive amount of added time before completing the packet.

Examples of using the timers are located in the reference SW available for this design.

NRZL_BITSON

[$0040] BitsOn Control Register

NRZL BitsOn	
DATA BIT	**DESCRIPTION**
31-0	Bits On

FIGURE 40 NRZL BITS ON REGISTER

NRZL_BITSOFF

[$0044] BitsOff Control Register

NRZL BitsOff	
DATA BIT	**DESCRIPTION**
31-0	Bits Off

FIGURE 41 NRZL BITS OFF REGISTER

If Bits On is set to a value other than 0x00 the clock pulse generator used to run the NRZ Tx state-machine switches modes to create clock enables in groups defined by the Bits on and Bits Off parameters. To create a square wave the Tx reference frequency is set to 2X the desired frequency and the pulses are sent 2 per bit. These registers are per pulse. The HW advances based on the count – to keep complete bits being transmitted even numbers should be used. 32 to send 16 bits for example [BitsOn]. Bits Off is the “time” in 2x bits between groups of BitsOn.

The output of the pulse generator is used to create the clock when the Idle Clk Mode is selected. The Clock pulses will be grouped based on the BitsOn and BitsOff parameters and have the intended ½ programmed frequency rate.

For example, program the reference rate of the PLL to 100 MHz, program the local divider for 10 MHz, and select Idle Clk Mode with Bits On = 0x00. The output on the clock signals will be a continuous 5 MHz clock. Change the BitsOn to 32 and BitsOff to 32. Now the output clock pulses will still be 5 MHz but with 16 clocks transmitted and 16 clocks not transmitted – for an effective rate of 2.5 MHz. If the Idle Clk Mode is not selected, and data is available to transmit along with packet descriptor and the enable to the Tx port the data will be transmitted with the BitsOn and BitsOff programmed

timing as a sort of modulation. Send N bits, wait for M bits, repeat until out of data or packet descriptors.

NRZL_TXCLKCNT

[$0048] Tx Clk Count Port

NRZL Tx Count Port	
DATA BIT	**DESCRIPTION**
31-0	Count

FIGURE 42 NRZL TX CLK CNT

NRZL_RXCLKCNT

[$004C] Rx Clk Count Port

NRZL Rx Count Port	
DATA BIT	**DESCRIPTION**
31-0	Count

FIGURE 43 NRZL RX CLK CNT

The Rx and Tx clock counts are captured by counting the transmitted or received clock and counting once per period. The reference clock selected is used to sample and generate pulses to count. The counter output is re-registered using the host clock reference. There may be some jitter to the counting as a result. The intended purpose is to measure the frequency based on the number of counts per unit time. The counters can be cleared by setting the associated reset in the Port control register.

UART Definitions

UART_CONT

UART CONTROL

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

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

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

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

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

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

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

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

FIGURE 44 UART CONTROL

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

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

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

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

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

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

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

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

DmaRdIEn/ DmaWrIEn DMA Interrupt Enable: These two bits, when set to one, enable the interrupts for DMA write and read completion for the referenced channel. These two

interrupts cannot be disabled by the master interrupt enable.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

RxMode 2:0
RxOneByte (RxMode “001”) when selected causes the received data to be loaded one byte per LW in the Rx Data FIFO.

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

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

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

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

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

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

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

UART_CONTB

UART CONTROL B

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

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

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

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

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

#define   HalfFullDuplex    0x01000000 //24 0 = Full Duplex, 1 = Half Duplex
#define   ForceRTS          0x02000000 //25 0 = normal, 1 = Force RTS to block
#define   InvertFlowCntl    0x04000000 //26 0 = normal, 1 = invert RTS/CTS
#define   UseCTS            0x08000000 //27 0 = ignore CTS, 1 = Use Flow Control

#define   Slew              0x10000000 //28 1 = limit Slew Rate, 0 = unlimited
#define   485/232           0x20000000 //29 1 = RS-485, 0 = RS-232
#define   UART Enable       0x40000000 //30 0 = Disabled, 1 = Enabled
#define   ReferenceSel      0x80000000 //31 0 = 32 MHz, 1 = PLL Reference
```

FIGURE 45 UART CHANB CONTROL

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

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

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

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

DirTx when set enables the external and internal buffers to transmit. Normally set to ‘1’. When set to ‘0’ the line level will tristate. Use when in Half Duplex mode to enable transmitting when the line is open. Otherwise set to ‘1’.

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

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

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

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

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

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

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

Programming note: When TxTimer32 is selected to start transmission, TxEnable is set at the end of the programmed countdown. If not masked by the FIFO status, TxEnable

can be set without data present leading to immeditate transmission of data when data is loaded. If the Timing strobe is required for the start of a burst of data the Mask should be used to make sure data is only transmitted immediately after the strobe from the timer function.

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

HalfFullDuplex when set '1' causes the UART buffer to operate in Half Duplex mode. '0' indicates Full Duplex mode.

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

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

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

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

About RTS and CTS HW implementation. RTS when in non-forced normal mode is asserted low to allow data to transfer anytime the data level in the RX FIFO has 16 bytes or more. In unpacked mode this is 16 positions, in packed mode it is 4 since each LW has 4 bytes – HW automatically adjusts based on the mode selected. Most HW can stop transmitting within 12 bytes of RTS deassertion. ccXMC-Serial-UART transmit function when CTS transitions to disabled will complete the current data word and then stop. This means 1-2 bytes in unpacked mode and up to 4 in packed, packetized and alternate packetized modes depending on when CTS changes state.

SLEW – set to limit to 250 KHz. Cleared for full bandwidth operation. Note: for baud rates near 250 and above use the full bandwidth mode.

485/232 – set for differential operation with RS-485 IO. Clear for single ended RS-232 operation. Affects TX, RX, RTS and CTS.

UART Enable – Set to '1' to enable the external transceiver. '0' when not in use.

ReferenceSel when '0' selects the 32 MHz oscillator for the UART clock reference. When set '1' the PLL associated with the port is used instead.
PLL Clock D is the alternate reference for the UART ports

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

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

UART_STAT

UART STATUS

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

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

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

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

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

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

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

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

FIGURE 46 UART STATUS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Write/Read DMA Error: When a one is read, it indicates that an error has occurred while the corresponding DMA was in progress. This could be a target or master abort or an

incorrect direction bit in one of the DMA descriptors. These bits are latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no DMA error has occurred. NA this design

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

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

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

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

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

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

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

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

LocalInt when set indicates one of the non DMA interrupt requests is active. This is after the individual interrupt masks and before the channel master interrupt enable.

IntStatus when set indicates this channel has a pending interrupt request. DMA and

local Interrupts [after the master enable].

TX_FIFO_CNT

TX FIFO Counts		
#define	CHAN_PKT_CNT_MASK_TX	00FF0000 //
#define	CHAN_DATA_CNT_MASK_TX	0000FFFF //

FIGURE 47 TX FIFO COUNTS

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

RX_FIFO_CNT

RX FIFO Counts		
#define	CHAN_PKT_CNT_MASK_RX	00FF0000 //
#define	CHAN_DATA_CNT_MASK_RX	0000FFFF //

FIGURE 48 RX FIFO COUNTS

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

UART_FIFO

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

FIGURE 49 UART FIFO

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

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

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

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

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

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

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

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

TXFIFO_LVL

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

FIGURE 50 UART AMT LEVEL

RXFIFO_LVL

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

FIGURE 51 UART AFL LEVEL

The FIFOs are 255 deep. Unused bits should be set to zero when programming.

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

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

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

FRAME_TIME

Programmable Time Out		
#define	FRAME_TIME_MASK	0x00FFFFFF //

FIGURE 52 UART FRAME TIME

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

BAUD_RATE

TX & RX Frequency		
#define	TX_BAUD_MASK	0x0000FFFF //
#define	RX_BAUD_MASK	0xFFFF0000 //

FIGURE 53 UART BAUD RATE

BAUD_RATE is a programmable count to determine the frequency of operation. The master clock is the reference which can be 32 MHz or the user programmed PLL rate associated with this port. See Channel Control Register B. The count programmed [N-1] determines the frequency of transmission or reception plus adjusts some of the filtering aspects of the receiver. Note: not all frequencies shown are available in all modes. RS-232 is limited in BW.

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

Using the PLL reference can provide more exact frequencies in some cases. Setting to 1.8432 MHz and using a divisor of 191 (192) will yield 9600 exactly. The Windows test software for this project programs 3.6864 MHz for the reference to provide many of the standard frequencies “exactly”.

PACKET_FIFO

PACKET FIFO		
#define	PKT_FIFO_MASK_TX	FFFF //
#define	PKT_FIFO_MASK_RX	0FFF //

FIGURE 54 UART PACKET FIFO

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

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

15 RxParErrLat
14 RxFrameErrLat
13 RxDataOvFlLt
12 RxPckOvFlLt

The definitions are found in the Channel Status register description.

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

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

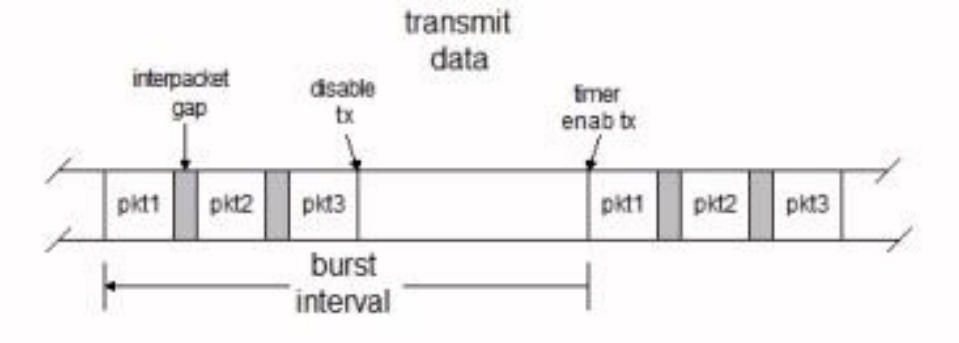
The Frame Timer should be programmed to determine the conditions for the end of frame. If left at the default setting packets will not be properly detected resulting in non-optimal behavior.

TX_TIMER_MOD

Tx Timer Modulus Reg		
#define	TX_TIMER_MOD_MASK	xFFFFFFFF //

FIGURE 55 UART TX MODULUS

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



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

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

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

TX_TIMER_CNT

Tx Timer Timer Reg		
#define	TX_TIMER_CNT_MASK	xFFFFFFFF //

FIGURE 56 UART TX TIMER CNT

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

Port I/O Line Mapping

ccXMC model is routed to Pn6

HDLC Port 0:

HDLC transmit data	=> I/O 0:	A1,B1
HDLC receive data	=> I/O 1:	D1,E1
HDLC transmit clock	=> I/O 2:	A3,B3
HDLC receive clock	=> I/O 3:	D3,E3

HDLC **Port 1:**

HDLC transmit data	=> I/O 4:	A5,B5
HDLC receive data	=> I/O 5:	D5,E5
HDLC transmit clock	=> I/O 6:	A7,B7
HDLC receive clock	=> I/O 7:	D7,E7

NRZ-L Port 2:

NRZ-L transmit data	=> I/O 8:	A9,B9
NRZ-L receive data	=> I/O 9:	D9,E9
NRZ-L transmit clock	=> I/O 10:	A11,B11
NRZ-L receive clock	=> I/O 11:	D11,E11

NRZ-L Port 3:

NRZ-L transmit data	=> I/O 12:	A13,B13
NRZ-L receive data	=> I/O 13:	D13,E13
NRZ-L transmit clock	=> I/O 14:	A15,B15
NRZ-L receive clock	=> I/O 15:	D15,E15

See Table for Pn6 for UART definitions within the table.
IO 0-15 are assigned to transceivers which can be outfit with RS-485 or LVDS devices. RS-485 is the default.

Interrupts

ccXMC-Serial interrupts are treated as auto-vectored. When software enters into an exception handler to deal with a ccXMC-Serial interrupt software must read the status register to determine the cause(s) of the interrupt, clear the interrupt request(s) and process accordingly. Power-on initialization will provide a cleared interrupt request and interrupts disabled.

For example, the ccXMC-Serial TX state machine(s) generates an interrupt request when a transmission is complete, and the TX int enable and Master interrupt enable bits are set. The transmission is considered complete when the last bit is output from the output shift register.

The interrupt is mapped/encoded to INTA, which is mapped to a system interrupt when the PCIe/PCI bus configures. The source of the interrupt is obtained by reading appropriate ISR status. The status remains valid until that bit in the status register is explicitly cleared.

When an interrupt occurs, the Master interrupt enable should be cleared, and the status register read to determine the cause of the interrupt. Next perform any processing needed to remove the interrupting condition, clear the latched bit and set the Master interrupt enable bit high again.

The individual enables operate after the interrupt holding latches, which store the interrupt conditions for the CPU. This allows for operating in polled mode simply by monitoring the Interrupt Status register. If one of the enabled conditions occurs, the interrupt status bit will be set, but unless the Master interrupt, and the channel interrupt enable is set, a system interrupt will not occur.

Loop-back

The Driver package has reference software, which includes external loop-back tests. ccXMC-Serial utilizes Pn6 rear panel connector. Installed onto a test platform the IO are mapped to a SCSI connector enabling loop-back. The test requires an external cable with the following pins connected. Make the following connections (TP2 unless noted). The IO numbers match **Pn6** definitions later in the manual. This version supports internally generated TX CLK for HDLC and 422 mode for the UARTs.
Note: TP1, 2 are both ordered as follows: 1, 35, 2, 36, 3, 37…32, 66, 33, 67, 34, 68.

HDLC SIGNAL	A	B	IO	A	B	IO
Port 0 Data TX to RX	1	35	0	2	36	1
Port 0 Clock TX to RX	5	39	2	6	40	3
Port 1 Data TX to RX	9	43	4	10	44	5
Port 1 Clock TX to RX	13	47	6	14	48	7

NRZ Signals	A	B	IO	A	B	IO
Port 2 Data Tx to Rx	17	51	8	18	52	9
Port 2 CLK Tx to Rx	21	55	10	22	56	11
Port 3 Data Tx to Rx	25	59	12	26	60	13
Port 3 CLK Tx to Rx	27	61	14	28	62	15

UART Signals	A	B	A	B
Port 4 Data Tx to Rx	29	63	30	64
Port 4 RTS to CTS	31	65	32	66
Port 5 Data Tx to Rx	67	23	68	24
Port 5 RTS to CTS	33	1	34	6
Port 6 Data Tx to Rx	2	3	7	8
Port 6 RTS to CTS	4	5	9	10

Highlighted connections are installed on HDR1 on XMC-UNIV-TEST

Loop-back is accomplished with HDEterm68 connected to PMC-UNIV-TEST.
The first HDEterm68 is connected as shown above.
A second test set-up with the lower 16 tied to the upper 16 [IO0-IO16 etc.] is used to test the parallel port.
A third HDEterm68 is used along with the test clock to independently test the IO and terminations.
See the following links for the required devices:
https://www.dyneng.com/HDEterm68.html
https://www.dyneng.com/PMC-UNIV-TEST.html
https://www.dyneng.com/HDEcabl68.html

XMC PCIe Pn5 Interface Pin Assignment

The figure below gives the pin assignments for the XMC Module PCIe Pn5 Interface on ccXMC-Serial. See the User Manual for your carrier board for more information. Unused pins may be assigned by the specification and not needed by this design.

A	B	C	D	E	F	
PCIeTXDP0	PCIeTxDN0	3.3V	PCIeTXDP1	PCIeTxDN1	VPWR	**1**
GND	GND	TRST#	GND	GND	PERST#	**2**
PCIeTXDP2	PCIeTxDN2	3.3V	PCIeTXDP3	PCIeTxDN3	VPWR	**3**
GND	GND	TCK	GND	GND	MRSTO#	**4**
PCIeTXDP4	PCIeTxDN4	3.3V	PCIeTXDP5	PCIeTxDN5	VPWR	**5**
GND	GND	TMS	GND	GND	12V	**6**
PCIeTXDP6	PCIeTxDN6	3.3V	PCIeTXDP7	PCIeTxDN7	VPWR	**7**
GND	GND	TDI	GND	GND	-12V	**8**
					VPWR	**9**
GND	GND	TDO	GND	GND	GA0	**10**
PCIeRXDP0	PCIeRxDN0	MBIST#	PCIeRXDP1	PCIeRxDN1	VPWR	**11**
GND	GND	GA1	GND	GND	MPRES	**12**
PCIeRXDP2	PCIeRxDN2	3.3VAUX	PCIeRXDP3	PCIeRxDN3	VPWR	**13**
GND	GND	GA2	GND	GND	MSDA	**14**
PCIeRXDP4	PCIeRxDN4		PCIeRXDP5	PCIeRxDN5	VPWR	**15**
GND	GND	MVRMO	GND	GND	MSCL	**16**
PCIeRXDP6	PCIeRxDN6		PCIeRXDP7	PCIeRxDN7		**17**
GND	GND		GND	GND		**18**
REFCLKP	REFCLKN		WAKE#	ROOT0#		**19**

FIGURE 57 PN5 INTERFACE

Some signals shown but not used.
Rx & Tx relative to XMC. i.e. Tx means transmitted from XMC and Rx means received by XMC

XMC IO Pn6 Interface Pin Assignment

The figure below gives the pin assignments for the XMC Module Pn6 Interface on ccXMC-Serial. See the User Manual for your carrier board for more information.

A	B	C	D	E	F	
IO_0P	IO_0N		IO_1P	IO_1N		**1**
GND	GND		GND	GND		**2**
IO_2P	IO_2N		IO_3P	IO_3N		**3**
GND	GND		GND	GND		**4**
IO_4P	IO_4N		IO_5P	IO_5N		**5**
GND	GND		GND	GND		**6**
IO_6P	IO_6N		IO_7P	IO_7N		**7**
GND	GND		GND	GND		**8**
IO_8P	IO_8N		IO_9P	IO_9N		**9**
GND	GND		GND	GND		**10**
IO_10P	IO_10N		IO_11P	IO_11N		**11**
GND	GND	U2_TXP	GND	GND	U2_RXP	**12**
IO_12P	IO_12N	U2_TXN	IO_13P	IO_13N	U2_RXN	**13**
GND	GND	U2_RTSP	GND	GND	U2_CTSP	**14**
IO_14P	IO_14N	U2_RTSN	IO_15P	IO_15N	U2_CTSN	**15**
GND	GND	U3_TXP	GND	GND	U3_RXP	**16**
U1_TXP	U1_TXN	U3_TXN	U1_RXP	U1_RXN	U3_RXN	**17**
GND	GND	U3_RTSP	GND	GND	U3_CTSP	**18**
U1_RTSP	U1_RTSN	U3_RTSN	U1_CTSP	U1_CTSN	U3_CTSN	**19**

FIGURE 58 PN6 INTERFACE

Notes:
1. See IO assignments for standard differentials in Port I/O Line Mapping table.
2. RS-422 UART connections shown.
3. For RS-232 UART operation use TXN, RXP, CTSP, RTSN connections – see SP335 data sheet for more information. See highlighted selections above.

Applications Guide

Interfacing

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

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

Watch the system grounds. All electrically connected equipment should have a fail-safe common ground that is large enough to handle all current loads without affecting noise immunity. Power supplies and power-consuming loads should all have their own ground wires back to a common point.

Power all system power supplies from one switch. Connecting external voltage to ccXMC-Serial when it is not powered can damage it, as well as the rest of the host system. This problem may be avoided by turning all power supplies on and off at the same time. ccXMC-Serial does have transorbs for ESD and signal excursion protection.

Keep cables short. Flat cables, even with alternate ground lines, are not suitable for long distances. The connector is pinned out for a standard SCSI II/III cable to be used. The twisted pairs are defined to match up with the ccXMC-Serial pin definitions. 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 (HDEterm68). The terminal block can mount on standard DIN rails.

https://www.dyneng.com/HDEterm68.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 RS-485 devices rated voltages. The transorb protection is meant to provide protection against transient events.

Construction and Reliability

ccXMC Modules are conceived and engineered for rugged industrial environments. The ccXMC-Serial is constructed out of 0.062 inch thick High Temp FR4 material. The PC Boards are ROHS compliant. Dynamic Engineering has selected gold immersion processing to provide superior performance, and reliability (not subject to tin whisker issues).

Through hole and surface mounting of components are used.

The PMC connectors are rated at 1 Amp per pin, 100 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable.

The ccXMC is secured against the carrier with multiple screws attached to the 2 stand-offs and thermal interface locations. The screws provide significant protection against shock, vibration, and incomplete insertion.

The ccXMC 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

The ccXMC-Serial design consists of CMOS circuits. The power dissipation due to internal circuitry is very low. It is possible to create a higher power dissipation with the externally connected logic. In a conduction cooled environment the thermal attachment points become very important.

ccXMC-Serial has an internal floating thermal plane attached to the thermal rib locations. In addition, full plane ground and power planes will help keep the PCB at an even temperature – avoiding hot spots. It is up to the system to provide the thermal interface to provide the path for the thermal flux to move from ccXMC-Serial into the cooling system of the chassis.

The devices utilized are rated at Industrial or better. It is recommended to provide adequate margin to allow for the thermal rise through the chassis to the PCB itself.

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, Suite B&C
Santa Cruz, CA 95060
(831) 457-8891
support@dyneng.com

Specifications

Host Interface:	(XMC) PCIe Mezzanine Card
Interface:	2 Full Duplex HDLC serial interfaces. Odd length message size supported, HW CRC, LSB first. 2 NRZ-L full duplex interfaces. Programmable LSB/MSB first, bit count, clock sense, data sense. Parallel Port can be selected on a bit-by-bit basis. 3 UART interfaces with Rx, Tx, RTS, CTS and programmable RS-232/RS-422 operation.
TX Data rates generated:	32 MHz oscillator used to generate 200 MHz HDLC I/O clock sampling frequency. PLLA, PLLB, PLLC selectable as references for HDLC and NRZ ports. User changeable for other unique frequencies PLLD [3.6864 MHz) used as UART reference along with 32 MHz DCM generated clock.
RX Data rates accepted:	HDLC rates 1-10 MHz accepted. NRZ-L programmable via Tx Rate register and selected PLL value.
Software Interface:	Control Registers, Status Ports, Dual Port RAM, Driver Available
Initialization:	Hardware reset forces all registers to 0.
Access Modes:	LW boundary Space (see memory map)
Interrupt:	HDLC: TX and Rx interrupts at end of message HDLC: TX interrupt at end of frame transmission HDLC: RX interrupt when abort received NRZL: TX and RX Packet complete interrupts Software interrupt UART: Tx, Rx transfer interrupts
DMA:	Not implemented at this time
DDR:	Not utilized for this version
Onboard Options:	All Options are Software Programmable
Interface Options:	ccXMC depends on carrier options.
Dimensions:	Standard Single ccXMC Module.
Construction:	High temp. FR4 Multi-Layer Printed Circuit, Through Hole and Surface Mount Components.
Temperature Coefficient:	2.17 W/^{o}C for uniform heat across XMC
Power:	Max. **TBD** mA @ 5V
Temperature range	Industrial Temperature components standard (-40 + 85)

Order Information

ccXMC-Serial-HDLC	ccXMC Module with 2 HDLC, 2 NRZ-L, and 3 UART ports, parallel port (overlaps with HDLC and NRZ channels) RS-485 I/O. 32-bit data interface. SOSA aligned IO.
-12MM	Option to include 12mm Standoffs to support mounting in a 12 mm inter-board system.
-CC	Add Conformal Coating
-ROHS	Add RoHS processing
Eng Kit–ccXMC-Serial-HDLC	HDEterm68 - 68 position screw terminal adapter https://www. dyneng. com/HDEterm68. html HDEcabl68 - 68 I/O twisted pair cable https://www. dyneng. com/HDEcabl68. html XMC-UNIV-Test – passive vertical adapter for PCIe to XMC conversion. SCSI support for Pn6. https://www.dyneng.com/XMC-UNIV-TEST.html

***Note**: The Engineering Kit is strongly recommended for first time cc*XMC-Serial-HDLC *purchases.*