OPERATING MANUAL

MODEL 4248

32-Channel, 125 kHz, Simultaneous Sampling
A/D Converter MIX Module
for VMEbus Systems

PENTEK

Pentek, Inc.
One Park Way
Upper Saddle River, NJ 07458
(201) 818-5900

Manual Part No: 800.42480 Rev: E - October 9, 1995

Pentek Model 4248 Operating Manual - Revision History

Date	Rev	Applicable Serial #'s	Comments
05/10/91	A	9119005	Initial Product Release
09/19/91	B	9119006 - 9144013	Input Impedance changed to 200Ω
10/30/91	C	9144014 - 9212004	Standard Full Scale Input Spec changed to ± 5 V_{DC}. ± 10 V_{DC} now optional (Opt. 001).
05/01/92	D	9212004 - 9242002	Input Impedance changed from 200Ω to 10kΩ.
10/09/95	E	9424003 - Forward	Manual format updated. Add information about Option 003 (14-bit A/D). MIX_Wait no longer asserted on read of Empty FIFO. Error bits not reliable when clocked externally - add note. New sample code.

The material presented in Appendix B of this manual is excerpted from Intel Corporation's Data Sheet for the 82C54 CHMOS Programmable Interval Timer. A complete copy of this Data Sheet may be found in Intel's Peripheral Components Data Book, which may be ordered from Intel Literature Sales, P. O. Box 7641, Mt. Prospect, IL 60056-7641 (Phone: (800) 548-4725). Pentek gratefully extends its thanks to Intel for their permission to use this copyrighted material.

WARRANTY

Pentek warrants that all products manufactured by Pentek conform to published Pentek specifications and are free from defects in materials and workmanship for a period of one year from the date of delivery when used under normal operating conditions and within the service conditions for which they were furnished.

The obligation of Pentek arising from a warranty claim shall be limited to repairing or at its option, replacing without charge, any product which in Pentek's sole opinion proves to be defective within the scope of the warranty.

Pentek must be notified in writing of the defect or nonconformity within the warranty period and the affected product returned to Pentek within thirty days after discovery of such defect or nonconformity.

Buyer shall prepay shipping charges, taxes, duties and insurance for products returned to Pentek for warranty service. Pentek shall pay for the return of products to buyer except for products returned from another country.

Pentek shall have no responsibility for any defect or damage caused by improper installation, unauthorized modification, misuse, neglect, inadequate maintenance, accident or for any product which has been repaired or altered by anyone other than Pentek or its authorized representatives.

The warranty described above is buyer's sole and exclusive remedy and no other warranty, whether written or oral, is expressed or implied. Pentek specifically disclaims fitness for a particular purpose. Under no circumstances shall Pentek be liable for any direct, indirect, special, incidental or consequential damages, expenses, losses or delays (including loss of profits) based on contract, tort, or any other legal theory.

Table of Contents

Table of Contents

Table of Contents

Page

Appendix B: Intel 82C54 Data Sheet (continued)

Appendix C: MIX Tutorial

Table of Contents

Chapter 1: General Description

1.1 Introduction

The Model 4248 is a data acquisition expansion module for the Pentek series of VME/MIX Baseboards. It provides up to 32 channels of A/D conversion to 12-bit accuracy (optionally 14 bits) with a total maximum conversion time of less than 10 µsec. The inputs are connected to 32 sample-and-hold amplifiers for simultaneous sampling, making this device the ideal choice for multi-channel applications in which phase relationships between channels must be maintained. The sampling rate is software programmable to 125 kHz with all channels active. The number of active channels is also programmable, in steps of one, and the sampling rate can be increased as the number of active channels is decreased.

Up to three Model 4248 modules may be stacked on a single Pentek MIX Baseboard, using the MIX stacking connector system. This architecture allows the Baseboard to communicate with the Model 4248 through a well-defined, high-bandwidth 32-bit data and address bus.

1.2 Simultaneous Sampling

Each of the 32 input signals is sampled simultaneously by its own Sample-and-Hold (S/H) amplifier, thus preserving the phase relationships between them. A 32-input analog multiplexer scans the S/H outputs for sequential Analog-to-Digital (A/D) conversion. Any combination of up to 32 active channels can be selected by loading a channel RAM. The A/D achieves 12-bit (or 14-bit) accuracy and can convert all 32 channels in less than 10 µsec, supporting a 125 kHz sampling rate.

A FIFO buffers the data between the A/D converter and the MIX interface. Interrupt flags for empty, full and half-full FIFO conditions permit efficient block transfers over the MIX bus with DMA operation fully supported.

1.3 Sampling Control

An on-board programmable counter divides a 10 MHz clock in integer steps to generate the sample clock. Each clock pulse triggers the S/H amplifiers to "hold" all inputs, and then starts the sequential conversion of the selected channels. Alternatively, a front panel BNC connector accepts an external TTL signal that may be used as, or be divided to derive, the sample clock.

The maximum sampling frequency ($f_{s_{max}}$), as a function of the number of channels to be converted (N), is given by the following equation:

$$f_{s_{max}} = \frac{10\ \text{MHz}}{2N + 15}$$

1.4 Block Diagram and Circuit Description

The block diagram for the Pentek's Model 4248 32-channel, 12-bit, 125 kHz, simultaneous sampling A/D converter MIX Module for VMEbus Systems, is shown in Figure 1-1, below.

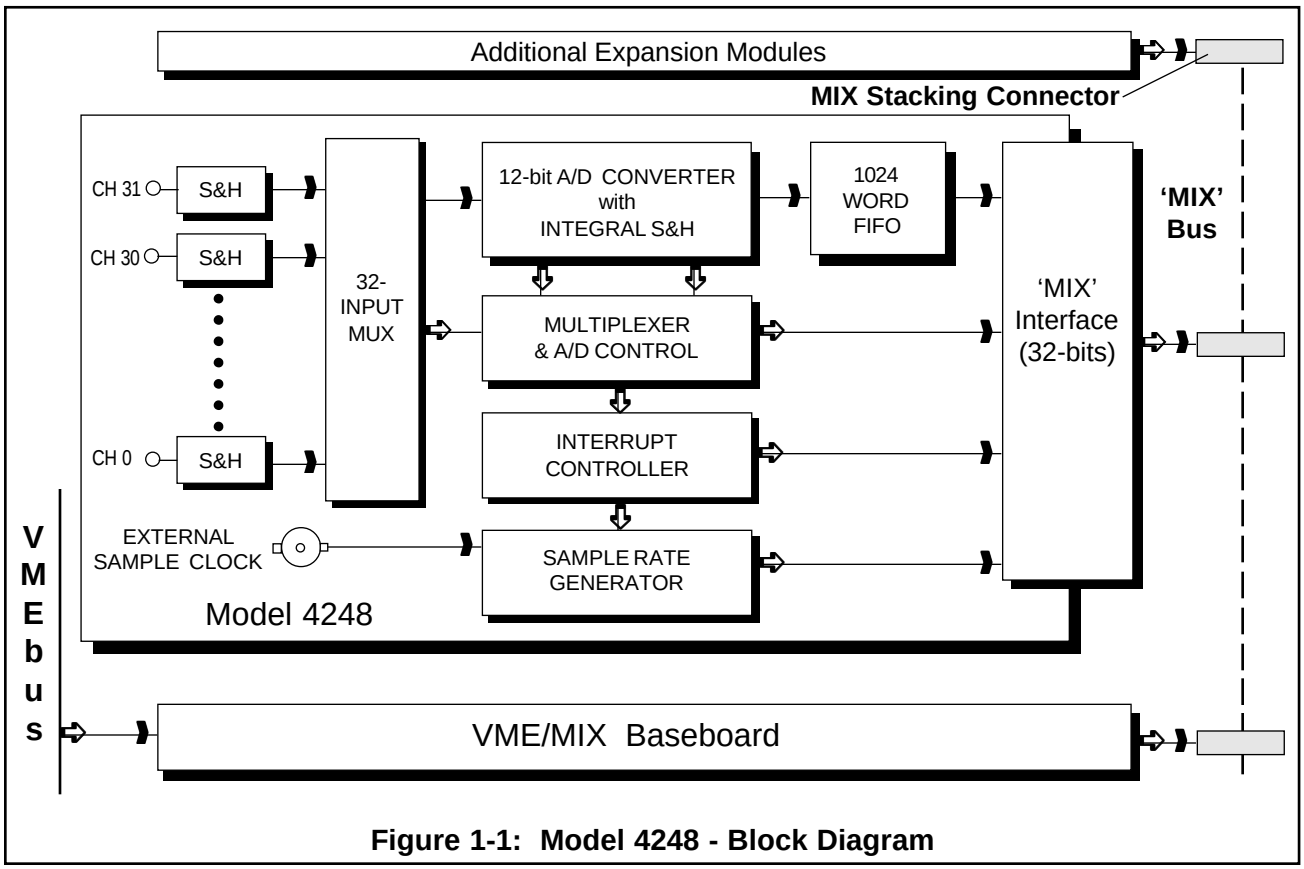


Figure 1-1: Model 4248 - Block Diagram

Each of the 32 inputs is connected to a sample-and-hold amplifier to permit simultaneous sampling. The outputs of the **S&H** amplifiers are scanned by the 32-input multiplexer (**MUX**) for sequential conversion by the 12-bit **A/D** converter. The digital samples from the A/D are received by the **FIFO** memory. The FIFO is unloaded by memory transfers over the MIX bus orchestrated by the MIX Baseboard.

The **Multiplexer Control** section allows the user to select which of the 32 channels are active in the conversion scan. The **Sample Rate Generator** contains a programmable counter circuit to set the sampling clock. An external TTL sample clock input is accepted from a front panel BNC connector.

The **Interrupt Control** section supports interrupt signaling of the Baseboard via the MIX bus in response to various FIFO conditions. Control and Status registers are accessible over the MIX bus as memory locations.

1.5 Specifications

Sample and Hold Amplifiers:

Number:	32
Inputs (each chan):	Single-ended
Input Impedance:	10kΩ standard (other impedances available, contact factory)
Full Scale Voltage:	
Standard:	±5V
Option 001:	±10V

Multiplexer & Multiplexer Control:

Active Channels:	1 to 32, programmable
Channel Selection:	Any combination, programmable
Scanning Sequence:	Any order, programmable

A/D Converter:

Standard:	
A/D Type:	Burr Brown ADC603
Resolution:	12 bits
Rate:	>126 kHz per channel for 32 channels, to >588 kHz for 1 channel
Option 003:	
A/D Type:	Burr Brown ADC614
Resolution:	14 bits
Rate:	>126 kHz per channel for 32 channels, to >588 kHz for 1 channel

Overall Distortion:	Standard	Option 003	Units
Harmonic Components: (for 10 kHz input sampled at 100 kHz)	-65	-70	dB below full scale
Spurious Components: (for 1 kHz input sampled at 100 kHz)	-60	-60	dB below full scale
Crosstalk: (for 1 kHz input sampled at 100 kHz)	-60	-60	dB below full scale

FIFO Memory:

Standard:	1k x 16 bits
Option 002:	16k x 16 bits

Sample Rate Generator:

Internal Sample Clock:	Programmable divide-by-N from 10 MHz, where $79 \leq N \leq 65{,}535$
External Sample Clock:	TTL Input, BNC

A/D Output Coding:

Format: 16-bit, 2's complement, left justified

Coding Examples:	Standard (12-bit)	Option 003 (14-bit)
+Full Scale:	0x7FF0	0x7FFC
+1 LSB:	0x0010	0x0004
Zero:	0x0000	0x0000
– 1 LSB:	0xFFF0	0xFFFC
– Full Scale:	0x8000	0x8000

1.5 Specifications (continued)

MIX Interface:

Memory Mapping:	Determined by Stack Position
FIFO Data:	16-bits (see A/D Coding, above)
Registers:	Status, Control, & Interrupt Mask
Interrupts:	FIFO Not Empty, Half-full, & Full Overload, Channel Sync, FIFO Write Counter, Error
Channel RAM:	Up to 128 entries
Sample Rate Generator:	16-bit divisor from 10 MHz or External Clock Input
Power:	1.8 A @ + 5 V_{DC} max 0.5 A @ +12 V_{DC} max 0.75 A @ – 12 V_{DC} max
Dimensions:	10.3" x 4.0" x 0.80"

Chapter 2: Installation and Connections

2.1 Inspection

After unpacking the unit, inspect it carefully for possible damage to connectors or components. If any damage is discovered, please contact Pentek immediately, at the phone number shown on this manual's title page. Please save the original shipping container and packing material, in case re-shipment is required.

2.2 Circuit Configurations and Jumper Settings

Before installing the Model 4248, the unit must be configured for the desired operating parameters. This is accomplished by setting the jumpers described in the sections that follow.

The Model 4248 consists of two PC assemblies: the MIX Board and the Mezzanine Board. (See Appendix C of this manual for a description of the MIX interface elements and glossary of terms.) These boards are joined together mechanically and electrically but they do **not** need to be separated in order to configure the jumpers.

For reference purposes, the component layout drawings for both the MIX and Mezzanine boards are included in this manual. Figure 2.2, on the next page, shows the MIX board, and Figure 2-3, on the page after next, shows the Mezzanine board.

2.2.1 Sample Clock Jumpers - JB1 and JB2, MIX Board

The sampling clock signal can be derived from an internal counter circuit or from an external TTL clock input. A simplified schematic of the sample clock circuitry is presented in Figure 2-1, below.

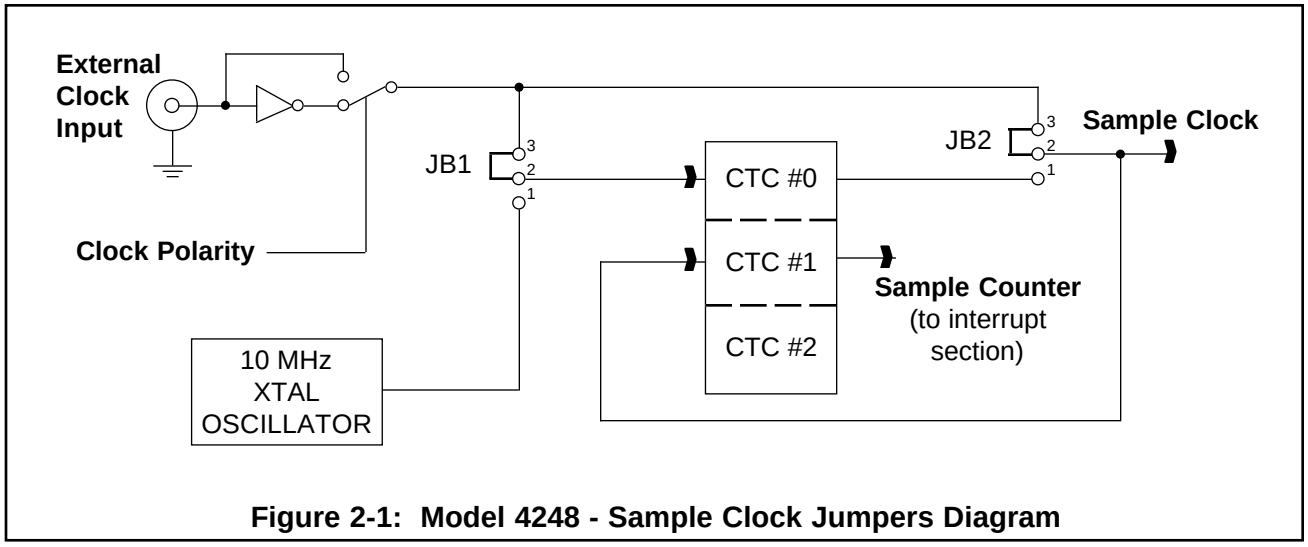


Figure 2-1: Model 4248 - Sample Clock Jumpers Diagram

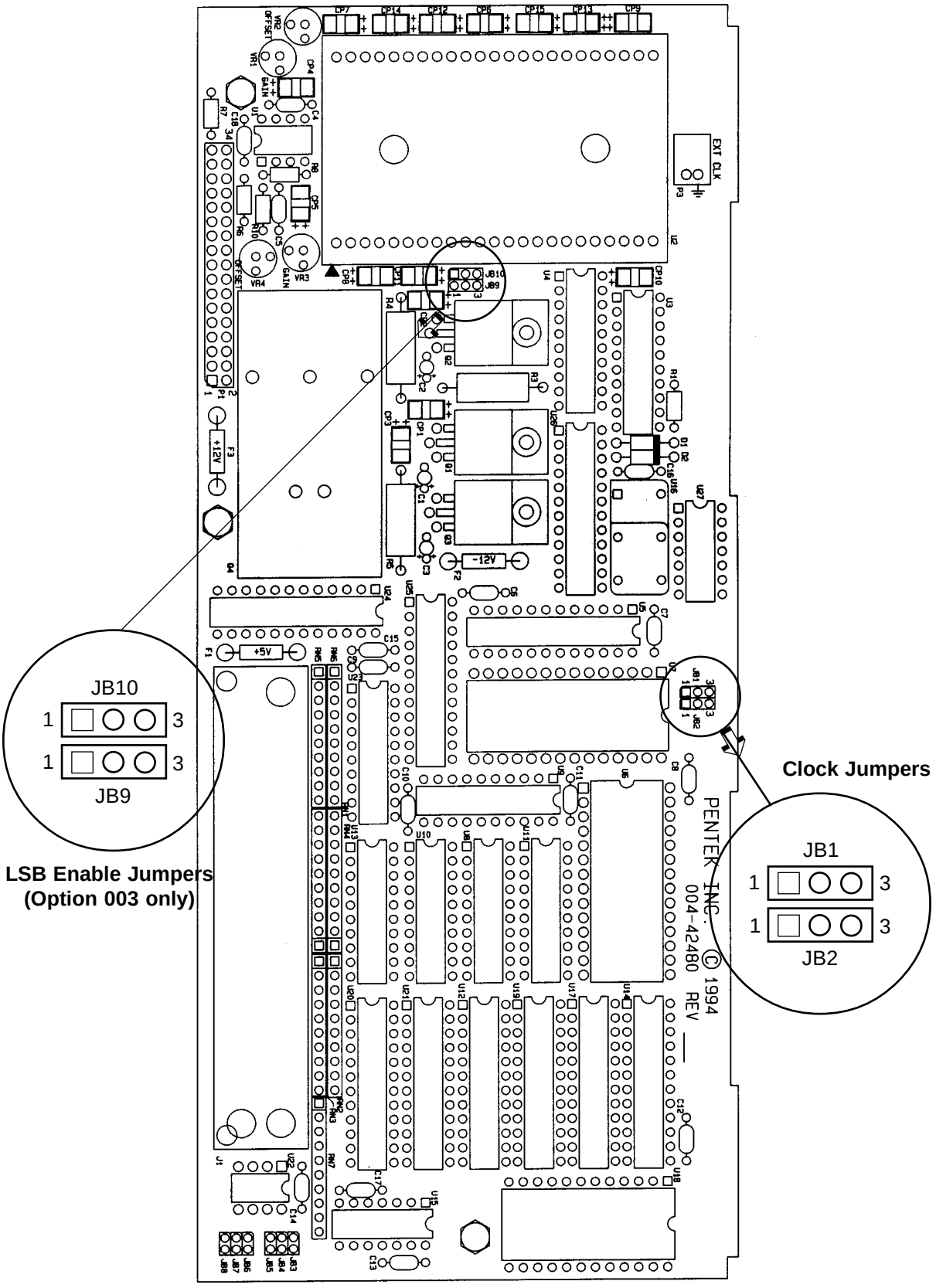


Figure 2-2: Model 4248 - MIX Board

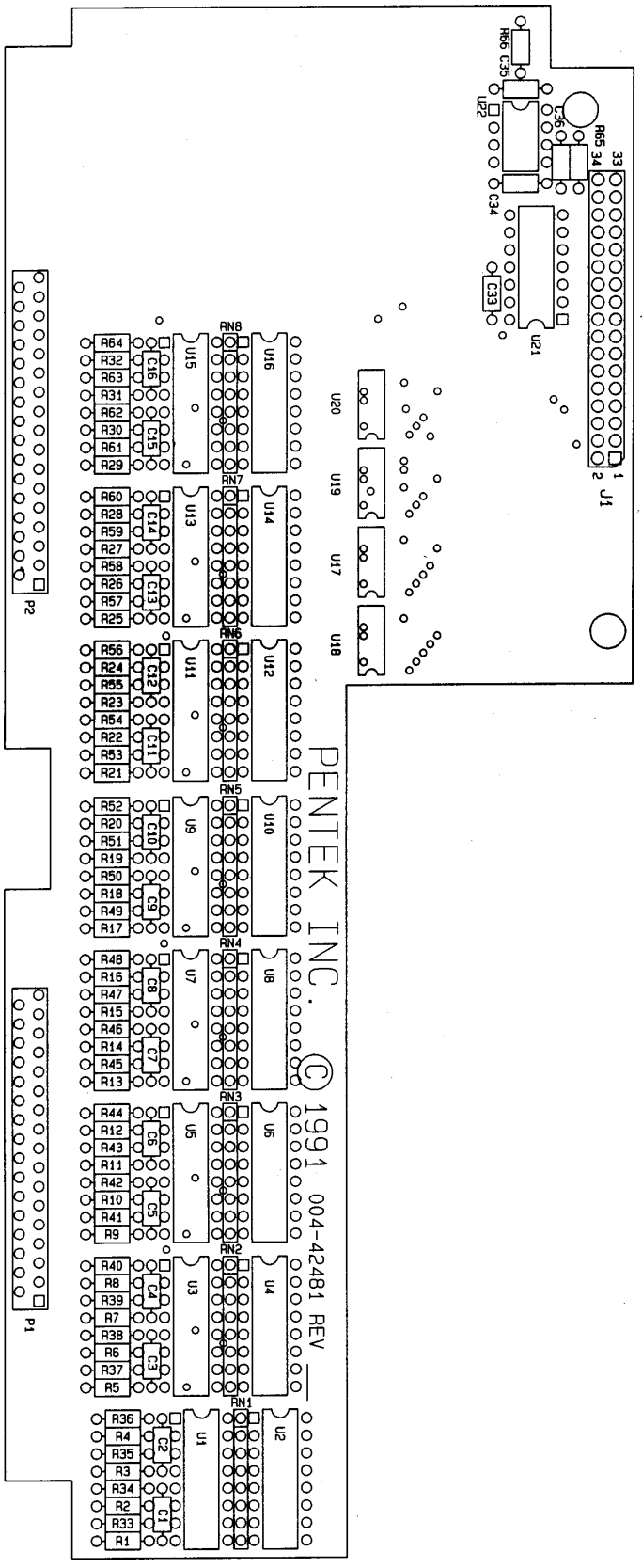


Figure 2-3: Model 4248 - Mezzanine Board

2.2 Circuit Configurations and Jumper Settings (continued)

2.2.1 Sample Clock Jumpers (continued)

Jumper block JB1 selects the input clock to Counter Timer Circuit, channel 0 (CTC #0). This clock signal can be either the external TTL clock signal from the front panel BNC, or the output of the internal, 10 MHz crystal oscillator.

Jumper block JB2 selects the source of the Sample Clock signal. This may be either the external TTL clock signal from the front panel BNC, or the output of CTC #0.

2.2.2 LSB Enable Jumpers - JB9 and JB10, MIX Board (Option 003 only)

The Model 4248, Option 003, features an A/D Converter with 14-bit resolution, as opposed to the 12-bit resolution of the standard unit. In units equipped with this option, the two additional bits are connected to the buffer between the ADC and the A/D FIFO via jumper blocks JB9 (for D13) and JB10 (for D14) (Note that, per the convention used by Burr-Brown for the 12-bit AD603 and the 14-bit AD614, D1 is the MSB, and there is no D0). These jumpers are located under the Mezzanine board, and they are factory-installed if Option 003 is included in a given unit. Thus, they should not need to be accessed by the user.

2.2.3 Other Jumper Blocks, Jumper Block Locations and Jumper Size

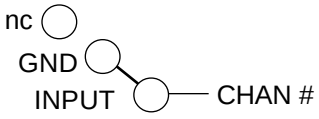
All jumper blocks on the Model 4248 not mentioned above are for factory configuration and testing purposes **ONLY**. Aside from JB1 and JB2 (and JB9 and JB10 in units equipped with Option 003), which are discussed in the section(s) above, **there are no other user-configurable jumpers on the Model 4248.**

The locations of jumper blocks JB1, JB2, JB9 and JB10 are shown in Figure 2-2, on the page before the previous. These jumper blocks accept shorting jumpers with 2 mm spacing. Pentek's part number for these jumpers is 356.00010.

2.3 Front Panel Connectors

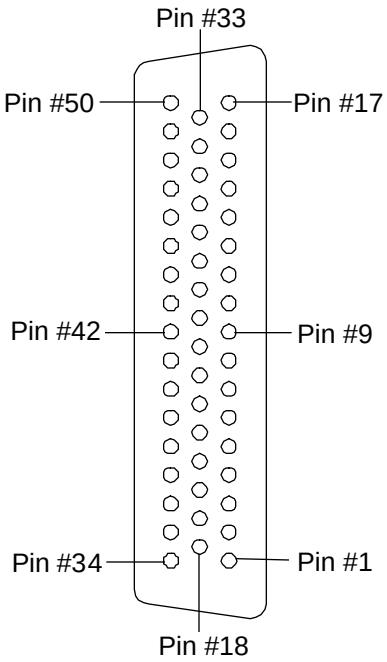
A pair of 50-pin "D"- type connectors is provided on the Model 4248's front panel, each of which can accept up to 16 analog input signals. Each input signal is single-ended and has two pins assigned to it, signal and ground. Also provided on the front panel is a BNC connector to accept an external clock signal. These connectors, and the characteristics of the signals to be applied there, will be discussed in the subsections below. Figure 2-4, on the next page, shows the Model 4248's front panel and gives the pinouts of the input signal connectors.

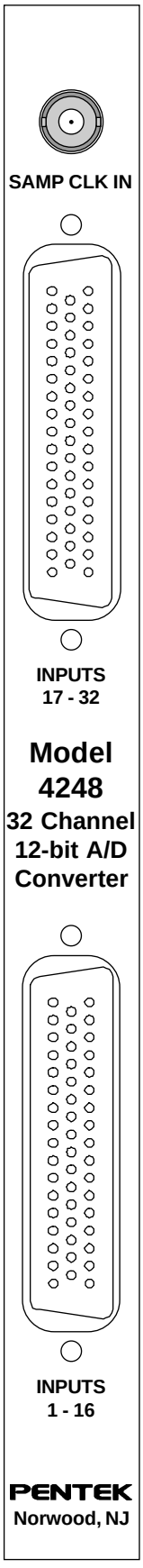
Key to Input Pin Connections
(typical each channel)



Connector Pin Numbering
Front Panel Female DD-50F
(front panel view)

Mating Male Plug DD-50M
(rear view - wiring side)





Connector Pin Chart for Chan 17 to 32

Input Channel	IN Pin#	GND Pin#
32	16	33
31	15	32
30	14	31
29	13	30
28	12	29
27	11	28
26	10	27
25	9	26
24	8	25
23	7	24
22	6	23
21	5	22
20	4	21
19	3	20
18	2	19
17	1	18

Pin #17 is GND
Pins #34 thru #50 are not connected

Connector Pin Chart for Chan 1 to 16

Input Channel	IN Pin#	GND Pin#
16	16	33
15	15	32
14	14	31
13	13	30
12	12	29
11	11	28
10	10	27
9	9	26
8	8	25
7	7	24
6	6	23
5	5	22
4	4	21
3	3	20
2	2	19
1	1	18

Pin #17 is GND
Pins #34 thru #50 are not connected

Figure 2-4: Model 4248 - Front Panel and Connector Pin Numbering

2.3 Front Panel Connectors (continued)

2.3.1 Analog Input Connectors

The input connector style is the standard D-Subminiature series DD-50. The connectors mounted on the front panel have female pins, and are manufactured by Positronics Corp. (their model number is MD50F50000, Pentek's part number is 351.05004). The mating connectors have male pins, and are available both in solder-cup versions for attaching discrete wiring (such as twisted pair or coaxial cable) and flat ribbon cable versions. Two hooded, mating connectors with solder-cup contacts are provided with each Model 4248. These connectors are also manufactured by Positronics. The model number for the connector is MD50M20000 (Pentek part number 351.05005), and the hood is model number D50000Y00 (Pentek part number 351.05006). The pin numbering of the connectors is shown on the next page, in Figure 2-4.

2.3.2 Analog Input Signals

Each analog input signal is connected to the Sample & Hold Amplifier through a resistive divider circuit as shown in Figure 2-5, below. The input impedance of the divider is 10 kΩ.

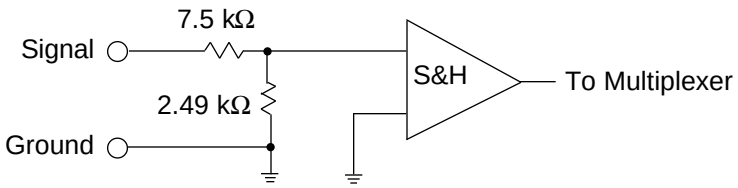


Figure 2-5: Model 4248 - Typical Analog Input Circuit (1 of 32)

The input signals are bipolar (centered around zero volts DC) and have full scale input ranges of ±5 V (10 V_{P-P}) or ±10 V (20 V_{P-P}), depending upon the option ordered.

2.3.3 External Sample Clock Input

The External Sample Clock input is connected to the front panel BNC connector. The input is a TTL logic signal with logic '0' = 0.0 to +0.8 V and logic '1' = +2.0 V to +5.0 V. The input will also accommodate 5V CMOS logic signals.

2.4 Installing the Model 4248 on the MIX Baseboard

See Appendix C of this manual for MIX module stack assembly instructions.

2.5 Installing the Assembly into the VMEbus Card Cage

Appendix C of this manual also covers insertion of the assembled module stack in the card cage.

This page is intentionally blank

Chapter 3: Memory Maps and Register Descriptions

3.1 Programming Overview

The following sections cover access to the Model 4248 from the MIX Baseboard and the issuing of commands for module control and data transfers. There are two cases to be considered, dependent upon the type of Baseboard used. This chapter gives details for Baseboards that use direct MIX addressing, such as Pentek's Models 4200, 4200A, 4201, 4201A, 4202 and 4284. General information on the use of these Baseboards is given in Appendix D. Appendix E gives general information on access to MIX modules from Baseboards that use an Address Page Register, such as Pentek's Model 4283. The Memory Map for the Model 4283 is also included in this chapter.

3.2 Address Maps

The sections below will present address maps for the resources available on the Model 4248, for each type of MIX Baseboard available from Pentek. Each of the six resources will be assigned a symbolic name, by which that resource will be referred to in subsequent sections of the manual. There are four cases to consider, each of which is covered in a separate section.

If the Baseboard you will be using is a Model 4200, 4200A, 4201 or 4201A, use the address map presented in Section 3.2.1. If you will be using your Model 4248 with a Model 4202 Baseboard, use the address map in section 3.2.2. If the Model 4283 will be the MIX Baseboard for your Model 4248, skip to Section 3.2.3, and use the address map found there. Finally, if your 4248 will be mounted on a Model 4284 Baseboard, use the address map in Section 3.2.4.

By convention, the MIX addresses given in this manual are generally given as 8-digit hexadecimal codes, with a space between the first and second four digits, simply for notational convenience. The form is MIX_base+0xYYYY ZZZZ. The leading '0x' is common notation for a hex value in the 'C' programming language.

Since each hex digit represents 4 bits, the 8-digit MIX address represents 32 bits of address. The MIX_base part of the address is dependent upon the Baseboard in use, and is used to select each module by its stack position. This geographical addressing scheme avoids the need for address jumpers and the conflicts they may cause, but requires that software must accommodate the physical placement of modules in the MIX stack.

The exception to this convention will be for the Model 4283 Baseboard, whose TMS320C30 processor only has a 24-bit address bus on the port it uses for the MIX bus. In this case, a page register is used for module selection (as opposed to different base addresses for each module), and the addresses are given as 6-digit hexadecimal codes, of the form 0xYY ZZZZ.

A final point must now be made about MIX nomenclature. Module 0 (or MIX position 0) refers to the closest module to the Baseboard, whether or not it is nested. Module 1 (or MIX position 1) is the module that may be mounted above Module 0, and Module 2 is the device that may be stacked above Module 1.

3.2 Address Map (continued)

3.2.1 Model 4248 Address Map for the 4200/01 Series Baseboards

Pentek's Models 4200 and 4201 Baseboards use a 16 MHz 68030 processor as the MIX bus controller. The 4200A and 4201A use a 40 MHz 68030. From the 68030's viewpoint, the MIX bus is mapped identically for all four of these Baseboards. The MIX_base address for Module 0 (which may either be nested, sharing the Baseboard's VMEbus slot, or be stacked in the adjacent VMEbus slot), is 0xC000 0000. For the next device in the stack, Module 1, the base address is 0xD000 0000. Finally for the last MIX Module in the stack, Module 2, the MIX_base address is 0xE000 0000. Table 3-1, below, gives the physical addresses, in the memory space of the 68030 processor on any of these four Baseboards, that correspond to the symbolic names used to represent the Model 4248's resources in subsequent sections of this manual.

Table 3-1: Model 4248 Memory Map for Use with Models 4200, 4200A, 4201 and 4201A MIX Baseboards- 68030 View

4200/01 68030 Address	Symbolic Name	Resource
MIX_base+0x0000 0000	Control_Reg	Control Register
MIX_base+0x0000 0004	Intr_Mask_Reg	Interrupt Mask Register
MIX_base+0x0000 0008	CTC_Reg	Counter Timer Circuit Register
MIX_base+0x0000 000C	Chan_RAM	Channel RAM
MIX_base+0x0800 0008	Status_Reg	Status Register (Read Only)
MIX_base+0x0800 000C	InData_Reg	A/D FIFO (Read Only)

MIX_base Address:

If 4248 is in MIX position 0, MIX_base = 0xC000 0000

If 4248 is in MIX position 1, MIX_base = 0xD000 0000

If 4248 is in MIX position 2, MIX_base = 0xE000 0000

Note that the 4200 series of VME/MIX Baseboards also support mastering of the MIX bus by Upper MIX Bus Masters (UMBMs), and by VMEbus Masters. For further information about MIX Bus access on these Baseboards in this manner, please refer to Appendix D of this manual (the MIX Baseboard Tutorial), or to the Baseboard's or UMBM's Operating Manual.

3.2 Address Maps (continued)

3.2.2 Model 4248 Address Map for the Model 4202 Baseboard

The Model 4202 is a VMEbus slave-only Baseboard, and contains no processor. This device maps the resources connected to its MIX Bus into VMEbus A32, A24 or A16 slave address space. The VMEbus A32 or A24 base address is programmed into a register in the 4202's A16 interface by a VMEbus Master (the A16 base address is set by switches on the board). This register is cleared at power-up, resulting in a default base address of 0x0000 0000 in A32 space, or 0x00 0000 in A24 space.

In direct addressing mode, the two most significant bits of the VMEbus address (i. e., A31 and A30 in A32 space, A23 and A22 in A24 space or A15 and A14 in A16 space) select a particular 4202 Baseboard, by comparison with the two Least Significant Bits (D1 and D0) of the VMEbus Address Register. The next two VMEbus address bits (i. e., A29 and A28 in A32 space, A21 and A20 in A24 space, or A13 and A12 in A16 space) select among the MIX modules stacked on the 4202. Thus, assuming A32 operation and that the VMEbus Address Register is NOT written to, the MIX_base address for Module 0 is 0x0000 0000. Making the same assumptions, Module 1's MIX_base address is 0x1000 0000 and the MIX_base address for Module 2 is 0x2000 0000.

Table 3-2, below, presents the memory map for a Model 4248 installed on a Model 4202 Baseboard, assuming direct addressing, A32 or A24 operation, and that nothing has been written to the 4202's VME Address Register. For information about using the Model 4202's Paged Addressing Mode, A16 MIX access, multiple 4202's in the same card cage or Upper MIX Bus Masters, please refer to the Operating Manual for the Model 4202 Baseboard or the UMBM.

Table 3-2: Model 4248 - Memory Map for Use with Model 4202 MIX Baseboard

VMEbus Address	Symbolic Name	Resource
MIX_base+0x0000 0000	Control_Reg	Control Register
MIX_base+0x0000 0004	Intr_Mask_Reg	Interrupt Mask Register
MIX_base+0x0000 0008	CTC_Reg	Counter Timer Circuit Register
MIX_base+0x0000 000C	Chan_RAM	Channel RAM
*MIX_base+0x0800 0008	Status_Reg	Status Register (Read Only)
*MIX_base+0x0800 000C	InData_Reg	A/D FIFO (Read Only)

Default MIX_base Address:

If 4248 is in MIX position 0, MIX_base = 0x0000 0000 for A32, or 0x00 0000 for A24

If 4248 is in MIX position 1, MIX_base = 0x1000 0000 for A32, or 0x10 0000 for A24

If 4248 is in MIX position 2, MIX_base = 0x2000 0000 for A32, or 0x20 0000 for A24

* - in A24 Space, access at 0x08 000z

3.2 Address Maps (continued)

3.2.3 Model 4248 Address Map for the Model 4283 Baseboard

On the Model 4283 Baseboard, there are two levels of addressing: 1) selection of the particular expansion module by its stack position and 2) selection of one of the resources within the module.

3.2.3.1 Module Selection

The TMS320C30 selects among MIX Modules by using a two-bit field in the Model 4283 Page Register, located at 'C30 address 0x80 0002. These two bits select one of the three possible MIX stack positions according to Table 3-3, below.

Table 3-3: Model 4248 - Module Selection - Model 4283 Page Register - 'C30 Address 0x80 0002

Page Register Bits>	D31 - D18	D17	D16	D15	D14	D13 - D0	Example
Stack Position 0	x	MX_A27	1	1	MX_A21	x	0x01 8000
Stack Position 1	x		1	0		x	0x01 0000
Stack Position 2	x		0	1		x	0x00 8000
x = other functions							

Access to some resources on certain MIX modules may require that either or both of the MIX bus A27 (MX_A27) and A21 (MX_A21) bits be set to the logic '1' state. On the 4283 baseboard, these bits are also contained in the Address Page Register. MX_A27 is the D17 bit of this register, and MX_A21 is the D14 bit. Table 3-3 also shows these bits.

3.2.3.2 Register Selection Within the Module

Once the module has been selected, all resources (including the RAM Banks) of the Model 4248 appear in the TMS320C30 address map according to Table 3-4, at the top of the next page. Accessing these registers is accomplished by simply reading and writing these locations.

The 24-bit addressing scheme accesses 32-bit data locations for the TMS320C30, even though many of the elements correspond to 8-bit or 16-bit devices.

3.2 Address Maps (continued)

3.2.3 Model 4248 Address Map for the Model 4283 Baseboard (continued)

3.2.3.2 Register Selection Within the Module (continued)

Table 3-4: Model 4248 - Memory Map for Use with Model 4283 MIX Baseboard		
4283 'C30 Address	**Symbolic Name**	**Resource**
0x80 4000	Control_Reg	Control Register
0x80 4001	Intr_Mask_Reg	Interrupt Mask Register
0x80 4002	CTC_Reg	Counter Timer Circuit Register
0x80 4003	Chan_RAM	Channel RAM
†0x80 4002	Status_Reg	Status Register (Read Only)
†0x80 4003	InData_Reg	A/D FIFO (Read Only)
† - D17 in Page Register = 1		

3.2.4 Model 4248 Address Map for the Model 4284 Baseboard

The Model 4284 Baseboard uses a TMS320C40 Digital Signal Processor to master transactions with the MIX bus. The three MIX module locations are each mapped into separate, 64 M-Longword regions of the 'C40's memory space. Module 0's MIX_base address in the 'C40 memory map is 0x2000 0000. The MIX_base address for Module 1 is 0x2400 0000, and the 'C40's MIX_base address for Module 2 is 0x2800 0000. Table 3-5, at the bottom of this page, presents the memory map for the six resources on the Model 4248, as seen by the 'C40 processor on the Model 4284 Baseboard.

This Baseboard also supports mastering of the MIX bus by processor modules in the MIX stack, known as UMBM's (Upper MIX Bus Masters). For further details on this type of transaction, please refer to Appendix D of this manual (MIX Baseboard Tutorial), the Model 4284 Operating Manual, or the UMBM's Operating Manual.

Table 3-5: Model 4248 - Memory Map for Use with Model 4284 MIX Baseboard		
4284 'C40 Address	**Symbolic Name**	**Resource**
MIX_base+0x0000 0000	Control_Reg	Control Register
MIX_base+0x0000 0001	Intr_Mask_Reg	Interrupt Mask Register
MIX_base+0x0000 0002	CTC_Reg	Counter Timer Circuit Register
MIX_base+0x0000 0003	Chan_RAM	Channel RAM
MIX_base+0x0200 0002	Status_Reg	Status Register (Read Only)
MIX_base+0x0200 0003	InData_Reg	A/D FIFO (Read Only)
MIX_base Address: If 4248 is in MIX position 0, MIX_base = 0x2000 0000 If 4248 is in MIX position 1, MIX_base = 0x2400 0000 If 4248 is in MIX position 2, MIX_base = 0x2800 0000		

3.3 **Control Register** - Read/Write @ Address Control_Reg

The Control Register allows the user to control the operation of the A/D converter, channel select RAM, FIFO and counter timer circuit (CTC). Table 3-6, below, shows the function of each bit. All bits are cleared to the logic '0' state at power up.

Table 3-6: Model 4248 - Control Register - Read/Write @ Address Control_Reg

Bit #	D7	D6	D5	D4	D3	D2	D1	D0
Function	Clock Divider Gate	FIFO Write Cntr Gate	Channel Run/ Load	Chan RAM Address Reset	Not Used	FIFO Reset	CTC Channel Select	
							Contents	CTC#
Bit Definition	0 = Off 1 = On	0 = Off 1 = On	0 = Load 1 = Run	0 = Reset 1 = Release	—	0 = Reset 1 = Release	00 = Clk Divider 01 = FIFO Wr.Ct 10 = not used 11 = CTC Control	0 1 2 3

The Control Register is a read/write device that uses only the bottom 8 bits of the 32-bit word from the MIX bus. Reading and writing to the Control Register is accomplished by accessing the symbolic memory address Control_Reg, whose physical address can be found in Table 3-1, 3-2, 3-3 or 3-4, depending upon the Baseboard in use.

When writing to the Control Register, data in bit positions D8 through D31 will have no effect. When reading from address Control_Reg, bits D0 through D7 will reflect the contents of the Control Register, and bits D8 through D31 will be indeterminate and should be ignored.

Each of the bits in the Control Register will be described in the subsections below.

3.3.1 **Counter/Timer Channel Select** - Control_Reg, Bits D0 and D1

These two bits select which section of the Intel 82C54 Counter Timer Circuit (U18 on the MIX board) is selected for addressing. Programming of this device is discussed briefly in Section 3.7 of this manual, and in greater detail in Appendix B, which is the 82C54 data sheet. The bit correspondence to CTC channels is given in Table 3-7, below.

Table 3-7: Model 4248 - Counter Timer Channel Selection Bits

Control Register		CTC Section	Function
Bit 1	Bit 0		
0	0	Channel 0	Clock Divider
0	1	Channel 1	FIFO Write Counter
1	0	Channel 2	not used
1	1	Control	CTC Control & Mode of Operation

3.3 Control Register (continued)

3.3.2 FIFO Reset - Control_Reg, Bit D2

This bit is used to reset all functions of the FIFO memory. This is useful when initializing the module and in recovering from any error condition. When this bit is cleared to the logic '0' state, the FIFO is cleared of all data, the EMPTY flag is set true, and the HALF-FULL and FULL flags are set false. When this bit is set to the logic '1' state, the FIFO is released to accept data.

3.3.3 Channel RAM Address Reset - Control_Reg, Bit D4

When the Channel RAM Address Reset bit is cleared to the logic '0' state, the Channel RAM Address Counter is reset to address zero. Setting this bit to the logic '1' state releases the counter. The Channel RAM holds the list of the channel numbers that are to be scanned by the MUX. This reset function is required to guarantee a zero starting address at the beginning of a load cycle over the MIX bus. It is also useful in ensuring that when the unit begins data collection, the first cycle begins with the first channel.

More detailed information about the operation of the Channel Address RAM may be found in Section 3.6 of this manual.

3.3.4 Channel Run/Load - Control_Reg, Bit D5

The Channel Run/Load bit is used to select the operating mode for the Channel Address Counter. When this bit is set to the logic '0' state, the Channel RAM Address Counter's clock input is connected to the decoded $\overline{\text{Read}}$/Write strobe from the MIX bus. This signal corresponds to the loading of each byte into the Channel RAM (MXWR). This causes each transferred byte to increment the counter to the next address automatically.

When the Channel Run/Load bit is set to the logic '1' state, the Channel RAM Address Counter's clock input is connected to the FIFO's Write clock. Each FIFO Write increments the counter to the next RAM address. More detailed information about the operation of the Channel Address RAM may be found in Section 3.6 of this manual. Figure 3-1, at the top of the next page, shows a simplified schematic diagram of the signals involved with control of the Channel Address RAM.

3.3 Control Register (continued)

3.3.4 Channel Run/Load (continued)

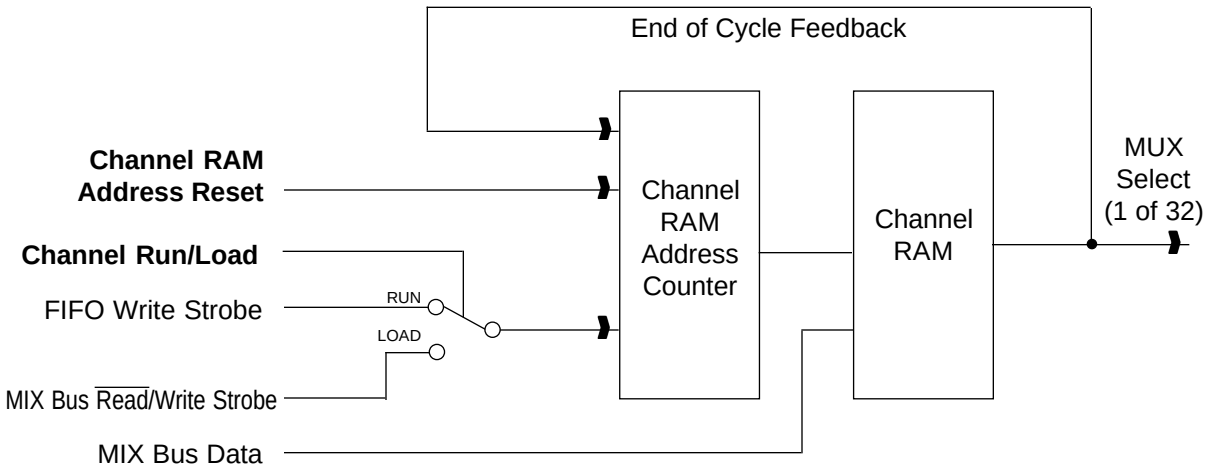


Figure 3-1: Model 4248 - Channel RAM Control Lines

3.3.5 FIFO Write Counter Gate - Control_Reg, Bit D6

The FIFO Write Gate bit serves as an enable signal for the clock input to Counter Timer Circuit #1. This counter is used to count each sample taken by the A/D converter and loaded into the FIFO. When this bit is set to the logic '1' state, the input clock is enabled and CTC channel 1 can count. When this bit is cleared to the logic '0' state, the input clock to CTC channel 1 is disabled. When the state of this bit is changed from logic '0' to logic '1', the FIFO Write Counter is reset to its programmed initial value by the next active clock edge, and will begin counting on the subsequent edge. This bit should be cleared before programming the CTC_Reg, to avoid clocking conflicts. This bit and its use in the CTC circuitry are shown schematically in Figure 3-2, at the top of the next page.

3.3.6 Clock Divider Gate - Control_Reg, Bit D7

The Clock Divider Gate bit serves as an enable signal for the clock input to Counter Timer Circuit #0. This counter is used to divide either the 10 MHz Crystal Oscillator or External Sample Clock signal for use as the sample clock. This bit should be cleared to the logic '0' state (to block the input clock) before programming the CTC_Reg, to avoid any conflicts that may arise between the clock signal and the program. When the state of this bit is changed from logic '0' to logic '1', the Clock Divider is reset to its programmed initial value by the next active clock edge, and will begin counting on the subsequent edge.

3.3 Control Register (continued)

3.3.6 Clock Divider Gate (continued)

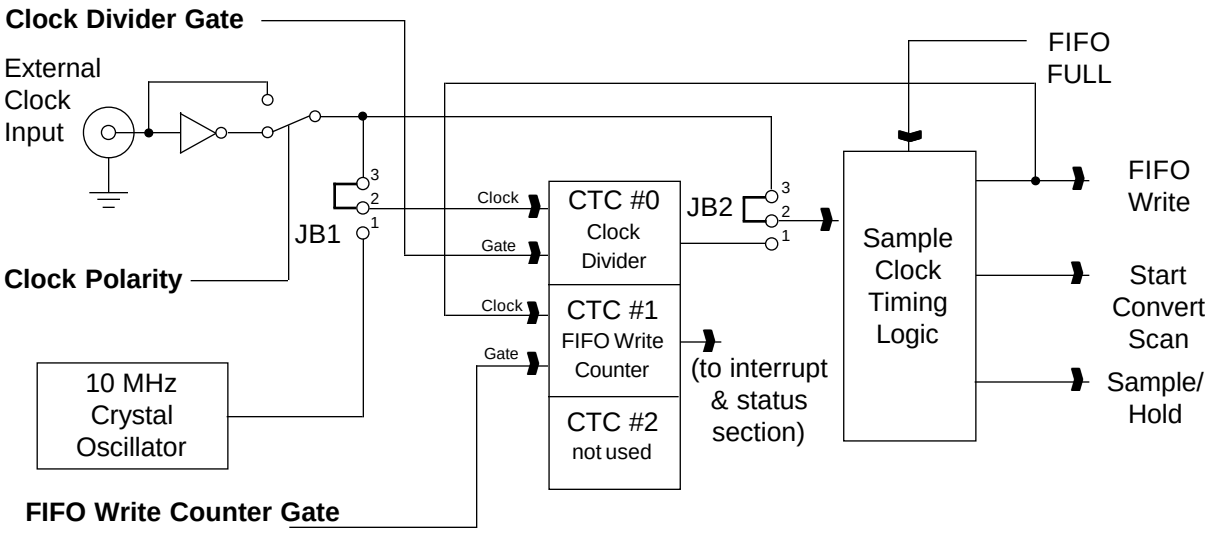


Figure 3-2: Model 4248 - Clock Control Lines

The Sample Clock Logic section accepts the clock selected by JB2 and generates timing signals for the sample and hold, A/D converter and FIFO write functions. When the FIFO is full, these timing signals are blocked to prevent data collection until the FIFO is able to accept more data. This is shown schematically in Figure 3-2, above.

3.4 Status Register - Read Only @ Symbolic Address Status_Reg

The Status Register reflects the current state of certain devices on Model 4248 and may be polled by the MIX Baseboard. This 8-bit register contains status bits as summarized in Table 3-8, below.

Table 3-8: Model 4248 - Status Register - Read Only @ Address Status_Reg

Bit #	D7	D6	D5	D4	D3	D2	D1	D0
Function	Channel Sync	FIFO Write Counter	Overload Detect	Error	Parity Error	FIFO Empty	FIFO Half Full	FIFO Full
Bit Definition	0 = Condition True 1 = Condition False							

3.4 Status Register (continued)

When this register is read, only the bottom 8 bits of the 32-bit word at symbolic address Status_Reg are meaningful. The 24 most significant bits should be ignored. Note that **all bits in the Status Register are low true.**

Once cleared to the logic '0' (true) state, all bits **except** the bottom three FIFO Status bits are latched until a Status Register read is performed. After a Status Register read cycle, all latched bits are set to the logic '1' (false) state. Reading the Status Register twice may be useful in recovering from error conditions or during program development. The first read clears any previously latched bits and the second read verifies the current status and the successful clearing of transient events.

The three FIFO Status bits (bits D0, D1 and D2) are **not** latched and always reflect current FIFO conditions. If the FIFO condition that caused an interrupt is still true when the Status Register is read, neither the status bit nor the interrupt will be cleared. We therefore recommend that the interrupting condition be dealt with **BEFORE** reading the Status Register

These status bits can be used to generate interrupts to the MIX bus. Reading the Status Register after an interrupt provides information regarding the cause of the interrupt. The Status Register may also be read at any time independent of interrupts. This can be used to implement a 'polled' control structure instead of an interrupt-driven scheme. The Status Register must be read at least once before enabling interrupts to clear any previous transient events. Detailed information on the interpretation of the status bits can be found in the subsections below.

3.4.1 Channel Sync - Status_Reg, Bit D7

The Channel Sync bit is cleared to the logic '0' (true) state when the last channel of a scan sequence (as determined by the contents of the Channel RAM) has been written to the FIFO.

3.4.2 FIFO Write Counter - Status_Reg, Bit D6

The FIFO Write Counter is implemented in Counter Timer Circuit #1. When the number of digital samples written to the FIFO is equal to the value programmed into the FIFO Write Counter, this bit is cleared to the logic '0' (true) state. Note that the FIFO Write Counter will be advanced N times per channel scan sequence, where N is the number of active channels.

3.4.3 Overload Detect - Status_Reg, Bit D5

The Overload Detect bit provides an indication that a digital sample out of the A/D converter was either positive full scale (0x7FF0) or negative full scale (0x 8000). This can be useful in guarding against acquiring data that is out of range. This bit will be latched until a Status Register read operation is performed.

3.4 Status Register (continued)

3.4.4 Error - Status_Reg, Bit D4

The Error bit is cleared to the logic '0' (true) state if either the Parity Error bit is cleared (D3 in this register), or if there is skewing in the FIFO flags. The Parity Error bit is discussed in Section 3.4.5, below. Flag skewing will be discussed in the next paragraph.

The Model 4248's 16-bit A/D FIFO actually consists of two 8-bit devices operating in parallel. Each FIFO has its own set of three FIFO status flags: full, half-full, and empty. Normally, if the FIFOs are reset in the initialization procedure, the two flag sets should be identical. This happens because data is always written into and read from the two FIFOs using the same write and read control lines. If these flags should become different for any reason, there is justification for discarding subsequent data and for re-initializing the unit.

The Error bit should never be true ('0') during FIFO write operations unless a real problem exists. The Error bit may occasionally be set during FIFO read operations due to asynchronous operation of the MIX bus relative to the 10 MHz clock on the Model 4248. To avoid this false error indication, read the Status Register after collecting data, to clear the Error flag.

3.4.5 Parity Error - Status_Reg, Bit D3

The Channel RAM integrity can be verified by use of the Parity Error bit. Each entry in the Channel RAM should be made with correct (even) parity. As each FIFO write is made, the parity for that Channel RAM byte is tested and, if in error, the Parity Error status bit is cleared to the logic '0' (true) state.

Although nothing should disturb the contents of the Channel RAM during normal operation, this feature permits an added level of error checking.

NOTE: The Error and Parity Error bits may report false errors when the Model 4248 is clocked externally.

3.4.6 FIFO Empty - Status_Reg, Bit D2

The presence of a logic '1' in the FIFO Empty bit indicates that at least one sample is in the FIFO. This bit can be used to signal the beginning of a scan sequence following a FIFO empty condition. It can also be used to unload each sample as it enters the FIFO.

3.4 Status Register (continued)

3.4.7 FIFO Half Full - Status_Reg, Bit D1

The presence of a logic '0' in the FIFO Half Full bit indicates that there are at least 513 samples in the FIFO of a standard Model 4248, or at least 8,193 samples in a unit equipped with Option 002. (Note that the status infoemation becomes availablewhen sample #(FIFO_size/2)+1 is written to the FIFO.) This bit can be used to initiate a block read sequence of up to half the FIFO depth. At least 513 (8,193 for Option 002) points of data are guaranteed to be available for transfer without having to test flags. Also, during the time it takes to unload the FIFO, at least 513 (8,193 for Option 002) more samples can be accepted by the FIFO without losing data.

3.4.8 FIFO Full - Status_Reg, Bit D0

The presence of a logic '0' in the FIFO Full bit indicates that there are 1024 samples in the FIFO of a standard Model 4248, or 16,384 samples in a unit equipped with Option 002. This bit can be used to initiate a block read sequence for the entire FIFO depth without having to test flags. The danger of using this flag is that once the FIFO is full, no more data can be written into the FIFO and some data may be lost. Another use of this flag is for indicating a FIFO overflow condition in applications where the FIFO should never reach the full state.

3.5 Interrupt Mask Register - Read/Write @ Symbolic Address Intr_Mask_Reg

The Interrupt Mask Register is used to enable or disable the possible sources of MIX Bus interrupts from the Model 4248. These interrupts may be directed to the processor on the MIX Baseboard or to a processor on an Upper MIX Bus Master (UMBM) module. This register also contains one bit that is **not** involved in the interrupt mask function. That bit is used to set the polarity of the external sampling clock.

When this register is accessed, only the bottom 8 bits of the 32-bit word at the symbolic MIX address Intr_Mask_Reg are meaningful. The 24 most significant bits should be ignored. Table 3-9, below, gives the bit arrangement of the Model 4248's Interrupt Mask Register. The subsections beginning on the next page provide operational details about this register.

Table 3-9: Model 4248 - Interrupt Mask Register - Read/Write @ Address Int_Mask_Reg

Bit #	D7	D6	D5	D4	D3	D2	D1	D0
Function	External Clock Polarity	Overload Detect	Channel Sync	FIFO Write Counter	Error	FIFO Not Empty	FIFO Half Full	FIFO Full
Definition	0 = Pos 1 = Neg	0 = Interrupt Masked 1 = Interrupt Enabled						

3.5 Interrupt Mask Register (continued)

3.5.1 External Clock Polarity - Intr_Mask_Reg, Bit D7

Bit D7 is the only bit in the 4248's Interrupt Mask Register is NOT associated with interrupt functions. This is the External Clock Polarity bit, which selects the active edge of the External Clock signal. When this bit is cleared to the logic '0' state, the low-to-high transition (positive edge) of the External Clock is active. When this bit is set to the logic '1' state, the high-to-low transition (negative edge) is active.

Care should be taken when changing the state of this bit during data collection. If the External Clock signal is at logic '0', then changing the state of this bit from '0' to '1', will cause an active clock edge to be generated. Similarly, when the External Clock signal is at logic '1', then changing bit D7 from '1' to '0' will cause the generation of one active clock edge. This arrangement can be used to achieve a software-generated sample clock if desired. With the External Clock Input connected to ground, simply toggle bit D7 from '1' to '0' and then back to '1' for each required software clock pulse.

3.5.2 Interrupt Mask Bits - Intr_Mask_Reg, bits D6 thruD0

Bits D0 through D6 in the Interrupt Mask Register each have corresponding bits in the Status Register. If an Interrupt Mask Register bit is set to the logic '1' state, then as soon as the corresponding Status Register bit goes true, a MIX bus interrupt will be generated.

Any number of interrupt mask bits can be enabled. In servicing an interrupt, the Status Register is typically read to determine the cause of the interrupt. Reading the Status Register also clears all status bits, thereby clearing the interrupt. Note that the Status Register read operation should be performed **AFTER** the interrupting condition has been cleared. The subsections below describe the conditions that these bits can either allow (enable) or prevent (mask) the generation of an interrupt in response to.

3.5.2.1 Overload Detect - Intr_Mask_Reg, Bit D6

When the Overload Detect bit in the Interrupt Mask Register is set to the logic '1' state, a MIX interrupt will be generated whenever the Overload Detect bit in the Status Register (D5) goes true (i. e., to the logic '0' state). The Status Register bit provides an indication that a digital sample out of the A/D converter was a full-scale value, either positive or negative. To mask interrupts in response to an overload condition, clear this bit to the logic '0' state.

3.5 Interrupt Mask Register (continued)

3.5.2 Interrupt Mask Bits (continued)

3.5.2.2 Channel Sync - Intr_Mask_Reg, Bit D5

When the Channel Sync bit in the Interrupt Mask Register is set to the logic '1' state, a MIX interrupt will be generated whenever the Channel Sync bit in the Status Register (D7) goes true (i. e., to the logic '0' state). The Status Register bit indicates that the last channel of a scan sequence has been written to the FIFO. To mask interrupts in response to the Sync indication, clear this bit to the logic '0' state.

3.5.2.3 FIFO Write Counter - Intr_Mask_Reg, Bit D4

When the FIFO Write Counter bit in the Interrupt Mask Register is set to the logic '1' state, a MIX interrupt will be generated whenever the FIFO Write Counter bit in the Status Register (D6) goes true (i. e., to the logic '0' state). The Status Register bit goes true when the number of digital samples written to the FIFO is equal to the value programmed into the FIFO Write Counter (CTC #1). To mask interrupts in response to the Sync indication, clear this bit to the logic '0' state.

3.5.2.4 Error - Intr_Mask_Reg, Bit D3

When the Error bit in the Interrupt Mask Register is set to the logic '1' state, a MIX interrupt will be generated whenever the Error bit in the Status Register (D4) goes true (i. e., to the logic '0' state). The Status Register bit goes true in the event of either a Parity Error (in which case, D3 of the Status Register will also be a '0') or a Flag Skew error (see Sections 3.4.4 and 3.4.5 for details about these error conditions). To mask interrupts in response to error conditions, clear this bit to the logic '0' state.

3.5.2.5 FIFO Flags - Intr_Mask_Reg, Bits D2 through D0

Set one of these bits to the logic '1' state to allow MIX interrupts to be generated in response to the FIFO condition flags. Note that the Empty (D2) condition works differently from the Half-full (D1) and Full (D0) conditions in terms of interrupt generation. The FIFO Half-full and FIFO Full interrupts will occur when one of those flags goes **TRUE** (i. e., to the logic '0' state). The FIFO **NOT** Empty interrupt will occur when the Empty Flag goes **FALSE**. In other words, the interrupt will occur when at least one sample is available in the FIFO, as opposed to when none are. To mask interrupts in response to FIFO condition flags, clear the appropriate bit to the logic '0' state.

3.6 **Channel RAM** - Read/Write @ Symbolic Address Chan_RAM

The Channel RAM is a versatile mechanism for specifying which of the 32 input channels are active. It is organized as a table of 128 entries of 8-bit data fields (128 x 8). Each entry specifies a channel number for the input multiplexer to scan. For each sample clock, the Channel RAM sequences from the first entry to the last and then resets to the first entry to be ready for the next sample clock.

The five least significant bits of each entry correspond to the binary code for the channel. The last channel in the list of Channel RAM entries is tagged with a 'last channel' bit in the most significant bit position. This controls the restarting of the sequence. A parity bit is also present to maintain even parity over the 6 least significant bits. Bits D6 and D7 are not involved in parity checking.

A summary of the bit fields for the Channel RAM entry is shown in Table 3-10, below.

Table 3-10: Model 4248 - Channel RAM Fields - Read/Write @ Address Chan_RAM

Bit #	D7	D6	D5	D4	D3	D2	D1	D0
Function	Last Channel	not used	Parity Bit	Channel Selection Field				
				C16	C8	C4	C2	C1
Definition	0 = False 1 = True	— —	See (2) below	0 = 0 1 = 16	0 = 0 1 = 8	0 = 0 1 = 4	0 = 0 1 = 2	0 = 0 1 = 1
			Parity Field					

The rules for encoding a channel entry are relatively straightforward:

1) The 5 least significant bits (D4 - D0) correspond to the straight binary representation of the channel number (0 through 31 decimal = 0 to 11111 binary = 00 to 1F hexadecimal).
2) Set the Parity bit (D5) so that there are always an even number of '1's over the 6 least significant bits in the entry.
3) Set the Last Channel bit (D7) to logic '1' for the last channel in the list only. This bit does **not** affect the parity.

In order to simplify the encoding, Table 2-11 has been prepared, which lists the possible Channel RAM entries. It is arranged in two columns: one for entries that are not last in the list (normal entries) and the second for last channel entries. The parity bit has been computed for each entry. All entries are expressed in both binary and hexadecimal notation for two hex digits or one byte. Table 2-11 appears at the top of the next page.

3.6 Channel RAM (continued)

Table 3-11: Model 4248 - Channel RAM Encoding Table

Chan. #	Normal Entry			Last Chan. Entry			Chan. #	Normal Entry			Last Chan. Entry		
(decimal)	(binary)		(hex)	(binary)		(hex)	(decimal)	(binary)		(hex)	(binary)		(hex)
0	00 0 0 0000		00	10 0 0 0000		80	16	00 1 1 0000		30	10 1 1 0000		B0
1	00 1 0 0001		21	10 1 0 0001		A1	17	00 0 1 0001		11	10 0 1 0001		91
2	00 1 0 0010		22	10 1 0 0010		A2	18	00 0 1 0010		12	10 0 1 0010		92
3	00 0 0 0011		03	10 0 0 0011		83	19	00 1 1 0011		33	10 1 1 0011		B3
4	00 1 0 0100		24	10 1 0 0100		A4	20	00 0 1 0100		14	10 0 1 0100		94
5	00 0 0 0101		05	10 0 0 0101		85	21	00 1 1 0101		35	10 1 1 0101		B5
6	00 0 0 0110		06	10 0 0 0110		86	22	00 1 1 0110		36	10 1 1 0110		B6
7	00 1 0 0111		27	10 1 0 0111		A7	23	00 0 1 0111		17	10 0 1 0111		97
8	00 1 0 1000		28	10 1 0 1000		A8	24	00 0 1 1000		18	10 0 1 1000		98
9	00 0 0 1001		09	10 0 0 1001		89	25	00 1 1 1001		39	10 1 1 1001		B9
10	00 0 0 1010		0A	10 0 0 1010		8A	26	00 1 1 1010		3A	10 1 1 1010		BA
11	00 1 0 1011		2B	10 1 0 1011		AB	27	00 0 1 1011		1B	10 0 1 1011		9B
12	00 0 0 1100		0C	10 0 0 1100		8C	28	00 1 1 1100		3C	10 1 1 1100		BC
13	00 1 0 1101		2D	10 1 0 1101		AD	29	00 0 1 1101		1D	10 0 1 1101		9D
14	00 1 0 1110		2E	10 1 0 1110		AE	30	00 0 1 1110		1E	10 0 1 1110		9E
15	00 0 0 1111		0F	10 0 0 1111		8F	31	00 1 1 1111		3F	10 1 1 1111		BF
		P Chan #			P Chan #				P Chan #			P Chan #	

NOTES:

(1) The 5 LSBs are simply the channel number in binary (see **Chan #** fields, above).
(2) The Parity bit (**P**, above) is set if there are an ODD number of 1's in the **Chan #** field.
(3) The MSB is set **ONLY** for the last channel in the list of channels to be converted.

3.6.1 Accessing the Channel RAM

The Channel RAM can be loaded and verified over the MIX bus using the addresses given in Section 3.2. This operation will be described by way of example. The Channel RAM can only be accessed sequentially. To ensure proper address alignment, it is always recommended that the Channel RAM Address Counter be reset before each access. This will then cause the counter to point to location zero in the RAM.

Suppose we wish to scan 5 channels: Channels 4, 5, 6, 7, and 8. The sequence of C commands shown at the top of the next page may be executed by the processor that has control of the MIX bus.

3.6 Channel RAM (continued)

3.6.1 Accessing the Channel RAM (continued)

```
unsigned int *cntrl, *chram;        /* define address pointers */
cntrl = MIX_base + 0x0;             /* address offset of control reg */
chram = MIX_base = 0x3;             /* address offset of channel RAM */
*cntrl     =  0x00;                 /* reset FIFO, turn off
                                       clock gates, set channel run/
                                       load to load, reset channel
                                       address counter to 0 */
*cntrl     =   0x10;                /* release channel RAM
                                       address counter */
*chram     =   0x24;                /* entry for channel 4 -
                                       RAM address 0 */
*chram     =   0x05;                /* entry for channel 5 -
                                       RAM address 1 */
*chram     =   0x06;                /* entry for channel 6 -
                                       RAM address 2 */
*chram     =   0x27;                /* entry for channel 7 -
                                       RAM address 3 */
*chram     =   0xA8;                /* entry for channel 8 -
                                       last channel - RAM addr 4 */
```

Here, the channel entries were taken from Table 3-11, on the previous page. Each time the Channel RAM is written into, the Channel RAM Address Counter increments automatically to point to the next location. To verify the contents of the Channel RAM and place them into a table:

```
unsigned int table[32];             /* define address pointer */

*cntrl     =   0x00;                /* reset FIFO, turn off
                                       clock gates, set channel run/
                                       load to load, reset channel
                                       address counter to 0 */
*cntrl     =   0x10;                /* release channel RAM
                                       address counter */
table[1]   =   *chram;              /* read channel RAM address 0 */
table[2]   =   *chram;              /* read channel RAM address 1 */
table[3]   =   *chram;              /* read channel RAM address 2 */
table[4]   =   *chram;              /* read channel RAM address 3 */
table[5]   =   *chram;              /* read channel RAM address 4 */
```

After verification, the Channel RAM Address Counter should be reset before sampling begins. This is performed by:

```
*cntrl     =   0x00;                /* reset FIFO, turn off
                                       clock gates, set channel run/
                                       load to load, reset channel
                                       address counter to 0 */
*cntrl     =   0x10;                /* release channel RAM
                                       address counter */
```

3.7 **Counter Timer Circuit** - Read/Write @ Symbolic Address CTC_Reg

The Counter Timer Circuit (CTC) is a complex device. It is manufactured by Intel Corporation as part number 82C54. This section presents a summary of how the CTC is used in the Model 4248. For a complete description of the part and all its operating modes, please refer to Appendix B of this manual, which is a reprint of Intel's Data Sheet for the 82C54.

The control circuitry surrounding the CTC is shown in Figure 3-4, below.

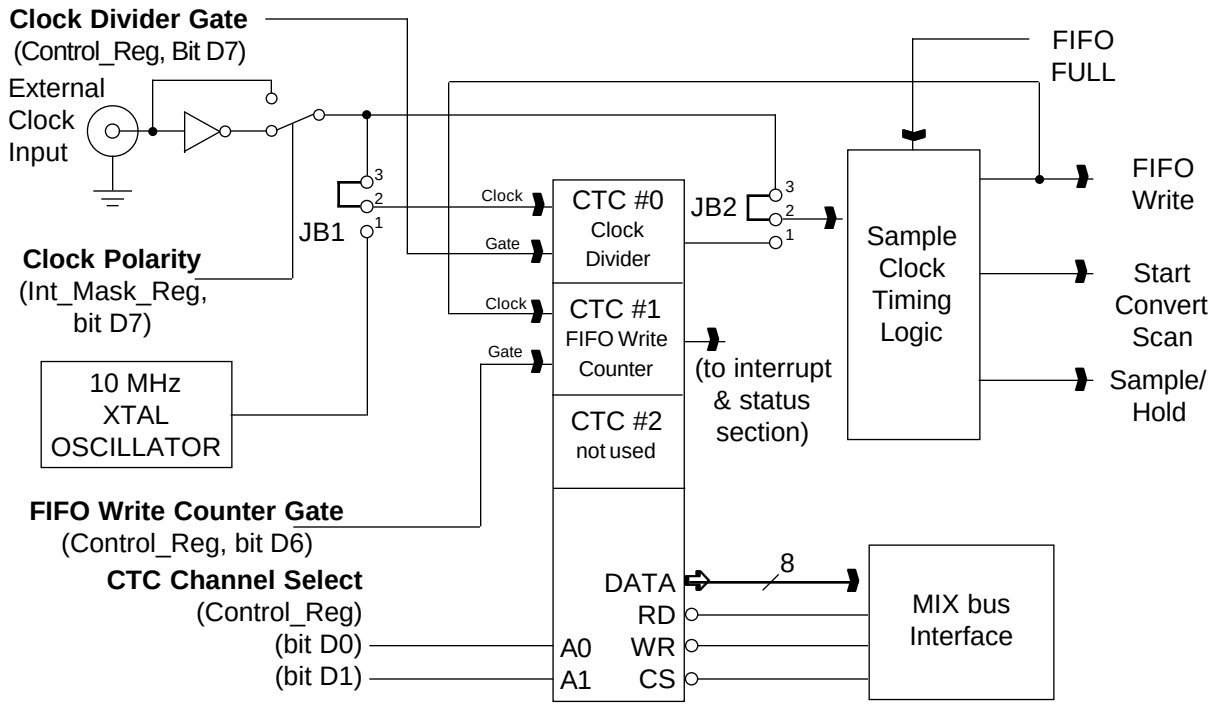


Figure 3-4: Model 4248 - Counter Timer Control Circuitry

The four sections within the CTC correspond to the three counter timer channels plus the control section. Selection is made by the two least significant Model 4248 Control Register bits, bits D1 and D0, which correspond to the CTC pins A1 and A0.

These two bits must first be set up in the Control Register before reading and writing. RD (read), WR (write), and CS (chip select) lines are all decoded and driven by the MIX Interface section.

The 82C54 is an 8-bit device and all I/O takes place on the least significant 8 bits of the 32-bit MIX data bus. The 24 most significant data bits are ignored in writing and meaningless for reads.

3.7 Counter Timer Circuit (continued)

3.7.1 Setting Up the Clock Divider - CTC #0

In order to describe the settings for the Clock Divider an example will be presented. Suppose we need a 20 kHz sample clock and wish to use the internal 10 MHz oscillator as the reference signal. In this case, we would set up the clock select jumpers as follows:

Jumper JB1: 1-2	(selects 10 MHz oscillator as input to CTC #0)
Jumper JB2: 1-2	(selects sample clock source to be CTC #0 output)

The recommended mode for clock divider operation of the 82C54 is Mode 2: Rate Generator. In this mode, the unit acts as a divide-by-N counter. N is programmed either as an 8-bit or 16-bit integer. The counter starts with the output pin at logic '1'. When the number of programmed counts has elapsed, the output goes to '0' for one clock interval and then returns to '1'. The counter continues to cycle every N counts.

If the gate input is low, input clocks are blocked and counting is inhibited. When the gate goes to logic '1', the counter is reloaded with the initial count on the next clock pulse and goes low N clock pulses after the initial count is loaded. The gate can be effectively used to reset and synchronize the counter.

To program a frequency of 20 kHz, we need a divisor of N = 10 MHz / 20 kHz = 500 = 0x1F4. This requires the use of the 16-bit divisor mode (since N > 256). The sequence of commands shown below may be executed by the MIX bus Master.

```
unsigned int *cntrl, *count;        /* define address pointers */
cntrl   =   MIX_base + 0x00;        /* address offset for control reg */
count   =   MIX_base + 0x02;        /* address offset for CTC_Reg */
*cntrl  =   0x00;                   /* reset FIFO, turn off
                                       clock gates, set channel run/
                                       load to load, reset channel
                                       address counter to 0 */
*cntrl  =   0x03;                   /* select CTC_Reg control
                                       section address */
*count  =   0x34;                   /* select control for CTC #0,
                                       set 16-bit divisor mode,
                                       select binary mode, select
                                       mode 2 operation */
*cntrl  =   0x00;                   /* select CTC #0 initial
                                       count address */
*count  =   0x0F4;                  /* load 8 LSB's of initial
                                       count value to CTC #0 */
*count  =   0x01;                   /* load 8 MSB's of initial
                                       count value to CTC #0 */
```

3.7 Counter Timer Circuit (continued)

3.7.1 Setting Up the Clock Divider (continued)

The system may now be started by releasing the Clock Divider Gate (Control_Reg, bit D7) and other functions by executing the following command:

```
*cntrl  =  0x0F4;                  /* release the following:
                                      clock divider gate, FIFO
                                      write gate, channel run/load,
                                      channel RAM address
                                      reset ,and FIFO reset */
```

3.7.2 Setting Up the FIFO Write Counter

As above, we will describe the settings for the FIFO Write Counter by means of an example. Suppose we wish to collect 16 blocks of samples repetitively, where each block contains one sample from each of the 32 A/D channels. This represents a total of 16 x 32 = 512 samples written into the FIFO. We also wish to interrupt the MIX bus Master at the end of each cycle so it can move 512 points at a time.

As with the Clock Divider section, the recommended mode for FIFO Write Counter is Mode 2: Rate Generator. In this mode, the unit acts as a divide-by-N counter. N is programmed either as an 8-bit or 16-bit integer. The counter starts with the output pin at logic '1'. When the number of programmed counts has elapsed, the output goes to '0' for one clock interval and then returns to '1'. The counter continues to cycle every N counts.

If the gate input is low, input clocks are blocked and counting is inhibited. When the gate goes to logic '1', the counter is reloaded with the initial count on the next clock pulse and goes low N clock pulses after the initial count is loaded. The gate can be effectively used to reset and synchronize the counter.

To program a value of 512 = 0x200 samples, the MIX bus Master may execute the sequence of commands shown at the top of the next page.

3.7 Counter Timer Circuit (continued)

3.7.2 Setting Up the FIFO Write Counter (continued)

```
unsigned int *cntrl, *count;        /* define address pointers */

cntrl    =   MIX_base + 0x00        /* control reg address offset */
count    =   MIX_base + 0x02;       /* CTC_Reg address offset */
*cntrl   =   0x00;                  /* reset FIFO, turn off
                                       clock gates, set channel run/
                                       load to load, reset channel
                                       address counter to 0 */
*cntrl   =   0x03;                  /* select CTC_Reg control
                                       section address */
*count   =   0x74;                  /* select control for CTC #1,
                                       set 16-bit divisor mode,
                                       select binary mode,
                                       select mode 2 operation */
*cntrl   =   0x01;                  /* select CTC #1 initial
                                       count address */
*count   =   0x00;                  /* load 8 LSB's of initial
                                       count value to CTC #1 */
*count   =   0x02;                  /* load 8 MSB's of initial
                                       count value to CTC #1 */
```

Now the Interrupt Mask Register must be set up to enable interrupts from the FIFO Write Counter.

```
unsigned int *intupt;               /* define address pointers */
intupt   =   MIX_base + 0x01;       /* define 4248 Interrupt
                                       Mask Register */
*intupt  =   0x10;                  /* enable FIFO Write
                                       Interrupt */
```

The system may now be started by releasing the FIFO Write Gate (Control Register, bit D6) and other functions by executing the following command:

```
*cntrl   =   0x0F4;                 /* release the following:
                                       clock divider gate, FIFO
                                       write gate, channel run/load,
                                       channel RAM address
                                       reset ,and FIFO reset */
```

3.8 Sample Clock Timing Logic

The Model 4248 Sample Control Timing Logic is a state machine that is triggered by a sample clock pulse. Once triggered, the state machine initiates the sequence of events shown in Table 3-12, below.

Table 3-12: Model 4248 - Sample Clock Timing - One Complete Cycle of N Channels

Time (nsec)	Event
0	Active edge of sample clock occurs S&H amplifiers switched to HOLD mode Wait for settling delay (500 nsec)
500	S&H settling time delay has completed Start first A/D conversion Switch MUX to second channel
700	Write first A/D output into FIFO Start second A/D conversion Switch MUX to third channel
900	Write second A/D output into FIFO Start third A/D conversion Switch MUX to fourth channel
1100	Write third A/D output into FIFO Start fourth A/D conversion Switch MUX to fifth channel
⋮	⋮
500 + (200 x N)	Write last A/D output into FIFO Set S&H to TRACK mode Wait for acquisition delay (1000 nsec)
1500 + (200 x N)	Acquisition delay complete Ready for next sample clock

As the table above indicates, samples are written into the FIFO at the rate of one sample every 200 nsec. From the above, we can derive an equation for the maximum sampling frequency (fs_{max}), as a function of the number of channels to be converted (N). The equation is shown below.

$$fs_{max} = \frac{10\ \text{MHz}}{2N + 15}$$

3.9 Interrupt Timing Considerations

The Model 4248 offers a wide variety of interrupt mechanisms. A few examples will be described below:

3.9.1 Interrupt on FIFO Not Empty

The Interrupt Mask Register should be written with bit D2 = '1'. This can be useful when the interrupt service routine must start at the earliest possible moment. The maximum transfer rate is dependent upon the device that is mastering the bus. Assembly language coding will usually result in faster transfer rates than can be achieved with 'C' programming, even if the 'C' code is optimized.

3.9.2 Interrupt on Channel Sync

The Interrupt Mask Register should be written with bit D5 = '1'. This is useful when the interrupt service routine is to be started after all data for one complete channel scan is finished. At this time, the FIFO contains N data samples, which can be read out at the maximum MIX bus rate.

3.9.3 Interrupt on FIFO Write Counter

The Interrupt Mask Register should be written with bit D4 = '1'. This can be used for interrupting on a group of complete channel scans, and is described in Section 3.7.2. As with the Channel Sync interrupt, all data is guaranteed to be ready and waiting in the FIFO so reading can take place at the maximum MIX bus transfer rate.

3.9.4 Interrupt on FIFO Half -Full

The Interrupt Mask Register should be written with bit D1= '1'. This is usually the most efficient, and therefore the most popular method of using FIFOs. By interrupting the processor when the FIFO is half full, a reasonable block of data (half the FIFO depth) can be efficiently transferred at once, saving interrupt overhead time for the processor. The other benefit is that while the interrupt routine is responding and completing, up to half a FIFO of additional data can be safely collected without any data loss.

3.9.5 Interrupt on FIFO Full

The Interrupt Mask Register should be written with bit D0 = '1'. This may be used when notification of possible data loss is required or when loss of data is acceptable.

3.9 Interrupt Timing Considerations (continued)

3.9.6 Interrupt on Error or Overload

These interrupt conditions can be included with any other interrupt condition as a means of error checking. By polling the Status Register in the interrupt service routine, the status bits corresponding to these conditions can be tested and appropriate action taken.

3.10 A/D FIFO Data Format - Read Only @ Symbolic Address InData_Reg

MIX bus Masters collect data from the Model 4248 by performing successive read operations at the symbolic address InData_Reg. The bit definition for the 32-bit data word delivered by the 4248, for both 12-bit (standard) and 14-bit (Option 003) units, is shown in Table 3-13, below.

Table 3-13: Model 4248 - FIFO Data Read Bit Definition

12-bit Data	D31 - - - - - - D20	D19	D18	D17	D16	D15 - - - - - - D4	D3	D2	D1	D0
Contents	MSB - - - - - - LSB	0	0	SU	SL	MSB - - - - - - LSB	0	0	SU	SL
14-bit Data	**D31 - - - - - - - - - - - - - - - D18**			**D17**	**D16**	**D15 - - - - - - - - - - - - - - - D2**			**D1**	**D0**
Contents	MSB - - - - - - - - - - - - - - - LSB			SU	SL	MSB - - - - - - - - - - - - - - - LSB			SU	SL

The 32-bit word consists of two identical adjacent 16-bit fields (i. e., the same sample is contained in both words of the MIX longword). The upper field extends from bit D31 to D16, and the lower field from D15 to D0. The A/D output word is left justified in each 16-bit field and is designated MSB (most significant bit) through LSB (least significant bit). The MSB of each sample can be found in bits D31 and D15. For 12-bit units, the LSB of each sample is found in bits D20 and D4, and bits D19, D18, D3 and D2 are always 0. For Option 003 (14-bits), the LSB is found in bits D18 and D2.

The bottom two bits of each 16-bit field reflect the upper and lower channel sync bits SL and SU, respectively. These bits indicate the last channel of a scan sequence, as shown in Table 3-14, below.

Table 3-14: Model 4248 - Upper and Lower Sync Bits

Function	SU	SL
Not Last Channel	0	0
Error	0	1
Error	1	0
Last Channel	1	1

3.10 A/D FIFO Data Format (continued)

These bits are useful in framing the sequence of data words when demultiplexing data. For example, when channels 4 through 13 are enabled, the SU and SL bits will be cleared to the logic '0' state for samples from channels 4 through 12, and will be set to the logic '1' state for channel 13 only. These bits will normally always be the same, i. e., both '0' or both '1'. In the event that they are different, the upper and lower FIFO's have become unsynchronized relative to each other and the data value will be corrupted. This is described in greater detail in Section 3.4.4.

A/D converter data values correspond to the 2's complement coding scheme within a 32-bit field, as shown in Table 3-15, below.

Table 3-15: Model 4248 - A/D Converter Data Values

12-bit Data	'Non-Last Channel' Words	'Last Channel' Words
+Full Scale:	0x7FF0 7FF0	0x7FF3 7FF3
+1 LSB:	0x0010 0010	0x0013 0013
Zero:	0x0000 0000	0x0003 0003
-1 LSB:	0xFFF0 FFF0	0xFFF3 FFF3
-Full Scale:	0x8000 8000	0x8003 8003
14-bit Data	**'Non-Last Channel' Words**	**'Last Channel' Words**
+Full Scale:	0x7FFC 7FFC	0x7FFF 7FFF
+1 LSB:	0x0004 0004	0x0007 0007
Zero:	0x0000 0000	0x0003 0003
-1 LSB:	0xFFFC FFFC	0xFFFF FFFF
-Full Scale:	0x8000 8000	0x8003 8003

If an attempt is made to read the FIFO when it contains no more data (i. e., when the Empty Flag is in the True (logic '0') state), meaningless data will be returned, and the bus cycle will complete normally.

This page is intentionally blank

Appendix A: Programming Examples

A.1 Introduction

The 'C' programs that follow were written to allow the Model 4248 to be used on any of Pentek's VME/MIX baseboards. Four sample source files are included.

Two header files are presented here, **mod.h** (see Section A.4, starting in the middle of page A-6) and **4248.h** (see Section A.2, below). The user initializes constants in the file **mod.h** prior to compiling, to select the type of MIX baseboard he will use and the MIX stack position of the Model 4248. This file (**mod.h**) initializes several symbolic constants used by the other files included here, and is intended as a general purpose header file for use with all Pentek MIX modules. The file **4248.h** contains header information specific to the Model 4248.

The file **p4248.c** (see Section A.3, beginning in the middle of page A-2) contains the code that allows the baseboard selected in **mod.h** to initialize the Model 4248, release it to accept analog inputs, monitor its status and read out the data it collects. Prior to compiling, the user sets symbolic constants in this file to select the input configuration, the input clock rate, and to decide whether interrupts or polling will be used to determine when the 4248 is ready to deliver data. Finally, the file **intr.c** (see section A.5, starting on page A-9) determines which (if any) MIX interrupts to enable, using information from **mod.h**.

A.2 4248.h

```
/* 4248.h  - This file contains 4248 module specific information */
#ifndef P4248_H
#define P4248_H

#define P4248_MAX_CHANS          32
#define P4248_FIFO_SIZE          (16 * 1024)
/* 4248 MIX Module Register Offsets */
#define P4248_CONTROL_REG        0x00000000L
#define P4248_INTR_MASK_REG      0x00000001L
#define P4248_CTC_REG            0x00000002L
#define P4248_STATUS_REG         0x00000002L | A27_ADDR_BIT
#define P4248_CHAN_RAM           0x00000003L
#define P4248_INDATA_REG         0x00000003L | A27_ADDR_BIT

/* Constants specific to each Register */

/* Status Register */
#define SR_FIFO_FULL             0x00000001L
#define SR_FIFO_HALF_FULL        0x00000002L
#define SR_FIFO_EMPTY            0x00000004L
#define SR_PARITY_ERROR          0x00000008L
#define SR_ERROR                 0x00000010L
#define SR_OVERLOAD_DETECT       0x00000020L
#define SR_FIFO_WRITE_CNTR       0x00000040L
#define SR_LAST_CHAN_SYNC        0x00000080L
```

A.2 4248.h (continued)

```
/* Control Register  */
#define CR_CTC_CHAN_SELECT_MASK  0x00000003L
#define CR_FIFO_RESET            0x00000004L
#define CR_CHAN_RAM_RESET        0x00000010L
#define CR_CHAN_RUN_LOAD         0x00000020L
#define CR_FIFO_WRITE_CNTR       0x00000040L
#define CR_CLOCK_DIV_GATE        0x00000080L

/* Channel Select Bit Mask (Control Register Bits 0-1) */
#define CR_CLK_DIV_SLCT          0x00000000L
#define CR_FIFO_WRITE_SLCT       0x00000001L
#define CR_CTC_CNTL_SLCT         0x00000003L

/* Interrupt Mask Register  */
#define IM_FIFO_FULL             0x00000001L
#define IM_FIFO_HALF_FULL        0x00000002L
#define IM_FIFO_NOT_EMPTY        0x00000004L
#define IM_ERROR                 0x00000008L
#define IM_FIFO_WRITE_CNTR       0x00000010L
#define IM_LAST_CAHN_SYNC        0x00000020L
#define IM_OVERLOAD_DETECT       0x00000040L
#define IM_CLOCK_POLORITY        0x00000080L

#endif
```

A.3 p4248.c

```
#include "mod.h"
#include "4248.h"

/* Method of determining Data Ready */
#define  POLLING          0
#define  INTERRUPTS       1


#define  NUM_CHANNELS     8         /* Number of Channels to enable */
#define  SMPL_RATE_DIV  500               /* Sample Rate Divisor = */
                                  /* 10 MHz/Sample Frequency */
unsigned int *PageAddressReg, *ExpBusControlReg;
unsigned int *Status, *Control, *ChanRam;
unsigned int *CtcMode, *InData, *IntrMask;
unsigned int num_samples;

unsigned int sig1[1024],sig2[1024],sig3[1024],sig4[1024];
unsigned int sig5[1024],sig6[1024],sig7[1024],sig8[1024];

extern void enableInterrupts();
extern void clearInterrupt();
```

A.3 p4248.c (continued)

```
main()
{
   unsigned int i;

   #if BB_4283
      ExpBusControlReg = (unsigned int *) 0x808060u;
      PageAddressReg   = (unsigned int *) 0x800002u;
   #endif

   Status      = (unsigned int *) (MOD | P4248_STATUS_REG);
   Control     = (unsigned int *) (MOD | P4248_CONTROL_REG);
   IntrMask    = (unsigned int *) (MOD | P4248_INTR_MASK_REG);
   InData      = (unsigned int *) (MOD | P4248_INDATA_REG);
   CtcMode     = (unsigned int *) (MOD | P4248_CTC_REG);
   ChanRam     = (unsigned int *) (MOD | P4248_CHAN_RAM);

   #if BB_4283
      *ExpBusControlReg = 0x68;        /* Remove MIX bus wait states */
      *PageAddressReg = MOD_SELECT;               /* Select module */
   #endif

   #if BB_4284
      *(unsigned long *) 0x100020 = 0;            /* Turn off timer */
   #endif

   #if MM_4270
      *(unsigned int *) 0x34000d = 0xff;   /* Disable NMI generation */
   #endif

   *Control=0x0u;                                    /* Reset FIFOs */

   #if INTERRUPTS

      #if BB_4283
         /* Select module w/A27 = 1 */
         *PageAddressReg = MOD_SELECT | D17_DATA_BIT;
      #endif

      /* Read Status Register to clear pending interrupts */
      i = *Status;

      #if BB_4283
         *PageAddressReg = MOD_SELECT;             /* Select module */
      #endif

      enableInterrupts();          /* Enable External MIX Interrupt */
   #endif

   init4248sampling(SMPL_RATE_DIV);     /* Set sampling rate divisor */
   init4248channel(NUM_CHANNELS);     /* Enable Channels in Chan RAM */
```

A.3 **p4248.c** (continued)

```
   /* Determine the number of samples per channel */
   num_samples = (P4248_FIFO_SIZE / 2) / NUM_CHANNELS;
   while (1)
   {
      #if POLLING
         for (i=0;i<num_samples;i++)
         {
            /* wait for input FIFO half-full */
            while ((*Status & SR_FIFO_HALF_FULL) != 0);

            sig1[i]=*InData;
            sig2[i]=*InData;
            sig3[i]=*InData;
            sig4[i]=*InData;
            sig5[i]=*InData;
            sig6[i]=*InData;
            sig7[i]=*InData;
            sig8[i]=*InData;
         }
      #else
         ;
      #endif
   }
}


/* Channel Ram Encoding Table - This table contains the channel number
                                and its parity                        */
unsigned int adseq[] = {0x00, 0x21, 0x22, 0x03, 0x24, 0x05, 0x06, 0x27,
                        0x28, 0x09, 0x0a, 0x2b, 0x0c, 0x2d, 0x2e, 0x0f,
                        0x30, 0x11, 0x12, 0x33, 0x14, 0x35, 0x36, 0x17,
                        0x18, 0x39, 0x3a, 0x1b, 0x3c, 0x1d, 0x1e, 0xbf};


void init4248channel(int num_channels)
{
   int      i;

   #if BB_4283
      *PageAddressReg=MOD_SELECT;        /* Select module 0, A27 = 0 */
   #endif

   *Control=0x10u;                          /* Initialize Channel RAM */

   for( i = 0 ; i < num_channels - 1 ; i++ )
      *ChanRam = adseq[i];                             /* Set Channels */
   *ChanRam = adseq[i] | 0x80;                  /* Set last channel */

   *Control=0;                                        /* Reset FIFO */

   #if INTERRUPTS
      *IntrMask = IM_FIFO_HALF_FULL;             /* Enable Interrupt */
   #endif
```

A.3 p4248.c (continued)

```
   *Control=0xf4;                          /* Release and Acquire */

   #if BB_4283
      /* Select module w/A27 = 1 */
      *PageAddressReg = MOD_SELECT | D17_DATA_BIT;
   #endif
}


void ResetFifo(void)
{
   #if BB_4283
      *PageAddressReg=MOD_SELECT;        /* Select module 0, A27 = 0 */
   #endif

   *Control=0;                                          /* Reset FIFO */

   #if INTERRUPTS
      *IntrMask = IM_FIFO_HALF_FULL;             /* Enable Interrupt */
   #endif

      *Control=0xf4;                          /* Release and Acquire */

   #if BB_4283
      /* Select module w/A27 = 1 */
      *PageAddressReg=MOD_SELECT | D17_DATA_BIT;
   #endif
}


void init4248sampling(unsigned int div)
{
   #if BB_4283
      *PageAddressReg = MOD_SELECT;      /* Select module 0, A27 = 0 */
   #endif

   *Control = 3;                       /* Select mode register of CTC */
   *CtcMode = 0x34;      /* Program mode register for 16-bit divisor */
   *Control = 0;                          /* Select counter 0 of CTC */
   *CtcMode = div & 0xff;                   /* Frequency Divisor LSB */
   *CtcMode = (div >> 8) & 0xff;            /* Frequency Divisor MSB */

   #if BB_4283
      /* Select module w/A27 = 1 */
      *PageAddressReg = MOD_SELECT | D17_DATA_BIT;
   #endif
}
```

A.3 p4248.c (continued)

```
void c_int01()
{
   unsigned int i;

   for (i = 0; i < num_samples; i++)
   {
      sig1[i]=*InData;
      sig2[i]=*InData;
      sig3[i]=*InData;
      sig4[i]=*InData;
      sig5[i]=*InData;
      sig6[i]=*InData;
      sig7[i]=*InData;
      sig8[i]=*InData;
   }

   /* Read Status Register to clear MIX Interrupt */
   i = *Status;

   /* Clear processor's interrupt */
   clearInterrupt();
}
```

A.4 mod.h

```
/* mod.h -  This file determines the MIX Module's base address based on
            the processor which is running the program, the MIX
            Baseboard, and the MIX stack position being addressed.

 Note:   This file only supports the following DSP processor boards:
         4283        4254        4257
         4284        4270

 Defining the Baseboard:
         Set the appropriate BB_42XX constant to a 1 and the others to 0

 Running programs on a 4270, 4254, or 4257 Mix Module:
         Set the MM_4270 constant if processor is on the 4270
         Set the MM_425X constant if processor is on the 4257

 Running program on the 4284 or 4283:
         Clear the MM_4270 and MM_425X constants

 Specifying the MIX Module stack position of Mix Module being addressed:
         For MIX stack position 0, set MOD to MOD0_ADDR
         For Mix stack position 1, set MOD to MOD1_ADDR
         For Mix stack position 2, set MOD to MOD2_ADDR

 Running programs on the 4283 Mix Baseboard(Setting the Page Register):
         To select MIX stack position 0, set MOD_SELECT to MOD0_SELECT
         To select MIX stack position 1, set MOD_SELECT to MOD1_SELECT
         To select MIX stack position 2, set MOD_SELECT to MOD2_SELECT */
```

A.4 **mod.h** (continued)

```
/* Define Baseboard Type */
#define BB_4283         0
#define BB_4284         0
#define BB_4200         0
#define BB_4201         1
#define BB_4202         0

/* Define MIX Module Type - Only if program is to be executed
                                    on a MIX Module Processor (UMBM) */
#define MM_4270         1
#define MM_425X         0

/* Specify MIX Module Position of module being addressed */
#define MOD             MOD1_ADDR

#if MM_4270 | MM_425X
   /* Specify MIX STACK Position for the 4270 or 425X processor boards */
   #define MIX_POS      0
#endif

#if BB_4283
   /* The 4283 Selects the MIX Module being addressed with a Page Register */
   #define MOD_SELECT   MOD1_SELECT
#else
   #define MOD_SELECT   MOD      /* Only being set to keep code similar */
#endif

/* Baseboard = 4283 - This Baseboard does not support Upper Mix Bus Masters */
#if BB_4283
   /* The 4283 Baseboard uses a page register */
   /* to select which MIX Module is being accessed */
   #define MOD0_SELECT  0x18000
   #define MOD1_SELECT  0x10000
   #define MOD2_SELECT  0x8000

   #define MOD0_ADDR    0x00804000
   #define MOD1_ADDR    0x00804000
   #define MOD2_ADDR    0x00804000

   /* Some Mix Modules require the setting of Address Bit 27 */
   /* This is selected in the Page Register for this baseboard */
   #define D17_DATA_BIT 0x00020000

   /* Address Bit in address which corresponds to A27
                                              - This is unused by 4283 */
   #define A27_ADDR_BIT 0x00000000

#endif
```

A.4 **mod.h** (continued)

```
/* Baseboard = 4284 */
#if BB_4284
   #if MM_4270 | MM_425X       /* MIX UMBM Module = 4270, 4254, or 4257 */
      #define MOD0_ADDR    0xC0000000
      #define MOD1_ADDR    0xC4000000
      #define MOD2_ADDR    0xC8000000
      /* Address Bit in address which corresponds to A27 */
      #define A27_ADDR_BIT 0x02000000
   #else                                    /* No MIX UMBM Module */
      #define MOD0_ADDR    0x20000000
      #define MOD1_ADDR    0x24000000
      #define MOD2_ADDR    0x28000000
      /* Address Bit in address which corresponds to A27 */
      #define A27_ADDR_BIT 0x02000000
   #endif
#endif

/* Baseboard = 4200 or 4201 */
#if BB_4200 | BB_4201
   #if MM_4270 | MM_425X       /* MIX UMBM Module = 4270, 4254, or 4257 */
      #define MOD0_ADDR    0xF0000000
      #define MOD1_ADDR    0xF4000000
      #define MOD2_ADDR    0xF8000000
      /* Address Bit in address which corresponds to A27 */
      #define A27_ADDR_BIT 0x02000000
   #else                                    /* No MIX UMBM Module */
      #define MOD0_ADDR    0xC0000000
      #define MOD1_ADDR    0xD0000000
      #define MOD2_ADDR    0xE0000000
      /* Address Bit in address which corresponds to A27 */
      #define A27_ADDR_BIT 0x02000000
   #endif
#endif

/* Baseboard = 4202 */
#if BB_4202
   #if MM_4270 | MM_425X       /* MIX UMBM Module = 4270, 4254, or 4257 */
      #define MOD0_ADDR    0xC0000000
      #define MOD1_ADDR    0xC4000000
      #define MOD2_ADDR    0xC8000000
      /* Address Bit in address which corresponds to A27 */
      #define A27_ADDR_BIT 0x02000000
   #endif
#endif

/*************************** End of File ****************************/
```

A.5 intr.c

```
#include "mod.h"
unsigned int   *int01_vector;
extern int     c_int01();
/* This Routine decides from the Information contained in mod.h
             which MIX Interrupt should be enabled by the processor */

void enableInterrupts()
{
   #if BB_4283
      /* If the MOD_SELECT value used to select each MIX Module
           Position matches, then enable the corresponding Interrupt */
      if (MOD_SELECT == MOD0_SELECT)
         Init1();
      else if (MOD_SELECT == MOD1_SELECT)
         Init2();
      else
         Init3();
   #endif

   #if BB_4284 & !(MM_4270 | MM_425X)
      /* If the Base Address used to address each MIX Module
           Position matches, then enable the corresponding Interrupt */
      if (MOD == MOD0_ADDR)
         Init1();
      else if (MOD == MOD1_ADDR)
         Init2();
      else
         Init3();
   #endif

   #if MM_4270 | MM_425X
      /* Based on the position of the 4270 or 425X Mix Modules
            and position of the MIX module generating the interrupts
                                  enable the corresponding Interrupt */
      if (MIX_POS == 0)
      {
         if (MOD == MOD1_ADDR)
            IntInit2();
         else
            IntInit3();
      }
      else if (MIX_POS == 1)
      {
         if (MOD == MOD0_ADDR)
            IntInit3();
         else
            IntInit2();
      }
      else
      {
         if (MOD == MOD0_ADDR)
            IntInit2();
         else
            IntInit3();
      }
```

A.5 **intr.c** (continued)

```
      #if BB_4200 | BB_4201
         /* If the Baseboard is a 4200 or 4201, Disable the VIC Local
               Interrupt Control Registers from responding to the MIX
                                          Interrupt from the Mix Module */
         if (MOD == MOD0_ADDR)
            *(unsigned long *) 0xffff0009 = 0x80;
         else if (MOD == MOD1_ADDR)
            *(unsigned long *) 0xffff000a = 0x80;
         else
            *(unsigned long *) 0xffff000b = 0x80;
      #endif
   #endif
}


void clearInterrupt()
{
   #if MM_4270 | MM_425X | BB_4284
      asm("    iack @0");            /* Clear Any Pending Interrupts */
   #endif

   #if BB_4283               /* Acknowledge 4283 interrupt processor */
      if (MOD_SELECT == MOD0_SELECT)
         *(int *)0x800000 = 0x0c;
      else if (MOD_SELECT == MOD1_SELECT)
         *(int *)0x800000 = 0x0a;
      else
         *(int *)0x800000 = 0x06;

      asm("    NOP");
      *(int *)0x800000 = 0x0e;    /* Re-arm 4283 interrupt processor */

      asm("    LDI 0,IF");      /* Clear out any spurious interrupts */
      asm("    LDI 0,IF");      /* Clear out any spurious interrupts */
   #endif
}


#if BB_4283 | BB_4284
void Init1()
{
   #if BB_4283
      /* setup int1 vector jump */
      *(int *)0x809804 = (int)&c_int01 | 0x60000000;

      *(int *)0x800000 = 0x00;   /* Arm IACK Pal */
      asm("    NOP");
      *(int *)0x800000 = 0x0e;

      /* enable interrupt 0 and 1 */
      asm("    LDI 0,IF");      /* Clear out any spurious interrupts */
      asm("    LDI 3,IE");
      asm("    OR 2000h,ST");
```

A.5 **intr.c** (continued)

```
   #else
      /* Enable C40 Int0 as MixInt0 */
      *(unsigned int *) 0x10000003 = *(unsigned int *) 0x10000003 |
                                                          0x00000008;
      int01_vector = (unsigned int *) c_int01;
      asm("    LDEP IVTP,AR0");
      asm("    LDI @_int01_vector,R0");
      asm("    STI R0,*+AR0(3)");
      asm("    IACK @0");
      asm("    LDI IIF,R0");
      asm("    ANDN 000fH,R0");
      asm("    OR 0009H,R0");
      asm("    LDI R0,IIF");
      asm("    OR 2000H,ST");
   #endif
}


void Init2()
{
   #if BB_4283
      /* setup int2 vector jump */
      *(int *)0x809808 = (int)&c_int01 | 0x60000000;

      *(int *)0x800000 = 0x00;                    /* Arm IACK Pal */
      asm("    NOP");
      *(int *)0x800000 = 0x0e;

      /* enable interrupt 0 and 2 */
      asm("    LDI 0,IF");      /* Clear out any spurious interrupts */
      asm("    LDI 5,IE");
      asm("    OR 2000h,ST");

   #else
      /* Enable C40 Int2 as MixInt1 */
      *(unsigned int *) 0x10000003 = *(unsigned int *) 0x10000003 |
                                                          0x00100000;
      int01_vector = (unsigned int *) c_int01;
      asm("    LDEP IVTP,AR0");
      asm("    LDI @_int01_vector,R0");
      asm("    STI R0,*+AR0(5)");
      asm("    IACK @0");
      asm("    LDI IIF,R0");
      asm("    ANDN 0f00H,R0");
      asm("    OR 0900H,R0");
      asm("    LDI R0,IIF");
      asm("    OR 2000H,ST");
   #endif
}
```

A.5 **intr.c** (continued)

```
void Init3()
{
   #if BB_4283
      /* setup int 3 vector jump */
      *(int *)0x80980c = (int)&c_int01 | 0x60000000;

      *(int *)0x800000 = 0x00;                        /* Arm IACK Pal */
      asm("    NOP");
      *(int *)0x800000 = 0x0e;

    /* enable interrupt 0 and 3 */
      asm("    LDI 0,IF");      /* Clear out any spurious interrupts */
      asm("    LDI 9,IE");
      asm("    OR 2000h,ST");

   #else
      /* Enable C40 Int 3 as MixInt2 */
      *(unsigned int *) 0x10000003 = *(unsigned int *) 0x10000003 |
                                                          0x20000000;
      int01_vector = (unsigned int *) c_int01;
      asm("    LDEP IVTP,AR0");
      asm("    LDI @_int01_vector,R0");
      asm("    STI R0,*+AR0(6)");
      asm("    IACK @0");
      asm("    LDI IIF,R0");
      asm("    ANDN 0f000H,R0");
      asm("    OR 09000H,R0");
      asm("    LDI R0,IIF");
      asm("    OR 2000H,ST");
   #endif
}
#endif

#if MM_4270 | MM_425X
void IntInit2()
{
   int01_vector = (unsigned int *) c_int01;
   asm("    LDEP IVTP,AR0");
   asm("    LDI @_int01_vector,R0");
   asm("    STI R0,*+AR0(6)");
   asm("    IACK @0");
   asm("    LDI IIF,R0");
   asm("    ANDN 0f000H,R0");
   asm("    OR 09000H,R0");
   asm("    LDI R0,IIF");
   asm("    OR 2000H,ST");
}
```

A.5 **intr.c** (continued)

```
void IntInit3()
{
   int01_vector = (unsigned int *) c_int01;
   asm("    LDEP IVTP,AR0");
   asm("    LDI @_int01_vector,R0");
   asm("    STI R0,*+AR0(5)");
   asm("    IACK @0");
   asm("    LDI IIF,R0");
   asm("    ANDN 0f00H,R0");
   asm("    OR 0900H,R0");
   asm("    LDI R0,IIF");
   asm("    OR 2000H,ST");
}
#endif
```

This page is intentionally blank