

PRELIMINARY

PMC-SIO4AR

User's Manual

**QUAD CHANNEL MULTI-PROTOCOL
SERIAL CONTROLLER
WITH DEEP TRANSMIT AND RECEIVE FIFOS
PMC REAR I/O INTERFACE**

PRELIMINARY

**General Standards Corporation
8302A Whitesburg Drive
Huntsville, AL 35802
Phone: (256) 880-8787
Fax: (256) 880-8788
URL: www.generalstandards.com
E-mail: techsupport@generalstandards.com**

Revision A

PRELIMINARY

PMC-SIO4AR Documentation History

1. Rev A - April 2003 – Original rev from PMC-SIO4A manual.

PRELIMINARY

PREFACE

Copyright © 2003 General Standards Corporation

Additional copies of this manual or other **General Standards Corporation** literature may be obtained from:

General Standards Corporation
8302A Whitesburg Drive
Huntsville, Alabama 35802
Telephone: (256) 880-8787
Fax: (256) 880-8788
URL: www.generalstandards.com

The information in this document is subject to change without notice.

General Standards Corporation makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Although extensive editing and reviews are performed before release to ECO control, **General Standards Corporation** assumes no responsibility for any errors that may exist in this document. No commitment is made to update or keep current the information contained in this document.

General Standards Corporation does not assume any liability arising out of the application or use of any product or circuit described herein, nor is any license conveyed under any patent right of any rights of others.

General Standards Corporation assumes no responsibility resulting from omissions or errors in this manual, or from the use of information contained herein.

General Standards Corporation reserves the right to make any changes, without notice, to this product to improve reliability, performance, function, or design.

All rights reserved

No parts of this document may be copied or reproduced in any form or by any means without prior written consent of **General Standards Corporation**.

PRELIMINARY

RELATED PUBLICATIONS

ZILOG Z16C30 USC® User's Manual

ZILOG Z16C30 USC® Product Specifications Databook

ZILOG, Inc.
210 East Hacienda Ave.
Campbell, CA 95008-6600
(408) 370-8000
<http://www.zilog.com/>

PLX PCI 9080 Data Book

PLX Technology Inc.
390 Potrero Avenue
Sunnyvale, CA 4085
(408) 774-3735
<http://www.plxtech.com/>

EIA-422-A – Electrical Characteristics of Balanced Voltage Digital Interface Circuits
(EIA order number EIA-RS-422A)

EIA-485 – Standard for Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems
(EIA order number EIA-RS-485)

EIA Standards and Publications can be purchased from:

GLOBAL ENGINEERING DOCUMENTS
15 Inverness Way East
Englewood, CO 80112
Phone: (800) 854-7179
<http://global.ihs.com/>

IEEE P1386 - Standard Mechanic for a Common Mezzanine Card Family: CMC

IEEE P1386.1 - Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC

Sponsored by the Microprocessor & Microcomputer Standards Committee (MMSC) of the IEEE Computer Society

Copies of IEEE specifications available from:

Institute of Electrical and Electronics Engineers
Service Center
445 Hoes Lane
Piscataway, NJ 08855-1331 USA
<http://www.ieee.org/>

PCI Local Bus Specification Revision 2.1 June 1, 1995.

Copies of PCI specifications available from:

PCI Special Interest Group
NE 2575 Kathryn Street, #17
Hillsboro, OR 97124
<http://www.pcisig.com/>

PRELIMINARY

TABLE OF CONTENTS

| | |
|--|-----------|
| CHAPTER 1: INTRODUCTION..... | 1 |
| 1.0 GENERAL DESCRIPTION..... | 1 |
| 1.1 FUNCTIONAL DESCRIPTION..... | 1 |
| 1.1.1 PMC INTERFACE..... | 2 |
| 1.1.2 LOCAL CONTROL LOGIC..... | 2 |
| 1.1.3 TRANSMIT/RECEIVE FIFOS..... | 2 |
| 1.1.4 UNIVERSAL SERIAL CONTROLLERS..... | 2 |
| 1.1.5 RS422/RS485 TRANSCEIVERS..... | 2 |
| 1.1.6 CONNECTOR INTERFACE..... | 2 |
| CHAPTER 2: LOCAL SPACE REGISTERS..... | 3 |
| 2.0 REGISTER MAP..... | 3 |
| 2.1 GSC FIRMWARE REGISTERS..... | 4 |
| 2.1.1 FIRMWARE REVISION: LOCAL OFFSET 0x0000..... | 5 |
| 2.1.2 BOARD CONTROL: LOCAL OFFSET 0x0004..... | 5 |
| 2.1.3 BOARD STATUS: LOCAL OFFSET 0x0008..... | 6 |
| 2.1.4 CLOCK CONTROL: LOCAL OFFSET 0x000C..... | 6 |
| 2.1.5 CHANNEL TX ALMOST FLAGS: LOCAL OFFSET 0x0010 / 0x0020 / 0x0030 / 0x0040..... | 7 |
| 2.1.6 CHANNEL RX ALMOST FLAGS: LOCAL OFFSET 0x0014 / 0x0024 / 0x0034 / 0x0044..... | 7 |
| 2.1.7 CHANNEL FIFO: LOCAL OFFSET 0x0018 / 0x0028 / 0x0038 / 0x0048..... | 7 |
| 2.1.8 CHANNEL CONTROL/STATUS: LOCAL OFFSET 0x001C / 0x002C / 0x003C / 0x004C..... | 8 |
| 2.1.9 CHANNEL SYNC DETECT BYTE: LOCAL OFFSET 0x0050 / 0x0054 / 0x0058 / 0x005C..... | 9 |
| 2.1.10 INTERRUPT REGISTERS..... | 9 |
| 2.1.10.1 INTERRUPT CONTROL: LOCAL OFFSET 0x0060..... | 10 |
| 2.1.10.2 INTERRUPT STATUS/CLEAR: LOCAL OFFSET 0x0064..... | 10 |
| 2.1.10.3 INTERRUPT EDGE/LEVEL & INTERRUPT HI/LO: LOCAL OFFSET 0x0068 / 0x006C..... | 10 |
| 2.1.11 CHANNEL PIN SOURCE: LOCAL OFFSET 0x0080 / 0x0084 / 0x0088 / 0x8C..... | 11 |
| 2.1.12 PROGRAMMABLE CLOCK REGISTERS: LOCAL OFFSET 0x00A0 / 0x00A4 / 0x00A8 / 0x00AC..... | 12 |
| 2.1.13 FEATURES REGISTER: LOCAL OFFSET 0x00FC..... | 13 |
| 2.2 UNIVERSAL SERIAL CONTROLLER REGISTERS..... | 14 |
| 2.2.1 USC RESET..... | 14 |
| 2.2.2 8-BIT USC REGISTER ACCESS..... | 14 |
| 2.2.3 USC DATA TRANSFER..... | 14 |
| 2.2.4 USC REGISTER MEMORY MAP..... | 15 |
| CHAPTER 3: PCI/PMC INTERFACE..... | 16 |
| 3.0 PCI/PMC INTERFACE REGISTERS..... | 16 |
| 3.1 PCI REGISTERS..... | 16 |
| 3.1.1 PCI CONFIGURATION REGISTERS..... | 16 |
| 3.1.2 LOCAL CONFIGURATION REGISTERS..... | 17 |
| 3.1.3 RUNTIME REGISTERS..... | 17 |
| 3.1.3 DMA REGISTERS..... | 17 |
| 3.1.3.1 DMA CHANNEL MODE REGISTER: (PCI 0x80 / 0x94)..... | 17 |
| CHAPTER 4: PROGRAMMING..... | 18 |
| 4.0 INTRODUCTION..... | 18 |
| 4.1 RESETS..... | 18 |
| 4.2 FIFO ALMOST FLAGS..... | 18 |
| 4.3 DMA..... | 19 |
| 4.4 INTERRUPTS..... | 19 |
| 4.5 CLOCK SETUP..... | 19 |
| 4.6 PROGRAMMABLE OSCILLATOR / PROGRAMMABLE CLOCKS..... | 21 |

| | | |
|--|--|-----------|
| 4.7 | UPPER/LOWER CONNECTOR NAMING CONVENTION..... | 21 |
| 4.8 | CTS/DCD | 22 |
| CHAPTER 5: HARDWARE CONFIGURATION..... | | 24 |
| 5.1 | BOARD LAYOUT | 24 |
| 5.2 | BOARD ID JUMPER J3 | 24 |
| 5.3 | INTERFACE CONNECTORS..... | 25 |
| 5.4 | TERMINATION RESISTORS..... | 26 |
| CHAPTER 6: ORDERING OPTIONS..... | | 27 |
| 6.0 | ORDERING INFORMATION | 27 |
| 6.0.1 | FIFO SIZE | 27 |
| 6.0.2 | INTERFACE CABLE..... | 27 |
| 6.0.3 | DEVICE DRIVERS..... | 27 |
| 6.1 | CUSTOM APPLICATIONS..... | 27 |
| APPENDIX A: COMMON PROGRAMMABLE CLOCK REGISTER VALUES | | 28 |
| APPENDIX B: CALCULATING PROGRAMMABLE CLOCK VALUES..... | | 29 |

CHAPTER 1: INTRODUCTION

1.0 General Description

The PMC-SIO4AR board is a four channel serial interface card which provides high speed, full-duplex, multi-protocol serial capability for PCI Mezzanine Card (PMC) applications. The PMC-SIO4AR combines two multi-protocol Dual Universal Serial Controllers (USC®) and 8 external FIFOs to provide four fully independent asynchronous or synchronous RS422/RS485 serial channels. In addition to the standard front edge I/O connector, the PMC-SIO4AR also provides a user interface via the PMC Rear IO connector. These features, along with a high performance PCI interface engine, give the PMC-SIO4AR unsurpassed performance in a serial interface card.

The PMC-SIO4AR incorporates the following features:

- Four Independent Multi-Protocol Serial Channels
- Synchronous Serial Data Rates up to 10 Mbits/sec
- Asynchronous Serial Data Rates up to 1 Mbit/sec
- PMC Rear IO User Interface.
- SCSI II type 68 pin front edge I/O Connector with optional cable adapter to four DB25 connectors.
- Independent Transmit and Receive FIFO Buffers for each Serial Channel – Up to 32k Deep Each
- Fast RS485/RS422 Differential Cable Transceivers to Provide Increased Noise Immunity
- Two Industry Standard Zilog Z16C30 Multi-Protocol Universal Serial Controllers (USC®)
- Serial Mode Protocols include Asynchronous, Bisync, SDLC, HDLC, Ethernet, and Nine-Bit Parity and CRC detection capability
- Two Serial Clocks, Two Serial Data signals, Data Carrier Detect, and Clear-To-Send per Channel
- Dual PCI DMA Engine to speed transfers and minimize host I/O overhead
- On-Board Programmable Oscillator provides increased flexibility for Baud Rate Clock generation
- A variety of device drivers are available, including VxWorks, WinNT, Win2k, Linux, and Labview

1.1 Functional Description

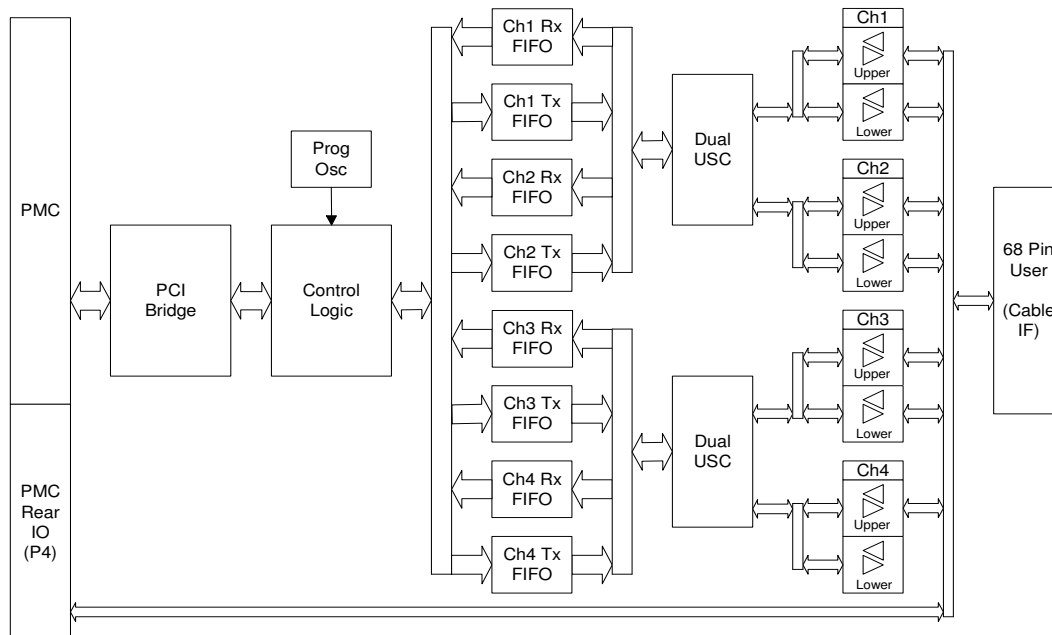


Figure 1-1 Block Diagram of PMC-SIO4AR

1.1.1 PMC Interface

The control interface to the PMC-SIO4AR is through the PCI Mezzanine Card (PMC) interface. An industry standard PCI9080 bridge chip from PLX Technology is used to implement PCI Specification 2.1. The PCI9080 provides the 32bit, 33MHz (132MBit/sec) interface between the PMC bus and the Local 32 bit bus. In addition, the PMC P4 connector provides the Rear IO Interface for the board.

1.1.2 Local Control Logic

The control functions and glue logic for the board are implemented in an on-board FPGA. This custom logic defines local space registers to provide software control over the board functions. These functions include setup, control, and transfer of data to and from the serial controller chips.

1.1.3 Transmit/Receive FIFOs

Eight independent Transmit and Receive FIFOs provide up to 32kbytes of data buffering per channel for the serial data. Each channel has a unique transmit and receive FIFO to allow the channels to operate independently. The FIFOs allow data transfer to continue to/from the IO interface independent of PMC interface transfers and software overhead. The required FIFO size may depend on several factors including data transfer size, required throughput rate, and the software overhead (which will also vary based on OS). Deep FIFOs ensure no data is lost for critical systems.

1.1.4 Universal Serial Controllers

Two Zilog Z16C30 Universal Serial Controllers provide the four serial data channels. The Z16C30 USCs serve as serial/parallel converters which can be software configured to provide a variety of serial protocols. The USCs are highly configurable to allow for a wide range of serial solutions.

1.1.5 RS422/RS485 Transceivers

Data is transferred over the user interface using high-speed, differential RS485/RS422 transceivers. Industry standard differential RS485 signaling allows for longer, faster, and more reliable data connections. Socketed termination resistors allow the board to comply with both RS485 and RS422 terminations, or they may be removed for a multi-drop configuration. All transceivers are bi-directional, so that any pin may be configured to receive or transmit. This allows the user more flexibility for wiring cable connections. It also enables two SIO4 cards to be connected together via a standard, straight-thru cable.

1.1.6 Connector Interface

The PMC-SIO4AR provides two user IO interfaces – a front-side card edge connector and a rear IO PMC connection. The front-side card connector is a high-density 68 pin SCSI II type connector. All four serial channels interface via this cable connector, and are grouped to simplify separating the cable into four distinct serial connectors. The Rear IO connector provides the same RS485/RS422 differential interface to the PMC P4 connector.

CHAPTER 2: LOCAL SPACE REGISTERS

2.0 Register Map

The PMC-SIO4AR is accessed through three sets of registers – PCI Registers, USC Registers, and GSC Firmware Registers. The GSC Firmware Registers and USC Registers are referred to as Local Space Registers and are described below. The PCI registers are discussed in Chapter 3.

The Local Space Registers are divided into two distinct functional register blocks – the GSC Firmware Registers and the USC Registers. The GSC Firmware Registers perform the custom board control functions, while the USC Registers map the Zilog Z16C30 registers into local address space. The register block for each USC channel is accessed at a unique address range. The table below shows the address mapping for the local space registers.

| Local Address Range | Base Address Offset | Register Block Description |
|---------------------|---------------------|----------------------------|
| 0x0000 – 0x00FF | 0x0000 | GSC Firmware Registers |
| 0x0100 – 0x013F | 0x0100 | Channel 1 USC Registers |
| 0x0140 – 0x01FF | | Reserved |
| 0x0200 – 0x023F | 0x0200 | Channel 2 USC Registers |
| 0x0240 – 0x02FF | | Reserved |
| 0x0300 – 0x033F | 0x0300 | Channel 3 USC Registers |
| 0x0340 – 0x03FF | | Reserved |
| 0x0400 – 0x043F | 0x0400 | Channel 4 USC Registers |

The GSC Firmware Registers are detailed in Section 2.1. The USC Registers are briefly touched on in Section 2.2 of this manual, but are described in much greater detail in the [Zilog Z16C30 Users Manuals](#).

2.1 GSC Firmware Registers

The GSC Firmware Registers provide the primary control/status for the PMC-SIO4AR board. These registers define general setup and control functions for the board, including interrupt handling, serial transceiver setup, and programmable clock functions. The on-board FIFOs which pass data to/from the serial channels are also mapped in this region. The following table shows the GSC Firmware Registers.

| Offset Address | Size | Access* | Register Name | Default Value (Hex) |
|----------------|------|---------|---------------------------|---------------------|
| 0x0000 | D32 | RW | Firmware Revision | 010201XX |
| 0x0004 | D32 | RW | Board Control | 00000000 |
| 0x0008 | D32 | RO | Board Status | 0000000X |
| 0x000C | D32 | RW | Clock Control | 00000000 |
| 0x0010 | D32 | RW | Ch 1 Tx Almost Full/Empty | 00070007 |
| 0x0014 | D32 | RW | Ch 1 Rx Almost Full/Empty | 00070007 |
| 0x0018 | D32 | RW | Ch 1 1 Data FIFO | 000000XX |
| 0x001C | D32 | RW | Ch 1 Control/Status | 0000CC00 |
| 0x0020 | D32 | RW | Ch 2 Tx Almost Full/Empty | 00070007 |
| 0x0024 | D32 | RW | Ch 2 Rx Almost Full/Empty | 00070007 |
| 0x0028 | D32 | RW | Ch 2 FIFO | 000000XX |
| 0x002C | D32 | RW | Ch 2 Control/Status | 0000CC00 |
| 0x0030 | D32 | RW | Ch 3 Tx Almost Full/Empty | 00070007 |
| 0x0034 | D32 | RW | Ch 3 Rx Almost Full/Empty | 00070007 |
| 0x0038 | D32 | RW | Ch 3 Data FIFO | 000000XX |
| 0x003C | D32 | RW | Ch 3 Control/Status | 0000CC00 |
| 0x0040 | D32 | RW | Ch 4 Tx Almost Full/Empty | 00070007 |
| 0x0044 | D32 | RW | Ch 4 Rx Almost Full/Empty | 00070007 |
| 0x0048 | D32 | RW | Ch 4 Data FIFO | 000000XX |
| 0x004C | D32 | RW | Ch 4 Control/Status | 0000CC00 |
| 0x0050 | D32 | RW | Ch 1 Sync Byte | 00000000 |
| 0x0054 | D32 | RW | Ch 2 Sync Byte | 00000000 |
| 0x0058 | D32 | RW | Ch 3 Sync Byte | 00000000 |
| 0x005C | D32 | RW | Ch 4 Sync Byte | 00000000 |
| 0x0060 | D32 | RW | Interrupt Control | 00000000 |
| 0x0064 | D32 | RW | Interrupt Status | 00000000 |
| 0x0068 | D32 | RO | Interrupt Edge/Level | FFFF7777 |
| 0x006C | D32 | RW | Interrupt High/Low | FFFFFFFF |
| 0x0070-0x007C | --- | -- | RESERVED | ----- |
| 0x0080 | D32 | RW | Channel 1 Pin Source | 00000020 |
| 0x0084 | D32 | RW | Channel 2 Pin Source | 00000020 |
| 0x0088 | D32 | RW | Channel 3 Pin Source | 00000020 |
| 0x008C | D32 | RW | Channel 4 Pin Source | 00000020 |
| 0x0090-0x009C | --- | -- | RESERVED | ----- |
| 0x00A0 | D32 | RW | Ch1 Programmable Clock | 00129128 |
| 0x00A0 | D32 | RW | Ch2 Programmable Clock | 00000000 |
| 0x00A0 | D32 | RW | Ch3 Programmable Clock | 00000000 |
| 0x00A0 | D32 | RW | Ch4 Programmable Clock | 00000000 |
| 0x00B0-0x00F8 | --- | -- | RESERVED | ----- |
| 0x00FC | D32 | RO | Features Register | 000000XX |

* RO = read only, RW = read/write capability

2.1.1 Firmware Revision: Local Offset 0x0000

The Firmware ID register provides version information about the firmware on the board. This is useful for technical support to identify the firmware version.

| | | |
|---------------|-------------------------|--|
| D6:0 | Firmware Revision | Identifies version of specific firmware |
| D7 | Internal/External FIFOs | 0 = External FIFOs / 1 = Internal FIFOs |
| D15:0 | Device ID | 01 = SIO4A Standard |
| D23:16 | SIO4 Board ID | 02 = PMC-SIO4AR |
| D31:24 | Misc | 01 = Features Register at Local Offset 0x00FC. |

2.1.2 Board Control: Local Offset 0x0004

The Board Control Register defines the general control functions for the board. The main function in this register defines the Demand mode DMA channel requests. For Demand mode DMA, there are only two physical DMA channels which must be shared between the eight serial channels (Rx and TX for each of four channels). The Demand Mode DMA Channel Request allows the software to multiplex the DMA channels. This is typically handled by the driver – the end user should have no need to change this register.

D2: 0 Demand Mode DMA Channel 0 Request

| D2 | D1 | D0 | Demand Mode DMA 0 Channel |
|----|----|----|---------------------------|
| 0 | 1 | 0 | Channel 1 Rx |
| 1 | 0 | 0 | Channel 1 Tx |
| 0 | 0 | 0 | Channel 2 Rx |
| 1 | 1 | 0 | Channel 2 Tx |
| 0 | 1 | 1 | Channel 3 Rx |
| 1 | 0 | 1 | Channel 3 Tx |
| 0 | 0 | 1 | Channel 4 Rx |
| 1 | 1 | 1 | Channel 4 Tx |

D3 Reserved

D4:6 Demand Mode DMA Channel 1 Request

| D6 | D5 | D4 | Demand Mode DMA 1 Channel |
|----|----|----|---------------------------|
| 0 | 1 | 0 | Channel 1 Rx |
| 1 | 0 | 0 | Channel 1 Tx |
| 0 | 0 | 0 | Channel 2 Rx |
| 1 | 1 | 0 | Channel 2 Tx |
| 0 | 1 | 1 | Channel 3 Rx |
| 1 | 0 | 1 | Channel 3 Tx |
| 0 | 0 | 1 | Channel 4 Rx |
| 1 | 1 | 1 | Channel 4 Tx |

D7 Reserved

D8 Rx FIFO Enable (Test Mode)
 1 = Rx FIFO enabled
 0 = Rx FIFO disabled

D31:9 Reserved

2.1.3 Board Status: Local Offset 0x0008

The Board Status Register gives general overall status for a board. The Board Jumpers are physical jumpers which can be used to distinguish between boards if multiple SIO4 boards are present in a system.

| | |
|---------------|--|
| D0 | Board Jumper 0 (0=jumper J3:1-2 installed) |
| D1 | Board Jumper 1 (0=jumper J3:3-4 installed) |
| D31:D2 | Reserved |

2.1.4 Clock Control: Local Offset 0x000C

The Clock Control Register enables the transmit and receive buffers for the cable clock signals to and from the user IO cable. See Section 4.7 **Upper/Lower Cable Naming Convention** for an explanation of “Upper/Lower”.

| | |
|------------|---|
| D0 | Ch 1 Upper Transmit Clock Enable 1 = Channel Upper TxClk driven to I/O connector. 0 = Channel Upper TxClk disabled (tristate). |
| D1 | Ch 1 Lower Transmit Clock Enable 1 = Channel Lower TxClk driven to I/O connector. 0 = Channel Lower TxClk disabled (tristate). |
| D2 | Ch 1 Upper Receive Clock Enable 1 = Channel Upper RxClk received in from I/O connector. 0 = Channel Upper RxClk disabled (tristate). |
| D3 | Ch 1 Lower Receive Clock Enable 1 = Channel Lower RxClk received in from I/O connector. Note: Lower RxClk will NOT be enabled if D2 (Upper RxClk) is enabled. 0 = Channel Lower RxClk disabled (tristate). |
| D4 | Ch 2 Upper Transmit Clock Enable 1 = Channel Upper TxClk driven to I/O connector. 0 = Channel Upper TxClk disabled (tristate). |
| D5 | Ch 2 Lower Transmit Clock Enable 1 = Channel Lower TxClk driven to I/O connector. 0 = Channel Lower TxClk disabled (tristate). |
| D6 | Ch 2 Upper Receive Clock Enable 1 = Channel Upper RxClk received in from I/O connector. 0 = Channel Upper RxClk disabled (tristate). |
| D7 | Ch 2 Lower Receive Clock Enable 1 = Channel Lower RxClk received in from I/O connector. Note: Lower RxClk will NOT be enabled if D6 (Upper RxClk) is enabled. 0 = Channel Lower RxClk disabled (tristate). |
| D8 | Ch 3 Upper Transmit Clock Enable 1 = Channel Upper TxClk driven to I/O connector. 0 = Channel Upper TxClk disabled (tristate). |
| D9 | Ch 3 Lower Transmit Clock Enable 1 = Channel Lower TxClk driven to I/O connector. 0 = Channel Lower TxClk disabled (tristate). |
| D10 | Ch 3 Upper Receive Clock Enable 1 = Channel Upper RxClk received in from I/O connector. 0 = Channel Upper RxClk disabled (tristate). |
| D11 | Ch 3 Lower Receive Clock Enable 1 = Channel Lower RxClk received in from I/O connector. Note: Lower RxClk will NOT be enabled if D10 (Upper RxClk) is enabled. 0 = Channel Lower RxClk disabled (tristate). |

- D12** Ch 4 Upper Transmit Clock Enable
 - 1 = Channel Upper TxClk driven to I/O connector.
 - 0 = Channel Upper TxClk disabled (tristate).
- D13** Ch 4 Lower Transmit Clock Enable
 - 1 = Channel Lower TxClk driven to I/O connector.
 - 0 = Channel Lower TxClk disabled (tristate).
- D14** Ch 4 Upper Receive Clock Enable
 - 1 = Channel Upper RxClk received in from I/O connector.
 - 0 = Channel Upper RxClk disabled (tristate).
- D15** Ch 4 Lower Receive Clock Enable
 - 1 = Channel Lower RxClk received in from I/O connector.
 - Note:** Lower RxClk will NOT be enabled if D14 (Upper RxClk) is enabled.
 - 0 = Channel Lower RxClk disabled (tristate).

2.1.5 Channel TX Almost Flags: Local Offset 0x0010 / 0x0020 / 0x0030 / 0x0040

The Tx Almost Flag Registers are used to program the Almost Full and Almost Empty Flags in the transmit FIFOs. The FIFO almost flags may be used to determine a fill level for a specific transfer size, or used for demand mode DMA. Setting this register does not automatically program the almost flags – the registers will be programmed during a FIFO Reset command. See 4.2 **FIFO Almost Flags** for further information concerning Tx FIFO Almost Flag programming.

- D15:0** TX Almost Empty Flag Value
 - Almost Empty Flag will be asserted when the FIFO contains “Almost Empty Value” words or fewer.
- D31:16** TX Almost Full Flag Value
 - Almost Full Flag will be asserted when the FIFO has space for “Almost Full Value” words or fewer (i.e. FIFO contains (FIFO Size – Almost Full Value) words or more.)

2.1.6 Channel Rx Almost Flags: Local Offset 0x0014 / 0x0024 / 0x0034 / 0x0044

The Rx Almost Flag Registers are used to program the Almost Full and Almost Empty Flags in the receive FIFOs. The FIFO almost flags may be used to determine a fill level for a specific transfer size, or used for demand mode DMA. Setting this register does not automatically program the almost flags – the registers will be programmed during a FIFO Reset command. See 4.2 **FIFO Almost Flags** for further information concerning Rx FIFO Almost Flag programming.

- D15:0** Rx Almost Empty Flag Value
 - Almost Empty Flag will be asserted when the FIFO contains “Almost Empty Value” words or fewer.
- D31:16** Rx Almost Full Flag Value
 - Almost Full Flag will be asserted when the FIFO has space for “Almost Full Value” words or fewer (i.e. FIFO contains (FIFO Size – Almost Full Value) words or more.)

2.1.7 Channel FIFO: Local Offset 0x0018 / 0x0028 / 0x0038 / 0x0048

The Channel FIFO Register passes serial data to/from the serial controller chips. The same register is used to access both the Transmit FIFO (writes) and Receive FIFO (reads).

- D7:0** Channel FIFO Data
- D31:8** Reserved

2.1.8 Channel Control/Status: Local Offset 0x001C / 0x002C / 0x003C / 0x004C

The Channel Control/Status Register provides the reset functions and data transceiver enable controls, and the FIFO Flag status for each channel.

D7:0 Channel Control Bits

D0 Reset Channel TX FIFO (Pulsed)

1 = Reset Channel TX FIFOs.

Notes:

- This value will automatically clear to '0'.
- If the Channel TX Almost Flag register is not zero, the Tx Almost Flag Values will automatically be programmed following the FIFO Reset.
- Software must wait approximately 1ms following a FIFO Reset before accessing the Local Board Registers again.

D1 Reset Channel Rx FIFO (Pulsed)

1 = Reset Channel Rx FIFOs.

Notes:

- This value will automatically clear to '0'.
- If the Channel Rx Almost Flag register is not zero, the Rx Almost Flag Values will automatically be programmed following the FIFO Reset.

D5:2 Enables for data signals through the transmit and receive buffers to and from the user IO connector. See **4.8 Upper/Lower Cable Naming Convention** for an explanation of "Upper/Lower".

D2 Channel Upper Transmit Data Enable

1 = Channel Upper TxD driven to IO connector.

0 = Channel Upper TxD disabled (tristate).

D3 Channel Lower Transmit Data Enable

1 = Channel Lower TxD driven to IO connector.

0 = Channel Lower TxD disabled (tristate).

D4 Channel Upper Receive Data Enable

1 = Channel Upper RxD received in from IO connector.

0 = Channel Upper RxD disabled (tristate).

D5 Channel Lower Rx Data Enable

1 = Channel Lower RxD received in from IO connector.

Note: Lower RxD will NOT be enabled if D4 (Upper RxD) also enabled.

0 = Channel Lower RxD disabled (tristate).

D6 Reserved

D7 Reset USC (Pulsed)

'1' = Reset USC chip

Notes:

- This value will automatically clear to '0'.
- Following a USC Reset, the next access to the USC must be a write of 0x00 to Local Offset 0x100 (Ch1/2) or Local Offset 0x300 (Ch3/4).
- Since two channels share each USC (Ch1 & Ch2, Ch3 & Ch4), resetting a USC will affect both channels.

D15:8 Channel Status Bits

| | | |
|------------|--------------------------------------|-----------------------------------|
| D8 | Channel Tx FIFO Empty Flag Lo | (Active Low -- 0=Tx Empty) |
| D9 | Channel Tx FIFO Almost Empty Flag Lo | (Active Low -- 0=Tx Almost Empty) |
| D10 | Channel Tx FIFO Almost Full Flag Lo | (Active Low -- 0=Tx Almost Full) |
| D11 | Channel Tx FIFO Full Flag Lo | (Active Low -- 0=Tx Full) |
| D12 | Channel Rx FIFO Empty Flag Lo | (Active Low -- 0=Rx Empty) |
| D13 | Channel Rx FIFO Almost Empty Flag Lo | (Active Low -- 0=Rx Almost Empty) |
| D14 | Channel Rx FIFO Almost Full Flag Lo | (Active Low -- 0=Rx Almost Full) |
| D15 | Channel Rx FIFO Full Flag Lo | (Active Low -- 0=Rx Full) |

2.1.9 Channel Sync Detect Byte: Local Offset 0x0050 / 0x0054 / 0x0058 / 0x005C

The Sync Detect Byte allows an interrupt to be generated when the received data matches the Sync Detect Byte.

| | | |
|--------------|--------------------------|--|
| D7:0 | Channel Sync Detect Byte | If the data being loaded into the Receive FIFO matches this data byte, an interrupt request (Channel Sync Detect IRQ) will be generated. The interrupt source must be enabled in the Interrupt Control Register in order for an interrupt to be generated. |
| D31:8 | Reserved | |

2.1.10 Interrupt Registers

There are 32 on-board interrupt sources (in addition to USC interrupts and PLX interrupts), each of which may be individually enabled. Four interrupt registers control the on-board interrupts – Interrupt Control, Interrupt Status, Interrupt Edge/Level, and Interrupt Hi/Lo. The 32 Interrupt sources are:

| IRQ # | Source | Default Level | Alternate Level |
|-------|--------------------------------|---------------|-----------------|
| IRQ0 | Channel 1 Sync Detected | Rising Edge | NONE |
| IRQ1 | Channel 1 Tx FIFO Almost Empty | Rising Edge | Falling Edge |
| IRQ2 | Channel 1 Rx FIFO Almost Full | Rising Edge | Falling Edge |
| IRQ3 | Channel 1 USC Interrupt | Level Hi | NONE |
| IRQ4 | Channel 2 Sync Detected | Rising Edge | NONE |
| IRQ5 | Channel 2 Tx FIFO Almost Empty | Rising Edge | Falling Edge |
| IRQ6 | Channel 2 Rx FIFO Almost Full | Rising Edge | Falling Edge |
| IRQ7 | Channel 2 USC Interrupt | Level Hi | NONE |
| IRQ8 | Channel 3 Sync Detected | Rising Edge | NONE |
| IRQ9 | Channel 3 Tx FIFO Almost Empty | Rising Edge | Falling Edge |
| IRQ10 | Channel 3 Rx FIFO Almost Full | Rising Edge | Falling Edge |
| IRQ11 | Channel 3 USC Interrupt | Level Hi | NONE |
| IRQ12 | Channel 4 Sync Detected | Rising Edge | NONE |
| IRQ13 | Channel 4 Tx FIFO Almost Empty | Rising Edge | Falling Edge |
| IRQ14 | Channel 4 Rx FIFO Almost Full | Rising Edge | Falling Edge |
| IRQ15 | Channel 4 USC Interrupt | Level Hi | NONE |
| IRQ16 | Channel 1 Tx FIFO Empty | Rising Edge | Falling Edge |
| IRQ17 | Channel 1 Tx FIFO Full | Rising Edge | Falling Edge |
| IRQ18 | Channel 1 Rx FIFO Empty | Rising Edge | Falling Edge |
| IRQ19 | Channel 1 Rx FIFO Full | Rising Edge | Falling Edge |
| IRQ20 | Channel 2 Tx FIFO Empty | Rising Edge | Falling Edge |

| | | | |
|-------|-------------------------|-------------|--------------|
| IRQ21 | Channel 2 Tx FIFO Full | Rising Edge | Falling Edge |
| IRQ22 | Channel 2 Rx FIFO Empty | Rising Edge | Falling Edge |
| IRQ23 | Channel 2 Rx FIFO Full | Rising Edge | Falling Edge |
| IRQ24 | Channel 3 Tx FIFO Empty | Rising Edge | Falling Edge |
| IRQ25 | Channel 3 Tx FIFO Full | Rising Edge | Falling Edge |
| IRQ26 | Channel 3 Rx FIFO Empty | Rising Edge | Falling Edge |
| IRQ27 | Channel 3 Rx FIFO Full | Rising Edge | Falling Edge |
| IRQ28 | Channel 4 Tx FIFO Empty | Rising Edge | Falling Edge |
| IRQ29 | Channel 4 Tx FIFO Full | Rising Edge | Falling Edge |
| IRQ30 | Channel 4 Rx FIFO Empty | Rising Edge | Falling Edge |
| IRQ31 | Channel 4 Rx FIFO Full | Rising Edge | Falling Edge |

For all interrupt registers, the IRQ source (IRQ31:IRQ0) will correspond to the respective data bit (D31:D0) of each register. (D0 = IRQ0, D1 = IRQ1, etc.)

All FIFO interrupts are edge triggered active high. This means that an interrupt will be asserted (assuming it is enabled) when a FIFO Flag transitions from FALSE to TRUE (rising edge triggered) or TRUE to FALSE (falling edge). For example: If Tx FIFO Empty Interrupt is set for Rising Edge Triggered, the interrupt will occur when the FIFO transitions from NOT EMPTY to EMPTY. Likewise, if Tx FIFO Empty Interrupt is set as Falling Edge Triggered, the interrupt will occur when the FIFO transitions from EMPTY to NOT EMPTY.

All Interrupt Sources share a single interrupt request back to the PCI9080 PLX chip. Likewise, all USC interrupt sources share a single interrupt request back to the interrupt controller and must be further qualified in the USC chip. See Section **4.4 Interrupts** for further interrupt programming information.

2.1.10.1 Interrupt Control: Local Offset 0x0060

The Interrupt Control register individually enables each interrupt source. A '1' enables each interrupt source; a '0' disables. An interrupt source must be enabled for an interrupt to be generated.

2.1.10.2 Interrupt Status/Clear: Local Offset 0x0064

The Interrupt Status Register shows the status of each respective interrupt source. If an interrupt source is enabled in the Interrupt Control Register, a '1' in the Interrupt Status Register indicates the respective interrupt has occurred. The interrupt source will remain latched until the interrupt is cleared, either by writing to the Interrupt Status/Clear Register with a '1' in the respective interrupt bit position, or the interrupt is disabled in the Interrupt Control register. If an interrupt source is not asserted or the interrupt is not enabled, writing a '1' to that bit in the Interrupt Status/Clear Register will have no effect on the interrupt.

If the interrupt source is a level triggered interrupt (USC interrupt), the interrupt status may still be '1' even if the interrupt is disabled. This indicates the interrupt condition is true, regardless of whether the interrupt is enabled. Likewise, if a level interrupt is enabled and the interrupt source is true, the interrupt status will be reasserted immediately after clearing the interrupt, and an additional interrupt will be requested.

2.1.10.3 Interrupt Edge/Level & Interrupt Hi/Lo: Local Offset 0x0068 / 0x006C

The Interrupt Edge/Level and Interrupt Hi/Lo Registers define each interrupt source as level hi, level lo, rising edge, or falling edge. All PMC-SIO4AR interrupts are edge triggered except the USC interrupts which are level triggered. Since the interrupt behavior is fixed, the Interrupt Edge/Level register cannot be changed by the user. (Read Only)

The FIFO Flags may be defined as rising edge or falling edge via the Interrupt Hi/Lo Register. For example, a rising edge of the TX Empty source will generate an interrupt when the TX FIFO becomes empty. Defining the source as falling edge will trigger an interrupt when the TX FIFO becomes "NOT Empty".

2.1.11 Channel Pin Source: Local Offset 0x0080 / 0x0084 / 0x0088 / 0x8C

The Channel Pin Source Register configures the Clock source and CTS direction for each channel. Section 4.5 describes the clock setup and Section 4.8 shows the CTS setup for the board.

D2:0 USC TxC Source

Defines the Clock Source for the USC TxC pin. The clock source must agree with the USC Clock setup (USC IO Control Reg D2:0) to ensure the signal is not being driven by both the USC and the FPGA.

| D2 | D1 | D0 | USC TxC Source | USC Setup |
|----|----|----|--------------------|---------------------------------|
| 0 | 0 | X | Programmable Clock | Input to USC -- IOCR D2:0 = 000 |
| 0 | 1 | 0 | '0' | Input to USC -- IOCR D2:0 = 000 |
| 0 | 1 | 1 | '1' | Input to USC -- IOCR D2:0 = 000 |
| 1 | 0 | 0 | Cable RxClk | Input to USC -- IOCR D2:0 = 000 |
| 1 | 0 | 1 | RESERVED | ----- |
| 1 | 1 | 0 | RESERVED | ----- |
| 1 | 1 | 1 | Driven from USC | Output from USC |

D5:3 USC RxC Source

Defines the Clock Source for the USC RxC pin. The clock source must agree with the USC Clock setup (USC I/O Control Reg D5:3) to ensure the signal is not being driven by both the USC and the FPGA.

| D5 | D4 | D3 | USC RxC Source | USC Setup |
|----|----|----|--------------------|---------------------------------|
| 0 | 0 | X | Programmable Clock | Input to USC -- IOCR D5:3 = 000 |
| 0 | 1 | 0 | '0' | Input to USC -- IOCR D5:3 = 000 |
| 0 | 1 | 1 | '1' | Input to USC -- IOCR D5:3 = 000 |
| 1 | 0 | 0 | Cable RxClk | Input to USC -- IOCR D5:3 = 000 |
| 1 | 0 | 1 | RESERVED | ----- |
| 1 | 1 | 0 | RESERVED | ----- |
| 1 | 1 | 1 | Driven from USC | Output from USC |

D8:6 TxClk Source

Defines the Clock Source for the TxClk signal to the IO connector.

| D8 | D7 | D6 | TxClock Source |
|----|----|----|--------------------|
| 0 | 0 | X | Programmable Clock |
| 0 | 1 | 0 | '0' |
| 0 | 1 | 1 | '1' |
| 1 | 0 | 0 | USC TxClk |
| 1 | 0 | 1 | USC RxClk |
| 1 | 1 | 0 | Cable RxClk |
| 1 | 1 | 1 | RESERVED |

D10:9 CTS Buffer Direction Setup

Defines the CTS direction for the USC CTS pin. The CTS direction must also with the USC CTS setup (USC I/O Control Reg D15:14) to ensure the signal is not being driven by both the USC and the FPGA.

| D10 | D9 | CTS Buffer Direction | USC Setup |
|-----|----|-------------------------|----------------------------------|
| 0 | 0 | Buffer Disabled | Don't Care -- IOCR D15:14 = XX |
| 0 | 1 | Input from IO Connector | Input to USC -- IOCR D15:14 = 0X |
| 1 | 0 | Input from IO Connector | Input to USC -- IOCR D15:14 = 0X |
| 1 | 1 | Output to IO Connector | Output from USC |

D31:11 Reserved

2.1.12 Programmable Clock Registers: Local Offset 0x00A0 / 0x00A4 / 0x00A8 / 0x00AC

The Programmable Clock Registers allow the user to program the on-board programmable oscillator and configure the channel clock post-dividers. Since there is only a single programmable oscillator, the output of this oscillator will be used as the input to all four channel clock post-dividers. Figure 2-1 shows the Programmable clock registers.

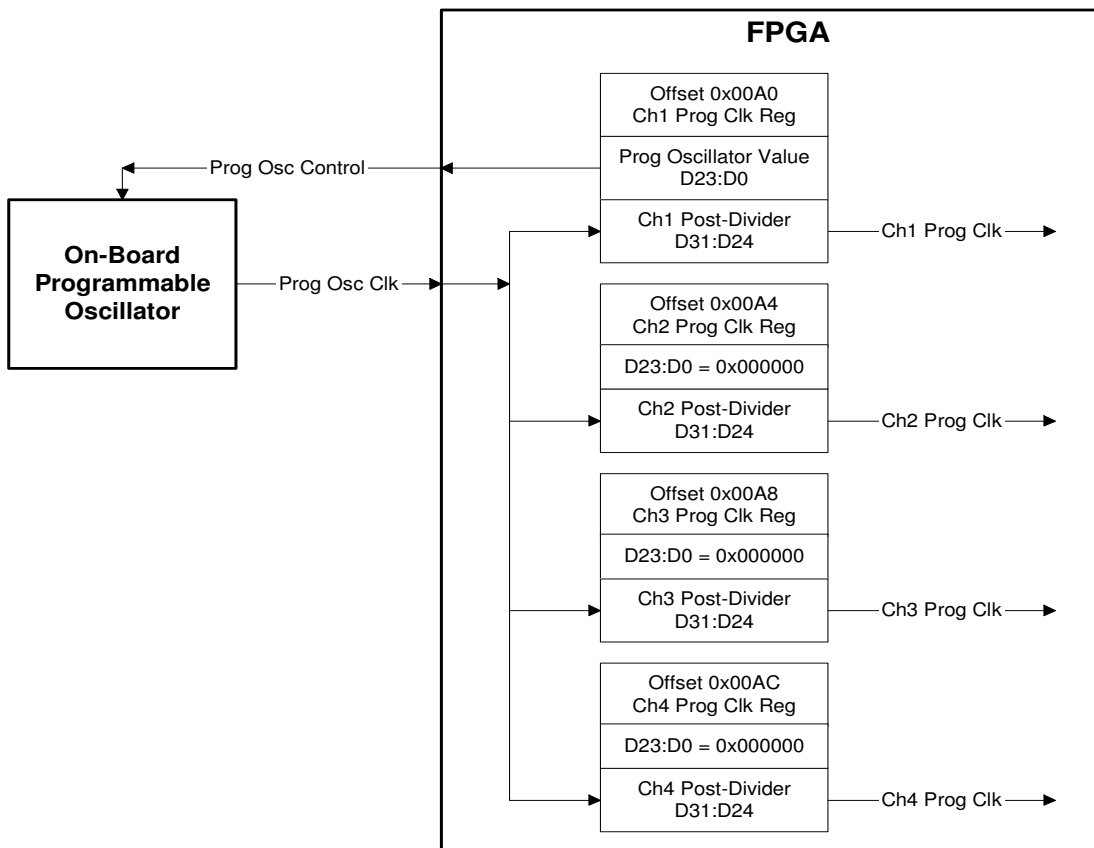


Figure 2-1: Channel Programmable Clock Registers

Loading the Programmable Clock Register for Channel 1 (Offset 0x00A0) will automatically program the on-board oscillator. See Section **4.6 Programmable Oscillator** for more information of programming the on-board oscillator.

Since the On-Board Programmable Oscillator has a minimum clock frequency of 400 kHz, bits D31:D24 of the Programmable Clock Register may be used to further “Post Divide” the Programmable Oscillator input. This allows for programmable clock frequencies below 2 kHz to be obtained.

For Channel 1 (Offset 0x00A0):

- D22:0** Programmable Oscillator Program Value
- D23** Programming In Progress Status
After writing the Ch1 Programmable Clock Register, this bit will remain ‘1’ while the on-board oscillator is being programmed.
- D31:24** Ch1 Programmable Clock Post-Divide Value (0-255)
NOTE: If the post divide value is set to 0x00 or 0x01, no post divide will be performed.
(Programmable Clock Frequency = Programmable Oscillator Frequency)

For Channels 2-4 (Offsets 0x00A4 / 0x00A8 / 0x00AC):

- D23:0** 0x000000
- D31:24** Programmable Clock Post-Divide Value (0-255)
NOTE: If the post divide value is set to 0x00 or 0x01, no post divide will be performed.
(Programmable Clock Frequency = Programmable Oscillator Frequency)

2.1.13 Features Register: Local Offset 0x00FC

The Features Register allows software to account for added features in the firmware versions. Bits will be assigned as new features are added.

- D3:0** Programmable Clock Configuration – 01 = ICD2053B
- D4** FIFO Counters/Size Present
- D31:5** RESERVED

2.2 Universal Serial Controller Registers

The internal registers of the Zilog Z16C30 Universal Serial Controller (USC) are memory mapped into Local Address space. It is beyond the scope of this manual to provide comprehensive USC programming information. For detailed programming information, please refer to the [Zilog High Speed Communication Controller Product Specifications Databook](#) for the Z16C30 and the [Zilog Z16C30USC User's Manual](#). These manuals may be obtained directly from Zilog (www.zilog.com), or copies of these manuals may be downloaded from the General Standards website (www.generalstandards.com).

Some specific setup information may be needed for a driver to interface to the USC. Typically, the driver will handle the hardware specific characteristics and the end user will only need to be concerned with the driver interface - the following hardware setup information may be safely ignored. If you aren't sure if you need this information, you probably don't.

2.2.1 USC Reset

The four serial channels are implemented in two Z16C30 Universal Serial Controllers – Channels 1 and 2 share one USC, and Channels 3 and 4 share the other. This implementation is important to realize since resetting a Z16C30 chip will have an effect on two serial channels. Since the USC chips are typically reset upon initialization, this means a “Reset USC” for Channel 1 will also “Reset USC” for Channel 2. In addition to making the second reset redundant and unnecessary, a Reset USC on one channel may inadvertently adversely affect normal operation on the second channel. Therefore, care must be exercised when resetting a USC (USC Reset bit in the Board Control Register), especially in multithreaded environments.

Since the USC Reset physically resets the USC, the first access to the USC following the reset must reinitialize the BCR in the USC. To complete the Reset process, the user should write data 0x00 to USC base address offset 0x100 or 0x300 to correctly initialize the BCR. Following this initial byte write, the USC may be accessed normally.

Due to the ability for a USC Reset to affect two channels, it is recommended that a single USC Channel be Reset via the RTRReset bit of the USC Channel Command/Address Register (CACR).

2.2.2 8-Bit USC Register Access

As the USC has a configurable bus interface, the USC must be set to match the 8-bit non-multiplex interface implementation of the PMC-SIO4AR. This setup information must be programmed into the USC Bus Configuration Register (BCR) upon initial power up and following every hardware reset of the USC. The BCR is accessible only following a USC hardware reset – the first write to the USC following a USC Reset programs the BCR. Even though the Zilog manual states the BCR has no specific address, the driver must use the channel USC base address – 0x100 for Ch 1 & Ch 2, 0x300 for Ch 3 & Ch 4 – as the BCR address. Failure to do so may result in improper setup. Since the user interface to the USC is an 8 bit interface, the software only needs to set the lower byte to 0x00 (hardware implementation will program the upper byte of the BCR).

2.2.3 USC Data Transfer

Although the Z16C30 USC contains 32 byte internal FIFOs for data transfer, these are typically not used on the SIO4A. Since the PMC-SIO4AR has much deeper external FIFOs (or internal FPGA FIFOs), the internal USC FIFOs are setup to immediately transfer data to/from the external FIFOs. Immediate transfer of received data to the external FIFOs eliminates the possibility of data becoming “stuck” in the USC internal receive FIFOs, while bypassing the USC internal transmit FIFOs ensures better control of the transmit data.

In order to automatically transfer data to and from the external FIFOs, the USC should use DMA to request a data transfer whenever one byte is available in the USC internal FIFOs. This “DMA” should not be confused with the

DMA of data from the PMC-SIO4AR external FIFOs to the PMC interface. To accomplish the USC-to-External FIFO DMA transfer, the TxReq/RxReq pins should be set as DMA Requests in the IOCR, and the TxAck/RxAck pins should be set as DMA Acknowledge inputs in the HCR. In addition, the Tx Request Level should be set to 0x1F (31) using TCSR/TICR and the Rx Request Level should be set to 0 using RCSR/RICR. See Z16C30 manual for further details on programming the DMA request levels.

2.2.4 USC Register Memory Map

To access the USC in 8-bit mode, the driver is required to access the upper and lower bytes of each register independently. The odd address byte will access the upper byte of each register (D15-D8), and the even address byte will access the lower byte (D7-D0). Each USC register must be accessed independently as a byte access– the software cannot perform word or long word accesses to the USC registers.

The USC register map is provided below. The Channel Offset Address depicted is from the Channel Base Address – (Ch 1 Base Address = 0x100, Ch 2 Base Address = 0x200, Ch 3 Base Address = 0x300, Ch 4 Base Address = 0x400). For further programming details, please refer to the Zilog Z16C30 data books.

| Channel Offset Address | Access* | Register Name |
|------------------------|--------------|---|
| 0x01 / 0x00 | CCAR Hi / Lo | Channel Command / Address Register |
| 0x03 / 0x02 | CMR Hi / Lo | Channel Mode Register |
| 0x05 / 0x04 | CCSR Hi / Lo | Channel Command / Status Register |
| 0x07 / 0x06 | CCR Hi / Lo | Channel Control Register |
| 0x11 / 0x10 | CMCR Hi / Lo | Clock Mode Control Register |
| 0x13 / 0x12 | HCR Hi / Lo | Hardware Configuration Register |
| 0x19 / 0x18 | IOCR Hi / Lo | I/O Control Register |
| 0x1B / 0x1A | ICR Hi / Lo | Interrupt Control Register |
| 0x1D / 0x1C | MISR Hi / Lo | Miscellaneous Interrupt Status Register |
| 0x1F / 0x1E | SICR Hi / Lo | Status Interrupt Control Register |
| 0x20 | RDR | Receive Data Register |
| 0x23 / 0x22 | RMR | Receive Mode Register |
| 0x25 / 0x24 | RCSR Hi / Lo | Receive Command / Status Register |
| 0x27 / 0x26 | RICR Hi / Lo | Receive Interrupt Control Register |
| 0x29 / 0x28 | RSR Hi / Lo | Receive Sync Register |
| 0x2B / 0x2A | RCLR Hi / Lo | Receive Count Limit Register |
| 0x2D / 0x2C | RCCR Hi / Lo | Receive Character Count Register |
| 0x2F / 0x2E | TC0R | Time Constant 0 Register |
| 0x30 | TDR | Transmit Data Register |
| 0x33 / 0x32 | RMR | Transmit Mode Register |
| 0x35 / 0x34 | TCSR Hi / Lo | Transmit Command / Status Register |
| 0x37 / 0x36 | TICR Hi / Lo | Transmit Interrupt Control Register |
| 0x39 / 0x38 | TSR Hi / Lo | Transmit Sync Register |
| 0x3B / 0x3A | TCLR Hi / Lo | Transmit Count Limit Register |
| 0x3D / 0x3C | TCCR Hi / Lo | Transmit Character Count Register |
| 0x3F / 0x3E | TC1R | Time Constant 1 Register |

CHAPTER 3: PCI/PMC INTERFACE

3.0 PCI/PMC Interface Registers

A PCI9080 I/O Accelerator from PLX Technology handles the PCI/PMC Interface. The PCI/PMC interface is compliant with the 5V, 33MHz 32-bit PCI Specification 2.1. The PCI9080 provides dual DMA controllers for fast data transfers to and from the on-board FIFOs. Fast DMA burst accesses provide for a maximum burst throughput of 132MB/s to the PCI interface. To reduce CPU overhead during DMA transfers, the controller also implements Chained (Scatter/Gather) DMA, as well as Demand Mode DMA.

Since many features of the PCI9080 are not utilized in this design, it is beyond the scope of this document to duplicate the [PCI9080 User's Manual](#). Only those features, which will clarify areas specific to the PMC-SIO4AR are detailed here. Please refer to the [PCI9080 User's Manual](#) (See Related Publications) for more detailed information. Note that the BIOS configuration and software driver will handle most of the PCI9080 interface. Unless the user is writing a device driver, the details of this PCI/PMC Interface Chapter may be skipped.

3.1 PCI Registers

The PLX 9080 contains many registers, many of which have no effect on the PMC-SIO4AR performance. The following section attempts to filter the information from the PCI9080 manual to provide the necessary information for a PMC-SIO4AR specific driver.

The PMC-SIO4AR uses an on-board serial EEPROM to initialize many of the PCI9080 registers after a PCI Reset. This allows board specific information to be preconfigured.

3.1.1 PCI Configuration Registers

The PCI Configuration Registers allow the PCI controller to identify and control the cards in a system.

PCI device identification is provided by the Vendor ID/Device ID (Addr 0x0000) and Sub-Vendor ID/Sub-Device ID Registers (0x002C). The following definitions are unique to the General Standards SIO4A boards. All drivers should verify the ID/Sub-ID information before attaching to this card. These values are fixed via the Serial EEPROM load following a PCI Reset, and cannot be changed by software.

| | | |
|---------------|--------|----------------|
| Vendor ID | 0x10B5 | PLX Technology |
| Device ID | 0x9080 | PCI9080 |
| Sub-Vendor ID | 0x10B5 | PLX Technology |
| Sub-Device ID | 0x2401 | GSC SIO4 |

The configuration registers also setup the PCI IO and Memory mapping for the PMC-SIO4AR. The PCI9080 is setup to use PCIBAR0 and PCIBAR1 to map the internal PLX registers into PCI Memory and IO space respectively. PCIBAR2 will map the Local Space Registers into PCI memory space, and PCIBAR3 is unused. Typically, the OS will configure the PCI configuration space.

For further information of the PCI configuration registers, please consult the [PLX Technology PCI9080 Manual](#).

3.1.2 Local Configuration Registers

The Local Configuration registers give information on the Local side implementation. These include the required memory size. The SIO4 memory size is initialized to 4k Bytes. All other Local Registers initialize to the default values described in the PCI9080 Manual.

3.1.3 Runtime Registers

The Runtime registers consist of mailbox registers, doorbell registers, and a general-purpose control register. The mailbox and doorbell registers are not used and serve no purpose on the PMC-SIO4AR. All other Runtime Registers initialize to the default values described in the PCI9080 Manual.

3.1.3 DMA Registers

The Local DMA registers are used to setup the DMA transfers to and from the on-board FIFOs. DMA is supported only to the four FIFO locations. The PMC-SIO4AR supports both Demand (DREQ# controlled) and Non-Demand mode DMA. Both Channel 0 and Channel 1 DMA are supported.

3.1.3.1 DMA Channel Mode Register: (PCI 0x80 / 0x94)

The DMA Channel Mode register must be setup to match the hardware implementation

| Bit | Description | Value | Notes |
|--------|--------------------------------------|--------------------------|---|
| D1:0 | Local Bus Width | 11 = 32 bit | Although the serial FIFOs only contain 8 bits of data, the register access is still a 32bit access. While it may be possible to “pack” the data by setting the Local Bus Width to 8, it is beyond the scope of this manual to determine if/how this could work. |
| D5:2 | Internal Wait States | 0000 = Unused | |
| D6 | Ready Input Enable | 1 = Enabled | |
| D7 | Bterm# Input Enabled | 0 = Unused | |
| D8 | Local Burst Enable | 1 = Supported | Bursting allows fast back-to-back accesses to the FIFOs to speed throughput |
| D9 | Chaining Enable (Scatter Gather DMA) | X | DMA source addr, destination addr, and byte count are loaded from memory in PCI Space. |
| D10 | Done Interrupt Enable | X | DMA Done Interrupt |
| D11 | Local Addressing Mode | 1 = No Increment | DMA to/from FIFOs only |
| D12 | Demand Mode Enable | X | Demand Mode DMA is supported for FIFO accesses on the PMC-SIO4AR. (See Section 4.3) |
| D13 | Write & Invalidate Mode | X | |
| D14 | DMA EOT Enable | 0 = Unused | |
| D15 | DMA Stop Data Transfer Enable | 0 = BLAST terminates DMA | |
| D16 | DMA Clear Count Mode | 0 = Unused | |
| D17 | DMA Channel Interrupt Select | X | |
| D31:18 | Reserved | 0 | |

CHAPTER 4: PROGRAMMING

4.0 Introduction

This section addresses common programming questions when developing an application for the SIO4.

4.1 Resets

Each serial channel provides control for three unique reset sources: a USC Reset, a Transmit FIFO Reset, and a Receive FIFO Reset. All three resets are controlled from the GSC Channel Control/Status Registers.

Section 2.2.1 provides information on the USC Reset. It is important to realize that since each Zilog Z16C30 chip contains two serial channels, a USC Reset to either channel will reset the entire chip (both channels affected). Due to the limitation of a USC Reset to affecting two channels, it is recommended that a single USC Channel be Reset via the RTRreset bit of the USC Channel Command/Address Register (CCAR).

The FIFO resets allow each individual FIFO (Tx and Rx) to be reset independently. In addition to clearing the FIFO, the Almost Empty and Almost Full flags are programmed following a FIFO reset. If the FIFO Almost Values are set to zero prior to a FIFO reset, the Almost Empty/Full flags are NOT programmed (the FIFO Flags will be set at their default value of 7). Section 4.2 provides further information about the FIFO Almost Flags.

4.2 FIFO Almost Flags

The FIFO Almost Empty and Almost Full flags on the PMC-SIO4AR provide a way for the user to approximate the amount of data in the FIFO. This is useful in determining when a data transfer is complete or determining the amount of data which can be safely read from/written to the FIFO. The flags are also used during Demand Mode DMA to set the request levels. The Almost Flags are fully programmable by the user.

Each channel provides two 32 bit registers for setting the Almost Full/Empty values: the TX FIFO Almost Register (See Section 2.1.5) and the RX FIFO Almost Register (See Section 2.1.6). Each of these registers is further divided into two 16 bit words: D31-D16 = Almost Full Value; D15-D0 = Almost Empty Value.

The Almost Flag value represents the number of bytes from each respective “end” of the FIFO. The Almost Empty value represents the number of bytes from empty, and the Almost Full value represents the number of bytes from full (NOT the number of bytes from empty). For example, the default value of “0x0007 0007” in the FIFO Almost Register means that the Almost Empty Flag will indicate when the FIFO holds 0x0007 bytes or fewer, and will transition as the 8th byte is read or written. The Almost Full Flag indicates the FIFO contains (FIFO Size – 0x7) bytes or more. For the standard 32Kbyte FIFO, an Almost Full value of 0x7 will cause the Almost Full flag to be asserted when the FIFO contains 32761 (32k – 7) or more bytes of data .

The values placed in the FIFO Almost Registers are programmed to the FIFO chips whenever a FIFO reset is performed. The proper steps to program these values are:

- Set the respective FIFO Almost Register(s)
- Perform a FIFO reset of the respective FIFO
- The value in the almost register is now programmed into the FIFO chips

If either (or both) FIFO Almost Empty Register or FIFO Almost Full Register is set to a value of 0x0000 during a FIFO reset, the Almost Flags will not be programmed and the flags will be set to the FIFO chip default of 7 bytes from empty and 7 bytes from full.

4.3 DMA

The PCI DMA functionality allows data to be transferred between host memory and the PMC-SIO4AR onboard FIFOs with the least amount of CPU overhead. The PCI9080 bridge chip handles all PCI DMA functions, and the device driver should handle the details of the DMA transfer.

There are two DMA modes – Demand Mode DMA and Non-Demand Mode DMA. Demand Mode DMA refers to data being transferred on demand. For receive, this means data will be transferred as soon as it is received into the FIFO. Likewise, for transmit, data will be transferred to the FIFOs as long as the FIFO is not Almost Full. The disadvantage to Demand Mode DMA is that the DMA transfers are dependent on the user data interface. If the user data interface hangs, the Demand mode DMA transfer will also stop. If a timeout occurs, there is no way to determine the exact amount of data transferred before it was aborted.

On the other hand, Non-Demand Mode DMA does not check the FIFO empty/full flags before or during the data transfer – it simply assumes there is enough available FIFO space to complete the transfer. If the transfer size is larger than the available data, the transfer will complete with invalid results.

Non-Demand Mode DMA is the safer mode, but it may require more user intervention to ensure the requested data size is available. Demand Mode DMA requires less software control, but runs the risk of losing data due to an incomplete transfer. The user will need to evaluate each situation separately to decide whether DMA is necessary, and if so, which mode.

4.4 Interrupts

The PMC-SIO4AR has a number of interrupt sources which are passed to the host CPU via the PCI Interrupt A. Since there is only one physical interrupt source, the interrupts pass through a number of “levels” to get multiplexed onto this single interrupt. The interrupt originates in the PCI9080 PCI Bridge, which combines the internal PLX interrupt sources (DMA) with the local space interrupt. The driver will typically take care of setting up and handling the PCI9080 interrupts. The single Local Interrupt is made up of the interrupt sources described in Section 2.1.10. In addition, the Zilog USC contains a number of interrupt sources which are combined into a single Local Interrupt. The user should be aware that interrupts must be enabled at each level for an interrupt to occur. For example, if a USC interrupt is used, it must be setup and enabled in the USC, enabled in the GSC Firmware Interrupt Control Register, and enabled in the PCI9080. In addition, the interrupt must be acknowledged and/or cleared at each level following the interrupt.

4.5 Clock Setup

Figure 4-1 shows the relationship of the various clock sources on the PMC-SIO4AR board. These clock sources can be most simply viewed in three sections: On-Board Programmable Clocks, IO connector Clocks, and USC Clocks.

The Programmable Clocks consist of a single on-board programmable oscillator and four post divide clocks (one for each channel). The single programmable oscillator clock is used as the input for each of the programmable clock post dividers, which will allow each channel to have a unique programmable clock input. These programmable clocks are further described in sections 2.1.12 and 4.6.

The IO Connector Clocks consist of the cable RxClk and cable TxClk for each channel. The RxClk is always an input and may be used as a clock source for either the cable TxClk or the USC Clocks. The cable TxClk is always an output. The cable TxClk source is set in the Pin Source register. See Section 2.1.11 for further information.

The USC Clocks (USC RxClk and USC TxClk) are bidirectional signals. Even though the names of these clocks seem to imply a receive clock and a transmit clock, both clocks are fully programmable and identical in function – either clock may be used for transmit or receive. The USC clocks may be sourced from either the USC or the FPGA (via the Pin Source register). The user must be careful to ensure that both the USC and Pin Source Register are setup to

agree. If a USC clock is set as an output in the USC, it should be programmed as an input in the Pin Source register. Likewise, if a USC clock source is driven from the Pin Source register, the user should program the pin as an input to the USC. Section 2.1.11 describes the Pin Source Registers.

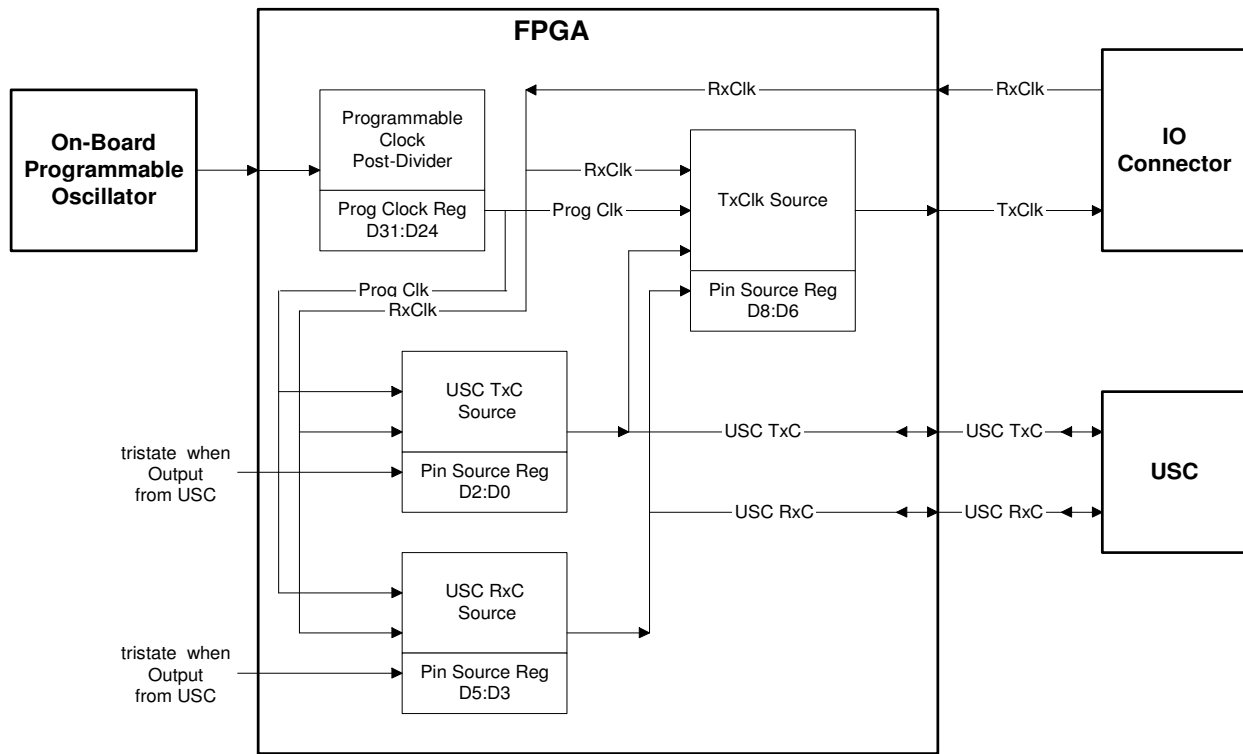


Figure 4-1 – Clock Configuration

The degree of sophistication for the programmable clocks on the PMC-SIO4AR provides flexibility to handle almost any clock configuration scenario. However, this flexibility can also complicate the clock setup, especially for simple setups. The following guidelines are typical asynchronous and synchronous setups which should work for most setups.

In asynchronous mode, the clock does not need to be transmitted with the data. Therefore, the USC Clock pins will be used for the input baud rate clock. Since the RxC and TxC pins have identical functions, the RxC and TxC pins may be used interchangeably. The async baud rate clock will be 16x / 32x / or 64x the actual baud rate due to the async oversampling. This oversample rate is set in the USC Channel Mode Register when async mode is selected. The simplest method will be to program the channel programmable clock to be 16/32/64 times the desired baudrate and use this clock as the source for the TxC/RxC pin. Section 2.1.11 describes how to program the Pin Source Register to set $TxC / RxC = \text{Programmable Clock}$. The USC should be programmed such that TxC / RxC is an input (in the USC I/O Control Register), and the USC baudrate generator will be bypassed completely. If both Rx and Tx are operating at the same baud rate, the same USC clock pin can be used for both the transmit and receive clocks.

For synchronous modes, the clock is transmitted and received on the cable along with the data. This can present a problem since the USC only has two clock pins. Since one clock is necessary for receive clock and the other is necessary for the transmit clock, there is no clock pin available for an input to the USC baud rate generators. The on-board programmable clocks provide a solution for this situation. By using the programmable oscillator and the programmable clock post-divider, the on-board programmable clock can usually be set directly to the desired transmit baud rate. The USC TxC pin and the Cable TxClk are both set equal to the Programmable Clock in the Pin Source Register. The USC RxC pin is used for the receive clock from the cable interface, so it will be set to the

cable RxClk in the Pin Source Register. Since the FPGA will source both USC clocks, they must be programmed as inputs in the USC I/O Control Register.

The preceding suggestions should work for most applications. The default Pin Source Register value should set the clocks to work with both scenarios – USC TxC pin = Programmable Clock, USC RxC Pin = Cable RxClk, Cable TxClk = Programmable Clock. (For async, use USC TxC is input clock).

4.6 Programmable Oscillator / Programmable Clocks

The On-Board Programmable Oscillator is programmed by writing to the Channel 1 Programmable Clock Register at Local Address offset 0x00A0 (See Section 2.1.12). In order to program the oscillator, it is necessary to calculate a “control word” to program the correct oscillator frequency. Cypress Semiconductor provides a Windows program called BitCalc to assist in calculating this value. This program should be included with the documentation package. If not, it is available from General Standards (contact tech support). The clock generator device is an ICD2053B with an input frequency of 20MHz. The default oscillator value at power up or following a board reset is 20MHz.

The output frequency of the programmable oscillator may be programmed from 20MHz to a minimum of about 400 kHz. Since a slower oscillator than 400 KHz may be required, bits D31:D24 of the Programmable Clock Registers allow the user to further divide the input clock. There is a separate post-divider for each serial channel, so each channel can have a unique clock frequency.

For further information, Section 2.1.12 details the Programmable Clock registers. Section 4.5 Clock Setup provides information on how the programmable clocks can be best utilized. Appendix A provides common frequencies and their associated programmable oscillator values. Appendix B describes how to calculate programmable clock register values for frequencies other than those listed in Appendix A.

4.7 Upper/Lower Connector Naming Convention

Since all the cable transceivers are bidirectional, the serial Data and Clock signals can be transmitted or received on two separate IO connector pins. The naming convention “Upper” and “Lower” is used in order to differentiate between these two pins with identical function. Typically, one pin is used for receive data, and the other pin is used for transmit. Separate controls for the transmit / receive enables allow the user flexibility to monitor the transmit line or perform a standalone loop back test. This also allows two SIO4 boards to be connected directly or two channels to be connected directly using standard cabling options by simply configuring the transmit and receive pins correctly.

Figure 4-2 shows the controls for the Upper/Lower Clock and Data signals. The clock enables are controlled from the GSC Clock Control Register (Section 2.1.4) and the data signals are enabled in the GSC Channel Control/Status Registers (Section 2.1.8). Even though the clock and data lines have separate enables, they will typically be set the same. For example, if you want to transmit on the Channel 1 Upper signals and receive on the Channel 1 Lower signals, the Upper Tx Clock and Upper Tx Data will be enabled for transmit; and the Lower Rx Clock and Lower Rx Data should be enabled for receive.

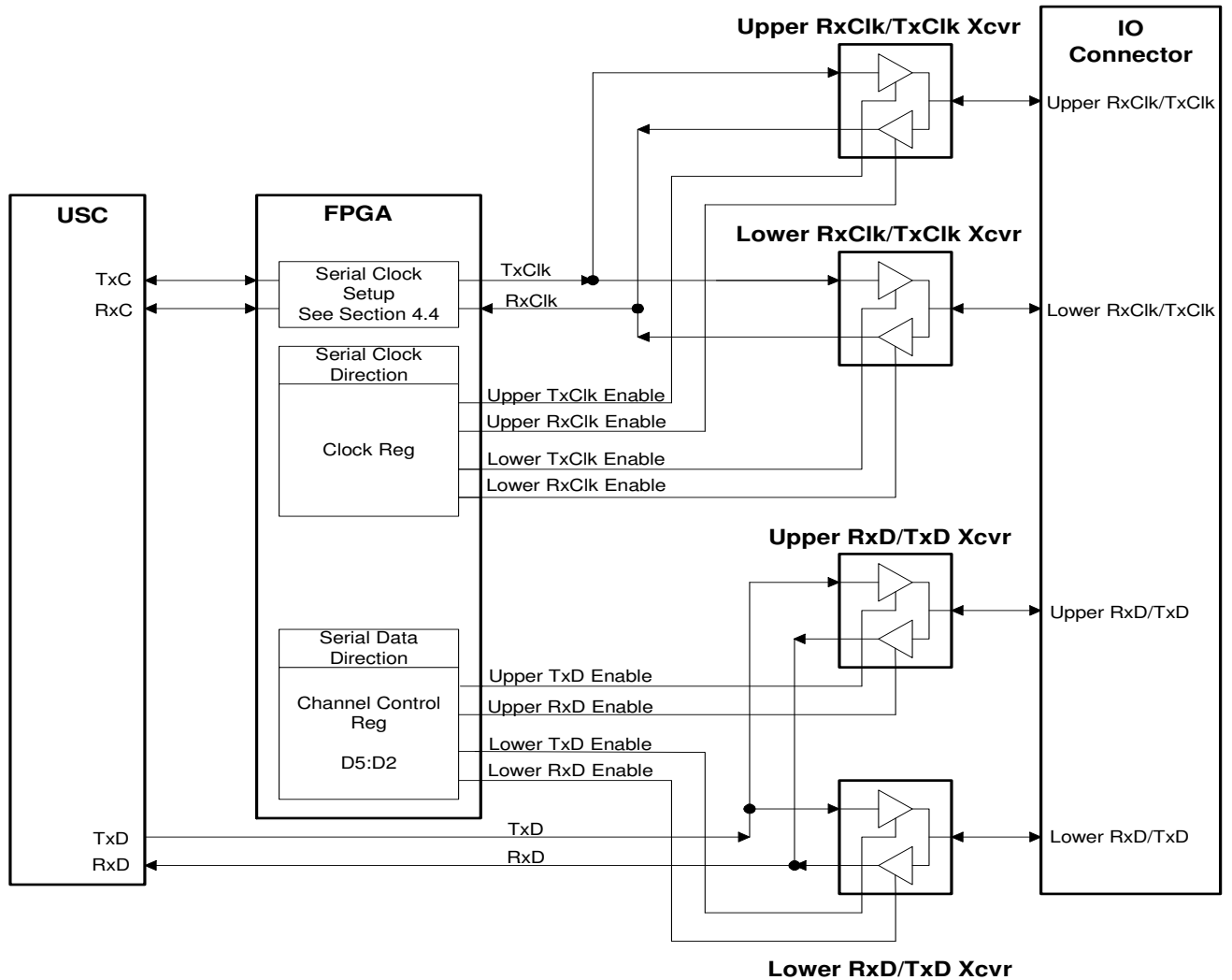


Figure 4-2 – Upper / Lower Clock and Data Configuration

4.8 CTS/DCD

The CTS and DCD signals are controlled directly from the USC IO Control Register (IOCR). They may be set as inputs for their dedicated function, or may be used as general purpose IO.

The CTS transceiver direction is controlled from the Pin Source Register (Section 2.1.11). The user must ensure that the Pin Source CTS direction agrees with the CTS setup in the USC. If CTS is set as an output from the USC chip, it should also be set as an output to the cable (or disabled) in the Pin Source register.

The DCD transceiver direction is controlled by the Lower Data transceiver enable. This means that if the DCD input function is used, the channel must have the Lower Data receive enabled (D5 of Channel Control/Status Register). Likewise, for DCD to be an output, the Lower Data transmit must be enabled.

Since the CTS and DCD signals can be driven from either the USC or the cable transceiver, the user must be careful to ensure both devices are not driving these signals at the same time (USC output and transceiver input). Figure 4-3 shows the CTS and DCD configuration.

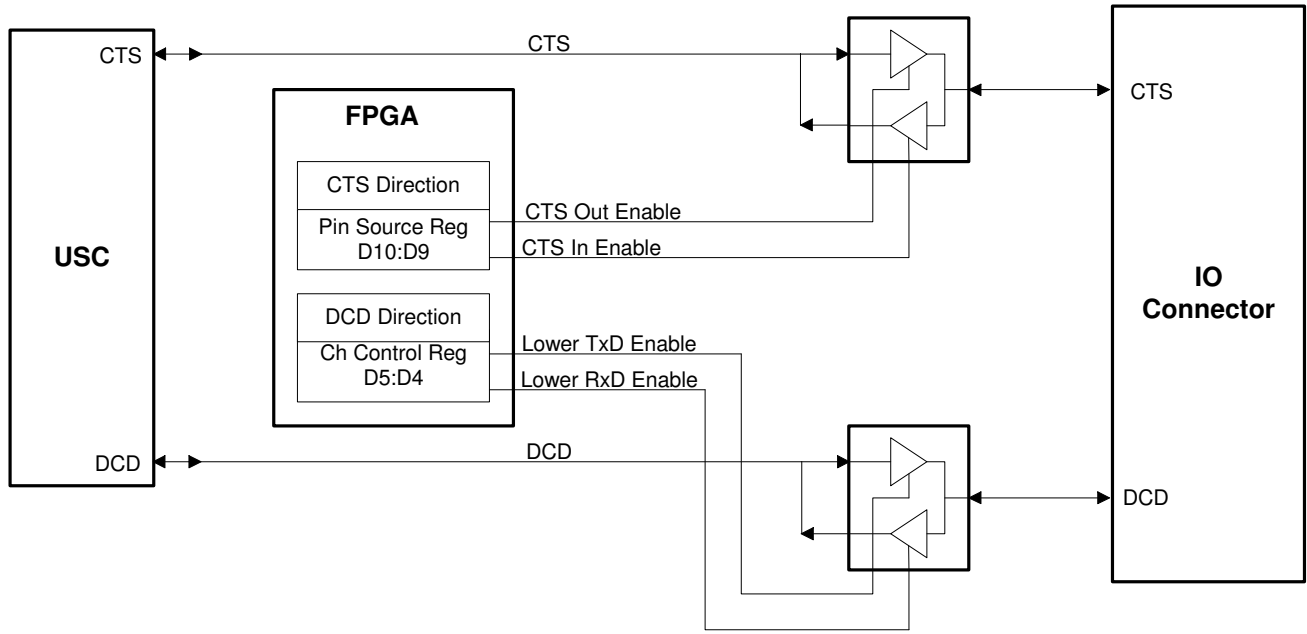


Figure 4-3 – CTS / DCD Configuration

CHAPTER 5: HARDWARE CONFIGURATION

5.1 Board Layout

The following figure is a drawing of the physical components of the PMC-SIO4AR:

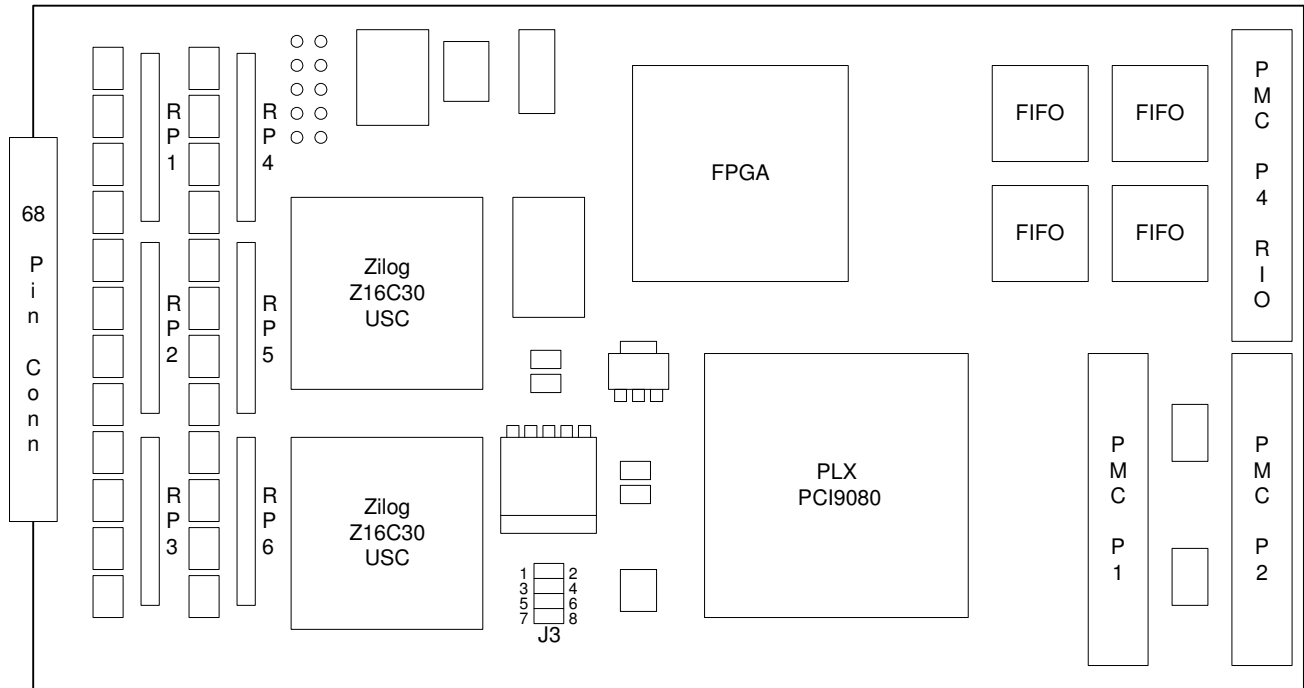


Figure 5-1: Board Layout

5.2 Board ID Jumper J3

Jumper J3 allows the user to set the Board ID in the GSC Board Status Register (See Section 2.1.3). This is useful to uniquely identify a board if more than one PMC-SIO4AR card is in a system. When the Board ID jumper is installed, it will read '0' in the Board Status Register. The Board Status Register bit will report '1' when the jumper is removed. Refer to Figure 5.1-1 for Jumper J3 location.

| J3 Jumper | Description | Notes |
|-----------|--------------------------|---|
| 1 - 2 | Board ID 0 | Defines Board ID 0 In Board Status Register |
| 3 - 4 | Board ID 1 | Defines Board ID 1 In Board Status Register |
| 5 - 6 | Reload FPGA on PCI Reset | Should be installed for normal operation |
| 7 - 8 | PLX EEPROM Config | Must be installed for normal operation |

Jumpers 5/6 and 7/8 are installed at the factory and should not be removed for normal operation.

5.3 Interface Connectors

There are two user interface connections on the PMC-SIO4AR - A SCSI II type 68-pin connector (female) mounted to the front edge of the board (P2), and the PMC P4 Rear IO Connector. The part number for the 68 pin front edge connector is AMP 787170-7. The mating connector is AMP 749111-6 or equivalent. The pin out for both connectors is shown below.

| P2, Row A | Pin # | P2, Row B | Pin # |
|-------------------------------|-------|-------------------------------|-------|
| No connect | 1 | No connect | 35 |
| No connect | 2 | No connect | 36 |
| No connect | 3 | No connect | 37 |
| No connect | 4 | No connect | 38 |
| Ch1 CTS + | 5 | Ch3 CTS + | 39 |
| Ch1 CTS - | 6 | Ch3 CTS - | 40 |
| Ch1 Lower Cable TxD/RxD + | 7 | Ch3 Lower Cable TxD/RxD + | 41 |
| Ch1 Lower Cable TxD/RxD - | 8 | Ch3 Lower Cable TxD/RxD - | 42 |
| Ch1 Lower Cable TxClk/RxClk + | 9 | Ch3 Lower Cable TxClk/RxClk + | 43 |
| Ch1 Lower Cable TxClk/RxClk - | 10 | Ch3 Lower Cable TxClk/RxClk - | 44 |
| Ch1 DCD + | 11 | Ch3 DCD + | 45 |
| Ch1 DCD - | 12 | Ch3 DCD - | 46 |
| Ch1 Upper Cable TxD/RxD + | 13 | Ch3 Upper Cable TxD/RxD + | 47 |
| Ch1 Upper Cable TxD/RxD - | 14 | Ch3 Upper Cable TxD/RxD - | 48 |
| Ch1 Upper Cable TxClk/RxClk + | 15 | Ch3 Upper Cable TxClk/RxClk + | 49 |
| Ch1 Upper Cable TxClk/RxClk - | 16 | Ch3 Upper Cable TxClk/RxClk - | 50 |
| GND | 17 | GND | 51 |
| GND | 18 | GND | 52 |
| Ch2 CTS + | 19 | Ch4 CTS + | 53 |
| Ch2 CTS - | 20 | Ch4 CTS - | 54 |
| Ch2 Lower Cable TxD/RxD + | 21 | Ch4 Lower Cable TxD/RxD + | 55 |
| Ch2 Lower Cable TxD/RxD - | 22 | Ch4 Lower Cable TxD/RxD - | 56 |
| Ch2 Lower Cable TxClk/RxClk + | 23 | Ch4 Lower Cable TxClk/RxClk + | 57 |
| Ch2 Lower Cable TxClk/RxClk - | 24 | Ch4 Lower Cable TxClk/RxClk - | 58 |
| Ch2 DCD + | 25 | Ch4 DCD + | 59 |
| Ch2 DCD - | 26 | Ch4 DCD - | 60 |
| Ch2 Upper Cable TxD/RxD + | 27 | Ch4 Upper Cable TxD/RxD + | 61 |
| Ch2 Upper Cable TxD/RxD - | 28 | Ch4 Upper Cable TxD/RxD - | 62 |
| Ch2 Upper Cable TxClk/RxClk + | 29 | Ch4 Upper Cable TxClk/RxClk + | 63 |
| Ch2 Upper Cable TxClk/RxClk - | 30 | Ch4 Upper Cable TxClk/RxClk - | 64 |
| No connect | 31 | No connect | 65 |
| No connect | 32 | No connect | 66 |
| No connect | 33 | No connect | 67 |
| No connect | 34 | No connect | 68 |

Table 5-1: Front Edge User Cable Pin-Out

| P2, Row A | Pin # | P2, Row B | Pin # |
|-------------------------------|--------------|-------------------------------|--------------|
| Ch1 CTS + | 1 | Ch3 CTS + | 33 |
| Ch1 CTS - | 2 | Ch3 CTS - | 4 |
| Ch1 Lower Cable TxD/RxD + | 3 | Ch3 Lower Cable TxD/RxD + | 35 |
| Ch1 Lower Cable TxD/RxD - | 4 | Ch3 Lower Cable TxD/RxD - | 36 |
| GND | 5 | GND | 37 |
| GND | 6 | GND | 38 |
| Ch1 Lower Cable TxClk/RxClk + | 7 | Ch3 Lower Cable TxClk/RxClk + | 39 |
| Ch1 Lower Cable TxClk/RxClk - | 8 | Ch3 Lower Cable TxClk/RxClk - | 40 |
| Ch1 DCD + | 9 | Ch3 DCD + | 41 |
| Ch1 DCD - | 10 | Ch3 DCD - | 42 |
| Ch1 Upper Cable TxD/RxD + | 11 | Ch3 Upper Cable TxD/RxD + | 43 |
| Ch1 Upper Cable TxD/RxD - | 12 | Ch3 Upper Cable TxD/RxD - | 44 |
| GND | 13 | GND | 45 |
| GND | 14 | GND | 46 |
| Ch1 Upper Cable TxClk/RxClk + | 15 | Ch3 Upper Cable TxClk/RxClk + | 47 |
| Ch1 Upper Cable TxClk/RxClk - | 16 | Ch3 Upper Cable TxClk/RxClk - | 48 |
| Ch2 CTS + | 17 | Ch4 CTS + | 49 |
| Ch2 CTS - | 18 | Ch4 CTS - | 50 |
| Ch2 Lower Cable TxD/RxD + | 19 | Ch4 Lower Cable TxD/RxD + | 51 |
| Ch2 Lower Cable TxD/RxD - | 20 | Ch4 Lower Cable TxD/RxD - | 52 |
| GND | 21 | GND | 53 |
| GND | 22 | GND | 54 |
| Ch2 Lower Cable TxClk/RxClk + | 23 | Ch4 Lower Cable TxClk/RxClk + | 55 |
| Ch2 Lower Cable TxClk/RxClk - | 24 | Ch4 Lower Cable TxClk/RxClk - | 56 |
| Ch2 DCD + | 25 | Ch4 DCD + | 57 |
| Ch2 DCD - | 26 | Ch4 DCD - | 58 |
| Ch2 Upper Cable TxD/RxD + | 27 | Ch4 Upper Cable TxD/RxD + | 59 |
| Ch2 Upper Cable TxD/RxD - | 28 | Ch4 Upper Cable TxD/RxD - | 60 |
| GND | 29 | GND | 61 |
| GND | 30 | GND | 62 |
| Ch2 Upper Cable TxClk/RxClk + | 31 | Ch4 Upper Cable TxClk/RxClk + | 63 |
| Ch2 Upper Cable TxClk/RxClk - | 32 | Ch4 Upper Cable TxClk/RxClk - | 64 |

Table 5-2: Rear IO PMC P4 Pin-Out

5.4 Termination Resistors

The RS485/RS422 cable interface requires termination of the differential signals to match the cable impedance. The PMC-SIO4AR is shipped with 150 Ohm termination resistor SIPs. These resistors are socketed so they can be changed or removed as necessary. There are six termination resistors – RP1 through RP6. The termination resistors are standard 8-pin isolated resistor SIPs (four resistors per SIP). Refer to Figure 5.1-1 for resistor pack locations.

CHAPTER 6: ORDERING OPTIONS

6.0 Ordering Information

Since the PMC-SIO4AR is designed to fit a variety of high-speed serial interface needs, there are several options that must be specified when ordering the PMC-SIO4AR board. Please consult our sales department with your application requirements to determine the correct ordering option. (sales@generalstandards.com).

6.0.1 FIFO Size

The SIO4 can accept FIFOs with depths ranging from 512 bytes to 32k bytes. Larger FIFO depth is important for faster interfaces to reduce the risk of data loss due to software overhead. The PMC-SIO4AR can be ordered with the following FIFO depths: 512 bytes, 1kbytes, 2kbytes, 4kbytes, 8kbytes, 16kbytes, or 32kbytes. Note that the FIFO size option in the board part number refers to the total FIFO size for all 8 channels, not the FIFO size of a single FIFO. For example, PMC-SIO4AR-8K would contain eight 1k deep FIFOs. Please consult our sales department for pricing and availability.

6.0.2 Interface Cable

General Standards Corporation can provide an interface cable for the PMC-SIO4AR board. This standard cable is a non-shielded twisted pair ribbon cable for increased noise immunity. Several standard cable lengths are offered, or the cable length can be custom ordered to the user's needs. Versions of the cable are available with connectors on both ends, or the cable may be ordered with a single connector to allow the user to adapt the other end for a specific application. A standard cable is available which will breakout the serial channels into four DB25 connectors. Shielded cable options are also available. Please consult our sales department for more information on cabling options and pricing.

6.0.3 Device Drivers

General Standards has developed many device drivers for The PMC-SIO4AR boards, including VxWorks, Windows, Linux, and LabView. As new drivers are always being added, please consult our website (www.generalstandards.com) or consult our sales department for a complete list of available drivers and pricing.

6.1 Custom Applications

Although the PMC-SIO4AR board provides extensive flexibility to accommodate most user applications, a user application may require modifications to conform to a specialized user interface. General Standards Corporation has worked with many customers to provide customized versions based on the PMC-SIO4AR boards. Please consult our sales department with your specifications to inquire about a custom application.

APPENDIX A: COMMON PROGRAMMABLE CLOCK REGISTER VALUES

The following table lists the Programmable Clock Register Values for common clock frequencies.

| Frequency | Programmable Clock Value | Post Divide Value | Programmable Oscillator Frequency |
|-------------|--------------------------|-------------------|-----------------------------------|
| 20 MHz | 00129128 | 0 | 20 MHz |
| 18.432 MHz | 002D9310 | 0 | 18.432 MHz |
| 18 MHz | 00109120 | 0 | 18 MHz |
| 16 MHz | 000E9120 | 0 | 16 MHz |
| 15 MHz | 000D9120 | 0 | 15 MHz |
| 14.7456 MHz | 00351480 | 0 | 14.7456 MHz |
| 12 MHz | 00169928 | 0 | 12 MHz |
| 11.059MHz | 00381B28 | 0 | 11.059MHz |
| 10 MHz | 00129928 | 0 | 10 MHz |
| 8 MHz | 000E9928 | 0 | 8 MHz |
| 7.3728 MHz | 002A1B90 | 0 | 7.3728MHz |
| 6 MHz | 0016A128 | 0 | 6 MHz |
| 5 MHz | 0012A128 | 0 | 5 MHz |
| 4 MHz | 000EA120 | 0 | 4 MHz |
| 3.6864 MHz | 002A2390 | 0 | 3.6864MHz |
| 2 MHz | 000EA920 | 0 | 2 MHz |
| 1.8432 MHz | 00372CB0 | 0 | 1.8432MHz |
| 1 MHz | 000EB120 | 0 | 1 MHz |
| 921.6 kHz | 002A3390 | 0 | 921.6kHz |
| 500 kHz | 000EB920 | 0 | 500 kHz |
| 460.8 kHz | 002A3B90 | 0 | 460.8kHz |
| 250 kHz | 0210EB920 | 2 | 500 kHz |
| 230.4 kHz | 022A3B90 | 2 | 460.8kHz |
| 115.2 kHz | 042A3B90 | 4 | 460.8kHz |
| 100 kHz | 050EB920 | 5 | 500 kHz |
| 5.76 kHz | 082A3B90 | 8 | 460.8kHz |
| 50 kHz | 0A0EB920 | 10 | 500 kHz |
| 38.4 kHz | 1038BB90 | 16 | 460.8kHz |
| 19.2 kHz | 2038BB90 | 32 | 460.8kHz |
| 10 kHz | 320EB920 | 50 | 500 kHz |
| 9.6 kHz | 4038BB90 | 64 | 460.8kHz |
| 5 kHz | 640EB920 | 100 | 500 kHz |
| 4.8 kHz | 8038BB90 | 128 | 460.8kHz |
| 2.5 kHz | C80EB920 | 200 | 500 kHz |
| 2.4 kHz | 0838BB90 | 200 | 480kHz |
| 2 kHz | C80EB970 | 200 | 400kHz |

APPENDIX B: CALCULATING PROGRAMMABLE CLOCK VALUES

The values in Appendix A were derived using the BitCalc program from Cypress Semiconductor for an ICD2053B with an input clock frequency of 20MHz. The correct BitCalc value to use is referred to as the “Hex word (Unstuffed)”. If a desired clock frequency is less than 400kHz (min frequency of programmable oscillator), the FPGA post divider (D31:D24) may be used to further reduce the clock. The programmable oscillator frequency will be set to a multiple of the desired frequency which is greater than 400kHz. Using this method, you should be able to achieve any frequency between 20MHz and 1.57kHz (400kHz / 255). If frequencies slower than 1.57kHz are required, the baud rate generators of the USC may also be used.

EXAMPLE:

To obtain a frequency of 9600Hz.

Since $9600 < 400k$, we will need to use the post-divider. Choose a post divide value to obtain a programmable oscillator frequency greater than 400kHz. For a value of 64 (0x40):

$$9600 * 64 = 614400\text{Hz}$$

So the on-board programmable oscillator will be set at 614.4kHz, and post-divider will be set to 64 to get clock frequency of 9600.

Using BitCalc with Reference Frequency of 20MHz and a Desired frequency of 0.6144MHz

Hex Word (Unstuffed) = 0x38BB90

Post divide: 64 (0x40)

Programmable Clock Register Word = 0x4038BB90

D23:D0 = 0x38BB90

D31:D24 = 0x40

Using the above method, you should be able to achieve any input clock frequency between 20MHz and 1.57kHz (400kHz / 255). If frequencies slower than 1.57kHz are required, the USC internal baud rate generator may be used to further divide the clock. Remember that asynchronous protocols oversample at 16x / 32x /or 64x the input clock frequency, so async baud rates below 30Hz are possible without using the USC internal baud rate generators.

If different clock frequencies are required for each channel, the programmable oscillator should be set to a common multiple of the desired frequencies. Since there is only a single programmable oscillator, the different frequencies are determined by different post-divider values.