

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Pentek 4283* at our website: ***Click HERE***

OPERATING MANUAL

MODEL 4249

1 MHz, 12-bit, Filtered A/D - D/A Converter
MIX Module for VMEbus Systems

Pentek, Inc.
One Park Way
Upper Saddle River, NJ 07458
(201) 818-5900
http://www.pentek.com/

Copyright © 1992 - 1998

Manual Part No: 800.42490 Rev: C.1 - July 2, 1998

Pentek Model 4249 Operating Manual
Revision History

Date	Rev	Applicable Serial #'s	Comments
05/11/92	A	9219022 - Forward	Initial Product Release
02/06/93	A.1	9219022 - Forward	Add MIX Tutorial, Appendix A.
03/31/95	B	9219022 - Forward	New memory maps for use with 4200-series, 4202 and 4284 MIX Baseboards, added in Chapter 3. Chapter 4, which gives Operating hints and tips, added. General reorganization of manual to new company format. Sample code moved to Appendix A, MIX tutorial updated and moved to Appendix C, Appendices B (CTC Data Sheet), D (MIX Baseboard Tutorial) and E (using the 4283 Baseboard) added.
10/20/97	C	9219022 - Forward	Add Memory Map for use w/4285 Baseboard. Update sample code in Appendix A to include 4285. Update & re-designate MIX Tutorial Appendices as X, Y & Z (were C,D & E). Add Figures & Tables to Table of Contents.
07/02/98	C.1	9219022 - Forward	Removed the Mix Tutorial from this manual. Mix Tutorial is now a separate manual (800.00001). Corrected Table 3-6 to have proper 'C40 addresses in both compressed and uncompressed mode.

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.

Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced in any form without written permission.

Table of Contents

Page

Chapter 1: General Information

1.1 Introduction 7
1.2 Anti-Alias Input Filter 7
1.3 Output Smoothing Filter 7
1.4 FIFO Buffers for Efficient Block Transfer 7
1.5 Block Diagram 8
Figure 1-1: Model 4249 - Block Diagram 8
1.6 Specifications 8

Chapter 2: Installation and Connections

2.1 Inspection 11
2.2 Configuration Jumper Settings 11
2.2.1 Separating the Mezzanine Board from the MIX Board 11
Figure 2-1: Jumper Blocks and Mounting Holes 12
2.2.2 Output Filter Bypass Jumper 13
Figure 2-2: Output Filter Bypass Jumper Diagram 13
2.2.3 Sample Clock Generator Jumpers 13
Table 2-1: Output Filter Bypass Jumper 13
Figure 2-3: A/D and D/A Converter Sampling Clocks 14
Table 2-2: Sample Clock Jumpers - JB7, JB8, JB9 - MIX Board 14
2.2.4 Input Filter Bypass Jumper 15
Figure 2-4: Input Filter Bypass Jumpers Diagram 15
2.2.5 Other Jumper Blocks 15
Table 2-3: Output Filter Bypass Jumper - JB10 - MIX Board 15
Figure 2-5: Model 4249 Front Panel 16
2.2.6 Reconnecting the Mezzanine Board to the MIX Board 16
2.3 The Model 4249 Front Panel 16
2.3.1 A/D Input Signal Connections 16
2.3.2 D/A Output Signal Connections 16
Figure 2-6: A/D Input Circuitry 17
Figure 2-7: D/A Output Circuitry 17
2.3.3 External Sample Clock In 17
2.4 Installing the Model 4249 on the MIX Baseboard 18
2.5 Installing the Assembly into the VMEbus Card Cage 18

Chapter 3: Memory Maps and Register Descriptions

3.1 Programming Overview 19

Table of Contents

Page

3.2 Address Maps ... 19
3.2.1 Model 4249 Address Map for the 4200/01 Series Baseboards ... 20
Table 3-1: Memory Map for the Model 42XX Series MIX Baseboards ... 20
3.2.2 Model 4249 Address Map for the Model 4202 Baseboard ... 21
Table 3-2: Memory Map for the Model 4202 MIX Baseboard ... 21
3.2.3 Model 4249 Address Map for the Model 4283 Baseboard ... 22
3.2.3.1 Module Selection ... 22
Table 3-3: Module Selection - Model 4283 Page Register ... 22
3.2.3.2 Register Selection Within the Module ... 23
Table 3-4: Memory Map for the Model 4283 MIX Baseboard ... 23
3.2.4 Model 4249 Address Map for the Model 4284 Baseboard ... 24
Table 3-5: Memory Map for the Model 4284 MIX Baseboard ... 24
3.2.5 Model 4249 Address Map for the Model 4285 Baseboard ... 25
Table 3-6: Memory Map for the Model 4285 MIX Baseboard ... 25
3.3 Control Register ... 26
3.3.1 Overload Interrupt Mask ... 26
3.3.2 DMA Request Control ... 26
3.3.3 FIFO Interrupt Selection Code ... 26
Table 3-7: Control Register ... 26
3.3.4 Output FIFO Reset ... 27
Table 3-8: FIFO Interrupt Selection Code (Part of Control Register) ... 27
3.3.5 Input FIFO Reset ... 28
3.4 Status Register ... 28
3.4.1 Input Overload Status ... 28
Table 3-9:Status Register ... 28
3.4.2 Interrupt Pending ... 29
3.4.3 Input and Output FIFO Flags ... 29
3.5 Counter Timer Registers ... 29
3.5.1 Programming the A/D Sample Clock ... 30
3.5.2 Programming the D/A Sample Clock ... 31
3.6 Input (A/D) FIFO ... 32
Table 3-10: Input (A/D) FIFO Data Format ... 32
3.7 Output (D/A) FIFO ... 33
Table 3-11: Output (D/A) FIFO Data Format ... 33

Chapter 4: Operating Procedures

4.1 Introduction ... 35
4.2 Selecting the A/D and D/A Sampling Clock Sources ... 35
Figure 4-1: A/D and D/A Converter Sampling Clocks ... 35
4.3 Programming the Sample Rate ... 36

Table of Contents

Page

Table 4-1: Sample Clock Jumpers - JB7, JB8, JB9 - MIX Board....36
Table 4-2: 82C54 - CTC Control Word Format....37
4.4 Using the Status Bits....38
4.5 Using Interrupts....39
Table 4-3: FIFO Status Bits - MIX Address Status_Reg....39
Table 4-4: FIFO Interrupt Selection Code (MIX Address Control_Reg)....39
4.6 FIFO Data Formats....40
Table 4-5: Input (A/D) FIFO Data Format....40
Table 4-6: Output (D/A) FIFO Data Format....40

Appendix A: Programming Examples

A.1 Introduction....A-1
A.2 4249.h....A-1
A.3 p4249.c....A-2
A.4 mod.h....A-5
A.5 init.c....A-9
A.6 intr.c....A-10

Appendix B: Intel 82C54 Data Sheet

B.1 Introduction....B-1
B.1.1 Key Features....B-1
B.2 Functional Description....B-1
Figure B-1: 82C54 - Block Diagram....B-2
B.3 Block Diagram....B-2
B.3.1 Data Bus Buffer....B-2
Figure B-2: Block Diagram Showing Data Bus Buffer & Read/Write Logic Functions....B-3
B.3.2 Read/Write Logic....B-3
B.3.3 Control Word Register....B-3
Figure B-3: Block Diagram Showing Control Word Register....B-4
B.3.4 Counter 0, Counter 1, Counter 2....B-4
Figure B-4: Internal Block Diagram of a Counter....B-5
B.4 82C54 System Interface....B-6
Figure B-5: System Interface....B-6
B.5 Operational Description....B-6
B.5.1 Programming the 82C54....B-7
Table B-1: Control Word Format....B-7
B.5.2 Write Operations....B-8
Table B-2: A Few Possible Programming Sequences....B-8

Table of Contents

Page

B.5.3 Read Operations B-9
B.5.3.1 Counter Latch Command B-9
Table B-3: Counter Latching Command Format B-9
B.5.3.2 Read-Back Command B-10
Table B-4: Read-Back Command Format B-11
Table B-5: Status Byte B-12
Table B-6: Null Count Operation B-12
Table B-7: Read-Back Command Example B-13
B.5.4 Mode Definitions B-14
B.5.4.1 Mode 0: Interrupt on Terminal Count B-14
Table B-8: Read/Write Operations Summary B-14
Figure B-6: Mode 0 ..... B-15
B.5.4.2 Mode 1: Hardware Retriggerable One-shot B-16
Figure B-7(a) : Mode 1 ..... B-17
Figure B-7(b) : Mode 2 ..... B-17
B.5.4.3 Mode 2: Rate Generator B-18
B.5.4.4 Mode 3: Square Wave Mode B-18
Figure B-8: Mode 3 ..... B-19
B.5.4.5 Mode 4: Software Triggered Strobe B-20
Figure B-9(a) : Mode 4 ..... B-21
Figure B-9(b) : Mode 5 ..... B-21
B.5.4.6 Mode 5: Hardware Triggered Strobe (Retriggerable) B-21
B.6 Operation Common to All Modes B-22
B.6.1 Programming B-22
B.6.2 GATE B-22
B.6.3 Counter B-23
Table B-9(a): Gate Pin Operations Summary B-23
Table B-9(b): Minimum and Maximum Initial Counts B-23

Chapter 1: General Information

1.1 Introduction

Designed as a complete analog I/O interface, the Model 4249 performs both A/D conversion of analog input signals and D/A conversion for analog output signals. Both A/D and D/A converters exhibit 12-bit resolution with conversion rates as high as 1 MHz. Sampling rates for both the A/D and D/A converters are programmable using the internal sample rate generator. Alternatively, an external TTL sampling clock can be used.

1.2 Anti-Alias Input Filter

The analog input to the A/D converter is preceded by a low-pass analog filter designed to remove all frequency components above 400 kHz, which would otherwise appear as false or aliased artifacts. The 7th-order elliptic filter has a sharp cutoff so that even nearby signal components at 600 kHz are attenuated by over 68 dB. Within the pass-band, ±1 dB flatness is maintained. This filter may be bypassed, if desired, by using on-board jumpers.

1.3 Output Smoothing Filter

A low-pass analog filter is included at the output of the D/A converter. This 400 kHz cutoff smoothing filter removes the higher frequency components inherent in the D/A conversion of discrete samples, producing a smooth analog signal without "steps" at the sample rate. This filter may also be bypassed, if desired, by using on-board jumpers.

1.4 FIFO Buffers for Efficient Block Transfer

A FIFO (First-In, First-Out) buffer is provided at the output of the A/D Converter to accumulate 12-bit digital words and allow efficient block transfers over the MIX bus. A second FIFO receives block transfers from the MIX bus and feeds digital data to the D/A converter at the output sampling rate. Each of these FIFOs is 1024 samples deep in the standard Model 4249. They are expanded to a depth of 16,384 samples in units equipped with Option 003.

1.5 Block Diagram

A block diagram of the Model 4249, 1 MHz, 12-bit, Filtered A/D - D/A Converter MIX Module for VMEbus Systems is presented in Figure 1-1, below.

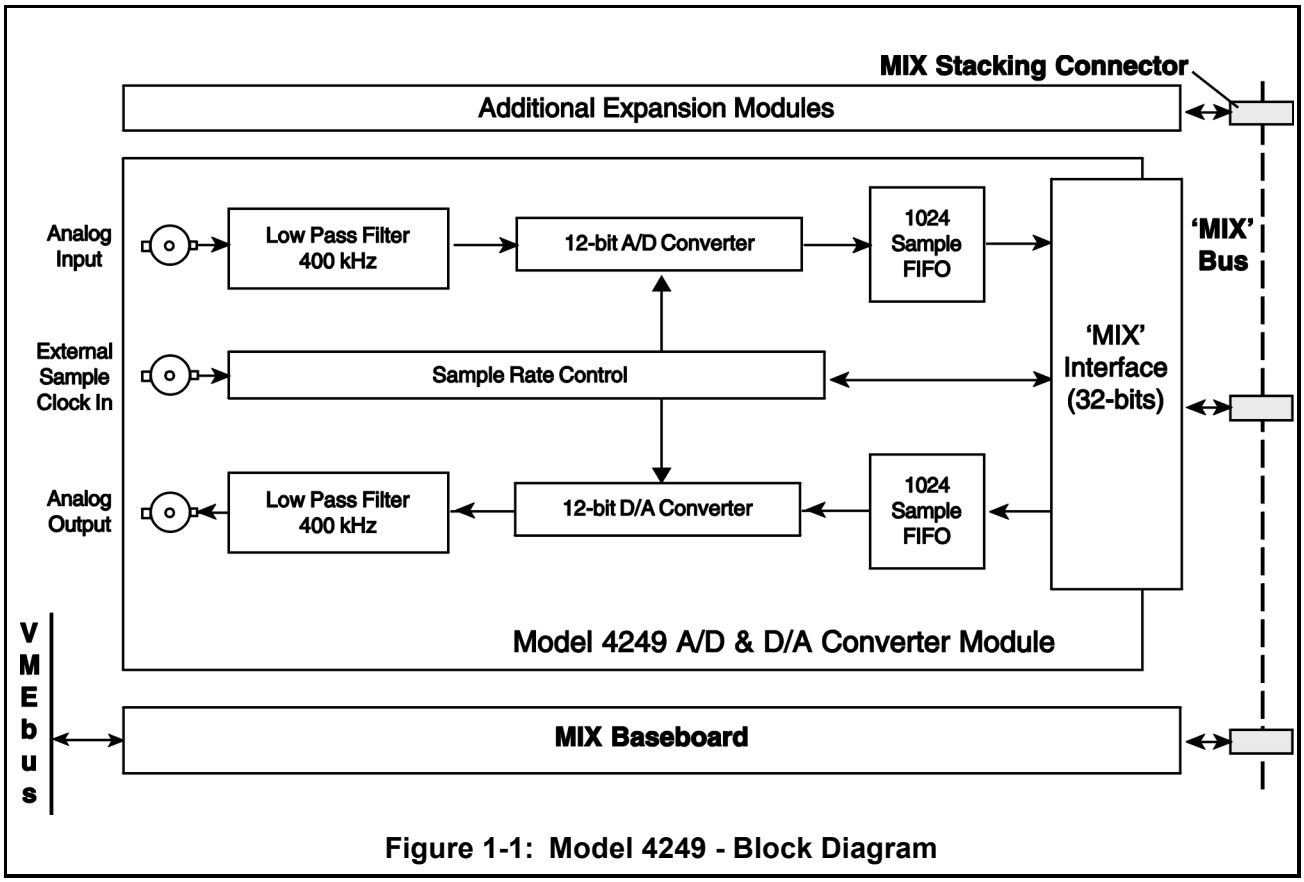


Figure 1-1: Model 4249 - Block Diagram

1.6 Specifications

Inputs:

Number:	1
Type:	Single-ended
Input Impedance:	50 Ω
Full Scale Input Voltage:	±1.00V

Outputs:

Number:	1
Type:	Single-ended
Output Impedance:	50 Ω
Full Scale Output Voltage:	±1.00 V

1.6 **Specifications** (continued)

A/D Converter and S & H:

A/D:	Analog Devices AD9003
Resolution:	12-bits
Conversion Rate:	Up to 1 MSample/sec
Sample & Hold:	Integral
Integral Linearity:	± 0.8 LSB, typical
Differential Linearity:	± 0.5 LSB, typical

D/A Converter:

D/A:	Analog Device AD565A
Resolution:	12-bits
Conversion Rate:	Up to 1 MSample/sec
Differential Linearity:	± 0.5 LSB, typical

A/D and D/A FIFO Data:

Format:	12 bits left justified in 32-bit field
Representation:	2's complement
Coding (32-bits):	+FS: 0x7FF0 zzzz Zero: 0x0000 zzzz - FS: 0x8000 zzzz (z = undriven by A/D, unused by D/A)

A/D FIFO Memory:

Size

Standard:	1,024 samples x 16 bits
Option 003:	16,384 samples x 16 bits

D/A FIFO Memory:

Standard:	1,024 samples x 16 bits
Option 003:	16,384 samples x 16 bits

Sample Rate Generator:

Internal Rate Generator:	152 Hz to 1 MHz, programmable
Internal Rate Programming:	$\text{Rate} = \frac{10\ \text{MHz}}{N}$, where $10 \le N \le 65{,}535$
External Rate:	Front panel TTL input via BNC connector

Input and Output Filters:

Number of Input Filters:	1
Number of Output Filters:	1
Characteristic:	7th Order Elliptic Low Pass
Cutoff Frequency:	400 kHz, fixed
Passband Flatness:	±1 dB
Stop Band Attenuation:	> 68 dB above 600 kHz

1.6 **Specifications** (continued)

Power Requirements:

From MIX Baseboard:	700 mA@ + $5V_{DC}$ typ
	25 mA@ + $12V_{DC}$ typ
	35 mA@ - $12V_{DC}$ typ

Dimensions: 10.3 x 4.0 x 0.80 inches (261.6 x 101.6 x 20.3 mm)

Ordering Information:

Model 4249:	1 MHz, 12-bit, Filtered A/D-D/A with 1k FIFOs
Model 4249-003:	1 MHz, 12-bit, Filtered A/D-D/A with 16k FIFOs
Model 4249-004:	1 MHz, 12-bit, Filtered D/A with 1k FIFO (No A/D)

Chapter 2: Installation and Connections

2.1 Inspection

After unpacking the unit, inspect it carefully for possible damage to connectors or components. If damage is discovered, file a claim with the carrier who transported the unit. The shipping container and packing materials should be saved in case reshipment is required.

2.2 Configuration Jumper Settings

The Model 4249 is comprised of two PC boards. The MIX board, the larger of the two, contains the MIX connector and interface, and the A/D conversion circuitry. The Mezzanine board (the smaller one) contains the D/A conversion circuitry. Configuration jumpers can be found on both boards.

Figure 2-1, on the next page, shows drawings of both boards, and indicates the locations of all user configurable jumpers. Since access to most of these jumper blocks requires that the boards be separated, the first subsection below will cover that topic. Each user-configurable jumper will then be discussed in a subsection of its own, followed by a subsection on the re-assembly of the unit.

CAUTION!!

Perform work in a static controlled environment.

2.2.1 Separating the Mezzanine Board from the MIX Board

a) Place the Model 4249 on a flat surface with the MIX board down and the mezzanine board on top.

b) Remove the three nylon screws in the corners of the mezzanine board.

c) Gently separate the two boards at the black plastic multipin interboard connector (P1) near the rear edge of the mezzanine board.

d) Gently swing the rear edge of the mezzanine board up and over the front panel, being careful not to stress the wires connecting the front edge of the mezzanine board to the connector on the front panel.

Figure 2-1, on the next page, shows the two boards separated in this manner.

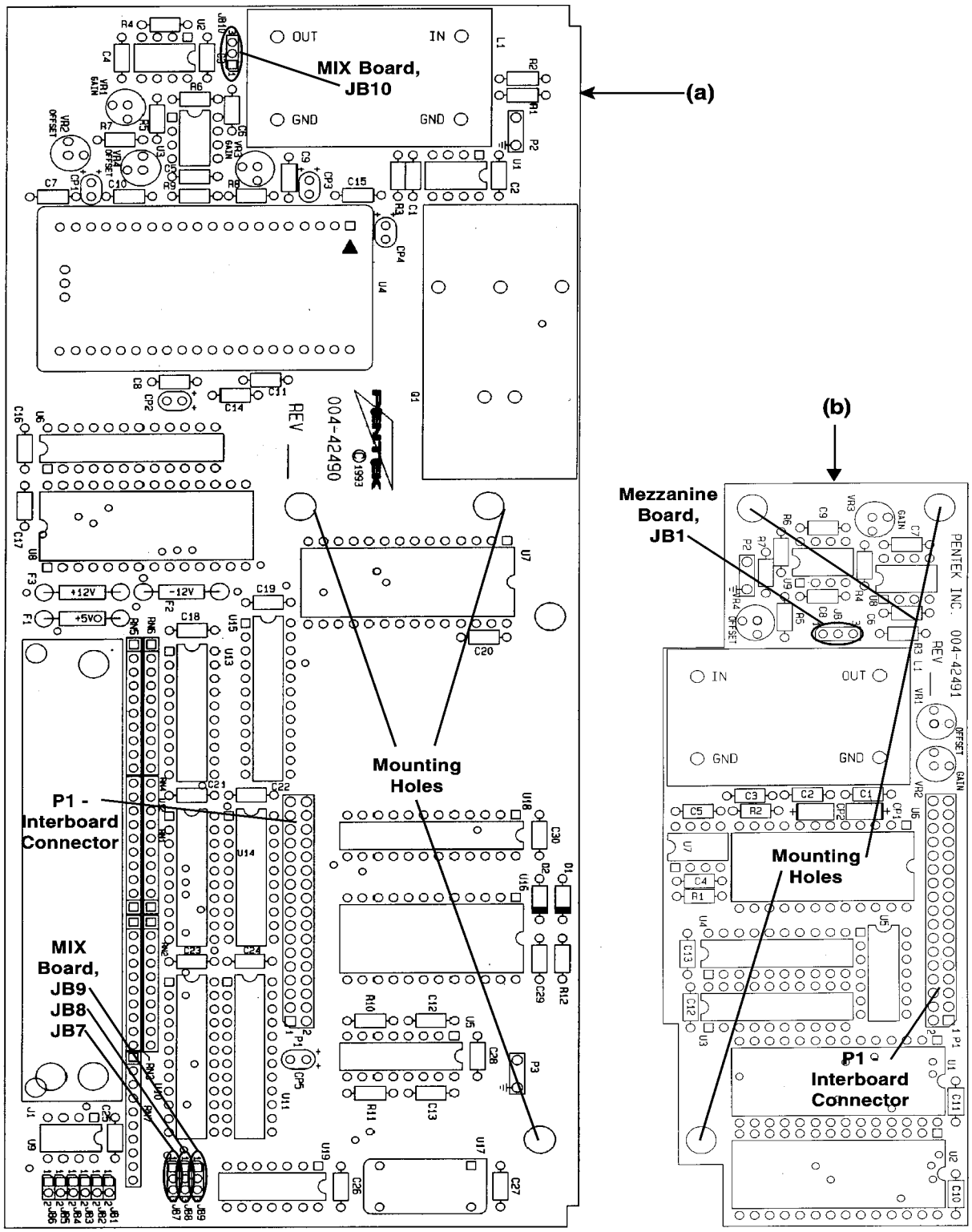


Figure 2-1: Model 4249 - Jumper Blocks and Mounting Holes - (a) MIX Board (b) Mezzanine Board

2.2 Configuration Jumper Settings (continued)

2.2.2 Output Filter Bypass Jumper - JB1, Mezzanine Board

The board is configured at the factory with a shorting jumper installed between pins 2 and 3 of jumper block JB1. In this configuration, the output filter is in the circuit. To remove the filter from the circuit (i. e., to bypass the filter) move this jumper such that it is between pins 1 and 2 of JB1. Figure 2-2, below, shows the settings of this jumper schematically. Table 2-1, below the figure, summarizes the settings.

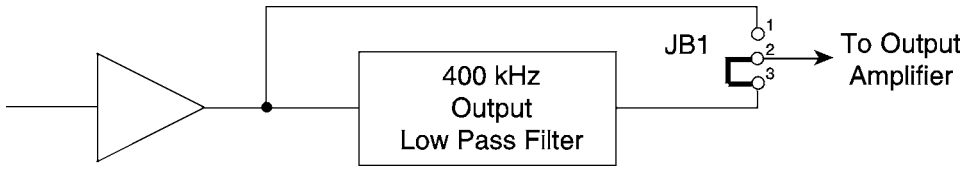


Figure 2-2: Model 4249 - Output Filter Bypass Jumper Diagram

Table 2-1: Model 4249 - Output Filter Bypass Jumper - JB1 - Mezzanine Board

Jumper	Function	1-2 Position	2-3 Position	Default
JB1	Output Filter Bypass	Bypassed	In Circuit	2-3

This is the only configuration jumper block on the mezzanine board. The spacing between terminals of this jumper is 0.1 in. If the jumper should become lost and need replacement, the Pentek part number for it is 356.00000.

2.2.3 Sample Clock Generator Jumpers - JB7, JB8, and JB9, MIX Board

The sampling clocks for the A/D Converter and D/A Converter are configured by jumpers as shown in the schematic diagram at the top of the next page, Figure 2-3. The Counter Timer Circuit (CTC - the Intel 82C54, described in Appendix B of this manual) is used to divide either the 10 MHz signal from the internal crystal oscillator or the External TTL Sample Clock input as determined by the setting of jumper block JB8. Two sections of the CTC are used: CTC #0 is used for generating the A/D sampling clock and CTC #1 is used to generate the D/A sampling clock.

CTC #0 is gated by the Input FIFO Full flag and CTC #1 is gated by the Output FIFO Empty flag. This prevents the generation of sampling clocks when the FIFO's are not capable of responding properly. CTC #2 is not used.

2.2 Configuration Jumper Settings (continued)

2.2.3 Sample Clock Generator Jumpers (continued)

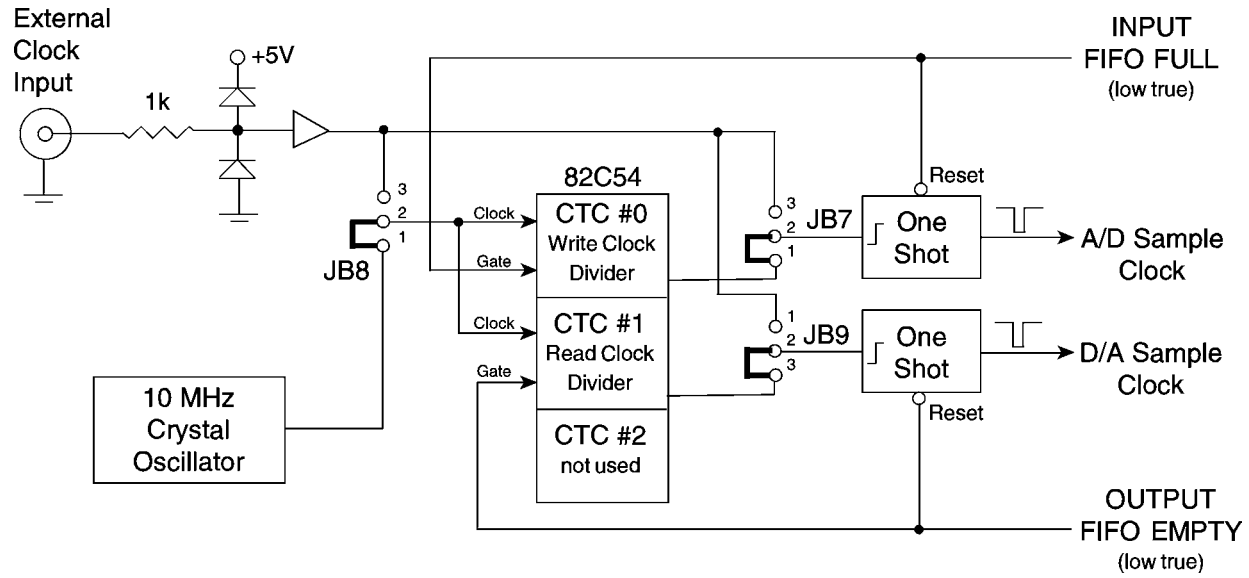


Figure 2-3: Model 4249 - A/D and D/A Converter Sampling Clocks

The A/D sampling clock can be driven by the output of CTC #0 or by the External TTL Sample Clock input, as determined by the setting of jumper block JB7.

The D/A sampling clock can be driven by the output of CTC #1 or by the External TTL Sample Clock input, as determined by the setting of jumper block JB9.

The A/D and D/A sampling clocks are each conditioned by a one-shot circuit, triggered by a positive edge at the input. The one-shots are reset by the same FIFO flags used to gate the CTC inputs.

The settings for these three jumpers are summarized in Table 2-2, below. The spacing between terminals for these jumpers is 2mm. If the jumper should become lost and need replacement, the Pentek part number for it is 356.00010.

Table 2-2: Model 4249 - Sample Clock Jumpers - JB7, JB8, JB9 - MIX Board

Jumper	Function	1-2 Position	2-3 Position	Default
JB7	A/D Sample Clk	CTC #0 Output	Ext Sample Clock In	1-2
JB8	CTC #0,1 input	10 MHz Osc	Ext Sample Clock In	1-2
JB9	D/A Sample Clk	Ext Sample Clock In	CTC #1 Output	2-3

2.2 Configuration Jumper Settings (continued)

2.2.4 Input Filter Bypass Jumper - JB10, MIX Board

The 400 kHz Input Filter may be configured to be in the input signal path or not, depending on the setting at jumper block JB10. Figure 2-4, below, shows the two positions for this jumper schematically. The board is configured at the factory with the filter in the circuit (jumper installed between pins 2 and 3 of JB10). To bypass the input filter, move the jumper such that it is between pins 1 and 2 of JB10.

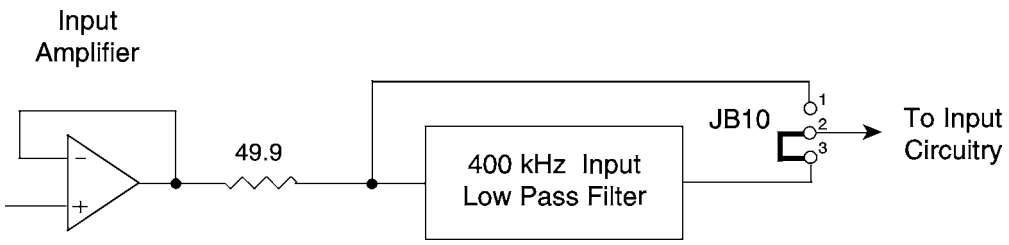


Figure 2-4: Model 4249 - Input Filter Bypass Jumpers Diagram

The spacing between terminals for these jumpers is 2 mm. If the jumper should become lost and need replacement, the Pentek part number for it is 356.00010. Note that this jumper block is accessible without separating the boards. Table 2-3, below, summarizes the settings of this jumper.

Table 2-3: Model 4249 - Output Filter Bypass Jumper - JB10 - MIX Board

Jumper	Function	1-2 Position	2-3 Position	Default
JB10	Input Filter Bypass	Bypassed	In Circuit	2-3

2.2.5 Other Jumper Blocks

Jumper blocks that are not discussed in the sections above are present on the board for factory testing and configuration purposes, and should not be tampered with by anyone but factory-trained, Pentek personnel. If you suspect that some of these jumper blocks have been re-configured, please contact Pentek at the phone number shown on the title page of this manual for assistance.

2.2 Configuration Jumper Settings (continued)

2.2.6 Reconnecting the Mezzanine Board to the MIX Board

a) **Gently** swing the mezzanine board up and then down onto the MIX board aligning the multi-pin male header with the socket on the MIX board. **Be careful not to shift the connector pins in the socket. Be careful not to stress the wires connecting the front panel BNC connectors.**

b) **Gently** push the mezzanine board down into the multi-pin connector until it is fully seated and resting on the three standoffs between the boards.

c) Install the three nylon 4-40 x 1/4" Philips head screws removed earlier and tighten them in rotation.

2.3 The Model 4249 Front Panel

A drawing of the Model 4249's front panel is shown in Figure 2-5, at the right. The panel features BNC connectors for the analog input signal to the A/D Converter, the analog signal out of the D/A converter, and an external sample clock connection. The characteristics of the signals to be applied to these connectors will be discussed in the subsections below.

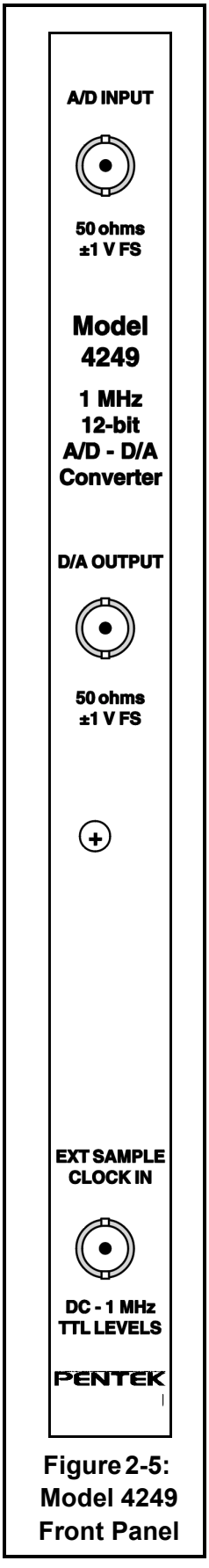


Figure 2-5: Model 4249 Front Panel

2.3.1 A/D Input Signal Connections

The A/D input circuitry is shown in Figure 2-6, at the top of the next page. This input signal is applied to the front panel BNC connector marked A/D INPUT. The input impedance is 50 Ω and the full scale input voltage is ± 1.00 Volts (2 V_{P-P}).

2.3.2 D/A Output Signal Connections

The D/A Output Circuitry is shown in Figure 2-7, on the next page. This output signal is available at the front panel BNC connector marked D/A OUTPUT. The output impedance is 50 Ω and the full scale output voltage is ± 1.00 Volts (2 V_{P-P}).

2.3 The Model 4249 Front Panel (continued)

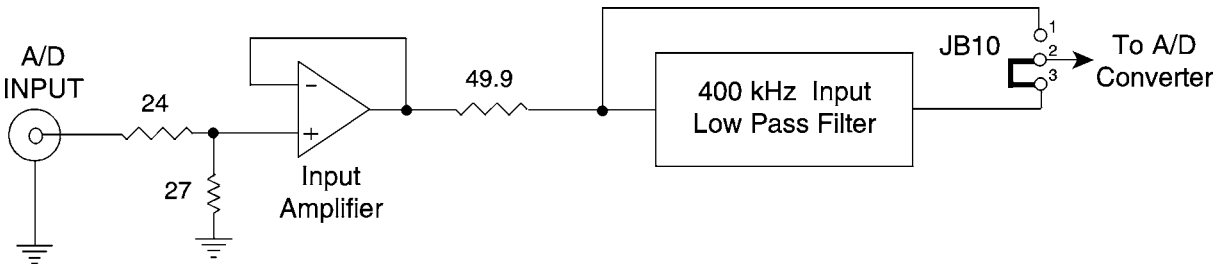


Figure 2-6: Model 4249 - A/D Input Circuitry

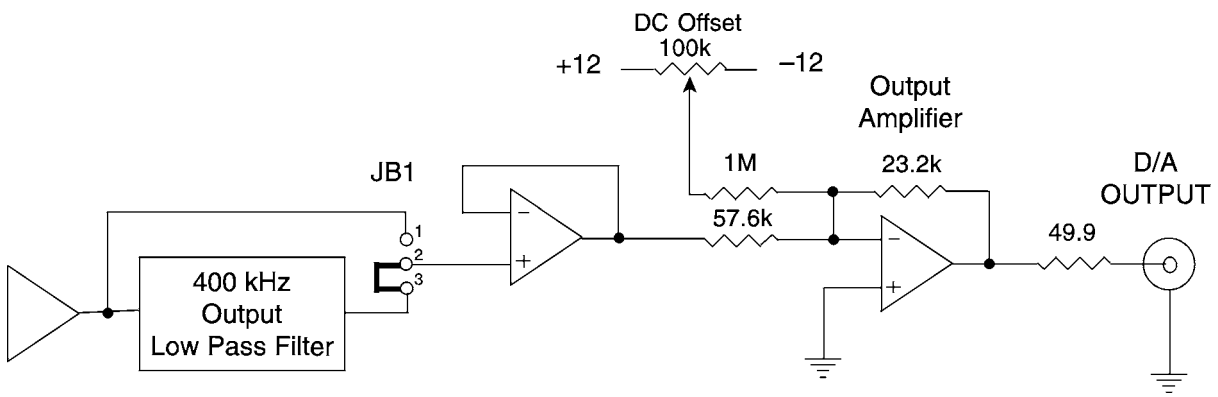


Figure 2-7: Model 4249 - D/A Output Circuitry

2.3.3 External Sample Clock In

The front panel BNC connector labeled EXT SAMPLE CLOCK IN accepts a TTL clock for use as the input to clock dividers CTC #0 and CTC #1, or as the sample clock directly (i. e., undivided). The circuitry involved with the external sample clock is shown in Figure 2-3, in Section 2.2.3.

The voltage limits for the signal applied here are standard TTL levels, where a logic '0' is between 0.0 and +0.8 V_{DC}, and a logic '1' is between +2.0 V_{DC} and +5.0 V_{DC}. The input frequency range is from DC to 10 MHz when the signal is used as an input to the CTC, and the range is DC to 1 MHz when the input is used directly as the sample clock for either the A/D or D/A converter. The duty cycle is not critical, but the logic '0' and logic '1' intervals should both be at least 40 nsec.

2.3 The Model 4249 Front Panel (continued)

2.3.3 External Sample Clock In (continued)

The active edge of the EXT SAMPLE CLOCK IN is the low-to-high transition.

If the cable connecting the External Sample Clock is longer than a few feet, it is recommended that the clock source be capable of driving a 50 Ω load, and that a 50 Ω termination resistor be placed at the front panel BNC. This will help to avoid overshoot and ringing. Notice that a diode clamping network restricts the excursion of the input voltage to a 0 to 5 V_{DC} range to help protect damage from over-voltage conditions.

2.4 Installing the Model 4249 on the MIX Baseboard

See the MIX Tutorial manual (P/N 800.80001) for MIX module stack assembly instructions.

2.5 Installing the Assembly into the VMEbus Card Cage

The MIX Tutorial manual also covers insertion of the assembled module stack in the card cage.

Chapter 3: Memory Maps and Register Descriptions

3.1 Programming Overview

The following sections cover access to the Model 4249 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, 4284 and 4285. General information on the use of these Baseboards is given in Mix Tutorial Manual. The Mix Tutorial Manual also provides information on access to MIX modules from Baseboards that use an Address Page Register, such as Pentek's Model 4283. A Memory Map for use with the Model 4283 is also included in this chapter.

3.2 Address Maps

The Model 4249 operates only as a MIX bus slave. The sections below will present address maps for the resources available on the Model 4249, for each type of MIX Baseboard available from Pentek. Each of the slave resources will be assigned a symbolic name, by which that resource will be referred to in subsequent sections of the manual. There are five 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 4249 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 4249, skip to Section 3.2.3, and use the address map found there. If your Model 4249 will be mounted on a Model 4284 Baseboard, use the address map in Section 3.2.4. Finally, if the Model 4285 is the baseboard that will control your MIX modules, refer to the address map in Section 3.2.5.

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 possible conflicts, 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.

3.2 Address Maps (continued)

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.

3.2.1 Model 4249 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 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 4249's slave resources in subsequent sections of this manual.

Table 3-1: Model 4249 - Memory Map for Use with Models 4200, 4200A, 4201 and 4201A MIX Baseboards - 68030 View

4200/01 68030 Address	Symbolic Name	Resource
MIX_base+0x0800 0000	AD_FIFO	A/D FIFO (Read Only)
MIX_base+0x0800 0000	DA_FIFO	D/A FIFO (Write Only)
MIX_base+0x0800 0004	Control_Reg	Control Register
MIX_base+0x0800 0008	Status_Reg	Status Register (Read Only)
MIX_base+0x0800 0010	CT_Ctl_Reg	Counter Timer Control Register
MIX_base+0x0800 0014	CT_2	Counter Timer #2 (unused)
MIX_base+0x0800 0018	CT_1	Counter Timer #1 (D/A Clock)
MIX_base+0x0800 001C	CT_0	Counter Timer #0 (A/D Clock)

MIX base Address:
If Model 4249 is in MIX position 0, MIX_base = 0xC000 0000
If Model 4249 is in MIX position 1, MIX_base = 0xD000 0000
If Model 4249 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 other VMEbus Masters. For further information about MIX Bus access on these Baseboards in this manner, please refer to the MIX Tutorial manual, or to the Operating Manual of the Baseboard (UMBM).

3.2 Address Maps (continued)

3.2.2 Model 4249 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 slave memory map for a Model 4249 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 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 4249 - Memory Map for Use with the Model 4202 MIX Baseboard

*VMEbus Address	Symbolic Name	Resource
MIX_base+0x0800 0000	AD_FIFO	A/D FIFO (Read Only)
MIX_base+0x0800 0000	DA_FIFO	D/A FIFO (Write Only)
MIX_base+0x0800 0004	Control_Reg	Control Register
MIX_base+0x0800 0008	Status_Reg	Status Register (Read Only)
MIX_base+0x0800 0010	CT_Ctl_Reg	Counter Timer Control Register
MIX_base+0x0800 0014	CT_2	Counter Timer #2 (unused)
MIX_base+0x0800 0018	CT_1	Counter Timer #1 (D/A Clock)
MIX_base+0x0800 001C	CT_0	Counter Timer #0 (A/D Clock)

Default MIX base Address:

If Model 4249 is in MIX position 0, MIX_base = 0x0000 0000 for A32, or 0x00 0000 for A24
If Model 4249 is in MIX position 1, MIX_base = 0x1000 0000 for A32, or 0x10 0000 for A24
If Model 4249 is in MIX position 2, MIX_base = 0x2000 0000 for A32, or 0x20 0000 for A24
* - in A24 Space, access at 0x08 00zz

3.2 Address Maps (continued)

3.2.3 Model 4249 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 4249 - 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	MXA27	1	1	MXA21	x	0x03 8000
Stack Position 1	x		1	0		x	0x03 0000
Stack Position 2	x		0	1		x	0x02 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. For access to the Model 4249, the MX_A27 bit (i.e., D17 in the Page Register) must always be set to the logic '1' state. This is indicated in the Example column of Table 3-3.

3.2 Address Maps (continued)

3.2.3 Model 4249 Address Map for the Model 4283 Baseboard (continued)

3.2.3.2 Register Selection Within the Module

Once the module has been selected, all the slave resources (including the RAM Banks) of the Model 4249 appear in the TMS320C30 address map according to Table 3-4, below. 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.

Table 3-4: Model 4249 - Memory Map for Use with the Model 4283 MIX Baseboard

†4283 'C30 Address	Symbolic Name	Resource
0x80 4000	AD_FIFO	A/D FIFO (Read Only)
0x80 4000	DA_FIFO	D/A FIFO (Write Only)
0x80 4001	Control_Reg	Control Register
0x80 4002	Status_Reg	Status Register (Read Only)
0x80 4004	CT_Ctl_Reg	Counter Timer Control Register
0x80 4005	CT_2	Counter Timer #2 (unused)
0x80 4006	CT_1	Counter Timer #1 (D/A Clock)
0x80 4007	CT_0	Counter Timer #0 (A/D Clock)
† - D17 in Page Register = 1		

3.2 Address Maps (continued)

3.2.4 Model 4249 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, below, presents the memory map for the slave resources on the Model 4249, as seen by the 'C40 processor on the Model 4284 Baseboard.

Table 3-5: Model 4249 - Memory Map for Use with the Model 4284 MIX Baseboard

4284 'C40 Address	Symbolic Name	Resource
MIX_base+0x0200 0000	AD_FIFO	A/D FIFO (Read Only)
MIX_base+0x0200 0000	DA_FIFO	D/A FIFO (Write Only)
MIX_base+0x0200 0001	Control_Reg	Control Register
MIX_base+0x0200 0002	Status_Reg	Status Register (Read Only)
MIX_base+0x0200 0004	CT_Ctl_Reg	Counter Timer Control Register
MIX_base+0x0200 0005	CT_2	Counter Timer #2 (unused)
MIX_base+0x0200 0006	CT_1	Counter Timer #1 (D/A Clock)
MIX_base+0x0200 0007	CT_0	Counter Timer #0 (A/D Clock)

MIX base Address:
If Model 4249 is in MIX position 0, MIX_base = 0x2000 0000
If Model 4249 is in MIX position 1, MIX_base = 0x2400 0000
If Model 4249 is in MIX position 2, MIX_base = 0x2800 0000

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 the MIX Tutorial manual, the Model 4284 Operating Manual, or the UMBM's Operating Manual.

3.2 Address Maps (continued)

3.2.5 Model 4249 Address Map for the Model 4285 Baseboard

When the Model 4285 is equipped with one of the MIX Expansion Bus Adapters (Options 031 or 036) all eight of its 'C40 processors can access modules on the MIX bus via the 4285's Main Bus. Additionally, four of the processors (A, B, E and F) have a separate access path to devices on the MIX bus via their Global I/O expansion ports. Separate control registers are provided for each of these five interfaces to the MIX bus. Two addressing modes, Normal and Compressed, are available for any of these access paths.

The control registers mentioned above each provide a 5-bit address page field. For this Baseboard, however, the page setting is NOT used to determine which MIX module is being addressed (i. e., the direct addressing method is used). The mapping of the page field onto the address bus is a function of the selected addressing mode. For complete details, refer to the Operating Manual for the Model 4285. For purposes of this operation summary, we will say that in Normal Addressing Mode, page bit 2 is used to assert the MXA27 address bit, while in Compressed Mode, MXA27 is asserted by 'C40 Global Address bit GA20.

Table 3-6, below, presents the memory map for the slave resources on the Model 4249, as seen by the 'C40 processors on the Model 4285 Baseboard. The table also provides the MIX_Base addresses, in the memory spaces of the 'C40 processors, for the Expansion bus and Global I/O access paths. For further information on the use of the 4285 as a MIX baseboard, including VME access to MIX resources on the 4285, please refer to Section 3.4 of the 4285 Operating Manual

Table 3-6: Model 4249 - Memory Map for Use with the Model 4285 MIX Baseboard

4285 'C40 Address Compressed Mode	4285 'C40 Address Uncompressed Mode	Symbolic Name	Resource
MIX_base+0x0010 0000	MIX_base+0x0000 0000	AD_FIFO	A/D FIFO (Read Only)
MIX_base+0x0010 0000	MIX_base+0x0000 0000	DA_FIFO	D/A FIFO (Write Only)
MIX_base+0x0010 0001	MIX_base+0x0000 0001	Control_Reg	Control Register
MIX_base+0x0010 0002	MIX_base+0x0000 0002	Status_Reg	Status Register (Read Only)
MIX_base+0x0010 0004	MIX_base+0x0000 0004	CT_Ctl_Reg	Counter Timer Control Register
MIX_base+0x0010 0005	MIX_base+0x0000 0005	CT_2	Counter Timer #2 (unused)
MIX_base+0x0010 0006	MIX_base+0x0000 0006	CT_1	Counter Timer #1 (D/A Clock)
MIX_base+0x0010 0007	MIX_base+0x0000 0007	CT_0	Counter Timer #0 (A/D Clock)
MIX_base Address for Expansion Bus:		**MIX_base Address for Global I/O Bus**:	
If Model 4249 is in MIX position 0, MIX_base = 0x9000 0000		If Model 4249 is in MIX position 0, MIX_base = 0xC800 0000	
If Model 4249 is in MIX position 1, MIX_base = 0x9020 0000		If Model 4249 is in MIX position 1, MIX_base = 0xC820 0000	
If Model 4249 is in MIX position 2, MIX_base = 0x9040 0000		If Model 4249 is in MIX position 2, MIX_base = 0xC840 0000	
Note: MXA27 is asserted via page bit 2 in Normal addressing mode, or via 'C40 address bit GA20 in Compressed addressing mode.			

3.3 Control Register - R/W @ Control_Reg

The Control Register provides a mechanism by which the user selects interrupt sources and controls the FIFO's. Table 3-7, below, shows the function of each bit. All bits in this register are cleared to the logic '0' state at power up.

Table 3-7: Model 4249 - Control Register - MIX Address Control_Reg - Read/Write

Bit #	D31	D30	D29 D28 D27 D26	D25	D24	D23 - D0
Function	Overload Mask	DMA Select (not used)	FIFO Interrupt Mask Selection	Output FIFO Reset (D/A)	Input FIFO Reset (A/D)	not used
Bit = 1	Enabled	n/a		Run	Run	
Bit = 0	Disabled	n/a	(See Section 3.3.3)	Reset	Reset	

The Control Register is a read/write device that uses only the eight Most Significant Bits (MSBs) of the 32-bit word written by the MIX Baseboard. When writing to the Control Register, data in bit positions D23 through D0 will have no effect.

When reading from address Control_Reg, bits D31 through D24 will reflect the contents of the Control Register, and bits D23 through D0 will be indeterminate and should be ignored.

3.3.1 Overload Interrupt Mask - Control_Reg, Bit D31

This bit enables the generation of an interrupt to the MIX Baseboard when an input overload condition is detected. The overload condition is defined by the digital output codes from the A/D converter of 0x8000 (for minus full scale), and 0x7FF0 (for plus full scale).

3.3.2 DMA Request Control - Control_Reg, Bit D30

This function is not supported, and this bit can be ignored.

3.3.3 FIFO Interrupt Selection Code - Control_Reg, Bits D26 thru D29

The four FIFO Interrupt Control bits determine which FIFO conditions will cause the Model 4249 to send an interrupt to the MIX Baseboard. Each 4-bit code pattern selects a different combination of interrupt conditions, as shown in Table 3-8, at the top of the next page.

For codes 0101 through 1000, interrupts will be generated if the condition in **either** column is met.

3.3 Control Register (continued)

3.3.3 FIFO Interrupt Selection Code (continued)

Table 3-8: Model 4249 - FIFO Interrupt Selection Code (Part of Control Register)

Control Register Bit #				Output FIFO (D/A) Interrupt Conditions	Input FIFO (A/D) Interrupt Conditions
D29	D28	D27	D26		
0	0	0	0	None	None
0	0	0	1	Output FIFO NOT Half Full	None
0	0	1	0	Output FIFO Empty	None
0	0	1	1	None	Input FIFO NOT Empty
0	1	0	0	None	Input FIFO Half Full
0	1	0	1	Output FIFO NOT Half Full	Input FIFO Full
0	1	1	0	Output FIFO NOT Half Full	Input FIFO Half Full
0	1	1	1	Output FIFO Empty	Input FIFO NOT Empty
1	0	0	0	Output FIFO Empty	Input FIFO Half Full

These functions are useful in synchronizing data collection and generation with DSP tasks and in minimizing processor overhead in servicing data input and output operations.

For example, if the Input FIFO is configured to send an interrupt to the MIX Baseboard when it becomes HALF-FULL, the Baseboard processor can initiate a block transfer of 512 points out of the Input FIFO. This is much more efficient than waiting for each sample to become available by testing for the presence of data by polling.

To disable all interrupts, clear all four control bits (D29 - D26) to the logic '0' state. This is the default power-up condition.

3.3.4 Output FIFO Reset - Control_Reg, Bit D25

This bit controls the $\overline{\text{RESET}}$ pin of the output (D/A) FIFO. When this bit is active (i. e., cleared to the logic '0' state), the FIFO will not accept or deliver any data. Any data that may have been present prior to the $\overline{\text{RESET}}$ event is cleared and lost. The FULL and HALF-FULL flags are set false and the EMPTY flag is set true. A function implementing a reset cycle for the output FIFO should first clear this bit to the logic '0' state, then set it to the logic '1' state to allow the collection of data for the D/A Converter.

3.3 Control Register (continued)

3.3.5 Input FIFO Reset - Control_Reg, Bit D24

This bit controls the $\overline{\text{RESET}}$ pin of the input (A/D) FIFO. When this bit is active (i. e., cleared to the logic '0' state), the FIFO will not accept or deliver any data. Any data that may have been present prior to the $\overline{\text{RESET}}$ event is cleared and lost. The FULL and HALF-FULL flags are set false and the EMPTY flag is set true. A function implementing a reset cycle for the input FIFO should first clear this bit to the logic '0' state, then set it to the logic '1' state to allow the collection of data from the A/D Converter.

3.4 Status Register - R/O @ Status_Reg

The Status Register allows the user to poll the Model 4249 to check the conditions of the FIFOs and digital overload detector status bits. Table 3-9, below, shows the bit definition for each status bit.

Table 3-9: Model 4249 - Status Register - MIX Address Status_Reg - Read Only

Bit #	D31-D24	D23	D22	D21	D20	D19	D18	D17	D16	D15-D0
Function	not used	Digital Input Overload	Interrupt Pending	Output FIFO Empty Flag	Output FIFO Half-Full Flag	Output FIFO Full Flag	Input FIFO Empty Flag	Input FIFO Half-Full Flag	Input FIFO Full Flag	not used
Bit = 1		False	False	False	False	False	False	False	False	
Bit = 0		True	True	True	True	True	True	True	True	

When reading the Status Register (at MIX address Status_Reg) only bits D23 through D16 correspond to the Status Register information. Bits D31 through D24, and bits D15 through D0, are indeterminate and their value should be ignored.

When the Status Register is read, any pending interrupt is cleared and the Input Overload bit is reset.

3.4.1 Input Overload Status - Status_Reg, Bit D23

When a digital full scale signal is detected in the A/D channel, the Input Overload status bit is latched in the active state (logic '0') so that it can be preserved for a status read.

3.4 Status Register (continued)

3.4.1 Input Overload Status (continued)

After the Status Register is read, the latched Input Overload bit is reset (to logic '1'). If the Status Register is immediately read again, and the overload bit is still active, this indicates that the overload event is still present. This may be due to a very large signal or DC offset at the input connector. Otherwise, the overload event was due to a transient.

The Overload Interrupt Control bit in the Control Register enables this condition to cause an interrupt to the MIX Baseboard (see Section 3.3.1).

3.4.2 Interrupt Pending - Status_Reg, Bit D22

This bit is set true (logic '0') when an interrupt is pending due to any of the interrupt sources on the Model 4249. It monitors the state of the $\overline{\text{MXINT}}$ line of the MIX bus and thus the input to one of the three MIX interrupt flip-flops on the MIX Baseboard.

When this line is at logic '1', no interrupt is being generated by the Model 4249.

3.4.3 Input and Output FIFO Flags - Status_Reg, Bits D21 thru D16

These six flags are driven directly from the Input and Output FIFO status lines and reflect the current condition of both FIFOs. These bits are **not** latched or cleared upon a status register read, as with the Input Overload bit.

The FIFO bits can be directed to cause an interrupt to the Model 4283 as described in Section 3.3.3.

3.5 Counter Timer Registers

The Model 4249 employs an Intel 82C54 triple-channel counter/timer chip (CTC) to generate the A/D conversion clock and the D/A conversion clock. Each of the three CTC channels acts as a 16-bit programmable clock divider and each can be programmed independently from the MIX Baseboard.

The input clock for each channel is jumper selectable as either a 10 MHz oscillator signal generated on-card, or an external TTL clock supplied to the EXT SAMPLE CLOCK input connector on the front panel. The active edge of the EXT SAMPLE CLOCK signal is the low-to-high transition. See Section 2.2.3 for details on jumper configurations and for a schematic drawing of the clock system.

3.5 Counter Timer Registers (continued)

Programming the CTC dividers is accomplished by first writing to the Counter/Timer Control Register, at symbolic address CT_Ctl_Reg (to set up the operating mode for each channel), and then writing to each of the CT divider registers (symbolic addresses CT_0 and CT_1). Physical addressing for these registers in a given system is dependent upon the MIX Baseboard in use, and can be found in Table 3-1, Table 3-2, Table 3-4 or Table 3-5.

The CTC is an 8-bit device and all **reads and writes to it by the MIX Baseboard utilize bits D23 through D16 only of the 32-bit MIX data word.** Bit D23 is the most significant bit of the 8-bit CTC byte. The least significant 16 bits and the most significant 8 bits of the MIX data word are ignored for writes and are indeterminate for reads.

For a complete description of the Intel 82C54, refer to Appendix B of this manual. For purposes of this discussion, we will assume the CTC is to be operated in Counter Mode 2 (rate generator mode), with a single 8-bit divisor for each section. This produces a square wave output whose frequency is equal to the input clock rate divided by N, where N is the 8-bit divisor value programmed for each channel.

3.5.1 Programming the A/D Sample Clock - R/W @ CT_0

The A/D converter sampling clock is produced by CTC channel 0. Remember that the A/D sample clock may also be driven directly by the EXT SAMPLE CLOCK input signal by setting the appropriate jumpers (see Section 2.2.3). If this is the case, CTC channel 0 is not used and can be left unprogrammed.

The maximum frequency for the A/D sampling clock is 1 MHz. There is no lower limit.

Suppose the CTC is to be used for the A/D sample clock, and a sampling rate of 500 kHz is required. In this case, we will use the internal 10 MHz oscillator as the CTC input (set MIX board jumpers JB7 - 1,2 and JB8 - 1,2). The divisor (N) is calculated as follows:

$$N = \frac{F_{IN}}{F_{SAMPLE}} = \frac{10\ MHz}{500\ kHz} = 20_{10} = 14_{16}$$

The assembly language pseudocode for programming this frequency from the 'C30 processor on the Model 4283 MIX Baseboard is shown at the top of the next page.

3.5 Counter Timer Registers (continued)

```
CTCR equ    804004H        ; define CTC control register address
CT_0 equ    804007H        ; define CTC #0 divisor address
     mov    CTCR, 140000H  ; setup CTC #0 for mode 2, 8-bit div.
     mov    CT_0, 140000H  ; set divisor to 0x14
```

Note that the bottom 24 bits of the data words are set to zero and only the D23 - D16 bits of the 32-bit words have any significance. Once the control register has been initialized, it need not be set for subsequent divisors. For divisors greater than N = 255, the 16-bit divisor mode must be used and a different control register code is required. Details are given in the Intel data sheet in Appendix B.

3.5.2 Programming the D/A Sample Clock - R/W @ CT_1

The D/A converter sampling clock is controlled in an identical manner to that of the A/D sampling clock except that CTC channel 1 is used. The input clock to the CTC can be either the 10 MHz oscillator signal generated on-card, or the EXT SAMPLE CLOCK signal. The D/A clock may also be connected directly to the EXT SAMPLE CLOCK signal. See Section 2.2.3 for a complete description of jumper configurations.

The maximum D/A sampling clock is 1 MHz and there is no lower limit.

Suppose we have an external 4 MHz TTL reference clock and wish to derive a 1 MHz D/A sample clock from it. In this case, the CTC Channel 1 clock jumper must be set to use the external clock (set MIX board jumpers JB8 - 2,3 and JB9 - 2,3).

Channel 1 of the CTC must be set for a divisor of 4. The assembly language pseudocode for programming this frequency from the 'C30 processor on the Model 4283 MIX Baseboard is shown below.

```
CTCR equ    804004H        ; define CTC control register address
CT_1 equ    804006H        ; define CTC #1 divisor address
     mov    CTCR, 540000H  ; setup CTC #1 for mode 2, 8-bit div.
     mov    CT_1, 040000H  ; set divisor to 4
```

3.6 Input (A/D) FIFO - R/O @ AD_FIFO

The Input FIFO is a 16-bit wide, 1,024-word deep memory device that stores output samples from the A/D converter. Units equipped with Option 003 extend the depth of the FIFO to 16,384 words. The 12-bit A/D output is left justified in the 16-bit FIFO word, with the four least significant bits filled with 0's.

The FIFO is addressed by the MIX Baseboard as a single 32-bit word at the symbolic memory location AD_FIFO. (For the baseboard-dependent, physical address of this resource, please refer to the table appropriate to the baseboard used in your application in Section 3.2.) The 16-bit FIFO output is left justified in the 32-bit word read by the Baseboard. The bottom 16-bits of this word are undriven and should be discarded.

Every time data is read from this location, the FIFO automatically fetches the next sample. When the FIFO is empty, the read cycle on the MIX bus will be suspended until FIFO output data is available. Three status flags indicate the FIFO condition: EMPTY, HALF-FULL, and FULL. These flags are available as bits in the Status Register (see Section 3.4). The data format for the FIFO is shown in Table 3-10, below.

Table 3-10: Model 4249 - Input (A/D) FIFO Data Format - MIX Address AD_FIFO - Read Only

Bit #	D31 D30 - - - - - D21 D20	D19 D18 D17 D16	D15 - - - - - - - - - - D0
Function	MSB - - - 12-bit A/D Out - - - LSB	- - - Zeroes - - -	- - - undriven - - -

This FIFO, and the A/D converter that feeds the data to it, are not included in the Model 4249 option 004.

3.7 Output (D/A) FIFO - W/O @ DA_FIFO

The Output FIFO is a 16-bit wide, 1024-word deep memory device that stores samples to be sent into the D/A converter. Units equipped with Option 003 extend the depth of the FIFO to 16,384 words. The 12-bit D/A input is left justified in the 32-bit MIX data word, with the 20 LSB's unused.

The FIFO is addressed by the MIX Baseboard as a single memory location at symbolic address DA_FIFO. (For the baseboard-dependent, physical address of this resource, please refer to the table appropriate to the baseboard used in your application in Section 3.2.) Every time data is written to this location, the FIFO automatically moves it into an output queue waiting for delivery to the D/A converter. When the FIFO is full, the write cycle on the MIX bus will be suspended until the FIFO can accept a new entry. Three status flags indicate the FIFO condition: EMPTY, HALF-FULL, and FULL. These flags are available as bits in the Status Register (see Section 3.4). The data format for the output FIFO is shown in Table 3-11, below.

Table 3-11: Model 4249 - Output (D/A) FIFO Data Format - MIX Address DA_FIFO - Write Only

Bit #	D31 D30 D29 - - - - - D22 D21 D20	D19 - - - - - - - - - - - - - - - - - D0
Function	MSB - - - - - - 12-bit A/D Output - - - - - - LSB	- - - - - - - Unused - - - - - - -

This page is intentionally blank

Chapter 4: Operating Procedures

4.1 Introduction

This chapter draws upon the information presented earlier in this manual to provide the user with procedures to follow when using the Model 4249. The information presented here assumes that the configuration jumpers on your 4249 have been set according to the requirements of your application (see Chapter 2 for details), and that you have already installed your MIX modules on the Baseboard of your choice (MIX stack assembly instructions are presented in Appendix X). Detailed descriptions of the registers referred to in the sections below were given in Chapter 3.

4.2 Selecting the A/D and D/A Sampling Clock Sources

The Data Conversion circuits in the Model 4249 may receive their clock signals directly from a signal applied to the Front Panel EXT SAMPLE CLOCK IN BNC connector, from frequency division of such a signal, or from frequency division of the internal 10 MHz crystal oscillator. There are three jumper blocks in this unit that are involved in the selection of the clock source.

One of these is JB7, which selects either the EXT SAMPLE CLOCK IN signal or the output of channel 0 of the CTC (Counter/Timer Chip - Intel 82C54, see Appendix B) as the ADC sample clock. Another is JB9, which selects either the EXT SAMPLE CLOCK IN signal or the output of channel 1 of the CTC as the DAC sample clock. The last is JB8, which selects either the EXT SAMPLE CLOCK IN signal or the on-board, 10 MHz crystal oscillator as the clock source for the CTC. The information above is presented schematically in Figure 4-1, below, and summarized in Table 4-1, at the top of the next page.

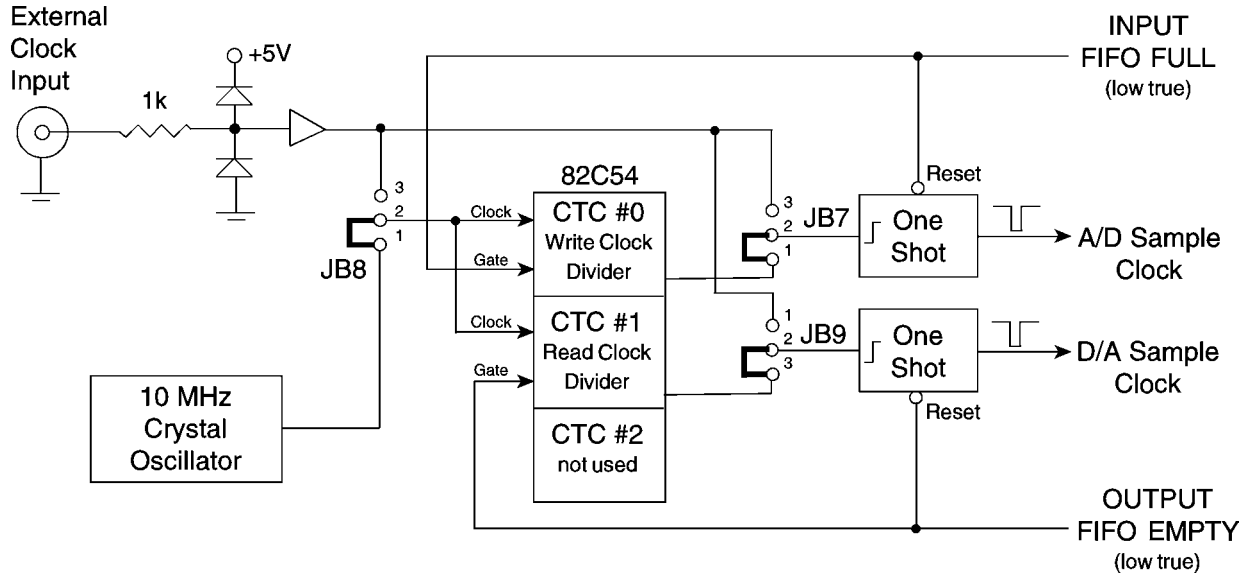


Figure 4-1: Model 4249 - A/D and D/A Converter Sampling Clocks

4.2 Selecting the A/D and D/A Sampling Clock Sources (continued)

Table 4-1: Model 4249 - Sample Clock Jumpers - JB7, JB8, JB9 - MIX Board				
Jumper	**Function**	**1-2 Position**	**2-3 Position**	**Default**
JB7	A/D Sample Clk	CTC #0 Output	Ext Sample Clock In	1-2
JB8	CTC #0,1 input	10 MHz Osc	Ext Sample Clock In	1-2
JB9	D/A Sample Clk	Ext Sample Clock In	CTC #1 Output	2-3

The signal applied to the EXT SAMPLE CLOCK IN BNC connector should comply with TTL levels, where a logic '0' is between 0.0 and +0.8 V_{DC}, and a logic '1' is between +2.0 V_{DC} and +5.0 V_{DC}. The input frequency range is from DC to 10 MHz when the signal is used as an input to the CTC, and the range is DC to 1 MHz when the input is used directly as the sample clock for either the A/D or D/A converter. The duty cycle is not critical, but the logic '0' and logic '1' intervals should both be at least 40 nsec. The active edge of the EXT SAMPLE CLOCK IN is the low-to-high transition.

If the cable connecting the External Sample Clock is longer than a few feet, it is recommended that the clock source be capable of driving a 50 Ω load, and that a 50 Ω termination resistor be placed at the front panel BNC. This will help to avoid overshoot and ringing. Notice that a diode clamping network restricts the excursion of the input voltage to a 0 to 5 V_{DC} range to help protect damage from over-voltage conditions.

4.3 Programming the Sample Rate

If the Counter/Timer Chip (CTC) is selected as the sample clock source for either the ADC or the DAC sections of the Model 4249 (i. e., if a jumper is installed on JB7 between pins 1 and 2, or on JB9 between pins 2 and 3), then the CTC must be programmed to divide its input clock signal to the sample rate required for your application. The CTC is an Intel Model 82C54, triple-channel, 16-bit programmable interval timer. Only two of the device's three channels are usable in the Model 4249, channel 0 for the A/D clock and channel 1 for the D/A sample clock. Channel 2 of the CTC is unused.

The CTC is an 8-bit device and all **reads and writes to it by the MIX Baseboard utilize bits D23 through D16 only of the 32-bit MIX data word.** Bit D23 is the most significant bit of the 8-bit CTC byte. The least significant 16 bits and the most significant 8 bits of the MIX data word are ignored for writes and are indeterminate for reads. Complete programming information for this device can be found in Appendix B of this manual, the 82C54 data sheet. Briefly, the part is programmed by means of three MIX addressable registers on the Model 4249. The physical addresses of these registers are dependent upon the MIX Baseboard that the 4249 is mounted on, and they can be found in Table 3-1, Table 3-2, Table 3-4 or Table 3-5.

4.3 Programming the Sample Rate (continued)

The Counter/Timer Control Register (at symbolic address CT_Ctl_Reg) determines which counter section will be accessed, which byte of the selected 16-bit counter will be accessed and the mode in which the counter will be used. The format of the control word written to this register is shown in Table 4-2, below. We recommend using the CTC in mode 2, called Rate Generator mode in Appendix B, which causes the device to function as a simple divide-by-N counter. The divisor (N) is written to the appropriate counter section's divisor register (symbolic addresses CT_0 for the A/D counter, and CT_1 for the D/A counter), using a single write cycle for an 8-bit divisor, or two writes for a 16-bit divisor.

Table 4-2: 82C54 - CTC Control Word Format

A_1,A_0 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC - Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

RW - Read/Write:

RW1	RW0	
0	0	Counter Latch Command (See Read Operations)
0	1	Read/Write least significant byte only
1	0	Read/Write most significant byte only
1	1	Read/Write least significant byte first, then most significant byte.

M - MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

4.4 Using the Status Bits

There are two ways of informing the MIX Baseboard or Upper MIX Bus Master (UMBM) in control of the Model 4249 that the A/D section has data to deliver, or that the D/A section needs data to convert. The controlling device can either poll the 4249's Status Register to see when service is required, or it can wait for an interrupt from the 4249. Interrupts will be covered in the next section. The Status register will be discussed below.

Data from the 4249's A/D converter is passed into a First-In, First-Out (FIFO) buffer. This buffer is 1,024 samples deep in a standard Model 4249, or 16,384 samples deep in units equipped with Option 003. The FIFO has three condition "flag" outputs whose states are passed directly to the 4249's Status Register, at symbolic MIX address Status_Reg (refer to Table 3-1, Table 3-2, Table 3-4 or Table 3-5 for the Baseboard-dependent physical address). These active-low bits will normally be in the logic '1' state, and will be cleared to logic '0' when the indicated condition is true.

Bit D18 of the Status Register will be a logic '0' when the FIFO is empty, i. e., after the FIFO has been cleared or after all samples have been read and no new samples have yet been written. If D18 is in the logic '1' state, then the FIFO contains at least one sample of data from the ADC.

The D17 bit of the Status Register will be in the logic '0' state when the FIFO has been filled to half of its capacity (512 samples for a standard unit or 8,192 samples for Option 003). This bit will remain in the '0' state until sufficient data is read from the FIFO to make it less than half full.

The Status Register's D16 bit will be in the logic '0' state when the FIFO has been filled to its maximum capacity. When the Input FIFO is full, the A/D sample clock is disabled, to prevent the A/D converter from taking samples that it cannot store. Thus, when this bit is in the logic '0' state, sampling of the input signal ceases.

In applications that require continuous real-time sampling of the input signal, the software should be configured such that the FIFO never becomes full. In such systems, the Baseboard (or UMBM) should read the FIFO either when the Half-full flag goes true (logic '0'), or when the Empty flag goes false (logic '1'). The status bits associated with the Input A/D converter's FIFO are summarized in the bottom half of Table 4-3, at the top of the next page.

The Model 4249's D/A Converter accepts data from a similar FIFO buffer. Again, this buffer is either 1,024 or 16,384 samples deep, depending upon the options purchased. The Status bits associated with the D/A's FIFO are D21 (FIFO Empty), D20 (FIFO Half-full) and D19 (FIFO Full). Note that when the D/A FIFO goes Empty, the last data value read from the FIFO (i. e., written to the DAC) is held at the analog output. The D/A FIFO status bits are summarized in the top half of Table 4-3, at the top of the next page.

4.4 Using the Status Bits (continued)

Table 4-3: Model 4249 - FIFO Status Bits - MIX Address Status_Reg		
Output (D/A) FIFO		
Bit	**Logic '1'**	**Logic '0'**
D21	Output FIFO has data	Output FIFO Empty
D20	Output FIFO less than Half Full	Output FIFO Half Full or more
D19	Output FIFO not Full	Output FIFO Full
Input (A/D) FIFO		
Bit	**Logic '1'**	**Logic '0'**
D18	Input FIFO has data	Input FIFO Empty
D17	Input FIFO less than Half Full	Input FIFO Half Full or more
D16	Input FIFO not Full	Input FIFO Full

4.5 Using Interrupts

The Model 4249 can be programmed to issue MIX bus interrupts based on the states of the FIFO status bits described in Table 4-3, above. Four bits in the 4249's Control Register select the condition(s) on which a MIX interrupt will be generated, according to Table 4-4, below. The Control Register is at symbolic MIX address Control_Reg (refer to Table 3-1, Table 3-2, Table 3-4 or Table 3-5 for the Baseboard-dependent physical address), and the Interrupt Mask bits are D29, D28, D27 and D26. When the 4249 issues a MIX interrupt, the D23 bit in the Status Register is cleared to the logic '0' state, to indicate that a pending interrupt request has been made by this device. This bit is set back to the logic '1' state, and the interrupt request is cleared, when the Status Register is read.

Table 4-4: Model 4249 - FIFO Interrupt Selection Code (MIX Address Control_Reg)					
Control Register Bit #				**Output FIFO (D/A) Interrupt Conditions**	**Input FIFO (A/D) Interrupt Conditions**
D29	**D28**	**D27**	**D26**		
0	0	0	0	None	None
0	0	0	1	Output FIFO NOT Half Full	None
0	0	1	0	Output FIFO Empty	None
0	0	1	1	None	Input FIFO NOT Empty
0	1	0	0	None	Input FIFO Half Full
0	1	0	1	Output FIFO NOT Half Full	Input FIFO Full
0	1	1	0	Output FIFO NOT Half Full	Input FIFO Half Full
0	1	1	1	Output FIFO Empty	Input FIFO NOT Empty
1	0	0	0	Output FIFO Empty	Input FIFO Half Full

4.6 FIFO Data Formats

The Model 4249 uses FIFOs that are 16 bits wide. For the A/D Input FIFO, the 12 data bits written to the FIFO by the ADC are justified in the FIFO's 12 Most significant bit positions. When reading the FIFO over the MIX bus, the 12 MSB's of the 32-bit MIX data word (D31 - D20) contain the ADC's output data and the next four bits (D19 - D16) will always be zeros. The 16 LSBs are not driven, thus their states will be indeterminate and they should be ignored. The data format for the A/D FIFO is shown in Table 4-5, below.

Table 4-5: Model 4249 - Input (A/D) FIFO Data Format - MIX Address AD_FIFO - Read Only

Bit #	D31 D30 - - - - - D21 D20	D19 D18 D17 D16	D15 - - - - - - - - - - D0
Function	MSB - - - 12-bit A/D Out - - - LSB	- - - Zeroes - - -	- - - undriven - - -

Data written by the MIX Baseboard or Upper MIX Bus Master to the D/A Output FIFO must be justified in the 12 most significant bits (D31 - D20) of the 32-bit MIX data word, as shown in Table 4-6, below.

Table 4-6: Model 4249 - Output (D/A) FIFO Data Format - MIX Address DA_FIFO - Write Only

Bit #	D31 D30 D29 - - - - - D22 D21 D20	D19 - - - - - - - - - - - - - - - - - D0
Function	MSB - - - - - - 12-bit A/D Output - - - - - - LSB	- - - - - - - unused - - - - - - -

Appendix A: Programming Examples

A.1 Introduction

The 'C' programs that follow were written to allow the Model 4249 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 at the top of Page A-5) and **4249.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 4249. 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 **4249.h** contains header information specific to the Model 4249.

The file **p4249.c** (see Section A.3, beginning on Page A-2) contains the code that allows the baseboard selected in **mod.h** to initialize the Model 4249, release it to accept analog and digital 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 4249 is ready to deliver data. The file **init.c** (see Section A.5 on Page A-9) performs any necessary pre-initialization chores on the baseboard or UMBM. Finally, the file **intr.c** (see Section A.6, starting at the top of Page A-10) determines which (if any) MIX interrupts to enable, using information from **mod.h**.

A.2 4249.h

```
/* 4249.h  - This file contains 4249 module specific information      */

#ifndef P4249_H
#define P4249_H


#define P4249_FIFO_SIZE         (1 * 1024)

/* 4249 Mix Module Register Offsets                                   */
#define P4249_INDATA_REG     0x00000000L | A27_ADDR_BIT
#define P4249_OUTDATA_REG    0x00000000L | A27_ADDR_BIT
#define P4249_CONTROL_REG    0x00000001L | A27_ADDR_BIT
#define P4249_STATUS_REG     0x00000002L | A27_ADDR_BIT
#define P4249_CTC_REG        0x00000004L | A27_ADDR_BIT
#define P4249_COUNTER_0_REG  0x00000007L | A27_ADDR_BIT
#define P4249_COUNTER_1_REG  0x00000006L | A27_ADDR_BIT
#define P4249_COUNTER_2_REG  0x00000005L | A27_ADDR_BIT
```

A.2 4249.h (continued)

```
/* Constants specific to each Register                                  */

/* Status Register                                                      */
#define SR_A_D_FIFO_FULL            0x00010000L
#define SR_A_D_FIFO_HALF_FULL       0x00020000L
#define SR_A_D_FIFO_EMPTY           0x00040000L
#define SR_D_A_FIFO_FULL            0x00080000L
#define SR_D_A_FIFO_HALF_FULL       0x00100000L
#define SR_D_A_FIFO_EMPTY           0x00200000L
#define SR_INTR_PENDING             0x00400000L
#define SR_OVERLOAD                 0x00800000L

/* Control Register                                                     */
#define CR_A_D_FIFO_RESET           0x01000000L
#define CR_D_A_FIFO_RESET           0x02000000L
#define CR_OVRLOAD_INTR             0x80000000L

/* Interrupt Bit Mask (Control Register Bits 26-29)                     */
#define IM_D_A_FIFO_NOT_HALF_FULL   0x04000000L
#define IM_D_A_FIFO_EMPTY           0x08000000L
#define IM_A_D_FIFO_NOT_EMPTY       0x0c000000L
#define IM_A_D_FIFO_HALF_FULL       0x10000000L

#endif
```

A.3 p4249.c

```
#include "mod.h"
#include "4249.h"

#define M4249      1

#define POLLING    0
#define INTERRUPTS 1

#if BB_4283
unsigned int *PageAddressReg, *ExpBusControlReg;
#endif

unsigned int *Status, *Control, *CtcMode, *Counter0, *Counter1, *Counter2;
unsigned int *InData, *OutData;
unsigned int sig[2048],*sigP;

extern void enableInterrupts();
extern void clearInterrupts();
extern void board_init();
```

A.3 **p4249.c** (continued)

```
main()
{
    unsigned int i;

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

    Status      = (unsigned int *) (MOD | P4249_STATUS_REG);
    Control     = (unsigned int *) (MOD | P4249_CONTROL_REG);
    InData      = (unsigned int *) (MOD | P4249_INDATA_REG);
    OutData     = (unsigned int *) (MOD | P4249_OUTDATA_REG);
    CtcMode     = (unsigned int *) (MOD | P4249_CTC_REG);
    Counter0    = (unsigned int *) (MOD | P4249_COUNTER_0_REG);
    Counter1    = (unsigned int *) (MOD | P4249_COUNTER_1_REG);
    Counter2    = (unsigned int *) (MOD | P4249_COUNTER_2_REG);

#if BB_4283
    *ExpBusControlReg = 0x68;             /* Remove MIX bus wait states */
    *PageAddressReg   = MOD_SELECT | D17_DATA_BIT;    /* Select module */
#endif

board_init();

    /* Initialize the 4249                                              */
    *Control  = 0x0u;                                 /* Reset FIFO's */
    *CtcMode  = 0x340000u;                                /* Ch0 mode */
    *Counter0 = 0x320000u;             /* Input Sample rate divisor lsb */
    *Counter0 = 0x000000u;             /* Input Sample rate divisor msb */
    *CtcMode  = 0x740000u;                                /* Ch1 mode */
    *Counter1 = 0x320000u;            /* Output Sample rate divisor lsb */
    *Counter1 = 0x000000u;            /* Output Sample rate divisor msb */

    i = *Status;                           /* Clear Pending Interrupts */

#if INTERRUPTS
    /* Enable Interrupts                                                */
    enableInterrupts();
#endif

#if INTERRUPTS
     /* Release and acquire with Input Fifo half full interrupt enabled   */
     *Control=CR_A_D_FIFO_RESET | CR_D_A_FIFO_RESET | IM_A_D_FIFO_HALF_FULL;
#else
     /* Release and acquire                                              */
        *Control= CR_A_D_FIFO_RESET | CR_D_A_FIFO_RESET;
#endif
```

A.3 **p4249.c** (continued)

```
         while(1)
         {
            ;
#if POLLING
         /* wait for input FIFO half-full                              */
         while ((*Status & SR_A_D_FIFO_HALF_FULL) != 0);

         /* Read Data from Input Fifo                                  */
         for (i=0;i<P4249_FIFO_SIZE/2;i++)
            sig[i] = *InData;
#if M4249
         /* wait for output FIFO not half-full                         */
         while ((*Status & SR_D_A_FIFO_HALF_FULL) == 0);

         /* Write Data to Output Fifo                                  */
         for (i=0;i<P4249_FIFO_SIZE/2;i++)
            *OutData = sig[i];
#endif

#endif
         }
}

c_int01()
{
    int temp, i;

    /* Copy Input Fifo to Output Fifo                                  */
    for(i=0;i<P4249_FIFO_SIZE/2;i++)
         sig[i] = *InData;

#if M4249
    /* Write Data to Output Fifo                                       */
    for (i=0;i<P4249_FIFO_SIZE/2;i++)
         *OutData = sig[i];
#endif

    temp = *Status;                          /* Clear interrupt latch */

    /* Clear Processor 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         4285

 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 4285, 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     */

/* Define BaseBoard Type                                             */
#define BB_4283        0
#define BB_4284        0
#define BB_4285        1
#define BB_4200        0
#define BB_4201        0
#define BB_4202        0

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

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

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

```
#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

#if BB_4285
/* The 4285 has 2 independent paths for accessing the MIX bus resources:
   The Expansion Bus and the Global I/O bus.  Note:  The Expansion Bus
   may be mastered by any of the on-card 'C40 processors or by off-card
   VMEbus masters.  The Global I/O Bus can be mastered only by 'C40
   processors A, B, E or F.
   Note: MIX interrupts are not supported over the Global I/O Bus     */

#define EXP_BUS         1
#define GLOB_BUS        0
#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

/* Addr 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 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 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 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 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
```

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

```
/* BaseBoard = 4202                                                    */
#if BB_4202

#if MM_4270 | MM_425X             /* Mix 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

/* BaseBoard = 4285                                                    */
#if BB_4285

#if MM_4270 | MM_425X             /* Mix 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 Module */

#if EXP_BUS
#define  MOD0_ADDR        0x90000000
#define  MOD1_ADDR        0x90200000
#define  MOD2_ADDR        0x90400000

/* Address Bit in address which corresponds to A27                     */
#define A27_ADDR_BIT      0x00100000
#endif

#if GLOB_BUS
#define MOD0_ADDR         0xC8000000
#define MOD1_ADDR         0xC8200000
#define MOD2_ADDR         0xC8400000

/* Address Bit in address which corresponds to A27                     */
#define A27_ADDR_BIT      0x00100000
#endif

#endif

#endif
/*************************** End of File ******************************/
```

A.5 init.c

```
#include "mod.h"

board_init(){

#if BB_4284 | BB_4285
    /* The bootcode initializes the timer to interrupt the CPU
       periodically to flash the LED on the front panel.  Turn off
       the timer to prevent unwanted interrupts.                    */

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

#if BB_4285 & !(MM_4270 | MM_425X)

/* Clear 4285 interrupt routing register                             */
*(unsigned long *) 0x90600001 = 0;

/* Setup 4285 Mix Control Register for compressed mode and release
   when done. Compressed mode allows you to avoid having to set a
   page register when accessing MIX modules.  Release when done
   prevents the C40 from hogging the MIX Bus.                        */
   #if EXP_BUS
      *(unsigned long *) 0x90600000 = 0x2;
   #endif
   #if GLOB_BUS
      *(unsigned long *) 0xC8600000 = 0x2;
      *(unsigned long *) 0xC8600001 = 0;
   #endif
#endif

#if MM_425X
    /* Set the release when done bit.  This will prevent the C40
       from hogging the MIX Bus.                                    */
    *(unsigned int *) 0x82000001 = 0x40;         /* Release when done */
#endif

#if MM_4270
    /* The 4270 can provide an interrupt to the CPU when it can not
       obtain a shared resource.  Disabling this feature will force
       the C40 to wait for the shared resource to be freed up.      */
   *(unsigned int *) 0x34000d = 0xff;       /* Disable NMI generation */
#endif
}
```

A.6 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_4285 & !(MM_4270 | MM_425X)
    /* The 4285 Uses IIOF3 for all MIX modules                        */
    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();
    }
```

A.6 **intr.c** (continued)

```
    else if (MIX_POS == 1)
    {
         if (MOD == MOD0_ADDR)
             IntInit3();
         else
             IntInit2();
    }
    else
    {
         if (MOD == MOD0_ADDR)
             IntInit2();
         else
             IntInit3();
    }

#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 generated 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 | BB_4285
    asm("   iack @0");                /* Clear Any Pending Interrupts */
#endif
#if BB_4283
    if (MOD_SELECT == MOD0_SELECT)
      *(int *)0x800000 = 0x0c;  /* Acknowledge 4283 interrupt processor */
    else if (MOD_SELECT == MOD1_SELECT)
      *(int *)0x800000 = 0x0a;  /* Acknowledge 4283 interrupt processor */
    else
      *(int *)0x800000 = 0x06;  /* Acknowledge 4283 interrupt processor */

    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
}
```

A.6 **intr.c** (continued)

```
#if BB_4283 | BB_4284 | BB_4285
void Init1()
{

#if BB_4283
        /* setup int 1 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");
#else
        /* Enable C40 Int 0 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 int 2 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");
```

A.6 **intr.c** (continued)

```
#else
         /* Enable C40 Int 2 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
}

void Init3()
{
unsigned int *IRR_85, *CR_85, proc_id, mod_pos;

#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

#if BB_4284
         /* Enable C40 Int 3 as MixInt2                                    */
         *(unsigned int *) 0x10000003 = *(unsigned int *) 0x10000003 |
                                                          0x20000000;
#endif

/* Setup 4285 Interrupt Routing Register based on the Processor
   ID and Mix module position                                              */

#if BB_4285
IRR_85 = (unsigned int *)0x90600001;
CR_85  = (unsigned int *)0x380000;
```

A.6 **intr.c** (continued)

```
proc_id = (1 << (*CR_85 & 0x7));    /* Get Proc ID from 85 Control Reg */
mod_pos = (((MOD >> 20) & 0xF) * 4);    /* Get Mod position from mod.h */

/* Write to 4285 IRR using processor id shifted by the mod position   */
*(unsigned long *) IRR_85 = (proc_id << mod_pos);
#endif

    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");
}
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
```

Appendix B: Intel 82C54 Data Sheet

B.1 Introduction

The Intel 82C54 is a high-performance, CHMOS version of the industry standard 8254 counter/timer, which is designed to solve the timing control problems common in microcomputer system design. It provides three independent 16-bit counters, each capable of handling clock inputs up to 10 MHz. All modes are software programmable. The 82C54 is pin compatible with the HMOS 8254, and is a superset of the 8253.

Six programmable timer modes allow the 82C54 to be used as an event counter, elapsed time indicator, programmable one-shot, and in many other applications.

The 82C54 is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent HMOS product. The 82C54 is available in 24-pin DIP and 28-pin plastic leaded chip carrier (PLCC) packages.

B.1.1 Key Features

- Compatible with all Intel and most other microprocessors
- High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188
- Handles Inputs from DC to 8 MHz (10 MHz for 82C54-2)
- Three independent 16-bit counters
- Completely TTL Compatible
- Six Programmable Counter Modes
- Binary or BCD counting
- Status Read Back Command

B.2 Functional Description

The 82C54 is a programmable interval timer/counter designed for use with Intel microcomputer systems. It is a general purpose, multi-timing element that can be treated as an array of I/O ports in the system software.

The 82C54 solves one of the most common problems in any microcomputer system, the generation of accurate time delays under software control. Instead of setting up timing loops in software, the programmer configures the 82C54 to match his requirements and programs one of the counters for the desired delay. After the desired delay, the 82C54 will interrupt the CPU. Software overhead is minimal and variable length delays can easily be accommodated.

B.2 Functional Description (continued)

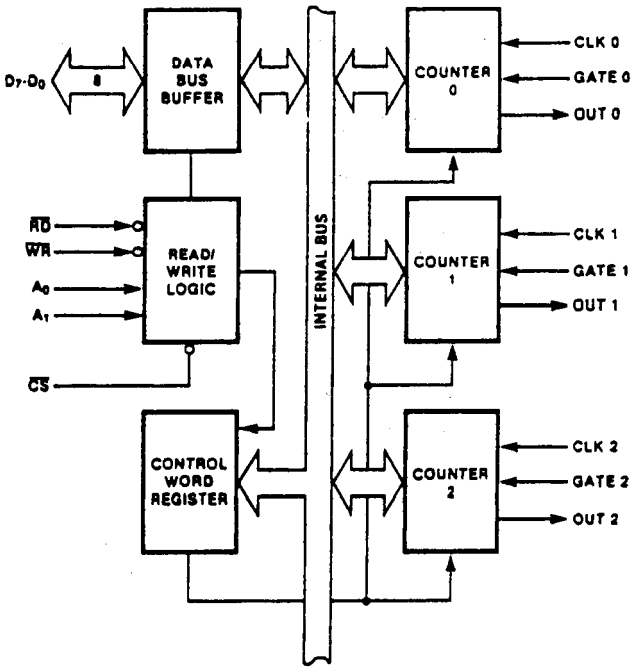


Figure B-1: 82C54 - Block Diagram

Some of the other counter/timer functions common to microcomputers which can be implemented with the 82C54 are :

- ❑ Real time clock
- ❑ Digital one-shot
- ❑ Square wave generator
- ❑ Complex waveform generator
- ❑ Event Counter
- ❑ Programmable rate generator
- ❑ Binary rate multiplier
- ❑ Complex motor controller

B.3 Block Diagram

B.3.1 Data Bus Buffer

This 3-state, bi-directional, 8-bit buffer is used to interface the 82C54 to the system bus (see Figure B-2, at the top of the next page).

B.3 Block Diagram (continued)

B.3.1 Data Bus Buffer (continued)

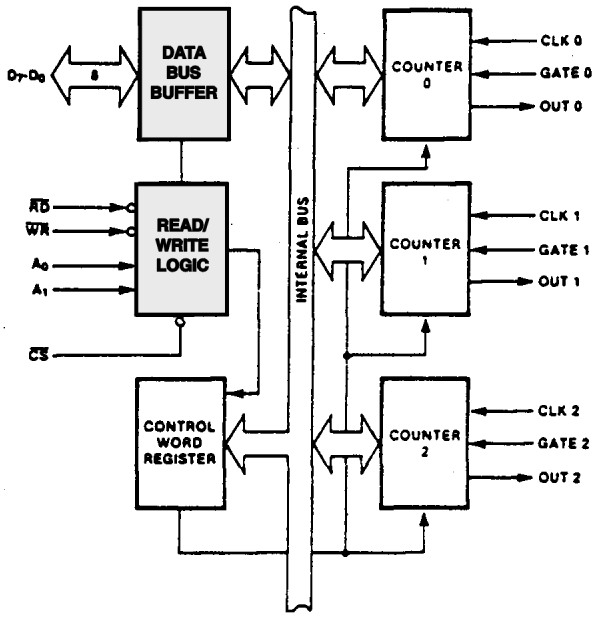


Figure B-2: 82C54 - Block Diagram Showing Data Bus Buffer and Read/Write Logic Functions

B.3.2 Read/Write Logic

The Read/Write Logic accepts inputs from the system bus and generates control signals for the other functional blocks of the 82C54. A_1 and A_0 select one of the three counters or the Control Word Register to be read from/written into. A "low" on the $\overline{RD}$ input tells the 82C54 that the CPU is reading one of the counters. A "low" on the $\overline{WR}$ input tells the 82C54 that the CPU is writing either a Control Word or an initial count. Both $\overline{RD}$ and $\overline{WR}$ are qualified by $\overline{CS}$; $\overline{RD}$ and $\overline{WR}$ are ignored unless the 82C54 has been selected by holding $\overline{CS}$ low.

B.3.3 Control Word Register

The Control Word Register (see Figure B-3, at the top of the next page) is selected by the Read/Write Logic when A_1, A_0 = 11. If the CPU then does a write operation to the 82C54, the data is stored in the Control Word Register and is interpreted as a Control Word used to define the operation of the Counters.

The Control Word Register can only be written to; status information is available with the Read-Back Command.

B.3 Block Diagram (continued)

B.3.3 Control Word Register (continued)

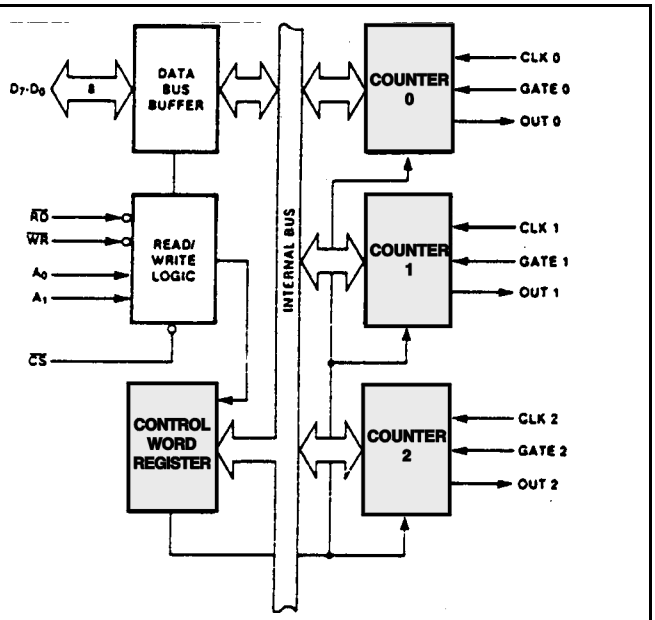


Figure B-3: 82C54 - Block Diagram Showing Control Word Register and Counter Functions

B.3.4 Counter 0, Counter 1, Counter 2

These three functional blocks are identical in operation, so only a single Counter will be described. The internal block diagram of a single counter is shown in Figure B-4, at the top of the next page.

The Counters are fully independent. Each Counter may operate in a different Mode.

The Control Word Register is also shown in Figure B-4; it is not part of the Counter itself, but its contents determine how the Counter operates.

The status register, shown in the figure, when latched, contains the current contents of the Control Word Register and status of the output and null count flag. (See detailed explanation of the Read-Back command.)

B.3 Block Diagram (continued)

B.3.4 Counter 0, Counter 1, Counter 2 (continued)

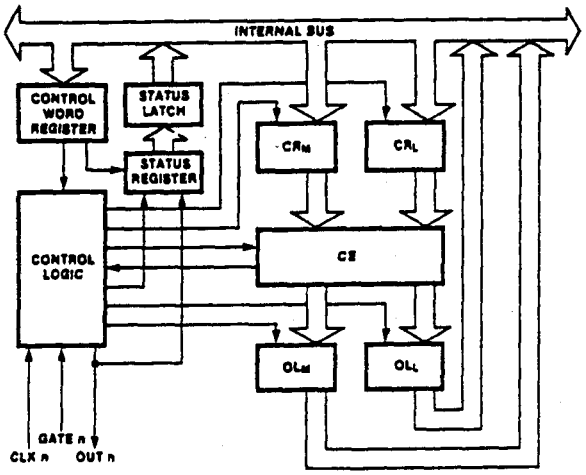


Figure B-4: 82C54 - Internal Block Diagram of a Counter

The actual counter is labelled CE (for "Counting Element"). It is a 16-bit presettable synchronous down counter.

OL_M and OL_L are two 8-bit latches. OL stands for "Output Latch"; the subscripts M and L stand for "Most significant byte" and "Least significant byte" respectively. Both are normally referred to as one unit and called just OL. These latches normally "follow" the CE, but if a suitable Counter Latch Command is sent to the 82C54, the latches "latch" the present count until read by the CPU and then return to "following" the CE. One latch at a time is enabled by the counter's Control Logic to drive the internal bus. This is how the 16-bit Counter communicates over the 8-bit internal bus. Note that the CE itself cannot be read; whenever you read the count, it is the OL that is being read.

Similarly, there are two 8-bit registers called CR_M and CR_L (for "Count Register"). Both are normally referred to as one unit and called just CR. When a new count is written to the Counter, the count is stored in the CR and later transferred to the CE. The Control Logic allows one register at a time to be loaded from the internal bus. Both bytes are transferred to the CE simultaneously. CR_M and CR_L are cleared when the Counter is programmed. In this way, if the Counter has been programmed for one byte counts (either most significant byte only or least significant byte only) the other byte will be zero. Note that the CE cannot be written into; whenever a count is written, it is written into the CR.

B.3 Block Diagram (continued)

B.3.4 Counter 0, Counter 1, Counter 2 (continued)

The Control Logic is also shown in the diagram. CLKn, GATEn, and OUTn are all connected to the outside world through the Control Logic.

B.4 82C54 System Interface

The 82C54 is treated by the systems software as an array of peripheral I/O ports; three are counters and the fourth is a control register for MODE programming.

Basically, the select inputs A_0, A_1 connect to the A_0, A_1 address bus signals of the CPU. The $\overline{CS}$ can be derived directly from the address bus using a linear select method. Or it can be connected to the output of a decoder, such as an Intel 8205 for larger systems.

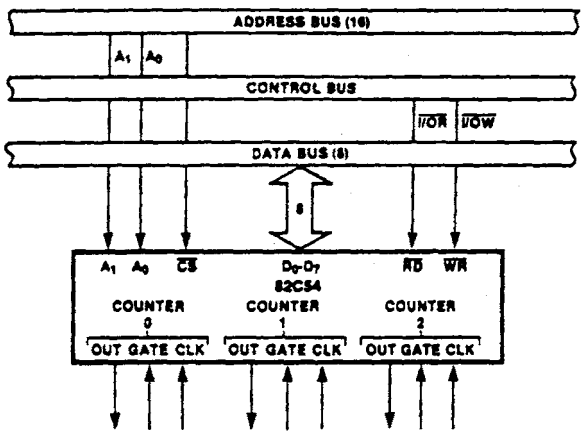


Figure B-5: 82C54 - System Interface

B.5 Operational Description

After power-up, the state of the 82C54 is undefined. The Mode, count value, and output of all Counters are undefined.

How each Counter operates is determined when it is programmed. Each Counter must be programmed before it can be used. Unused counters need not be programmed.

B.5 Operational Description (continued)

B.5.1 Programming the 82C54

Counters are programmed by writing a Control Word and then an initial count. The control word format is shown in Table B-1, below.

Table B-1: 82C54 - Control Word Format

A_1,A_0 = 11 $\overline{CS}$ = 0 $\overline{RD}$ = 1 $\overline{WR}$ = 0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

SC - Select Counter:

SC1	SC0	
0	0	Select Counter 0
0	1	Select Counter 1
1	0	Select Counter 2
1	1	Read-Back Command (See Read Operations)

RW - Read/Write:

RW1	RW0	
0	0	Counter Latch Command (See Read Operations)
0	1	Read/Write least significant byte only
1	0	Read/Write most significant byte only
1	1	Read/Write least significant byte first, then most significant byte.

M - MODE:

M2	M1	M0	
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

BCD:

0	Binary Counter 16-bits
1	Binary Coded Decimal (BCD) Counter (4 Decades)

NOTE: Don't care bits (X) should be 0 to insure compatibility with future Intel products.

All Control Words are written into the Control Word Register, which is selected when A_1, A_0 = 11. The Control Word itself specifies which Counter is being programmed.

By contrast, initial counts are written into the Counters, not the Control Word Register. The A_1, A_0 inputs are used to select the Counter to be written into. The format of the initial count is determined by the Control Word used.

B.5 Operational Description (continued)

B.5.2 Write Operations

The programming procedure for the 82C54 is very flexible. Only two conventions need to be remembered:

1) For each Counter, the Control Word must be written before the initial count is written.

2) The initial count must follow the count format specified in the Control Word (least significant byte only, most significant byte only, or least significant byte and then most significant byte.)

Since the Control Word Register and the three Counters have separate addresses (selected by the A_1, A_0 inputs), and each Control Word specifies the Counter it applies to (SC0, SC1 bits), no special instruction sequence is required. Any programming sequence that follows the conventions above is acceptable.

A new initial count may be written to a Counter at any time without affecting the Counter's programmed Mode in any way. Counting will be affected as described in the Mode definitions. The new count must follow the programmed count format.

If a Counter is programmed to read/write two-byte counts, the following precaution applies: a program must not transfer control between writing the first and second byte to another routine which also writes into that same Counter. Otherwise, the Counter will be loaded with an incorrect count.

Table B-2: 82C54 - A Few Possible Programming Sequences

	A_1	A_0		A_1	A_0
Control Word - Counter 0	1	1	Control Word - Counter 2	1	1
LSB of count - Counter 0	0	0	Control Word - Counter 1	1	1
MSB of count - Counter 0	0	0	Control Word - Counter 0	1	1
Control Word - Counter 1	1	1	LSB of count - Counter 2	1	0
LSB of count - Counter 1	0	1	MSB of count - Counter 2	1	0
MSB of count - Counter 1	0	1	LSB of count - Counter 1	0	1
Control Word - Counter 2	1	1	MSB of count - Counter 1	0	1
LSB of count - Counter 2	1	0	LSB of count - Counter 0	0	0
MSB of count - Counter 2	1	0	MSB of count - Counter 0	0	0
	A_1	A_0		A_1	A_0
Control Word - Counter 0	1	1	Control Word - Counter 1	1	1
Counter Word - Counter 1	1	1	Control Word - Counter 0	1	1
Control Word - Counter 2	1	1	LSB of count - Counter 1	0	1
LSB of count - Counter 2	1	0	Control Word - Counter 2	1	1
LSB of count - Counter 1	0	1	LSB of count - Counter 0	0	0
LSB of count - Counter 0	0	0	MSB of count - Counter 1	0	1
MSB of count - Counter 0	0	0	LSB of count - Counter 2	1	0
MSB of count - Counter 1	0	1	MSB of count - Counter 0	0	0
MSB of count - Counter 2	1	0	MSB of count - Counter 2	1	0

NOTE:
In all four examples, all counters are programmed to read/write two-byte counts.
These are only four of many possible sequences.

B.5 Operational Description (continued)

B.5.3 Read Operations

It is often desirable to read the value of a Counter without disturbing the count in progress. This is easily done in the 82C54.

There are three possible methods for reading the counters: a simple read operation, the Counter Latch Command, and the Read-Back Command. Each is explained below. The first method is to perform a simple read operation. To read the Counter, which is selected with the A_1, A_0 inputs, the CLK input of the selected Counter must be inhibited by using either the GATE input or external logic. Otherwise, the count may be in the process of changing when it is read, giving an undefined result.

B.5.3.1 Counter Latch Command

The second method uses the "Counter Latch Command". Like a Control Word, this command is written to the Control Word Register, which is selected when A_1, A_0 = 11. Also like a Control Word, the SC0, SC1 bits select one of the three Counters, but two other bits, D5 and D4, distinguish this command from a Control Word.

Table B-3: 82C54 - Counter Latching Command Format

A_1,A_0 = 11; $\overline{CS}$ = 0; $\overline{RD}$ = 1; $\overline{WR}$ = 0

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	0	0	X	X	X	X

SC1, SC0 - specify counter to be latched

SC1	SC0	Counter
0	0	0
0	1	1
1	0	2
1	1	Read-Back Command

D5, D4 - 00 designates Counter Latch Command

X - Don't Care

NOTE:

Don't care bits (X) should be 0 to insure compatibility with future Intel products.

B.5 Operational Description (continued)

B.5.3 Read Operations (continued)

B.5.3.1 Counter Latch Command (continued)

The selected Counter's output latch (OL) latches the count at the time the Counter Latch Command is received. This count is held in the latch until it is read by the CPU (or until the Counter is reprogrammed). The count is then unlatched automatically and the OL returns to "following" the counting element (CE). This allows reading the contents of the Counters "on the fly" without affecting counting in progress. Multiple Counter Latch Commands may be used to latch more than one Counter. Each latched Counter's OL holds its count until it is read; Counter Latch Commands do not affect the programmed Mode of the Counter in any way.

If a Counter is latched and then, some time later, latched again before the count is read, the second Counter Latch Command is ignored. The count read will be the count at the time the first Counter Latch Command was issued.

With either method, the count must be read according to the programmed format; specifically, if the Counter is programmed for two byte counts, two bytes must be read. The two bytes do not have to be read one right after the other; read or write or programming operations of other Counters may be inserted between them.

Another feature of the 82C54 is that reads and writes of the same Counter may be interleaved; for example, if the Counter is programmed for two byte counts, the following sequence is valid.

1) Read least significant byte.
2) Write new least significant byte.
3) Read most significant byte.
4) Write new most significant byte.

If a Counter is programmed to read/write two-byte counts, the following precaution applies; a program must not transfer control between reading the first and second byte to another routine which also reads from that same Counter. Otherwise, an incorrect count will be read.

B.5.3.2 Read-Back Command

The third method uses the Read-Back command. This command allows the user to check the count value, programmed Mode, and current state of the OUT pin and Null Count flag of the selected counter(s).

B.5 Operational Description (continued)

B.5.3 Read Operations (continued)

B.5.3.2 Read-Back Command (continued)

The command is written into the Control Word Register and has the format shown in Table B-4, below. The command applies to the counters selected by setting their corresponding bits D3, D2, D1 = 1.

Table B-4: 82C54 - Read-Back Command Format

A_1,A_0 = 11; $\overline{CS}$ = 0; $\overline{RD}$ = 1; $\overline{WR}$ = 0

D7	D6	D5	D4	D3	D2	D1	D0
1	1	$\overline{COUNT}$	$\overline{STATUS}$	CNT 2	CNT 1	CNT 0	0

D5: 0 = Latch count of selected counter(s)
D4: 0 = Latch status of selected counter(s)
D3: 1 = Select counter 2
D2: 1 = Select counter 1
D1: 1 = Select counter 0
D0: Reserved for future expansion, must be 0

The read-back command may be used to latch multiple counter output latches (OL) by setting the N bit D5=0 and selecting the desired counter(s). This single command is functionally equivalent to several counter latch commands, one for each counter latched. Each counter's latched count is held until it is read (or the counter is reprogrammed). That counter is automatically unlatched when read, but other counters remain latched until they are read. If multiple count read-back commands are issued to the same counter without reading the count, all but the first are ignored; i. e., the count which will be read is the count at the time the first read-back command was issued.

The read-back command may also be used to latch status information of selected counter(s) by setting $\overline{STATUS}$ bit 04=0. Status must be latched to be read; status of a counter is accessed by a read from that counter.

The counter status format is shown in Table B-5, at the top of the next page. Bits D5 through D0 contain the counter's programmed Mode exactly as written in the last Mode Control Word. OUTPUT bit D7 contains the current state of the OUT pin. This allows the user to monitor the counter's output via software, possibly eliminating some hardware from a system.

B.5 Operational Description (continued)

B.5.3 Read Operations (continued)

B.5.3.2 Read-Back Command (continued)

Table B-5: 82C54 - Status Byte

D7	D6	D5	D4	D3	D2	D1	D0
OUTPUT	NULL COUNT	RW1	RW0	M2	M1	M0	BCD

D7 1 = Out Pin is 1
0 = Out Pin is 0
D8 1 = Null Count
0 = Count Available for reading
D5 - D0 Counter Programmed Mode (See Figure B-7(a))

NULL COUNT bit D6 indicates when the last count written to the counter register (CR) has been loaded into the counting element (CE). The exact time this happens depends on the Mode of the counter and is described in the Mode Definitions, but until the count is loaded into the counting element (CE), it can't be read from the counter. If the count is latched or read before this time, the count value will not reflect the new count just written. The operation of Null Count is shown in Table B-6, below.

Table B-6: 82C54 - Null Count Operation

This Action:	Causes:
A. Write to the control word register: [1]	Null count = 1
B. Write to the count register (CR); [2]	Null count = 1
C. New count is loaded into CE(CR → CE);	Null count = 0

[1] Only the counter specified by the control word will have its null count set to 1. Null count bits of other counters are unaffected.
[2] If the counter is programmed for two-byte counts (least significant byte then most significant byte) null count goes to 1 when the second byte is written.

B.5 Operational Description (continued)

B.5.3 Read Operations (continued)

B.5.3.2 Read-Back Command (continued)

If multiple status latch operations of the counter(s) are performed without reading the status, all but the first are ignored; i. e., the status that will be read is the status of the counter at the time the first status read-back command was issued.

Both count and status of the selected counter(s) may be latched simultaneously by setting both $\overline{\text{COUNT}}$ and $\overline{\text{STATUS}}$ bits D5, D4=0. This is functionally the same as issuing two separate read-back commands at once, and the above discussions apply here also. Specifically, if multiple count and/or status read-back commands are issued to the same counter(s) without any intervening reads, all but the first are ignored. This is illustrated in Table B-7, below.

If both count and status of a counter are latched, the first read operation of that counter will return latched status, regardless of which was latched first. The next one or two reads (depending on whether the counter is programmed for one or two type counts) return latched count. Subsequent reads return unlatched count.

Table B-7: 82C54 - Read-Back Command Example

Command								Description	Results
D7	D6	D5	D4	D3	D2	D1	D0		
1	1	0	0	0	0	1	0	Read back count and status of Counter 0	Count and status latched for Counter 0
1	1	1	0	0	1	0	0	Read back status of counter 1	Status latch for Counter 1
1	1	1	0	1	1	0	0	Read back status of Counters 2,1	Status latched for Counter 2, but not Counter 1
1	1	0	1	1	0	0	0	Read back count of Counter 2	Count latched for Counter 2
1	1	0	0	0	1	0	0	Read back count and status of Counter 1	Count latched for Counter 1, but not status
1	1	1	0	0	0	1	0	Read back status of Counter 1	Command ignored, status already latched for Counter 1

B.5 Operational Description (continued)

B.5.3 Read Operations (continued)

B.5.3.2 Read-Back Command (continued)

Table B-8: 82C54 - Read/Write Operations Summary

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	
0	1	0	0	0	Write into Counter 0
0	1	0	0	1	Write into Counter 1
0	1	0	1	0	Write into Counter 2
0	1	0	1	1	Write Control Word
0	0	1	0	0	Read from Counter 0
0	0	1	0	1	Read from Counter 1
0	0	1	1	0	Read from Counter 2
0	0	1	1	1	No-Operation (3-State)
1	X	X	X	X	No-Operation (3-State)
0	1	1	X	X	No-Operation (3-State)

B.5.4 Mode Definitions

The following are defined for use in describing the operation of the 82C54.

CLK PULSE: A rising edge, then a falling edge, in that order, of a Counter's CLK input.

TRIGGER: A rising edge of a Counter's GATE input.

COUNTER LOADING: The transfer of a count from the CR to the CE (refer to the "Functional Description")

B.5.4.1 Mode 0: Interrupt on Terminal Count

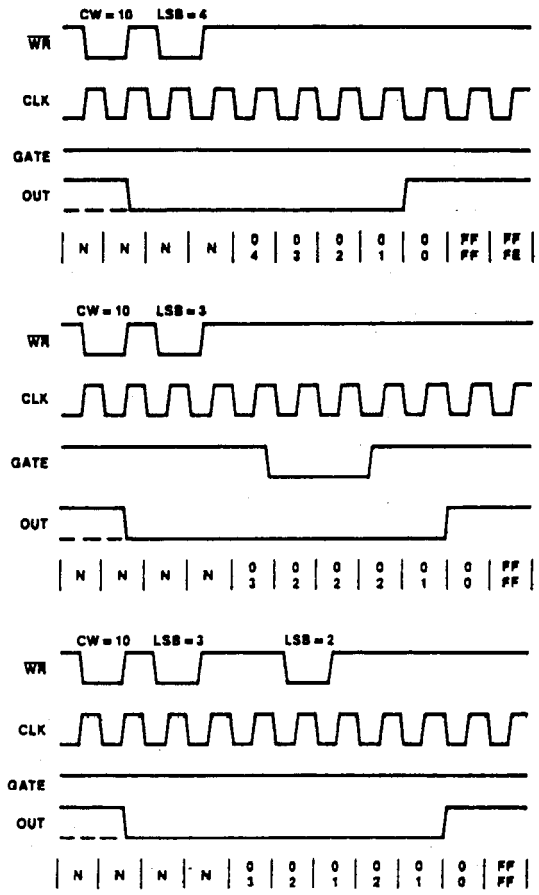
Mode 0 is typically used for event counting. After the Control Word is written, OUT is initially low, and will remain low until the Counter reaches zero. OUT then goes high and remains high until a new count or a new Mode 0 Control Word is written into the Counter.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.1 Mode 0: Interrupt on Terminal Count (continued)



NOTES: The Following Conventions Apply To All Mode Timing Diagrams:

1. Counters are programmed for binary (not BCD) counting and for Reading/Writing least significant byte (LSB) only.
2. The counter is always selected (CS always low).
3. CW stands for "Control Word"; CW = 10 means a control word of 10, hex is written to the counter.
4. LSB stands for "Least Significant Byte" of count.
5. Numbers below diagrams are count values.
   The lower number is the least significant byte.
   The upper number is the most significant byte. Since the counter is programmed to Read/Write LSB only, the most significant byte cannot be read.
   N stands for an undefined count.
   Vertical lines show transitions between count values.

Figure B-6: 82C54 - Mode 0

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.1 Mode 0: Interrupt on Terminal Count (continued)

After the Control Word and initial count are written to a Counter, the initial count will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not go high until N+1 CLK pulses after the initial count is written.

If a new count is written to the Counter, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1) Writing the first byte disables counting. OUT is set low immediately (no clock pulse required).

2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

This allows the counting sequence to be synchronized by software. Again, OUT does not go high until N+1 CLK pulses after the new count of N is written.

If an initial count is written while GATE = 0, it will still be loaded on the next CLK pulse. When GATE goes high, OUT will go high N CLK pulses later; no CLK pulse is needed to load the Counter as this has already been done.

B.5.4.2 Mode 1: Hardware Retriggerable One-shot

OUT will be initially high. OUT will go low on the CLK pulse following a trigger to begin the one-shot pulse, and will remain low until the Counter reaches zero. OUT will then go high and remain high until the CLK pulse after the next trigger.

After writing the Control Word and initial count, the Counter is armed. A trigger results in loading the Counter and setting OUT low on the next CLK pulse, thus starting the one-shot pulse. An initial count of N will result in a one-shot pulse N CLK cycles in duration. The one-shot is retriggerable, hence OUT will remain low for N CLK pulses after any trigger. The one-shot pulse can be repeated without rewriting the same count into the counter. GATE has no effect on OUT.

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.2 Mode 1: Hardware Retriggerable One-shot (continued)

If a new count is written to the Counter during a one-shot pulse, the current one-shot is not affected unless the Counter is retriggered. In that case, the Counter is loaded with the new count and the one-shot pulse continues until the new count expires.

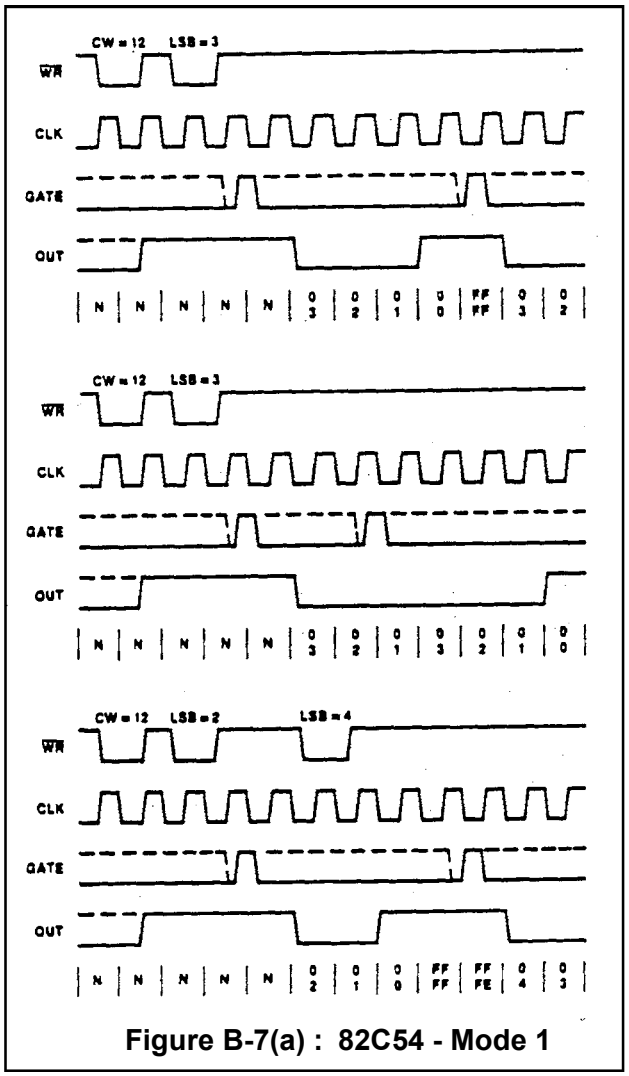
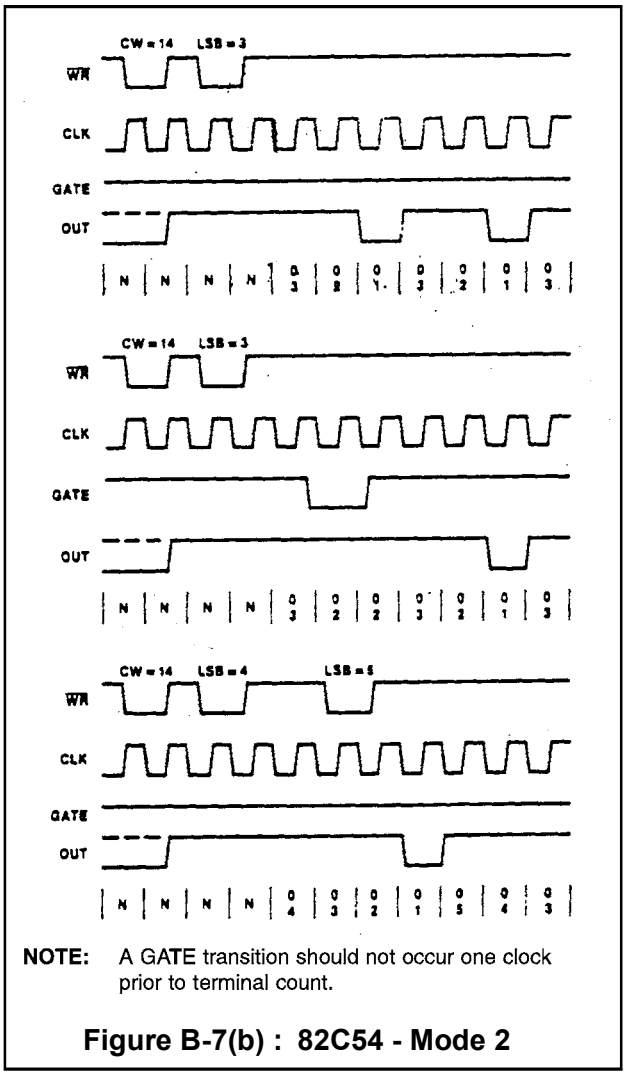


Figure B-7(a) : 82C54 - Mode 1



NOTE: A GATE transition should not occur one clock prior to terminal count.

Figure B-7(b) : 82C54 - Mode 2

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.3 Mode 2: Rate Generator

This Mode functions like a divide-by-N counter. It is typically used to generate a Real Time Clock interrupt. OUT will initially be high. When the initial count has decremented to 1, OUT goes low for one CLK pulse. OUT then goes high again, the Counter reloads the initial count and the process is repeated. Mode 2 is periodic; the same sequence is repeated indefinitely. For an initial count of N, the sequence repeats every N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low during an output pulse, OUT is set high immediately. A trigger reloads the Counter with the initial count on the next CLK pulse; OUT goes low N CLK pulses after the trigger. Thus the GATE input can be used to synchronize the Counter.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. OUT goes low N CLK pulses after the initial count is written. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current period, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current counting cycle. In mode 2, a COUNT of 1 is illegal.

B.5.4.4 Mode 3: Square Wave Mode

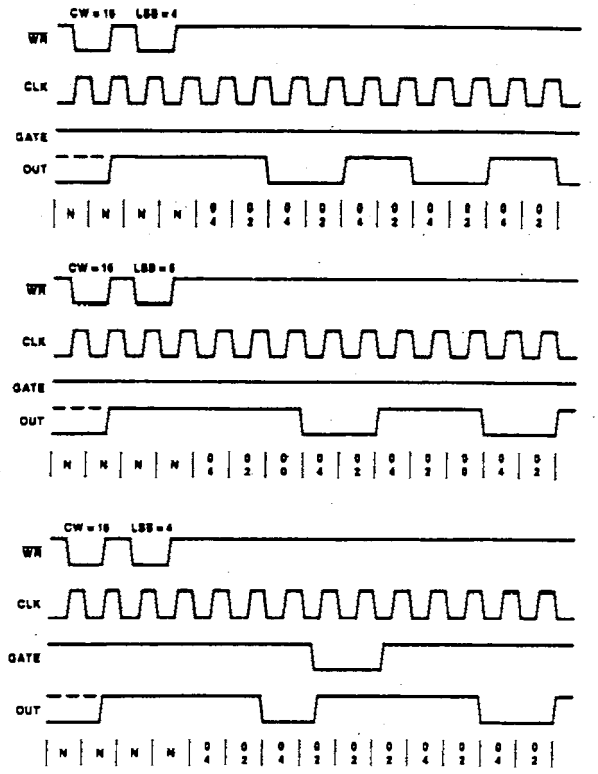
Mode 3 is typically used for Baud rate generation. Mode 3 is similar to Mode 2 except for the duty cycle of OUT. OUT will initially be high. When half the initial count has expired, OUT goes low for the remainder of the count. Mode 3 is periodic; the sequence above is repeated indefinitely. An initial count of N results in a square wave with a period of N CLK cycles.

GATE = 1 enables counting; GATE = 0 disables counting. If GATE goes low while OUT is low, OUT is set high immediately; no CLK pulse is required. A trigger reloads the Counter with the initial count on the next CLK pulse. Thus the GATE input can be used to synchronize the Counter.

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.4 Mode 3: Square Wave Mode (continued)



NOTE: A GATE transition should not occur one clock prior to terminal count.

Figure B-8: 82C54 - Mode 3

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This allows the Counter to be synchronized by software also.

Writing a new count while counting does not affect the current counting sequence. If a trigger is received after writing a new count but before the end of the current half-cycle of the square wave, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from the new count. Otherwise, the new count will be loaded at the end of the current half-cycle.

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.4 Mode 3: Square Wave Mode (continued)

Mode 3 is implemented as follows:

Even counts: OUT is initially high. The initial count is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. When the count expires OUT changes value and the Counter is reloaded with the initial count. The above process is repeated indefinitely.

Odd counts: OUT is initially high. The initial count minus one (an even number) is loaded on one CLK pulse and then is decremented by two on succeeding CLK pulses. One CLK pulse after the count expires, OUT goes low and the Counter is reloaded with the initial count minus one. Succeeding CLK pulses decrement the count by two. When the count expires, OUT goes high again and the Counter is reloaded with the initial count minus one. The above process is repeated indefinitely. So for odd counts, OUT will be high for (N+1)/2 counts and low for (N-1)/2 counts.

B.5.4.5 Mode 4: Software Triggered Strobe

OUT will be initially high. When the initial count expires, OUT will go low for one CLK pulse and then go high again. The counting sequence is "triggered" by writing the initial count.

GATE = 1 enables counting; GATE = 0 disables counting. GATE has no effect on OUT.

After writing a Control Word and initial count, the Counter will be loaded on the next CLK pulse. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N+1 CLK pulses after the initial count is written.

If a new count is written during counting, it will be loaded on the next CLK pulse and counting will continue from the new count. If a two-byte count is written, the following happens:

1) Writing the first byte has no effect on counting.

2) Writing the second byte allows the new count to be loaded on the next CLK pulse.

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.5 Mode 4: Software Triggered Strobe (continued)

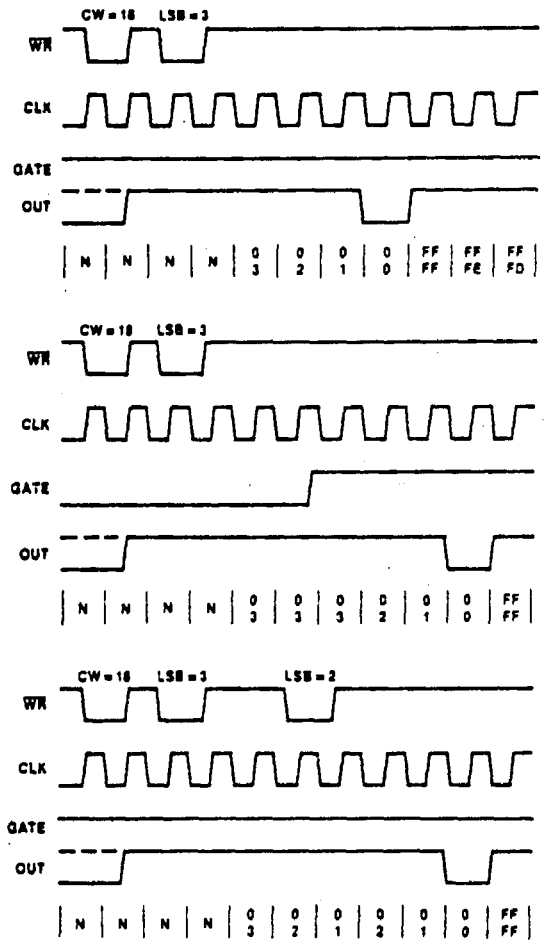


Figure B-9(a) : 82C54 - Mode 4

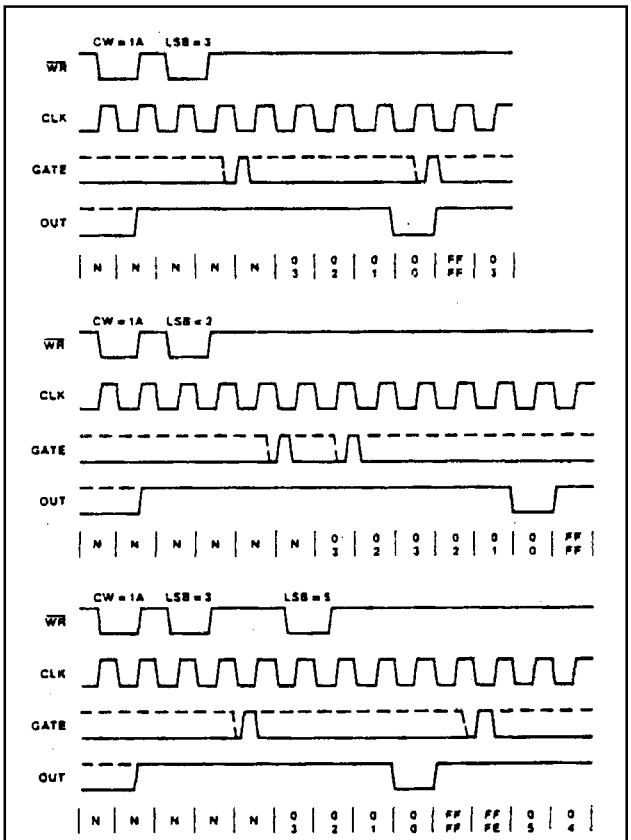


Figure B-9(b) : 82C54 - Mode 5

This allows the sequence to be "retriggered" by software. OUT strobes low N+1 CLK pulses after the new count of N is written.

B.5.4.6 Mode 5: Hardware Triggered Strobe (Retriggerable)

OUT will initially be high. Counting is triggered by a rising edge of GATE. When the initial count has expired, OUT will go low for one CLK pulse and then go high again.

B.5 Operational Description (continued)

B.5.4 Mode Definitions (continued)

B.5.4.6 Mode 5: Hardware Triggered Strobe (Retriggerable) (continued)

After writing the Control Word and initial count, the counter will not be loaded until the CLK pulse after a trigger. This CLK pulse does not decrement the count, so for an initial count of N, OUT does not strobe low until N+1 CLK pulses after a trigger.

A trigger results in the Counter being loaded with the initial count on the next CLK pulse. The counting sequence is retriggerable. OUT will not strobe low for N+1 CLK pulses after any trigger. GATE has no effect on OUT.

If a new count is written during counting, the current counting sequence will not be affected. If a trigger occurs after the new count is written but before the current count expires, the Counter will be loaded with the new count on the next CLK pulse and counting will continue from there.

B.6 Operation Common to All Modes

B.6.1 Programming

When a Control Word is written to a Counter, all Control Logic is immediately reset and OUT goes to a known initial state; no CLK pulses are required for this.

B.6.2 GATE

The GATE input is always sampled on the rising edge of CLK. In Modes 0, 2, 3, and 4 the GATE input is level sensitive, and the logic level is sampled on the rising edge of CLK. In Modes 1, 2, 3, and 5 the GATE input is rising-edge sensitive. In these Modes, a rising edge of GATE (trigger) sets an edge-sensitive flip-flop in the Counter. This flip-flop is then sampled on the next rising edge of CLK; the flip-flop is reset immediately after it is sampled. In this way, a trigger will be detected no matter when it occurs - a high logic level does not have to be maintained until the next rising edge of CLK. Note that in Modes 2 and 3, the GATE input is both edge- and level-sensitive. In Modes 2 and 3, if a CLK source other than the system clock is used, GATE should be pulsed immediately following $\overline{WR}$ of a new count value.

B.6 Operation Common to All Modes (continued)

B.6.3 Counter

New counts are loaded and Counters are decremented on the falling edge of CLK.

The largest possible initial count is 0; this is equivalent to 216 for binary counting and 104 for BCD counting.

The Counter does not stop when it reaches zero. In Modes 0, 1, 4, and 5 the Counter "wraps around" to the highest count, either FFFF hex for binary counting or 9999 for BCD counting, and continues counting. Modes 2 and 3 are periodic; the Counter reloads itself with the initial count and continues counting from there.

Table B-9(a): 82C54 - Gate Pin Operations Summary

Signal Status Modes	Low Or Going Low	Rising	High
0	Disables counting	-	Enables counting
1	-	1) Initiates counting 2) Resets output after next clock	-
2	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
3	1) Disables counting 2) Sets output immediately high	Initiates counting	Enables counting
4	Disables counting	-	Enables counting
5	-	Initiates counting	-

Table B-9(b): 82C54 - Minimum and Maximum Initial Counts

Mode	Min Count	Max Count
0	1	0
1	1	0
2	2	0
3	2	0
4	1	0

NOTE: 0 is equivalent to 2^{16} for binary counting and 10^4 for BCD counting

This page is intentionally blank

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP