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

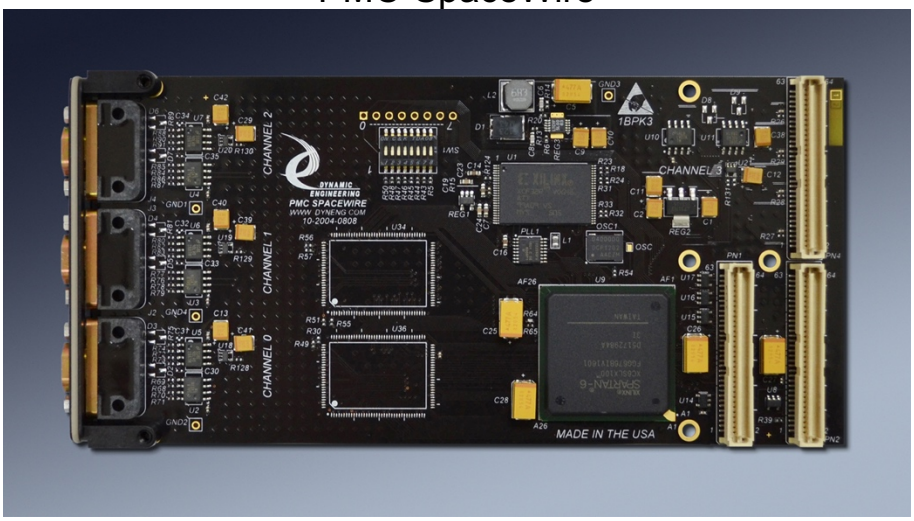
User Manual

SpaceWire Hardware Manual

Four-Channel SpaceWire Interface
ccPMC, PMC, PCI, PC104p

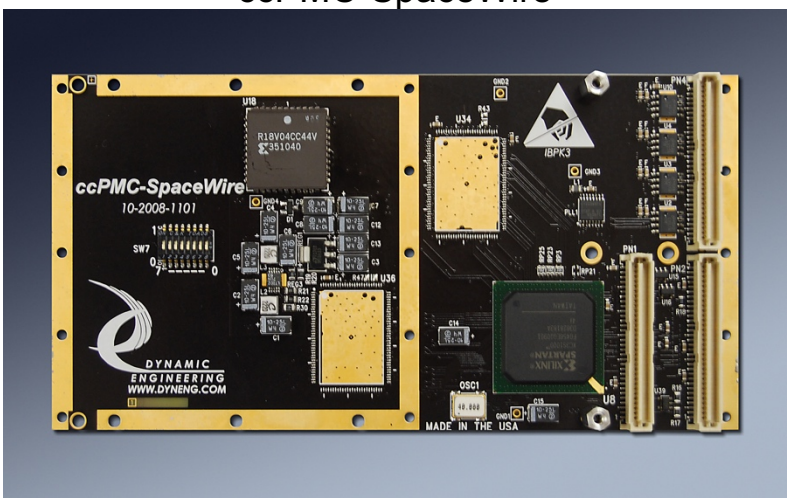
Manual Revision 12P1

PMC-SpaceWire



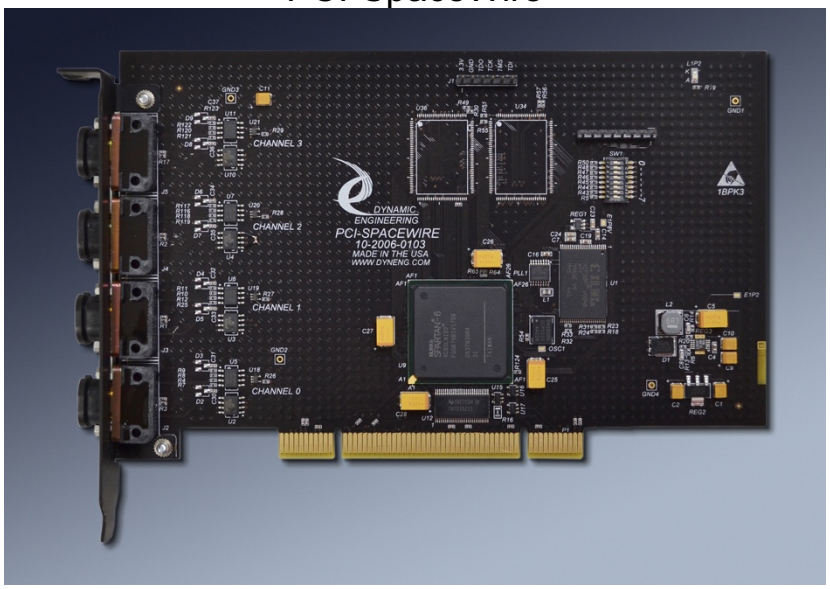
Corresponding Hardware: 10-2004-08(08,09)

ccPMC-SpaceWire



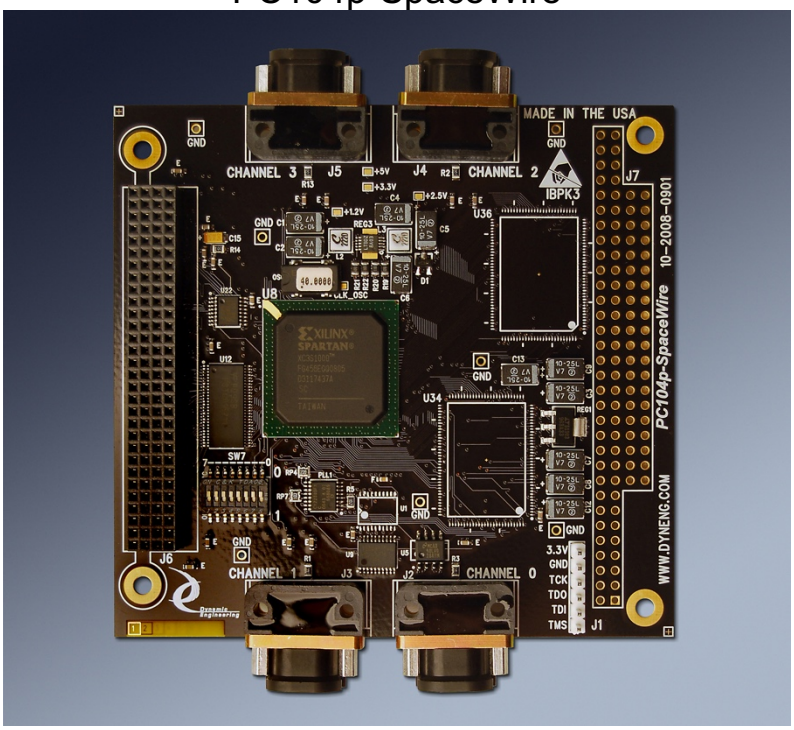
Corresponding Hardware: 10-2008-1101

PCI-SpaceWire



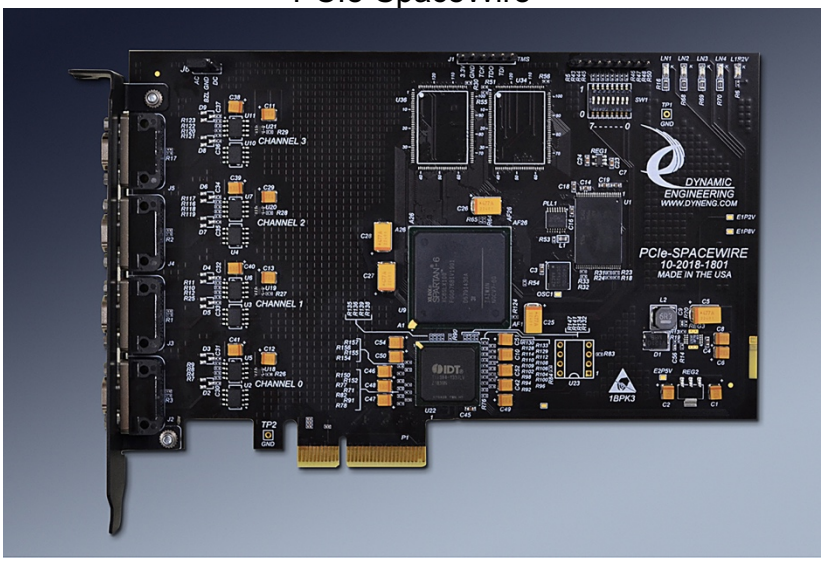
Corresponding Hardware: 10-2006-01(04)

PC104p-SpaceWire



Corresponding Hardware: 10-2009-09(02)

PCIe-SpaceWire



Corresponding Hardware: 10-2018-1801

SpaceWire
ccPMC, PMC, PCI, PCIe, PC104p
Four-Channel SpaceWire Interface

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

Revised 11/18/19

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.

Connection of incompatible hardware is likely to cause serious damage.

Table of Contents

List of Figures

Product Description

SpaceWire is part of the Dynamic Engineering family of modular I/O. There are 5 formats currently available: PMC, ccPMC, PCI, PCIe, and PC104p. The PC104p version can be built with or without the ISA connector for PCI-104 compatibility. Each has similar functionality with some variation in the IO connectors. Four channels are supported per card, each with internal FIFO and separate DMA engines to support high speed operation.

The “K” series is in continued production to support current and previous projects utilizing these models. Please see the BK models for an enhanced feature set including more memory, larger DMA segments and more. BK is recommended for new systems and designs. BK is very similar to K to make porting from one to the other easy to accomplish.

In addition, external FIFO’s can be installed on two of the IO channels for additional burst capability. Currently options are available to have the external FIFO attached to channel 0 RX and TX or channels 0 and 1 RX only.

The following diagram shows the standard SpaceWire configuration:

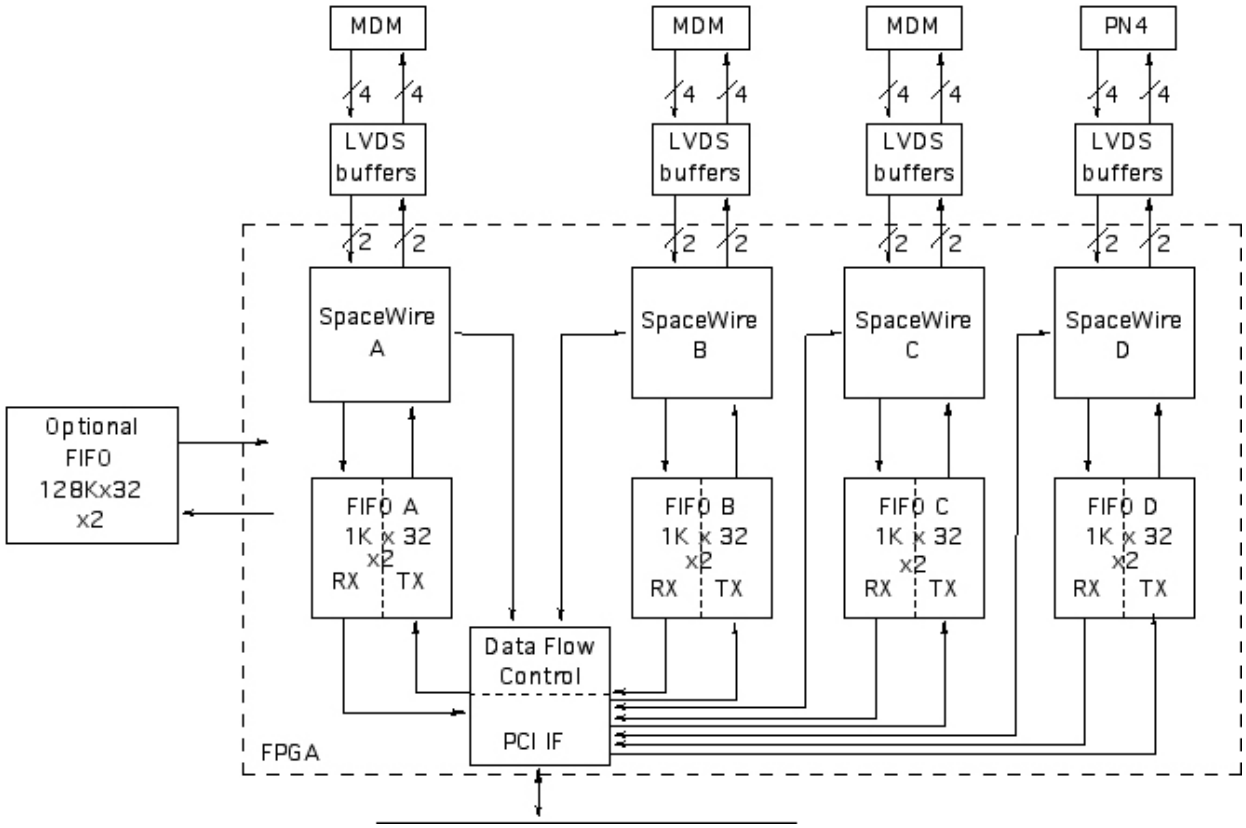


FIGURE 1 SPACEWIRE BLOCK DIAGRAM

Note: Not shown in diagram are the Packet FIFO's which are 1K x 32 per channel per direction [8] to store packet sizes for transmission or definitions from reception.

The channel D connector implementation varies depending on the format. The PCI, PCIe, and PC104p versions have four MDM connectors for the four I/O channels. The PMC version can be configured with 3 MDM connectors and have 1 channel on Pn4 or all 4 channels routed to Pn4. The ccPMC version has all 4 channels routed to Pn4. In all cases high speed, differential routing with controlled impedance and matched lengths are used for the SpaceWire signaling. It is recommended to use a SpaceWire compatible cable to interconnect your hardware. Dynamic Engineering has several standard lengths of cable and offers custom lengths as well.
http://www.dyneng.com/SpaceWireCable.html

Other custom interfaces are available. We will redesign the state machines and create a custom interface protocol. Please see our web page for current protocols offered. Please contact Dynamic Engineering with your custom application.

The SpaceWire protocol implemented provides Low Voltage Differential Signaling (LVDS) data inputs and outputs. The transmit data rate is selected by a combination of the programmed output frequencies of the PLL and the divisor values in the channel control registers. The PLL is programmed via software over a serial I2C interface. Transmit data rates are selectable from 2 Mbps to 200 Mbps. The receiver will automatically adjust to the data rate seen.

The SpaceWire specification requires that the transmit frequency be 10 Mbps during the link connection process. In order to accommodate this, the PLL frequency should be at least close to a multiple of 10 MHz. Once the link protocol has established a connection, the transmit speed will convert to the desired transmit rate specified in the channel control registers.

Four independent SpaceWire channels are provided per card. Each SpaceWire channel has two LVDS signal pairs for input and two LVDS signal pairs for output. The electrical interface for SpaceWire is as specified in document ECSS-E-50-12C, published by the European Cooperation for Space Standardization dated 31 July 2008. Connections for the first three SpaceWire channels on the card are with MDM style connectors as required by the specification. The fourth PMC channel is only available on the PN4 connector. Any or all of the four channels can be routed to the PN4 connector (PMC only) rather than the MDM connectors by selecting which 0Ω resistor packs are installed on the board. The PCI, PCIe, and PC104p versions have four MDM connectors, one for each channel. The ccPMC version has all ports routed to Pn4. The (cc)PMC-SpaceWire conforms to the PMC and CMC draft standards. This guarantees compatibility with multiple PMC Carrier boards. Because the PMC may be mounted on different form factors, while maintaining plug and software compatibility, system prototyping may be done on one PMC Carrier board, while final system implementation uses a different one.

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

Mechanically, PCI-SpaceWire is designated as a short length 32-bit add-in card, commonly referred to as a half-size PCI card (4.2 x 6.6 inch).

The PC104p version is a standard size stackable card with connectors in the legal connector mounting area. PC104p-SpaceWire meets the vertical height restrictions. The ground plane is tied to the stacking hardware to allow for conduction cooled environments.

All formats have a common software interface allowing for porting between systems. Dynamic Engineering offers drivers and reference software for Windows® and Linux.

Each SpaceWire channel is supported by two 1K by 32-bit FIFOs. The TX FIFOs support long-word writes and the RX FIFOs support long-word reads. For testing a FIFO test bit in each channel control register enables the data to be routed from the TX to the RX FIFO for a full 32-bit path for loop-back testing of the FIFOs. Data is latched and the bus immediately released on a write-cycle. As soon as data is present in the FIFO it is pre-read to be immediately available for a read cycle. This allows minimal delay on the PCI write to TX FIFO path and PCI read from the RX FIFO path as well as the accesses for the TX and RX state machines.

Each SpaceWire receive channel can receive SpaceWire packets and store them in the associated input FIFOs. Each SpaceWire packet will be zero filled to align with 32-bit long words. The actual number of bytes in the packet will be stored in the RX packet-length FIFO [1Kx32] and the Data is stored in the 1K by 32-bit FIFO, optional external memory can be added to expand the receive data FIFO by an additional 256K bytes on channel 0 and 1. The host can poll the FIFO flags or wait for the packet received or RX almost full interrupts. The packet can then be read over the PCI bus directly from the RX data FIFO. Flow control into the receive FIFO is controlled by the SpaceWire protocol, and prevents over-flow situations automatically.

Each SpaceWire transmit channel has a separate 1K x 32-bit FIFO. The FIFO is written as long words. The number of bytes to be transmitted is specified by writing a byte count to the TX packet-length FIFO. Whenever TX data and packet-length values have been written, the transmitter will send data bytes provided the connection is established

and the flow control from the destination side has authorized the transfer. A packet disable bit in the channel control register allows the interface to function without packetizing the data. When in this mode the transmitter will send data as soon as it is written to the TX FIFO. This essentially treats the data as one infinite length packet. Optional external memory can be added to expand the transmit data FIFO by an additional 256K bytes on channel 0.

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 is received, zero-padding of the unused upper-bytes will occur before the long-word is written to the FIFO.

The SpaceWire board supports various interrupts. An interrupt can be configured to occur when the TX FIFO is almost empty, the RX FIFO is almost full, when a SpaceWire packet has been received, when a time-code character is received or when an error has occurred. 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.

In command and control situations direct read-write access to the FIFO's makes sense. The messages tend to be short and the added overhead of setting up DMA is not justified. For data transfer DMA is recommended. Each channel has a separate transmit and receive DMA engine for a total of 8 programmable units. The DMA engines can be programmed for long transfers and handle scatter-gather requirements automatically. With only 1 interrupt to deal with at the end of the transfer it is the lowest overhead transfer method for medium and large transfers. Internal to the design is an 8-channel DMA arbiter which controls which channel can access the PCI bus for DMA operation. The arbiter operation is completely automatic.

Theory of Operation

The SpaceWire designs are for transferring data from one point to another using the SpaceWire protocol as specified in document ECSS-E-50-12C, published by the European Cooperation for Space Standardization dated 31 July 2008.

Continuous development in the SpaceWire community means fairly frequent updates for new features. Generally new features can be added to a released card with a FLASH update. New features are designed to allow for non-updated software to still function – please set undefined bits to zero when programming to facilitate the migration path.

The SpaceWire board features a Xilinx FPGA. The FPGA contains the PCI interface, all of the registers, FIFO's and protocol controlling elements of the SpaceWire design. Only the transceivers and clock circuitry are external to the Xilinx device.

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

Scatter-gather DMA is provided for in this design. Once the physical address of the first chaining descriptor is written to the DMA pointer register, the interface will read a 12-byte block from this location. The first four bytes comprise a long-word indicating the physical address of the first block of the IO buffer passed to the read or write call. The next four bytes represent a long-word indicating the length of that block (only the lower 22 bits are valid). The final four bytes are a long-word indicating the physical address of the next chaining descriptor along with two flag bits, in bit position 0 and 1. Bit zero is set to one if this descriptor is the last in the chain. Bit one is set to one if the IO transfer is from the SpaceWire board to host memory, and zero if the transfer is from memory to the board. These bits are then replaced with zeros to determine the address of the next descriptor, if there is one.

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

The eight DMA controllers obtain access to the PCI bus by asserting a request to the DMA arbiter. Once a controller is granted PCI access, it keeps the bus until it drops its request. At this point if another controller is requesting the bus, it will be granted access. If multiple DMA controllers are asserting requests, the arbiter grants access in

a round-robin pattern, but no controller loses its grant until it drops its request.

A controller will drop request when it reaches the end of a scatter-gather list entry or the end of a DMA descriptor acquisition. It will also drop request when a transfer from the device to memory is almost out of data, or when a transfer from memory to the device is almost out of room to store the data, or if it has held the PCI bus for 1024 PCI clocks.

A DMA controller can be forced to drop request if it is preempted by another controller that has an urgent need to transfer data. The ability to preempt other controllers is enabled by the write/read DMA priority enable bits in the channel control registers and is triggered by the transmit almost empty and receive almost full programmable FIFO levels.

In the rev. H design message-signaled interrupts are requested. If the operating system grants this request, the interrupt controller must have the ability to preempt all active DMA controllers in order to transfer the interrupt message in a timely manner. This preemption cannot be overridden or disabled when MSI is enabled.

Beginning with rev. G, a retry counter was added to all the DMA controllers. This six-bit counter is incremented whenever the DMA controller initiates a PCI bus-cycle and is cleared whenever the controller drops bus request or a data-word is successfully transferred. If the count reaches the count stored in the base control register (bits 31-26), the controller is forced to drop bus request, which allows another controller to gain access. A stored count value of zero disables this mechanism. The purpose of the retry counter is to prevent a DMA controller from hanging-up the PCI bus when it is unable to move any data because of some PCI bus or memory access problem.

Several state-machines within the FPGA control the link, FIFOs, data transceivers, and flow-control for each channel. The transmitter and receiver for each channel are interdependent. The transmitter requires flow-control information from the receiver to function and the receiver requires the transmitter to send flow-control tokens to regulate the flow of received data.

In a typical transmit sequence the local receiver receives flow control tokens (FCTs) from the remote node. Each FCT authorizes the local transmitter to send eight N(ormal)-Chars (a data-byte, End-Of-Packet token (EOP) or Error End-of-Packet (EEP)) back to the remote node. An up-down counter is used to keep track of the number of N-Chars authorized, and the number of N-Chars that have been sent. Likewise, the local transmitter sends FCTs to the remote receiver node to enable that node's transmitter to send N-Chars to the local receiver. The number of outstanding FCTs is based on the amount of room available in the receive FIFO, but can never exceed 7 FCTs (56 bytes).

The transmitters will multiplex Time-Code characters, FCT characters, N-characters and NULL characters (in that order of priority) onto the data stream to regulate the flow of data in both directions. At the end of a transmitted packet, the transmitter will append an EOP (or possibly an EEP if relaying a packet with an error) character to the message stream to alert the receiver to the completion of the current packet.

The SpaceWire board uses Data-Strobe encoding where clock and data information are sent on two paired serial links. Exactly one transition occurs in either the data line or the strobe line at the end of each bit period allowing the clock to be recovered from the data strobe pair. The timing is shown in figure 2.

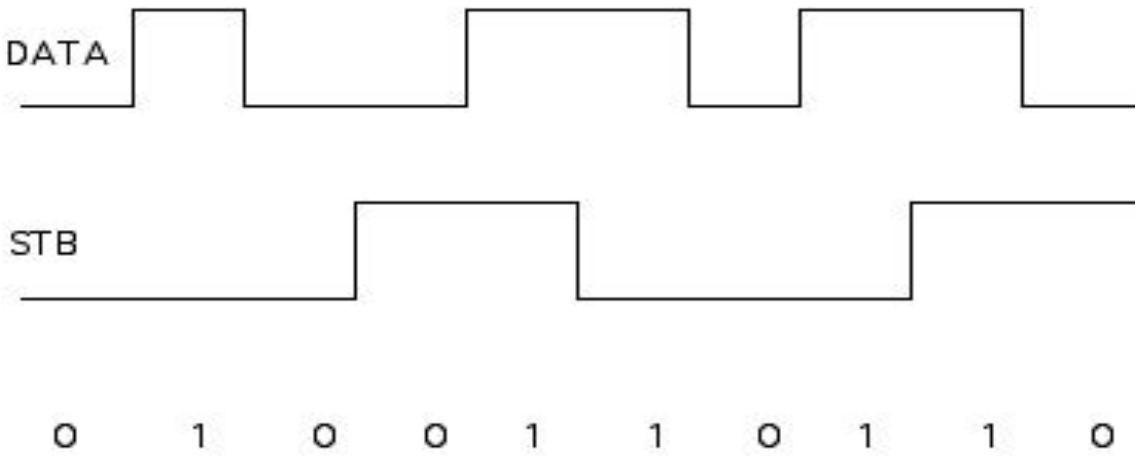


FIGURE 2 SPACEWIRE DATA STROBE ENCODING

Before the transmitter can start operation, it must establish a link with the connected node. When the link enable control bit is set, the connection state-machine resets the NULL-detected latch and waits 6.4 microseconds before enabling the receiver. The receiver then monitors SpaceWire link activity for 12.8 microseconds. If any link errors or tokens other than NULLs are seen during this period, the receiver is disabled and the process starts over.

At this point the state-machine is in the ready state waiting for a start or auto-start control bit to be set to move to the starting state. The start control bit causes the transmitter to immediately begin sending NULLs and the receiver to look for a NULL in response, whereas the auto-start control bit causes the state-machine to wait for a NULL to be received before transitioning to the starting state and sending NULLs. If a NULL is not seen within 12.8 microseconds while in the starting state, or if errors or tokens other than NULLs are seen, the transmitter and receiver are disabled and the process starts over.

When the transmitter has sent a NULL and the receiver has received a NULL character, the transmitter sends an FCT character and the receiver looks for an FCT in return. If this occurs, it means that the remote node has recognized the NULL and is ready to connect. Once an FCT has been sent and received the link proceeds to the run state.

The transmit frequency then switches to the requested operational frequency from the 10 MHz connection frequency and Time-codes, FCTs and N-chars (data or end-of-packet tokens) can be sent to and from the node. (See figure 8-2: State diagram for SpaceWire link interface on page 64 of the SpaceWire spec. ECSS-E-ST-50-12C)

The SpaceWire specification allows for a maximum of 56 bytes of credit (seven FCTs) to be outstanding at any time. For each eight bytes that are received, the receiver will request the transmitter to send another FCT as long as sufficient buffer space exists to accommodate the additional data. If the receiver receives a data byte when the outstanding credit is zero, or if an FCT is received that causes the transmit credit to exceed 56 bytes, a credit error has occurred.

The credit error is one of several error conditions that can disrupt the link connection. The others include parity error, disconnect error and escape error. If any of these occur a receive error and the offending error condition status will be latched, the link will disconnect and after a delay, attempt to re-establish the connection.

The parity error occurs when the relevant bits of two successive characters do not constitute odd parity (an odd number of ones). Parity coverage for SpaceWire is offset from the character boundaries by two bits. Although each character begins with a parity bit followed by a data control flag, these two bits are combined with the payload bits from the previous character to determine the parity. The disconnect error occurs when there is no activity on the strobe or data lines to the receiver for a period of 850 nanoseconds. The escape error occurs when an escape character is followed by anything other than an FCT or a data character.

Beginning with Rev K minor rev 3, a Receive Packet FIFO control register was added to select when to back pressure the link to prevent receive packet FIFO overflows from occurring. Previous versions allowed overflows to occur and the register can be set to 0x3FF or 0x000 to operate in this/legacy mode. A receive FIFO overflow will also cause a receive error to be latched, but will not in itself cause a link disconnect

If a receive data-packet is in progress when an error occurs, the receiver ceases writing data to the receive data FIFO and writes the receive packet-length FIFO with the current byte-count with the packet error bit set. If this packet fragment is to be relayed to a remote node, it should be terminated with an EEP rather than an EOP.

If a transmit packet was in progress when the error occurred, the transmitter stops sending data and attempts to purge the remainder of the current packet from the transmit FIFO. Data will be read from the FIFO at the rate of 132 Mbytes per second, rather than using the FIFO reset, to avoid deleting data belonging to subsequent packets.

A 30 microsecond delay is provided for this purpose, however, if the packet-length is large or if all the packet data is not currently present in the transmit FIFO, the transmitter will be unable to complete the data purge within the 30 microsecond period and the link state-machine will go ahead and disable the receiver and transmitter in preparation for reestablishing the link. The remote receiver that is receiving the data will detect an error when the link is disabled, if it has not already, and will be aware that the packet has been prematurely terminated.

This will clear the transmitter and receiver error state, but a transmitter purge error status bit will be latched to alert the user of this error condition. The user can then read the transmit packet-length FIFO count to determine how many packets are pending, reset the transmit FIFO (which also clears the transmit packet-length FIFO) and perform whatever error recovery is indicated.

See the SpaceWire specification for clarification or elaboration on any of these features.

Programming

Programming the SpaceWire board requires only the ability to read and write data from the host. The base address is determined during system configuration of the PCI bus. The base address refers to the first user address for the slot in which the board is installed. The VendorId = 0x10EE. The CardId = 0x002A.

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

If DMA is to be used it may be necessary to acquire a block of non-paged memory that is accessible from the PCI bus in which to store chaining descriptor list entries. If the Dynamic Engineering device drivers are used, the I/O channel driver will handle all the DMA internal mechanics automatically.

In order to transfer data to another SpaceWire node, several steps must be performed. First a physical connection must be established with the appropriate interface cable. Then the PLL must be programmed to the desired clock configuration. The PLL is connected to the Xilinx by an I^2C serial bus. The PLL internal registers are loaded with 40 bytes of data that are derived from a .jed file that can be generated by the CyberClock utility from Cypress semiconductor.

Reference: http://www.dyneng.com/CyberClocks.zip. Select CyClocksRT; the specific PLL part number is CY22393; and the external reference clock frequency is 40 MHz. You can then specify frequencies for CLKA – CLKD which supply the I/O reference clocks for channels 0 – 3 respectively. Click the Calculate button and save the file.

A four-bit count field in each channel's control register allows the I/O reference clock to be divided by any integer value from 1 to 16 (count values 0 – 0xF) to yield the final operational bit-rate. There is also a 5-bit initial count field for each channel in the base control register that is used to divide the I/O reference clock in order to generate the 10 Mbit per second connection rate. To allow the channels to achieve this connection rate, the PLL should not be programmed to frequencies below 10 MHz and should be at least close to a multiple of 10 MHz.

Routines to program the PLL are included in the driver and UserApp code provided in the engineering kit for the board. The driver will analyze the 40-byte PLL register field to determine the requested frequencies and set the initial count values automatically whenever the PLL is reprogrammed. If you are writing your own driver, contact Dynamic Engineering and we can send you a file with code excerpts from our driver and

test software that cover each step of the process from parsing the .jed file to the low-level bit manipulation of the I^2C bus.

Next the link Enable bit and either the Start or Auto-start bit must be set in the channel control register along with the desired operational clock divisor and any interrupt enables that are to be monitored. When the link has been established as reported in the channel status register, data may be written to the transmit FIFO using either single-word writes or DMA. If the packet disable bit has been set, data will be sent as soon as it is written, provided that the receiving node has authorized the transfer. In this mode the data is treated as a single infinite-length packet.

If packetizing is not disabled, the data will not be sent until a packet-length has been written to the SPWR_CHAN_TX_PKT_LEN FIFO. If the packet-length is not divisible by four, the remainder of the last 32-bit FIFO word will be ignored and the next packet's data will begin with the next FIFO word. A packet-length must be written for each packet to be sent, unless the constant packet-length control bit is set. If this bit is set, the first packet-length read will be used continuously until either the data in the transmit data FIFO is exhausted or the constant packet-length control bit is cleared.

Once the link has been established, the receiver will automatically adapt to the frequency of the remote node's transmitter. As data is received, a 32-bit data word will be written to the receive FIFO for every four bytes that are received. When an end-of-packet character is received, the remainder of the received data will be written to the FIFO regardless of whether four bytes have been received (the unused bytes are written as zeros) and the received packet-length (byte count) will be written to the SPWR_CHAN_RX_PKT_LEN FIFO. The packet done bit will also be set and an interrupt generated, if it has been enabled.

Firmware Updates

Revision A: This is the first fully channelized SpaceWire version.

Revision B: Modified the transmit flow control at the end of packets to avoid credit errors when the EOP token is sent, but not authorized; changed the receive FIFO space to receiver flow-control to a 3-bit FCT authorized count to promote smoother behavior.

Revision C: Suppressed writing the receive packet-length when an error occurs and the packet-length is zero; updated DMA arbitration logic.

Revision D: Improved high-speed performance for packet-lengths not divisible by four; extended the maximum packet-lengths to 128 K Bytes and 2 G Bytes depending on Xilinx part and FIFO configuration; added a control bit to reuse a single packet-length and added latched transmit FIFO almost empty and receive FIFO almost full status bits.

Revision E: Corrected low-speed connection problem caused by missing back-to-back FCTs; revised handling of EEPs so that they do not cause a connection error when received but still set the error bit in the receive packet-length FIFO; added a corresponding error bit to the transmit packet-length FIFO to allow sending of an EEP (needed by router nodes); revised Xilinx configuration/revision numbering.

Revision F: Added transmit packet-length FIFO count field above transmit data FIFO count field at the same address; replaced transmit packet-length FIFO full status bit with transmitter purge error latched status bit to alert user to a failed attempt to fully purge aborted transmit packet data after a connection error; revised transmit flow-control to be more responsive and efficient.

Revision G: Added receive packet-length FIFO count field above receive data FIFO count field at the same address; revised input and output DMA state-machines to avoid a potential lock-up condition when preemption (DMA arbitration priority) is enabled. Six-bit retry counters were added to each of the eight DMA controllers. The counts are compared to a value written to bits 31-26 of the base control register.

Revision H: Special version developed for use with the INtime® real-time operating system. INtime works in conjunction with Windows to allow the creation of real-time processes that coexist and communicate with Windows using dedicated core(s) on a multi-core processor or sharing the core of a single-core processor. INtime cannot share interrupts with a Windows device (due to real-time determinism issues), so legacy (INTA-D) interrupts are difficult or impossible to use. This version requests MSI (message-signaled-interrupts) so that the SpaceWire card does not have to share

interrupts with another PCI device. Although Windows XP does not support MSI, the INtime software does, so devices that are passed to INtime control can use this feature.

Revision I: Fixed a problem with the credit-count calculation related to received error-end-of-packets (EEP). Power-management capability reporting (states D0/D3) was added, but no additional power-management capabilities. Removed DMA retry count and counters. Added channel interrupt status at the base level (SPWR_USER_SWITCH register).

Revision J: This version replaces the XC3S1500 Xilinx with an XC3S2000 Xilinx. Design types 0, 1 and 2 are now used for the XC3S1500 Xilinx and the XC3S1000 Xilinx is no longer supported. All design types are only available with 2 G Byte maximum packet-lengths. Channel control register bit 25 was added to enable the returning of only valid packet-lengths. When this feature is enabled and a packet-length has already been read, a length of zero will be returned instead of the old packet-length.

Revision K: Synchronized time-code data and flags to the PCI clock to eliminate parity errors due to data bits changing during the read process. Revision K will be the default FLASH load. It is recommended to order the –BK version to get the latest version for new designs and designs that can be updated. For legacy and re-order situations the default Revision K will be appropriate.

Revision 11 – minor rev3: Ported and resynthesized design into Spartan 6. Added logic to select/support BigEndian. Added design version indicator and minor revision bits. Removed switch value inversion. Implemented Receive Packet FIFO control register and logic. Changed PCI bus access logic, FIFO Full logic, and added timer to prevent bus lockups. Changed TSI 384 bridge settings to prevent cache coherency issue in certain system configurations.

Address Map

Register Name	Offset	Description
SPWR_BASE_CNTRL	0x0000	// Base control register
SPWR_USER_SWITCH	0x0004	// User switch read port
SPWR_TIME_CNTRL	0x0008	// Time control register
SPWR_RX_PKT_FF_FULL_CNTRL	0x000C	// RX Packet FIFO Full Control register
SPWR_CHAN_CNTRL_0	0x0010	// Channel 0 Control register
SPWR_CHAN_STATUS_0	0x0014	// Channel 0 Status register
SPWR_CHAN_FIFO_0	0x0018	// Channel 0 TX/RX FIFOs single word access
SPWR_CHAN_WR_DMA_PNTR_0	0x001C	// Channel 0 write DMA physical PCI dpr address
SPWR_CHAN_TX_FIFO_COUNT_0	0x001C	// Channel 0 transmit FIFO data count
SPWR_CHAN_RD_DMA_PNTR_0	0x0020	// Channel 0 read DMA physical PCI dpr address
SPWR_CHAN_RX_FIFO_COUNT_0	0x0020	// Channel 0 receive FIFO data count
SPWR_CHAN_TX/RX_PKT_LEN_0	0x0024	// Channel 0 Write TX/Read RX packet-length
SPWR_CHAN_TX_AMT_0	0x0028	// Channel 0 TX almost empty level
SPWR_CHAN_RX_AFL_0	0x002C	// Channel 0 RX almost full level
SPWR_CHAN_CNTRL_1	0x0030	// Channel 1 Control register
SPWR_CHAN_STATUS_1	0x0034	// Channel 1 Status register
SPWR_CHAN_FIFO_1	0x0038	// Channel 1 TX/RX FIFOs single word access
SPWR_CHAN_WR_DMA_PNTR_1	0x003C	// Channel 1 write DMA physical PCI dpr address
SPWR_CHAN_TX_FIFO_COUNT_1	0x003C	// Channel 1 transmit FIFO data count
SPWR_CHAN_RD_DMA_PNTR_1	0x0040	// Channel 1 read DMA physical PCI dpr address
SPWR_CHAN_RX_FIFO_COUNT_1	0x0040	// Channel 1 receive FIFO data count
SPWR_CHAN_TX/RX_PKT_LEN_1	0x0044	// Channel 1 Write TX/Read RX packet-length
SPWR_CHAN_TX_AMT_1	0x0048	// Channel 1 TX almost empty level
SPWR_CHAN_RX_AFL_1	0x004C	// Channel 1 RX almost full level
SPWR_CHAN_CNTRL_2	0x0050	// Channel 2 Control register
SPWR_CHAN_STATUS_2	0x0054	// Channel 2 Status register
SPWR_CHAN_FIFO_2	0x0058	// Channel 2 TX/RX FIFOs single word access
SPWR_CHAN_WR_DMA_PNTR_2	0x005C	// Channel 2 write DMA physical PCI dpr address
SPWR_CHAN_TX_FIFO_COUNT_2	0x005C	// Channel 2 transmit FIFO data count
SPWR_CHAN_RD_DMA_PNTR_2	0x0060	// Channel 2 read DMA physical PCI dpr address
SPWR_CHAN_RX_FIFO_COUNT_2	0x0060	// Channel 2 receive FIFO data count
SPWR_CHAN_TX/RX_PKT_LEN_2	0x0064	// Channel 2 Write TX/Read RX packet-length
SPWR_CHAN_TX_AMT_2	0x0068	// Channel 2 TX almost empty level
SPWR_CHAN_RX_AFL_2	0x006C	// Channel 2 RX almost full level
SPWR_CHAN_CNTRL_3	0x0070	// Channel 3 Control register
SPWR_CHAN_STATUS_3	0x0074	// Channel 3 Status register
SPWR_CHAN_FIFO_3	0x0078	// Channel 3 TX/RX FIFOs single word access
SPWR_CHAN_WR_DMA_PNTR_3	0x007C	// Channel 3 write DMA physical PCI dpr address
SPWR_CHAN_TX_FIFO_COUNT_3	0x007C	// Channel 3 transmit FIFO data count
SPWR_CHAN_RD_DMA_PNTR_3	0x0080	// Channel 3 read DMA physical PCI dpr address
SPWR_CHAN_RX_FIFO_COUNT_3	0x0080	// Channel 3 receive FIFO data count
SPWR_CHAN_TX/RX_PKT_LEN_3	0x0084	// Channel 3 Write TX/Read RX packet-length
SPWR_CHAN_TX_AMT_3	0x0088	// Channel 3 TX almost empty level
SPWR_CHAN_RX_AFL_3	0x008C	// Channel 3 RX almost full level

FIGURE 3 SPACEWIRE ADDRESS MAP

Register Definitions

SPWR_BASE_CNTL

[0x0000] Base Control Register (read/write)

Base Control Register

Data Bit	Description
31	BigEndianDma
30-26	Spare
25-24	Time-Code Control Flags
23	PLL Sdata Output
22	Spare
21	PLL Sclk Output
20	PLL Enable
19-15	IO Clock D Initial Divisor
14-10	IO Clock C Initial Divisor
9-5	IO Clock B Initial Divisor
4-0	IO Clock A Initial Divisor

FIGURE 4 SPACEWIRE BASE CONTROL REGISTER

All bits are active high and are reset on system power-up or reset, except PLL enable, which defaults to enabled (high) on power-up or reset.

IO Clock A-D Initial Divisor: These four fields determine the divisor used to generate the 10 MHz connection clock rate. IO clock A is used for channel 0, IO clock B is used for channel 1, IO clock C is used for channel 2 and IO clock D is used for channel 3. Depending on the frequencies programmed into the PLL, different divisors are required to achieve the 10 MHz bit-rate required by the SpaceWire specification for establishing the link connection. These fields specify those divisors. The frequency divisor is actually one more than the value entered. A value of zero corresponds to a divisor of one. A value of one corresponds to a divisor of two etc. If the Dynamic Engineering device driver is used, these values are written automatically when the PLL is programmed.

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

PLL Sclk/Sdata Output: These signals are used to program the PLL over the I2C serial

interface. Sclk is always an output whereas Sdata is bi-directional. This is where the output value is specified. When Sdata is an input it is read from the user switch port. Please note that the reference clock rate for the PLL is 40 MHz.

Time-Code Control Flags: These two bits are added to the six-bit time count in bit positions 7 and 6. Their purpose is currently not defined in the SpaceWire specification.

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

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

SPWR_USER_SWITCH

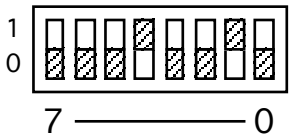
[0x0004] User Switch Port - read only

Dipswitch Port

Data Bit	Description
31-28	Spare
27	Channel 3 Interrupt Active
26	Channel 2 Interrupt Active
25	Channel 1 Interrupt Active
24	Channel 0 Interrupt Active
23	PLL Sdata Input
22-21	External Memory Configuration
20	MSI latched error (rev. H only)
19-17	K Minor Revision Number
16	MSI enabled (rev. H only)
15-13	Xilinx Design Configuration Type
12-8	Xilinx Design Revision Number
7-0	Switch Setting

FIGURE 5 SPACEWIRE USER SWITCH PORT

Switch 7-0: The user switch is read through this port. The bits are read as the lowest byte. Access the read-only port as a long word and mask off the undefined bits. The dip-switch positions are defined in the silkscreen. For example, the switch figure below indicates a 0x12.



MSI enabled: (rev. H only) A one indicates that MSI interrupts will be used; a zero indicates legacy interrupts will be used.

K Minor Revision Number: Three bits to indicate K revision number.

MSI latched error: (rev. H only) A one indicates that an error occurred when the interrupt message was written on the PCI bus. This bit is latched and is cleared by writing a one to bit 20.

External Memory Configuration: Two bits that indicate how the 256K bytes of external FIFO memory is configured for use.
0 => Spare.
1 => STD: Standard configuration, external memory not used.
2 => 128: 128K bytes allocated for both the CH0 RX and CH0 TX Data paths.
3 => 128RX: 128K bytes allocated to CH0 RX and CH1 RX Data paths.

PLL Sdata Input: The PLL_Sdata bi-directional line is read using this bit. This line is used to read the register contents of the PLL.

Channel 0-3 Interrupt Active: (Added in rev. I) when a one is read, it indicates that the corresponding channel's interrupt is active. When a zero is read, that interrupt is inactive.

Xilinx Design Configuration Type and Xilinx Design Revision Number: The value of the second byte of this port describes the channel configuration and revision of the Xilinx design. Beginning with rev. E, a new numbering scheme has been put in place to accomplish this. The upper 3 bits specify the Xilinx part and channel configuration:

Rev. J and later:
0 => XC3S1500 with 4 Kbyte internal data FIFOs and 2 G byte maximum packet-lengths for all channels.

1 => XC3S1500 with 128 Kbyte external transmit and receive data FIFOs for channel 0 and four Kbyte internal data FIFOs for channels 1-3. Two G byte maximum packet-lengths for all channels.

2 => XC3S1500 with Four Kbyte internal transmit data FIFOs and 128 Kbyte external receive data FIFOs for channels 0 and 1. Four Kbyte internal data FIFOs for channels 2 and 3. Two G byte maximum packet lengths for all channels.

3 => XC3S2000 with internal Four Kbyte data FIFOs and Two G byte maximum packet-lengths for all channels.

4 => XC3S2000 with 128 Kbyte external transmit and receive data FIFOs for channel 0 and Four Kbyte internal data FIFOs for channels 1 – 3. Two G byte maximum packet-lengths for all channels.

5 => No longer used.

6 => XC3S2000 with four Kbyte internal transmit data FIFOs and 128 Kbyte external receive data FIFOs for channels 0, 1. Four Kbyte internal data FIFOs for channels 2, 3. Two G byte maximum packet-lengths for all channels.

7 => XC6SLX100 All versions: internal FIFOs, external transmit and receive data FIFOs for channel 0 or external receive data FIFOs for channels 0 and 1.

The lower 5 bits are the revision of the design which has been standardized for all configurations and is currently at revision K (0x0B).

Prior to rev. J

0 => XC3S1000 with four Kbyte internal data FIFOs and 128 Kbyte maximum packet-lengths for all channels.

1 => XC3S1000 with 128 Kbyte external transmit and receive data FIFOs and 2 Gbyte maximum packet lengths for channel 0 and four Kbyte internal data FIFOs and 128 Kbyte maximum packet lengths for channels 1 – 3.

2 => XC3S1000 with four Kbyte internal transmit data FIFOs and 128 Kbyte external receive data FIFOs and 2 Gbyte maximum packet lengths for channels 0, 1 and internal 4 Kbyte data FIFOs and 128 Kbyte maximum packet lengths for channels 2, 3.

3 => XC3S1500 with internal 4 Kbyte data FIFOs and 2 Gbyte maximum packet-lengths for all channels.

4 => XC3S1500 with 128 Kbyte external transmit and receive data FIFOs for channel 0 and 4 Kbyte internal data FIFOs for channels 1 – 3 and 2 Gbyte maximum packet-lengths for all channels.

5 => XC3S1500 with four Kbyte internal transmit data FIFOs and 128 Kbyte external receive data FIFOs and two Gbyte maximum packet lengths for channels 0, 1 and four Kbyte internal data FIFOs and 128 Kbyte maximum packet lengths for channels 2, 3.

6 => XC3S1500 with four Kbyte internal transmit data FIFOs and 128 Kbyte external receive data FIFOs for channels 0, 1 and four Kbyte internal data FIFOs for channels 2, 3 and 2 Gbyte maximum packet-lengths for all channels.

SPWR_TIME_CNTRL

[0x0008] Time Control Register (read/write)

Time Control Register

Data Bit	Description
31-29	Channel 3 Time-Code Source Control
28-26	Channel 2 Time-Code Source Control
25-23	Channel 1 Time-Code Source Control
22-20	Channel 0 Time-Code Source Control
19-0	Master Timer Divider Count

FIGURE 6 SPACEWIRE TIME CONTROL REGISTER

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

Master Timer Divider Count: This count is used to generate the TICK_IN signal when a channel is used as the source for time-codes. The counter is clocked by the 80 MHz link clock and this count represents the count at which the counter resets to zero, increments the time-code and issues a TICK_IN signal. If this field is set to zero, the six-bit time-code counter and the 20-bit tick timer are reset to zero; when the count is set to a non-zero value, counting proceeds. The 20-bit counter allows a maximum time between ticks of approximately 13 milliseconds.

Channel 0-3 Time-Code Source Control: These four fields control the source of the time-code signals for the four channels as follows:

“000”-> Disabled: Time-code characters are not sent by this channel.

“001”-> Master: Time-code characters for this channel come from the master timer and time-code counter.

“010”-> Channel 0: Time-code characters received by channel 0 are subsequently sent by this channel.

“011”-> Channel 1: Time-code characters received by channel 1 are subsequently sent by this channel.

“100”-> Channel 2: Time-code characters received by channel 2 are subsequently sent by this channel.

“101”-> Channel 3: Time-code characters received by channel 3 are subsequently sent by this channel.

SPWR_RX_PKT_FF_FULL_CNTRL

[0x000C] RX Packet FIFO Full Control Register (read/write)

RX Packet FIFO Full Control Register	
Data Bit	**Description**
9-0	RX Packet FIFO Full Level

FIGURE 7 SPACEWIRE PACKET FIFO FULL CONTROL REGISTER

These bits are set to 0x3DF on system power-up or reset.

RX Packet FIFO Full Level: These bits set the RX Packet FIFO Full Threshold level. By default these bits are set to 0x3DF. The RX Packet FIFO has 1024 locations (0x3FF) therefore, by default, the logic will see the FIFO as full once 960 or more locations are occupied, once the threshold is reached (or passed) the logic will back pressure the Space Wire link by not providing additional FCT's until the number of FIFO locations drops below the threshold.

Per the Space Wire specification the maximum outstanding FCT's is 56. If single byte transfers are being used and the maximum FCT's are outstanding when the FIFO goes full, 56 additional packets maybe received, 0x3DF was chosen to be the default value to cover this case.

Setting this register to either 0x3FF or 0x000 puts this logic in legacy mode.

SPWR_CHAN_CNTRL_0-3

[0x0010, 0x0030, 0x0050, 0x0070] Channel Control Register (read/write)

Channel Control Register

Data Bit	Description
31	Read DMA Ready (read only)
30	Write DMA Ready (read only)
29-28	Time-Code Flags (read only)
27	Enable Reading Output DMA Word Count
26	Enable Reading Input DMA Word Count
25	Return Valid Packet-Lengths Only Enable
24	Transmit Packet-Length Repeat
23	Receive FIFO Programmable Level Load
22	Transmit FIFO Programmable Level Load
21	Read DMA Arbitration Priority Enable
20	Write DMA Arbitration Priority Enable
19	Read DMA Interrupt Enable
18	Write DMA Interrupt Enable
17	Force Interrupt
16	Master Interrupt Enable
15	Tick Received Interrupt Enable
14	Packet Received Interrupt Enable
13	RX Error Interrupt Enable
12	RX Almost Full Interrupt Enable
11	TX Almost Empty Interrupt Enable
10	Packet Disable
9	Link Auto-Start
8	Link Start
7	Link Enable
6	FIFO Loop-Back Enable
5	Receive FIFO Reset
4	Transmit FIFO Reset
3-0	IO Clock Divisor

FIGURE 8 SPACEWIRE CHANNEL CONTROL REGISTER

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

IO Clock Divisor: This field determines the divisor used to generate the operational

clock rate. The frequency divisor is actually one more than the value entered. A value of zero corresponds to a divisor of one, one corresponds to a divisor of two etc.

Transmit/Receive FIFO Reset: When one or both of these bits are set to a one, the corresponding data FIFO, packet-length FIFO and control and status circuitry will be reset. When these bits are zero, normal FIFO operation is enabled. FIFO resets are referenced to the PCI clock, two periods are required for proper reset.

FIFO Loop-Back Enable: 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 FIFOs without connecting to another SpaceWire node. When this bit is zero, normal operation is enabled.

Link Enable: When this bit is set to a one, the link connection process will be initiated. The connection state-machine will proceed to the Ready state until Start or Auto-Start is asserted. When this bit is zero, the link connection will be disabled.

Link Start: When this bit is set to a one, the link state-machine will move from the Ready state to the Started state and will attempt to establish a connection with another node. When this bit is zero, the state-machine will remain in the Ready state, provided it has already achieved this state. Once the state-machine has left the Ready state, this bit has no effect.

Link Auto-Start: The behavior of this bit is similar to Link start, however, when this bit is set and Link start is not set, the state-machine will not proceed to the Started state unless a Null character has been seen, which indicates that the other node is attempting to establish a connection. This bit allows the connection process to be cleanly initiated from one side of the link only.

Packet Disable: When this bit is set to a one, data is transferred without being separated into packets. No end-of-packet characters are generated or received and the packet-length FIFOs are not used. As soon as data is written to the transmit FIFO it will be sent out, provided all other conditions allow this. When this bit is zero, the data will be sent in packets. Data must be written to the transmit data FIFO and packet lengths must be written to the TX packet-length FIFO, before data can be transferred.

TX Almost Empty Interrupt Enable: When this bit is set to a one, an interrupt will be generated when the transmit FIFO level becomes equal or less than the value specified in the SPWR_CHAN_TX_AMT register, provided the channel master interrupt enable is asserted. When this bit is zero, an interrupt will not be generated, but the status can still be read from the channel status register.

RX Almost Full Interrupt Enable: When this bit is set to a one, an interrupt will be generated when the receive FIFO level becomes equal or greater to the value specified in the SPWR_CHAN_RX_AFL register, provided the channel master interrupt enable is asserted. When this bit is zero, an interrupt will not be generated, but the status can still be read from the channel status register.

RX Error Interrupt Enable: When this bit is set to a one, an interrupt will be generated when a receiver error condition is detected, provided the channel master interrupt enabled is asserted. When a zero is written to this bit, an interrupt will not be generated, but the latched status can still be read from the channel status register.

Packet Received Interrupt Enable: When this bit is set to a one, an interrupt will be generated when a complete packet is received, provided the channel master interrupt enable is asserted. When a zero is written to this bit, an interrupt will not be generated, but the latched status can still be read from the channel status register.

Tick Received Interrupt Enable: When this bit is set to a one, an interrupt will be generated when a valid time-code is received, provided the channel master interrupt enable is asserted. When a zero is written to this bit, an interrupt will not be generated, but the latched status can still be read from the Interrupt Status register.

Master Interrupt Enable: When this bit is set to a one all enabled interrupts for the referenced channel (except the DMA interrupts) will be gated through to the PCI host; when this bit is a zero, the interrupts can be used for status without interrupting the host.

Force Interrupt: When this bit is set to a one a system interrupt will occur provided the channel master interrupt enable is set. This is useful to test interrupt service routines.

Write/Read 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.

Write/Read DMA Arbitration Priority Enable: These two bits, when set to one, enable the DMA arbiter to use the TX almost empty and/or RX almost full status to give priority to a channel that is approaching the limits of its FIFOs. The levels written to the TX almost empty and RX almost full registers are used to determine these status values. When these bits are zero normal round-robin arbitration is used to determine access to the PCI bus for DMA transfers.

Transmit/Receive FIFO Programmable Level Load: These bits are only valid for channels with external data FIFOs. The load bits must be active during FIFO reset to select the programmable level feature. Once selected, these bits must be set back to zero for normal FIFO operation. When set to one, data accesses are instead directed to the almost empty and almost full level registers. See External FIFO Almost Empty/Almost Full Levels section for further details.

Transmit Packet-Length Repeat: When this bit is set to a one, the current transmit packet-length will be used continually until there is no more transmit data or this control bit is cleared. When a zero is written to this bit, the packet-length FIFO must be read to obtain a new packet-length for each transmitted packet.

Return Valid Packet-Lengths Only Enable: When this bit is set to a one, only valid packet-lengths will be returned. If no new packet has been received since the packet-length FIFO was read, the packet-length will be returned as zero. When a zero is written to this bit and no new packet has been received since the packet-length FIFO was read, the packet-length from the last packet received will be returned. When enabled, this control allows packet-lengths to be confidently read without first checking the Receive Packet Length Valid status bit in the channel status register. This control bit was added starting with rev. J.

Enable Reading Input / Output DMA Word Count: When either or both of these bits is set to one, all the other bits in this register remain unchanged and only the DMA word count function is affected. When both bits are set to one, the word counters in the DMA engines are reset. When only one bit is set to one, the corresponding DMA's word counter's output is enabled. A subsequent read of the channel control register will output the 30-bit word count from that DMA engine and will clear the state of the DMA word count function. The counts are cleared when a new DMA command is posted to the respective DMA engine or when explicitly cleared as described above, but are not cleared when a transfer is cancelled before it is complete. This allows the user to discover how much data was actually transferred.

Time-Code Flags: The time-code flags have been moved to the control register to make room for the latched almost empty/full status bits that were added to the status register. These two read-only bits are currently undefined in the SpaceWire specification and will most likely always be seen as zeros.

Write/Read DMA Ready: These two read-only bits report the DMA state-machine status. If they are read as a one, the corresponding DMA state-machine is idle and available to start a transfer. If the bits are read as a zero, the corresponding DMA state-machine is processing a data transfer.

SPWR_CHAN_STATUS_0-3

[0x0014, 0x0034, 0x0054, 0x0074] (Status read/Latch clear write)

Channel Status Register

Data Bit	Description
31	Latched Receive FIFO Almost Full
30	Latched Transmit FIFO Almost Empty
29-24	Time-Code Data
23	Interrupt Active
22	Receive Packet Length Valid
21	Transmit Purge Error (Added in rev. F)
21	Transmit Packet Length Queue Full (<= rev. E)
20	SpaceWire Link Established
19	Read DMA Error
18	Write DMA Error
17	Read DMA List Complete
16	Write DMA List Complete
15	TICK_OUT Received
14	Packet Received
13	Receive Error
12	Receive FIFO Overflow
11	Credit Error Detected
10	Escape Error Detected
9	Disconnect Error Detected
8	Parity Error Detected
7	Receive Data Valid
6	Receive FIFO Full
5	Receive FIFO Almost Full
4	Receive FIFO Empty
3	Transmit Data Valid
2	Transmit FIFO Full
1	Transmit FIFO Almost Empty
0	Transmit FIFO Empty

FIGURE 9 SPACEWIRE CHANNEL STATUS REGISTER

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 SPWR_CHAN_TX_AMT 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.

Transmit Data Valid: When a one is read, there is at least one word of valid transmit data. When either the transmitter or the FIFO bypass is enabled, the first word written to the transmit FIFO will be read to be available to the transmitter or the FIFO bypass circuit. Therefore although the FIFO is empty, if this bit is set, there is one additional long-word of transmit data. A zero indicates that there is no valid transmit data.

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.

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 SPWR_CHAN_RX_AFL 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.

Receive Data Valid: When a one is read, there is at least one word of valid receive data. When data is written to the receive FIFO, the first four words are read and held in latches to be ready for a PCI read DMA or single-word read. Therefore although the FIFO is empty, if this bit is set, there are as many as four additional long-words of receive data. A zero indicates that there is no valid receive data.

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.

Disconnect Error Detected: When a one is read, it indicates that a disconnect 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 disconnect error has occurred. A disconnect error occurs when there is no activity on the data or strobe line for 850 nanoseconds.

Escape Error Detected: When a one is read, it indicates that an escape 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 escape error has occurred. An escape error occurs when an escape character is followed by anything other than an FCT or a data character.

Credit Error Detected: When a one is read, it indicates that a credit error 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 credit error has occurred. A credit error is a violation of the SpaceWire data flow-control protocol.

Receive FIFO Overflow: When a one is read, it indicates that either an attempt has been made to write data to a full receive data FIFO or a packet-length value to a full packet-length FIFO. Neither of these conditions should occur if the data flow-control protocol is functioning correctly. This bit is latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no overflow condition has occurred.

Receive Error: When a one is read, it indicates that one of the five preceding error conditions has been detected since this bit 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 receive error has occurred.

Packet Received: When a one is read, it indicates that a packet has been received since this bit 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 a packet has not been received.

TICK_OUT Received: When a one is read, it indicates that a valid time code has been received by the referenced channel since this bit 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 a valid time-code has not been received.

Write/Read DMA List Complete: When a one is read, it indicates that the corresponding DMA has completed. These bits are latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that the corresponding DMA has

not completed.
Write/Read DMA Error: When a one is read, it indicates that an error has occurred while the corresponding DMA was in progress. This could be a target or master abort or an incorrect direction bit in one of the DMA descriptors. These bits are latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no DMA error has occurred.

SpaceWire Link Established: A one indicates that the referenced link is in the run state, which allows all types of link and normal characters to be exchanged. A zero indicates that the link is not in the run state.

Transmit Purge Error: (Added in rev. F) When a one is read, it indicates that a connection error occurred while a transmit packet was in progress and the transmitter was unable to completely delete the remainder of the packet's data from the transmit data FIFO. This can occur if the packet-length was longer than 4 Kbytes or the packet data had not been all written to the transmit data FIFO. The link connection state-machine allows a 30 microsecond delay for purging transmit data after a link error. At 132 Mbytes/second this allows almost 4 Kbytes of data to be discarded. This bit is latched and must be cleared by writing the same bit back to the channel status port. A zero indicates that no transmit purge error has occurred.

TX packet Length Queue Full: (rev. E or earlier: rev. F adds a TX packet-length FIFO count which supersedes this bit) When a one is read, the transmit packet-length FIFO for the corresponding channel is full; when a zero is read, there is room for at least one more packet-length value in the FIFO.

RX packet Length Valid: When a one is read, there is at least one valid receive packet-length value available. When this bit is a zero, it indicates that there are no valid receive packet-length values.

Interrupt Active: When a one is read, it indicates that an enabled interrupt condition (other than the DMA interrupts) is active for the referenced channel. A zero indicates that no enabled interrupt condition is active.

Time-Code Data: The last time-code value received can be read from this six-bit data-field. The TICK_OUT received status bit will indicate if the data is a new valid time-code value. A time-code is considered valid if it is one more than the previous stored value. If the time-code is the same as the stored value, it is assumed to be a duplicate resulting from a cycle in the SpaceWire network and is therefore ignored. If the time-code meets neither of these conditions, it is stored, but the TICK_OUT signal is not asserted until the next time-code is received and is one more than that last stored value. At this point the time-code is deemed to be re-synchronized.

Latched Receive FIFO Almost Full: When a one is read, it indicates that the receive FIFO data count has become greater than or equal to the value in the SPWR_CHAN_RX_AFL register. A zero indicates that the FIFO has not become almost full. This bit is latched and can be cleared by writing back to the Status register with a one in this bit position.

Latched Transmit FIFO Almost Empty: When a one is read, it indicates that the transmit FIFO data count has become less than or equal to the value in the SPWR_CHAN_TX_AMT register. A zero indicates that the FIFO has not become almost empty. This bit is latched and can be cleared by writing back to the Status register with a one in this bit position.

SPWR_CHAN_FIFO_0-3

[0x0018, 0x0038, 0x0058, 0x0078] Write TX/Read RX FIFO Ports

RX and TX FIFO Ports	
Data Bit	**Description**
31-0	FIFO Data-Word

FIGURE 10 SPACEWIRE CHANNEL RX/TX FIFO PORTS

These ports are used for single-word accesses into the channel TX FIFO and out of the channel RX FIFO.

SPWR_CHAN_WR_DMA_PNTR_0-3

[0x001C, 0x003C, 0x005C, 0x007C] Write only

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

FIGURE 11 SPACEWIRE CHANNEL WRITE DMA POINTER PORT

This write-only port is used to initiate a scatter-gather input DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads three successive long words beginning at that address. The first is the address of the first memory block of the DMA buffer, the second is the length in bytes of that block (only the lower 22 bits are valid), and the third is the address of the next chaining descriptor in the list of buffer memory blocks. This process is continued until the end-of-chain bit is set in one of the next pointer values. Writing a zero to this port will abort a write DMA in progress.

The segment size is 21-0 however the byte count is shifted down two positions to create a LW count in memory. This means the actual count is 1M-1 [4Mbytes minus 1 LW] or x3FFFFC for the max byte count.

SPWR_CHAN_TX_FIFO_COUNT_0-3

[0x001C, 0x003C, 0x005C, 0x007C] TX FIFO Data Count (read only)

TX FIFO Data Count

Channels with External TX Data FIFOs

Data Bit	Description
31, 30	Spare
29-20	Number of TX Packet-Length Values Stored
19, 18	Spare
17-0	Number of TX Data-Words Stored

Channels with Internal TX Data FIFOs

31, 30	Spare
29-20	Number of TX Packet-Length Values Stored
19-11	Spare
10-0	Number of TX data-Words Stored

FIGURE 12 SPACEWIRE CHANNEL TX FIFO DATA COUNT PORT

These read-only register ports report the number of 32-bit data words in the corresponding transmit FIFO. There is an additional latch that may contain data which allows this value to be a maximum of 0x401 for channels with internal transmit data FIFOs or 0x20001 for a channel with an external transmit data FIFO.
Starting with rev. F, a transmit packet-length count field has been added to this register. This allows the user to know how many packet-length values can be safely written to the packet-length FIFO without causing an overflow. It also aids in error recovery if there was a connection error during a packet transmission and the transmitter is unable to automatically purge the remaining data from the impacted packet. In this case the user must intervene to reestablish a consistent packet/data state by resetting the transmit FIFO and rewriting packet data and packet-length values. Knowing how many packets were pending when the error occurred will greatly aid in this procedure.

SPWR_CHAN_RD_DMA_PNTR_0-3

[0x0020, 0x0040, 0x0060, 0x0080] Write only

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

FIGURE 13 SPACEWIRE CHANNEL READ DMA POINTER PORT

This write-only port is used to initiate a scatter-gather output DMA. When the address of the first chaining descriptor is written to this port, the DMA engine reads three successive long words beginning at that address. The first is the address of the first memory block of the DMA buffer, the second is the length in bytes of that block (only the lower 22 bits are valid), and the third is the address of the next chaining descriptor in the list of buffer memory blocks. This process is continued until the end-of-chain bit is set in one of the next pointer values. Writing a zero to this port will abort a read DMA in progress.

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

The segment size is 21-0 however the byte count is shifted down two positions to create a LW count in memory. This means the actual count is 1M-1 [4Mbytes minus 1 LW] or x3FFFFC for the max byte count.

SPWR_CHAN_RX_FIFO_COUNT_0-3

[0x0020, 0x0040, 0x0060, 0x0080] RX FIFO Data Count (read only)

RX FIFO Data Count

Channels with External RX Data FIFOs

Data Bit	Description
31, 30	Spare
29-20	Number of RX Packet-Length Values Stored
19, 18	Spare
17-0	Number of RX Data-Words Stored

Channels with Internal RX Data FIFOs

31, 30	Spare
29-20	Number of RX Packet-Length Values Stored
19-11	Spare
10-0	Number of RX Data-Words Stored

FIGURE 14 SPACEWIRE CHANNEL RX FIFO DATA COUNT PORT

These read-only register ports report the number of 32-bit data words in the corresponding receive FIFO. There are four additional latches in the read DMA pipeline that may contain data, which allows this value to be a maximum of 0x404 for channels with internal receive data FIFOs or 0x20004 for a channel with an external receive data FIFO.

Starting with rev. G, a receive packet-length count field has been added to this register. This allows the user to know how many packet-length values are stored in the packet-length FIFO. The first receive packet-length written to the packet-length FIFO is read as soon as it is written to be immediately available. This causes the receive packet-length valid status bit to be set, and the packet-length count to become one. As subsequent packets are received, the count will increment and as lengths are read, the count will decrement.

SPWR_CHAN_TX_PKT_LEN_0-3

[0x0024, 0x0044, 0x0064, 0x0084] TX Packet length FIFO Ports (write only)

TX Packet Length FIFO Ports

Channels with 2 Gbyte Maximum Packet-Lengths	
Data Bit	**Description**
31	Terminate Packet with an Error-End-of-Packet
30-0	Packet Length (31 bits)
Channels with 128 Kbyte Maximum Packet-Lengths	
31-18	Unused
17	Terminate Packet with an Error-End-of-Packet
16-0	Packet Length (17 bits)

FIGURE 15 SPACEWIRE TX PACKET LENGTH FIFO PORTS

SPWR_CHAN_RX_PKT_LEN_0-3

[0x0024, 0x0044, 0x0064, 0x0084] RX Packet length FIFO Ports (read only)

RX Packet Length FIFO Ports

Channels with 2 Gbyte Maximum Packet-Lengths	
Data Bit	**Description**
31	Connection Error or Error-End-of-Packet
30-0	Packet Length (31 bits)
Channels with 128 Kbyte Maximum Packet-Lengths	
31-18	Unused
17	Connection Error or Error-End-of-Packet
16-0	Packet Length (17 bits)

FIGURE 16 SPACEWIRE RX PACKET LENGTH FIFO PORTS

These ports access the write transmit packet-length FIFO and the read receive packet-length FIFO ports for the respective channels. These FIFOs are used to store packet lengths for sending transmit packets and reading received packets.

Beginning with the rev. D firmware (rev. A for the 128RX), the packet-length fields have been increased to 17 bits (131,071 byte maximum packet-length) for channels with internal data FIFOs and 31 bits (2,147,483,647 byte maximum packet-length) for channels with at least one external data FIFO.

Beginning with the rev. E firmware the packet-length fields have been increased to 31 bits (2,147,483,647 byte maximum packet-length) for all channels in some configurations (see the description of Xilinx design configuration types on page 24).

The bit above the packet-length field is an error flag. If this bit is read as a one, it indicates that either the packet was terminated with an EEP (Error-End-of-Packet) or an error condition occurred while the packet was being received. If this bit is written as a one, it indicates that the transmit packet should be terminated with an EEP rather than an EOP.

SPWR_CHAN_TX_AMT_0-3

[0x0028, 0x0048, 0x0068, 0x0088] TX Almost-Empty Level (read/write)

TX Almost Empty Level Register

Data Bit	Description
31-0	TX FIFO Almost-Empty Level

FIGURE 17 SPACEWIRE CHANNEL TX ALMOST EMPTY LEVEL REGISTER

These read/write ports access the transmitter almost-empty level registers for the respective channels. When the number of data words in the transmit data FIFO is equal or less than this value, the almost empty status bit is set and an interrupt may be generated if it is enabled. Only 11 bits are needed for internal data FIFOs.

SPWR_CHAN_RX_AFL_0-3

[0x002C, 0x004C, 0x006C, 0x008C] RX Almost-Full Level (read/write)

RX Almost Full Level Register

Data Bit	Description
31-0	RX FIFO Almost-Full Level

FIGURE 18 SPACEWIRE CHANNEL RX ALMOST FULL LEVEL REGISTER

These read/write ports access the receiver almost-full level registers for the respective channels. When the number of data words in the receive data FIFO is equal or greater than this value, the almost full status bit is set and an interrupt may be generated if it is enabled. Only 11 bits are needed for internal data FIFOs.

External FIFO Almost Empty/Almost Full Levels

To enhance system performance Dynamic Engineering optionally offers 1024K [2x128Kx32] bytes of external memory to provide an additional FIFO storage for two channels data path(s). There are two configurations for customers to select from called 128 and 128RX. Two bits [22:21] in the SPWR_USER_SWITCH register to indicate the external memory configuration of the board, which are:

0 => Spare.
1 => STD: Standard configuration, external memory not used.
2 => 128: 512K bytes allocated for CH0 RX and CH0 TX Data paths.
3 => 128RX: 512K bytes allocated to CH0 RX and CH1 RX Data paths.

Note: Previous to revision K minor rev1 the driver would detect if external memory was present (and its size) then configure the internal and external FIFO memory logic Almost Full and Almost Empty indicators.

To set the programmable FIFO levels for external FIFOs follow these steps:

1) Assert Rx/Tx FIFO Reset bits – SPWR_CHAN_CNTRL bits [5:4].

2) De-assert Rx/Tx FIFO Reset bits [5:4] and assert Tx/Rx FIFO Programmable Level Load bits [23:22] during same write of SPWR_CHAN_CNTRL register.

3) Write the SPWR_CHAN_TX_AMT register with almost empty threshold.

4) Write the SPWR_CHAN_RX_AFL register with almost full threshold.

5) Assert Rx/Tx FIFO Reset bits [5:4] and assert Tx/Rx FIFO Programmable Level Load bits [23:22] during same write of SPWR_CHAN_CNTRL register.

6) De-assert Rx/Tx FIFO Reset bits [5:4] and de-assert Tx/Rx FIFO Programmable Level Load bits [23:22] during same write of SPWR_CHAN_CNTRL register.

Notes: The first write accesses the almost empty register and the second accesses the almost full register (subsequent accesses repeat this in circular fashion); The FIFO used is an IDT 72v361XX X36 - see datasheet for more details if desired.

The Dynamic Engineering device drivers handle all the complexities of programming the external FIFO almost full/empty registers.

SpaceWire I/O Channel Pin Assignment

MDM Connectors

J2

DIN 0 +	1	
DIN 0 -		6
SIN 0 +	2	
SIN 0 -		7
SHIELD0	3	
SOUT 0 +		8
SOUT 0 -	4	
DOUT 0 +		9
DOUT 0 -	5	

J3

DIN 1 +	1	
DIN 1 -		6
SIN 1 +	2	
SIN 1 -		7
SHIELD1	3	
SOUT 1 +		8
SOUT 1 -	4	
DOUT 1 +		9
DOUT 1 -	5	

J4

DIN 2 +	1	
DIN 2 -		6
SIN 2 +	2	
SIN 2 -		7
SHIELD2	3	
SOUT 2 +		8
SOUT 2 -	4	
DOUT 2 +		9
DOUT 2 -	5	

J5

DIN 3 +	1	
DIN 3 -		6
SIN 3 +	2	
SIN 3 -		7
SHIELD3	3	
SOUT 3 +		8
SOUT 3 -	4	
DOUT 3 +		9
DOUT 3 -	5	

FIGURE 19 MDM I/O CONNECTOR PINOUTS

J#'s are consistent across boards – if an MDM is installed it will be numbered as shown.

PMC PCI Pn1 Interface Pin Assignment

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

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

FIGURE 20 (CC)PMC-SPACEWIRE PN1 INTERFACE

PMC PCI Pn2 Interface Pin Assignment

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

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

FIGURE 21 (CC)PMC-SPACEWIRE PN2 INTERFACE

PMC Pn4 User Interface Pin Assignment

The figure below provides the pin assignments for the PMC-SpaceWire Module routed to Pn4. Also, see the User Manual for your carrier board for information on interfacing with Pn4. Ports 0-2 are only implemented on PMC Pn4 when the rear IO ordering option is selected.

DOUTA_0-	DINA_0-	1	2
DOUTA_0+	DINA_0+	3	4
GND REF		5	6
GND REF		7	8
SOUTA_0-	SINA_0-	9	10
SOUTA_0+	SINA_0+	11	12
		13	14
		15	16
DOUTA_1-	DINA_1-	17	18
DOUTA_1+	DINA_1+	19	20
GND REF		21	22
GND REF		23	24
SOUTA_1-	SINA_1-	25	26
SOUTA_1+	SINA_1+	27	28
		29	30
		31	32
DOUTA_2-	DINA_2-	33	34
DOUTA_2+	DINA_2+	35	36
GND REF		37	38
GND REF		39	40
SOUTA_2-	SINA_2-	41	42
SOUTA_2+	SINA_2+	43	44
		45	46
		47	48
DOUTA_3-	DINA_3-	49	50
DOUTA_3+	DINA_3+	51	52
GND REF		53	54
GND REF		55	56
SOUTA_3-	SINA_3-	57	58
SOUTA_3+	SINA_3+	59	60
		61	62
		63	64

FIGURE 22 (CC)PMC-SPACEWIRE PN4 INTERFACE

Applications Guide

Interfacing

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

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

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

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

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

Construction and Reliability

Dynamic Engineering Modules are conceived and engineered for rugged industrial environments. The SpaceWire family is constructed out of 0.062-inch thick High-Temp ROHS compliant FR4 material.

ROHS and standard processing are available options.

Through-hole and surface-mount components are used. PMC connectors are rated at 1 Amp per pin, 100 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable. The PCI gold fingers are gold over nickel for high reliability and long lasting connections. PC104p stacking connectors are mounted in accordance with the manufacturers specifications and using hard gold plated mounting holes for reliable connections.

PMC's and ccPMC's are secured against the carrier with four screws attached to the 2 stand-offs and 2 locations on the front panel. The four screws provide significant protection against shock, vibration, and incomplete insertion. PC104p are stacked and retained with inter-module stand-offs and mounting hardware. PCI cards can be retained with bezel mount screws and internal card guides.

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

The ccPMC version has internal thermal planes which are isolated from the electrical planes and tied to the thermal strips where the carrier conduction cooling webs are mounted. The exposed surfaces are gold plated for superior contact and thermal transfer with the web. The components are Industrial temperature rated or better. Thermal vias are added under components to tie in with the thermal plane directly.

The PC104p version of the design has the ground plane tied in with the mounting hardware to allow for inter-stack cooling in conduction cooled environments. Air cooling is also a viable method.

Thermal Considerations

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

Warranty and Repair

Please refer to the warranty page on our website for the current warranty offered and options. http://www.dyneng.com/warranty.html

Service Policy

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering Products not purchased directly from Dynamic Engineering contact your reseller. Products returned to Dynamic Engineering for repair by other than the original customer will be treated as out-of-warranty.

Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

For Service Contact:

Customer Service Department
Dynamic Engineering
150 Dubois Street, Suite C
Santa Cruz, CA 95060
831-457-8891
831-457-4793 fax
support@dyneng.com

Specifications

Formats:	ccPMC, PMC, PCI, PC104p, PCI-104
Host Interface (PCI):	PCI Interface 33 MHz. 32-bit
Serial Interfaces:	Four SpaceWire channels
TX Bit-rates generated:	2 – 180 MHz for each SpaceWire channel
Software Interface:	Control Registers, FIFOs, and Status Ports
Initialization:	Hardware reset forces all registers to 0 except as noted
Access Modes:	Long-word boundary space (see memory map)
Wait States:	One for all addresses
Interrupt:	Each channel has an interrupt for TX almost-empty, Rx almost-full, Time-code received, Rx packet done, and Rx error. Read and write DMA interrupts are also implemented for each channel.
DMA:	Independent input and output Scatter/Gather DMA Support implemented for each channel
Onboard Options:	All Options are Software Programmable
Interface Options (PMC):	Three 9-pin MDM connectors for channel 0 – 2, channel 3 available on Pn4 only. All channels can be strapped to interface with Pn4 only if desired

Interface Options (PCI/PC104p): Four 9-pin MDM connectors for channel 0 – 3 only

Interface Options (ccPMC) Pn4

Dimensions ((cc)PMC):	Standard Single (cc)PMC Module
Dimensions (PCI):	Dimensions: 6.600 inches by 4.200 inches
Dimensions (PC104p)	Standard PC104p or PCI-104 card

Construction:	High Temp ROHS compliant FR4 Multi-Layer Printed Circuit, Through-Hole and Surface-Mount Components
Temperature Coefficient:	2.17 W/^{o}C for uniform heat across PMC [similar for other formats]
Power	5V & 3.3V required

Order Information

Please refer to our SpaceWire webpage for the most up to date information:
http://www.dyneng.com/spacewire.html

PMC-SpaceWire	http://www.dyneng.com/pmc_SpaceWire.html Standard version with two 4KB FIFOs per channel, standard SpaceWire [ECSS-E-50-12C] timing and protocol. Three channels through the Bezel and 1 on Pn4 Industrial Temperature STD for “05” and later PCB revision.
PCI-SpaceWire	http://www.dyneng.com/pci_SpaceWire.html Standard version with two 4KB FIFOs per channel, standard SpaceWire [ECSS-E-50-12C] timing and protocol. Four channels through the Bezel. Industrial Temperature STD for “03” and later PCB revision.
PCIe-SpaceWire	http://www.dyneng.com/PCIe-SpaceWire.html Standard version with two 4KB FIFOs per channel, standard SpaceWire [ECSS-E-50-12C] timing and protocol. Four channels through the Bezel. Industrial Temperature.
PCI-104-SpaceWire	http://www.dyneng.com/pc104p_SpaceWire.html Standard version with two 4KB FIFOs per channel, standard SpaceWire [ECSS-E-50-12C] timing and protocol. Four channels with MDM connectors. No ISA connector installed.
PC104p-SpaceWire	http://www.dyneng.com/pc104p_SpaceWire.html Standard version with two 4KB FIFOs per channel, standard SpaceWire [ECSS-E-50-12C] timing and protocol. Four channels with MDM connectors. ISA connector installed for pass through only.
ccPMC-SpaceWire	http://www.dyneng.com/ccPmcSpaceWire.html Standard version with two 4KB FIFOs per channel, standard SpaceWire [ECSS-E-50-12C] timing and protocol. Four channels through Pn4. Standard with industrial temperature components. Conduction Cooled.

Options:

-ET	Add Industrial temp components to PCI-104 and PC104p models
-ROHS	Add ROHS compliant processing. Please note: Standard leaded processing will be used if this option is not selected.
-128	Add external 128Kx32 FIFO's to RX and TX on channel 0.
-128Rx	Add external 128Kx32 FIFO's to RX on channel 0 and channel 1.
-Pn4	Add for all channels through rear connector [Pn4] PMC model only
-RHF	Add Rad Hard FLASH – ccPMC model only
Software	For SpaceWire clients: Software Driver, sample application. Please specify Windows®, Linux included. VxWorks requires a onetime/project fee.
MDMCable9	9-pin MDM connectors (2) - four shielded twisted pairs. Standard and custom lengths are available.
DESWBO	http://www.dyneng.com/deswbo.html Dynamic Engineering SpaceWire BreakOut [monitor / Debugger] with LED's for traffic status, headers to work with scope or analyzer for packet counts etc. Pass through configuration to monitor complete traffic between nodes.
DESWCB	http://www.dyneng.com/deswcb.html Dynamic Engineering SpaceWire Connector Board – multi-channel cable to single channel cable adapter. Up to 28 MDM connectors can be mounted – specify number when ordering – Plastic housing. Route multi-channel cable to side or rear of housing and break out in the front. Matched length traces, with strain relief for easy mounting of cable.