

# **Professional** 300 series

Real-Time Interface Module Analog Data Module Technical Manual



The PC3XX-AA module, when used with the BCC10-03, BCC11-03, and BCC12-03 cables and a Class B peripheral device, has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. To be operated in a Class B environment, only peripherals certified to comply with the Class B limits may be attached to this module. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

The PC3XX-AB optional cable and connector pod has been verified to comply with the limits for a Class A computing device, pursuant to Subpart J of Part 15 of FCC Rules. Use of this cable and connector pod may result in interference to radio and TV reception in a home or residential environment.

The Analog Data Module has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

The specifications and drawings herein are the property of Digital Equipment Corporation and shall not be reproduced or used or copied in whole or in part as the basis for the manufacture or sale of items without written permission.

Copyright © 1985 by Digital Equipment Corporation.
All Rights Reserved.

The manuscript for this book was created using generic coding and, via a translation program, was automatically typeset on DIGITAL'S DECset Integrated Publishing System. Book production was done by Educational Services Development and Publishing in Bedford, MA.

INTEL is a registered trademark of Intel Corporation, Santa Clara, CA.

The following are trademarks of Digital Equipment Corporation:

CTI BUS	MASSBUS	Rainbow
DEC	PDP	RSTS
<b>DECmate</b>	P/OS	RSX
DECsystem-10	PRO/BASIC	Tool Kit
DECSYSTEM-20	PRO/Communications	UNIBUS
DECUS	Professional	VAX
<b>DECwriter</b>	PRO/FMS	VMS
DIBOL	PRO/RMS	VT
	PROSE	Work Processor
	PROSE PLUS	

# **Preface**

This manual provides technical information for understanding the PC3XX-AA Real-Time Interface Module and the Analog Data Module. It includes detailed register information necessary for assembly language programming. Chapters 1 through 6 describe the Real-Time Interface Module. Chapters 7 through 12 describe the Analog Data Module.

Installation information for the Real-Time Interface Module can be found in the PC3XX-AA Real-Time Interface Module Installation Guide (EK-PC3AA-IN).

Information for using the Real-Time Interface Module or the Analog Data Module with the Professional Real-Time Interface Library (PRTIL) software can be found in the PC3XX-AA Real-Time Interface Module Owner's Manual (EK-PC3AA-OM) and the Analog Data Module Owner's Manual (EK-ADMPC-OM).

Additional technical information about the Real-Time Interface Module and the Analog Data Module can be found in the Maintenance Print Sets for each product. The part numbers are listed below.

Real-Time Interface Module

MP-01713

Maintenance Print Set

MP-01985-01

Analog Data Module Maintenance Print Set

# Contents

CHAPTER 1	THE REAL-	TIME	INTER	RFACE	MODULE	
Overview						
<b>Functional Comp</b>	onents					
Interrupt Log	gic and Bus Inte	rface				
Serial Logic .						
IEEE Logic .						
Parallel Logic	e					
Self-Test RO!	м					
Register Overview	w					
Access to the Mo	dule					
CHAPTER 2	INTERRUPT	LOG	IC AN	D BUS	INTERFACE	ł
Theory of Operat	ion					
Interrupt Log	ie					1
Bus interface						
	ator					
Interrupt Status	Register					1-

CHAPTER 3	SERIAL LOGIC
Theory of Operat	tion
Programming In	formation
	ling Register 2
	lding Register 2
	Register 2
	egisters
	nd Register 2
	SLU 2
	ta
	Y Interrupt
	Lines
Modelli Colleton	
CHAPTER 4	IEEE-488 BUS LOGIC
Theory of Opera	tion
Control Line	s 3
	Lines
	DIOI through DIOS
	s I
	ures
	ommands 1
	ata Holdoff 4
	ed Holdoff 1
	formation
	ister
	tus Register 1
	mmand Register
	of Auxiliary Commands 4
	ass Through Register
	ister
	gister
	tatus Register6
	ask Registers
	atus Registers 6
	Register
	Register
Serial Foll B	register

CHAPTER 5	PARALLEL LOGIC	
Theory of Operatio	on	75
	es	77
	teristics	77
	teristics	77
	teristies	78
	rmation	78
	rination	78
		79
Control Registe	er	79
Mode Control C	Commands	
		*1
	Mode 1 or Mode 2	81
		82
		82
Mode 1 Operation .		82
Mode 1 Input F	Handshaking Signals	32
Mode 1 Input I	nterrupts	8
Mode 1 Output	Handshaking Signals	84
Mode 1 Output	Interrupts	8.
	•••••	8
	Output Handshaking Signals	*
	pts	87
	lodes	**
Notes on Connectin	ng External Devices for Handshaking Signals	*
Adies on Connectin	ig Date that Devices for Flameshaving English	C.M.
CHAPTER 6	SELF-TEST ROM LOGIC AND DIAGNOSTICS	
Theory of Operation	on	91
<b>Programming Info</b>	rmation	9:
<b>ROM Data Reg</b>	rister	9:
		9:
Loopback Connecto	or	93
		94
CHAPTER 7	THE ANALOG DATA MODULE	
Overview		q
• •	nalog Inputs 1	
True Infferential I	nput and Auto-Zeroing Circuit 1	130

	eamplifier with Autoranging	
Programmable Re	eal-Time Clock	11
Track-and-Hold C	'ireuitry	)1
<b>Triggering Modes</b>	il	11
Digital Input and	Output 10	12
IEEE and Serial	Line Ports 10	12
ADM Panel		13
	nel LEDs 10	
	Connectors10	
	Barrier Strip	
	t Barrier Strip 10	
	Larrier Strip	
	Connector 16	
	nation Cable Connector	
Serial Line U	nits 1 and 2 Connectors	
		1. )
Reset		1.5
Reset Analog Data Mod	ule Owner's Manual	15 16
Reset Analog Data Mod		15 16
Reset	ule Owner's Manual 16 ons 16	15 16
Reset Analog Data Mod	ule Owner's Manual	15 16
Reset Analog Data Mod Cabling Connection	ule Owner's Manual 16 ons 16	15 16 16
Reset Analog Data Mod Cabling Connection  CHAPTER 8  Converting Analog	THEORY OF OPERATION  B Data	15 16 16
Reset Analog Data Mod Cabling Connection	THEORY OF OPERATION	15 16 16
Reset	THEORY OF OPERATION  B Data	15 16 16
Reset	THEORY OF OPERATION  B Data  COLLECTING ANALOG DATA	15 16 16
Reset	Ule Owner's Manual 16 ons 16 THEORY OF OPERATION  g Data 16  COLLECTING ANALOG DATA	15 16 16 19
Reset	THEORY OF OPERATION  g Data 16  COLLECTING ANALOG DATA  17  18  19  19  19  10  10  10  10  10  10  10	15 16 16 19
Reset	TEORY OF OPERATION  COLLECTING ANALOG DATA  es 11 1 and Multiple Samples 11	15 16 16 17 17 18
Reset	THEORY OF OPERATION  COLLECTING ANALOG DATA  es 11 and Multiple Samples 11 gers 12	15 16 16 17 17 18 19
Reset	THEORY OF OPERATION  g Data  COLLECTING ANALOG DATA  es 11 1 and Multiple Samples 11	15 16 16 16 17 17 18 19 20
Reset	THEORY OF OPERATION  COLLECTING ANALOG DATA  es 11 and Multiple Samples 11 gers 11 ing	15 16 16 17 17 18 19 20 24
Reset	tule Owner's Manual 16  THEORY OF OPERATION  g Data 16  COLLECTING ANALOG DATA  es 17  and Multiple Samples 17  gers 17  ing 18  lock Frequency 17  18	15 16 16 17 17 18 19 20 24 25
Reset	ule Owner's Manual 16 ons 16  TMEORY OF OPERATION  g Data 16  COLLECTING ANALOG DATA  es 17 and Multiple Samples 17 gers 17 ing 18 ock Frequency 17 uisition 17	15 16 16 17 17 18 19 20 24 25 26
Reset	ule Owner's Manual 16 but ons 16  TMEORY OF OPERATION  g Data 16  COLLECTING ANALOG DATA  es 17 and Multiple Samples 17 ing 18 lock Frequency 17 ulsition 17 Autoranging 18	15 16 16 17 17 18 19 20 24 25 26 26
Reset	ule Owner's Manual 16 ons 16  TMEORY OF OPERATION  g Data 16  COLLECTING ANALOG DATA  es 17 and Multiple Samples 17 gers 17 ing 18 ock Frequency 17 uisition 17	15 16 16 17 17 18 19 20 24 25 26 28

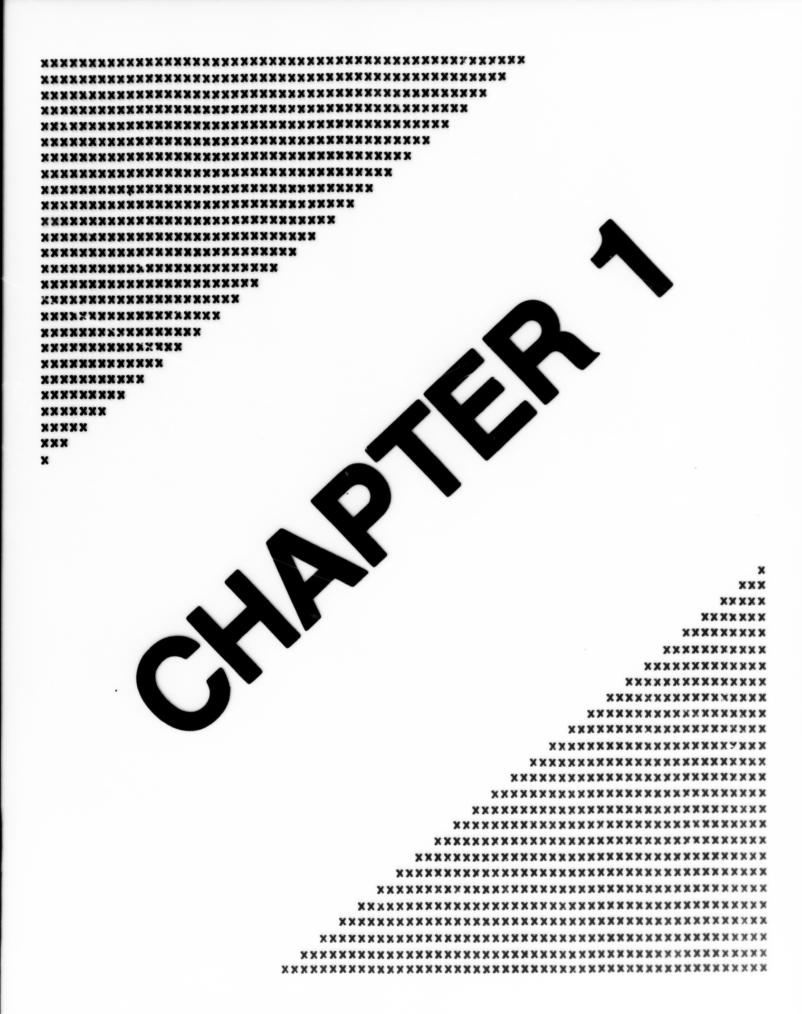
CHAPTER 10	USING THE DIGITAL INPUT/OUTPUT BARRIER STR	IF3
Making Connection The Digital Input I Digital Input I Using the Digi Using Switch C Using the Digi Collecting Digi	ns to the Barrier Strips Barrier Strip Barrier Strip Logic tal Input Barrier Strip for External Triggering Closures for External Triggering tal Input Barrier Strip for Status ital Data	133 134 134 134 135 136
Using the Digital (	Output Barrier Strip	137
CHAPTER 11	PROGRAMMING THE ADM	
Analog and Digita Writing the Co Analog Data Form Coding Format Status Byte Error Bits and Digital Output Mo	he Real-Time Interface Module I Data Acquisition Mode ommand Bytes and Collecting Data nat  Error Conditions de alibration  CALIBRATION	141 144 150 151 152 153 154 155
Offset Calibrat Gain Calibrati	alibrationtionon	159 159 161
APPENDIX A	PERFORMANCE AND ELECTRICAL SPECIFICATION	
Analog Performan A/D Converter Sy Input Multiplexer Programmable Ga Coding Format Digital Performan	ations ce stem in Preamplifier	165 165 166 166 167 168
	sts	

	•••••	
APPENDIX B	ADM SEQUENCE OF EVENTS	
FIGURES		
Figure 1-1: RTI S	implified Block Diagram	.,
	nal Cable Connector	
	X-AB Cable and Connector Pod	
	rupt Logic and Bus Interface	
	Diagram	12
	Interrupt Status Register	
	e Holding Register	
	mit Holding Register	
Figure 3-4: SLUS	Status Register	23
	Mode Registers	21
Figure 3-6: SLU (	'ommand Register	26
Figure 4-1: IEEE	Logic Simplified Block Diagram	36
Figure 4-2: IEEE	-188 Bus Lines	37
Figure 4-3: Addre	ss Register	11
Figure 4-4: Addre	ss Status Register	1.5
Figure 1-5: Auxili	ary Command Register	17
Figure 4-6: Comm	and Pass Through Register	JA.
Figure 4-7: Data-	In Register	59
Figure 4-8: Data-	Out Register	(61)
Figure 4-9: Interf.	ace Command Format	62
Figure 4-10: Inter	face Commands	63
Figure 4-11: IEEE	E Bus Status Register	64
	rupt Mask Registers 0 and 1 (WRITE ONLY)	64
Figure 4-13: Inter	rupt Status Registers 0 and 1 (READ ONLY)	67
Figure 4-14: Para	llel Poll Register	
Figure 4-15: Seria	d Poll Register	71
Figure 5-1: Parall	lel Logic Block Diagram	76
	A Register	78
Figure 5-3: Port I	B Register	TH
	Register	
Figure 5-5: Contro	ol Register	79

Figure 5-6: Handshaking Timing Diagram Input	83
Figure 5-7: Handshaking Timing Diagram Output	
Figure 6-1: Self-Test ROM Logic Block Diagram	
Figure 6-2: ROM Data Register	93
Figure 6-3: ROM Address Register	
Figure 7-1: The Analog Data Module	
Figure 7-2: The ADM Panel	
Figure 8-1: ADM Block Diagram	
Figure 8-2: Successive Approximation Block Diagram	
Figure 10-1: Example of Switch Debouncing	
Figure 11-1: RTI/ADM Handsh: king Timing Diagram	
Figure 11-2: Data Flow from the ADM to the RTI	151
TABLES	
Table 2-1: Bus Interface Signals	13
Table 2-2 IRQ B Interrupt Status Register Contents	15
Table 3-1: SLU Register Addresses	21
Table 3-2: SLU Status Register Data	23
Table 3-3: SLU Mode 1 Register Data	
Table 3-4: SLU Mode 2 Register Data	
Table 3-5: SLU Command Register Data	
Table 4-1: IEEE-488 Logic Read Registers	
Table 1-2: IEEE-488 Logic Write Registers	13
Table 1-3: Address Register	
Table 1-4: Address Status Register Data	
Table 4-5: Auxiliary Command Register Data	
Table 4-6: SCEN and SC Bits	
Table 4-7: Interrupt Mask Register 0 Data	6.5
Table 4-8: Interrupt Mask Register 1 Data	
Table 4-9: Interrupt Status Register 0	
Table 4-10: Interrupt Status Register 1	69
Table 5-1: Control Register Data - Mode Control Commands	*()*
Table 5-2: Control Register Port C Bit Set/Reset Commands	×1
Table 5-3: Location of Handshaking Signals in Port C-Modes 1 and 2	××
Table 9-1: ADM Clock Source Frequencies.	122
Table 11-1: PC7 and PC6 Meanings	
Table 11-2: Commands to Set or Reset PC6 and PC7	143
Table 11-3: Command Bytes Overview	
Table 11-4: Command Byte 0	

Table	11-5	Command Byte 1	١.																 	 		143
Table	17-6:	Command Byte 2	2 .											. ,					 			 146
Table	11-7:	Command Byte 3	3 .																 			 149
Table	11-8:	Coding Format .																	 			 152
Table	11-9:	ADM Status Byt	e																			 15.
Table	A-1:	<b>Autoranging Scal</b>	e	in	it	te	h	P	0	iı	nt								 			 167





# 

The Real-Time Interface Module

# Chapter 1

# The Real-Time Interface Module

### **OVERVIEW**

The Real-Time Interface Module (RTI) enables a Professional 300 series computer to communicate with remote devices in a real-time environment. It consists of the module itself (part number 54-15539-01) and an internal cable (part number 17-00404-01), and is mounted in the sixth physical slot (CTI Bus Slot 5) at the rear of the computer's card cage.

The RTI contains three separate interfaces that operate independently and can interrupt the Professional computer's CPU. An interrupt status register is provided to identify which interface interrupted the CPU. The three interfaces are:

- □ Serial—This interface consists of two asynchronous Serial Line Units that conform to the EIA RS-232C/RS-423 standard. Each is capable of independent full-duplex or half-duplex operation with user-selectable baud rate, character length, number of stop bits, and parity generation and detection.
- □ IEEE-488—This interface consists of a byte-serial, bit parallel bus interface that conforms to IEEE Standard 488-1978. Standard IEEE-488 protocol is handled automatically in talker, listener, or controller operational modes. Every IEEE-488 device interface function is implemented.

Parallel - This interface consists of targe 8-bit ports that can be
programmed for a wide variety of input and output combinations.
Three modes of input/output (I/O) operation are provided. Mode (
allows general input and output without handshaking or interrupt
capabilities. Mode 1 allows both handshaking capabilities and
interrupts, if desired, through two 8-bit ports. Mode 2 allows
bidirectional I/O through one 8-bit port with a four-wire
handshaking scheme.

### **FUNCTIONAL COMPONENTS**

The Real-Time Interface Module (RTI) consists of the following five functional components:

The Interrupt Logic and Bus Interface
The Serial Logic
The IEEE Logic
The Parallel Logic
The Self-Test ROM

The functional components are summarized below and shown in Figure 1-1, RTI Simplified Block Diagram.

# Interrupt Logic and Bus Interface

The interrupt logic receives interrupts from the three I/O interfaces and transfers either an IRQ A or an IRQ E signal to the CTI Bus. An IEEE interrupt becomes an IRQ A signal, and all other interrupts become an IRQ B signal. When the RTI is addressed by the CPU, the interrupt logic responds with the BRPLY signal.

The bus interface receives control signals from the CTI bus, translates them, and transfers them to the other four functional components. It also serves as a bidirectional buffer for addresses and data.

## Serial Logic

The serial logic contains two independent serial line units (SLUs). The SLUs communicate with remote devices according to EIA RS-232/423 protocols. Parameters and baud rate of the serial data can be specified by your program. The SLUs generate interrupts upon completion of each transmitted and received message.

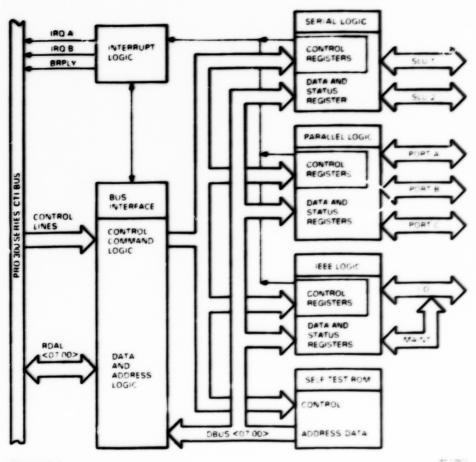


Figure 1-1 RTI Simplified Block Diagram

## **IEEE Logic**

The IEEE logic allows the RTI to be configured as the system controller, controller-in-charge, talker, or listener on an IEEE-488 bus. The IEEE logic provides interrupt driven data transfer and allows handshaking lines to be controlled to allow data processing during interrupts. An auxiliary command register is provided to simplify some of the commonly used functions in an IEEE-488 environment.

# **Parallel Logic**

The parallel logic contains three 8-bit parallel input/output ports. There are three modes of operation, providing for data transfer either with or without handshaking signals. Interrupts are available in the second and third mode of operation (mode 1 and mode 2) and can be enabled or disabled.

### Self-Test ROM

The self-test ROM contains a 4K byte diagnostic program that tests the L/O interfaces. It is activated automatically each time that power is applied to the system in which the RTI is installed. The ROM does not generate interrupts itself, but uses the L/O interfaces to signal the CPU when diagnostic tests are completed.

### **REGISTER OVERVIEW**

Table 1-1 lists the registers and register groups that exist on the Real-Time Interface Module and their addresses.

Table 1-1 RTI Registers

Address	Function
17775200	ROM Data Register
17775202	ROM Address Register
17775206	Interrupt Status Register
17775210 - 17775216	Parallel Logic Register Group
17775220 - 17775236	IEEE Logic Register Group
17775240 - 17775246	Serial Logic SLU 1 Register Group
17775250 - 17775256	Serial Logic SLU 2 Register Group
17775260 - 17775276	IEEE Maintenance Register Group

NOTE: Although the RTI registers can contain 16 bits, only the low byte (bits 7-0) is significant in all cases because the RTI is not connected to bits 15-8 of the CTI BDAL bus. When you write to a register, the high byte (bits 15-8) is ignored.

## **ACCESS TO THE MODULE**

Cabling connections to the RTI are made through the internal cable's connector. This 62-pin connector is on the back of the Professional computer's chassis and allows connection of one of four optional cables. The pins available in the internal cable connector are shown in Figure 1-2.

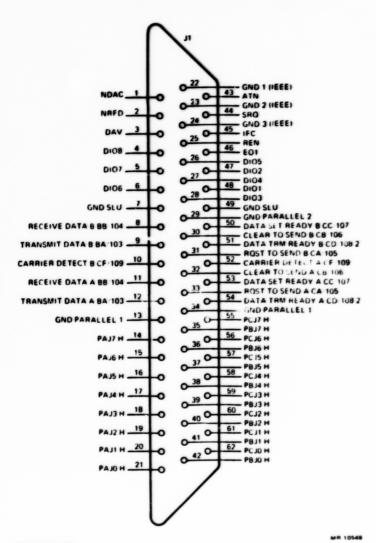


Figure 1-2
Internal Cable Connector

The optional cables available should be chosen according to which of the interfaces you plan to use. To use all of the interfaces simultaneously, the PC3XX-AB cable and connector pod should be used. This pod has two 25-pin D-connectors for the serial line units, an IEEE-488 connector, and a 30-pin barrier strip for access to the parallel I/O ports. This cable and pod is shown in Figure 1-3. Other cables for use of one interface at a time are also available. The BCC10 cable allows use of the serial line units. The BCC11 cable allows use of the IEEE-488 interface. The BCC12 cable allows use of the programmable parallel port. The PC3XX-AA Owner's Manual contains complete details on these cables.

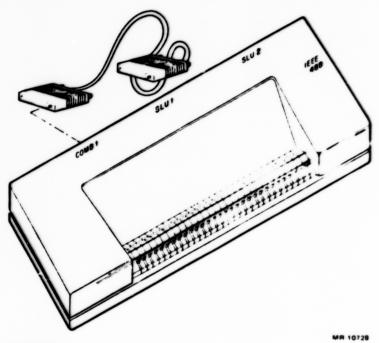
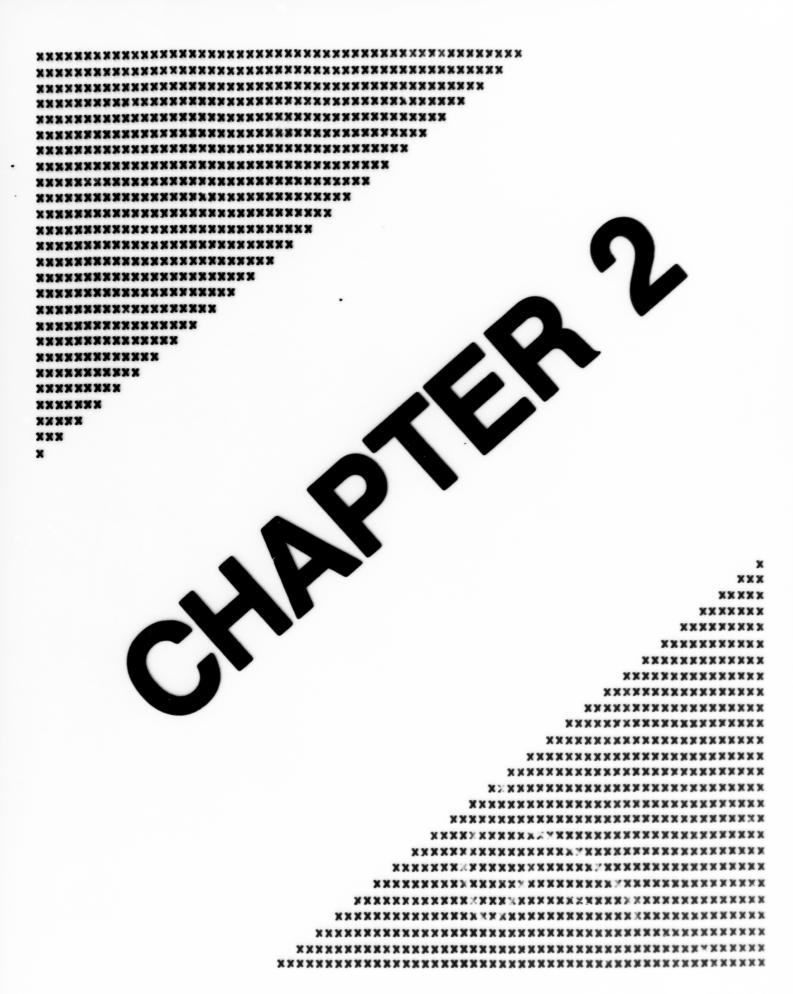


Figure 1-3
PC3XX-AB Cable and Connector Pod



# 2

Interrupt Logic and Bus Interface

# Chapter 2

# Interrupt Logic and Bus Interface

## THEORY OF OPERATION

Figure 2-1, Interrupt Logic and Bus Interface Simplified Block Diagram, summarizes the operation of the interrupt logic.

# Interrupt Logic

An interrupt from the IEEE Logic causes the RTI to generate an IRQ A interrupt to the CPU. Any other interrupt (serial or parallel interrupt) generates an IRQ B interrupt. The serial logic generates an interrupt whenever either serial line unit completes a transmit or receive function. The parallel logic generates an interrupt only under certain conditions. The parallel I/O interrupt decoder generates a PAR I/O INTA signal when the INTEA 1, PC 3, and I/O WRT signals are true simultaneously, and generates a PAR I/O INTB signal when the INTEB 1, PC 0, and I/O WRT signals are true simultaneously.

When the RTI is addressed by the CPU, the combination of the BSLOT SEL and DATA STB signals being true causes the interrupt logic to send the BRPLY signal back to the CPU as an acknowledgment.

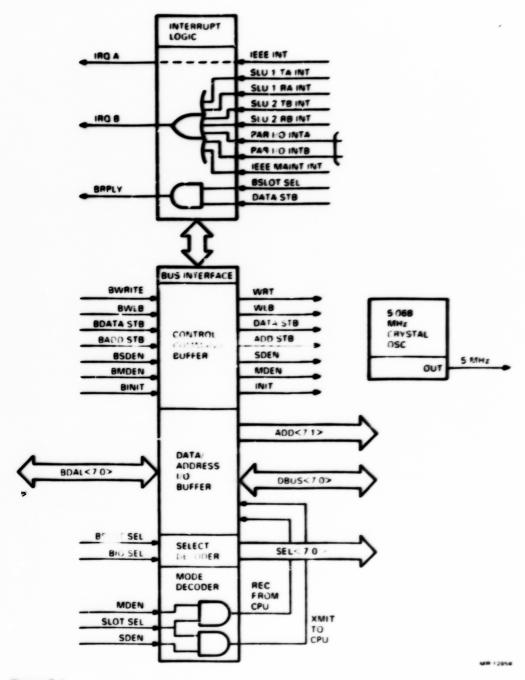


Figure 2-1
Interrupt Logic and Bus Interface Simplified Block Diagram

# **Bus** Interface

The control command buffer accepts command signals from the CTI bus and transfers them to the serial, IZEE, and parallel logic interfaces and to the self-test ROM. The bus signals are described briefly below.

Table 2-1 Bus Interface Signals

Signal	Name	Function
BWRITE	Bus Write	Becomes WRT within the RTI used by the serial and parallel logic and the self test ROM
BWLB	Bus Write Low Byte	Used only by the IEEE logic as a write enable signal
BDAT4 STB	Bus Data Strobe	Used by the CPU to tell the bus interface (and other Professional computer options on the bus) that the information on the buffered data/address lines is data. It is also used by the serial and parallel logic to distinguish between addresses and data.
BADD S1B	Bus Address Strobe	Used by the CPU to tell the bus interface (and other Professional computer options on the bus) that the information on the buffered data/address lines is an address. This combination of signals generates the address signals (ADD 1 through ADD 7) for the rest of the logic on the RTI
BSDEN	Bus Slave Drive Enable	Used by the IEEE and parallel logic to configure them to receive data from remote devices and make the data available to the bus interface. It is also used by the mode decoder section of the bus interface in combination with BSLOT SEL to place the data/address I/O buffer into the transmit mode to transfer data to the CPU. This combination enables the transfer of data from remote devices to the CPU.
BMDEN	Bus Master Data Enable	Used only by the mode decoder section of the bus- interface, in combination with BSLOT SEL to place the data/address I/O buffer into the receive mode to transfer data from the CPU.

Table 2-1 Bus Interface Signals (Cont.)

Signal	Name	Function
BINIT	Initialize	Asserted automatically by the CPU when the Professional computer is first turned on It sets the RTI (and the rest of the Professional computer) to a known state. This signal is also asserted by a software RESET instruction.

NOTE: Use of the RESET instruction is not recommended since it may disrupt program execution, adversely affect system operation, and cause the loss of data.

BDAL 0 through BDAL 7	Buffered Data/Address Lines	Connected to a bidirectional three state transceiver. Within the RTI, the same lines are labeled DBUS 0 through DBUS 7.
<b>BSLOT SEL</b>	<b>Bus Slot Select</b>	Used together with:
BIO SEL	Bus Input/Output Select	and the ADD 3, 4, 5, and 6 signals to generate the SELect 0 through 7 signals that are used by all the logic in the RTI

# **Crystal Oscillator**

The RTI contains a 5.0688 MHz crystal oscillator to develop its internal timing signals. The buffered output is at oscillator frequency and is labeled 5 MHz. The clock frequency is not user addressable and cannot be changed. The clock is used to develop timing for the SLU's baud rates; these baud rates are user selectable.

# INTERRUPT STATUS REGISTER

An interrupt status register is provided for the IRQ B interrupts. When an IRQ B interrupt occurs and this register is read, the register will identify the section of the RTI that is requesting service. See Figure 2-2.

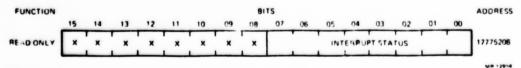


Figure 2-2
IRQ B Interrupt Status Register

Table 2-2 IRQ B Interrupt Status Register Contents

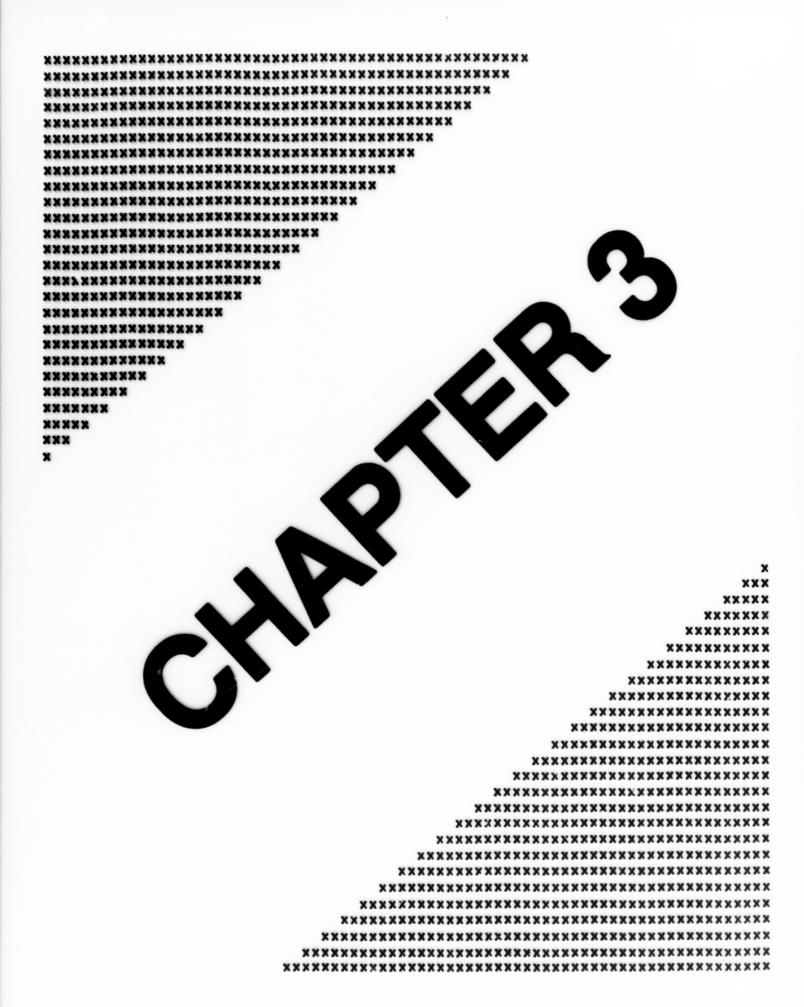
Bit Assignments		Interrupt	
	0	NO IRQ B INTERRUPT	
	2	IEEE MAINTENANCE	
	4	PARALLEL PORT B	
Bits	6	PARALLEL PORT A	
7 through 0	10	SLU 2 RECEIVE	
	12	SLU 2 TRANSMIT	
	14	SLU 1 RECEIVE	
	16	SLU 1 TRANSMIT	

Note that the coding of the interrupts lends itself easily to the construction of a table within your program for branching. The table should contain the branch addresses for the portions of your program devoted to handling each I/O interface. The contents of the interrupt status register can be added directly to the base address of the table to obtain the proper branching address.

The IEEE port interrupts the CPU on the IRQ A line. This has priority over IRQ B and is dedicated to the IEEE port only. All other interrupts occur on the IRQ B line.

The IRQ B logic is driven by a ring counter (not shown in Figure 2-1) at 2.5344 MHz. It continuously samples the inputs until it finds one asserted. Then the counter stops, loads the interrupt status register with the device code, and asserts the IRQ B interrupt to the Professional computer's CPU. The CPU must read the IRQ B interrupt status register to determine which interface is requesting service. Once the interface is serviced, it deasserts its interrupt request line to the IRQ B logic, and the logic resumes cycling through the inputs. This method permits only one interrupt to occur at a time and gives each interface an equal opportunity to request service from the Professional computer's CPU.





# 

Serial Logic

# Chapter 3

# Serial Logic

### THEORY OF OPERATION

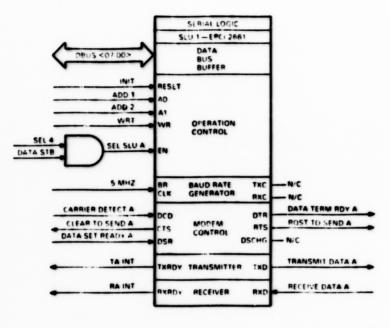
Figure 3-1, Serial Logic Simplified Block Diagram, summarizes the operation of the serial line units.

There are two serial line units (SLUs), labeled SLU 1 and SLU 2, within the serial logic section of the RTI; they are programmed and operate independently. Each uses a Signetics 2661 Enhanced Programmable Communication Interface (EPCI) chip plus EIA drivers and receivers. The EPCI chip contains its own I/O buffers and shift registers for controlling asynchronous character protocol. This chip converts parallel data characters from the DBUS into serial data for transmission to remote devices. In addition, it converts serial data from remote devices into parallel data for CPU processing.

Both SLUs are controlled by the INIT, ADD 1, ADD 2, and WRT signals received from the bus interface. SLU 1 is selected by the logical AND of the SEL 4 and DATA STB signals; SLU 2 is selected by the logical AND of the SEL 5 and DATA STB signals from the bus interface.

The 2661 chip's baud rate generator receives an input of 5.0688 MHz from the oscillator on the RTI and can be programmed to operate the SLU at speeds from 50 to 9600 haud.

The modem control logic receives CARRIER DETECT, CLEAR TO SEND, and DATA SET READY handshaking signals from remote devices, and transmits DATA TERMINAL READY and REQUEST TO SEND handshaking signals. Refer to the PC3XX-AA Real-Time Interface Module Owner's Manual (EK-PC3AA-OM) for a discussion of EIA RS-232C/423 handshaking protocols.



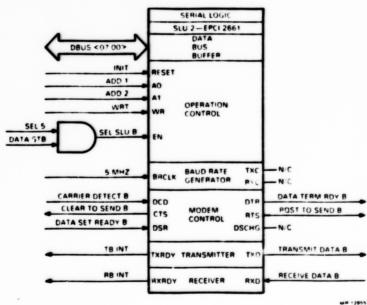


Figure 3-1
Serial Logic Simplified Block Diagram

The transmitter logic is used to send serial data to a remote device and generates a TA or TB INTERRUPT when it is ready to accept another byte from the Professional computer's CPU. Details on the generation of this interrupt signal are described below.

When it has received a complete byte from a remote device and is ready to be read by the CPU, the receiver logic generates an RA or RB INTERRUPT. Details on the generation of this interrupt signal are described below.

The receiver and transmitter are specifically enabled or disabled by your program. Both full-duplex and half-duplex operations are available. Full-duplex operation occurs when both the transmitter and the receiver are active simultaneously. Half-duplex operation occurs when only one (either the receiver or the transmitter) is active.

The SLU's transmitter and receiver conform to EIA RS-232/432 standard with respect to signal levels and to the RS-232/432 handshaking signals as described below.

### PROGRAMMING INFORMATION

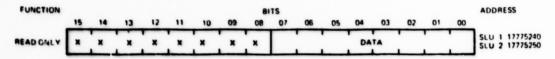
The SLUs contain holding, status, mode, and command registers. Table 3-1 lists all the serial line unit registers and addresses.

Table 3-1 SLU Register Addresses

Address	Register	
17775240	SLU 1 Receive/Transmit Holding Register	
17775242	SLU 1 Status Register	
17775244	SLU 1 Mode Register	
17775246	SLU 1 Command Register	
17775250	SLU 2 Receive/Transmit Holding Register	
17775252	SLU 2 Status Register	
17775254	SLU 2 Mode Register	
17775256	SLU 2 Command Register	

# Receive Holding Register

The receive holding register (Figure 3-2) is an 8-bit read-only register that contains the data being received by the SLU. The SLU receiver assembles incoming characters one bit at a time. When a full character is assembled, the receiver places the data in this register. When the register is read, the RxRDY signal is deasserted. Bit 1 clears in the SLU status register.



---

Figure 3-2 Receive Holding Register

# **Transmit Holding Register**

The transmit holding register (Figure 3-3) is an 8-bit double-buffered writeonly register for the data being transmitted by the SLU. Its associated shift register is not directly addressable. Data is written into the transmit holding register by the program. If the transmitter is enabled, and the clear-to-send modem control signal is true, the data is automatically moved (in parallel, that is, all eight bits at once) to the SHIFT register where it is transmitted serially to remote devices. See the section Transmitting Data.

Note that the transmit holding register and the receive holding register are at the same address. When the address is written to, the register is used as the transmit holding register. When the address is read, the register is used as the receive holding register.

Both the receiver and transmitter can be enabled simultaneously (full-duplex operation); it is possible to use both registers simultaneously.

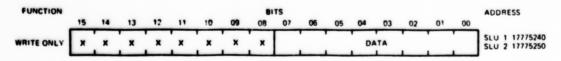


Figure 3-3 Transmit Holding Register

# **SLU Status Register**

The SLU status register is shown in Figure 3-4.

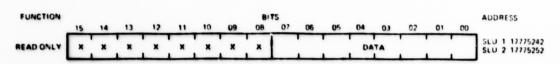


Figure 3-4 SLU Status Register

Table 3-2 SLU Status Register Data

Bit Number	Value	Significance
0	0	Transmitter Holding Register Busy
	1	Transmitter Holding Register Empty (Data can be written to it.)
1	0	Receiver Holding Register Empty
	1	Receiver Holding Register contains data which can be read
2	0	Transmitter Shift Register is not empty
	1	Transmitter Shift Register is empty or a change has occurred in one of the input modem control $\lim_{t\to\infty}$
3	0	No Error
	1	Parity Error
4	0	No Error
	1	Overrun Error (A new received data byte was written into the receive holding register before the previous data byte was read.)
5	0	No Error
	1	Framing Error (missing stop bit on receive)
6	0	Carrier Detect input is False
	1	Carrier Detect input is True (Asserted)
7	0	Data Set Ready input is False
	1	Data Set Ready input is True (Asserted)

Bits 0 and 1 indicate the current status of the TxRDY signal and the RxRDY signal, respectively. These signals are true or false depending on whether the transmit holding register or the receive holding register currently has data in it. For details, see Transmitting Data and Receiving Data.

When set, bit 2 in the status register means one of two things. When either the transmitter or receiver is enabled, and a change occurs in either the incoming carrier detect or incoming data set ready signal, bit 2 becomes set. The change can be either low to high or high to low. Bit 2 also indicates the current status of the transmit holding register when the transmitter is enabled. If set, bit 2 indicates that the transmit holding register is empty. When the status register is read or a data byte is written into the transmit holding register, bit 2 is cleared.

The error indicators (bits 3, 4, and 5) are cleared when the next received word enters the receive holding register, or when they are specifically reset by bit 4 in the command register. The other bits in the status register are set or cleared as the conditions they monitor change. For example, bit 7 remains set as long as DSR remains asserted and is cleared only when DSR drops low.

# **SLU Mode Registers**

The mode registers (Figure 3-5) allow you to specify the parameters of data transmission. There are two mode registers, mode 1 and mode 2, in each SLU. An internal pointer is set to the mode 1 register by the INIT signal at power-up or by reading the command register. Each time a mode register is written or read, the pointer toggles to the other register. When you first set the data parameters, the mode 1 register should be written first. See Tables 3-3 and 3-4 for the mode register data.

NOTE: INIT clears the mode, command, and status registers, and puts the SLU into the idle state.

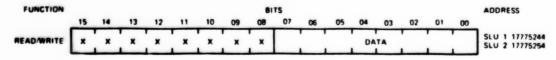


Figure 3-5 SLU Mode Registers

Table 3-3 SLU Mode 1 Register Data

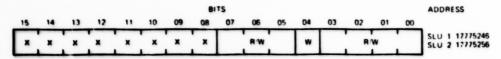
Bit Number	Value	Significance
1-0	01	Always (forces asynchronous operation)
3-2		These bits determine character length
	00	5 Bits
	01	6 Bits
	10	7 Bits
	11	8 Bits
4	0	Parity disabled
	1	Parity enabled
5	0	Odd parity enabled
	1	Even parity enabled
7-6		These bits determine the number of stop . Is
	00	Invalid
	01	1 Stop Bit
	10	1 1/2 Stop Bits (receive only)
	11	2 Stop Bits

Table 3-4 SLU Mode 2 Register Data

Bit Number	Value	Significance
3-0		These bits select the baud rate
	0000	50
	0001	75
	0010	110
	0011	134 5
	0100	150
	0101	300
	0110	600
	0111	1200
	1000	1800
	1001	2000
	1010	2400
	1011	3600
	1100	4800
	1101	7200
	1110	9600
7-4	0011	Always (Causes the SLU to generate the baud rate internally, using the 5 0688 MHz clock)

# **SLU Command Register**

The SLU command register is shown in Figure 3-6.



MR 12000

Figure 3-6 SLU Command Register

Table 3-5 SLU Command Register Data

<b>Bit</b> Number	Value	Significance
U	1	Transmitter Enabled
	0	Transmitter Disabled
1	1	Forces Data Terminal Ready (DTR) True
	0	Forces Data Terminal Ready (DTR) False
2	1	Receiver Enabled
	0	Receiver Disabled
3	0	Normal Operation
	1	Force Break Condition
4	0	Normal
	1	Resets Error Flags in Status Register
5	0	Forces the Request-To-Send (RTS) output False one clock time after the transmit shift register is emptied.
	1	Forces the Request-To-Send (RTS) output True.
7-6		These bits control the operating mode.
	00	Normal Operation
	01	Automatic Echo Mode
	10	Local Loopback
	11	Remote Loopback

Bits 0 and 2 enable the transmitter and receiver, respectively, if set. See Transmitting Data and Receiving Data.

Bits 1 and 5 control the output modem control signals used by the SLU. See Figure 1-2 for the location of these signals on the internal cable connector. See Modem Control for a description of the purpose of each signal.

Bit 3, when set, puts the data transmit line into a break condition (continual low). The data line remains low until this bit is cleared.

Bit 4, when set, clears the error flags (bits 3, 4, and 5) in the status register. Unlike the other bits in the command register, bit 4 is not latched. When you set bit 4, the status register error flags are cleared; bit 4 then clears itself. When the command register is read, bit 4 always reads zero.

Automatic echo, local loopback, and remote loopback are used for diagnostic testing of the SLU. When automatic echo is selected, all data received is automatically placed in the transmit holding register as well as in the receive holding register. The receiver must be enabled for this mode, but the transmitter does not need to be enabled. The transmitter transmits the received data as it is placed in the transmit holding register by the receiver. The TxRDY line remains continually asserted. You can still read the incoming data by accessing the receive holding register. Remote loopback is the same as automatic echo except that you cannot access the incoming data in the receive holding register. You can however, monitor the error bits in the status register. In local loopback, data you place in the transmit holding register is also fed into the receiver and placed back in the receive holding register.

#### INITIALIZING THE SLU

There is no special initialization command for the SLUs. However, you should specify the parameters of the transmitted or received data by writing to the mode registers before writing anything into the command register. In addition, you may want to monitor the status register before writing to the command register to check the status of the incoming modem control lines. Both the transmitter and receiver require their respective handshaking line to be asserted low before they will be operative.

#### TRANSMITTING DATA

The following steps are necessary to transmit data through either serial line unit. First, set the transmit enable (TxEN) bit in the status register. Write the first data byte to the transmit holding register. This clears bit 0 in the status register. When the clear-to-send modem signal is received true from the receiving device, the transmitter becomes operative and the data byte is auto-

matically transferred to the shift register. The transmitter is operative when both clear-to-send and TxEN are true. When the transmitter is operative, the contents of the shift register are transmitted.

When the data byte is moved (in parallel, that is, all eight bits at once) to the shift register, TxRDY is asserted low, meaning that the transmitter is operative and the holding register is empty. This generates a B interrupt in the interrupt logic. A new data byte can be written to the holding register.

The transmitter automatically sends a start bit before each data byte. The least significant bit of the data is sent first. After the most significant bit, a parity bit is sent, if parity is enabled. The specified number of stop bits are then sent. Transmission always begins as soon as the shift register is full, if the transmitter is enabled. If after sending the contents of the shift register there is no data in the holding register, the transmit output line of the serial line unit (pin 12 on J1 for SLU 1 and pin 9 on J1 for SLU 2) remains high. If you want to keep the transmit line in a continuous low state, set the send-break bit (bit 3) in the command register. The send-break bit must be cleared before data transmission can resume.

When the shift register is empty, the next data byte (assuming it has been written into the holding register) is automatically transferred from the holding register if the transmitter is still operative. If the holding register is empty, TxRDY stays true (low). If the holding register is not empty, the data is transferred to the shift register.

The transmitter becomes inoperative whenever the shift register is empty, and either the clear-to-send signal becomes false, or the TxEN bit is cleared. Therefore, if the clear-to-send signal becomes low while data is in the shift register, that data is allowed to transmit to completion. However, as soon as the shift register is empty, the transmitter becomes inoperative and TxRDY is not asserted true. If the shift register is currently empty and the clear-to-send signal becomes false, the transmitter is disabled immediately.

#### DISABLING TXRDY INTERRUPTS

Under the sequence described above, the TxRDY interrupts are generated after every data byte, including the last data byte, is transferred from the holding register. When TxRDY is true (low), B interrupts are generated and remain asserted until TxRDY is false. To disable this interrupt after the last data byte has been sent, wait until the last TxRDY is true, meaning that the last data byte is in the shift register. Clear the TxEN bit (bit 0 in the status register).

The transmitter will be disabled as soon as the contents of the shift register have been transmitted. Write a dummy byte into the holding register. This will cause TxRDY to go false and stay false when the transmitter actually becomes inoperative.

To enable the TxRDY interrupts when you want to start transmitting again, write the first data byte into the holding register. Then set TxEN true in the command register. As soon as the clear-to-send signal is true, the transmitter will become operative and the data will be transferred to the shift register. This will cause TxRDY to become true and to generate a B interrupt. Data transmission can then continue.

#### RECEIVING DATA

Data is placed in the receive holding register when the receiver is operative and data is present on the input lines. The receiver is operative when the RxEN bit is set in the command register and the carrier-detect signal is being received low from the external device. The receiver detects data on the input line by searching for a valid start bit. As bits are received, they are assembled one bit at a time by the receiver. When the entire 8-bit character is assembled, the data is placed in the receive holding register starting with the low order bit. If the received character is less than 8 bits, the high order bits are set to zero. When the receive holding register has data in it, the RxRDY signal is asserted true (low), generating a B interrupt. Bit 1 in the status register also indicates if the receive holding register has data in it. When the receive holding register is read, RxRDY goes false, bit 1 in the status register is cleared, and the receiver places the next assembled data byte in the received holding register.

Therefore, to receive data, set RxEN true by setting bit 2 in the command register. When carrier detect is received true, the receiver begins monitoring the data lines. Assembled data is placed in the receive holding register and RxRDY becomes true. A B interrupt is generated. Read the receive holding register, which sets RxRDY false and clears the interrupt register.

The receiver becomes disabled when either the carrier detect input becomes high, or the RxEN bit is cleared. If the receiver becomes disabled in the middle of assembled a data byte, that data byte is discarded and no information is placed in the receive holding register.

#### DATA PARAMETERS

Parameters specifying the type of data you are sending or receiving are specified using the mode registers. The RTI always operates in asynchronous mode. The character length specification is for the number of bits actually used in the character. Parity bits, start bits, and stop bits are not included in this specification. If enabled by setting bit 4 in the mode 1 register, the transmitter automatically adds a parity bit to the data it has just sent. When receiving data, if parity is enabled, the receiver checks the parity bit against the data and reports an error in the status register if the parity bit does not match the parity of the data. Bit 5 in the mode 1 register specifies odd or even parity and has no meaning if parity is not enabled. Stop bits are selectable as 1, 1.5, or 2 stop bits. On transmit however, if 1.5 stop bits is selected, one stop bit will be sent. 1.5 stop bits is set only when received data has 1.5 stop bits.

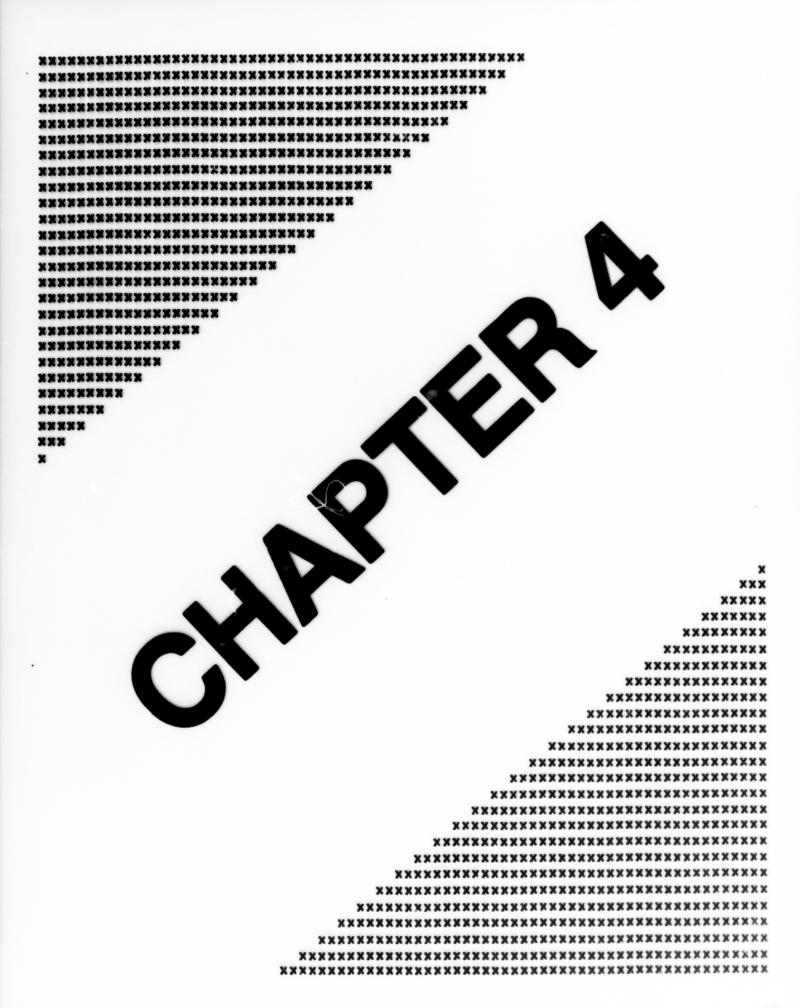
Mode 2 register is used solely for band rate selection and applies both to data transmitted and data received.

#### **MODEM CONTROL LINES**

The SLUs receive carrier detect, clear-to-send, and data set ready modem control signals. A high data set ready indicates that the external device is ready to send data; it can be used as a general purpose handshaking signal. A high data set ready is often acknowledged by a high data terminal ready signal, indicating that the SLU is ready for data transfer. A low on carrier detect indicates that the external device is sending data. Carrier detect must be low for the SLU receiver to be operative. (The receiver must also have been enabled—see above.) Clear-to-send is sent by an external receiving device in response to the SLU's asserting request-to-send. Clear-to-send must be low for the SLU transmitter to be operative. (The transmitter must also have been enabled—see above.)

The SLU's send data terminal ready and request-to-send modem control signals. Both signals are asserted or go low depending on the condition of their corresponding bit in the command register. Thus, writing a one to bit 1 of the command register causes the SLU to assert data terminal ready low. Data terminal ready is used to respond to an external device's data set ready signal. Generally when the external device receives data terminal ready low, it asserts carrier detect and begins sending data. Request-to-send is used when the SLU is transmitting data. Request-to-send is generally asserted low to interrogate the remote device's availability. If the remote device is capable of receiving data, it pulls clear-to-send low. This activates the SLU's transmitter and data transmission begins. Request-to-send is controlled by setting or clearing bit 5 in the command register. If bit 5 is cleared, but there is still data in the transmit shift register, request-to-send will go high one clock time after the last bit is transmitted.

# PAGE 32 INTENTIONALLY LEFT BLANK



IEEE-488 Bus Logic

# Chapter 4

# IEEE-488 Bus Logic

#### THEORY OF OPERATION

Figure 4-1, IEEE Logic Simplified Block Diagram, summarizes the operation of the IEEE logic.

The IEEE-488 logic is implemented using a Texas Instruments TMS 9914A General Purpose Interface Bus (GPIB) Adapter chip. There are also a data buffer, a signal buffer, and some select logic. The data section of the TMS 9914A transfers data between the RTI DBUS and an internal data bus. The data buffer transfers data between the internal data bus and remote devices over the IEEE bus. The IEEE bus data lines are labled DIO1 through DIO8.

The TMS 9914A chip controls the data and signal buffers by generating a Talk Enable (TE A or TE B) signal in its talk enable [T]EN logic. The interrupt logic generates either IEEE INT or IEEE MAINT INT when service is requested from the CPU. The Write Enable (WREN) logic is activated by the WLB signal from the CPU. This function writes data or control signals out to remote devices. The SDEN signal from the CPU activates the Data Input (DIN) logic. which reads data in from remote devices and transfers it to the CPU. The chip enable EN logic is activated by either the SEL 2 or SEL 3 (SEL 6 or SEL 7 for MAINT) signals from the bus interface. The INIT signal from the CPU activates the RESET logic. The clock (CLK 1) signal (2.5344 MHz) from the RTI provides the timing clock for the TMS 9914A chip. The controller (CTLR) logic activates the Direct Control (DIR CTL) logic of the signal buffer. The internal signal bus transfers control signals between the TMS 9914A and the signal buffer. The internal registers of the TMS 9914A chip are addressed by the register select (RS) lines 0 through 2, which are activated by the ADD 1 through 3 signals from the bus interface.

The select logic shown at the bottom of the block diagram (Figure 4-1) enables the system control (SYS CTL) logic of the signal buffer. Note that the only difference between the two sets of logic is the use of either the SEL 2 or SEL 6 signals, shown at the bottom left of each section of the diagram.

The signal buffer transfers control and handshaking signals between the RTI and the remote devices.

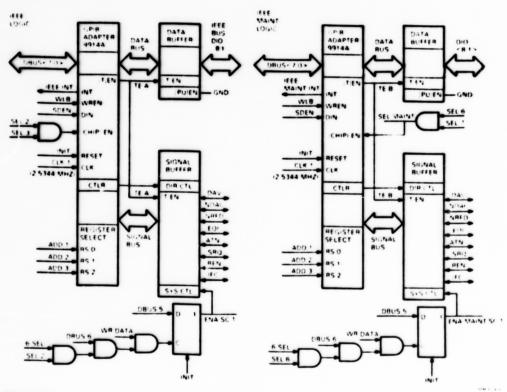


Figure 4-1 IEEE Logic Simplified Block Diagram

#### **BUS OVERVIEW**

It is beyond the scope of this document to describe fully the use of the IEEE-488 bus specification. The following description should not be considered comprehensive. Full details on the IEEE-488 bus specification can be found in the IEEE-488 publication *IEEE Standard Digital Interface for Programmable Instrumentation*, number IEEE Std. 488-1978. A supplement to the standard was also published by the IEEE in 1980.

The IEEE-488 bus is a bit-parallel, byte serial bus which implements a three-wire handshaking scheme. The handshaking is designed so that data transfer occurs at a rate set by the slowest instrument on the bus. This ensures that all data on the bus is available to all instruments. The bus consists of 24 lines connecting instruments. Figure 4-2 shows the IEEE-488 bus lines. Three of the lines are used for handshaking, five are used for bus control, eight are used for data transfer, and eight are used as grounds. Instruments on the bus are capable of being in one of three states, called controller-in-charge, talker, and listener. A device which is by default the controller-in-charge is called the system controller. The RTI is capable of being the system controller, the controller-in-charge, a talker, or a listener.

The control and handshaking lines used in the IEEE bus are described below.

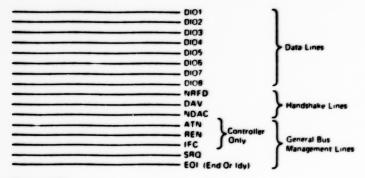


Figure 4-2 IEEF-488 Bus Lines

#### **Control Lines**

The five control lines are ATN, REN, IFC, SRQ, and EOI.

#### ATN

The ATN (attention) line can be asserted only by the controller-in-charge. The state of the ATN line signifies the meaning of the data lines DI01 through DI08. When the ATN line is true (low), the data lines are carrying interface commands. (See Figure 4-10.) When the ATN line is false (high), the data lines are carrying data.

#### REN

The REN (Remote Enable) control line is also sent only by the system controller. The REN line controls whether a remote device can be controlled by its front panel or by the IEEE bus. When the REN line is true (low), front panel controls on a remote device that is addressed as a listener are inoperative.

NOTE: If a device has a Return to Local button or switch on its front panel, this function is not disabled by the REN line. The return to local function on a device can be disabled using the Local Lockout interface command. (See Data-Out Register.) Also note that when the REN line is false, devices do not actually enter the remote state until they are addressed by the controller-in-charge as a listener.

When the REN line is false (high), the remote device may be controlled by its own front panel controls. Note that not all devices will be affected by the REN line.

# IFC

The IFC (Interface Clear) line can be asserted only by the system controller. When the IFC is asserted true (low), the bus is placed in a known idle state equivalent to the power up condition.

#### SRQ

The SRQ (Service Request) line may be asserted by any device on the bus. When a device asserts this line true (low), it is signaling the controller-in-charge that it needs service. Generally, the controller-in-charge must serially poll all the devices on the bus in order to determine which device asserted SRQ.

#### EOI

The EOI (End or Identify) line can be sent by either a talker or a controller-incharge. It has two meanings, depending on the current state of the ATN line. When the ATN line is false (high), the EOI line is set true (low) to indicate the end of a data transmission. When the ATN line is true, the EOI line is asserted true by the controller-in-charge, initiating execution of a parallel poll of the remote devices.

# Handshaking Lines

The three handshaking lines are DAV, NDAC, and NRFD.

#### DAV

The DAV (Data Valid) line is asserted low by the device sending data to indicate that it has just placed valid data on the data lines.

### NDAC

The NDAC (Not Data Accepted) line is used by devices receiving data to indicate that they have latched the data on the data lines into their receiving buffers. This line is called Not Data Accepted because it uses negative logic to indicate acceptance of the data. NDAC is kept in the true state until the data has been latched. The transition into the false (high) state indicates data acceptance.

#### NRFD

The NRFD (Not Ready for Data) line is used by devices receiving data to indicate that they are now ready to receive the next data byte from the sending device. This line is called Not Ready for Data because it uses negative logic to indicate readiness. NRFD is kept in the true state as long as a device is unable to receive data. When the device is ready to receive data, the NRFD line is placed in the false (high) state.

The handshaking lines operate in the following manner for the transfer of data.

- 1. All active listeners indicate their readiness to receive data by placing the NRFD line in the false state. Since the low state indicates that a device is not ready for data, a single device holding the NRFD line low will keep the line low. In this way the NRFD line is only in the false (high) state when all devices have released the line. This ensures that data transmission occurs at a speed set by the slowest device on the bus and that all devices are able to receive data concurrently.
- When the device sending data (the active talker) asserts the data lines, it also asserts the DAV line in the true state (low). This indicates that valid data is now on the bus.

- 3. When the listeners detect DAV in the true state, they set NRFD true. Then each listener latches the data lines to read the data byte and releases the NDAC line. When all listeners have released the NDAC line, the NDAC returns to the high state.
- When the talker detects that NDAC is in the high state, it releases the DAV line.
- 5. When the listeners detect that DAV is in the high state, and if they have finished processing the latched data byte, they assert the NDAC in the low state and allow NRFD to go to the high state. The handshaking sequence begins again.

# Data Lines DIO1 through DIO8

The data lines are labeled DIO1 through DIO8.

The least significant bit of the data lines is carried on DIO1. The most significant bit is carried on DIO8. The data lines use negative TTL (Transistor-Transistor Logic). The true state indicates a 1 and is represented by a voltage between 0 and 0.5 volts. The false state indicates a 0 and is represented by a voltage greater than 2.0 volts and less than 5.0 volts.

#### **Device States**

Any device on the bus may be in one of four states, talker, listener, controller-in-charge, and system controller. Devices in the talker state are set up to transmit data across the DIO lines. Devices in the listener state are set up to receive data from the DIO lines. The controller-in-charge is the device which determines what remote devices are talkers or listeners and can control data flow. The system controller is the device which is the controller-in-charge when power is first applied to the bus, or when the bus is cleared by the IFC line.

The system controller is by default the controller-in-charge when the bus first becomes active. However, the system controller can pass the controller-in-charge function to other devices. In this case, the system controller can become a talker, a listener, both, or neither. The system controller can regain the controller-in-charge state at any time by asserting the IFC line. The system controller is the only device permitted to control the IFC and REN lines.

The controller-in-charge device specifies which devices are talkers or listeners and can, in fact, address itself as talker or listener. The controller-in-charge is the only device permitted to control the ATN line.

All devices, regardless of whether they are configured for talker or listener, may be in an active or passive state. A device in a passive state is not affected by the control or handshaking lines and does not place data on the data lines.

The RTI may be configured to be the system controller, the controller-incharge, a talker, or a listener.

# TMS 9914A FEATURES

To facilitate programming for the IEEE-488 bus, the TMS 9914A chip implements some useful features. These include the use of auxiliary commands and DAC and RFD holdoffs.

# **Auxiliary Commands**

Much of the complexity of the bus has been simplified in the TMS 9914A chip through the use of auxiliary commands. These commands allow commonly used functions to be performed by writing a command byte to the auxiliary command register. Some auxiliary commands are state dependent; that is, they may be used only when the RTI is controller in charge, etc. The auxiliary command format includes a Clear/Set (C/S) bit which allows certain features selected by an auxiliary command to be turned on or off. For example, the holdoff on all data (hdfa) auxiliary command, when written with the C/S bit set to 1, places the RTI in a holdoff on all data mode. (See Ready For Data Holdoff, below.) This feature then stays in effect until the hdfa auxiliary command is written with the C/S bit set to 0. Other auxiliary commands are immediate execution type and the C/S bit has no meaning for them. The feature selected by the command occurs immediately and the program continues.

The auxiliary commands are described in the section Descriptions of Auxiliary Commands.

# Ready for Data Holdoff

The RFD holdoff is an interruption of the normal sequence of handshaking. This selectable feature allows your program to examine the currently received byte and to alter program flow if desired. When an RFD holdoff occurs, the acceptor handshake sequence is not completed. Even though the RTI has received the data byte, if the holdoff on all data feature is selected, the NRFD line is held low. This puts the data or command sender on hold. The NRFD line is not released until the program specifically writes the auxiliary command rhdf (release RFD holdoff). Writing this command causes the NRFD line to go high, indicating that the RTI is ready to receive the next data byte.

The RFD holdoff feature is selected by writing the holdoff on all data (hdfa) auxiliary command with the C/S bit set to 1. This causes an RFD holdoff to occur every time a data byte is latched into the data in register. Writing the hdfa holdoff auxiliary command with the C/S bit set to 0 clears the RFD holdoff feature. (See Descriptions of Auxiliary Commands.)

The RFD holdoff can be simulated by enabling the BI (byte in) interrupt. When this interrupt is selected, an A interrupt is generated as the data-in register latches the data lines. The NRFD line is held low until the data-in register is read.

The BI interrupt and hdfa auxiliary command operate independently. If you write the hdfa command with the C/S bit set to 0 (to clear RFD holdoffs), but leave the BI interrupt unmasked, RFD holdoffs still occur every time a data byte is latched into the data in register. In this instance, however, using the rhdf auxiliary command is not necessary since reading the data-in register releases the NRFD line.

# **Data Accepted Holdoff**

The data accepted holdoff (DAC holdoff) is similar to the RFD holdoff, but affects the NDAC line instead. The DAC holdoff is an interruption of the normal sequence of handshaking. This selectable feature allows your program to examine the currently received byte and to alter program flow if desired. When a DAC holdoff occurs, the acceptor handshake sequence is not completed. Even though the RTI has received the data byte, if the DAC holdoff feature is selected, the NDAC line is held low. This puts the data or command sender on hold. The program resumes data transmission by writing the auxiliary command dacr (release DAC holdoff). Writing this command causes the NDAC line to go high, indicating that the RTI has received the data byte.

The DAC holdoff is selected by enabling the interrupts which cause DAC holdoffs. The following interrupts cause DAC holdoffs when the interrupt condition occurs: GET (group execute trigger); UNC (unrecognized command); APT (address pass through); DCAS (device clear state); and MA (my address). See Interrupt Status Registers and Interrupt Mask Registers. There is no auxiliary command for enabling DAC holdoffs.

#### PROGRAMMING INFORMATION

The IEEE registers have addresses ranging from 17775220 to 17775236; the IEEE MAINT register addresses range from 17775260 to 17775276. The maintenance section is used in conjunction with the IEEE port to test the functional-

ity of the bus and IEEE logic. During power-up self-test, the maintenance section becomes active at IEEE bus address 30, while the normal IEEE port is assigned address 0. After the self-test completes, the maintenance section is logically disconnected from the IEEE bus.

Tables 4-1 and 4-2 below list the TMS 9914A Read and Write Registers. Note that the logic reads the addresses by looking at the address bits 1, 2, and 3, corresponding to the ADD 1, 2, and 3 signals.

Table 4-1 IEEE-488 Logic Read Registers

Address	Bits 4 3 2 1 0	Register Name
177752(20)	10000	Interrupt Status 0
177752(22)	10010	Interrupt Status 1
177752(24)	10100	Address Status
177752(26)	10110	Bus Status
177752(34)	11100	Command Pass Through
177752(36)	11110	Data In

Table 4-2 IEEE-488 Logic Write Registers

	Bits		
Address	43210	Register Name	
177752(20)	10000	Interrupt Mask 0	
177752(22)	10010	Interrupt Mask 1	
177752(26)	10110	Auxiliary Command	
177752(30)	1 1 0 0 0	Address	
177752(32)	1 1 0 1 0	Serial Poll	
177752(34)	11100	Parallel Poli	
177752(36)	1:110	Data-Out	

The use of each register in the chip is described below.

# **Address Register**

The address register (Figure 4-3) is used to specify the IEEE bus address of the RTI. The address is selected with bits 4 through 0, but 31 (decimal) is illegal. Thus, any combination of bits 4 through 0 is possible, except the combination of bits 4 through 0 being all 1s (decimal 31).

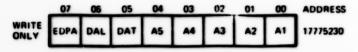


Figure 4-3 Address Register

Table 4-3 Address Register

8it	Name	Set	Significance
7	EDPA	1	Places RTI in dual primary addressing mode
		0	Allows only one address for RTI
6	DAL	1	Disables RTI as a listener
		0	Enables RTI as a listener
5	DAT	1	Disables RTI as a talker
		0	Enables RTI as talker
4-0	A5-A1	٠	User selected address for RTI. Can be any number between 0 and 30 decimal (00000 and 11110 binary)

Bits 5 and 6, if set, disable the RTI as a talker or listener, respectively. Writing this register with bits 5 or 6 clear enables the RTI as a talker only or as a listener only, respectively. Enabling the RTI as a listener or talker however does not place the RTI into the listener or talker states (LADS or TADS). This must still be done explicitly by the controller-in-charge. However, if the controller-in-charge addresses the RTI as a listener (or talker), and the RTI is disabled as a listener (or talker), it will be as if the controller-in-charge addressed a nonexistent device. The disable talker or listener function allows you to use the same address on the bus for two devices. For example, if you assign the RTI address 1 but disable the talker function, address 1 on the bus can be used for a different device that is a talker. When the controller-in-charge places a my talk address of 1 on the bus, the RTI does not recognize that as its address since the talker function is disabled.

The dual primary addressing mode allows two consecutive IEEE bus addresses to be assigned to the RTI. If bit 7 is set, the least significant bit of any address placed on the bus is ignored by the RTI. Thus, when attempting to detect if it is being addressed, the RTI recognizes as its address any match between bits 4 through 1 and the bus address. This allows the RTI to be addressed through two addresses; one with bit 9 equal to 1, and one with bit 0 equal to 0. The least significant bit of any address placed on the bus is not lost and may be determined by reading the ulpa bit of the address status register. Dual addressing is useful if you want to have separate addresses for the RTI in two different states. For example, you may want one address for the RTI as a talker and a separate address for the RTI as a listener.

# **Address Status Register**

The address status register is shown in Figure 4-4.

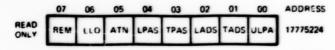


Figure 4-4 Address Status Register

Table 4-4 Address Status Register Data

Bit	Name	Set	Significance
7	REM	1	The RTI is in the remote state.
		0	The RTI is in the local state
6	LLO	1	The RTI is in local lockout
		0	The RTI is not in local lockout
5	ATN	1	The ATN line is currently true (low)
		0	The ATN line is currently false (high)
4	LPAS	1	The RTI is in the listener primary addressed state (LPAS) as defined by the IEEE-488 standard. The RTI has received its primary listen address and the LE function is ready to receive its secondary address.
		0	The RTI is not in LPAS

Table 4-4 Address Status Register Data (Cont.)

9it	Name	Set	Significance
3	TPAS	•	The RTI is in the talker primary addressed state (TPAS) as defined by the IEEE-488 standard. The RTI has received its primary talk address and the TE function is ready to receive its secondary address.
		0	The RTI is not in TPAS
2	LADS	'	The RTI is in the listener addressed state (LADS) as defined by the IEEE-488 standard. The RTI has recognized its listen address and is prepared to perform, but is not currently involved in, data transfer. Note: This bit remains set when the RTI passes into the listener active state (LACS).
		0	The RTI has not yet been addressed to listen
1	TADS	•	The RTI is in the talker addressed state (TADS) as defined by the IEEE 488 standard. The RTI has recognized its talk address and is prepared to perform, but is not currently involved in, data transfer. Note: This bit remains set when the RTI passes into the talker active state (TACS).
		0	The RTI has not yet been addressed to talk
0	ULPA	•	This bit shows the least significant bit of the most recent address recognized by the 9914. This bit is useful when using dual primary addressing. (See Address Register.)

NOTE: The address status register obtains its information from the TMS 9911A chip at the time it is read.

# **Auxiliary Command Register**

The auxiliary command register (Figure 4-5) allows you to perform commonly used functions of the IEEE bus by writing a single command. The commands themselves are described individually below.

Bits 0 through 4 are used to specify the desired command. Bits 5 and 6 have meaning only when they are part of the set software reset command.

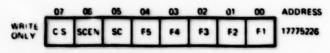


Figure 4-5 Auxiliary Command Register

Table 4-5 Auxiliary Command Register Data

Oit	Signal	Significance
7	C/S	Clear or set operation (where applicable)
6	SCEN .	System Controller ENable (1 = Enable)
5	sc ·	System Controller (1 = System Controller, 0 = Slave)
4-0	F4-F0	Auxiliary command select (See Auxiliary Commands)

WR 13004

Some of the auxiliary commands make use of bit 7, the clear/set bit. These commands, when written with bit 7 set, enable a feature which stays in effect until the command is written with bit 7 set to 0. These commands are listed below. The C/S bit is shown as a c.

Disable all interrupts	dai	c0010011
Holdoff on all data	hdfa	c0000011
Holdoff on EOI only	hdfe	c0000100
Listen only	lon	c0001001
Release DAC holdoff	dacr	c0000001
Request parallel poll	rpp	c0001110
Request service bit 2	rsv2	c0011000
Return to local	rtl	c0000111
Send interface clear	sic	c0001111
Send remote enable	sre	c0010000
Shadow handshake	shdw	c0010110
Short T1 settling time	std1	c0010101
Software reset	swrst	c0000000
Talk only	ton	c0001010
Very short settling time	vstd1	c0010111

<sup>·</sup> Applicable only as part of the software reset command.

The other auxiliary commands are immediate execution commands. When they are written to the auxiliary command register, the selected function occurs and the program continues. For these type of commands, bit 7, shown as an x below, has no meaning and may be set or clear. The immediate execution commands are:

Go to standby	gts	x0001011
New byte available false	nbaf	x0000101
Pass through next secondary	pts	x0010100
Release control	rle	x0010010
Release RFD holdoff	rhdf	x0000010
Request control	rge	x0010001
Set EOI true with next byte	feoi	x0001000
Take control asynchronously	tca	x0001100
Take control synchronously	tcs	x0001101

# **Descriptions of Auxiliary Commands**

On the following pages, the auxiliary commands are described. The command format is shown with the least significant bit of the command byte at the right-hand side. Some bits may be either one or zero in the command byte; these are shown as "x" and may be thought of as "don't care" bits. When the auxiliary command is the clear/set type, both the clear and set commands are shown. When the command is an immediate execution type, the C/S bit (bit 7) is shown as an "x."

Name:	Disable all interrupts
Mairie.	Disable an interrupt

Mnemonic: dai
Clear command: 00010011
Set command: 10010011

The dai command allows you to suspend indefinitely the generation of any A interrupts. The set dai command disables the ability of any unmasked interrupts in the interrupt mask register to generate an A interrupt. It does not change the contents of the interrupt mask register, however. Since the interrupts are disabled, DAC holdoffs do not occur. RFD holdoffs, if enabled by the hdfa or hdfe commands, are not affected. The clear dai command reenables the interrupt mask register's ability to generate A interrupts. The dai command is therefore used to temporarily disable the effect of the interrupt mask registers.

Name

Go to standby

Mnemonic

ats

Command

×0001011

The gts command can be used only when the RTI is the controller-in-charge. When this command is written, the ATN line goes false.

Name

Holdoff on all data

Mnemonic
Clear command
Set command

hdfa 00000011

10000011

The holdoff on all data command controls whether RFD holdoffs are in effect. The set hdfa command places the RTI in RFD holdoff mode, and an RFD holdoff occurs every time a new data byte is latched in the data-in register. (See RFD Holdoff above.) The RFD holdoff is released only when the program writes the rhdf command. The clear hdfa command removes the RTI from RFD holdoff mode, and the NRFD handshaking line is controlled automatically by the RTI as each data byte is received.

Name

Holdoff on EOI

Mnemonic
Clear command

hdfe

Clear command

10000100

The set hdfe command places the RTI in the holdoff on EOI mode. In this mode, the RTI enters an RFD holdoff when it detects the end of a data string message. This message is the EOI line true while the ATN line is false. (See Ready for Data Holdoff above.) The RFD holdoff must be released by writing the rhdf command. The clear hdfe command removes the RTI from this mode.

# CHAPTER 4 | IEEE-488 BUS LOGIC

Name

Listen only

Mnemonic.

ion

Clear command: Set command: 00001001 10001001

The lon command is used when the RTI is the controller-in-charge and wishes to set itself up to listen. If the bus has no current controller-in-charge, lon could also be used to configure the RTI. The set lon command places the RTI into the listener addressed state (LADS). The clear lon command removes the RTI from the listener state (LADS, LACS, etc.). Note that a bus interface command could also remove the RTI from the listener state.

Name:

New byte available false

Mnemonic:

nbat

Command

×0000101

This command is used only in the following circumstance. If the RTI is a talker and has placed a byte in the data-out register, but an interface command or other interruption prevents the byte from being sent, that byte is stored. (See Data-Out Register below.) When the ATN line again goes false, the byte is normally placed on the data lines. This command is used to suppress the transmission of the buffered byte and should be written before the ATN line returns to false.

Name:

Pass through next secondary

Mnemonic

pts

Command

x0010100

The pts command allows a remote device to configure the RTI for a parallel poll. It is used in conjunction with the UNC (unrecognized command) interrupt. If the UNC interrupt is enabled, the ppc (parallel poll configure) interface command causes an interrupt. Your program then examines the contents of the command pass through register and determines that the command is a ppc command. When you write the pts auxiliary command, the next received byte also causes a UNC interrupt and your program can obtain that byte from the command pass through register. This byte should be the ppe (parallel poll enable) interface command from the remote device. Your program then configures and enables for the parallel poll.

Name:

Release control

Mnemonic

ric

Command:

x0010010

The rlc command is used to pass control from the RTI to another device on the bus. The command may be written only when the RTI is currently the control-ler-in-charge. The command is used after the RTI has sent the TCT (take control) interface message to another device. The rlc command completes the handshaking sequence by releasing the ATN (letting it go high). The other device then becomes the controller-in-charge.

Name:

Release DAC holdoff

Mnemonic:

dacr

Clear command: Set command: 00000001 10000001

If a DAC holdoff has occurred, the clear dacr command releases the DAC holdoff. (See DAC Holdoff above.) The RTI can then receive the next byte of data from the transmitter. The handshaking sequence is handled automatically once the release DAC holdoff command is written.

The set darr has meaning only if the data byte just received was a secondary address. In this event, the C/S bit is used to indicate that the data byte was a valid secondary address. If the data byte was not a valid secondary address, the clear darr command should be written.

Name

Release RFD holdoff

Mnemonic

rhdf

Command

×0000010

The release RFD holdoff command releases a ready for data holdoff (see RFD Holdoff above). The C/S bit is not applicable and can be either 0 or 1.

Name

Request control

Mnemonic

rac

Command

×0010001

The rqc command is used when the RTI is not currently controller-in-charge and has received the TCT (take control) message from the controller-in-charge. The TCT message is recognized by the unidentified command pass through feature. When the RTI receives the TCT command, the rqs command should be written to the auxiliary command register. When the RTI detects that the

ATN line is false (this is done automatically; your program does not have to monitor ATN), the RTI enters the controller active state (CACS) and now has control of the bus as controller-in-charge.

Name Request parallel poll

Mnemonic rpp Clear command 00001110 Set command 10001110

This command may be used only when the RTI is the controller-in-charge, and the ATN line is asserted. The rpp command places the parallel poll interface command on the data lines. The remote devices respond by placing their bit of status information on the data line assigned to that device. The byte representing the devices' responses can then be read in the command pass through register. A minimum delay of 2 microseconds should be allowed after writing the rpp command before reading the command pass through register. The clear rpp command releases the ATN line, completing the parallel poll.

Name: Request service bit 2

Mnemonic: rsv2 Clear command: 00011000 Set command: 10011000

The rsv2 command provides a second means of requesting service other than the rsv1 bit in the serial poll register. When the set rsv2 command is written, the SRQ line is asserted true, notifying the controller-in-charge that the RTI is requesting service. When the controller-in-charge polls the RTI for the contents of the serial poll register, the SRQ line is released to the false state. The clear rsv2 command also releases the SRQ line in the elent that the controller-in-charge does not respond to the service request and you want your program to release SRQ.

The rsv1 bit in the serial poll register and the set rsv2 command both control the SRQ line. However, you should use only one or the other to request service from the controller. The set rsv2 command sets the rsv1 bit the serial poll status message and, when polled by the controller-in-charge, asserts the DIO7 line true, thus identifying the RTI as the device which requested service. However, when used in this way, the rsv1 bit is cleared when the controller-in-charge reads the serial poll register. Thus, to request service using the rsv2 command, you should have previously written a status word to the serial poll

register (with rsv1 clear). This could perhaps be done during configuration of the RTI (the set swrst state). Then, to request service, write the set rsv2 command to the auxiliary command register. The SRQ line goes true; the controller-in-charge reads the serial poll register, identifies the RTI as requestor since the DI07 line is true, and determines the RTI's status from the other seven bits. After the serial poll register is read, the SRQ line is set false. It is not necessary to write the clear rsv2 auxiliary command if the controller-in-charge reads the serial poll register. When you wish to request service again, write the set rsv2 command again. See Serial Poll Register.

Name: Return to local

Mnemonic: rtl

Clear command 00000111
Set command 10000111

Both the set rtl command and the clear rtl command place the RTI in the local state, assuming that local lockout is not in effect. If the RLC interrupt is enabled in interrupt mask register 0 (bit 1 is set to 1), then an A interrupt is generated when this command is written. In the local state, the RTI does not respond to any data or commands on the bus. When the set rtl command is used to place the RTI in the local state, the clear rtl command must first be written before the RTI will return to remote. The REN line will not effect the RTI until the clear rtl command is written. If the RTI is placed in local by the clear rtl command, the REN line can place the RTI back in the remote state at any time.

Name Send interface clear message

Mnemonic sic

Clear command: 00001111
Set command: 10001111

This auxiliary command, which may be used only when the RTI is the system controller, controls the IFC control line. The set sic command asserts the IFC line true. This message is an IEEE-488 defined message to clear the bus interface. All remote devices stop any data transmission and return to idle states. The IEEE-488 standard sets a minimum time of 100 microseconds for IFC true to clear the interface. Your program, therefore, should have a wait cycle of at least 100 microseconds for the interface to be cleared. The clear command is then written to place the IFC back in the false state.

# CHAPTER 4 | IEEE-488 BUS LOGIC

Name:

Send remote enable

Mnemonic:

sre

Clear command:

00010000

Set command:

10010000

The sre command, which may be used only when the RTI is the system controller, controls the REN (Remote Enable) control line. The set sre command asserts the REN line true, placing devices on the bus in the remote state. The line stays true until the clear sre command is written.

Name:

Set EOI true with next byte

Mnemonic:

feoi

Command:

x0001000

The set EOI true command, or force EOI, is used to indicate the end of data string message. When this command is written, the EOI goes true as soon as the next data byte is transmitted by the RTI. The EOI returns to false when the acceptor handshaking sequence completes.

Name:

Shadow handshake

Mnemonic:

shdw

Clear command:

00010110

Set command:

10010110

The shdw command is used only when the RTI is the controller-in-charge. It allows the RTI to participate in the listener acceptor handshaking sequence without actually receiving data. When the set shdw command is written, the DAC line is asserted true for 1.2 microseconds (3 clock cycles) as soon as the DAV line is true. As soon as DAV goes false, the NRFD line is allowed to go false. This simulates the handshaking sequence that occurs when data is received. After the set shdw command is written, the RTI performs the shadow handshaking sequence every time the DAV line goes true. The clear shdw command removes the RTI from the shadow handshaking mode of operation. The shadow handshake is particularly useful in conjunction with the tes (take control synchronously) auxiliary command. If your program issues the shadow handshake auxiliary command before issuing the tcs command, the RTI will be synchronized with the other devices. ATN will not be asserted as a result of the tes command until the end of any data transfer currently occurring. Therefore, any data transfer occurring when the tcs command is written will not be lost. If you want to allow multiple byte transfers to conclude before the RTI takes control synchronously, the END interrupt should be enabled. Your interrupt

service routine can then issue the tcs command and clear the shadow handshake, if desired. This strategy allows all data transfer to complete before the RTI takes control synchronously.

Name Short T1 settling time

Mnemonic std1
Twice T1 command 00010101
T1 command 10010101

The std1 command allows you to alter the bus settling time, T1. If the set std1 command (the C/S bit set to 1) is written, the bus settling time is set to 2.4 microseconds (6 clock cycles). If the clear std1 command is written, the bus settling time is set to 4.4 microseconds (11 clock cycles). The default T1 settling time is 4.4 microseconds.

Name: Software reset

Mnemonic: swrst
Clear command: 00000000
Set command: 1\*\*00000

The software reset command is used to bring the RTI to a known idle state on the bus. When the set swrst command is written, the RTI enters the following IEEE-488 defined states:

SIDS	Source Idle State	CIDS	Controller Idle State
AIDS	Acceptor Idle State	LOCS	Local State
TIDS	Talker Idle State	<b>NPRS</b>	Negative Poll Response State
TPIS	Talker Primary Idle State	PPSS	Parallel Poll Standby State
LIDS	Listener Idle State	SPIS	Serial Poll Idle State
LPAS	Listener Primary State		

The software reset state is equivalent to the power-on state. While in this known idle state, the RTI should be configured. This includes writing the address register to define the RTI's address, writing the interrupt mask registers to specify which interrupts are enabled, and writing any auxiliary commands such as holdoff on all data (hdfa) that may be desired. In addition, if the RTI is to be the system controller, bits 5 and 6 should be set as described below.

<sup>\*\*</sup> These bits are used to set the RTI's status as system controller. See below.

After the RTI is configured, the clear software reset auxiliary command should be written. At that point, the RTI is a defined entity on the bus and may participate in bus interactions.

Bits 5 and 6 of the set swrst command are used to define whether or not the RTI is to be the system controller. Although the controller-in-charge role can be passed between devices capable of control, there can be only one system controller on the bus. The system controller is the only device that can assert the IFC line. Whenever the IFC line becomes true, control always returns to the system controller.

The SCEN and SC bits together define the RTI's role as system controller. If the RTI is to be configured as system controller, both the SCEN and SC bits must be set when the set swrst command is written. (This is 11100000.)

If the SCEN bit is clear (0), the SC bit is ignored and may be either 1 or 0. If this is the first software reset command, the RTI is not configured as the system controller. If the RTI was previously defined as the system controller, and the software reset command is written with the SCEN bit clear, the RTI will remain the system controller.

The RTI can give up its status as system controller. However, this is not recommended as the IEEE standard defines the bus as having only one system controller. The RTI, however, can be programmed to no longer be the system controller if it has already been defined as the system controller. All activity on the bus must cease and all devices must be placed in their power-up states for this to happen. If the RTI was previously the system controller, and the set swrst command is being used to specify that there is no system controller, or to change the role of system controller to another device (not recommended), the SCEN bit is set to 1 and the SC bit is set to 0. This combination clears the RTI's configuration as the system controller.

Note that bits 5 and 6 of the auxiliary command byte should be used only for system controller definition as part of the set swrst command. They should not be used as part of the clear swrst command and should be zeroes. With all other auxiliary commands, bits 5 and 6 must both be zeroes.

Table 4-6 SCEN and SC bits

SCEN Bit 6	SC Oit 5	Result
1	1	RTI becomes system controller
0	0/1	No change
1	0	RTI is no longer system controller

NOTE: The SC bit is CLEARED during CPU bus initialization.

Name:	Take control asynchronously
Mnemonic	tca
Command:	×0001100

The tca command can be used only when the RTI is the controller-in-charge. When the set tca command is written, the ATN line is asserted true and any data transmission on the bus stops. If the bus is currently in the process of a data transfer, the data transfer is not allowed to complete, and data may be lost.

Name	Take control synchronously
Mnemonic	tcs
Command	x0001101

The tcs command can be used only when the RTI is the controller-in-charge. The tcs command allows the RTI to take control of the bus without corrupting or losing any data transfer currently in progress. When the tcs command is written, the shadow handshaking command (shdw) should also be used to allow the RTI to participate in the data handshaking sequence. See shdw below. After the tcs command is written, the RTI waits until the NRFD, NDAC, and DAV lines are all false, then asserts ATN.

#### CHAPTER 4 | IEEE-488 BUS LOGIC

Name	Talk only
Mnemonic	ton
Clear command:	00001010
Set command:	10001010

The ton command is used when the RTI is the controller-in-charge and wishes to set itself up to talk. If the bus has no current controller-in-charge, ton could also be used to configure the RTI. The set ton command places the RTI into the talker addressed state (TADS). The clear ton command removes the RTI from the talker state (TADS, TACS, etc.). Note that a bus interface command could also remove the RTI from the talker state.

Name: Very short T1 delay
---------------------------

Mnemonic:	vstd1
T1 command:	00010111
Short T1 command:	10010111

The vstd1 auxiliary command allows further control of the bus settling time, T1. The set vstd1 command sets T1 to 1.2 microseconds (3 clock cycles) regardless of any previous time delay set by the std1 command. When the clear std1 command is written, the T1 time delay is set back to whatever time delay was previously specified by the std1 delay. If the std1 command was not previously written, the clear vstd1 command returns T1 to the default delay of 4.4 microseconds (11 clock cycles).

# **Command Pass Through Register**

The bits in the command pass through register (Figure 4-6) always reflect the current state of the data lines in the bus. The primary purpose of this register is to read the data lines in the event of unrecognized interface commands. Unrecognized interface commands will cause a UNC interrupt if this interrupt is enabled in interrupt register 1. The register can also be used to read secondary addresses in the event of an APT interrupt, provided the APT interrupt has been enabled in interrupt register 1.

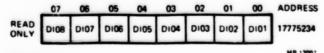


Figure 4-6 Command Pass Through Register

The command pass through register is also used to read the data lines after sending an rpp (request parallel poll) auxiliary command. A delay of at least 2 microseconds should occur between the set rpp auxiliary command and the reading of this register.

# **Data-In Register**

The data-in register (Figure 4-7) is used to receive data when the RTI is configured as a listener.

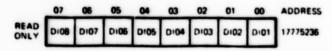


Figure 4-7 Data-In Register

When the talker asserts the DAV line low, the RTI (as listener) latches the data on the data lines into this register. If you have enabled the BI (byte in) interrupt, an A interrupt is generated. At this time the RTI also asserts the NRFD line low, indicating that it is not ready to receive any new data. Your interrupt service routine can then read this register and obtain the data byte. When the read operation is performed, the NRFD and NDAC lines are released (go high), indicating that the data has been accepted and the next byte can be sent.

The NRFD line must go false before the talker can transmit the next data byte. In default mode (no auxiliary commands written), the NRFD and NDAC lines go high immediately after the read operation and the talker begins the next handshaking sequence. However, if desired, your program can delay the release of the NRFD line through the use of the hdfa (holdoff on all data) auxiliary command. By writing this auxiliary command with the C/S bit set, the NRFD line is not immediately released upon a read operation. The line is kept in the low state until your program specifically writes the rhdf (release rfd holdoff) auxiliary command with the C/S bit set to 0. Called an RFD (Ready for data) holdoff, this delay in the release of the NRFD line is useful when you wish to process each data byte as received, or wish to alter program flow depending upon the value of each received data byte, etc.

In addition to the RFD holdoff described above, which causes a holdoff on the reception of every data byte, an RFD holdoff can be caused when the end-of-message string is received. The end-of-message string indicates that the data byte currently on the data lines is the last in a transmitted string. Devices recognize the end-of-data string message when the EOI line is true and ATN is false. The set hdfe (holdoff on end) auxiliary command specifies that an RFD holdoff occur when the end-of-string message occurs. The hdfe command would not generally be used along with the hdfa command.

Although the data-in register shares the same address as the data out register, reading the data-in register does not affect the data-out register in any way.

# **Data-Out Register**

The data-out register (Figure 4-8) is used when the RTI is configured to be a talker or controller-in-charge. As a talker, the register is used to place data on the eight data lines. As a controller-in-charge, the register is used to place either data or interface commands on the eight data lines.

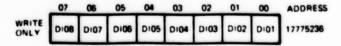


Figure 4-8 Data-Out Register

Use of the data-out register for sending data as a talker or controller-in-charge is similar. The BO (byte out) interrupt should be enabled prior to data transfer. The BO interrupt causes an A interrupt to occur any time a data byte in the register is placed on the data lines. The listener acknowledges reception by completing the handshaking sequence, indicating that the listener has received the data byte in the data-out register. Your routine which services the BO interrupt can now write the next data byte to the data-out register.

If the BO interrupt is enabled, a BO interrupt is caused when the RTI enters the talker state. This prompts the loading of the first data byte. If the controller-in-charge asserts the ATN line low while the RTI is talking, data transfer stops. If the RTI has written a data byte to the data-out register, but the byte has not yet been placed on the bus, the byte will remain in the register. When the ATN line goes high, the byte will be placed on the data lines. If you do not want to have the byte placed on the data lines in this instance, you should write the auxiliary command nbaf (new byte available false). This auxiliary command is an immediate execution command and should be written while the ATN line is still false. The nbaf auxiliary command will reset the transmission hardware and the byte in the data-out register will not be sent when the ATN line goes true. The register is not actually cleared, and a BO interrupt is not generated. The next byte written to the data-out register will overwrite the byte that was not sent.

When the BO interrupt is enabled and the RTI is configured as a talker, the data transfer handshaking sequence is carried out automatically by the RTI as soon as your program writes data to the data-out register.

When the RTI is configured as the controller-in-charge, the data-out register is used to transmit either data or interface commands. The state of the ATN line determines whether the information placed on the data lines is interpreted as data or as an interface command. If the ATN line is false (high), the information on the data lines is data. How the listener interprets data depends on the device and your program. If the ATN line is true (low), the information on the data lines is interpreted as interface commands, which are defined by the IEEE-488 standard. Devices and programs that comply with the IEEE-488 standard will react predictably to these commands. See below for more details on the interface commands.

Transmitting data when the RTI is configured as the controller-in-charge is similar to transmitting data when the RTI is configured as a talker. The RTI addresses itself as a talker and data transfer occurs as described above.

Transmitting interface commands when the RTI is configured as the controller-in-charge involves writing the commands to the data-out register while the RTI is in the controller active state (CACS). In this state, the ATN line is constantly held in the true (low) state. The format for interface commands is shown in Figure 4-9. The IEEE-488 interface commands themselves are shown in Figure 4-10. For the purposes of this document, an interface command is defined as the meaning attributed to the data lines whenever the ATN line is false. The interface commands are defined by the IEEE-488 standard, and it is beyond the scope of this document to fully describe the action of each interface command.

The handshaking sequence for interface commands is identical to that for data. The BO interrupt also functions identically for interface commands and for data.

	Type of							
Data Line # → 8	7	6	5	4	3	2	1	Command
0	0	0	0	•	•	•	•	Addressed
0	0	0	1	•	•	•		Universal
0	0	1	•	trum	•	•	•	MLA
0	1	0	•	trum	•	•	•	MTA
0	1	1	•	onda	•	Data	•	MSA
0	1	1	0	nditi	-	Line		PPE

\* = 0 or 1

MR 2134

Figure 4-9 Interface Command Format

	Data Line # 8 7 6 5 4 3 2 1	Command Abbreviation	Command Name
Addressed Commands	000000001 00000100 00000101	PPC	Go To Local Selected Device Clear Parallel Poli Configure Group Execute Trigger
Universal Commands	00010001 00010100 00010101 00011000 00011001		Local Lockout Device Clear Parallel Poll Unconfigure Serial Poll Enable Serial Poll Disable
Listener Address Commands	00100000	MLAO MLA1 ; MLA30 UNL	My Listen Address 0 My Listen Address 1 : My Listen Address 30 Unlisten
Talker Address Commands	01000000 01000001 : 01011110 01011111	MTAO MTA1 : MTA30 UNT	My Talk Address 0 My Talk Address 1  : My Talk Address 30 Untalk
Secondary Address Commands	01100000	MSA0 MSA1 : MSA30	My Secondary Address 0 My Secondary Address 1 My Secondary Address 30
Parallel Poll Enable Commands	01100000 01100001 i: 01100111 01101000 011010101 i: 01101111	PPE : PPE PPE : PPE PPE PPE PPE	Parallel Poll Enable Condition 0, Data Line 1 Condition 0, Data Line 2 : Condition 0, Data Line 8 Condition 1, Data Line 1 Condition 1, Data Line 2 : Condition 1, Data Line 8 Parallel Poll Disable

MR 2136

Figure 4-10 Interface Commands

# IEEE Bus Status Register

The IEEE status register (Figure 4-11) provides you with the ability to view the current status of the five control lines and the three handshaking lines in the bus at any time. The register is not latched and continually reflects the current state of the bus when read. In combination with the command pass through register, which continually reflects the current status of the eight data lines, a program can determine the state of the bus at any time.

NOTE: The IFC bit (bit 1) will not indicate a true (1) value if the RTI is system controller and is using the sic auxiliary command.

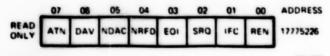


Figure 4-11 IEEE Bus Status Register

## Interrupt Mask Registers

The interrupt mask registers (Figure 4-12) are latched until new information is written into them.

NOTE: The interrupt mask registers are write only. If you think it may be necessary to read what interrupts are currently masked or unmasked during the course of your program, you may want to keep a local copy of the byte masks written to this register. The 9914A chip does not provide a means of interrogating the chip to determine what interrupts are currently enabled.

WR 1300

	07	06	05	04	03	02	01	00	REGISTER	ADDRESS
WRITE	×××	xxx	81	80	END	SPAS	ALC	MAC	INT MASK 0	17775220
ONLY	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC	INT MASK 1	177 <b>75222</b>

Figure 4-12 Interrupt Mask Registers 0 and 1. (WRITE ONLY)

Table 4-7 Interrupt Mask Register 0 Data

<b>Bit</b>	Signal	Set	Significance
7-6		XX	Not used May be either 0 or 1
5	BI	1	Causes an A interrupt when the Bi condition occurs
		0	Disables byte in interrupts
4	BO	1	Causes an A interrupt when the BO condition occurs
		0	Disables byte out interrupts
3	END	1	Causes an A interrupt on end of message conditions
		0	Disables end-of message interrupts
2	SPAS	1	Causes an A interrupt when the serial poll active state condition occurs
		0	Disables SPAS interrupts
1	RLC	1	Causes an A interrupt when the RTI changes its remote/local status
		0	Disables interrupts on remote to local and local to remote changes
0	MAC	'	Causes an A interrupt on my address change conditions. This occurs whenever the RTI recognizes its address (or the unlisten command) on the bus and the current state of the RTI changes. If secondary addressing is used, the interrupt does not occur when the secondary address is placed on the lines. It dual addressing is being used, the interrupt does not occur if the RTI is readdressed using its other primary address.
		0	Disable my address change interrupts

Table 4-8 Interrupt Mask Register 1 Data

8it	Signal	Set	Significance
7	GET	1	Causes an A interrupt when a group execute trigger interface command is recognized on the bus and the RTI is not the controller-in-charge. A DAC holdoff is entered which must be released using the dacr auxiliary command
		0	No interrupt occurs when the GET condition occurs.
6	ERR	1	Causes an A interrupt when the handshaking error condition occurs.
		0	No interrupt occurs when the ERR condition occurs.
5	UNC	1	Causes an A interrupt when the UNC condition occurs. A DAC holdoff is entered and must be released using the dacr auxiliary command. The unrecognized command can be read during the holdoff by reading the command pass through register.
		0	No interrupt occurs when the UNC condition occurs.
4	APT	1	Causes an A interrupt when the APT condition occurs A DAC holdoff is entered and must be released using the dacr auxiliary command. Before releasing the holdoff, your program can read the secondary address from the command pass through register. The C/S bit of the DACR command is used to indicate whether the secondary address was valid or invalid. If valid the C/S bit is set to 1; if invalid, the C/S bit is set to 0. See "Auxiliary Commands." When the dacr command is used for releasing other DAC holdoffs, the C/S bit has no meaning.
		0	No interrupt occurs when the APT condition occurs.
3	DCAS	1	Causes an A interrupt when the DCAS condition occurs. A DAC holdoff is entered and must be released using the dacr auxiliary command.
		0	No interrupt occurs when the DCAS condition occurs
2	MA	1	Causes an A interrupt when the My Address condition occurs. A DAC holdoff is entered and must be released using the dacr auxiliary command.
		0	No interrupt occurs when the MA condition occurs.
1	SRQ	1	Causes an A interrupt when the SRQ condition occurs When the RTI is not the controller in charge, this interrupt would normally not be enabled. (However, the SRQ status bi in interrupt status register 1 is always set when SRQ becomes true, regardless of whether the RTI is the controller-in-charge.)

WP 13000

Table 4-8 Interrupt Mask Register 1 Data (Cont.)

Bit	Signal	Set	Significance
		0	No interrupt occurs when the SRQ condition occurs
0	IFC	,	Causes an A interrupt when the IFC condition occurs. If the RTI is the system controller, no interrupt occurs even if the bit is set. (The IFC status bit is set, however, when the RTI is system controller.)
		0	No interrupt occurs when the IFC condition occurs

#### **Interrupt Status Registers**

The interrupt status registers (Figure 4-13) record the status of various conditions affecting the RTI as they occur. The conditions that are monitored are explained in Tables 4-9 and 4-10. The interrupt mask registers allow you to select which of the monitored conditions are enabled to cause A interrupts in the RTI interrupt logic (see Chapter 2). If one of the monitored conditions occurs, the bit in the interrupt status register is always set. If the corresponding condition is unmasked in the interrupt mask register, an A interrupt occurs. The interrupt status registers should be read in order to determine what condition caused the interrupt.

	07	06	05	04	03	02	01	00	REGISTER	ADDRESS
HEAD	INTO	INT1	81	80	END	SPAS	ALC	MAC	INT REG 0	17775220
JNLY	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC	INT REG 1	17775222

Figure 4-13
Interrupt Status Registers 0 and 1. (READ ONLY)

NOTE: You must read both status registers in order to clear the RTI's IRQ line.

The first two bits in the interrupt status register 0, INTO and INT1, are set only if an interrupt unmasked in the interrupt mask register occurs. They are not set if a masked condition occurs. See Interrupt Mask Registers.

The status bit representing the masked or unmasked condition is always set however.

Tables 4-9 and 4-10 describe the conditions causing each status bit to become set. Tables and describe the interrupt and holdoff, if any, caused when the unmasked condition occurs.

Both interrupt status registers are cleared when they are read. In addition, some individual status bits are cleared when the condition they reflect changes. These bits are described below.

Table 4-9 Interrupt Status Register 0

<b>O</b> II	Signal	Condition Set On
7	INTO	Interrupt 0. When set, this bit indicates that one or more of the interrupt bits in interrupt status register 0 has been set and that interrupt has been previously enabled (unmasked) in interrupt mask register 0.
6	INT1	Interrupt 1. When set, this bit indicates that one or more of the interrupt bits in interrupt status register 1 has been set and that interrupt has been previously enabled (unmasked) in interrupt mask register 1.
5	Ві	Byte In Condition. A new data byte is now in the data in register and can be read. Reading the data in register also clears this bit.
4	ВО	Byte Out Condition. The last data byte placed in the data out register has been accepted by the receiver and a new byte can be written to the data out register. Writing a new byte to the data out register also clears this bit.
3	END	End of Message Condition. This bit is set when the EOI line becomes true while the ATN is false. This combination is used to indicate the end of a data string.
2	SPAS	Serial Poll Condition. This bit is set when the serial poll active state has been entered. This occurs whenever the serial poll register is written with the rsv1 bit set or when the auxiliary command set rsv2 is written. (See Auxiliary Command Register for definitions of rsv1 and rsv2.)
1	RLC	Remote/Local Change Condition. When set, this bit indicates that a change in the remote/local status of the RTI has occurred. This bit is set when the RTI changes from either remote to local or local to remote. Note that this bit does not reflect the status of the REN line.
0	MAC	My Address Change condition. When set, this bit indicates that the RTI has received an interface command from the bus telling it to change its current addressed state. The bit is not set when a secondary address is received, nor is it set when dual primary addressing is being used and the other primary address is received. (See Address Register.)

Table 4-10 Interrupt Status Register 1

8it	Signal	Condition Set On
7	GET	Group Execute Trigger Condition. When set, this bit indicates that a group execute trigger interface command has been received from the controller in charge.
6	ERR	Error Condition. When set, this bit indicates that an error in the handshaking sequence has occurred. Specifically, it means that the DAV line is true and both the NDAC and NRFD lines are false. This indicates that no devices can currently receive data.
5	UNC	Unrecognized Command Condition. When set, this bit indicates that an uninterpretable command has been received. It may be that the command is valid, but not for the current state of the RTI. Uninterpretable commands cause this interrupt when the RTI is in LADS (listener addressed state) or in TADS (talker addressed state). If secondary commands are being used, they will set this bit if the PTS (pass through next secondary) auxiliary command has been written.
4	APT	Address Pass Through When this bit is set, it indicates that a secondary address has been received and can be read from the command pass through register. In order for the RTI to identify a secondary address, the previously received primary command must have been a primary talk or listen address which identified the RTI. If the next command is a secondary address that identifies the RTI, this bit is set.
3	DCAS	Device Clear - Active State When set, this bit indicates that the RTI has been placed in LADS (listener addressed state), but before any data transfer occurred, a device clear (DCL) or selected device clear (SDC) interface command was received
2	MA	My Address Condition. When a primary talk address or primary listen address interface command which the RTI recognizes as its own is on the bus, this bit is set.
1	SRQ	Service Request. This bit is set when the SRQ line becomes true It stays true until interrupt register 1 is read and does not become clear if the SRQ goes false before the read operation. This interrupt is provided for situations when the RTI is the controller in charge and a device on the bus has requested service.
0	IFC	Interface Clear This bit is set when the IFC line becomes true This is useful when the RTI is not the system controller as it indicates that the bus has been returned to the idle state. If the RTI is the system controller, the interrupt is not set by IFC becoming true.

# **Parallel Poll Register**

The parallel poll register (Figure 4-14) is used when the RTI is not the controller-in-charge and the controller-in-charge has requested a parallel poll. In a parallel poll, each device uses one of the eight data lines to indicate that it is the device that requested service. The convention that indicates which device uses what data line is up to the programmer and is not specified by the IEEE standard. When a bit is set in this register, the corresponding data line is put in the true state when the controller-in-charge requests a parallel poll. Typically, this register is written with only one bit set when the RTI is in need of service. The controller-in-charge then reads the eight data lines and identifies the RTI by the data line which is true.

The bits in the parallel poll register correspond directly to the data lines in ascending fashion. Bit PP1 is placed on the data line DIO1, bit PP2 is placed on data line DIO2, etc. If a bit in the parallel poll register is 1, the corresponding data line is pulled true (low).

The parallel poll register is double buffered and can be written to while a parallel poll is occurring. This register is not reset by the software reset auxiliary command.

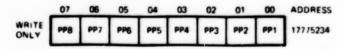


Figure 4-14
Parallel Poll Register

## Serial Poll Register

When the RTI is not the controller-in-charge, the serial poll register (Figure 4-15) is used to respond to serial polls by the controller-in-charge. This is a double-buffered register and may be written to at any time. Bit 6, the RSV1 bit, puts the SRQ line into the true state when it is set. Therefore, writing to this register with bit 6 set notifies the controller-in-charge to conduct a serial poll. The contents of this register are sent back to the controller-in-charge when a serial poll is conducted. Therefore, bits 0-5 and bit 7 can be used to convey status information. The convention and meanings assigned to these bits are defined by the program, not the IEEE standard. Bit 6 will of course be set, indicating that the RTI requested service.

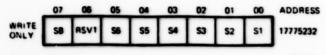


Figure 4-15 Serial Poll Register

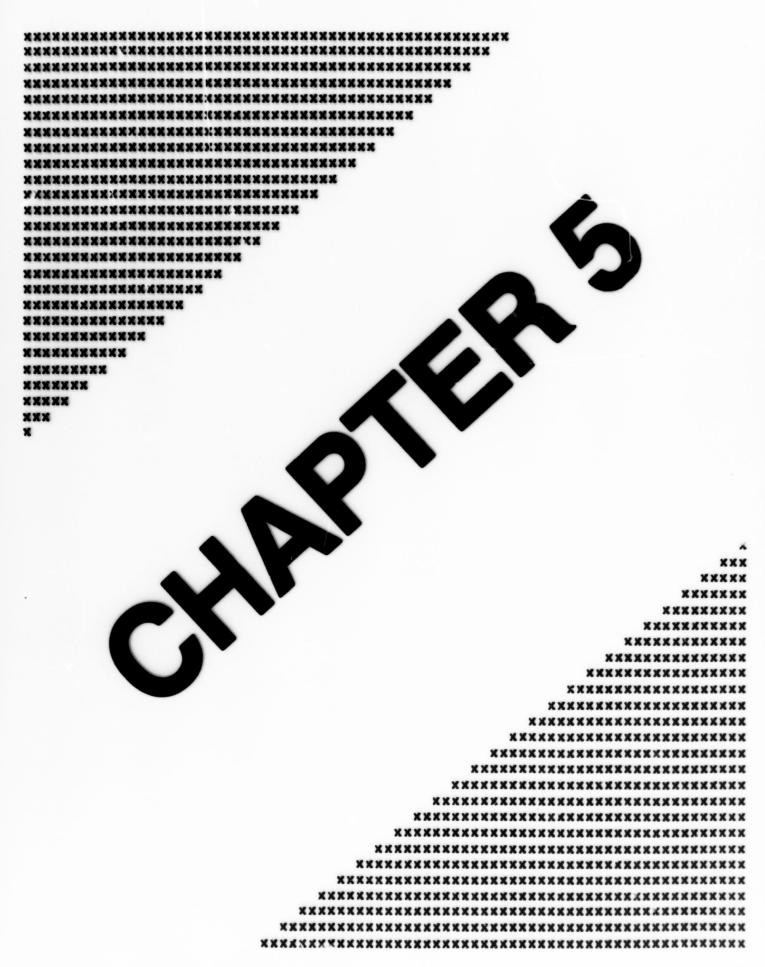
The bits in the serial poll register correspond directly to the data lines in ascending fashion. Bit S1 is placed on the data line DIO1, bit S2 is placed on data line DIO2, etc. If the bit in the serial line register is 1, the corresponding data line is pulled true (low). Thus DIO7 corresponds to bit 6 and, when true, indicates that the RTI requested service.

This register is double buffered, and if written to during a serial poll, no data is lost. The byte written to the register is saved and when the serial poll is complete, the new data byte is written to the register.

The rsv1 bit is not cleared when a serial poll is taken, but the SRQ line is returned to the false state. The serial poll register is not cleared by the software reset auxiliary command.

The rsv2 auxiliary command provides an alternative method of requesting service. Writing the set rsv2 command pulls the SRQ line true and sets bit 6 in the serial poll register. The controller-in-charge then reads the serial poll register when it responds to the serial poll. In this instance, bit 6 is cleared after the serial poll register is read. This provides a method of requesting that the controller read the serial poll register and allows you at any time to write status information to the register with bit 6 clear. The status information then remains in the register and can be read by the controller-in-charge when you write the rsv2 auxiliary command.





# 

Parallel Logic

# Chapter 5

# **Parallel Logic**

#### THEORY OF OPERATION

Figure 5-1 summarizes the parallel logic in block diagram form.

The parallel I/O logic is implemented with an Intel 8255A communications chip. There are three 8-bit data ports (port A, port B, and port C) that may be configured for either input or output. Three modes of operation are available. The first (mode 0) provides basic input and output communications with each data port operating independently. The second mode (mode 1) allows handshaking lines from the third port to be associated with the other two ports; this allows strobed input or output through two independent 8-bit data ports. The third mode (mode 2) allows bidirectional data flow through port A, with handshaking lines from port C.

The data section of the chip communicates with the bus interface over the DBUS 0 through DBUS 7 lines and transfers data to the three ports internally. The port select section of the chip utilizes the ADD1 and ADD2 signals to activate the desired port. The chip is enabled to receive data from remote devices and to transfer it to the host Professional computer by a combination of the SDEN and WRT signals. The combination of the DATA STB and WRT signals enables the chip to receive data from the host Professional computer and transfer it to the remote devices. The chip is reset by the INIT signal and selected by the SEL1 signal.

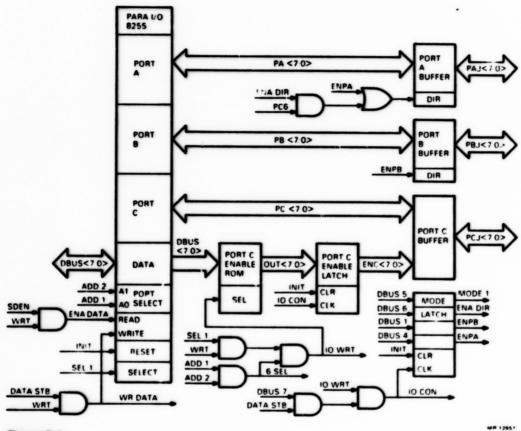


Figure 5-1 Parallel Logic Block Diagram

The mode latch uses the DBUS 5, DBUS 6, DBUS 1, and DBUS 4 signals to generate the MODE 1, Enable Direct (ENA DIR), Enable Port B (ENPB), and Enable Port A (ENPA) signals, respectively. The mode latch is cleared by the INIT signal and clocked by a combination of DBUS 7, DATA STB, and IO WRT, which generate the IO CON signal.

The port A buffer is enabled either by the ENPA or by a combination of the ENA DIR and PC6 signals. The port B buffer is enabled by the ENPB signal. The port C buffer requires a combination of the PC0 through PC7 and ENC0 through ENC7 signals in order to transfer data because the individual bits of port C may be used either for data or for handshaking signals, depending on which operating mode has been selected.

The port C enable ROM is selected by the IO WRT signal, which is produced by a combination of SEL1 and WRT; and ADD1 and ADD2, which generates 6 SEL. The ROM produces output signals labelled OUT 0 through OUT 7, and transfers them to the port C enable latch, which is cleared by the INIT signal and clocked by IO CON.

#### INPUT/OUTPUT MODES

The parallel I/O logic has three modes of operation, mode 0, mode 1, and mode 2. These are described in the following paragraphs.

#### Mode 0 Characteristics

Mode 0 is for basic input and output without handshaking control through two 8-bit data perts and two 4-bit data ports. The two 8-bit ports are labeled port A and port B. Port C is divided into two 4-bit ports which are addressable independently and are labeled port C upper and port C lower. Port C upper consists of bits 4 through 7 and port C lower consists of bits 0 through 3. Each port may be configured for either input or output and operate independently of each other. When configured for output, the data ports are latched. When configured for input, the data ports are not latched. Since there are four separate ports and each port can be either input or output, there are sixteen separate combinations of input/output configurations.

#### **Mode 1 Characteristics**

Mode I allows strobed input or output of data. In mode 1, there are two groups of data bits available for input or output. Each group consists of one 8-bit data port and one 4-bit port used for handshaking lines. Group A consists of the eight data bits in port A and the four bits in the upper part of port C (bits 4 through 7). Group B consists of the eight data bits in port B and the four bits in the lower part of port C (bits 0 through 3). In each group, the 8-bit data port is used for either input or output or data. The data lines are latched on both input and output. The associated four control lines are specifically defined as handshaking signals for controlling the flow of data either in or out of the data port.

#### **Mode 2 Characteristics**

Mode 2 allows bidirectional data flow through the eight bits of port A. Handshaking lines to control the data flow are implemented in port C. Only the group A ports can be configured for mode 2 operation. That is, port A and the upper four bits of port C can be used for mode 2, but group B bits are not available for this mode. This is because one bit from the lower portion of port C is used to generate interrupts internally. Port B and bits PC0 through PC2 are available for mode 1 or mode 0 operation, however, when port A is configured for mode 2 operation.

In mode 2, as in mode 1, all data lines are latched for input and output. In addition, the same bits in port C are used for handshaking lines in mode 2 as in mode 1.

#### PROGRAMMING INFORMATION

Ports A, B, and C are accessed by reading from or writing to the three port registers. A control register is also provided for configuring the parallel ports and controlling individual bits in port C. These registers are described below.

# **Port Registers**

Figures 5-2 through 5-4 illustrate the three registers used for the three data ports. In either mode, ports A and B are addressable by writing or reading to the listed address. Port C can always be read by using the listed address. Individual bits in port C can be set or cleared without affecting the other bits in the port. See Port C Bit Set/Reset Commands.

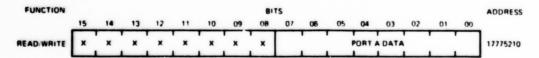


Figure 5-2 Port A Register

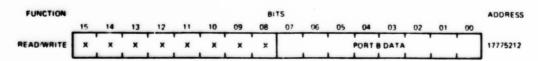


Figure 5-3 Port B Register

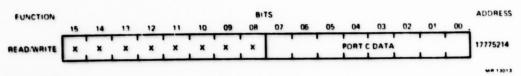


Figure 5-4
Port C Register

# **Control Register**

Control over the parallel logic is performed by writing commands to the control register, which cannot be read. The control register (Figure 5-5) functions in two modes, controlled by whether bit 7 is set or cleared.

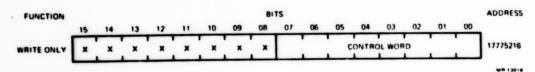


Figure 5-5 Control Register

#### **Mode Control Commands**

When a command is written to the control register with bit 7 set, the command is interpreted as a mode control command. The mode control command sets up the desired mode (0, 1, or 2) for each port and defines the direction of data flow for each port. Ports A and B can be set up in different modes, if desired.

At power-up INIT all ports are set to input, mode 0.

The meaning of each bit in a mode control command is shown in Table 5-1.

Table 5-1
Control Register Data—Mode Control Commands

Bit Number	Value	Significance
7	1	Enable Mode Control (The bits are interpreted as shown below.)
		Group A Control
6-5	65	Port A Mode Selection
	00	Mode 0
	01	Mode 1
	10	Mode 2
	11	Not used
4		Port A Direction (ignored if mode 2 is selected)
	1	Input
	0	Output
3		Port C (Bits 7-4) Direction
	1	Input
	0	Output
		Group B Centrol
2		Port B Mode Selection
	0	Mode 0
	1	Mode 1
1		Port B Direction
	1	Input
	0	Output
0		Port C (Bits 3-0) Direction
	1	Input
	0	Output

Note that the modes for port A and port B can be separately defined, but no mode can be specified for port C. When mode 1 is specified for port A or port B, the associated upper or lower part of port C is automatically configured for handshaking lines. If port A or port B is configured for mode 0, the associated upper or lower part of port C is automatically configured for mode 0.

The control register should be written only once for each data acquisition environment. If you want to use more than one port at a time, set up all ports the first time you write to the control register. The control register cannot be read.

#### Port C Bit Set/Reset Commands

When a command is written to the control register with bit 7 clear, the command is interpreted as a port C set/reset command. An individual bit in port C can be set or reset by this command; the other bits in port C are unaffected. Only one bit in port C can be changed at a time. The meaning of each bit in a port C set/reset command is shown in Table 5-2.

Table 5-2
Control Register Port C Bit Set/Reset Commands

Bit Number	Value	Significance
7	0	Signifies command is a port C bit set/reset command
6	0	Not used, but must be 0
5-4	54 00 01 10	Mode definition Identifies the current configuration being used Mode 0 Mode 1 Port A Input Mode 1 Port A Output 1 ode 2
3-1	321 000 001 010 011 100 101 110	Bi Selected 0 1 2 3 4 5
0	1 0	Bit Command Set Reset

#### Port C I/O in Mode 1 or Mode 2

When port A and port B are programmed in mode 1, two bits in port C are unused for handshaking lines. By using the bit set/reset command, these bits can be used for data input or output, or for additional customized hand shaking. When port A is configured for mode 1 input, bits 6 and 7 in port C (PC6 and PC7) can be used for input or output. When port A is configured for mode 1 output, bits 4 and 5 in port C (PC4 and PC5) can be used for input or output. If only one of either port A or port B is being used for mode 1, the port C bits

unused for handshaking may be used for I/O. For example, if port A is being used for mode 1 input, and port B is in mode 0 or not used, all of the lower portion of port C is available for I/O. If port A is programmed for mode 2 and port B is not being used, bits PC0 through PC2 are available for I/O.

#### Use of Port C as a Status Word

Port C can always be read regardless of the current mode configuration of the parallel ports. Reading port C in mode 1 or mode 2 can provide with you the status of all the associated handshaking lines. This allows you to check the status of external devices' handshaking lines and alter program flow in the event of a hangup or other unwanted situation.

#### MODE 0 OPERATION

Mode 0 operation provides basic input or output through ports A, B, upper-C, or lower-C. No handshaking signals are used. Interrupt driven data transfer is not available. Each port may be separately configured for input or output.

There are sixteen different combinations of input and output ports in mode 0. These combinations are selected by the combinations of bits 0, 1, 3, and 4 in commands written to the control register. Bits 2, 5, and 6 select mode 0 for ports A and B when clear. Bit 0 is associated with port C (lower). Bit 1 is associated with port B. Bit 3 is associated with port C (upper). Bit 4 is associated with port A. A 1 in the specified bit selects input and a 0 in the specified bit selects output. For example, to set port A and port C (upper) to input, and port B and port C (lower) to output, write a command byte to the control register which has bits 3 and 4 set to 1, and bits 0 and 1 set to 0. (Bits 2, 5, and 6 are all clear for mode 0.)

#### **MODE 1 OPERATION**

Mode 1 operation provides stobed input or output of data. Handshaking lines are used in port C as described below. Interrupt driven data transfers are possible; the interrupt can be enabled or disabled.

#### **Mode 1 Input Handshaking Signals**

In mode 1, there are two handshaking signals for controlling input from an external device to the RTI. These are the strobe (STB) and input buffer full (IBF) signals. Both signals are implemented in port C and are associated with the data lines of either port A or port B as described below.

Strobe is sent from the external device to the RTI to indicate that the external device has just placed valid data on the data lines. When this signal goes low for at least 500 nanoseconds, then goes high, the data present on the eight data lines is latched. For port A in mode 1 input, bit 4 in port C (PC4) is used for the strobe signal. For port B in mode 1 input, bit 2 in port C (PC 2) is used for the strobe signal.

The input buffer full signal serves as an acknowledgment signal from the RTI to the external device. The IBF signal uses the high state as a true condition. IBF goes high when the RTI detects a strobed input pulse. IBF goes low again when the data has been read from the port register. For port A in mode 1 input, bit 5 in port C (PC5) is used for the IBF signal. For port B in mode 1 input, bit 1 in port C (PC1) is used for the IBF signal.

A timing diagram showing the relationship between data flow and the strobe and input buffer full signals is shown in Figure 5-6.

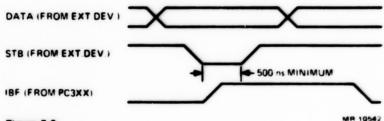


Figure 5-6 Handshaking Timing Diagram Input

#### Mode 1 Input Interrupts

In mode 1, the parallel port can generate interrupts to the CPU, using the IRQ B interrupt line. The interrupt can be enabled or disabled by writing a port C bit set/reset command.

The input interrupt signal, when enabled, sets an interrupt flag in the interrupt B register when the strobe signal is currently false (high) and the input buffer full signal is currently true (high). This combination indicates that the external device has data available to be read by the RTI. The interrupt signal is reset when the associated port register is read.

The interrupt signal can be enabled or disabled for port A and port B individually. To enable the input interrupt for port A, write a port C bit set/reset command to the control register which sets bit PC4. See Table 5-2.

To enable the input interrupt for port B, write a port C bit set/reset command to the control register which sets bit PC2. See Table 5-2.

The input interrupt uses bit PC3 (for port A) and bit PC0 (for port B) internally. Therefore these bits are not available for input/output purposes when interrupts are enabled in mode 1. If interrupts are enabled in mode 1, they must be disabled before entering mode 0 operation.

NOTE: Repeated reading of the port register by the Professional computer's CPU without intervening STB pulses causes the IBF signal to toggle high. The INTR signal is not affected.

## **Mode 1 Output Handshaking Signals**

In mode 1, there are two handshaking signals for controlling output from the RTI to the external device. These are the output buffer full (OBF) signal and the acknowledge (ACK) signal. Both signals are implemented in port C and are associated with the data lines of either port A or port B as described below.

Output buffer full signal is sent from the RTI to the external device and is used to indicate that the RTI is ready to send data. It can be thought of as a strobe signal to the external device. When data is written to the output port register, the OBF signal is asserted low. The OBF signal is reset high when the external device sends an acknowledge signal (see next paragraph). For port A in mode 1 output, bit 7 in port C (PC7) is used for the OBF signal. For port B in mode 1 output, bit 1 in port C (PC1) is used for the OBF signal.

The acknowledgment signal, used to indicate that the device has received the data from the RTI, is sent to the RTI by the external device. To function properly, the external device must assert this signal low for a minimum of 300 nanoseconds. For port A in mode 1 output, bit 6 in port C (PC6) is used for the ACK signal. For port B in mode 1 input, bit 2 in port C (PC2) is used for the ACK signal.

A timing diagram showing the relationship between data flow and the output buffer full and acknowledge signals is shown in Figure 5-7.

**NOTE:** Reading an output port (by the Professional computer's CPU) in mode 1 will reset the OBF signal if it is low. Output data is not affected, however.

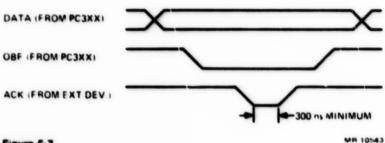


Figure 5-7
Handshaking Timing Diagram Output

#### **Mode 1 Output Interrupts**

In mode 1, the parallel port can generate interrupts to the CPU, using the IRQ B interrupt line. The interrupt can be enabled or disabled by writing a port C bit set/reset command.

The output interrupt signal, when enabled, sets an interrupt flag in the interrupt B register when both the acknowledge (ACK) signal and the output buffer full signal are currently false (high). This combination indicates that the external device has received the last data sent and is ready for the next data byte. The interrupt signal is reset when data is written to the associated port register.

The interrupt signal can be enabled or disabled for port A and port B individually. To enable the output interrupt for port A, write a port C bit set/reset command to the control register which sets bit PC6. See Table 5-2.

To enable the output interrupt for port R, write a port C bit set/reset command to the control register which sets bit PC2. See Table 5-2.

The output interrupt uses bit PC3 (for port A) and bit PC0 (for port B) internally. Therefore these bits are not available for input/output purposes when interrupts are enabled in mode 1.

The final interrupt must be handled after the last valid data is written to an output port or read from an input port.

If interrupts are enabled in Mode 1, they must be disabled before entering Mode 0 operation.

#### **MODE 2 OPERATION**

Mode 2 operation provides the capability of bidirectional strobed input output. This allows the RTI both to transmit data to and to receive data from remote devices over an 8-bit bus.

Mode 2 operation is available only for the group A port. The bits used in mode 2 operation are the eight bits of port A, used for bidirectional data transfer; the four bits of upper port C, used for handshaking signals; and bit PC3, used internally for interrupt generation.

## Mode 2 Input/Output Handshaking Signals

Mode 2 uses the same handshaking signals for input and output as port A in mode 1 does. There are two handshaking signals for controlling input from an external device to the RTI. These are the strobe (STB) and input buffer full (IBF) signals. Both signals are implemented in port C.

Strobe is sent from the external device to the RTI to indicate that it has just placed valid data on the data lines. When this signal goes low for at least 500 nanoseconds, then goes high, the data present on the eight data lines is latched. For mode 2 input, bit 4 in port C (PC4) is used for the strobe signal.

The input buffer full signal serves as an acknowledgment signal from the RTI to the external device. The IBF signal uses the high state as a true condition. IBF goes high when the RTI detects a strobed input pulse. IBF goes low again when the data has been read from the port register. For mode 2 input, bit 5 in port C (PC5) is used for the IBF signal.

A timing diagram showing the relationship between data flow and the strobe and input buffer full signals is shown in Figure 5-6.

In mode 2, there are two handshaking signals for controlling output from the RTI to the external device. These are the output buffer full (OBF) signal and the acknowledge (ACK) signal. Both signals are implemented in port C.

Output buffer full is sent from the RTI to the external device and is used to indicate that the RTI is ready to send data. It can be thought of as a strobe signal to the external device. When data is written to the output port register, the OBF signal is asserted low. The OBF signal is reset high when the external device sends an acknowledge signal (see next paragraph). For mode 2 output, bit 7 in port C (PC7) is used for the OBF signal.

The acknowledgment signal, used to indicate that the device has received the data from the RTI, is sent to the RTI by the external device. To function properly, the external device must assert this signal low for a minimum of 300 nanoseconds. For mode 2 output, bit 6 in port C (PC6) is used for the ACK signal.

A timing diagram showing the relationship between data flow and the output buffer full and acknowledge signals is shown in Figure 5-7.

# **Mode 2 Interrupts**

Mode 2 interrupts operate in a manner similar to mode 1 interrupts for port A. The interrupts can be enabled or disabled by writing a port C bit set/reset command.

The input interrupt signal, when enabled, sets an interrupt flag in the interrupt B register when the strobe signal is currently false (high) and the input buffer full signal is currently true (high). This combination indicates that the external device has data available to be read by the RTI. The interrupt signal is reset when the associated port register is read.

The interrupt signal can be enabled or disabled. To enable the input interrupt for mode 2, write a port C bit set/reset command to the control register which sets bit PC4. To disable the input interrupt, use the bit set/reset command to clear PC4. See Table 5-2.

An output interrupt is also provided. The interrupt can be enabled or disabled by writing a port C bit/set reset command.

The output interrupt signal, when enabled, sets an interrupt flag in the interrupt B register when both the acknowledge (ACK) signal and the output buffer are currently false (high). This combination indicates that the external device has received the last data sent and is ready for the next data byte. The interrupt signal is reset when data is written to the associated port register.

The interrupt signal can be enabled or disabled. To enable the output interrupt for mode 2, write a port C bit set/reset command to the control register which sets bit PC6. To clear the output interrupt, write a bit set/reset command which clears PC6. See Table 5-2.

The final interrupt must be handled by your program after the last valid data is written to an output port or read from an input port.

The mode 2 interrupts use bit PC3 internally. Therefore this bit, although it is part of group B, is not available for input/output purposes when interrupts are enabled in mode 2.

If interrupts are enabled in mode 2, they must be disabled before entering mode 0 operation.

#### **COMBINATIONS OF MODES**

The four ports (A, B, upper C, lower C) can used simultaneously in different modes. That is, port A can be used for mode 1 input while port B is used for mode 0 output. Port B can be used for mode 1 input or output regardless of the configuration of port A. If ports A and B are used in mode 0, port C can be used in mode 0.

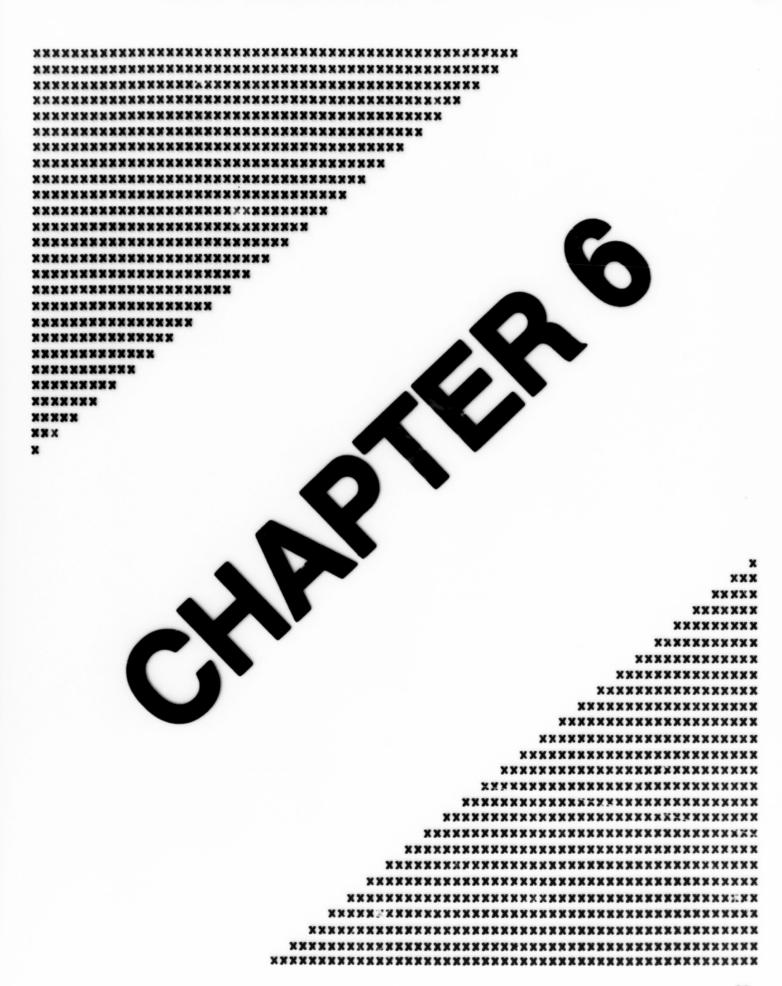
#### NOTES ON CONNECTING EXTERNAL DEVICES FOR HANDSHAKING SIGNALS

External devices generating handshaking signals may or may not use the same terminology as that described above. For example, the acknowledge signal may be called Ready for Data, or some other term. However, if the signal serves the purpose described above and conforms to TTL standard levels, it can be used for the handshaking signal.

To connect the external device's handshaking signal, connect a wire from the signal's terminal to the appropriate bit in port C. See Table 5-3 for the bits used. Also connect the terminal's ground to a port C ground pin. (See Figure 1-2.)

Table 5-3 Location of Handshaking Signals in Port C—Modes 1 and 2

Direction of Transfer	Signal	Port A	Port B
Input	STB	PC4 PC5	PC2 PC1
Output	OBF	PC7 PC6	PC1 PC2



# 

Self-Test ROM Logic and Diagnostics

# Chapter 6

# Self-Test RC® Logic and Diagnostics

#### THEORY OF OPERATION

Figure 6-1, Self-Test ROM Logic Block Diagram, summarizes the operation of the self-test ROM logic and diagnostics.

The self-test ROM logic consists of a 12-bit counter, a 4K × 8 Erasable Programmable ROM (EPROM), and enabling logic. The counter is reset to zero by the power-up INIT signal. The combination of the ADD1, WRT, SEL0, and ADD2 signals generates the LD ROM ADD signal, which loads the counter with address 0. The combination of the ADD1 and WRT signals, or the SEL0 and ADD2 signals, generates the READ ROM signal. READ ROM, together with Slave Drive Enable (SDEN) generates the ENABLE ROM signal, which enables the ROM and clocks the counter. Each time the ENABLE ROM signal occurs, the ROM transfers a byte of data to the Professional computer's CPU, and the counter increments to point to the next byte in the ROM. The program contained in the ROM is thus loaded into the Professional computer's CPU and run to check out the three logic interface types on the Real-Time Interface Module. The PC3XX-AA Real Time Interface Owner's Manual continues an explanation of the possible error codes in the event the self-test ROM detects a problem.

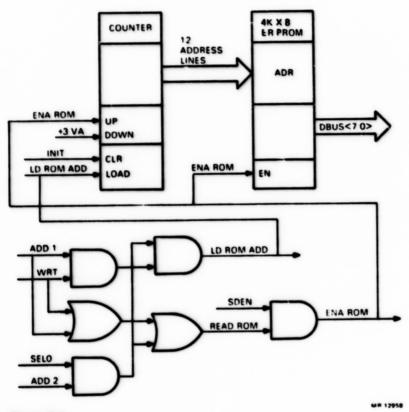


Figure 6-1 Self-Test ROM Logic Block Diagram

NOTE: After the power-up self-tests have run, the outputs of the interfaces are in the conditions listed below. The voltages are approximate and are valid only if no external device is connected.

- ☐ Serial Interface—Unasserted, High level: +5 volts minimum

  Low level: -5 volts maximum
- ☐ IEEE Interface—Unasserted, Output High: 2.5 volts minimum
  Output Low: 0.5 volts maximum
- ☐ Parallel Interface—Unasserted, three-state condition (high impedance)

#### PROGRAMMING INFORMATION

The ROM logic contains a data register and an address register. Both registers are cleared at power-up and at bus INIT. See Figures 6-2 and 6-3 for the information contained in each register.

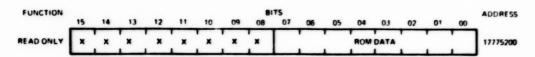


Figure 6-2 ROM Data Register

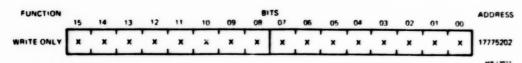


Figure 6-3 ROM Address Register

## **ROM Data Register**

The low byte of the ROM data register is a data window into the ROM. Each time the ROM data register is accessed, the next location of the ROM data is made available.

# **ROM Address Register**

The ROM address register is used to select address 0 of the onboard ROM. This register is cleared when any number is written into it. The CPU Bus INIT also clears this register.

#### LOOPBACK CONNECTOR

A loopback connector (part number 12-21246-01) is available as an option. It operates in the service mode as described in the *PC300 Series Pocket Service Guide* (Order Number EK-PC350-PS-001). The *PC3XX-AA Real-Time Interface Module Owner's Manual* contains information on ordering the loopback connector and Pocket Service Guide. The loopback connector is designed to be used with the diagnostic diskette and provides more extensive testing of the Real-Time Interface Module by looping SLU 1 of the serial logic back to SLU 2, and Port A of the parallel logic back to Port B. The loopback connector does not provide loopback testing for the IEEE-488 port.

#### **DIAGNOSTIC DISKETTE**

In order to perform diagnostic testing on the Real-Time Interface Module, the P/OS Installation and Maintenance diskette must first be updated to include the RTI diagnostics. The updating procedure is as follows.

# Diskettes Required

- 1. P/OS System Diskette V1.7 or later
- 2. Installation and Maintenance Diskette
- 3 Maintenance Test Diskette
- 4. Real-Time Interface Module Diagnostic Diskette

NOTE: Instead of a separate Maintenance Test Diskette, you may have a Professional Installation and Maintenance Diskette Number 1 and Diskette Number 2. In this case, the updating procedure is exactly the same. Simply substitute Number 2 wherever the Test diskette is called for in the following procedure.

# **Updating Procedure**

- Put the P/OS system diskette into drive 1 and turn the power on.
   The system runs the power-up self-test, and if successful, will display the Digital logo in reverse video and load the operating system into memory.
- 2. A prompt appears to load an application program. Remove the system diskette, replace it with the installation and maintenance diskette, and press RESUME. The maintenance services program is loaded and you are prompted to place the test diskette in drive 2.
- The maintenance services menu is displayed. Select update services and press DO. Press DO again to continue.
- 4. Remove the test diskette from drive 2 and replace it with the diskette containing the new program. Place the RTI diagnostic diskette in drive 2 and press RESUME. The RTI diagnostic is copied onto the installation and maintenance diskette. "UPDATE MAINTENANCE SERVICES" is displayed on the screen.

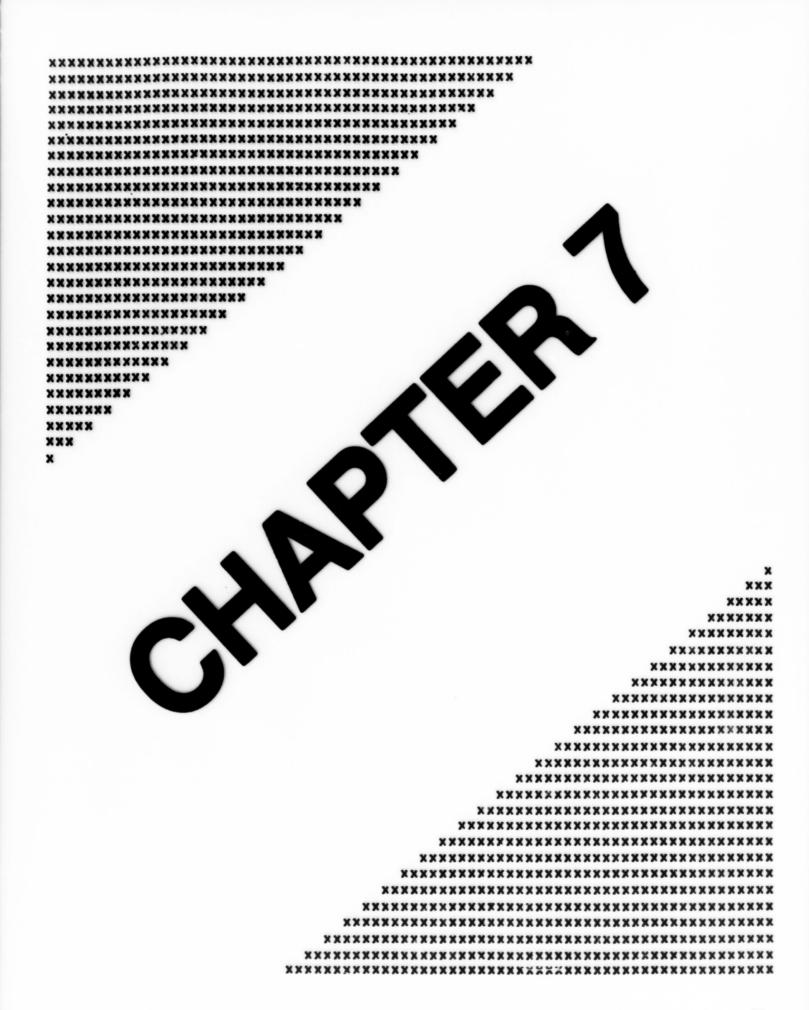
5. Remove the RT1 diagnostic diskette from drive 2 and file it in a safe place. Place the test diskette back in drive 2 and press RESUME. The RT1 diagnostic is copied from the installation and maintenance diskette onto the test diskette. When this is successfully completed, the following screen display appears:

#### "SUCCESS"

"The installation of the diagnostics is now complete."

The procedure for running the Real-Time Interface Module diagnostics is found in the *PC3XX-AA Real-Time Interface Owner's Manual* (EK-PC3AA-OM).

PAGE 96 INTENTIONALLY LEFT BLANK



## 7 The Angle

The Analog Data Module

## Chapter 7

## The Analog Data Module

#### OVERVIEW

The Analog Data Module (ADM), shown in Figure 7-1, allows real-time collection of analog data. The data is sampled and converted to representative digital values which can then be stored and processed easily.

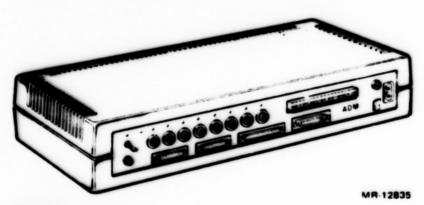


Figure 7-1 The Analog Data Module

The ADM is designed to be run in conjunction with a Professional 300 series computer and a PC3XX-AA Real-Time Interface Module. The Real-Time Interface Module is installed in the card cage of the Professional computer and connects to the ADM through a 62-pin cable supplied with the ADM.

Many laboratory devices produce data in analog form. A continuously varying voltage represents the data to be collected. In general, these laboratory devices also make use of a triggering signal or status signal to convey additional information about the data or the data transfer.

The following paragraphs highlight the features of the ADM.

#### HIGH RESOLUTION ANALOG INPUTS

The ADM features eight channels of true differential analog input. The ADM converts analog signals into 16-bit digital values, offering a resolution of one part in 65,536 (96 dB). Input values can range between -5 volts and +5 volts. Protection circuitry prevents damage to the unit for input signals up to 24 volts.

#### SAMPLING SPEED

The ADM hardware is capable of a peak analog sampling rate of 5000 samples per second. A continuous sampling rate will be less than this, and depends upon the specifics of your application and the amount of software processing you do.

#### TRUE DIFFERENTIAL INPUT AND AUTO-ZEROING CIRCUIT

The ADM features true differential input which provides a high common-mode rejection ratio and reduces the problem of noise-induced error.

Most data producing devices have two terminals: one positive terminal which produces the data representative voltage, and one minus or return terminal for completing this circuit. Some A/D converters take as input only the positive terminal signal and measure its potential difference to ground as data. The ADM's differential preamplifier takes both sides of the data device's signal as input and measures the difference between their respective voltages as data.

The ADM also features an auto-zeroing circuit which subtracts all input offset voltages attributable to the preamplifier.

#### PROGRAMMABLE PREAMPLIFIER WITH AUTORANGING

The ADM also features a programmable gain preamplifier which extends dynamic range to 132 dB, and allows low-level analog signals to be converted with greater resolution. The preamplifier may be software loaded with a desired gain, or it may be programmed to determine the appropriate gain automatically on a real-time basis. Gains of 1, 4, 16 and 64 are provided.

#### PROGRAMMABLE REAL-TIME CLOCK

A programmable real-time clock allows the collection of data samples to occur at a software-selected frequency. The clock may be set to operate at any frequency between 0 and 5 kHz for analog data and between 0 and 128 kHz for digital data. The slowest selectable clock period is 16.4 seconds. Once set, a data point is sampled and converted at every tick of the clock.

#### TRACK-AND-HOLD CIRCUITRY

The ADM employs a track-and-hold circuit for analog conversions. The circuit continually tracks or follows the output of the ADM preamplifier. When a command to convert a sample is received, the voltage level is held constant while the A/D converter circuit generates the digital value.

The track-and-hold circuitry helps increase the speed of the ADM. The hold circuitry helps prevent the converter from misinterpreting a rapidly changing signal. A signal which varies more than 153 microvolts during the conversion time would be misinterpreted if there were no hold circuit. The measurement of the least significant bits would vary from what the actual value was at the time the signal was acquired. (The least significant bits of the digital value are determined last.)

#### TRIGGERING MODES

Several types of analog data collections are available. The ADM allows collection of one data sample, multiple data samples on one channel, or multiple data samples on multiple channels (called sweeps). In addition, you can select how data collections begin; either a software trigger, where collection begins as soon as a command byte executes; or a hardware trigger, where collection begins when the ADM detects a negative voltage transition on the digital input strip. (See Digital Input and Output, below.) Collection of multiple samples can be controlled by the ADM's real-time programmable clock or by subsequent hardware triggers.

#### DIGITAL INPUT AND OUTPUT

The ADM contains one 8-bit digital input barrier strip and one 8-bit digital output strip. These general purpose ports can be used for a variety of tasks including hardware triggering of data acquisition, collecting status information from devices, collecting 8-bit digital data, and writing commands to external devices.

The digital input barrier strip features Schmitt triggers for more versatile applications. Schmitt triggers respond to voltage levels rather than transitions, and can therefore detect as a transition a signal which changes slowly. A logical 0 is set on a digital bit when a signal drops below 0.8 volts. A logical 1 is set when a voltage rises above 1.6 volts. This allows the digital input strip to respond to Transistor-Transistor Logic (TTL) level signals and nonstandard signals.

Each digital input bit is associated with an analog channel for hardware triggering. (See Triggering Modes, above.) When a transition from logical level 1 to logical level 0 (voltage drops below 0.8 volts) is detected on a digital input bit, data collection begins on the associated analog channel. This is true only for hardware triggered data acquisitions.

The digital output barrier strip uses Transistor-Transistor Logic (TTL) for output. This standard defines a logical 0 (TTL Low) as a voltage of less than 0.5 volts and a logical 1 (TTL High) as a voltage of greater than 2.4 volts. Devices which have input ports that conform to TTL can be connected directly to the ADM's digital output barrier strip.

The digital output barrier strip also features three-state output so that a customized bus can be designed. Three-state output is a high impedance state which is neither TTL Low nor TTL High.

#### IEEE AND SERIAL LINE PORTS

Connectors on the ADM's panel allow the IEEE-488 bus and serial line unit ports of the Real-Time Interface Module to be accessed. The ADM passes these connections directly to the Real-Time Interface Module and performs no processing on them. This allows simultaneous use of the ADM with IEEE bus and serial data transmission. For information on the use of these ports and the types of cables available for device connections, refer to the PC3XX-AA Real-Time Interface Module Owner's Manual (EK-PC3AA-OM-002).

#### **ADM PANEL**

The ADM panel is used for making all connections to the ADM. Figure 7-2 shows the ADM panel and identifies each feature.

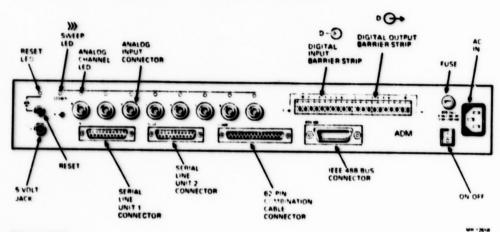


Figure 7-2 The ADM Panel

#### Reset LED

This LED, when lit, indicates that a reset condition exists. Data collection is halted while this light is on.

#### Sweep LED

This LED, when lit, indicates that the ADM is currently collecting data in sweep mode. (See Chapter 3 for definitions of data acquisition modes.)

#### **Analog Channel LEDs**

These LEDs, when lit, indicate that the ADM is currently sampling data on the indicated channel.

#### **Analog Input Connectors**

The analog input connectors are used for collecting analog data. Cables and connectors are provided for making connection to these connectors. (See Chapter 4 for details on constructing the cables.)

#### **Digital Input Barrier Strip**

The digital input barrier strip allows easy connection to the digital input port. The strip consists of ten pushtab connectors: one pushtab for each bit in the digital input port and two pushtabs for ground references. Connections are made by pushing in the pushtab and inserting the conductor. Releasing the pushtab holds the conductor in place. Be sure to strip back any insulation on your conductor before inserting it into the pushtab.

#### **Digital Output Barrier Strip**

The digital output barrier strip allows easy connection to the digital output to the strip consists of ten pushtab connectors: one pushtab for each bit in the digital output port and two pushtabs for ground references. Connections are made by pushing in the pushtab and inserting the conductor. Releasing the pushtab holds the conductor in place. Be sure to strip back any insulation on your conductor before inserting it into the pushtab.

#### Fuse

The ADM's fuse is a 250 volt 2 ampere slow-blow fuse for 120 volt environments and a T2.0 ampere fuse for 240 volt environments. (The ADMPC-A2 is designed for a 120 volt environment; the ADMPC-A3 is designed for a 240 volt environment.) If the fuse needs to be replaced, do not replace it with a fuse different from the original.

#### AC In

The AC In receptacle is used to provide power to the ADM.

#### On/Off

On the On/Off switch, 1 indicates ON, 0 indicates OFF.

#### IEEE-488 Bus Connector

This connector provides access to the IEEE-488 bus on the Real-Time Interface Module. The ADM does not process the IEEE signals on this port but passes them directly to the Real-Time Interface Module. This connector does not allow the ADM to be specified as a device on an IEEE bus.

#### 62-Pin Combination Cable Connector

One end of the combination cable (part number: 17-00036-02) attaches to this connector; the other end attaches to the internal cable of the Real-Time Interface Module.

#### Serial Line Units 1 and 2 Connectors

The serial line unit connectors, like the IEEE bus connector, are passed directly to the serial line unit ports on the Real-Time in orface Module. The ADM does not process these signals in any way.

#### +5 Volt Jack

This jack provides +5 volts of power at 0.25 amperes and is for Digital Field Service use only. The jack is fuse protected, but the fuse is not user replaceable. Users making connection to this jack do so at their own risk.

#### Reset

The reset switch is a momentary contact toggle switch. When depressed, it provides the same function as a power-off, power-on cycle for the ADM.

When the reset switch is depressed, all input and output processing ceases, the clock stops, and any data in the hardware data buffer is cleared. The digital output strip is set to tri-state mode. After a reset, the ADM should be reinitialized using the AINIT subroutine of the PRTIL software.

WARNING: Use of the RESET switch during an application is unpredictable. If an application becomes hung and RESET is used to clear the ADM, the application should be restarted to ensure data collection proceeds properly.

#### ANALOG DATA MODULE OWNER'S MANUAL

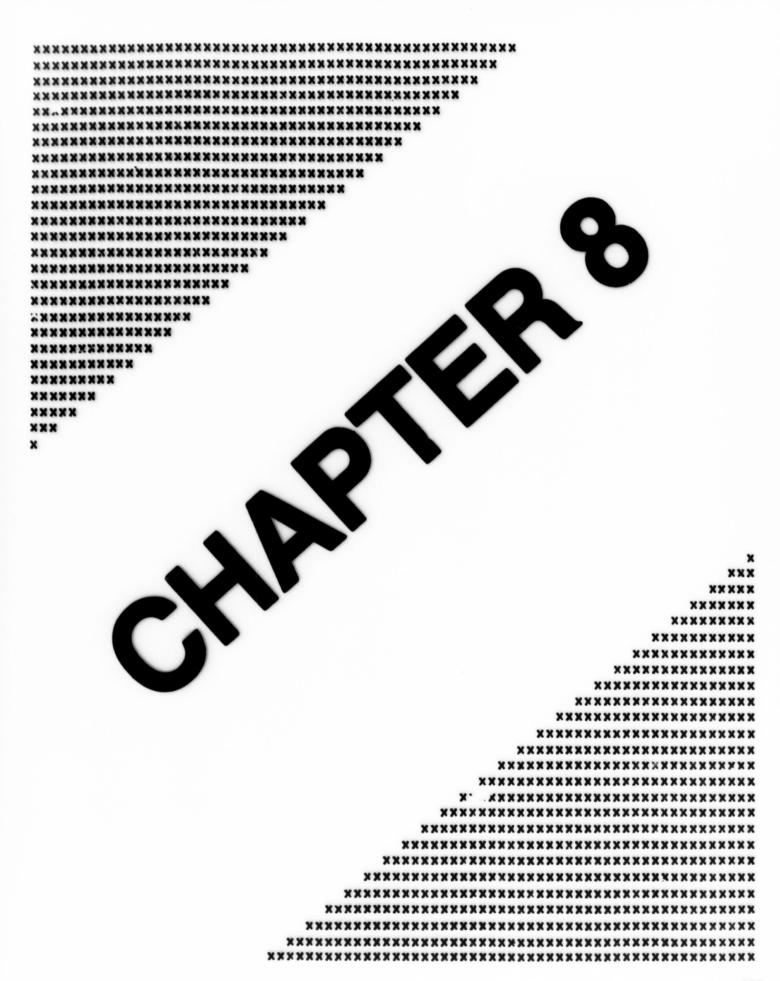
The Analog Data Module Owner's Manual supplied with the ADM provides additional information about the ADM. However, Chapter 3 of the Analog Data Module describes triggering modes and data acquisition features which require the use of the Professional Real-Time Interface Library or PRO/Tool Kit Real-Time Interface Library (both abbreviated as PRTIL). Information provided in Chapter 3 of the Analog Data Module Owner's Manual may or may not be true when programming the ADM in assembly language. Some of the triggering modes described in the Analog Data Module Owner's Manual are provided by the PRTIL software, not the hardware. You should, therefore, rely on this manual for information pertaining to triggering modes when programming the ADM in assembly language.

The information in the other chapters of the Analog Data Module Owner's Manual, including the information on the ADM's front panel, constructing analog cables, and using the digital input/output barrier strip, is all valid when programming the ADM in assembly language.

#### **CABLING CONNECTIONS**

The ADM is connected to the RTI's internal cable using the 62-pin cable supplied with the ADM. This cable carries all of the RTI's input/output signals. On the ADM, however, the IEEE-488 bus connector and the two SLU connectors are pass-through connectors only. The ADM does not process signals on these interfaces in any way. The ADM is controlled through the RTI's parallel I/O port. When you use the ADM with the RTI, the parallel port is not available for independent programming.

The Analog Data Module Owner's Manual provides full instructions for constructing analog signal cables and making connection to the analog channels and digital input/output barrier strips.



# 

Theory of Operation

### Chapter 8

### Theory of Operation

#### **CONVERTING ANALOG DATA**

The ADM is primarily designed to collect analog data: however, digital data can be collected as well. This is discussed in Chapter 10. The following discussion describes how the ADM converts a single analog sample. Figure 8-1 provides a block diagram overview of the ADM.

When an external voltage is connected to one of the ADM's analog input channels on the front panel, the voltage can be converted to a 16-bit digital value representing that voltage. The conversion can be initiated either by the ADM's internal clock, an external triggering wire connected to the digital input barrier strip, or immediately upon the reception of a command from the Professional computer to convert a voltage.

The ADM's analog channel connectors are differential input connectors. A differential input connector takes both sides of a data signal (plus and minus) as input and subtracts the minus from the plus signal. The resultant signal is used as the data voltage. This method provides some noise reduction since it eliminates common-mode voltages present on both signals.

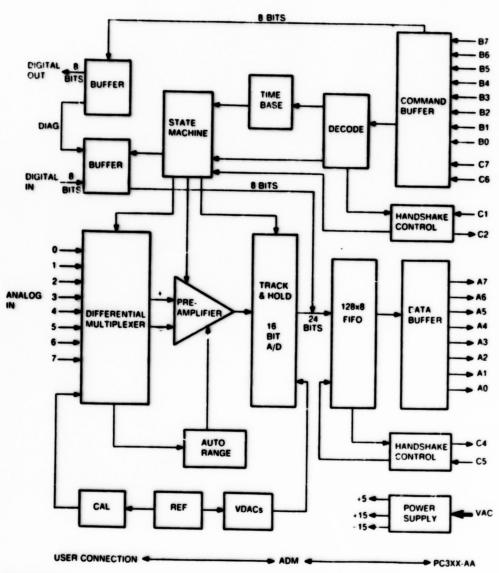


Figure 8-1 ADM Block Diagram

All the internal sequencing of the ADM is controlled by a state machine that receives commands through port B of the RTI. The state machine sets up the ADM to collect analog data, according to the commands received. When data acquisition and conversion starts, the state machine controls all the sequencing of events from the multiplexer channel s vitching to the loading of the first-in first-out (FIFO) data buffer.

When the command to acquire data is received (in some modes, the set-up procedure described below does not occur until the trigger event happens; in other modes, the set-up occurs as soon as the ADM receives the configuration command), the ADM's input multiplexer feeds the voltage from the channel to be sampled into the conversion circuitry. Only one channel can be sampled at a time. After the multiplexer has routed the sample voltage to the conversion circuitry, the state machine turns on the autoranging circuitry.

The autoranging circuitry quickly samples the analog voltage with a low resolution flash A/D converter. This allows the circuitry to determine the relative magnitude of the signal and select a gain for the preamplifier which will make optimal use of the ADM's resolution. This gain is then fed into the preamplifier, and the voltage is amplified according to this gain.

The autoranging circuitry can be disabled by your application. In this instance, your program loads the preamplifier with one of the four possible gains. Gains of 1 (unity), 4, 16, or 64 are possible.

The ADM's track-and-hold circuitry acquires and holds the amplified output of the preamplifier. This circuit continually "follows" any fluctuations in the preamplifier's output, regardless of whether the ADM is currently converting a signal or not.

After the track-and-hold circuit has acquired the signal, the front-end nodes of the analog multiplexer channel are switched internally to a ground reference; this ground reference is then fed through the preamplifier using the same gain as for the data voltage. To eliminate any offset in the data signal attributable to the preamplifier, this reference is subtracted from the voltage in the hold circuit. This process is called auto-zeroing. After a short time delay during which the preamplifier is allowed to settle, the resultant signal is converted.

The state machine now feeds the held voltage into the A/D (analog-to-digital) conversion circuitry. The hold circuit freezes the voltage so that the A/D converter (where the signal is fed after the preamplifier) will see a constant voltage throughout the conversion process.

The amplified signal is held constant while the A/D converter determines a digitized data value. The A/D converter used in the ADM is a 16-bit successive approximation converter, providing very high resolution. At high gain (64), this converter can detect a voltage increment of as little as 2.4 microvolts. The Analog Data Module Owner's Manual contains a detailed discussion of resolution.

The ADM uses the successive approximation technique for converting a voltage input to a digital value. This method of conversion uses a series of approximations, each with twice the resolution of the previous approximation. Starting with the most significant bit, each approximation sets or clears one bit in the digital value. The approximation occurs 16 times, once for each bit in the successive approximation register (SAR). The following algorithm illustrates the successive approximation method.

- Set the most significant remaining bit of the SAR to 1; with this
  output, generate a comparison level in a precision digital-to-analog
  converter (DAC).
- Compare this level with the held equivalent of the sampled signal.
- If the comparison level is less than the held sample, leave this bit set.
- If the comparison voltage is greater than the held sample, set the present bit to 0.
- 5. Prepare to operate on the next significant bit.
- Go to step 1. When all 16 bits have been compared, the resultant value represents the analog voltage.

This algorithm is illustrated in Figure 8-2.

When a digitized value is complete, the state machine loads the 16-bit value into the 8-bit FIFO buffer, with the high data byte being entered first. The state machine also reads the status register (which constantly records the channel used, gain used, and error conditions) and loads the contents of the status register into the FIFO buffer after the two data bytes.

Handshaking signals from port A of the RTI are automatically handled by the FIFO/output buffer circuitry, and data values are thus transferred to the host.

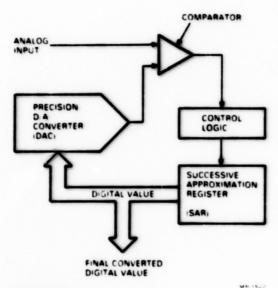
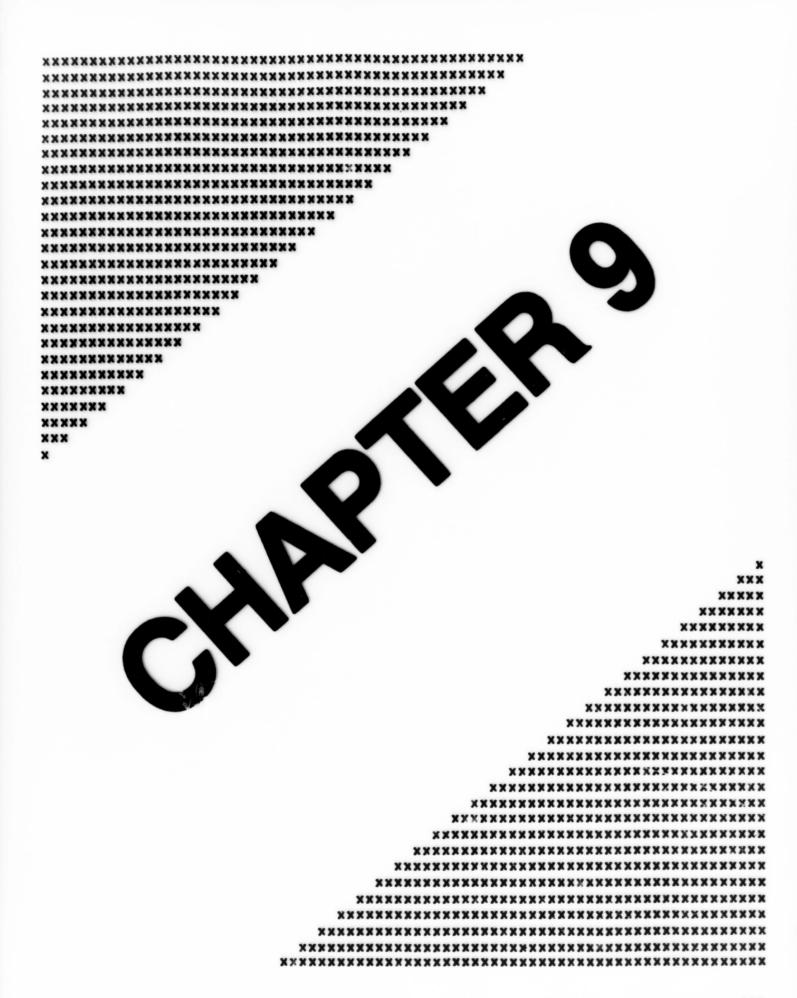


Figure 8-2
Successive Approximation Block Diagram

The above description describes the sequence of one conversion. When multiple samples are being taken on a single channel, the state machine waits for either the next "tick" of the ADM's programmable real-time clock, or the recognition of an external triggering line on the digital input barrier strip (depending on how the ADM has been configured). When the proper type of triggering event has occurred, the state machine begins the conversion process all over again. The number of samples to be acquired is not specified, so the ADM continues to acquire samples in the programmed mode until it is reprogrammed, or the RESET switch is depressed, or the inhibit bit is written as part of command byte 2. The inhibit bit stops all current operation of the ADM, but does not reprogram it or clear data in the FIFO buffer. (Refer to Chapter 11 for information on the inhibit bit and command bytes.)

The ADM can also be programmed to acquire multiple samples from multiple channels; these are called sweeps. In this event, the state machine always starts the first conversion on channel 0 when the specified triggering event occurs (either clock or external trigger). When that conversion is done, the state machine immediately switches the multiplexer circuitry to the next channel (channel 1) and immediately begins the conversion process for that voltage. In this way, the ADM proceeds from one channel to the next in the fastest possible manner. In most applications involving sweeps, it is actually desirable to measure all connected voltages simultaneously. Since this is not possible, the channels are sampled as quickly as possible, one after the other. The last sampled channel in a sweep will be the farthest away (in time) from the sample on channel 0, but this time discrepancy is minimized by the ADM. Although sweeps always begin on channel 0, you can select the number of channels in the sweep.



# 9

Collecting Analog Data

### Chapter 9

## **Collecting Analog Data**

#### **OVERVIEW**

Analog data is collected by connecting the external devices to the analog input channels on the front panel of the ADM. Cables for making these connections are provided with the ADM and instructions for using the cables are provided in the Analog Data Module Owner's Manual (EK-ADMPC-OM). The analog channels accept any voltage within the range -5 volts to +5 volts. Voltages exceeding these limits may damage the ADM.

The ADM is programmed to collect data by writing four command bytes to the command byte registers within the ADM. The four command bytes select the features described in this chapter. For complete information on the command bytes, refer to Chapter 11.

#### SELECTABLE FEATURES

Before writing your application to collect analog data, it is necessary to consider the following aspects of your application:

How data acquisition is triggered (started)
Sampling speed
Signal level
Number of samples to be collected

feature	ecifics of your application will determine which of the ADM's selectable es you should use. The ADM provides flexibility in the method of analog sions that you can use. These selectable features include:
	The ability to take one or multiple samples on a single channel
	The ability to sample all channels sequentially as quickly as possible in order to sweep several inputs
	Six triggering modes for analog data
	An internal clock for timing sampling
	A preamplifier for low-level signals-four gain settings are available
	Gain may be set by your program or determined automatically by the ADM
	A first-in first-out data buffer for more efficient data collection.
	IEL SELECTION AND MULTIPLE SAMPLES
The A simple ADM channel	DM can be configured for a variety of data acquisition modes. The st of these is the acquisition of a single sample on a single channel. The can also be configured to collect one sample per channel from all eightels (or less) in sequential order. In other words, the ADM can be conditionally taking one sample on channel 0, then channel 1, then let 2, etc., up to a maximum channel number you specify; this is called a
sweep.	A single sweep is defined as the acquisition of one sample from each el. A sweep can have between two channels and eight channels.
There	are therefore three types of channel and sample selection:
	Single sample on a single channel
	Multiple samples on a single channel
	Multiple samples from multiple channels - sweep (one sample per channel in each sweep).

#### TRIGGERING MODES

Triggering is the method by which a sample is initiated. It refers to the acquisition of the first sample as well as subsequent samples. The ADM provides three types of triggers:

Command Write Triggering	Acquistion starts as soon as the software command byte is received. This could also be called immediate execution.	
External Triggering	Acquisition starts when a negative transition is detected on the digital input barrier strip. Each channel	

is associated with the corresponding digital input bit (channel 0 with digital input bit 0). This allows you to wire an external handshaking line or manual switch to the ADM for use in starting data acquisition.

Clock Acquisition of subsequent samples can be triggered by Triggering the "tick" of the ADM's internal clock. The frequency of the clock is selectable between 0.06 Hz (a period of 16.32

seconds) and 5 kHz.

The three types of triggering are combined with the three types of channel and sample selection to provide for a variety of acquisition choices, called triggering modes. The ADM provides six triggering modes for collecting analog samples. (There are two additional triggering modes for collecting digital data; see Chapter 10.) The variety of triggering modes allows you to choose the method by which sampling begins and the method by which subsequent samples are taken.

The ADM's six analog triggering modes are:

- Mode 0 A single sample is taken on a single channel as soon as the configuration command is received.
- Mode 1 Multiple samples are taken on a single channel. Each sample is taken when an external trigger occurs. A negative transition detected on the digital input bit causes acquisition of each individual sample.
- Mode 2 Multiple samples are taken on a single channel. The first sample is taken as soon as the configuration command is received. The subsequent samples are taken at a user-specified clock frequency. Each time the clock ticks, the next sample is acquired.

- Mode 3 Multiple samples are taken on a single channel. The first sample is taken when an external trigger occurs. The subsequent samples are taken at a user-specified clock frequency.
- Mode 1 Samples are taken on multiple channels, a sweep. A single sweep is defined as the acquisition of one sample from each channel. A single sweep samples each specified channel once per sweep. A sweep can have between two channels and eight channels. In mode 4, the first sample on channel 0 is taken as soon as the configuration command is received. The second sample on channel 1 is taken as soon as possible after the channel 0 sample is taken. Each sample on the next channel in the sweep is taken as quickly as possible until the highest numbered channel in the sweep is sampled. Both the highest numbered channel and the clock frequency are user specified. In mode 4, the next sweep of the specified channels occurs at the next clock tick.
- Mode 5 Samples are taken on multiple channels, a sweep. A single sweep is defined as the acquisition of one sample from each channel. A single sweep samples each specified channel once per sweep. Thus a single sweep can have between two channels and eight channels. In mode 5, the first sample on channel 0 is taken when an external trigger occurs. The second sample on channel 1 is taken as soon as possible after the channel 0 sample is taken. Each sample on the next channel in the sweep is taken as quickly as possible until the highest numbered channel in the sweep is sampled. The highest numbered channel is user specified. In mode 5, the next sweep of the specified channels occurs when an external trigger is detected on channel 0.

NOTE: There are also two triggering modes (modes 6 and 7) for the collection of digital data. See Chapter 10

NOTE: Sweeps are always defined as starting on channel 0. You can select the highest channel used in a sweep.

#### **External Triggers**

The external trigger is useful when your analog data device has a TTL handshaking line indicating valid data or you wish to construct a manual or automatic switch to initiate data acquisition. The external trigger makes use of the digital input barrier strip on the ADM's front panel. Each digital input bit consists of a Schmitt trigger which can detect both voltage transitions and voltage levels. When an external voltage is present on the digital input bit, changes in the voltage will cause the digital input bit to be set or cleared. When an external voltage is higher than 1.6 volts, the digital input bit is set to 1. When an external voltage drops below 0.8 volts, the digital input bit is set to 0.

When the ADM is configured for a mode involving external triggering, the digital input bit is monitored by the ADM's hardware. When the digital input bit changes from 1 to 0, the ADM recognizes this as an external trigger and starts data acquisition. For a second external trigger to occur, the digital input bit must be reset to 1. This means that the voltage present on the connected line must rise to at least 1.6 volts to set the bit to 1. If a digital bit is currently 0 and the voltage rises (for example, to 1.2 volts), then drops again to 0.5 volts, this will not be recognized as an external trigger.

The digital bits have Schmitt trigger inputs which fire at the voltages listed below:

Maximum voltage level to set a logic 1: 1.6 volts

Miminum voltage level to set a logic 0: 0.8 volts

Since the Schmitt triggers respond to voltage levels rather than to transitions, the digital input bit will respond to a very slowly changing voltage as well as to a rapid transition in voltage levels. Because of the levels chosen for the Schmitt trigger firings, the digital input bits will respond to TTL lines without any alteration. If your device has a handshaking line which uses TTL to indicate that valid data is now being transmitted, this line can be connected directly to the digital input barrier strip. Also, see Chapter 10.

Each digital input bit is associated with the analog channel numbered correspondingly. For example, digital input bit 0 is associated with channel 0, and digital input bit 1 is associated with channel 1. When using a single channel triggering mode, the external triggering line is connected to the digital input bit associated with the analog channel you are sampling. If you are sampling on analog channel 5, the external triggering line should be connected to digital input bit 5.

Sweeps always start on channel 0. For external triggering of sweeps, the triggering line should be connected to digital input bit 0. Triggering lines should not be connected to the other digital input bits since the ADM moves from one analog channel to the next as quickly as possible when configured for a sweep.

Triggering lines connected to the digital input barrier strip must be part of a complete circuit. The ground connections on the digital input barrier strip should be used to complete the circuit and should be connected to the ground (associated with the handshaking line you are using) on your device. Chapter 4 of the Analog Data Module Owner's Manual (EK-ADMPC-OM) contains more details on making connections to the digital input barrier strip bits.

#### Clock Triggering

The ADM's internal clock can be used to trigger the acquisition of subsequent data samples. The clock allows you to choose a frequency between 0.06 Hz and 5.0 kHz. When you choose a triggering mode that uses the clock to collect subsequent samples, the ADM can be programmed to take the first sample immediately, or when an external trigger is detected. The clock begins ticking at the selected frequency as soon as first sample is acquired. After the first data sample is converted, the next sample is acquired on the next clock tick. In this way, you can take samples at a known frequency.

The clock frequency is selected by choosing a source frequency and a divider. Both of these numbers are written to the ADM's command registers, and the combination of the two is used to produce the clock frequency. There are eight source frequency ranges. Your divider is used to indicate how far above the minimum frequency in a source frequency range that the clock is to run. Table 9-1 and the algorithm can be used to arrive at the source frequency range and divider to be specified once you have decided on the frequency range of the clock you would like to use.

Table 9-1 ADM Clock Source Frequencies

Source Frequency Number	Binary Equivalent	Maximum Frequency	Minimum Frequency
7	111	256 kHz	1 kHz
6	110	64 kHz	250 Hz
5	101	16 kHz	62 5 Hz
4	100	4 kHz	15 625 Hz
3	011	1 kHz	3 90265 Hz
2	010	250 Hz	0 976563 Hz
1	001	62 5 Hz	0 244141 Hz
0	000	15 625 Hz	0 061035 Hz

- 1. Choose the frequency appropriate for your application.
- Using the minimum frequency column of Table 9-1, find the two source frequencies between which the desired frequency lies.
- The source frequency will be the lower of these two source frequency ranges. The source frequency number (between 0 and 7) will be specified to the ADM as part of command bytes 0 and 1.
- Divide the maximum frequency in the source frequency range by the desired frequency.
- 5. Round off the quotient of this division to the nearest whole number
- 6. Subtract I from the quotient.
- Convert the number in step 6 to binary format. This is the time base divider to be supplied in command bytes 0 and 1.

NOTE: For any desired clock frequency greater than 1 kHz, use source frequency 7.

The algorithm thus gives you two binary numbers used to select the clock frequency. Because the quotient is rounded, sometimes there is a deviation from the desired clock frequency. The maximum possible deviation is always less than or equal to +/- 0.8% of the desired frequency range.

The following example illustrates the use of the clock frequency algorithm

- 1. Assume the desired frequency is 30 Hz.
- Using the minimum frequency column of the source frequency table, you can see that 30 Hz lies between the minimum frequencies of source frequencies 4 and 5.
- The lower of the two sources is 4. This number (binary 100) is written as part of command byte 1. The maximum frequency range of source range 4 is 4 kHz.
- 4. Divide 4 kHz by 30 Hz. The quotient is 133.3333.
- 5. Round off the quotient to 133.
- 6. Subtract 1 from 133 to give 132.
- 132 in binary is 10000100. This is the divider to be supplied in command bytes 0 and 1.

Thus for a clock frequency of 30 Hz, the 8-bit divider to be specified is 10000100, and the 3-bit source frequency is 100. These two numbers are specified in command bytes 0 and 1. In this particular example, the actual clock frequency used by the ADM is 30.075 Hz, which is a positive error of 0.25%. See Chapter 11 for the command byte format.

#### **Choosing a Clock Frequency**

The frequency you choose for the collection of analog data should be determined by your application. The Nyquist theorem states that the sampling frequency used should be at least twice as high as the frequency of the highest component in your signal. This is to avoid aliasing, which is the misinterpretation of a signal's frequency due to inadequate samples. Determining the frequency of the highest component of your signal is sometimes difficult.

If you can identify the fastest transient that you can expect and if you can also determine what sampling rate will adequately identify that transient, you can use that sampling speed for the entire signal. For example, if you expect that the fastest transients in your signal will be about 0.1 seconds in duration and that ten samples would adequately identify those transients, a rate of 100 samples per second (100 Hz) could be used for the entire signal. By ensuring that the fastest transients in the signal are not lost, you also ensure that the entire signal is represented.

When choosing a clock frequency, do not choose a frequency that causes erroneous triggering. An erroneous trigger is defined as a clock tick which occurs before the ADM has finished converting a sample already held. Some erroneous ticks cannot be detected. (See next paragraph.) In the case of sweeps, an erroneous clock tick is defined as one which occurs before the highest channel in the sweep has completed its conversion.

If an erroneous clock tick occurs while the ADM is digitizing a held voltage, the digitization stops, and the new voltage is held. The erroneous clock tick is not detected. In fact, the ADM will become hung because the clock will probably tick again while the newly held voltage is being converted. This problem can be avoided by choosing a clock rate within the limits listed below:

#### Maximum Recommended Clock Rates

When using a programmed gain of 1 or 4:	5.0 kHz
When using a programmed gain of 16 or 64:	2.5 kHz
When using autoranging:	2.5 kHz

#### Sweeps

In a sweep, each channel is sampled once as quickly as possible. After all channels in the sweep have been sampled once, the ADM waits until the next triggering event (clock or external) before it begins the next sweep. A sweep of channels is designed to occur as quickly as possible in order to make the successive data points in a single sweep as close to each other in time as possible. You select the number of channels in a sweep by specifying the highest numbered channel in the sweep as part of command byte 3 (see chapter 11). Generally, sweeps are useful when more than one device is connected to the ADM, and it is desired to "read"all the devices at specified time intervals.

Sweeps are always initiated on channel 0, and it is recommended that the swept channels be sequential. The ADM always samples every channel between 0 and the upper limit you specify in command byte 3. This design minimizes the time needed to switch from one channel to the next channel. To take advantage of this, you should make all channels in a sweep sequential in order to place the data samples closely together in time. In addition, you should always start sweeps on channel 0 (have a device connected to channel 0).

You should not have any channels with open inputs in a sweep. A completely open channel, one with nothing connected to it, will saturate the ADM's preamplifier. The ADM's preamplifier will rise to its highest output on an open channel. When the sweep moves to the next channel, which has valid data, the preamplifier may not adjust to a level appropriate for that data in time for the conversion. Therefore, the digital data for that sample will most likely be too high.

You can avoid this problem if there are no open channels in a sweep. If you have devices which are used in your application occasionally, connect these devices to the higher numbered analog channels. For example, if you have three devices, but one device is used only sometimes, connect that device to channel 2. Connect the other two devices to channels 0 and 1. Then, when you run a sweep without the third device, you need to change only the channel specification in command byte 3. You will not have to make changes on the ADM's front panel.

If this solution is inappropriate for your application, and you want to have a channel unused in your sweep, you can build a NULL connector for the analog channel. For example, if you have five devices and you sometimes want a sweep of channels 0 through 4 and sometimes of channels 0, 1, 2, 3, and 5, you can build a NULL connector for channel 4 when it is not being used.

A NULL connector is a standard analog connector (analog connectors and cables are supplied with the ADM) having the plus pin shorted to the return pin. No other connections need to be made. When you wish to run a sweep without channel 4, simply disconnect the device from channel 4 and connect the NULL connector in its place. The Analog Data Module Owner's Manual (EK-ADMPC-OM) contains additional details on how to construct analog cables, connectors, and NULL connectors.

#### **ENDING DATA ACQUISITION**

Once the ADM begins sampling data, the ADM continues to sample and load the FIFO buffer until one of the following occurs:

The inhibit bit is written for a minimum of 100 microseconds. (See Command Byte 3, Chapter 11.)
The FIFO buffer fills and a triggering event(s) occurs. The ADM misses a data sample(s) in this case, but resumes conversion when the FIFO buffer clears.
The RESET switch is depressed.
Command byte 3 is written with a new triggering mode specified. (See Chapter 11.)

#### PREAMPLIFIER AND AUTORANGING

Some signal levels are so low that only a portion of the ADM's full scale range is being used. The ADM is capable of converting a signal in the range of -5 to +5 volts. The 16-bit A/D converter that the ADM uses, provides a resolution of one part in 65,536 or 153 microvolts over the full scale range. Each LSB in the converted value represents a unit of 153 microvolts.

However, if the signal magnitude is very low and varies, for example, between 0 and +0.1 volts, a resolution of 153 microvolts is possibly a significant percentage of the signal. In this case, the ADM's preamplifier can be used to increase the resolution of converted values.

The ADM's preamplifier amplifies the analog signal before it is converted. A low-level signal can therefore be amplified to "look like" a higher level signal. The value of each LSB in the converted value thus decreases. At high gain (64), the ADM can detect a change of as little as 2.4 microvolts.

The preamplifier has four possible gain settings. The available gains are 1 (or unity gain), 4, 16, or 64 times the magnitude of the incoming signal.

The preamplifier can be loaded either with a gain you select or with a gain selected by the autoranging circuitry of the ADM. The autoranging circuitry samples the input voltage prior to conversion and determines a gain that makes use of the largest portion of the ADM's range. The autoranging circuitry does this for every sample taken and can therefore adjust to a changing signal. A voltage of 0.05 volts might be given a gain of 64. The A/D converter then sees a signal of about 3.2 volts, and the least significant bits of the converted value will provide much greater resolution than if the signal had not been amplified. If the voltage then rises, for example, to 0.15 volts, the autoranging circuitry would select a gain of 16 and the converter would see a signal of 2.4 volts.

Autoranging is enabled by writing the autoranging enable bit in command byte 2. (See Chapter 11.)

Autoranging can be disabled, and the preamplifier can be loaded with a gain you select. In this case, the preamplifier applies the selected gain to all input voltages, regardless of level, until reprogrammed. If the input voltage times the gain rises above 5.0 volts, an erroneous conversion will result. For example, if you expect a signal magnitude to vary between 0 and 1.0 volts, you might load the preamplifier with a gain of 4. Should the signal rise to greater than 1.25 volts, the converter will be fed a voltage greater than 5.0 volts and be unable to convert the value accurately. In some applications, of course, the signal magnitude can be predicted accurately, and this situation will not apply.

The convenience of autoranging is offset by the additional time the autoranging circuit adds to the conversion time. The total conversion time when using autoranging is generally 400 microseconds. This compares to a conversion time of 200 microseconds for conversions using a programmed gain (autoranging disabled) of 1 or 4. Conversion time when using a programmed gain of 16 or 64 (autoranging disabled) is 400 microseconds, equivalent to the autoranging delay. Thus if you know, as in the example above, that your signal will vary only between 0 and 1.0 volts, autoranging (as opposed to loading the preamplifier with a gain of 4) is not advised, since it doubles the conversion time.

To select a constant programmed gain for the preamplifier, disable the autoranging circuitry by clearing the autoranging bit in command byte 2 and specify the desired gain using the two gain bits in command byte 2. (See Chapter 11 for the command byte format.)

#### EFFECTIVE APERTURE DELAY—SOME CAUTIONS

Whatever gain you decide is appropriate for your application, including unity gain, you should consider the possible effect of the effective aperture delay on the accuracy of your data. In an ideal world, the conversion of the analog voltage would occur at precisely the same time as the occurrence of the trigger that initiates data acquisition, whether the trigger is the clock, an external trigger, or immediate execution. However, this simultaneous conversion is not possible, and a slight delay always occurs between the time of a trigger and the conversion of a voltage. The delay is due to multiplexer settling times, preamplifier settling times, and aperture delay times in the track-and-hold circuit. The sum total of all these times is called the effective aperture delay. Generally the delay is negligible, but in some cases it may be significant.

If your application requires that the collected data be as closely synchronized with the triggering event as possible, the effective aperture delay may be a problem. Your digital data will actually be somewhat later in time than the triggering event. To alleviate this problem, the ADM (when collecting multiple samples on a single channel) settles out the preamplifier output before the triggering event occurs. When the triggering event occurs, the hold circuit immediately freezes the preamplifier output and conversion begins. The effective aperture delay is diminished. This mode of operation, called presettling, is used when the ADM is programmed for multiple samples on a single channel.

When using autoranging in conjunction with mode 1 or mode 3 (multiple samples on a single channel using external triggering), you should consider the fact that, with presettling, the autoranging circuit loads the preamplifier before the trigger occurs. The autoranging circuit samples the voltage present on the inalog channel as soon as the configuration command (command byte 3) is received. This voltage may or may not be anything like the voltage that will be present when the external trigger occurs. If the channel voltage is significantly different by the time the external trigger occurs, the loaded gain may be completely inappropriate and the digitized value be over- or underranged. In fact, if the voltage is greatly different, the first few samples may be inappropriately amplified since it takes a while for the amplifier to settle if it is saturated. In cases like this, you may want to choose a software selected gain and disable autoranging. Or, if appropriate to your application, you may want to discard the first few samples taken.

In modes where the samples are taken by the clock (mode 2), the autoranging samples the voltage present at the analog channel and loads the appropriate gain into the preamplifier. The first clock tick then occurs immediately. Provided an appropriate clock frequency has been chosen, the above problem is avoided.

In mode 5 (sweeps initiated by external triggers), the autoranging circuitry does not sample the voltage present on channel 0 for the first sample until the external trigger is received. Presettling cannot be used in sweeps since the autorange gain selected would apply to the previous channel and not to the sample actually collected. Therefore, the problem described above is avoided. To alleviate effective aperture delay problems in sweeps, you can select a low gain without autoranging. This shortens the delay considerably.

Appendix B shows how the sequence of each event, channel acquisition, autoranging, auto-zeroing, and conversion occur in each of the different modes.

#### THE FIFO BUFFER

After an analog voltage is converted, the digital value is placed in the first-in first-out (FIFO) data buffer along with a status byte which records the specifics of the conversion. The FIFO buffer allows data values to accumulate in the ADM if the host is unable to retrieve them immediately.

The FIFO buffer contains 128 bytes of memory. In addition, there is an 8-bit output data buffer which is used to interface with the RTI's parallel port lines (port A). When the RTI asserts Input Buffer Full (IBF) true, it signals that it received the previous data byte and is ready to receive the next data byte. When the ADM detects IBF true, it loads the data byte at the top of the FIFO buffer into the output data buffer. This means that at any one time, 129 bytes are available for stored converted values (128 in the FIFO buffer plus 1 in the data output buffer). Since each converted analog value uses three bytes (two for the digitized value and one for the status byte), the ADM is capable of buffering 43 analog values for transmission to the host (129/3 = 43).

In order for data to flow to the host without being lost, it is necessary for the host to continually empty the FIFO buffer as the ADM is filling it. If the host cannot retrieve data quickly enough, the ADM fills the FIFO buffer and will not digitize any new values until the FIFO buffer can accept the next byte. If a triggering event occurs before at least one data byte is emptied, the data sample is not acquired and is lost. A status byte sent with the data bytes indicates that the FIFO buffer is full or that a triggering error has occurred. (See Status Byte, Error Conditions, in Chapter 11.)

While this approach requires some thoughtful analysis when collecting data at high rates, it does allow you to take "bursts" of data at the ADM's maximum A/D rate, and collect up to 43 valid samples at your program's convenience. For example, the ADM can convert samples at a rate of up to 5 kHz. If your application needs to sample short events at the recognition of a trigger, for example, and 43 samples will represent the event, you can fill the ADM's FIFO buffer as quickly as possible when the trigger occurs. This data will remain in the FIFO buffer until retrieved; your application, therefore, does not need to retrieve it immediately. The last status byte will indicate a triggering error, but if 43 samples are enough for your application, you can ignore the last data sample.

The maximum rate at which your application can retrieve data from the ADM will be specific to your application. The ADM can convert analog data values at a peak maximum rate of 5 kHz. A continuous sampling and transmission rate will be less than this, but it depends upon the specifics of your application and the amount of software processing you do.

The FIFO buffer is cleared only under two conditions: when a new triggering mode and channel specification command byte is written (command byte 3); or the RESET button is depressed on the ADM's front panel. Writing the inhibit bit (in command byte 2) does not clear the FIFO buffer. (See Chapter 11.)



## 10

Using the Digital Input/Output Barrier Strips

#### Chapter 10

#### Using the Digital Input/Output Barrier Strips

#### INTRODUCTION

The ADM's front panel contains an 8-bit parallel digital input barrier strip and a 8-bit parallel digital output barrier strip. The digital input barrier strip is primarily intended to be used as a means for external triggering. However, two triggering modes are provided in the ADM for the reception of digital data. The digital output barrier strip allows you to send 8-bit digital data out to remote devices and is primarily intended to allow application-specific commands to be sent to remote data devices.

#### MAKING CONNECTIONS TO THE BARRIER STRIPS

Each bit in the barrier strip is a pushtab connector. To connect a wire from an external device, use a small screwdriver, ball-point pen, or your fingers to press in the red slot on the barrier strip. Insert the uninsulated end of the wire. Release the red tab to hold the wire in place. Two ground pushtab connections are also provided for each digital strip.

#### THE DIGITAL INPUT BARRIER STRIP

The digital input barrier strip may be used for external triggering, for status bit purposes, or for receiving digital data.

#### Digital Input Barrier Strip Logic

A voltage of 0.8 volts or less sets a logical 0 in the digital input bit's logic. A minimum voltage of 1.6 volts sets a logical 1. The digital input bits are Schmitt triggers which respond to voltage levels rather than to transitions. A slowly changing triggering signal will cause the Schmitt trigger to fire when the voltage crosses the threshold of 0.8 volts even if that negative transition is made very slowly. When the Schmitt trigger fires, the digital input bit's internal logic is set to 0.

The digital input bit must be reset to 1 before the Schmitt triggers can fire again. This means that any external line connected to the bit must rise to a voltage of at least 1.6 volts or higher to set the digital input bit to logical 1 before a negative transition can again set a logical 0.

Any device that uses Transistor-Transistor Logic (TTL) will correctly fire the digital input bit Schmitt triggers. A TTL High signal is defined (for output) as a signal between 2.4 volts and 5.0 volts. A TTL Low signal is defined as a signal between 0 and 0.5 volts. Thus TTL high will cause a 1 to be set in the digital input bit and TTL low will cause a 0 to be set.

#### Using the Digital Input Barrier Strip for External Triggering

Each digital input connector is associated with the analog channel numbered correspondingly for external triggering. When you have programmed the ADM for single channel acquisition started by an external trigger (either single or multiple samples—mode 1 or mode 3), the digital input bit associated with that channel is monitored for detection of a trigger signal. A trigger signal is a voltage carried on a wire connected to the digital input bit. When that voltage drops from a level above 1.6 volts to a level below 0.8 volts, the digital input bit is set to a logical value of 0; this is defined as an external trigger signal. The ADM in this case begins data acquistion.

The external trigger is wired to the digital input bit associated with the selected analog channel. If you are collecting samples on channel 4, connect the external trigger wire to digital input bit 4.

When you program the ADM for a sweep, the external triggering line is connected to digital input bit 0. Sweeps always start on channel 0, and an external trigger detection on digital input bit 0 starts the entire sweep. A triggering connection to the digital input bits associated with the other channels used in the sweep is not necessary.

The external triggering capability allows data producing devices themselves to trigger data collection. Some devices make use of a handshaking line called "Data Valid" or "Data Ready," or some other name which indicates that valid data is now being produced. If your device has such a line and the line conforms to TTL levels, that line can be used directly for an external trigger. The circuit should be completed by connecting a wire from the digital input barrier strip ground bits to the ground on your device.

#### **Using Switch Closures for External Triggering**

The digital input bits can be wired so that external triggering is performed by switch closures. Each digital input bit is set to logical 1 when there is no voltage present. The digital input bits are internally wired to input voltage to themselves when no other connections are made to them. Therefore, if a normally open switch is wired between a digital input bit and ground, the digital bit will "sense"logical 1 while the switch is open. Closing the switch shorts the digital input bit to ground; the voltage drops to 0, setting a logical 0 on the digital input bit. This serves as an external trigger.

If the switch closure method is used, a capacitor should be used to bypass the switch to ground. This will debounce the switch and prevent false second triggering. Figure 10-1 shows how this capacitor should be wired. There are two ground inputs on the digital input barrier strip. The size of the capacitor should be between 9.1 microfarads and 10 microfarads.

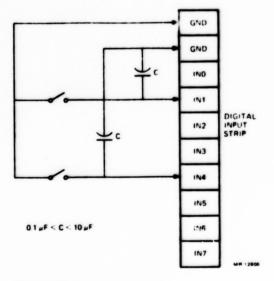


Figure 10-1 Example of Switch Debouncing

The Analog Data Module Owner's Manual (EK-ADMPC-OM) provides further information on using the digital input barrier strip for external triggering. Also, refer to Chapter 9 for further information on modes that use external triggering.

#### Using the Digital Input Barrier Strip for Status

Since the status byte returns the status of the digital input bit associated with the sampled analog channel regardless of whether the ADM is programmed for external triggering, the validity bit can be used as a status bit. If you wish to use a device's "Data Valid"line for status purposes instead of external triggering, this can be done easily. The handshaking line is connected to the digital input barrier strip in the same manner as described above for external triggering lines. In this case, however, the ADM is programmed for data acquisition based upon a clock, or repeated executions of single sample commands. The validity bit can then be examined for every data byte that is returned. Data bytes with validity bits indicating invalid data can be discarded.

Since the status byte returns the digital input bit for all data acquisition modes, the validity bit can be used in conjuction with external triggering for sweeps. For example, if you want to sweep channels 0, 1, and 2 using external triggering to start the sweep, you can still connect a status line to digital input bits 1 and 2. These bits can then be used as validity bits since they play no role in the external triggering of the sweep.

#### **Collecting Digital Data**

The ADM provides two triggering modes for collecting 8-bit digital data through the digital input barrier strip. Since the ADM can be configured for only one triggering mode at a time, using the ADM for digital data collection precludes simultaneous collection of analog data. However, at times this might be desirable. Digital data can be collected at rates up to 128 kHz. As with analog data at 5 kHz, this is a burst mode of operation. A continuous sampling and transmission rate would be less than this and would depend on the specifics of your application.

Digital data is read from the FIFO buffer in the same manner as analog data, except that each digital data sample occupies only one byte. The FIFO buffer, therefore, can hold up to 129 digital data samples. The digital data logic matches that of the incoming data lines. A 1 in the digital byte indicates that that data line was in the high state when the sample was taken. Digital data in port A corresponds directly to the digital input bits. The MSB in port A (PA7) reflects the data present on digital input bit 7, and the LSB (PA0) reflects the data present on digital input bit 0.

NOTE: No status byte is provided for digital data. There is therefore no error indication if the FIFO buffer becomes full.

Triggering mode 6 provides for collection of a single byte of digital data from the digital input barrier strip. The data is read immediately when the command byte is written to the ADM.

Triggering mode 7 provides for collection of multiple bytes of digital data from the digital input barrier strip. The first byte is collected immediately when the command byte is written to the ADM. Subsequent command bytes are read at each "tick" of the ADM's clock.

No handshaking lines are provided for triggering modes 6 and 7. Synchronization of the ADM's reception of data with the remote device's transmission is application dependent.

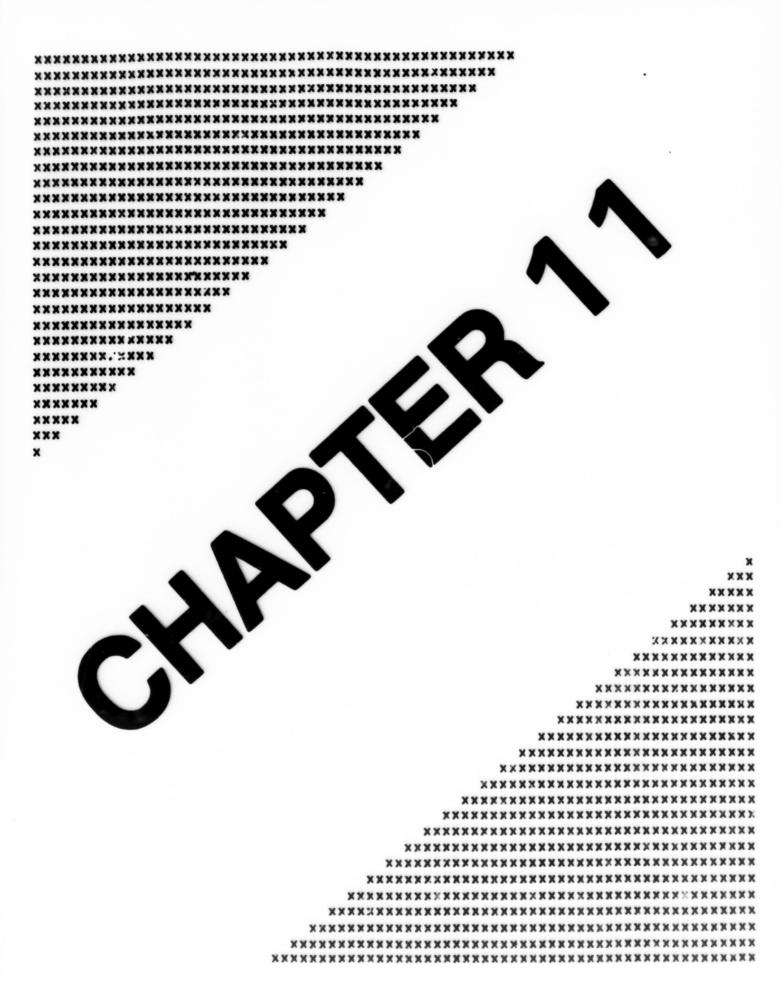
Selecting triggering mode 6 or mode 7 is done in command byte 3, in the same manner that analog triggering modes are selected. (See Chapter 11 for the command byte formats.)

#### USING THE DIGITAL OUTPUT BARRIER STRIP

The ADM can be configured for digital output by setting PC6 to 1 and PC7 to 0. This disables the normal command byte decoding logic, and the ADM passes any data received from the RTI port B lines directly to the digital output barrier strip. In this way, digital data can be sent from the ADM.

The digital output barrier strip is provided to allow control over external devices. Many data devices have digital input ports for command information, such as controlling calibration, initiating or stopping analog dtat transmissions, or other functions. Refer to the documentation provided with your device for information on how to control it.

The digital output strip uses Transistor-Transistor Logic for data transmission. A digital bit of 1 produces a high output (voltage greater than 2.4 volts) and a digital bit of 0 produces a low output (voltage less than 0.5 volts). Data written to the digital output barrier strip remains latched until the next data byte is received.



## 

Programming the ADM

#### Chapter 11

#### Programming the ADM

#### INTERFACING WITH THE REAL-TIME INTERFACE MODULE

The ADM is connected to the RTI's internal cable by the 62-pin cable supplied with the ADM. This cable carries all the RTI's input/output signals. On the ADM, however, the IEEE-488 bus connector and the two SLU connectors are pass-through connectors only. The ADM does not process signals on these interfaces in any way. The ADM is controlled through the RTI's parallel I/O port. When you use the ADM with the RTI, the parallel port is not available for independent programming.

To control the ADM, you must first configure the RTI's parallel port. The proper configuration for ADM control is port A, mode 1, input, port B, mode 1, output, and both port C upper and port C lower configured for output. This configuration is accomplished by writing a control word of 264 (8), (10110100), to the parallel port command register. See Chapter 5.

Port A is used to receive data from the ADM. The ADM's FIFO output buffer is connected to the port A lines. The handshaking lines in port C are controlled automatically by the ADM. When there is a data byte in the FIFO output buffer, the ADM asserts the STB signal true. The RTI asserts the IBF signal true (high) indicating that it has received the data byte. When your application reads the RTI port A register, IBF goes low, signaling the ADM to transmit the next data byte from the FIFO buffer. These control lines are handled automatically by the ADM.

Port B is used to transmit data to the ADM. The information received on these lines by the ADM is interpreted either as a configuration command (for analog or digital data collection), as digital output data, or as calibration commands for either offset calibration or gain calibration. How the data on the port B lines is interpreted depends upon the current combined states of the PC6 and PC7 lines. See Table 11-1. Handshaking control lines between the RTI and the ADM are handled automatically for port B transmissions.

The handshaking timing diagrams for the RTI/ADM data transfers are shown in Figure 11-1.

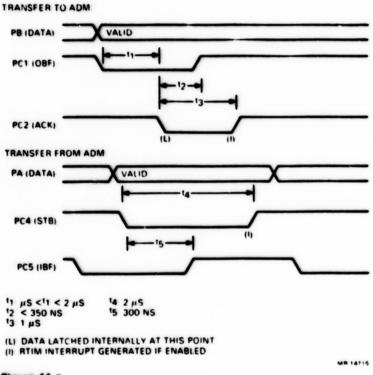


Figure 11-1 RTI/ADM Handshaking Timing Diagram

Most of the bits in port C are used for handshaking lines for port A and port B data transmissions. However, some bits are left over and are used to control how the ADM interprets the port B data sent to it. Bits PC6 and PC7 are used to control the operational mode of the ADM. Table 11-1 shows how the combined states of PC6 and PC7 indicate the operational mode of the ADM.

Table 11-1 PC7 and PC6 Meanings

PC7	PC6	Meaning
0	0	Analog and digital command mode. Port B data is interpreted as command bytes.
0	1	Digital output mode. Port B data is sent directly to the digital output barrier strip.
1	0	Offset calibration mode. Port B data is loaded into the offset vernier DAC. See Chapter 12.
1	1	Gain calibration mode. Port B data is loaded into the gain vernier DAC. See Chapter 12.

Bits PC6 and PC7 are controlled by port C bit set/reset commands written to the RTI's parallel port command register. Chapter 5 contains complete information on the parallel port command register. Table 11-2 summarizes the commands that need to be written to this command register to set or clear PC6 and PC7.

Table 11-2 Commands to Set or Reset PC6 and PC7

Desired Result	Octal Command	Sinary Command (first eight bits all zeroes)
To set PC6	035 (8)	00011101
To set PC7	037 (8)	00011111
To clear PC6	034 (8)	00011100
To clear PC7	036 (8)	00011110

PC6 and PC7 remain latched when a port C bit set/reset command is written. Therefore, the ADM remains in the chosen configuration until PC6 and PC7 are specifically rewritten.

The following paragraphs describe each operational mode.

#### ANALOG AND DIGITAL DATA ACQUISITION MODE

When PC6 and PC7 are both set to 0, the ADM is configured for analog and digital data acquisition. In this mode, all bytes written to the ADM are considered to be command bytes. The ADM decodes these command bytes and configures itself for either analog or digital data acquisition. Only four command bytes are necessary to configure the ADM for data acquisition. In some instances, some of the command bytes do not need to be written.

The four command bytes are described in Table 11-3.

Table 11-3 Command Bytes Overview

Command Byto	Bito	Specifies
0	5-4	Time base divider bits 7 and 6
	3	Digital barrier strip wraparound -diagnostic only
	2-0	Clock source frequency bits
1	5-0	Time base divider bits 5-0
2	5	inhibit bit
	4	Calibration bit
	3	Digital output enable bit
	2-1	Preamplifier gain bit
	0	Autoranging enable bit
3	5-3	Channel select bits
	20	Triggering mode select bits

Tables 11-4 through 11-7 describe the setting of each bit in the command bytes.

Table 11-4 Command Byte 0

Bit Number	Value	Significance		
7-6	00	Signifies that this is	command byte 0	
5		Time base divider b	17	
4		Time hase divider b	16	
3	1	For diagnostic purposes only enables digital barrier strip wraparound		
	0	Normal digital barrie	r strip operation	
2-0	•••	These three bits select the real time clock source frequency as shown		
	210	Source Frequency	Maximum Frequency	
	111	7	256 kHz	
	110	6	64 kHz	
	101	5	16 kHz	
	100	4	4 kHz	
	011	3	1 kHz	
	010	2	250 Hz	
	001	1	62 5 Hz	
	000	0	15 625 Hz	

Table 11-5 Command Byte 1

Bit Number	Value	Significance
7-6	01	Signifies that this is command byte 1
5	•	Time base divider bit 5
4		Time base divider bit 4
3		Time base divider bit 3
2		Time base divider bit 2
1	•	Time base divider bit 1
0	•	Time base divider bit 0

Command bytes 0 and 1 are used to select a clock frequency. By providing a time base divider and a source frequency, the real-time clock can be programmed. See Chapter 9 for information on how to determine the time base divider and source frequency.

Table 11-6 Command Byte 2

Bit Number	Value	Significance
7-6	10	Signifies that this is command byte 2
5	1	Inhibits triggering. Setting this bit (for at least 100 microseconds) causes all data acquisition to stop. The FIFO buffer is not cleared
	0	Normal triggering operation
4	1	Disables all analog input from the multiplexer and feed calibration inputs to channels 0 through 3
	0	Normal analog input from the analog channels
3	1	Disables digital output barrier strip. The bits are placed in the high impedance three state mode.
	0	Enables normal digital output through the digital output barrier strip
2-1	•	Specifies the gain to be used by the preamplifier as shown. These bits are overridden if the autoranging enable bit is set to 1.
	21	Gain
	00 01 10	1 (unity) 4 16 64
0	1	Enables autoranging. The autoranging circuitry determines the optimal gain for each signal. Overrides any gain specified by bits 2 and 1.
	0	Disables autoranging. Bits 2 and 1 specify the gain to be used by the preamplifier.

The inhibit triggering bit (bit 5) is used to stop any data acquisation currently occurring when the command byte is written. When the inhibit bit is written set for at least 100 microseconds, the ADM stops whatever it is doing and enters a hold mode. This hold mode can be released only by writing command byte 2 again with the inhibit bit clear. Data acquisition can then be restarted by writing command byte 3 again with either equivalent or new triggering information. Because the inhibit bit must remain set for at least 100 microseconds, you will probably not be able to write the new command byte information immediately.

When the inhibit bit is set, the data in the FIFO buffer is not cleared if the inhibit bit is set while a conversion is occurring, that conversion is alleved to finish, but the status byte records a false triggering error.

The ADM can be restarted only by writing command byte 3. However, writing command byte 3 automatically clears the FIFO buffer. Therefore, if you want to obtain the data in the FIFO buffer before restarting the ADM, you can repeatedly read port A of the RTL Each read of port A will toggle the handshaking lines between the ADM and the RTL and transfer the next data byte from the FIFO buffer.

The calibration bit (bit 4) is used during calibration. When bit 4 is set, the normal inputs to the multiplexer channels are disabled and the following inputs are fed to channels 0 through 4:

' Channel 0 Analog ground reference

Channel 1 Negative full-scale reference

Channel 2 Positive full-scale reference

Channel 3 Bipolar triangular ramp

Only the inputs on channels 0 and 1 are used for calibration. The channel 2 and 3 inputs are used during diagnostic testing.

Refer to Chapter 12 for information on calibrating the ADM.

When cleared, the digital output enable bit (bit 3) places the bits on the digital output barrier strip in the high-impedance three-state mode which is neither a 1 nor a 0. The bits remain in this state until command byte 2 is rewritten with bit 3 clear, the ADM enters digital output mode, and the first data byte is written to the output strip. See Digital Output Mode below.

The programmable gain bits (bits 2 and 1) select a gain to be used if autoranging is to be disabled.

The autoranging bit (bit 0) enables or disables autoranging

Choosing autoranging or a gain for the preamplifier is discussed more fully in Chapter 9.

Table 11-7 Command Byte 3

Bit Number	Value	Significanc					
7-6	11	Signifies that this is command byte 3					
5-3	•••	Analog channel selection. The three bits specify the channel to be used for single channel sampling, or the highest channel in a sweep					
	543	Channel S	elected				
	111	Channel 7					
	110	Channel 6					
	101	Channel 5					
	100	Channel 4					
	011	Channel 3					
	010	Channel 2					
	001	Channel 1					
	000	Channel 0					
2-0		Triggering Mode Selection. These three bits specify the triggering mode to as shown below.					
	210	Trigger Mode	Description				
	111	7	Multiple digital values input through the digital input barrier strip first sample initiated immediately subsequent samples initiated by clock				
	110	6	A single digital value input through the digital input barrier strip				
	101	5	Analog channel sweeps each sweep initiated by an external trigger				
	100	4	Analog channel sweeps first sweep starts immediately, subsequent sweeps initiated by clock				
	011	3 Single analog channel multiple samples first sample initiated by external trigger, subsequent samples initiated by clock					
	010	2	Single analog channel multiple samples first sample starts immediately subsequent samples initiated by clock				
	001	1	Single analog channel multiple samples each sample (first and subsequent) initiated by external trigger				
	000	0	Single analog channel, single sample, initiated immediately when command is written				

Writing command byte 3 clears any data in the FIFO buffer

The meaning of the channel specification bits depends upon the triggering mode selected If you select a single channel acquisition mode, the channel specification bits identify the channel to be used. If you select a sweep triggering mode, the channel specification bits identify the highest numbered channel in the sweep. Sweeps always start on channel 0 and sweep through every channel until the specified highest number channel is reached.

There should be no unused analog channels in a sweep. See Chapter 9. If you select a triggering mode for digital data acquisition, the channel specification bits are ignored.

The triggering mode bits specify the triggering mode to be used for data acquisition. If you select an analog triggering mode which starts immediately tmodes 0, 2, and 4), the ADM begins converting the voltage on the specified channel immediately. If you select a triggering mode which starts on an external trigger tmodes 1, 3, and 5), the ADM is fully configured when command byte 3 is received and will wait indefinitely until an external trigger is received. Data acquisition of the first sample then begins

In all triggering modes involving the clock, the clock starts when the first data sample is acquired. This ensures that the clock period between all samples is equal. For triggering modes that start with an external trigger, the clock does not start until the external trigger is received and the first sample is acquired.

Chapter 9 contains more detailed information on the triggering modes.

#### Writing the Command Bytes and Collecting Data

To write the command bytes to the ADM, first write to the RTI parallel port control register the port C bit set reset commands which clear both PC6 and PC7; these are 034 (8) and 036 (8). This places the ADM in analog operational mode. Then write the four command bytes, starting with command byte 0, to the RTI port B register. The ADM automatically controls the handshaking lines for these transfers. When you have written command byte 3, either data acquisition starts immediately, or the ADM waits for an external trigger.

Then it is necessary only to read the data from the RTI's port A register. If you have enabled interrupts for the RTI's parallel port, the data transfers can be entirely interrupt driven.

Command byte 3 is the only command byte that actually starts data acquisitions. The other command bytes configure the ADM but do not initiate any action by the ADM. You can therefore write command bytes 0 through 2 at any time. The specifications they select remain in place until the command byte is rewritten.

Therefore, if you find it necessary to restart data acquisition using the same parameters (clock speed, gain, etc.), it is not necessary to rewrite command bytes 0 through 2; rewriting command byte 3 is sufficient.

Once data acquisition starts, the ADM continues running until specifically turned off by the inhibit bit. This allows the ADM to continue converting analog data while also outputting digital data through the digital output barrier strip.

#### **ANALOG DATA FORMAT**

Analog data is stored as three bytes. When a conversion is complete, the converter holds a 16-bit digital value representing the analog voltage. This value is written into the FIFO buffer along with a status byte that records information such as the error status, analog channel sampled, validity bit, and the gain applied. These are explained below.

The high byte of the 16-bit data value is written into the FIFO buffer first. The low byte is written next, followed by the status byte. This is illustrated below in Figure 11-2.

PA7	PA6	PA5	PA4	PA3	PA2	PA1	PAO	RTI PORT A
MSB	D14	D13	D12	D11	D10	D9	D8	HIGH DATA BYTE
D7	D6	D5	D4	D3	D2	D1	D0	LOW DATA BYTE
S7	S6	S5	S4	S3	S2	S1	S0	STATUS BYTE

Figure 11-2
Data Flow from the ADM to the RTI

The ADM data output buffer is connected to port A of the RTI. The high-order bit of the data output buffer is connected to bit PA7, and the low-order bit of the output buffer is connected to bit PA0.

#### Coding Format

Table 11-8 illustrates the coding format for converted analog values.

The coding format is 2's complement.

Table 11-8 Coding Format

Differential Input Voltage	Word Output	
Positive Full Scale (or overrange)	077777(8)	
Positive Full Scale - 1 LSB	077776(8)	
Zero volts + 1 LSB	000001(8)	
Zero volts	00000(8)	
Zero volts - 1 LSB	177777(8)	
Negative Full Scale + 1 LSB	100001(8)	
Negative Full Scale (or underrange)	100000(8)	

Table 11-9 summarizes the bits in the status byte.

#### Status Byte

Table 11-9 summarizes the bits in the status byte.

Table 11-9 ADM Status Byte

Bit Number	Value	Significance
7	1	An error was detected (type is indicated by bit 6)
	0	No errors were detected. If bit 7 is 0, bit 6 has no meaning
6	1	A triggering event was detected during the conversion of this value (or previous value—see below)
	0	The ADM FIFO buffer was filled when this value was written (or previous value—see below)
5-3		These three bits specify the analog channel used
	543	Channel Used
	000	0
	001	i
	010	2
	011	3
	100	4
	101	5
	110	6
	111	7
2	1	The validity bit indicates the state of the associated digital input bit when the value was converted. A 1 indicates the digital bit was set to 1 (a high voltage was present).
	0	The digital input bit associated with this analog channel was set to 0 (a low voltage was present)
1-0		These two bits indicate the gain used by the preamplifier for this conversion. The gain bits do not indicate whether the gain was chosen by autoranging or by program control.
	10	Gain
	00	1
	01	4
	10	16
	11	64

NOTE: The validity bit is collected regardless of whether the digital input barrier strip is being used for external triggering or not

#### **Error Bits and Error Conditions**

Once set, the error bits in the status byte remain set until a new triggering mode is specified using command byte 3 (see Chapter 11). When bit 7 is set, bit 6 has a meaning as described below.

Bit 6, when clear, indicates that your application is not emptying the FIFO buffer at a fast enough rate. When the FIFO buffer becomes full, and a digitized value cannot be entered into the FIFO buffer, status bit 7 becomes set and bit 6 is clear. However, since the FIFO buffer is full, the status byte cannot be entered into the FIFO buffer until your application has emptied at least three bytes (one sample). The converted data value is then entered along with the status byte indicating the error condition. Your application, however, will not receive this byte until the 42 data samples previously entered are retrieved. This error condition is an indication that your application needs to be optimized to retrieve data faster from the FIFO buffer.

When set, bit 6 indicates that an erroneous triggering event was detected. This could be due to the fact that the clock frequency you have chosen is too fast, and a clock tick occurred while a voltage was being converted. However, it may also be due the fact that your application is not retrieving data fast enough from the FIFO buffer, as when bit 6 is clear. When the FIFO buffer first fills up and a data sample cannot be entered immediately into the FIFO buffer, bit 7 becomes set and bit 6 is clear. If the data sample cannot be entered into the FIFO buffer quickly enough and another triggering event occurs telock or external), bit 6 is set and will be written into the FIFO buffer as set, when there is room in the FIFO buffer. The data sample that should have been digitized by the triggering event is lost. In this instance, although bit 6 is set, the real problem is that your application is not retrieving data fast enough from the FIFO buffer.

Since the status bit indicating the FIFO buffer is full cannot be loaded into the FIFO buffer until you have emptied at least three bytes, your application will not find out immediately that some data has been lost. You will not be able to determine how much data has been lost, nor its value. In this case, you should probably consider all the data after the error condition as corrupt, though this judgment is application specific. You must determine that the data transmission between the host and ADM is fast enough to prevent lost data.

#### DIGITAL OUTPUT MODE

Writing the RTI's PC6 set to 1 and PC7 clear places the ADM in the digital output mode. In this mode, any data sent from the RTI through port B to the ADM is treated as digital output data and is latched directly into the digital output buffer. The bits in the digital output barrier strip are latched with this data and do not change until the next data byte is received from the RTI. If the ADM is taken out of the digital output mode, the digital output barrier strip remains latched according to the last data byte sent to it. You can return the digital output barrier strip to the three-state high-impedance output by using bit 3 of command byte 2.

For more information on the digital output mode, refer to Chapter 10.

#### OFFSET AND GAIN CALIBRATION

When PC6 is set to 0 and PC7 is set to 1, port B data coming from the RT1 is interpreted as calibration data for the offset vernier DAC. When PC6 and PC7 are both set to 1, port B data coming from the RT1 is interpreted as calibration data for the gain vernier DAC. Calibration is described in Chapter 12.





# 

Calibration

#### Chapter 12

#### Calibration

#### OFFSET AND GAIN CALIBRATION

The ADM provides two vernier DACs (digital-to-analog converters) to allow fine calibration of the offset and gain measurement.

The ADM should be allowed to warm up for approximately twenty minutes before calibrating it. Should the temperature of your application environment change by a gradient of more than 2°C during fifteen minutes, the ADM may need recalibration.

#### Offset Calibration

The following steps should be used to calibrate the ADM for offset.

- Place the ADM in analog operational mode by writing PC6 and PC7 clear.
- Configure the ADM for calibration by writing the following command bytes:

Command byte 0: 00000000 000 (8)

Command byte 1: 00100001 101 (8)

Command byte 2: 10010000 220 (8)

Command byte 3: 11001000 310 (8)

This configures the ADM for calibration (bit 4 in command byte 2 set) by feeding the negative full-scale reference voltage into channel 1 and the positive full-scale reference voltage into channel 2. The negative full-scale reference is used to calibrate for offset. The positive full scale reference will be used to calibrate for gain.

- The ADM begins converting the voltage present on channel 1 and transmitting the data to the RTI.
- Set PC6 to 0 and PC7 to 1 to place the RTI in the offset calibration mode.
- Your calibration routine should provide a means of averaging the returned values as they are received. The averaged value should be compared to a value of 100000 (8). Steps 6 and 7 will provide a way to adjust the returned converted values until they equal 100000 (8).
- 6. In the offset calibration mode, data bytes written to port B of the RTI are written to the ADM's offset vernier DAC. The vernier DAC powers on containing the value 200 octal (10000000). The contents of this DAC can be changed by writing a new data value to port B. A change of one LSB in the vernier DAC will change the ADM's returned converted values by one-half of an LSB. When you are using this technique, your calibration routine should write a loop which continues to decrease (or increase) the value in the offset vernier DAC until the returned values average 100000 (8).
- Subtract one LSB from the vernier DAC register. This places the offset as close to zero as possible.
- 8. The ADM is now calibrated for offset.

#### **Gain Calibration**

The following steps should be used to calibrate the ADM for gain.

- Place the ADM in analog operational mode by writing PC6 and PC7 clear.
- Configure the ADM for gain calibration by writing the following command bytes:

Command byte 3: 11010000 320 (8)

This ADM remains configured for calibration from the previous writing of command bytes 0, 1, and 2. Only command byte 3 needs to be rewritten to specify that samples are now to be taken from channel 1. The positive full-scale reference will be used to calibrate for gain.

- 3 The ADM begins converting the voltage present on channel and transmitting the data to the RTL Ideally, this should be 0777777 (8). If it is not, the gain vernier DAC needs to be adjusted.
- Set PC6 to 1 and PC7 to 1 to place the RTI in the gain calibration mode.
- 5. Your calibration routine should provide a means of averaging the returned values as they are received. The averaged value should be compared to a value of 077776 (8). Steps 6 and 7, will provide a way to adjust the returned converted values until they equal 077776 (8).
- 6. In the gain calibration mode, data bytes written to port B of the RTI are written to the ADM's gain vernier DAC. The vernier DAC powers on containing the value 200 octal. The contents of this DAC can be changed by writing a new data value to port B. A change of two LSBs in the vernier DAC will change the ADM's returned converted values by one LSB. However, unlike the offset vernier DAC, increasing the value in the gain vernier DAC decreases the returned digitized value. Decreasing the gain vernier DAC value increases the returned digitized value. Your calibration routine should write a loop which continues to decrease (or increase) the value in the gain vernier DAC until the returned values average 077776 (8).
- The ADM is now calibrated for gain.

PAGE 162 INTENTIONALLY LEFT BLANK



## Appendix A

#### Appendix A

#### Performance and Electrical Specifications

#### ELECTRICAL SPECIFICATIONS (25°C unless noted otherwise)

#### ANALOG PERFORMANCE

The following analog specifications assume a stable operating temperature environment. A stable operating temperature is defined as no more than a 2°C change in exhaust air over a 15 minute period.

#### A/D CONVERTER SYSTEM (including track and hold)

Fuil Scale Range (FSR): 10 Vdc

Bipolar Input ± 5 Vde

Resolution: 16 Bits

(Binary weighted, 1 part in 65536 of FSR; 1 LSB = 15.3 ppm of FSR.)

Absolute Zero Error: 0.0106% FSR Absolute Scaling Error: 0.0125% FSR

Monoticity (no missing codes 14 Bits

guaranteed):

No more than 3 missing codes at: 14, 12, 34 FSR

Differential Nonlinearity: 65525 states < 1 LSB Relative Accuracy Error: + 8 LSB **Temperature Coefficients:** Differential linearity: ± 0.07 LSB/°C + 0.63 LSB/°C Bipolar offset: ± 0.82 LSB/°C Gain: 1 LSB Noise RMS: (based on Gaussian distribution): PEAK: 3 LSB Total Conversion Time (maximum): Programmed gains 01, 04: 200 µs Programmed gains 16, 64: 400 µs Autoranging enabled (any gain): 100 us External Start Aperture Delay: 175 ns typ 300 ns max INPUT MULTIPLEXER (each input line) On Channel Leakage Current: + 16 nA Off Input Leakage Current: + 2 nA Input Capacitance (each line to ground): OFF channel: 25 pF ON channel: 70 pF Continuous Overvoltage Protection: + 24 V Power ON: Power OFF: + 10 V PROGRAMMABLE GAIN PREAMPLIFIER Selectable Gains: 1, 4, 16, 64 Gain Accuracy: 0.020% G = 1G = 40.035% G = 160.050% G = 640.080%

Common-mode Rejection Ratio (min): 74 dB

Input Impedance: 10 megohms

Full Scale Interchannel Settling Error: + 0.006% FSR

Autoranging Scale Switch Points: shown in Table A-1

Table A-1 Autoranging Scale Switch Points

Input Range	Gain
Fullscale - 24% Fullscale	1
24% Fullscale - 5.8% Fullscale	4
5.8% Fullscale - > 1% Fullscale	16
1° o Fullscale 0	64

NOTE: These ranges are typical values. They could deviate up to < 0.5% of full scale.

#### CODING FORMAT

The coding format is 2's complement.

DIFFERENTIAL INPUT VOLTAGE	WORD OUTPUT
Positive Full Scale (or overrange)	077777(8)
Positive Full Scale - 1 LSB	077776(8)
Zero volts + 1 LSB	000001(8)
Zero volts	000000(8)
Zero volts - 1 LSB	177777(8)
Negative Full Scale + 1 LSB	100001(8)
Negative Full Scale (or underrange)	100000(8)

#### DIGITAL PERFORMANCE

#### FRONT PANEL INPUTS

Minimum voltage level to set a logic 1: 1.6 Vdc

Maximum voltage level to reset logic 0: 0.8 Vdc

Maximum voltage range: 2-15 Volts

Minimum by steresis (Noise Margin): 50 mV

Minimum input impedance: 12K ohms

Minimum trigger Pulse Width: 200 ns

Trigger polarity: Negative transition

#### FRONT PANEL OUTPUTS

Minimum output voltage for a logic 1: 2 40 Vdc source current: @ 15 mA Maximum output voltage for a logic 0: 0.50 Vdc

sink current: @ 21 mA

#### HOST PORT

8-Bit Input to ADM from RTI on: PB0 - PB7 with 2-wire handshaking on: PC1, PC2 2-Bit Input to ADM from RTI on: PC6, PC7

PC6, PC7
without handshaking

8-Bit Output from ADM to RTI on: PA0 - PA7 with 2-wire handshaking on: PC4, PC5

Minimum voltage level to set a logic 1: 2.0 Vdc

Maximum voltage level to reset logic 0: 0.8 Vdc

#### REAL-TIME CLOCK

Eight Selectable Frequency sources:

256 kHz 64 kHz 16 kHz 4 kHz 1 kHz 250 Hz 62.5 Hz 15.625 Hz

Division Counter Size:

8 bits

Minimum Time Interval:

S us

Maximum Time Interval:

16.32 seconds

Maximum error in programmed value with interval selected between 250  $\mu s$  and 16.32 seconds:

± 0.8 %

#### POWER SUPPLY

AC Input Ranges:

90 to 130 Vac (@ .3 A) 180 to 260 Vac (@ .15 A)

AC Input Frequency:

47 to 63 Hz





## Appendix B

#### Appendix B

#### **ADM Sequence of Events**

The time line on the next page illustrates the sequence of events taken by the ADM in different trigger modes. Note that in all cases, the sequence of events is actually: channel acquisition, autoranging, auto-zeroing, and conversion. However, the triggering events occur at different places in this sequence as shown below. Additional information on how the different sequences of events may affect your data is found in the section Effective Aperture Delay—Some Cautions in Chapter 9.

Command write single sample:

<CA><AR><A2><CV>

Externally triggered single samples:

Command write clock triggering:

Externally started clock triggering:

Command write clocked sweeps:

<CA><AR><AZ><CV0><CA><AR><AZ><CV1>...CCT — |

Externally started sweeps:

- TIME

T=0 (Command Write)

Key

CA = Channel acquisition period

AR = Autoranging and amplifier settling period

AZ = Auto-zeroing period and signal capture (hold)

CV = Conversion to digital period

CS = Clock start synchronization point

CT = Clock trigger point

ET = External trigger point

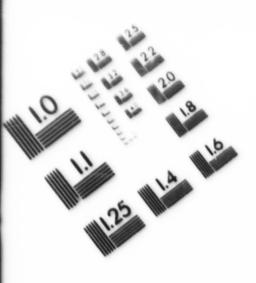
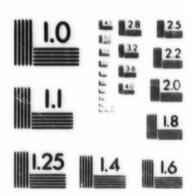
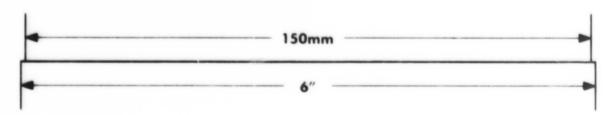


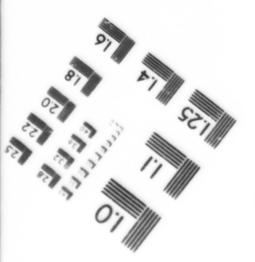
IMAGE EVALUATION TEST TARGET (MT-3)











### BELL HOWELL

Publication Systems Division

PHOTOGRAPHIC SCIENCES CORPORATION
770 BASKET ROAD
P.O. BOX 338
WEBSTER, NEW YORK 14580
(716) 265-1600

Oil Sill GE