GPIB-PCIIA

Technical Reference Manual

September 1989 Edition
Part Number 320045-01

GPIB-PCIIA Technical Reference Manual

GPIB-PCIIA
Technical Reference Manual

September 1989 Edition

Part Number 320045-01

Important Notice

The material in this manual is subject to change without notice. National Instruments assumes no responsibility for errors which may appear in this manual. National Instruments makes no commitment to update, nor to keep current, the information contained in this document.

Copyright

Preface

Following is a description of each section of the GPIB-PC2A Technical Reference Manual.

Section

1. General Information. This section contains a brief description of the GPIB-PC2A kit, a list of equipment supplied, a list of optional equipment, and a list of applicable documents. Terminology used in this manual is also explained in this section.

2. GPIB-PC2A Description. This section contains the physical and electrical specifications for the GPIB-PC2A and describes the characteristics of key interface board components.

3. Register Descriptions. This section contains detailed descriptions of the interface registers as well as summary tables for easy reference.

4. Programming Considerations. This section explains important considerations for programming the GPIB-PC2A.

5. Theory of Operation. This section contains a detailed technical discussion of the major elements of the GPIB-PC2A. References to the schematic diagrams are provided.

6. GPIB-PC2A Diagnostic and Troubleshooting Test Procedures. This section contains test procedures for determining if the GPIB-PC2A is installed properly and operating correctly.

Appendix

A. Parts List, PAL Drawings, and Schematic Diagram. This appendix includes a parts list, PAL drawings, and a detailed schematic diagram.

B. Multiline Interface Command Messages. This appendix provides a listing of the multiline GPIB interface command messages.

C. Mnemonics Key. This appendix contains an alphabetical listing of all mnemonics used in this manual and indicates whether the mnemonic represents a bit, register, function, remote message, local message or a state.

TABLE OF CONTENTS

Section Four - Programming Considerations

Section Five - Theory of Operations

Section Six - IB2AIFT - Operation and Specification

APPENDIX

LIST OF TABLES

LIST OF FIGURES

Section One

General Information

This section contains general information about the National Instruments GPIB-PC2A interface kit. This information includes a brief description of the GPIB-PC2A kit, a list of equipment supplied, a list of optional equipment, and a list of applicable reference documents.

1.1 GPIB-PC2A INTERFACE KIT DESCRIPTION

The National Instrument GPIB-PC2A, shown in Figure 1.1, is a multifunction interface for the IBM PC and compatibles such as the PC Expansion chassis, XT, Portable, and AT; COMPAQ and COMPAQ Plus; AT&T (or Olivetti) 6300; Zenith Z-150 and Z-160. The GPIB-PC2A provides an interface from the personal computer to the IEEE-488 General Purpose Interace Bus (GPIB), along with an optional time-of-day, real-time clock, with alarm interval and battery back-up.

In this manual, the GPIB standard is referred to as the IEEE-488 standard; the General Purpose Interface Bus system is referred to as the GPIB; and the time-of-day clock is referred to as the clock.

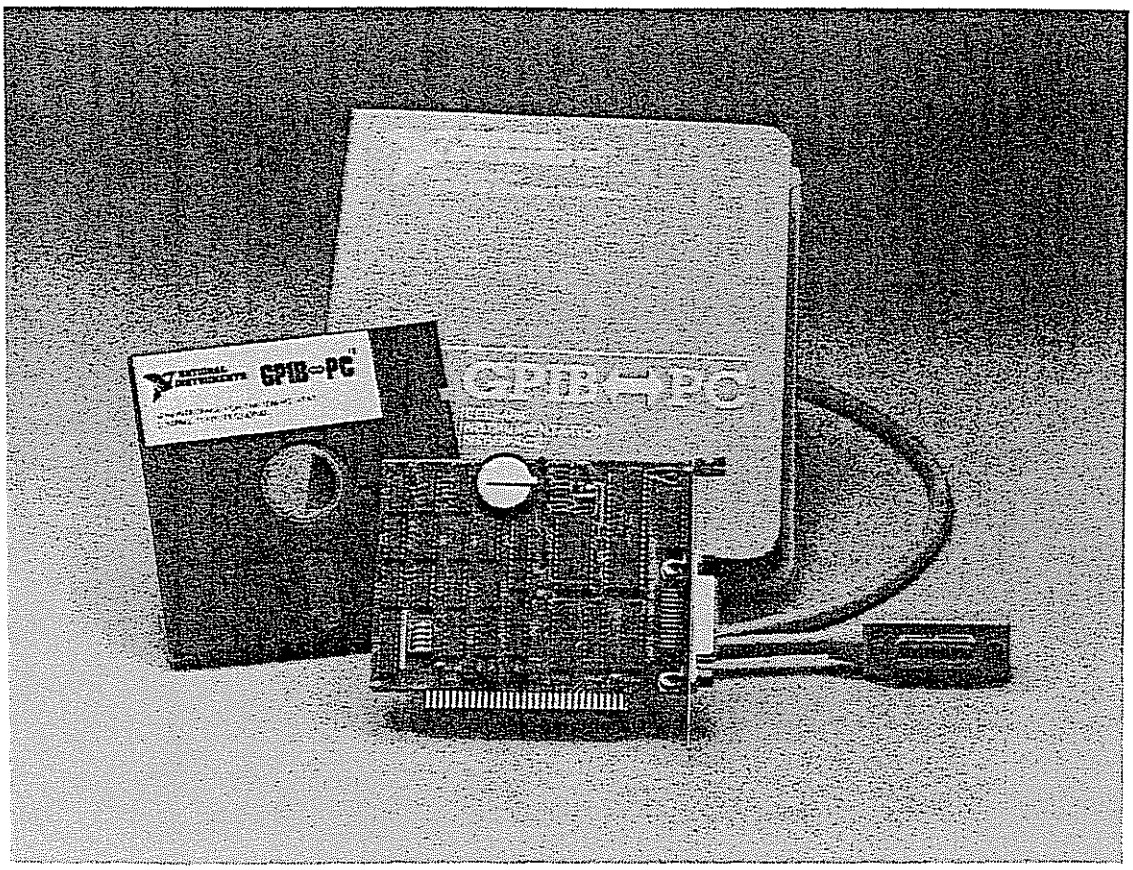


Figure 1.1 - GPIB-PC2A Interface Board with Optional Clock

1.2 EQUIPMENT SUPPLIED

Table 1.1 lists the equipment supplied in the National Instruments GPIB-PC2A Interface Kit. Optional equipment is listed in Table 1.2.

Table 1.1 - Equipment Supplied in GPIB-PC2A Kit

Equipment	NI Part Number
Bus Interface Card	
GPIB-PC2A	180210-01
or	
GPIB-PC2A with Clock Option	180210-02
GPIB-PC2A User Manual Supplement	320014-50

Table 1.2 - Accessory Equipment for GPIB-PC2A

Equipment	NI Part Number
GPIB Extension Cable - 1 meter	763001-01
GPIB Extension Cable - 2 meter	763001-02
GPIB Extension Cable - 4 meter	763001-03
GPIB-PC2A Technical Reference Manual	320045-02
GPIB-PC2A Technical Reference Manual,in Binder	320045-01

1.3 APPLICABLE DOCUMENTS

The following documents are references which cover in greater detail specific topics introduced in this manual:

* IEEE Std. 488-1978, "Standard Digital Interface for Programmable Instrumentation"

* uPD7210 GPIB-IFC User's Manual, NEC Electronics U.S.A., Inc., One Natick Executive Park, Natick, Massachusetts 01760

* uPD7210 INTELLIGENT GPIB INTERFACE CONTROLLER, Engineering Data Sheet, NEC Electronics U.S.A., Inc., Microcomputer Division

* HOW TO INTERFACE A MICROCOMPUTER SYSTEM TO A GPIB, (& The NEC uPD7210 TLC), NEC Electronics U.S.A., Inc.

1.4 TERMINOLOGY

The term "set" is used throughout this manual to mean writing a value of 1 to a memory element. The term "clear" is used to mean writing a value of 0 to a memory element. The terms "set to zero," "set false," "preset," and "reset" are avoided in preference to the use of "set" and "clear." Bit signatures are written in uppercase letters. An asterisk (*) after a bit signature indicates the signal is active low. An asterisk is equivalent to an overbar. For example, if bit ATN*=0, the GPIB ATN signal line is asserted. After the mnemonic of a register name is introduced and the name written out, the mnemonic is used thereafter.

Where it is necessary to specify a particular bit of a register, the bit position appears as a decimal number in square brackets after the mnemonic (e.g., ISR1[2] indicates the ERR bit of Interrupt Status Register 1).

Uppercase mnemonics are used for control, status, data registers, register contents, and interface functions, as well as GPIB remote messages, commands, and logic states, as defined in the IEEE-488 standard.

The term "addressed" means the interface has been configured to perform a function from the GPIB side, while the term "programmed" means that it has been configured to perform a function via instructions from the PC processor. This distinction is important because many functions, such as making the interface a Talker or Listener, can be activated from either side.

In logical expressions a minus sign (-) indicates logical negation, an ampersand (&) represents AND, and a plus sign (+) represents OR.

All numbers are decimal unless specified otherwise. Register offsets are given in hexadecimal.

Section Two

GPIB-PC2A Description

This section contains the physical and electrical specifications for the GPIB-PC2A. It provides a brief overview of the key interface board components including a functional block diagram.

2.1 GPIB-PC2A SPECIFICATIONS

This paragraph contains the electrical specifications and data transfer features of the GPIB-PC2A and describes the characteristics of key interface board components.

2.1.1 Physical Characteristics

The GPIB-PC2A measures 4.20 by 5.00 inches and is designed to fit in any slot in the IBM PC or slots J1 through J7 in the IBM PC/XT. The card is supplied with a standard 24-pin GPIB connector which fits through the rear panel slot.

2.1.2 Electrical Characteristics

Following is a list of the signals used by the GPIB-PC2A and the line loading presented by the circuitry on the card (in terms of device types and their part numbers):

Bus Signals	Driver	Receiver
D0 - D7	74LS245	74LS245
A15	---	74LS00 X 2
A14 - A13	---	74LS266
A12 - A10	---	74LS541
A9 - A0,AEN	---	PAL16L2
IOR*,IOW*,T/C, DACK1*,DACK2*, DACK3*,RESET DRV	---	74LS541
DRQ1,DRQ2,DRQ3, IRQ2-IRQ7,IORDY	74LS125A	---
CLOCK	---	74LS32

The GPIB transceivers meet the requirements of the IEEE-488 standard. The GPIB-PC2A requires regulated +5 VDC power from the IBM PC I/O channel. Current load is typically 0.5 A.

2.1.3 IBM PC I/O Channel Characteristics

The details of the GPIB-PC2A interface to the IBM PC I/O Channel are given in the following paragraphs.

2.1.3.1 Input/Output Addressing

The GPIB-PC2A card decodes the low-order ten address bits, A0 through A9, to obtain a hardwired base address of 2E1. The three address bits A10 through A12 are used by the NEC uPD7210 as register select lines RS0 through RS2, respectively. The address bits A10 through A12 are also used to select the two clock registers, CCR and CDR. The two address bits A13 and A14 are used to select one of the four possible block numbers, 1 through 4, thereby allowing up to four boards in one backplane. Address line A15 is used to select the NEC uPD7210 (A15=0) or the NS MM58167A (A15=1). Address lines A16 through A19 are not used.

A complete address map for each base address is given below. To change the GPIB-PC2A base address or interrupt level, refer to Section Two A of the User Manual.

BOARD #0

8088 Port Address	uPD7210 Register Mnemonic (Read)	(Write)	8088 Port Address	Clock Register Mnemonic (Read)	(Write)
02E1	DIR	CDOR	96E1	-	CCR
06E1	ISR1	IMR1	9AE1	CDR	CDR
0AE1	ISR2	IMR2			
0EE1	SPSR	SPMR			
12E1	ADSR	ADMR			
16E1	CPTR	AUXMR			
1AE1	ADR0	ADR			
1EE1	ADR1	EOSR			

BOARD #1

8088 Port Address	uPD7210 Register Mnemonic (Read)	(Write)	8088 Port Address	Clock Register Mnemonic (Read)	(Write)
22E1	DIR	CDOR	B6E1	-	CCR
26E1	ISR1	IMR1	BAE1	CDR	CDR
2AE1	ISR2	IMR2			
2EE1	SPSR	SPMR			
32E1	ADSR	ADMR			
36E1	CPTR	AUXMR			
3AE1	ADR0	ADR			
3EE1	ADR1	EOSR			

BOARD #2

8088 Port Address	uPD7210 Register Mnemonic (Read)	(Write)	8088 Port Address	Clock Register Mnemonic (Read)	(Write)
42E1	DIR	CDOR	D6E1	-	CCR
46E1	ISR1	IMR1	DAE1	CDR	CDR
4AE1	ISR2	IMR2			
4EE1	SPSR	SPMR			
52E1	ADSR	ADMR			
56E1	CPTR	AUXMR			
5AE1	ADR0	ADR			
5EE1	ADR1	EOSR			

BOARD #3

8088 Port Address	uPD7210 Register Mnemonic (Read)	(Write)	8088 Port Address	Clock Register Mnemonic (Read)	(Write)
62E1	DIR	CDOR	F6E1	-	CCR
66E1	ISR1	IMR1	FAE1	CDR	CDR
6AE1	ISR2	IMR2			
6EE1	SPSR	SPMR			
72E1	ADSR	ADMR			
76E1	CPTR	AUXMR			
7AE1	ADR0	ADR			
7EE1	ADR1	EOSR			

The following two points also need to be considered when determining the I/O space used by the GPIB-PC2A.

* In addition to the I/O space required for each board, an additional address block at hex 02F0-02F7 is reserved for a special interrupt handling feature of the GPIB-PC2A.

* Even if your GPIB-PC2A does not have the clock option, the clock address space cannot be used by another device.

2.1.3.2 Interrupts

Interrupt events that originate from the TLC are as follows:

GPIB Data In (DI)	Address Status Change (ADSC)
GPIB Data Out (DO)	Secondary Address received (APT)
END message received (END RX)	Service Request received (SRQI)
GPIB Command Out (CO)	Trigger command received (DET)
Remote mode change (REMC)	Device Clear received (DEC RX)
GPIB handshake error (ERR)	Unrecognized Command received (CPT)
Lockout change (LOKC)	

The TLC interrupt can be connected to any one of the six IBM PC interrupt request signals (IRQ2 through IRQ7) by means of an on-board jumper.

Interrupt events that originate from the clock are as follows:

10 times per second
Once per second
Once per minute
Once per hour
Once a day
Once a week
Once a month
Interrupt when a RAM/real-time counter comparison occurs

The clock interrupt can also be connected to any one of the six IBM PC interrupt request signals by means of an on-board jumper.

2.1.3.3 Direct Memory Access

Two GPIB events may cause a request for a DMA transfer. They are Data Out and Data In.

The GPIB-PC2A is designed to allow the user to select one of three pairs of DMA Channel signal lines by orienting the configuration jumpers. The DMA channel jumpers select Channel 1 as configured by the manufacturer.

2.1.3.4 Data and Command Input/Output

Command and data bytes are transferred between the IBM PC I/O Channel and the GPIB via the Command/Data Out and Data In registers of the GPIB adapter. Bidirectional transceivers buffer the bytes at the interface to each bus.

2.1.4 Data Transfer Features

The GPIB-PC2A transfers data to and from the GPIB using DMA and programmed I/O.

The GPIB-PC2A is capable of transferring data at rates up to 300K bytes per second using DMA transfers. The actual data rate and overall throughput achievable in any system are dependent on the transfer speed of the slowest device participating in the transfer and the number of bytes transferred in one operation. As a result, data rates and throughput can vary widely.

The data transfer rate of programmed I/O depends on how fast the program code executes.

2.2 DESCRIPTION OF THE GENERAL PURPOSE INTERFACE BUS

The IEEE-488 GPIB provides a means for communication among a group of interconnected devices. Two types of messages are carried by the bus:

* Interface messages, which are used for bus management, and

* Device dependent messages, which are communicated among the various devices via the interface bus but are not used or processed by the bus.

The three types of devices which organize and manage the flow of information on the bus are a Listener (L), a Talker (T), and (optionally) a Controller (C). A Listener has the capability of being addressed by an interface message to receive device dependent messages. A Talker has the capability of being addressed by an interface message to send device dependent messages. A Controller can address devices to Listen or Talk. A Controller can also send interface messages to command other specific actions within interfaced devices. A single bus may have one or more Controllers. If more than one Controller is connected to the bus, one is designated as the System Controller and may temporarily pass control to any other Controller.

The GPIB-PC2A is capable of being a Listener, a Talker, a Controller, and, in particular, a System Controller.

2.2.1 GPIB Operation

This section contains a simplified description of the operation of the GPIB. For more details, refer to the IEEE-488 standard.

Bus operation is divided into two logical functions: interface functions (for bus management) and device functions (for device control and communication). The interface functions, described in this manual, are basically used to establish an environment in which device functions may be performed. The device functions are unique to individual instruments and are beyond the scope of this manual.

The Source Handshake (SH) and Acceptor Handshake (AH) interface functions are used to transmit and receive bit-parallel, byte-serial messages (multiline messages) on the bus. These messages are interpreted either as commands to the interface or as data bytes for the device, depending on the status of the bus signal Attention (ATN). ATN is asserted for interface commands.

The Talker or Extended Talker (TE) and Listener or Extended Listener (LE) interface functions are used to set up devices for data transfers. The transfers are made using the handshake functions. When an interface has been addressed as a Talker, the applicable device function is allowed to use the interface Source Handshake function to transmit data bytes. The device functions of any interfaces that were addressed as Listeners must use the Acceptor Handshake function to receive the data bytes. Only one device at a time may be the Talker although multiple Listeners may exist.

The Controller function is the originator of all interface messages. Only one interface at a time may be the Controller-In-Charge and protocol exists to pass the control capability among interfaces. When the Controller-In-Charge is asserting the ATN line, it is in its active state and is often referred to as the Active Controller. When not asserting ATN, it is in its standby state. Any controller that is not the Controller-In-Charge remains in its idle state. At any time, there is, at the most, one interface on the bus that has the capability of making itself the Controller-In-Charge. It is referred to as the System Controller.

The Controller is responsible for addressing and unaddressing Talkers and Listeners, as well as performing other bus management operations. These other operations include conducting a Parallel Poll using the uniline message Identify (IDY); setting device functions in remote or local mode using the uniline message Remote Enable (REN); and initializing the bus by asserting the uniline message Interface Clear (IFC). The latter two operations are actually capabilities only of the System Controller and are not transferable by the transfer of control protocol.

2.2.2 GPIB Signal Lines

A total of 24 signal lines are used to implement the bus. Of these lines, 16 are signal lines, one is ground, one is the cable shield, and six are twisted pair common for six of the signal lines. The 16 signal lines are used to carry all information, interface messages, and device dependent messages among interconnected devices.

The 16 signal lines are organized into three sets, as follows:

* 8 data input/output lines

* 3 handshake lines

* 5 interface management signal lines

Negative logic with standard TTL levels is used on the GPIB. That is, a false (0) logic state corresponds to a TTL high level of 2.0 V or higher, and a true (1) logic state corresponds to a TTL low level of 0.8 V or lower.

The eight data lines, DI01 through DI08, carry all data including input, output, and device dependent messages. In many instruments data is based on the seven-bit ASCII code set.

The three handshake lines, NRFD, DAV, and NDAC, are used to effect the transfer of each byte of data on the DIO lines from an addressed Talker to all addressed Listeners. The three handshake lines provide a means to asynchronously transfer data among instruments.

The NRFD (Not Ready for Data) line is used to indicate the condition of readiness of devices to accept data. All instruments drive NRFD false when ATN becomes true. Only addressed Listeners drive NRFD false when ATN becomes false. The NRFD line is monitored by the Controller when ATN is true and by the addressed Talker when ATN is false. The NRFD line is false when all Listeners are ready for data and true when one or more Listeners are not ready for data.

The DAV (Data Valid) line is used to indicate the validity of data on the data lines. DAV is driven by the Controller when ATN is true and by the addressed Talker when ATN is false. The DAV line is monitored by all instruments if ATN is true and by addressed Listeners when ATN is false.

The NDAC (Not Data Accepted) line is used to indicate acceptance of data by addressed Listeners when ATN is false and acceptance of commands by all devices when ATN is true. Listeners indicate acceptance of data by setting NDAC false. When NDAC is true, one or more Listeners have not accepted the data.

The five bus management lines are ATN, IFC, REN, SRQ, and EOI. ATN and IFC are used by all instruments while the remaining three may or may not be used by a particular instrument.

All devices on the GPIB must monitor the ATN line. This line is set true by the Controller-In-Charge when it sends interface messages, such as device talk and listen addresses, secondary addresses, and polling configuration messages. When ATN is false, the active Talker can send device dependent messages, such as data and programming information, to active Listeners.

The IFC (Interface Clear) line is used by the System Controller to place the bus in a known quiescent state. The IFC line can only be driven true by the System Controller and must be monitored by all other instruments. To clear a device, the IFC line must be set true for at least 100 microseconds. IFC may be set true by the System Controller at any time.

The REN (Remote Enable) line is used to send the REN message. REN must be asserted as one of the conditions for operating an instrument in remote mode. The use of the remote function is optional. The REN line is driven true only by the System Controller and may be changed at any time. Instruments which use the REN line must monitor it at all times and return to local control whenever it becomes false.

The SRQ (Service Request) line is used by an instrument to asynchronously request service from the Controller-In-Charge.

The EOI (END or Identify) line is used either to indicate the end of a data string or to conduct a Parallel Poll, depending on the state of the ATN line. When ATN is false, the addressed Talker may send the END message to indicate the end of its data by setting EOI true at the same time it places the last data byte on the DIO lines. The Controller-In-Charge may send the IDY message to initiate a Parallel Poll of all instruments with Parallel Poll capability by setting ATN and EOI true simultaneously.

2.2.3 GPIB Physical Characteristics

Bus cables are used to interconnect instruments. There are three basic restrictions on cable length. First, the maximum length of any single span between two devices (loads) is four meters. Second, the maximum average length is two meters per device. Third, the maximum total length for all interconnected devices is 20 meters. With special high-performance cable, such as the HP 10833 series, the distance among devices is not critical provided the second and third restrictions are met. Instruments may be connected in a linear or star configuration. For applications requiring greater device separation, an extender such as the National Instruments GPIB-100 can be used.

Two 24-pin piggy-back connectors, one male and one female, are used on either end of the interconnecting cables. An overall cable shield is used to reduce susceptibility to noise.

2.3 GPIB-PC2A FUNCTIONAL DESCRIPTION

The GPIB-PC2A can be characterized as a bus translator, converting messages and signals of the IBM PC bus into appropriate GPIB messages and signals. Expressed in GPIB terminology, the GPIB-PC2A implements GPIB interface functions for communicating with other GPIB devices and device functions for communicating with the central processor and memory. From the point of view of the computer, the GPIB-PC2A is an interface to the outside world.

A block diagram of the GPIB-PC2A is shown in Figure 2.1. This diagram will be discussed in greater detail in Sections Three, Four, and Five.

The GPIB-PC2A interface consists of the buffers, drivers, and transceivers for the address, data, and control lines used on the PC I/O Channel, plus other logic circuitry that converts internal signals to bus-compatible signals. The software accesses registers within the TLC to configure, control, and monitor the GPIB interface functions. The registers can be used by the interface to interrupt the controlling program to inform it of the occurrence of an anticipated event. Special transceivers interface the TLC to the GPIB itself.

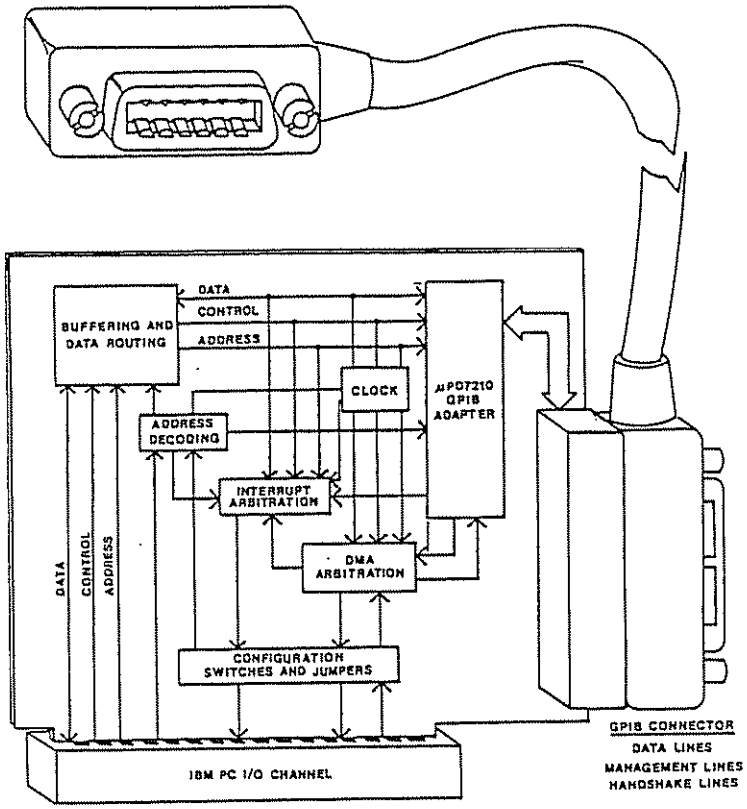


Figure 2.1 - GPIB-PC2A Block Diagram

Table 2.1 lists the capabilities of the GPIB-PC2A in terms of the codes in Appendix C of the IEEE-488 standard.

Table 2.1 - GPIB-PC2A IEEE-488 Interface Capabilities

Capability Code	Description
SH1	Complete Source Handshake capability
AH1	Complete Acceptor Handshake capability DAC and RFD holdoff on certain events
T5	Complete Talker capability Basic Talker Serial Poll Talk Only mode Unaddressed on MLA Send END or EOS Dual primary addressing
TE5	Complete Extended Talker capability Basic Extended Talker Serial Poll Talk Only Mode Unaddressed on MSA*LPAS Send END or EOS Dual extended addressing with software assist
L3	Complete Listener capability Basic Listener Listen Only mode Unaddressed on MTA Detect END or EOS Dual primary addressing
LE3	Complete Extended Listener capability Basic Listener Listen Only mode Unaddressed on MSA*TPAS Detect END or EOS Dual extended addressing with software assist
SR1	Complete Service Request capability
RL1	Complete Remote Local capability with software interpretation
PP1	Remote Parallel Poll configuration
PP2	Local Parallel Poll configuration with software assist
DC1	Complete Device Clear capability with software interpretation

Table 2.1 - GPIB-PC2A IEEE-488 Interface Capabilities (cont.)

Capability Code	Description
DT1	Complete Device Trigger capability with software interpretation
C1-5	Complete Controller capability System Controller Send IFC and take charge Send REN Respond to SRQ Send interface messages Receive Control Pass Control Parallel Poll Take Control Synchronously or Asynchronously
E1/2	Three-state bus drivers with automatic switch to open collector drivers during Parallel Poll

The GPIB-PC2A has complete Source and Acceptor Handshake capability. The GPIB-PC2A can operate as a basic Talker or Extended Talker and can respond to a Serial Poll. It may be placed in a Talk Only mode, and it will be unaddressed to Talk when it receives its Listen address. The interface can operate as a basic Listener or Extended Listener. It may be placed in a Listen Only mode, and it will be unaddressed to Listen when it receives its Talk address. The GPIB-PC2A has full capabilities for requesting service from another Controller. The ability to place the GPIB-PC2A in local mode is included but the interpretation of remote versus local mode is software dependent. Full Parallel Poll capability is included in the interface although local configuration requires software assistance. Device Clear and Trigger capability is included in the interface but the interpretation is software dependent. All Controller functions as specified by the IEEE-488 standard are included in the GPIB-PC2A. These include the capability to:

* Be System Controller

* Initialize the interface

* Send Remote Enable

* Respond to Service Request

* Send multiline command messages

* Receive control

* Pass control

* Conduct a Parallel Poll

* Take control synchronously or asynchronously

Section Three

Register Descriptions

This section presents detailed information on the use of the GPIB-PC2A internal program registers. This information is essential to programmers working with the GPIB-PC2A.

3.1 TERMINOLOGY

Refer to paragraph 1.4 for a description of the terminology used in this section. After a mnemonic has been defined, the mnemonic is used in the remainder of the section. Mnemonics are assigned to messages, states, registers and bits. Most mnemonics contain a clue to their meaning. Table 3.1 contains a list of clues for which to look.

Table 3.1 - Clues to Understanding Mnemonics

Clue	Mnemonic Probably Stands For:
Ends in IE	Interrupt enable bit
Ends in EN	Enable bit
4 letters, ends in S	Interface function as defined in the IEEE-488 standard
Ends in R, R0, R1, R2	GPIB program register
3 letters, uppercase	Remote message
3 letters, lowercase	Local message

Appendix C contains an alphabetical list of mnemonics.

3.2 INTERFACE REGISTERS

Eight of the ten program registers are contained in the NEC uPD7210 Talker/Listener/Controller (TLC) integrated circuit. The other two registers are used to control data flow to the clock and to enable interrupts from the clock. All software control of the GPIB-PC2A is performed through these I/O interface registers. Many of the registers are read only or write only. Some registers are not storage registers at all, but buffers through which status signals can be read or through which control signals can be sent. Five "hidden" registers are accessed indirectly. Each of the interface registers is addressed relative to one of the eight GPIB-PC2A block addresses.

Figure 3.1 shows the regular GPIB-PC2A TLC registers. Figure 3.2 shows the hidden registers of the TLC and illustrates the method of writing to those registers via the Auxiliary Mode Register. Notations within the figures indicate the read/write accessibility and the relative address of each register. A detailed function description of all the interface registers is provided in the paragraphs following the figures.

Legend

(Contents of Read Register)							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(Contents of Write Register)							

Address		6	5	4	3	2	1	0	Read/ Write
DIR	+0	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	R
CDOR		CDO7	CDO6	CDO5	CDO4	CDO3	CDO2	CDO1	CDO0	W
ISR1	+1	CPT	APT	DET	END RX	DEC	ERR	DO	DI	R
IMR1		CPT IE	APT IE	DET IE	END IE	DEC IE	ERR IE	DO IE	DI IE	W
ISR2	+2	INT	SRQI	LOK	REM	CO	LOKC	REMC	ADSC	R
IMR2		X	SRQI IE	DMAO	DMAI	CO IE	LOKC IE	REMC IE	ADSC IE	W
SPSR	+3	S8	PEND	S6	S5	S4	S3	S2	S1	R
SPMR		S8	rsv	S6	S5	S4	S3	S2	S1	W
ADSR	+4	CIC	ATN*	SPMS	LPAS	TPAS	LA	TA	MJMN	R
ADMR		ton	lon	TRM1	TRM0	X	X	ADM1	ADM0	W
CPTR	+5	CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	R
AUXMR		CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0	W
ADR0	+6	X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	R
ADR		ARS	DT	DL	AD5	AD4	AD3	AD2	AD1	W
ADR1	+7	EOI	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	R
EOSR		EC7	EC6	EC5	EC4	EC3	EC2	EC1	EC0	W

X indicates bit is not used. Not used read bits may read as 1 or 0.

Figure 3.1 - GPIB-PC2A TLC Registers

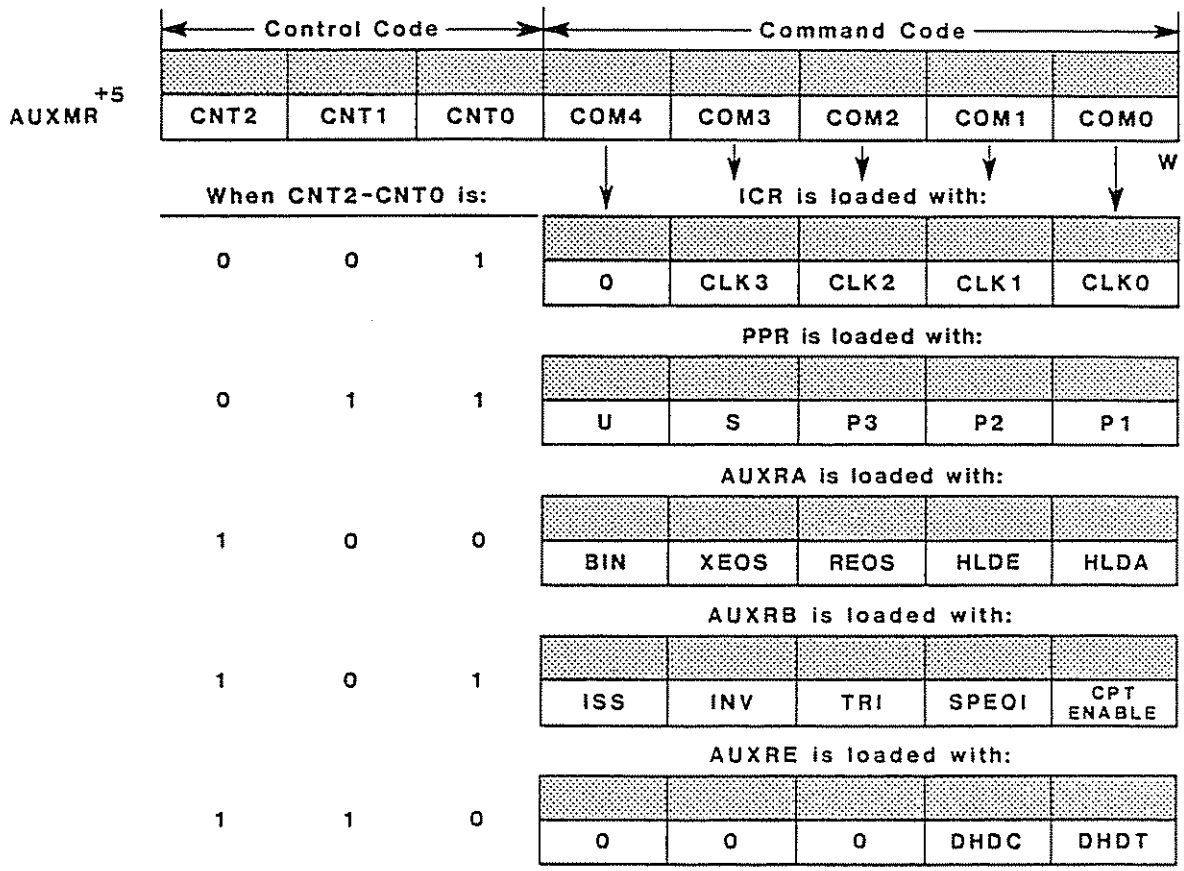


Figure 3.2 - Writing to the Hidden Registers

3.2.1 Data In Register (DIR)

7	6	5	4	3	2	1	0	R
DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	

I/O Channel Address: Block1 - 2E1; Block2 - 22E1;
Block3 - 42E1; Block4 - 62E1;

Attributes: Read Only

The DIR is used to move data from the GPIB to the PC when the interface is a Listener. Incoming information is separately latched by this register and is not destroyed by a write to the Command/Data Out Register, CDOR. The GPIB ready for data message, RFD, is held false until the byte is read from the DIR by the PC. The Acceptor Handshake (AH) completes automatically after the byte has been read. In RFD holdoff mode (refer to the description of Auxiliary Register A, AUXRA) the GPIB handshake is not finished until the Finish Handshake auxiliary command is issued telling the TLC to release the holdoff. By using the RFD holdoff mode, the same byte may be read several times, or an anxious GPIB Talker may be held off until the program is ready to proceed.

DI0 is the least significant bit of the data byte and corresponds to GPIB DIO1. DI7 is the most significant bit of the data byte and corresponds to GPIB DIO8.

Bit	Mnemonic	Description
7-0r	DIR7-DIR0	Data In Byte

3.2.2 Command/Data Out Register (CDOR)

7	6	5	4	3	2	1	0
CDO7	CDO6	CDO5	CDO4	CDO3	CDO2	CDO1	CDO0
							W

CDOR
I/O Channel Address: Block1 - 2E1; Block2 - 22E1;
Block3 - 42E1; Block4 - 62E1;

Attributes: Write Only

The CDOR register is used to move data from the PC to the GPIB when the TLC is the GPIB Talker or the Active Controller. Outgoing data is separately latched by this register and is not destroyed by a read from the DIR. When a byte is written to the CDOR, the TLC GPIB Source Handshake (SH) function is initiated and the byte is transferred to the GPIB.

Bit	Mnemonic	Description
7-0w	CDOR7-CDOR0	Command/Data Out Byte

3.2.3 Interrupt Status Register 1 (ISR1)
Interrupt Mask Register 1 (IMR1)

7	6	5	4	3	2	1	0	R
CPT	APT	DET	END RX	DEC	ERR	DO	DI	
CPT IE	APT IE	DET IE	END IE	DEC IE	ERR IE	DO IE	DI IE	
								W

ISR1

I/O Channel Address: Block1 - 6E1; Block2 - 26E1;
Block3 - 46E1; Block4 - 66E1;

Attributes: Read Only,
Bits are cleared when read

IMR1

I/O Channel Address: Block1 - 6E1; Block2 - 26E1;
Block3 - 46E1; Block4 - 66E1;

Attributes: Write Only

ISR1 is composed of eight interrupt status bits. IMR1 is composed of eight interrupt enable bits which directly correspond to the interrupt status bits in ISR1. As a result, ISR1 and IMR1 service eight possible interrupt conditions, where each condition has an interrupt status bit and an interrupt enable bit associated with it. If the enable bit is true when the corresponding status condition or event occurs, a hardware interrupt request is generated. Bits in ISR1 are set and cleared by the TLC regardless of the status of the interrupt bits in IMR1. If an interrupt condition occurs at the same time ISR1 is being read, the TLC holds off setting the corresponding status bit until the read has finished.

Bit	Mnemonic	Description
7r	CPT	Command Pass-Through
7w	CPT IE	Command Pass-Through Interrupt Enable

CPT is set on:

```
[UCG + ACG & (TADS + LADS)]
& undefined & ACDS &  -(CPT ENAB)
+ UDPCF & SCG & ACDS & CPT ENAB
```

CPT is cleared by:

```
pon + (Read ISR1)
```

Notes

UCG:	GPIB Universal Command Group message
ACG:	GPIB Addressed Command Group message
TADS:	GPIB Talker Addressed State
LADS:	GPIB Listener Addressed State
defined:	GPIB command automatically recognized and executed by TLC
undefined:	GPIB command not automatically recognized and executed by TLC
ACDS:	GPIB Accept Data State
CPT ENAB:	AUXRB[0]w
UDPCF:	Undefined primary command function (see below)
SCG:	GPIB Secondary Command Group message
pon:	power on reset
TAG:	GPIB Talk Address Group message
LAG:	GPIB Listen Address Group message
Read ISR1:	Bit is cleared immediately after it is read

UDPCF is set on:

```
[UCG + ACG & (TADS + LADS)] & undefined
& ACDS & CPT ENAB
```

UDPCF is cleared on:

```
[(UCG + ACG) & defined + TAG + LAG]
& ACDS + CPT ENAB* + pon
```

Bit	Mnemonic	Description

The CPT bit flags the occurrence of a GPIB command not recognized by the TLC, and all following GPIB secondary commands when the Command pass-through feature is enabled by the CPT ENAB bit, AUXRB[0]w. Any GPIB command message not decoded by the TLC is treated as an undefined command (for example, the Go To Local command, GTL). However, any addressed command is automatically ignored when the TLC is not addressed.

Undefined commands are read using the Command Pass Through Register (CPTR). The TLC holds off the GPIB Acceptor Handshake in the Accept Data State (ACDS) until the Valid auxiliary command function code, hex 0F, is written to the AUXMR. If the CPT feature is not enabled, undefined commands are simply ignored.

6r	APT	Address Pass-Through
6w	APT IE	Address Pass-Through Interrupt Enable

APT is set by:

ADM1 & ADM0 & (TPAS + LPAS) &
SCG & ACDS

APT is cleared by:

pon + (Read ISR1)

Notes

ADM1:	Address Mode Register bit 1, ADMR[1]w
ADM0:	Address Mode Register bit 0, ADMR[0]w
TPAS:	GPIB Talker Primary Addressed State
LPAS:	GPIB Listener Primary Addressed State
SCG:	GPIB Secondary Command Group
ACDS:	GPIB Accept Data State
pon:	power on reset
Read ISR1:	Bit is cleared immediately after it is read.

Bit	Mnemonic	Description

The APT bit indicates that a secondary GPIB address has been received and is available in the CPTR for inspection (Note: the application program must check this bit when using TLC address mode 3). When APT is set, the DAC message is held and the GPIB handshake stops until either the Valid or Non-Valid auxiliary command is issued. The secondary address can be read from the CPTR.

5r	DET	Device Execute Trigger
5w	DET IE	Device Execute Trigger Interrupt Enable

DET is set by:

DTAS

DET is cleared by:

pon + (Read ISR1)

Notes

DTAS:	GPIB Device Trigger Active State
pon:	power on reset
Read ISR1:	Bit is cleared immediately after it is read.

The DET bit indicates that the GPIB Device Execute Trigger (DET) command has been received while the TLC was a GPIB Listener (the TLC has been in DTAS).

Bit	Mnemonic	Description
4r	END RX	End Received
4w	END IE	End Received Interrupt Enable

END RX is set by:

LACS & (EOI + EOS & REOS) & ACDS

END RX is cleared by:

pon + (Read ISR1)

Notes

LACS:	GPIB Listener Active State
EOI:	GPIB End Or Identify Signal
EOS:	GPIB End of String message
REOS:	Reception of GPIB EOS allowed, AUXRA[2]w
ACDS:	GPIB Accept Data State
pon:	power on reset
Read ISR1:	Bit is cleared immediately after it is read.

The END RX bit is set when the TLC is a Listener and the GPIB uniline message, END, is received with a data byte from the GPIB Talker, or the data byte in the DIR matches the contents of the End Of String Register (EOSR).

Bit	Mnemonic	Description
3r	DEC	Device Clear
3w	DEC IE	Device Clear Interrupt Enable

DEC is set by:

DCAS

DEC is cleared by:

pon + (Read ISR1)

Notes

DCAS:	GPIB Device Clear Active State
pon:	power on reset
Read ISR1:	Bit is cleared immediately after it is read.

The DEC bit indicates that the GPIB Device Clear (DCL) command has been received or that the GPIB Selected Device Clear (SDC) command has been received while the TLC was a GPIB Listener (the TLC is in DCAS).

Bit	Mnemonic	Description
2r	ERR	Error
2w	ERR IE	Error Interrupt Enable

ERR is set by:

TACS & SDYS & DAC & RFD + SIDS &
(Write CDOR) + (SDYS->SIDS)

ERR is cleared by:

pon + (Read ISR1)

Notes

TACS:	GPIB Talker Active State
SDYS:	GPIB Source Delay State
DAC:	GPIB Data Accepted message
RFD:	GPIB Ready For Data message
SIDS:	GPIB Source Idle State
(Write CDOR):	Bit is set immediately after writing to the Command/Data Out Register
SDYS->SIDS:	Transition from GPIB Source Delay State to Source Idle State
pon:	power on reset
Read ISR1:	Bit is cleared immediately after it is read.

The ERR bit indicates that the contents of the CDOR have been lost. ERR is set when data is sent over the GPIB without a specified Listener or when a byte is written to the CDOR during SIDS or during the SDYS to SIDS transition.

Bit	Mnemonic	Description
1r	DO	Data Out
1w	DO IE	Data Out Interrupt Enable

DO is set as:

(TACS & SGNS) becomes true

DO is cleared by:

(Read ISR1) + TACS* + SGNS*

Notes

TACS:	GPIB Talker Active State
SGNS:	GPIB Source Generate State
Read ISR1:	Bit is cleared immediately after it is read.

The DO bit indicates that the TLC is ready to accept another data byte from IBM PC for transmission onto the GPIB when the TLC is the GPIB Talker. The DO bit is cleared when a byte is written to the CDOR and also when the TLC ceases to be the Active Talker. When performing a DMA operation, DO IE should be clear so that an interrupt request does not occur; instead, the DMAO bit in the Interrupt Mask Register 2 (IMR2[5]w) should be set to enable a DMA request when DO is asserted.

Bit	Mnemonic	Description
Or	DI	Data In
Ow	DI IE	Data In Interrupt Enable

DI is set by:

LACS & ACDS & Continuous Mode

DI is cleared by:

pon + (Read ISR1) + (Finish Handshake)
& (Holdoff Mode) + (Read DIR)

Notes

LACS:	GPIB Listener Active State
ACDS:	GPIB Accept Data State
Continuous Mode:	Listen In Continuous Mode auxiliary command in effect
pon:	power on reset
Read ISR1:	Bit is cleared immediately after it is read
Finish Handshake:	Finish Handshake auxiliary command issued
Holdoff Mode:	RFD holdoff state
Read DIR:	Read Data In Register

The DI bit indicates that the TLC, as a GPIB Listener, has accepted a data byte from the GPIB Talker. When performing a DMA operation, DI IE should be clear so that an interrupt request does not occur; instead, the DMAI bit in the Interrupt Mask Register 2 (IMR2[4]w) should be set to enable a DMA request when DI is asserted.

3.2.4 Interrupt Status Register 2 (ISR2)
Interrupt Mask Register 2 (IMR2)

7	6	5	4	3	2	1	0	R
INT 0	SRQI SRQI IE	LOK DMAO	REM DMAI	CO CO IE	LOKC LOKC IE	REMC REMC IE	ADSC ADSC IE	
								W

ISR2
I/O Channel Address: Block1 - AE1; Block2 - 2AE1;
Block3 - 4AE1; Block4 - 6AE1;

Attributes: Read Only,
Bits are cleared when read

IMR2
I/O Channel Address: Block1 - AE1; Block2 - 2AE1;
Block3 - 4AE1; Block4 - 6AE1;

Attributes: Write Only

ISR2 consists of six interrupt status bits and two TLC internal status bits. IMR2 consists of five interrupt enable bits and two TLC internal control bits. If the enable bit is true when the corresponding status condition or event occurs, an interrupt request is generated. Bits in ISR2 are set and cleared regardless of the status of the bits in IMR2. If a condition occurs which requires the TLC to set or clear a bit or bits in ISR2 at the same time ISR2 is being read, the TLC holds off setting or clearing the bit or bits until the read is finished.

Bit	Mnemonic	Description
7r	INT	Interrupt

This bit is the logical OR of all the enabled interrupt status bits in both ISR1 and ISR2, each one ANDed with its interrupt enable bit (refer below) There is no corresponding mask bit for INT. If the INT=1, the INT output pin of the TLC signal is asserted causing an interrupt pulse on the selected IRQ line.

INT is set by:

```
(CPT & CPT IE) + (APT & APT IE) +
(DET & DET IE) + (ERR & ERR IE) +
(END RX & END IE) + (DEC & DEC IE) +
(DO & DO IE) + (DI & DI IE) +
(SRQI & SRQI IE) + (REMC & REMC IE) +
(CO & CO IE) + (LOKC & LOKC IE) +
(ADSC & ADSC IE)
```

Bit	Mnemonic	Description
7w		Unused

Write zero to this bit.

6r	SRQI	Service Request Input
6w	SRQI IE	Service Request Input Interrupt Enable

SRQI is set when:

(CIC & SRQ & -(RQS & DAV)) becomes true or
(CIC & SRQ & RQS & DAV) becomes true
[see IMPORTANT NOTE below]

SRQI is cleared by:

pon + (Read ISR2)

Notes

CIC:	GPIB Controller In Charge
SRQ:	GPIB Service Request message
RQS:	GPIB Request Service message
DAV:	GPIB Data Valid message
pon:	power on reset
Read ISR2:	Bit is cleared immediately after it is read.

The SRQI bit indicates that a GPIB Service Request (SRQ) message has been received while the TLC Controller function is active (CIC=1).

IMPORTANT NOTE

The second equation shown above for setting SRQI applies exclusively to situations in which two or more devices are issuing the SRQ message.

5r	LOK	Lockout

LOK is used, along with the REM bit, to indicate the status of the TLC GPIB Remote/Local (RL) function. If set, the LOK bit indicates that the TLC is in Local With Lockout State (LWLS) or Remote With Lockout State (RWLS). LOK is a non-interrupt bit.

Bit	Mnemonic	Description
5w	DMAO	DMA Out Enable

The DMAO bit must be set to allow DMA transfers from IBM PC memory to the CDOR. DO IE should be clear, DMAO should be set, and the TLC must be the active GPIB Talker whenever a "DMA out" operation is initiated. If DMAO is set, the DO condition causes a DMA transfer request rather than an interrupt request. When DMAO is set, the GPIB-PC2A is enabled to drive the selected DMA Channel DRQ line. After the DMAO bit is set, the IBM PC DMA Controller should be set up to respond to a DMA request from the GPIB-PC2A. The DMA transfer will be a read from memory.

4r	REM	Remote

The REM bit is true whenever the TLC GPIB RL function is in one of two states: Remote State (REMS) or Remote With Lockout State (RWLS). The TLC RL function enters one of these states when the System Controller has asserted the Remote Enable line (REN), and the Controller-In-Charge addresses the TLC as a Listener.

4w	DMAI	DMA Input Enable

The DMAO bit must be set to allow DMA transfers from the DIR to IBM PC memory. DI IE should be clear, DMAI should be set, and the TLC must be an active GPIB Listener whenever a "DMA in" operation is initiated. If DMAI is set, the DI condition causes a DMA transfer request rather than an interrupt request. When DMAI is set, the GPIB-PC2A is enabled to drive the selected DMA Channel DRQ line. After the DMAI bit is set, the IBM PC DMA Controller should be set up to respond to a DMA request from the GPIB-PC2A. The DMA transfer will be a write to memory.

Bit	Mnemonic	Description
3r	CO	Command Out
3w	CO IE	Command Out Interrupt Enable

CO is set when:

(CACS & SGNS) becomes true

CO is cleared by:

(Read ISR2) + CACS* + SGNS*

Notes

CACS:	GPIB Controller Active State
SGNS:	GPIB Source Generate State
Read ISR2:	Bit is cleared immediately after it is read.

CO = 1 indicates that the CDOR is empty and that another command may be written to it for transmission over the GPIB without overwriting a previous command.

2w	LOKC	Lockout Change
2r	LOKC IE	Lockout Change Interrupt Enable

LOKC is set by:

any change in LOK

LOKC is cleared by:

pon + (Read ISR2)

Notes

LOK:	ISR2[5]r
pon:	power on reset
Read ISR2:	Bit is cleared immediately after it is read.

LOKC is set whenever there is a change in the LOK bit, ISR2[5]r, (REMS + RELS).

Bit	Mnemonic	Description
1w	REMC	Remote Change
1r	REMC IE	Remote Change Interrupt Enable

REMC is set by:

any change in REM

REMC is cleared by:

pon + (Read ISR2)

Notes

REM:	ISR2[4]r
pon:	power on reset
Read ISR2:	Bit is cleared immediately after it is read.

REMC is set whenever there is a change in the REM bit, ISR2[4]r, (REMS + RELS).

0r	ADSC	Addressed Status Change
0w	ADSC IE	Addressed Status Change Interrupt Enable

ADSC is set by:

[(any change in TA) + (any change in LA) + (any change in CIC) + (any change in MJMN)] & -(lon + ton)

ADSC is cleared by:

pon + (Read ISR2)

Notes

TA:	Talker Active bit, ADSR[1]r
LA:	Listener Active bit, ADSR[2]r
CIC:	Controller-In-Charge bit, ADSR[7]r
MJMN:	Major/Minor bit, ADSR[0]r
lon:	Listen Only bit, ADMR[6]w
ton:	Talk Only bit, ADMR[7]w
pon:	power on reset
Read ISR2:	Bit is cleared immediately after it is read.
ADSR:	Address Status Register
ADMR:	Address Mode Register

Bit	Mnemonic	Description
		ADSC is set whenever there is a change in one of the four bits: TA, LA, CIC, MJMN of the Address Status Register (ADSR).

3.2.5 Serial Poll Status Register (SPSR)
Serial Poll Mode Register (SPMR)

7	6	5	4	3	2	1	0	R
S8	PEND	S6	S5	S4	S3	S2	S1	
S8	rsv	S6	S5	S4	S3	S2	S1	
								W

SPSR
I/O Channel Address: Block1 - EE1; Block2 - 2EE1;
Block3 - 4EE1; Block4 - 6EE1;

Attributes: Read Only

SPMR
I/O Channel Address: Block1 - EE1; Block2 - 2EE1;
Block3 - 4EE1; Block4 - 6EE1;

Attributes: Write Only

Bit	Mnemonic	Description
7r	S8	Serial Poll Status Byte
7w, 5-0w, 5-0r	S6-S1	

Cleared by power on reset and by issuing the Chip Reset auxiliary command. These bits are used for sending device or system dependent status information over the GPIB when the TLC is serially polled. When the TLC is addressed as the GPIB Talker and receives the GPIB multiline Serial Poll Enable command message, SPE, it transmits a byte of status information, SPMR[7-0], to the Controller-In-Charge after the Controller Goes to Standby and becomes an Active Listener.

6r PEND Pending

PEND is set when rsv=1 and cleared when NPRS and rsv=1. (NPRS stands for Negative Poll Response State.) Reading the PEND status bit can confirm that a request was accepted and that the Status Byte (STB) was transmitted (PEND=0).

Bit	Mnemonic	Description
6w	rsv	Request Service

The rsv bit is used for generating the GPIB local request service message. When rsv is set and the GPIB Active Controller is not serially polling the TLC, the TLC enters the Service Request State (SRQS) and asserts the GPIB SRQ signal. When the Active Controller reads the STB during the poll, the TLC clears rsv at the Affirmative Poll Response State (APRS). The rsv bit is also cleared by power on reset and by issuing the Chip Reset auxiliary command.

3.2.6 Address Status Register (ADSR)

I/O Channel Address: Block1 - 12E1; Block2 - 32E1
Block3 - 52E1; Block4 - 72E1

Attributes: Read Only

7	6	5	4	3	2	1	0	R
CIC	ATN*	SPMS	LPAS	TPAS	LA	TA	MJMN	

The ADSR contains information that can be used to monitor the TLC GPIB address status.

Bit	Mnemonic	Description
7r	CIC	Controller-In-Charge CIC = -(CIDS + CADS) CIC indicates that the TLC GPIB Controller function is in an active or standby state, with ATN* on or off, respectively. The Controller function is in an idle state, with ATN* off, if CIC=0.
6r	ATN*	Attention* ATN* is a status bit which indicates the current level of the GPIB ATN* signal. If ATN* is 0, the GPIB ATN* signal is asserted.
5r	SPMS	Serial Poll Mode State If SPMS=1, the TLC GPIB Talker (T) or Talker Extended (TE) function is enabled to participate in a serial poll. SPMS is set when the TLC has been addressed as a GPIB Talker and the GPIB Active Controller has issued the GPIB Serial Poll Enable (SPE) command message. SPMS is cleared when the GPIB Serial Poll Disable (SPD) command is received by power on reset, LMR (CR0[2]w); or by issuing the Chip Reset auxiliary command.
4r	LPAS	Listener Primary Addressed State The LPAS bit is used when the TLC is configured for extended GPIB addressing and, when set, indicates that the TLC has received its primary Listen address. In Mode 3 addressing (see Address Mode Register Description in paragraph 3.2.7), LPAS=1 indicates that the secondary address being received on the next GPIB command

Bit	Mnemonic	Description

may represent the TLC Extended (Secondary) GPIB Listen address. LPAS is cleared by power on reset or by issuing the Chip Reset auxiliary command.

3r TPAS Talker Primary Addressed State

TPAS is used when the TLC is configured for extended GPIB addressing and, when set, indicates that the TLC has received its primary GPIB Talk address. In Mode 3 addressing, TPAS=1 indicates that the secondary address being received as the next GPIB command message may represent the TLC extended (secondary) GPIB Talk address.

2r LA Listener Active

LA is set whenever the TLC has been addressed or programmed as a GPIB Listener (i.e., the TLC is in the Listener Active State, LACS, or the Listener Addressed State, LADS). The TLC can be addressed to listen either by sending its own listen or extended listen address while it is Controller-In-Charge or by receiving its listen address from another Controller-In-Charge. It can also be programmed to listen using the lon bit in the Address Mode Register (ADMR).

If the TLC is addressed to Listen, it is automatically unaddressed to Talk. LA is cleared by power on reset, or by issuing the Chip Reset auxiliary command.

1r TA Talker Active

TA is set whenever the TLC has been addressed or programmed as the GPIB Talker (i.e., the TLC is in the Talker Active State, TACS; the Talker Addressed State, TADS; or the Serial Poll Active State, SPAS). The TLC can be addressed to talk either by sending its own talk or extended talk address while it is Controller-In-Charge or by receiving its talk address from another Controller-In-Charge. It can also be programmed to talk using the ton bit in the ADMR.

If the TLC is addressed to Talk it is automatically unaddressed to Listen. TA is cleared by power on reset or by issuing the Chip Reset auxiliary command.

Bit	Mnemonic	Description
0r	MJMN	Major-Minor

The MJMN bit is used to determine whether the information in the other ADSR bits applies to the TLC major or minor Talker/Listener function. MJMN is set to 1 when the TLC GPIB minor Talk address or minor Listen address is received. MJMN is cleared on receipt of the TLC major Talk or major Listen address. It should be noted that only one Talker/Listener may be active at any one time. Thus, the MJMN bit indicates which, if either, of the TLC Talker/Listener functions is addressed or active. MJMN is always zero unless a dual primary addressing mode (Mode 1 or Mode 3) is enabled (see the description of the Address Mode Register (ADMR) in paragraph 3.2.7).

3.2.7 Address Mode Register (ADMR)

I/O Channel Address: Block1 - 12E1; Block2 - 32E1
Block3 - 52E1; Block4 - 72E1

Attributes: Write Only

7	6	5	4	3	2	1	0
ton	lon	TRM1	TRM0	0	0	ADM1	ADM0
							W

Bit	Mnemonic	Description
7w	ton	Talk Only

Setting ton programs the TLC to be a GPIB Talker. If ton is set, the lon, ADM1, and ADM0 bits should be cleared. This method should be used in place of the addressing method when the TLC will be only a Talker.

Note: Clearing ton does not by itself take the TLC out of GPIB Talker Active State (TACS). It is also necessary to execute the Chip Reset or Immediate Execute pon auxiliary command.

6w lon Listen Only

Setting lon programs the TLC to be a GPIB Listener. If lon is set, ton, ADM1, and ADM0 should be cleared.

Note: Clearing lon does not by itself take the TLC out of GPIB Listener Active State (LACS). It is also necessary to execute the Chip Reset or Immediate Execute pon auxiliary command.

Bit	Mnemonic	Description
5-4w	TRM1-0	Transmit/Receive Mode

TRM1 and TRM0 control the function of the TLC T/R2 and T/R3 output pins in the following manner:

TRM1	TRM0	T/R2	T/R3
0	0	EOI OE	TRIG
0	1	CIC	TRIG
1	0	CIC	EOI OE
1	1	CIC	PE

Key

EOI OE = GPIB EOI signal output enable
CIC = Controller-In-Charge
TRIG = Trigger
PE = Pull-up Enable

For proper operation, set both TRM1 and TRM0 (which selects T/R2 = CIC and T/R3 = PE).

3-2w Unused

Write zeros to these bits.

1-0w ADM1-0 Address Mode

These bits specify the addressing mode which is in effect; i.e., the manner in which the information in ADR0 and ADR1 is interpreted. If both bits are 0 then the TLC does not respond to GPIB address commands; instead, the ton and lon bits are used to program the Talker and Listener functions, respectively. The ton and lon bits must be cleared if Mode 1, 2, or 3 addressing is selected, and the AMD1-0 bits must be cleared if either of the bits ton or lon are set.

Bit Mnemonic Description

Mode	ADM1	ADM0	Title
0	0	0	ton/lon
1	0	1	Normal dual addressing
2	1	0	Extended single addressing
3	1	1	Extended dual addressing

In Mode 1 ADR0 and ADR1 contain the major and minor addresses, respectively, for dual primary GPIB address applications (i.e., the TLC responds to two GPIB addresses, a major one and a minor one). The MJMN bit in the ADSR indicates which address was received. In applications where the TLC needs to respond to only one address, the major Talker/Listener function is used and the minor Talker/Listener function should be disabled.

In Mode 2 (ADM1=1, ADM0=0), the TLC recognizes two sequential GPIB address bytes, a primary followed by a secondary. Both GPIB address bytes must be received in order to enable the TLC to Talk or Listen. In this manner, Mode 2 addressing implements the Extended Talker and Extended Listener functions as defined in IEEE-488, without requiring computer program intervention. In Mode 2, ADR0 and ADR1 contain the TLC primary and secondary GPIB addresses, respectively.

In Mode 3 (ADM1=1, ADM0=1), the TLC handles addressing just as it does in Mode 1, except that each major or minor GPIB primary address must be followed by a secondary address. All secondary GPIB addresses must be verified by computer program when Mode 3 is used. When the TLC is in TPAS or LPAS (Talker/Listener Primary Addressed State), and a secondary address byte is on the GPIB DIO lines, the APT bit of ISR2 is set and the secondary GPIB address may be inspected in the CPTR (Command Pass Through Register). The TLC Acceptor Handshake is held up in the Accept Data State (ACDS) until the Valid or Non-Valid auxiliary command is written to the AUXMR, signaling a valid or invalid secondary address, respectively, to the TLC.

ADM0 and ADM1 should both be cleared when either of the two programmable bits ton or lon is set.

3.2.8 Command Pass Through Register (CPTR)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

Attributes: Read Only

7	6	5	4	3	2	1	0	R
CPT7	CPT6	CPT5	CPT4	CPT3	CPT2	CPT1	CPT0	

Bit	Mnemonic	Description
7-0r	CPT7-0	Command Pass Through Byte

This register is used to transfer undefined multiline GPIB command messages from the GPIB DIO lines to the IBM PC. When the CPT feature is enabled (CPT ENAB=1, AUXRB[0]w), any GPIB Primary Command Group (PCG) message not decoded by the TLC is treated as an undefined command. All GPIB secondary command group (SCG) messages following an undefined GPIB PCG message are also treated as undefined. In such a case, when an undefined GPIB message is encountered, it is held in the CPTR and the TLC Acceptor Handshake function is held off (in ACDS) until the Valid auxiliary command is written to the AUXMR. The CPTR is also used to inspect secondary addresses when Mode 3 addressing is used. The TLC Acceptor Handshake function is held off (in ACDS) until the Valid or Non-Valid auxiliary command is written to the AUXMR.

The CPTR is read during a TLC-initiated Parallel Poll operation to fetch the Parallel Poll response, and the PPR message is latched into the CPTR when CPPS is set, until CIDS is set, or a command byte is sent over the GPIB.

3.2.9 Auxiliary Mode Register (AUXMR)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

Attributes: Write Only,
Permits Access to Hidden Registers

7	6	5	4	3	2	1	0
CNT2	CNT1	CNT0	COM4	COM3	COM2	COM1	COM0
							W

The AUXMR is used to issue auxiliary commands. It is also used to program the five hidden registers:

1. Auxiliary Register A (AUXRA)
2. Auxiliary Register B (AUXRB)
3. Parallel Poll Register (PPR)
4. Auxiliary Register E (AUXRE)
5. Internal Counter Register (ICR)

Table 3.2 shows the control and command codes implemented.

Bit	Mnemonic	Description
7-5w	CNT2-CNT0	Control Code
		These bits specify the control code; i.e., the manner in which the information in bits COM4-COM0 is to be used. If CNT2-CNT0 are all 0, then the special command selected by COM4-COM0 is executed. Otherwise the hidden register selected by CNT2-CNT0 is loaded with the data from COM4-COM0.
4-0w	COM4-COM0	Command Code
		These bits specify the command code of the special function if the control code is 000. Table 3.3 is a summary of the implemented special functions. Table 3.4 explains the functioning details of the special functions. If the control code is not 000, then these bits are written to one of the hidden registers (indicated by the control code in CNT2-CNT0).

Table 3.2 - Auxiliary Command Summary

Function Code* (COM4-COM0)					Octal	
4	3	2	1	0	Code**	Auxiliary Command
0	0	0	0	0	000	Immediate Execute pon
0	0	0	1	0	002	Chip Reset
0	0	0	1	1	003	Finish Handshake
0	0	1	0	0	004	Trigger
0	0	1	0	1	005	Return to Local
0	0	1	1	0	006	Send EOI
0	0	1	1	1	007	Non-Valid Secondary Command or Address
0	1	1	1	1	017	Valid Secondary Command or Address
0	0	0	0	1	001	Clear Parallel Poll Flag
0	1	0	0	1	011	Set Parallel Poll Flag
1	0	0	0	1	021	Take Control Asynchronously (pulsed)
1	0	0	1	0	022	Take Control Synchronously
1	1	0	1	0	032	Take Control Synchronously on End
1	0	0	0	0	020	Go To Standby
1	0	0	1	1	023	Listen
1	1	0	1	1	033	Listen in Continuous Mode
1	1	1	0	0	034	Local Unlisten
1	1	1	0	1	035	Execute Parallel Poll
1	1	1	1	0	036	Set IFC
1	0	1	1	0	026	Clear IFC
1	1	1	1	1	037	Set REN
1	0	1	1	1	027	Clear REN
1	0	1	0	0	024	Disable System Control

* CNT2-CNT0 set to 000 binary
** Represents all eight bits of the Auxiliary Mode Register

Table 3.3 - Auxiliary Commands: Detail Description

The following functions are executed when the AUXMR Control Code (CNT2-CNT0) is loaded with 000 (binary) and the Command Code (COM4-COM0) is loaded as shown below.

Command Code (COM4-COM0) 4 3 2 1 0	Description
0 0 0 0 0	Immediate Execute pon This command generates a local pon (power on reset) message that places the following GPIB interface functions into their idle state: AIDS Acceptor Idle State CIDS Controller Idle State LIDS Listener Idle State LOCS Local State LPIS Listener Primary Idle State NPRS Negative Poll Response State PPIS Parallel Poll Idle State PUCS Parallel Poll to Unaddressed to Configure State SIDS Source Idle State SIIS System Control Interface Clear Idle State SPIS Serial Poll Idle State SRIS System Control Remote Enable Idle State TIDS Talker Idle State TPIS Talker Primary Idle State If the command is sent while a pon message is already active (by either an external reset pulse or the Chip Reset auxiliary command), the pon local message becomes false.
0 0 0 1 0	Chip Reset The Chip Reset command performs the same function as an external reset pulse. The TLC is reset to the following conditions: - local message pon is set and the interface functions are placed in their idle states;

Table 3.3 - Auxiliary Commands: Detail Description (continued)

Command Code (COM4-COM0) 4 3 2 1 0	Description
	- all bits of the serial poll mode register are cleared; - EOI bit is cleared; - all bits of the auxiliary A, B, and E registers are cleared; - the Parallel Poll flag and RSC local message are cleared; - sets NF=8 (F3 set to 1; F2, F1, and F0 set to 0) - clears the TRM0 bit and the TRM1 bit;
0 0 0 1 1	Finish Handshake (FH) The Finish Handshake command finishes a GPIB handshake that was stopped because of a holdoff on RFD or DAC.
0 0 1 0 0	Trigger The Trigger command generates a high pulse on the TRIG pin (T/R3 pin when TRM1=0) of the TLC. The Trigger command performs the same function as if the DET (Device Trigger) bit (ISR1[5]r) were set. (The DET bit is not set by issuing the Trigger command.)
0 0 1 0 1 0 1 1 0 1	Return to Local (rtl) Return to Local (rtl) The two Return to Local commands implement the rtl message as defined by IEEE-488. When COM3 is 0, the message is generated in the form of pulses. When COM3 is 1, the rtl command is set in the standard manner.
0 0 1 1 0	Send EOI (SEOI) The Send EOI command causes the GPIB EOI line to go true with the next byte transmitted. The EOI line is then cleared upon completion of the handshake for that byte. The TLC recognizes the Send EOI command only if TA=1 (i.e., the TLC is addressed as the GPIB Talker).

Table 3.3 - Auxiliary Commands: Detail Description (continued)

Command Code (COM4-COM0) 4 3 2 1 0	Description
0 0 1 1 1	Non-Valid Secondary Command or Address The Non-Valid command releases the GPIB DAC message held off by the address pass through. The TLC is permitted to operate as if an OSA (Other Secondary Address) message has been received.
0 1 1 1 1	Valid Secondary Command or Address The Valid command releases the GPIB DAC message held off by address pass through and allows the TLC to function as if an MSA (My Secondary Address) message had been received. The DAC message is released at the time of command pass through. DAC is also released if DCAS or DTAS is in holdoff state.
0 0 0 0 1 0 1 0 0 1	Clear Parallel Poll Flag Set Parallel Poll Flag These commands set the Parallel Poll Flag to the value of COM3. The value of the Parallel Poll flag is used as the local message ist when bit 4 of Auxiliary Register B is 0. The value of SRQS is used as the ist when ISS=1.
1 0 0 0 0	Go To Standby The Go To Standby command sets the local message gts if the TLC is in Controller Active State (CACS) or when it enters CACS. When the TLC leaves CACS, gts is cleared.
1 0 0 0 1	Take Control Asynchronously The Take Control Asynchronously command pulses the local message tca.
1 0 0 1 0	Take Control Synchronously The Take Control Synchronously command sets the local message tcs. The local message tcs is effective only when the TLC is in CSBS (Controller Standby State) or CSWS (Controller Synchronous Wait State). The local message tcs is cleared when the TLC enters CACS (Controller Active State).

Table 3.3 - Auxiliary Commands: Detail Description (continued)

Command Code (COM4-COM0) 4 3 2 1 0	Description
1 1 0 1 0	Take Control Synchronously on END
	The Take Control Synchronously on END command sets the local message tcs when the data block transfer end message (END bit equal to 1) is generated at CSBS (Controller Stand By State). The tcs message is cleared when the TLC enters CACS (Controller Active State).
1 0 0 1 1	Listen
	The Listen command generates the local message ltn in the form of a pulse.
1 1 0 1 1	Listen in Continuous Mode
	The Listen in Continuous Mode command generates the local message ltn in the form of a pulse and places the TLC in continuous mode.
	In continuous mode, the local message rdy is issued when the (Acceptor Not Ready State) ANRS is initiated unless data block transfer end is detected (END bit is 1). When END is detected, the TLC is placed in the RFD holdoff state, preventing generation of the rdy message. In continuous mode, the DI bit is not set when a data byte is received. The continuous mode caused by the Listen in Continuous Mode command is released when the Listen auxiliary command is issued or the TLC enters the Listener Idle State (LIDS).
1 1 1 0 0	Local Unlisten
	The Local Unlisten command generates the local message lun in the form of a pulse.

Table 3.3 - Auxiliary Commands: Detail Description (continued)

Command Code (COM4-COM0) 4 3 2 1 0	Description
1 1 1 0 1	Execute Parallel Poll
	The Execute Parallel Poll command sets the local message rpp (request parallel poll). The rpp message is cleared when the TLC enters either CPPS (Controller Parallel Poll State) or CIDS (Controller Idle State). The transition of the TLC Controller interface function is not guaranteed if the local messages rpp (request parallel poll) and gts (go to standby) are issued simultaneously when the TLC is in CACS (Controller Active State) and STRS (Source Transfer State) or SDYS (Source Delay State).
1 1 1 1 0	Set IFC
1 0 1 1 0	Clear IFC
	These commands generate the local message rsc (request system control) and set IFC to the value of COM3. In order to meet the IEEE-488, you must not issue the Clear IFC command until IFC has been held true for at least 100 microseconds.
1 1 1 1 1	Set REN
1 0 1 1 1	Clear REN
	These commands generate the local message rsc (request system control) and set REN to the value in COM3. In order to meet IEEE-488 requirements, you must not issue the Set REN command until REN has been held false for at least 100 microseconds.
1 0 1 0 0	Disable System Control
	The Disable System Control command clears the local message rsc (request system control).

3.2.10 Hidden Registers

The hidden registers are loaded through the Auxiliary Mode Register, AUXMR. AUXMR[7-5] is loaded with the hidden register number, and AUXMR[4-0] is loaded with the data to be transferred to the hidden register. The hidden registers cannot be read, and in some cases the contents are setable only; i.e., they can be cleared or reset to initialized conditions only by issuing the Chip Reset auxiliary command, by a power on reset, or by LMR (CR0[2]w). Figure 3.3 shows the five hidden registers and illustrates how they are loaded with data from the AUXMR.

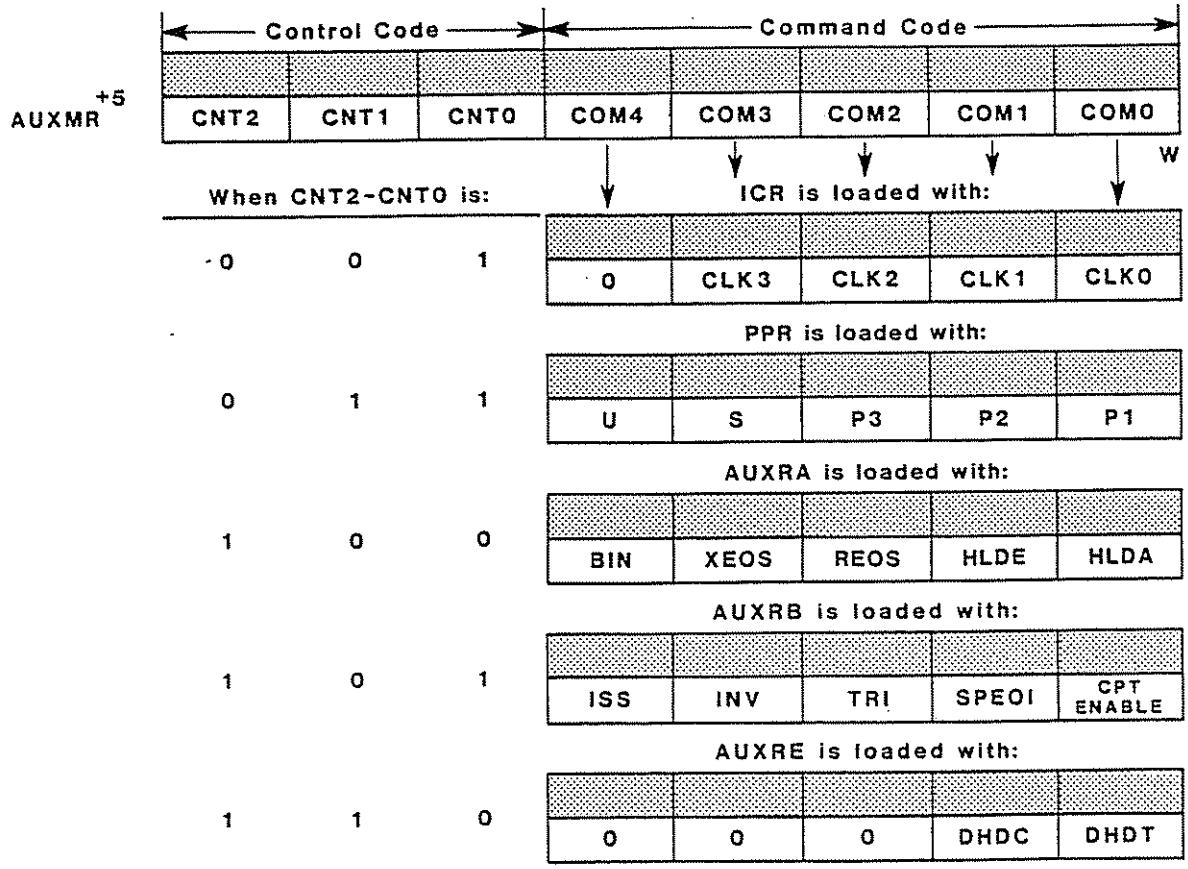


Figure 3.3 - Writing to the Hidden Registers

3.2.10.1 Internal Counter Register (ICR)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

AUXMR Control Code: 001 (binary, bits 7 - 5)
Attributes: Write Only, Accessed through AUXMR

4	3	2	1	0
0	CLK3	CLK2	CLK1	CLK0

W

Bit	Mnemonic	Description
4w		Not used. Write zero to this bit.
3-0w	CLK3-CLK0	Clock

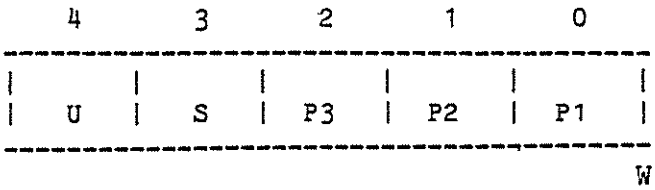
The contents of the ICR are used to divide internal counters that generate TLC state change delay times that are specified by the IEEE-488 specification. The most familiar of these times, T1, is the minimum delay between placing the data or command bytes on the GPIB DIO lines and asserting DAV. These delay times vary depending on the type of transfer in progress and the value of the AUXRB bit TRI.

For proper operation in the GPIB-PC2A, ICR should be set to 4 because the TLC is clocked at 3.6864 MHz.

3.2.10.2 Parallel Poll Register (PPR)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

AUXMR Control Code: 011 (binary, bits 7 - 5)
Attributes: Write Only, Internal to TLC



Writing to the Parallel Poll Register is done via the AUXMR. Writing the binary value 011 into the Control Code (CNT2-CNT0) and a bit pattern into the command code portion (COM4-COM0) of the AUXMR causes the command code to be written to the Parallel Poll Register (PPR). When COM4-COM0 is written to the PPR, the bits are named as shown in the figure above. This five-bit command code determines the manner in which the TLC responds to a Parallel Poll.

When using remote Parallel Poll configuration (capability code PP1), do not write a non-zero value to the PPR. The TLC implements remote configuration fully and automatically without software assistance. The hardware recognizes, interprets, and responds to the Parallel Poll configure (PPC), enable (PPE), disable (PPD), and identify (IDY) messages. The user need only set or clear the ist message (using the Set/Clear Parallel Poll Flag auxiliary commands) according to pre-established system protocol convention.

When using local parallel poll configuration (capability code PP2), a valid PPE or PPD message should be written to the PPR in advance of the poll.

Bit	Mnemonic	Description
4w	U	Parallel Poll Unconfigure

The U bit determines whether or not the TLC participates in a Parallel Poll. If U=0, the TLC participates in Parallel Polls and responds in the manner defined by PPR[3] through PPR[0] and by ist. If U=1, the TLC does not participate in a Parallel Poll.

The U bit is equivalent to the local message lpe* (local poll enable, active low). When U=0, S and P3-1 mean the same as the bit of the same name in the PPE message, and the I/O write operation (to the PPR) is the same as the receipt of the PPE message from the GPIB Controller. When U=1, S

Bit	Mnemonic	Description

and P3-1 do not carry any meaning, but they should be cleared.

3w S Status Bit Polarity

The S bit is used to indicate the polarity of the TLC local ist (individual status) message. If S=1, the status is 'in phase', meaning that if, during a Parallel Poll response, S=ist=1, and U=0, the TLC responds to the Parallel Poll by driving one of the eight GPIB DIO lines low (thus asserting it to a logic one). If S=1 and ist=0, the TLC does not drive the DIO line.

If S=0, the status is 'in reverse phase', meaning that if, during a Parallel Poll, ist=0, and U is 0, the TLC responds to the Parallel Poll by driving one of the eight GPIB DIO lines low. If S=0 and ist=1, the TLC does not drive the DIO line.

Refer to the description of AUXRB and the Set/Clear Parallel Poll Flag auxiliary commands for more information.

3w P3-P1 Parallel Poll Response

PPR bits 2 through 0, designated P3 to P1, contain an encoded version of the Parallel Poll Response. P3-P1 indicate which of the eight DIO lines is to be asserted during a Parallel Poll (equal to N-1). The GPIB-PC2A normally drives the GPIB DIO lines using three-state drivers. During Parallel Poll responses, however, the drivers automatically convert to open-collector mode, as required by IEEE-488. For example, if P3-P1=010 (binary), GPIB DIO line DIO3* is driven low (asserted) if the GPIB-PC2A is Parallel Polled (and S=ist).

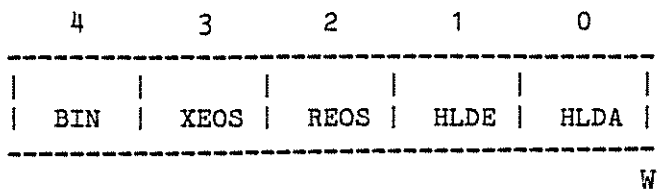
Some examples of configuring the Parallel Poll Register are as follows:

Written to the AUXMR 7	6	5	4	3	2	1	0	Result
0	1	1	1	0	0	0	0	Unconfigures PPR.
0	1	1	0	0	0	0	0	0 0 0 0 0 is written to the PPR. GPIB-PC2A participates in a Parallel Poll asserting the DIO1 line if ist is 0. Otherwise, it does not participate.
0	1	1	0	1	0	0	1	0 1 0 0 1 is written to the PPR. GPIB-PC2A participates in a Parallel Poll asserting the DIO2 line if ist is 1. Otherwise, it does not participate.

3.2.10.3 Auxiliary Register A (AUXRA)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

AUXMR Control Code: 100 (binary, bits 7 - 5)
Attributes: Write Only, Internal to TLC



Writing to Auxiliary Register A is done via the AUXMR. Writing the binary value 100 into the Control Code (CNT2-CNT0) and a bit pattern into the command code portion (COM4-COM0) of the AUXMR causes the the command code to be written to Auxiliary Register A. When the data is written to AUXRA, the bits are denoted by the mnemonics shown in the figure above. This five-bit code controls the data transfer messages holdoff and EOS/END.

Bit	Mnemonic	Description
4w	BIN	Binary

The BIN bit selects the length of the EOS message. Setting BIN causes the End of String Register (EOSR) to be treated as a full 8-bit byte. When BIN=0, the EOSR is treated as a 7-bit register (for ASCII characters) and only a 7-bit comparison is done with the data on the GPIB.

3w	XEOS	Transmit END with EOS

The XEOS bit permits or prohibits automatic transmission of the GPIB END message at the same time as the EOS message when the TLC is in TACS (Talker Active State). If XEOS is set and the byte in the CDOR matches the contents of the EOS register, the EOI line is sent true along with the data.

2w	REOS	End on EOS Received

The REOS bit permits or prohibits setting the END RX bit at reception of the EOS message when the TLC is in LACS (Listener Active State). If REOS is set and the byte in the DIR matches the byte in the EOS register, the END RX bit is set.

Bit	Mnemonic	Description
1-0w	HLDE	Holdoff on End
	HLDA	Holdoff on All

HLDE and HLDA together determine the GPIB data receiving mode. The four possible modes are as follows:

HLDE	HLDA	Data Receiving Mode
0	0	Normal handshake
0	1	RFD holdoff on All Data
1	0	RFD holdoff on End
1	1	Continuous

In Normal handshake mode, the local message rdy is generated when data is received from the GPIB. When the received data is read from the DIR, rdy is generated in ANRS (Acceptor Not Ready State), the RFD message is transmitted, and the GPIB handshake continues.

In RFD Holdoff on All Data mode, RFD is not sent true after data is received until the Finish Handshake auxiliary command is issued. Unlike normal handshake mode, the RFD Holdoff on All Data mode does not generate the rdy message even if the received data is read through the DIR (that is, the GPIB RFD message is not generated).

In RFD Holdoff on End mode, operation is the same as the RFD Holdoff on All Data mode, but only when the end of the data block is detected, i.e., the END message is received or, if REOS is set, the EOS character is received. Handshake holdoff is released by the Finish Handshake auxiliary command.

Bit	Mnemonic	Description

In continuous mode, the rdy message is generated when in ANRS (Acceptor Not Ready State) until the end of the data block is detected. A holdoff is generated at the end of a data block. The Finish Handshake auxiliary command must be issued to release the holdoff. The continuous mode is useful for monitoring the data block transfer without actually participating in the transfer (no data reception). In continuous mode, the DI bit (ISR1[0]r) is not set by the reception of a data byte.

3.2.10.4 Auxiliary Register B (AUXRB)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

AUXMR Control Code: 101 (binary, bits 7 - 5)
Attributes: Write Only, Internal to TLC

4	3	2	1	0
				CPT
ISS	INV	TRI	SPEOI	ENAB

W

Writing to AUXRB is done via the AUXMR. Writing the value 101 into the Control Code (CNT2-CNT0) and a bit pattern into the command code portion (COM4-COM0) of the AUXMR causes the command code to be written to Auxiliary Register B. When the data is written to AUXRB, the bits are denoted as shown in the figure above. This five-bit code affects several interface functions, as described in the following paragraphs.

Bit	Mnemonic	Description
4w	ISS	Individual Status Select

The ISS bit determines the value of the TLC individual status (ist) message. When ISS=1, ist becomes the same value as the TLC SRQS (Service Request State). (The TLC asserts the GPIB SRQ message when it is in SRQS). When ISS=0, ist takes on the value of the TLC Parallel Poll flag. The Parallel Poll flag is set and cleared using the Set Parallel Poll Flag and Clear Parallel Poll Flag auxiliary commands.

3w INV — Invert

The INV bit affects the polarity of the TLC INT pin. Setting INV causes the polarity of the INT pin on the TLC to be active low. As implemented on the GPIB-PC2A, INV should always be cleared (0) and should never be set (1) except for diagnostic purposes.

INV = 0 : INT pin is active high

INV = 1 : INT pin is active low

Bit	Mnemonic	Description
2w	TRI	Three-State Timing

The TRI bit determines the TLC GPIB Source Handshake timing. TRI may be set to enable high speed data transfers (T1 = high speed) when three-state GPIB drivers are used. (The GPIB-PC2A uses three-state GPIB drivers except during parallel poll responses, in which case the GPIB drivers automatically switch to open collector.) Setting TRI enables T1 (high speed) timing as T1 of the GPIB Source Handshake after transmission of the first byte. Clearing TRI sets the T1 timing to low speed in all cases.

1w SPEOI Send Serial Poll EOI

The SPEOI bit permits or prohibits the transmission of the END message in SPAS (Serial Poll Active State). If SPEOI is set, EOI is sent true when the TLC is in SPAS; otherwise, EOI is sent false in SPAS.

0w CPT ENAB Command Pass Through Enable

The CPT ENAB bit permits or prohibits the detection of undefined GPIB commands and permits or prohibits the setting of the CPT bit (ISR1[7]r) on receipt of an undefined command. When CPT ENAB is set, GPIB commands not recognized by the TLC can be handled by software.

3.2.10.5 Auxiliary Register E (AUXRE)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

AUXMR Control Code: 110 (binary, bits 7 - 5)
Attributes: Write Only, Internal to TLC

4	3	2	1	0
0	0	0	DHDC	DHDT
				W

Writing to AUXRE is done via the AUXMR. Writing the binary value 110 into the Control Code (CNT2-CNT0) and a bit pattern into the the lower five bits (COM4-COM0) of the AUXMR causes the two lowest order bits to be written to AUXRE. The two-bit code, DHDC and DHDT, determines how the TLC uses DAC holdoff.

Bit	Mnemonic	Description
1w	DHDC	DAC Holdoff on DCAS Setting DHDC enables DAC holdoff when the TLC enters DCAS (Device Clear Active State). Clearing DHDC disables DAC holdoff on DCAS. Issuing the Finish Handshake auxiliary command releases the holdoff.
0w	DHDT	DAC Holdoff on DTAS Setting DHDT enables DAC holdoff when the TLC enters DTAS (Device Trigger Active State). Clearing DHDT disables DAC holdoff on DTAS. Issuing the Finish Handshake auxiliary command releases the holdoff.

3.2.11 Address Register 0 (ADR0)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

Attributes: Read Only, Internal to TLC

7	6	5	4	3	2	1	0	R
X	DT0	DL0	AD5-0	AD4-0	AD3-0	AD2-0	AD1-0	

ADR0 reflects the internal GPIB address status of the TLC as configured using the ADMR. In addressing Mode 2, ADR0 indicates the address and enable bits for the primary GPIB address of the TLC. In dual primary addressing (Modes 1 and 3) ADR0 indicates the TLC major primary GPIB address. (Refer to the description of ADMR for information on addressing modes.)

Bit	Mnemonic	Description
7r		Not used. May read as 0 or 1.
6r	DT0	Disable Talker 0 If DT0 is set, it indicates that the Mode 2 primary (or Mode 1 and 3 major) Talker is not enabled; i.e., the TLC does not respond to a GPIB talk address matching AD5-0 to AD1-0. If DT0=0, the TLC responds to a GPIB talk address matching AD5-0 to AD1-0.
5r	DL0	Disable Listener 0 If DL0 is set, it indicates that the Mode 2 primary (or Mode 1 and 3 major) Listener is not enabled; i.e., the TLC does not respond to a GPIB listen address matching bits AD5-0 to AD1-0. If DL0=0, the TLC responds to a GPIB listen address matching AD5-0 to AD1-0.
4-0r	AD5-0 thru AD1-0	Mode 2 Primary GPIB Address These are the lower bits of the TLC GPIB primary (or major) address. (The primary talk address is formed by adding octal 100 to bits AD5-0 through AD1-0, while the listen address is formed by adding octal 40.)

3.2.12 Address Register (ADR)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

Attributes: Write Only, Internal to TLC

7	6	5	4	3	2	1	0
ARS	DT	DL	AD5	AD4	AD3	AD2	AD1
							W

ADR is used to load the internal registers ADR0 and ADR1. Both ADR0 and ADR1 must be loaded for all addressing modes.

Bit	Mnemonic	Description
7w	ARS	Address Register Select ARS is 0 or 1 to select whether the seven lower-order bits of ADR are to be loaded into internal registers ADR0 or ADR1, respectively.
6w	DT	Disable Talker DT should be set if recognition of the GPIB talk address formed from AD5 through AD1 (ADR[4-0]w) is not to be enabled.
5w	DL	Disable Listener DL should be set if recognition of the GPIB listen address formed from AD5 through AD1 is not to be enabled.
4-0w	AD5-1	Address These bits specify the five low-order bits of the GPIB address that is to be recognized by the TLC. (The corresponding GPIB talk address is formed by adding octal 100 to AD5-AD1, while the corresponding GPIB listen address is formed by adding octal 40.) The value written to AD5-AD1 should not all be ones, since the corresponding Talk and listen addresses would conflict with the GPIB Untalk (UNT) and Unlisten (UNL) commands.

3.2.13 Address Register 1 (ADR1)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

Attributes: Read Only, Internal to TLC

7	6	5	4	3	2	1	0	R
EOI	DT1	DL1	AD5-1	AD4-1	AD3-1	AD2-1	AD1-1	

ADR1 indicates the status of the GPIB address and enable bits for the secondary address of the TLC if Mode 2 addressing is used, or the minor primary address of the TLC if dual primary addressing is used (Modes 1 and 3). If Mode 1 addressing is used and only a single primary address is needed, then both the talk and listen addresses should be disabled in this register. If Mode 2 addressing is used then the talk and listen disable bits in this register should match those in ADR0.

Bit	Mnemonic	Description
7r	EOI	End or Identify

EOI indicates the value of the GPIB EOI line latched when a data byte is received by the TLC GPIB AH (Acceptor Handshake) function. If EOI=1, the EOI line was asserted with the received byte. EOI is cleared by power on reset, or by using the Chip Reset auxiliary command.

6r DT1 Disable Talker 1

If DT1 is set, it indicates that the Mode 2 secondary (or Mode 1 and 3 minor) Talker is not enabled; i.e., the TLC does not respond to a GPIB secondary address (or minor primary talk address) matching AD5-1 to AD1-1. If DT1=0, the secondary address is checked only if the TLC received its primary talk address.

5r DL1 Disable Listener 1

If DL1 is set, it indicates that the Mode 2 secondary (or Mode 1 and 3 minor) Listener is not enabled; i.e., the TLC does not respond to a GPIB secondary address (or minor primary listen address) matching AD5-1 to AD1-1. If DL1=0, the secondary address is checked only if the TLC received its primary listen address.

Bit	Mnemonic	Description
4-0r	AD5-1 thru AD1-1	Mode 2 Secondary GPIB Address

These are the lower bits of the TLC GPIB secondary (or minor) address. (The secondary address is formed by adding octal 140 to AD5-1 to AD1-1)

3.2.14 End of String Register (EOSR)

I/O Channel Address: Block1 - 16E1; Block2 - 36E1
Block3 - 56E1; Block4 - 76E1

Attributes: Write Only, Internal to TLC

7	6	5	4	3	2	1	0
EOS7	EOS6	EOS5	EOS4	EOS3	EOS2	EOS1	EOS0
							W

The EOSR holds the byte used by the TLC to detect the end of a GPIB data block transfer. A 7- or 8-bit byte (ASCII or binary) may be placed in the EOSR to be used in detecting the end of a block of data. The length of the EOS byte to be used in the comparison is selected by the BIN bit in Auxiliary Register A, AUXRA[4]w.

If the TLC is a Listener and bit REOS of AUXRA is set, then the END bit is set in ISR1 whenever the byte in the DIR matches the EOS register. If the TLC is a Talker and the data is being transmitted, and bit XEOS of AUXRA is set, the END message (GPIB EOI* line asserted low) is sent along with the data byte whenever the contents of the CDOR matches the EOS register.

Bit	Mnemonic	Description
7-0w	EOS7-EOS0	End of String Byte

3.3 CLOCK REGISTERS

All programming of the MM58167A takes place through two I/O registers, CCR and CDR. Clock registers are addressed indirectly through the CDR using the clock register address programmed in the CCR. Both the CCR and the CDR are described in more detail in the following paragraphs.

3.3.1 Clock Control Register (CCR)

I/O Channel Address: Block1 - 96E1; Block2 - B6E1
Block3 - D6E1; Block4 - F6E1

Attributes: Write Only

7	6	5	4	3	2	1	0
CIE	SIE	CLK IE	CKA4	CKA3	CKA2	CKA1	CKA0
							W

The CCR is used to control access to the MM58167A internal registers and also to control the clock interrupts.

Bit	Mnemonic	Description
7w	CIE	Clock Interrupt Enable If CIE is set, interrupt requests from the clock circuitry will be enabled into the special shared interrupt circuitry used by the GPIB chip. CIE is cleared by power on reset.
6w	SIE	Separate Interrupt Enable If SIE is set, interrupt requests from the clock circuitry will be enabled onto the PC IRQ bus lines using a bus driver unique from the one used by the shared interrupt circuitry. SIE is cleared by power on reset.
5w	CLK IE	Clock Interrupt Enable CLK IE is a master interrupt enable for the clock circuitry. When CLK IE is cleared, the clock circuitry is prohibited from generating any interrupts. CLK IE overrides the CIE and SIE bits, and must be set for CIE and/or SIE to be effective. CLK IE is cleared by power on reset.

Bit	Mnemonic	Description
4-0w	CKA4-0	Clock Address Lines

The clock address lines determine which internal clock register is accessed by an I/O read or write to the CDR. All five bits are cleared by power on reset.

CKA4	CKA3	CKA2	CKA1	CKA0	FUNCTION
0	0	0	0	0	Counter - 1/10,000 of seconds
0	0	0	0	1	Counter - 1/100 & 1/10 of sec.
0	0	0	1	0	Counter - Seconds
0	0	0	1	1	Counter - Minutes
0	0	1	0	0	Counter - Hours
0	0	1	0	1	Counter - Day of Week
0	0	1	1	0	Counter - Day of Month
0	0	1	1	1	Counter - Month
0	1	0	0	0	RAM - 1/10,000 of seconds
0	1	0	0	1	RAM - 1/100 & 1/10 of seconds
0	1	0	1	0	RAM - Seconds
0	1	0	1	1	RAM - Minutes
0	1	1	0	0	RAM - Hours
0	1	1	0	1	RAM - Day of Week
0	1	1	1	0	RAM - Day of Month
0	1	1	1	1	RAM - Month
1	0	0	0	0	Interrupt Status Register
1	0	0	0	1	Interrupt Control Register
1	0	0	1	0	Counters Reset
1	0	0	1	1	RAM Reset
1	0	1	0	0	Status Bit
1	0	1	0	1	GO Command
1	0	1	1	0	Standby Interrupt
1	1	1	1	1	Test Mode

3.3.2 Clock Data Register (CDR)

I/O Channel Address: Block1 - 9AE1; Block2 - BAE1
Block3 - DAE1; Block4 - FAE1

Attributes: Read and Write

7	6	5	4	3	2	1	0	R
CD7	CD6	CD5	CD4	CD3	CD2	CD1	CD0	W

Bit	Mnemonic	Description
7-0	CD7-0	Clock Data Byte

This register is used to access clock data from the MM58167A register programmed in the CCR.

Section Four

Programming Considerations

This section explains important considerations for programming the GPIB-PC2A.

4.1 INITIALIZATION

On power up, the IBM PC issues a bus reset by driving the RESET DRV line high, which drives the RESET line high and the RESET* line low on the GPIB-PC2A. RESET clears the uPD7210 and RESET* clears the System Controller bit, disables interrupts from the clock, and causes the selected interrupt request line(s) and DMA request line to be tri-stated.

RESET resets the NEC uPD7210 Talker/Listener/Controller (TLC) integrated circuit as follows:

* Local message pon is set and the interface functions are placed in their idle states (SIDS, AIDS, TIDS, SPIS, TPIS, LIDS, LPIS, NPRS, LOCS, PPIS, PUCS, CIDS, SRIS, SIIS).

* All bits of the Serial Poll Mode Register (SPMR) are cleared.

* EOI bit is cleared.

* All bits of the Auxiliary Registers A, B, and E (AUXRA, AUXRB, and AUXRE) are cleared.

* The Parallel Poll flag and request system control (rsc) local message are cleared.

* The Internal Clock Register (ICR) is set to a count of 8.

* The TRM0 and TRM1 bits in the Address Mode Register (ADMR) are cleared.

All other register contents should be considered as undefined while RESET DRV is asserted and after RESET DRV has been cleared. All Auxiliary Mode Register commands are cleared and cannot be executed. All other GPIB-PC2A registers can be programmed while the GPIB-PC2A internal signal pon is set. When pon is released or cleared (by issuing an Immediate Execute pon auxiliary command to the GPIB-PC2A), the interface functions are released from the pon state and the auxiliary commands can be executed.

A typical programmed initialization sequence for the GPIB-PC2A might include the following steps:

1. Write the Chip Reset command to the auxiliary command register to place the GPIB-PC2A in a known, quiescent state.

2. Set or clear the desired interrupt enable bits in Interrupt Mask Register 1 (IMR1) and Interrupt Mask Register 2 (IMR2).

3. Load the GPIB-PC2A primary GPIB address in Address Register 0 (ADR0) and Address Register 1 (ADR1).

4. Enable or disable the GPIB Talker and Listener functions and addressing mode using the ADMR.

5. Set the TRM0 and TRM1 bits in the ADMR.

6. Load the Serial Poll response in the SPMR.

7. Load the Parallel Poll response in the Parallel Poll Register (PPR) if local configuration is used. If using remote configuration, clear the PPR.

8. Clear pon by issuing the Immediate Execute pon auxiliary command.

9. Execute the desired auxiliary commands.

4.2 THE GPIB-PC2A AS GPIB CONTROLLER

The GPIB-PC2A Controller function is generally in one of two modes: idle or in-charge. When in-charge, the Controller function is either active (asserting ATN) or standby (not asserting ATN). The following paragraphs discuss the various transitions between these modes.

4.2.1 Becoming Controller-In-Charge (CIC) and Active Controller

The GPIB-PC2A can become CIC either by being the System Controller and taking control (by issuing the Set IFC auxiliary command which also sets the System Controller bit) or by being passed control of the GPIB from the current Active Controller.

The GPIB-PC2A is only capable of driving the GPIB IFC and REN lines (which thus allows the GPIB-PC2A to function as GPIB System Controller) when the System Controller bit is set. To take control, issue the Set IFC auxiliary command, wait for a minimum of 100 microseconds, and then issue the Clear IFC auxiliary command. The ensuing GPIB IFC message initializes the GPIB interface functions of all devices on the bus. As soon as any existing CIC goes to idle (unasserting ATN if it was active) the GPIB-PC2A becomes CIC and Active Controller and asserts the GPIB ATN line.

The System Contoller bit is cleared by issuing the Chip Reset or Disable System Control auxiliary command. Another Active Controller passes control to the GPIB-PC2A by sending the GPIB-PC2A GPIB talk address (MTA) followed by the GPIB TCT (Take Control) message. The GPIB-PC2A, upon receiving these two messages (MTA and TCT), automatically becomes CIC when ATN is unasserted. The exact sequence of events is as follows:

* MTA (My Talk Address) is received by the GPIB-PC2A, and it enters into TADS (Talker Addressed State); this operation can be transparent to a program. The TA bit in the ADSR is set when the GPIB-PC2A receives its GPIB Talk address.

* Next, the GPIB TCT message is received by the GPIB-PC2A.

* The current Active Controller sees the completed handshake, goes to idle and unasserts ATN.

* As soon as the ATN line on the GPIB is unasserted, the GPIB-PC2A automatically becomes CIC and asserts ATN.

As soon as the GPIB-PC2A becomes CIC, the CIC bit in the Address Status Register (ADSR) is set, and the Command Output bit (CO) in Interrupt Status Register 2 (ISR2) is set. Using these two bits, the program can unambiguously determine that the GPIB-PC2A is the GPIB Active Controller and can send remote messages.

4.2.2 Sending Remote Multiline Messages (Commands)

The GPIB-PC2A sends commands as Active Controller simply by writing to the Command/Data Out Register (CDOR) in response to the CO status bit in ISR2.

The GPIB-PC2A can address itself to be both Talker and Listener in address modes 1 or 2; that is, the TLC recognizes its address when it sends or receives it.

4.2.3 Going from Active to Standby Controller

If the GPIB-PC2A is GPIB Active Controller, the Controller Standby State (CSBS) is entered upon reception of the Go To Standby auxiliary command. The ATN line is unasserted as soon as the GPIB-PC2A enters CSBS. Even though the GPIB-PC2A GPIB Controller state machine is in standby, the CIC bit in the ADSR is still set. Do not issue the Go To Standby auxiliary command unless the CO bit in ISR2 is set.

There are three cases to consider when going to standby:

* Case 1. The GPIB-PC2A is to become the GPIB Talker when ATN is unasserted. To do this, wait for CO to be set, send the GPIB-PC2A GPIB Talk Address (MTA), wait for CO to be set again, and then issue the Go To Standby auxiliary command.

* Case 2. The GPIB-PC2A is to become a GPIB Listener when ATN is unasserted. To do this, wait for CO to be set, issue the Listen auxiliary command, wait for CO to be set again, and then issue the Go To Standby auxiliary command.

* Case 3. The GPIB-PC2A is to be neither GPIB Talker or Listener. In this case, issue the Listen auxiliary command and set the Holdoff on End (HLDA) and Holdoff on All (HLDA) bits in AUXRA before going to standby. Once this mode is enabled, the GPIB-PC2A participates in the GPIB handshake without setting the DI (Data In) bit. A handshake holdoff occurs as described in paragraph 3.2.10.3. When holdoff occurs, the GPIB-PC2A can take control synchronously. This means that the Talker must finish its transmission with the END or EOS message. It can then take control synchronously when necessary.

4.2.4 Going from Standby to Active Controller

The manner in which the GPIB-PC2A resumes GPIB Active Control depends on how it went to standby. Consider the three cases:

* Case 1. The GPIB-PC2A, as a Talker, takes control upon receipt of the Take Control Asynchronously auxiliary command. Do not issue the Take Control Asynchronously auxiliary command until there are no more bytes to send and the DO bit is set.

* Case 2. The GPIB-PC2A, as a Listener, takes control upon receipt of the Take Control Synchronously auxiliary command. If programmed I/O is used, the Take Control Synchronously auxiliary command should be issued between seeing a DI status bit and reading the last byte from the DIR.

* Case 3. The GPIB-PC2A as neither Talker nor Listener, takes control synchronously with the Take Control Synchronously auxiliary command after detecting the END RX bit set in ISR1. This indicates that a holdoff is in progress.

When the Take Control Synchronously auxiliary command is used, the GPIB-PC2A takes control of the GPIB only at the end of a data transfer. This implies that one transfer must follow or be in progress when the Take Control Synchronously auxiliary command is issued. If this is not the case, the Take Control Asynchronously auxiliary command must be used. Of course, the Take Control Asynchronously auxiliary command may be used in place of the Take Control Synchronously auxiliary command when the possibility of disrupting an in-progress GPIB handshake (before all GPIB Listeners have accepted the data byte) is acceptable.

In Cases 2 and 3 above, the END IE bit in IMR1 can also be set to indicate to the program that the GPIB-PC2A (functioning as a GPIB Listener) has received its last byte.

In all cases, a CO status indicates that the GPIB-PC2A is now Active Controller.

4.2.5 Going from Active to Idle Controller

Going from Active to Idle GPIB Controller, also known as passing control, requires that the GPIB-PC2A be the Active Controller initially (in order to send the necessary GPIB command messages). After the GPIB-PC2A has become the GPIB Active Controller, the sequence of events required to pass control are as follows:

* Write the GPIB Talk address of the device being passed control to the CDOR.

* In response to the next CO status, write the GPIB TCT (Take Control) message to the CDOR.

* As soon as the TCT command message is accepted by all devices on the GPIB, the GPIB-PC2A automatically unasserts ATN and the new Controller asserts ATN.

4.3 THE GPIB-PC2A AS GPIB TALKER/LISTENER

The GPIB-PC2A may be either GPIB Talker or Listener, but not both simultaneously. Either function is deactivated automatically if the other is activated. The TA, LA, and ATN* bits in the ADSR together indicate the specific state of the GPIB-PC2A:

ATN*	TA	LA	
0	1	0	Addressed Talker -- cannot send data
1	1	0	Active Talker -- can send data
0	0	1	Addressed Listener -- cannot receive data
1	0	1	Active Listener -- can receive data

The status bits ADSC (Address Status Change), CO (Command Output), APT (Address Pass Through), DO (Data Out), and DI (Data In) are used to prompt the program (with an interrupt request if enabled) when a change of state occurs.

The following paragraphs discuss several aspects of data transfers.

4.3.1 Programmed Implementation of Talker/Listener

When there is no Controller in the GPIB system, the ton and lon address modes (refer to the description of the ADMR) are used to activate the GPIB-PC2A GPIB Talker and Listener functions. If used, ton or lon should be set during GPIB-PC2A initialization.

When the GPIB-PC2A is GPIB Active Controller, the Listen and Local Unlisten programmed auxiliary commands are used to activate and deactivate the GPIB-PC2A GPIB Listener function.

4.3.2 Addressed Implementation of Talker/Listener

When the GPIB-PC2A is the GPIB Active Controller, it can address itself to talk by sending its own GPIB Talk address (MTA) using the CO bit and the CDOR. When there is another device on the GPIB acting as Controller, the GPIB-PC2A is addressed with GPIB command messages to become a Talker or Listener.

4.3.2.1 Address Mode 1

If the GPIB-PC2A ADMR has been configured for Address Mode 1, the GPIB-PC2A responds to the reception of two primary GPIB addresses: major and minor. Upon receipt of its major or minor MTA or its major or minor MLA from the GPIB Active Controller, the GPIB-PC2A is addressed as Talker or Listener. If the GPIB-PC2A has received its GPIB Talk Address, the TA bit in the ADSR is set, the ADSC bit in ISR2 is set, and the DO bit in ISR1 is set. If the GPIB-PC2A has received its GPIB Listen address, the LA bit in the ADSR is set, the ADSC bit in ISR2 is set, and the

DI bit in ISR1 is set when the first GPIB data byte is received.

4.3.2.2 Address Mode 2

Address Mode 2 is used when Talker Extended (TE) or Listener Extended (LE) functions are to be used. TE and LE functions require receipt of two addresses (primary and secondary) before setting TA or LA. The GPIB-PC2A GPIB primary address is specified by the byte written to ADR0. The secondary address is specified by the byte written to ADR1. Upon receipt of both the primary and secondary GPIB addresses the GPIB-PC2A becomes an addressed Talker or Listener. If the GPIB-PC2A has received its primary GPIB Talk address, the TPAS (Talker Primary Addressed State) bit in the ADSR is set. If the GPIB-PC2A receives its secondary GPIB talk address before receiving another GPIB Primary Command Group (PCG) message that is not its MTA, the TA bit in ADSR, the ADSC bit in ISR2, and the DO bit in ISR1 are set. If the GPIB-PC2A has received its Primary GPIB Listen address, the LPAS bit in the ADSR is set. If the GPIB-PC2A receives its secondary GPIB listen address before receiving another GPIB Primary Command Group (PCG) message that is not its MLA, the LA bit in the ADSR is set, the ADSC bit in ISR2 is set, and the DI bit in ISR1 is set when the first GPIB data byte is received. The MJMN bit in the ADSR indicates whether the address status refers to the major or minor address.

4.3.2.3 Address Mode 3

Address Mode 3, like Address Mode 2, is used to implement Extended GPIB talk and listen address recognition. However, unlike Address Mode 2, Address Mode 3 provides for both major and minor primary addresses, and the user's program must identify the secondary address by reading the CPTR. The sequence of events for proper operation using Address Mode 3 is as follows:

* During initialization of the GPIB-PC2A, enable Address Mode 3 (and optionally set the APT IE bit in IMR1 to enable an interrupt request on receipt of a secondary GPIB address). Write the GPIB-PC2A major GPIB primary address to ADR0 and the GPIB-PC2A minor GPIB primary address to ADR1.

* Receipt of the GPIB-PC2A major or minor primary GPIB Talk Address (MTA) or major or minor primary GPIB Listen Address (MLA) sets TPAS or LPAS, indicating that the primary address has been received.

* If the next GPIB command following the primary address is a secondary address, the APT bit is set and a DAC handshake holdoff is activated (the GPIB DAC message is held false).

* In response to APT, the program must:

  - Determine whether the command just received is a listen, talk, major, or minor address by reading the

LPAS, TPAS, and MJMN bits of the ADSR; and

- Read the secondary address in the CPTR and determine whether or not it is the address of the GPIB-PC2A.

* If it is not the GPIB-PC2A address, issue the Non-Valid auxiliary command. If it is the GPIB-PC2A address, issue the Valid auxiliary command.

* When the Valid auxiliary command is issued, the GPIB-PC2A assumes that the My Secondary Address (MSA) message has been received, which causes:

  - The LA bit to be set and the TA bit to be cleared (LADS=TIDS=1) if LPAS was set, or the TA bit to be set and the LA bit to be cleared (TADS=LIDS=1) if TPAS was set; and

  - The GPIB DAC message to be sent true, and the GPIB handshake is finished.

* When the Non-Valid auxiliary command is issued, the GPIB-PC2A assumes that the Other Secondary Address (OSA) message has been received, which causes:

  - The GPIB-PC2A Talker or Listener function to go to its idle state (TIDS=1 or LIDS=1) if the either the TPAS or LPAS bit was set; and

  - The GPIB DAC message to be sent true, and the handshake is finished.

Until a GPIB Primary Command Group (PCG) message is received (i.e., as long as the subsequent messages are secondary addresses), the APT bit is set and a DAC holdoff is in effect each time a GPIB secondary address is received. In this way, the GPIB CIC can address several devices having the same primary address without repeating the primary address each time. If a PCG message is received before a secondary address is received, the TPAS and LPAS bits are cleared.

4.4 SENDING/RECEIVING MESSAGES

When the GPIB-PC2A is a GPIB Talker or Listener, data (device dependent messages) can be sent or received using DMA or programmed I/O. Setting up the GPIB-PC2A for DMA is described in 4.8.

4.4.1 Sending and Receiving Data

To send data, wait until the GPIB-PC2A has been programmed or addressed to talk and the CDOR is empty. When this occurs, the DO bit in the ISR1 is set, indicating that it is safe to write a byte to the CDOR. The DO bit is set again once the byte has been received by all Listeners. If DMA is enabled, (by setting the DMAO bit), the GPIB-PC2A requests a DMA transfer from the PC by driving DRQ high under the same conditions that the DO bit is set. If the PC completes the DMA transfer, the GPIB-PC2A drives DRQ low. and a byte is transferred from PC memory and sent to the GPIB. When all listeners have received the byte, the GPIB-PC2A again requests a DMA transfer.

To receive data, wait until the GPIB-PC2A has been programmed or addressed to listen and the CDOR is empty. When this occurs, the DI bit in ISR1 is set, indicating that the GPIB Talker has written a byte to the DIR. Once that byte has been read, the DI bit is set again when a new byte is received from the Talker. If DMA is enabled, (by setting the DMAI bit), the GPIB-PC2A requests a DMA transfer from the PC by driving DRQ high under the same conditions that the DI bit is set. If the PC completes the DMA transfer, the GPIB-PC2A drives DRQ low. and the byte received from the GPIB Talker is transferred to PC memory. When a new byte has been received from the Talker, the GPIB-PC2A again requests a DMA transfer.

Determining when the CDOR is empty or the DIR is full can be done by polling ISR1 until the DO or DI status first appears or by allowing a program interrupt or DMA transfer to occur on the respective event. Remember, however, that the status bits and interrupt signal are cleared when the register is read, so the absence of a true DO or DI status does not indicate that the CDOR is still full or that the DIR is still empty.

4.4.2 Sending END or EOS

The GPIB END message is sent by issuing the Send EOI auxiliary command just before writing the last data byte to the CDOR. The GPIB EOS message is sent simply by making the last byte the EOS code.

4.4.3 Terminating on END or EOS

The END status bit or interrupt is used to inform the program of the occurrence of an END message or an EOS message.

4.5 SERIAL POLLS

Serial polls allow the Controller-In-Charge to obtain detailed status information on each device configured for responding.

4.5.1 Conducting Serial Polls

The GPIB-PC2A, as CIC, serially polls other devices as described in the IEEE-488 specification. From the programming point of view, the GPIB-PC2A must first become Active Controller to send the addressing and enabling commands to the device being polled, make itself a GPIB Listener by issuing the Listen auxiliary command, and then go to standby with the Go To Standby auxiliary command in order to read the status byte.

4.5.2 Responding to a Serial Poll

The CIC can conduct Serial Polls to determine which device is asserting the GPIB SRQ signal to request service.

Before requesting the service, the recommended practice is to check the PEND bit of the SPSR to be certain it is zero, indicating that the GPIB-PC2A is not presently in the middle of a Serial Poll (SPAS=0). If PEND=0 write the desired Status Byte (STB) into the SPMR with the rsv bit set. At that time, PEND is set and remains set until the Serial Poll completes.

Once rsv is set, the GPIB-PC2A waits until any current Serial Poll is complete and then asserts the GPIB SRQ signal. In response to that signal, the CIC starts the poll addressing the GPIB-PC2A to talk, and when the CIC unasserts ATN, the GPIB-PC2A unasserts SRQ and tranfers the STB message onto the GPIB data bus with DIO7, the RSQ signal, asserted.

While the Serial Poll is in progress (SPAS=1), the CIC normally reads the STB only once but may read it any number of times provided that it asserts ATN between each one byte read. However, RSQ is set only during the first read; after the first read, rsv also is cleared. PEND is cleared when the CIC asserts ATN to terminate the poll.

The GPIB EOI line is asserted along with the status byte (i.e., the END message is sent) during the serial poll if bit SPEOI of AUXRB is set.

4.6 PARALLEL POLLS

Parallel Polls are used by the GPIB Active Controller to check the status of several devices simultaneously. The meaning of the status returned by the devices being polled is device dependent, but there are two general ways in which Parallel Polls are useful.

* When the GPIB Controller sees SRQ asserted in a system with several devices, it can quickly determine which one needs to be serially polled using, usually, only one Parallel Poll.

* In systems in which the Controller response time requirement to service a device is low and the number of devices is small, Parallel Polls can replace Serial Polls entirely, provided that the Controller polls frequently.

Although the Controller can obtain a Parallel Poll response quickly and at any time, there can be considerable front end overhead during initialization to configure the devices to respond appropriately. This is contrasted with Serial Polls, where the overhead, in the form of addressing and enabling command messages, occurs with each poll.

4.6.1 Conducting a Parallel Poll

The GPIB-PC2A as Active Controller has the capability to conduct a Parallel Poll. When the Execute Parallel Poll auxiliary command is issued and the GPIB-PC2A internal local message rpp is set, the Parallel Poll is executed (the GPIB message IDY is sent true) as soon as the GPIB-PC2A Controller interface function is placed in the proper state (CAWS or CACS). The Parallel Poll Response (PPR) is automatically read from the GPIB DIO lines into the CPTR and the rpp local message is cleared. A program can determine that the Parallel Poll operation is complete based on the condition of CO (CO=1 when the poll is complete). The response can be obtained by reading the contents of the CPTR. The response is held in the CPTR until a GPIB command is transmitted or the GPIB-PC2A Controller function becomes inactive.

In response to IDY, each device participating in the Parallel Poll drives one and only one GPIB DIO line (its Parallel Poll response or PPRn) active true or passive false, while it drives the other lines passive false.

Since there are eight data lines, and for each line there can be two responses (true or false), there are 16 possible responses. The line a device uses and how that device drives the line depends on how it was configured and whether its local individual status message (ist) is one or zero. Thus, each device on the GPIB can be configured to drive its assigned DIO

line true if ist=1 and to drive the DIO line false if ist=0; or it can be configured to do exactly the opposite: that is, to drive the DIO line true if ist=0 and false if ist=1. (The meaning of the value of ist, whether one or zero, is system or device dependent.)

Because the data lines are driven open collector during Parallel Polls, more than one device can respond on each line. The device or devices asserting the line true overrides any device asserting the line false. Obviously, the Controller must know in advance whether a true response means the local ist message of the device is one or zero. To do this, the device must be configured to respond in the desired way, and two methods of accomplishing this can be used.

Local configuration (Parallel Poll function subset PP2) involves assigning a response line and sense from the device side in a manner similar to assigning the device GPIB address. Thus, one device might be assigned to respond with remote message PPR1 (driving DIO1), while a second device is assigned to respond with remote message PPR3 (driving DIO3); both positive (i.e., true response if ist=1). Local configuration is static in the sense that it does not change after the system is integrated (system installed and configured).

Remote configuration (Parallel Poll function subset PP1) involves the dynamic assigning of response line and sense to devices on the GPIB. This is accomplished using PPE (Parallel Poll Enable) and PPD (Parallel Poll Disable) commands, which are issued by the Active Controller. Following is the recommended sequence for remotely configuring devices:

* Become Active Controller.

* Send the GPIB UNL (Unlisten) message to unaddress all GPIB Listeners.

* Send the Listen address of the first device to be configured.

* Send the GPIB PPC message followed by the PPE message for that device.

* Repeat from the 2nd step (UNL) for each additional device.

The same procedure should be followed to disable polling with PPD, e.g., when changing responses during reconfiguration.

4.6.2 Responding To a Parallel Poll

Before the GPIB-PC2A can be polled by the CIC, the GPIB-PC2A must be configured either locally by the user program at initialization time, or remotely by the CIC. Configuration involves the following:

* enabling the GPIB-PC2A to participate in polls,

* selecting the sense or polarity of the response, and

* selecting the GPIB data line on which the response is asserted when the CIC issues the IDY message.

With remote configuration (PP1), the GPIB-PC2A interprets the configuration commands received from the CIC without any software assistance or interpretation from the user program. With local configuration (PP2), the three actions listed above must be explicitly handled in the software by writing the appropriate values to the U, S, and P3 to P1 bits of the PPR. Refer to the PPR register description for more information.

Once the PPR is configured, all that remains for the user program is to determine the source and value of the local individual status (ist) message. If the ISS bit in the AUXRB is zero, ist is set and cleared via the Set and Clear Parallel Poll auxiliary commands. If ISS is one, ist is set if the GPIB-PC2A's Service Request function is in the Service Request State (SRQS) and the GPIB-PC2A is asserting the GPIB SRQ signal line and cleared otherwise. Consequently, setting ISS ties the Parallel Poll function to the Service Request function and also to the Serial Poll process.

The particular response sent by the GPIB-PC2A duing a Parallel Poll is determined by the value of ist and the configuration of the GPIB-PC2A. The value of ist and the actual configuration must be decided by the GPIB system integrator. The response can be changed dynamically during program execution by changing the value of ist and, when remote configuration is used, by reconfiguration.

4.7 INTERRUPTS

Interrupts must be enabled via hardware jumpers to any of the six available IRQ lines on the IBM PC I/O Channel. Interrupts from the TLC and clock can share an interrupt line by setting the common interrupt enable bit (CIE) in the CCR. Interrupts from the TLC and Clock can also have their own separate interrupt lines by setting the separate interrupt enable bit (SIE) in the CCR and enabling each interrupt with separate hardware jumpers to the respective IRQ lines. If the TLC and Clock are configured to interrupt on the same line, the clock will be using the special shared interrupt logic provided on the GPIB-PC2A and any interrupt handling routine for the clock must follow the guidlines set below.

GPIB interrupt requests from the TLC are enabled using the IE bits in IMR1 and IMR2. Interrupt requests from the clock are enabled by using the IE bits in the clock interrupt control register.

The DMA Terminal Count (DMA TC) interrupt is implemented external to the 7210 TLC and is enabled whenever either the DMAO or DMAI bit is set and at least one of the IE bits of the TLC is set. The DMA TC Interrupt is further explained in section 4.8.

The selected IRQ line is driven by the GPIB-PC2A whenever at least one of the IE bits is set, otherwise the selected IRQ line is tri-stated.

Once asserted, the interrupt request line remains asserted until the corresponding status register is read.

The DMA Terminal Count interrupt has no corresponding status register or bit but is cleared when ISR2 is read.

Interrupts are arbitrated by the 8259A Interrupt Controller Chip located on the IBM PC System Card. The 8259A should never be enabled to respond to an interrupt request from the GPIB-PC2A when no interrupts are enabled on the GPIB-PC2A. This tri-states the GPIB-PC2A IRQ line. A tri-stated IRQ line may look like an interrupt request to the 8259A.

An Interrupt Handler routine for the GPIB-PC2A must perform the following steps:

* Read ISR2 to see whether the INT bit is set, confirming that the GPIB-PC2A has indeed issued an interrupt. If the DMA TC interrupt has been enabled, the INT bit will not be set when it occurs, however the DMA Controller will indicate whether or not it has reached terminal count for the GPIB-PC2A DMA Channel.

* Read ISR1. Reading ISR2 and ISR1 will clear an interrupt caused by any of the 14 possible conditions.

* Read the clock interrupt status register to determine if any condition of the clock caused the interrupt. Reading this register will clear any interrupt caused by the clock.

* Write the End Of Interrupt (EOI) command to the 8259A Interrupt Controller.

* Write to I/O address 2FX where X is the interrupt level being used by the GPIB-PC2A. Writing to 2FX reenables interrupts on the GPIB-PC2A. If the Clock caused the interrupt and is not using the Common Interrupt Enable (CIE) logic, this step is not necessary.

In a system where several GPIB-PC2As share the same interrupt level, the same steps should be used in an interrupt handler routine except that each of the boards should be polled until one is found that caused the interrupt. When 2FX is written to, any pending interrupt from another GPIB-PC2A will be allowed to occur, and that GPIB-PC2A may then be serviced.

To emphasize once more, the status bits in ISR1 or ISR2 are all automatically cleared when the register is read, even if the conditions are still true. If two conditions are true at the same time (i.e., more than one bit in ISR1 or ISR2 is set), a software copy of the register should be maintained if the program is going to analyze the conditions one at a time.

4.7.1 Programming the 8259A Interrupt Controller

Programming information for the 8259A Interrupt Controller Chip may be found in the Intel Component Data Catalog and in the IBM PC Technical Reference Manual. The following consideration should be taken:

* The 8259A is initialized upon power-up by the IBM PC Bios program located in ROM on the system board. Bios initializes the 8259A in a way that is required for the IBM PC to operate correctly. Software written for handling GPIB-PC2A interrupts must in no way change the overall configuration of the 8259A and commands written to the 8259A should only affect the selected GPIB-PC2A IRQ line and no other.

The manner in which the 8259A is configured and used by Bios may be found in the listing of the Bios program in Appendix A of the IBM PC Technical Reference.

4.8 DMA TRANSFERS

DMA transfers must be enabled via two hardware jumpers to one of the three available pairs of DMA transfer lines on the IBM PC I/O Channel

DMA requests from the TLC are enabled using the DMAO and DMAI bits in IMR2. The TLC generates a DMA request under the same conditions that the DO and DI bits in ISR1 are set. A DMA request indicates that the TLC either requires a byte to be written to the CDOR or requires a byte to be read from the DIR. The DMA request signal (DRQ) is cleared by a low on the DMA Acknowledge line (DACK*).

The selected DRQ line is driven and the DACK line enabled by the GPIB-PC2A whenever the DMAO or the DMAI bit is set in IMR2, otherwise the selected DRQ line is tri-stated, and DACK line is disabled.

DMA transfers are arbitrated by the 8237A-5 DMA Controller Chip located on the IBM PC System Card. The 8237A-5 should never be enabled to respond to a DMA request from the GPIB-PC2A when neither the DMAO nor DMAI bit is set in the TLC. This tri-states the GPIB-PC2A DRQ line. A tri-stated DRQ line may look like a DMA request from the GPIB-PC2A.

Once asserted, the DMA request line (DRQ) remains asserted until a DMA transfer occurs or until a read from the DIR or a write to the DOR occurs depending on the direction of the DMA transfer selected by the DMAO or DMAI bit.

The DMA Terminal Count interrupt is enabled whenever either the DMAO or DMAI bit is set and at least one of the interrupts internal to the uPD7210 is enabled (IE bits ISR1 or ISR2).

The DMA Terminal Count interrupt is asserted when the DMAO or DMAI bit is set and the IBM PC I/O Channel TC line sends a high pulse during a DMA transfer to the GPIB. A high pulse on the TC line indicates that the DMA Controller Chip has reached terminal count; i.e., the DMA Controller Byte Count has gone from 0 to FFFF for the corresponding DMA Channel. Therefore, the DMA Terminal Count interrupt indicates that the 8237A-5 has reached terminal count for the GPIB-PC2A DMA Channel.

The DMA Terminal Count interrupt is cleared when ISR2 is read.

The DMA Terminal Count interrupt can be detected by reading the Status register or channel Byte Count register of the 8237A-5. The Terminal Count bit corresponding to the GPIB-PC2A selected DMA Channel is set in the 8237A-5 Status register when the DMA Controller reaches terminal count for that DMA channel. All Terminal Count bits are cleared when the 8237A-5 Status

Register is read. The channel byte count should be FFFF unless the DMA Controller has been programmed to auto-initialize that channel.

4.8.1 Programming the 8237A-5 DMA Controller

Programming information for the 8237A-5 DMA Controller Chip may be found in the Intel Component Data Catalog and in the IBM PC Technical Reference Manual. The following consideration should be taken:

* The 8237A-5 is initialized upon power-up by the IBM PC Bios program located in ROM on the system board. Bios initializes the 8237A-5 in a way that is required for the IBM PC to operate correctly. Software written for handling GPIB-PC2A DMA transfers must in no way change the overall configuration of the 8237A-5 and commands written to the 8237A-5 should only affect the selected GPIB-PC2A DMA Channel and no other.

The manner in which the 8237A-5 is configured and used by Bios may be found in the listing of the Bios program in Appendix A of the IBM PC Technical Reference.

4.9 PROGRAMMING THE TIME-OF-DAY CLOCK

Programming information for the MM58167A Real-Time Clock may be found in the National Semiconductor Data Book. The following considerations should be taken when programming the MM58167A:

* To save I/O space, all registers of the clock are accessed indirectly through the CCR. To access the clock registers, the address of the desired register must first be written to the low order five bits of the CCR. Data can then be read or written directly to the clock register by either writing to or reading from the CDR.

* Interrupts from the clock can either be on the same level as the TLC and use the shared interrupt logic, or they can be on a separate level from the TLC. See section 3.3.1 for more information on enabling clock interrupts.

* The Standby Interrupt feature of the MM58167A is not implemented on the GPIB-PC2A.

Section Five

Theory of Operation

5.1 INTRODUCTION

This document describes the theory of operations of the GPIB-PC2A interface hardware. The information in this section follows the fundamental outline presented in the block diagram of the GPIB-PC2A and makes extensive references to the schematic diagram, Appendix A.

5.2 DATA BUS BUFFER

The GPIB-PC2A is interfaced to the IBM PC I/O channel data bus through a 74LS245 transceiver. The LS245 provides proper loading and hysteresis when it is configured for receiving, and has sufficient drive capability for proper operation when used to source data from the card to the channel data bus. Both sides of the LS245 are held at 3-state (pin 19 = high) unless the card is being addressed (MY BOARD = high) or the 8237A-5 DMA controller is responding to a DMA request from the board (DMAACK = high). The LS245 transceiver direction is controlled, indirectly, by the backplane IOR* signal. If IOR* is low, the LS245 is configured to source data onto the IBM PC I/O channel data bus from the uPD7210. If IOR* is high, the LS245 receives data from the channel data bus and buffers it to the uPD7210, PAL20R4 and optionally to the MM58167A and 74LS273 clock support chips.

The channel data bus signals, D0 through D7, when buffered through the LS245 become the internal data bus, GD0 through GD7.

5.3 CONTROL SIGNAL BUFFER

The IBM PC I/O channel control signals IOR*, IOW*, RESET DRV, and T/C; the DMA controller acknowledge signals DACK1*, DACK2*, and DACK3*; and the address lines A10, A11, and A12; are all received using a 74LS541 line receiver before being used on the board. The LS541 provides high input impedance and hysteresis to minimize bus loading and susceptibility to noise. The channel signals, once received by the LS541, are given different signatures to avoid confusion.

Channel Name	GPIB-PC2A Name
A10	RS0
A11	RS1
A12	RS2
IOR*	RD*
IOW*	WR*
RESET DRV	RESET
T/C	G T/C
DACK1/2/3*	DACK*

5.4 DMA CIRCUITRY

The DMA acknowledge and request signals on the channel connector are brought to wire-wrap pin arrays on the board. These pin arrays are arranged so that two small pin-to-pin shunt connectors, which are supplied with the card, can be used to select the proper pair of DMA signals for the board's use.

DACK* (DMA acknowledge) is ANDed with DMA EN* (which is generated by the INTERFACE CONTROL LOGIC) and becomes DMAACK (DMA acknowledge). When DMAACK* goes low the outputs of U12, an eight-bit shift register, are cleared and the DMAACK* pin on the uPD7210 goes low. When DMAACK* returns high the DMAACK* pin on the uPD7210 goes high, and the shift register delays the low-to-high transistion of DMAACK* by at least one clock cycle (approximately 210 nanoseconds, assuming a 4.77 MHz system clock).

The delayed DMAACK* signal is then ANDed with the DMAREQ signal from the uPD7210, thereby inhibiting a GPIB-PC2A bus DREQ until at least 210 ns after DMAACK* goes from low to high. This delay is implemented to solve a possible problem when using the uPD7210 with the Intel 8237A-5 DMA controller (i.e., if DMAACK* returns high before a minimum of 200 nanoseconds after RD* or WR* go high, a spurious uPD7210 DMAREQ signal of short duration may be generated, according to NEC). This solution inhibits any spurious uPD7210 DMAREQ pulses from being sent to the 8237A-5.

The signal used to drive the shift register, CLK (U12.8) is a received and buffered version of the CLOCK signal on the channel connector (4.77 MHz for the PC and PC/XT). CLK is also used by the uPD7210 for determining the GPIB T1 delay.

5.5 INTERRUPT CIRCUITRY

The interrupt circuitry is made up of logic and an array of wire-wrap pins to allow the user to select the channel connector interrupt request signal. The interrupt logic consists of two gates of a 74LS125A three-state buffer, each of which drives a row of wire-wrap pins. One of the three-state buffers is used to drive interrupt requests from the GPIB-PC2A's optional clock circuitry, while the other is used to request interrupts from the GPIB-PC2A's shared interrupt logic. The pin array is arranged so that small pin-to-pin shunt connectors (supplied with the card) can be used to select the desired interrupt request level for both the shared interrupt logic and optional clock. Note that it is not possible to apply shunt connectors so that both the shared interrupt logic and the clock circuitry interrupt on the same level, but the clock can be programmed to interrupt through the card's shared interrupt logic circuitry. The uPD7210 and DMA TC interrupts are logically ORed together to form a single interrupt request signal, TLC INT, which is used as an input to the shared interrupt circuitry.

5.5.1 Shared Interrupt Logic

The GPIB-PC2A contains special circuitry which allows it to share an interrupt request line with other cards which also contain the special interrupt circuitry. This shared interrupt circuitry allows multiple cards (and not just GPIB-PC2A cards) to share a common PC interrupt request line. The primary mechanism used to allow multiple cards to share an interrupt request line are as follows:

* The IRQX line is treated as an open-collector style line, even though it is connected to the output of a three-state gate (LS125A). The LS125A pulls the line to a low level (sinks current) when an interrupt is being requested by the card. Other cards using the same IRQX line can also pull the line to ground at the same time, which is a legal operation electrically. Simultaneously, those other cards using the same IRQX line which are <u>not</u> requesting an interrupt are not driving the line at all, since their LS125As are in high-impedance mode.

* The special circuitry, which is discussed in greater detail below, generates a low-going pulse on the selected interrupt request line in order to request an interrupt from the processor. The rising edge of this pulse is detected by the 8259A on the PC motherboard. The shared interrupt circuitry is inhibited from generating another pulse until either the interrupt is removed (presumably by the interrupt service routine) or until the board receives an I/O write to location 2FX. The duration of this pulse is determined by the clock period (210 nanoseconds for the PC and XT) signals.

The input of the LS125A, used in the shared interrupt logic circuitry, is tied to ground and the enable for the gate is driven by INT REQ* (interrupt request), which is generated by the SHARED INTERRUPT LOGIC. Therefore, an active interrupt request (INT REQ* = low) corresponds to a low level being transmitted onto the selected interrupt request bus line. When INT REQ* is high, no interrupt is being requested, and the output of the LS125A gate is in high impedance mode. The 8.2K ohm pull-up resistor attached to the output of the gate ensures that the IRQX signal is pulled up to a high level when the SHARED INTERRUPT CIRCUITRY is not requesting an interrupt.

Interrupt request inputs to the SHARED INTERRUPT LOGIC can originate in three places: the uPD7210 (INT), the DMA TC INTERRUPT logic (DMA TC INT), or the optional Time-of-Day Clock (CLK IRQ).

The DMA TC INT signal goes high whenever the DMA controller on the PC motherboard sends a T/C (Terminal Count) signal with the DACKn signal, and only if interrupts are enabled by the interrupt enable bits in ISR1 and/or ISR2, and only if one of the DMA transfer enable bits (DMAO or DMAI) in ISR2 is set.

Interrupts from the clock are processed through the shared interrupt logic if interrupts from the clock have been enabled (CLK IE = high) and clock interrupts have been programmed to use the shared interrupt logic (CIE = high). These three interrupt request conditions, uPD7210, DMA TC and clock, are logically ORed together to generate the IRQ signal. Note that it is the responsibility of the interrupt service routine to determine which, if any, of the three logical components on the GPIB-PC2A board is causing the interrupt.

IRQ is used as the A input of the shift register (U4) so that at the next low-to-high transition of CLK, the interrupt request, which is still present at a high level on the A input of the shift register, will be shifted in. The shift register output QB will then go high on the next CLK low-to-high transition. This will cause INT REQ* to go low. As soon as INT REQ* goes low, IRQX will go low and the selected channel interrupt request line will go low.

IRQX going low will cause the B input of the shift register (U4.2) to go low. After two low-to-high transistions of CLK, the INT REQ* signal will go high, thus causing IRQX to go high. The low-to-high transition of IRQX will also clock a flip-flop (U5.3) of the SHARED INTERRUPT LOGIC, which causes the INT INHIBIT* signal to go low, thus prohibiting any further INT REQ* high-to-low transistions.

Presumably at this point, the interrupt handler is invoked, and the source of the interrupt is cleared. This causes the A input to the shift register to go low, thus preventing further interrupt requests from being clocked through the SHARED INTERRUPT LOGIC until another valid interrupt condition occurs.

Finally, the interrupt handler must re-arm the SHARED INTERRUPT LOGIC by causing the INT INHIBIT* signal to return to a high level. This is accomplished by writing a byte to I/O port hex 2FX, where X is 0, 1, 2, 3, 4, 5, 6, or 7, depending on the interrupt request level being used and the configuration of the interrupt acknowledge jumpers described in the ADDRESS DECODE LOGIC section of this document.

If, for some reason, the interrupt handler chooses not to clear the source of the interrupt (perhaps another board got serviced), the IRQX high-to-low-to-high pulse will simply be regenerated after the handler writes to port hex 2FX. Thus, the SHARED INTERRUPT LOGIC will generate a continuous series of IRQX pulses, thereby causing a series of interrupts, until it gets serviced by the handler.

5.5.2 DMA TC Interrupt Logic

The DMA acknowledge signal (DACKn*) and the DMA terminal count signal (T/C) from the 8237A-5 DMA controller, which are selected and received by the DMA CIRCUITRY and the CONTROL SIGNAL BUFFER CIRCUITY to become the DMAACK* (DMA acknowledge) signal and the G T/C signal, are used with the INT EN (interrupt enable) signal from the INTERFACE CONTROL LOGIC to generate an interrupt request (DMA TC INT) when the DMA controller has reached terminal count (i.e., the DMA transfer has completed). The G T/C signal is used to clock a flip-flop (U5.11) which has the IE signal connected to the D input (U5.12). Thus, DMA TC INT will go high when the DMA controller reaches terminal count (ie., when G T/C makes a low-to-high transition) if the GPIB-PC2A has been previously armed for interrupt operation (by setting one of the interrupt enable bits in IMR1 or IMR2). The DMA TC INT signal is cleared to a low state on two conditions: when the RESET signal goes high or when the 8088 processor reads the uPD7210 Interrupt Status Register 2 (ISR2). The ISR2 read is the mechanism used by the interrupt handler to clear the DMA terminal count interrupt request.

5.5.3 Separate Clock Interrupt Logic

Logic is provided on the GPIB-PC2A to allow the clock to interrupt seperately from the shared interrupt logic. A second shunt connector can be configured on the wire-wrap array in addition to the shunt connector for the shared interrupt logic.

The LS125A used to drive interrupt requests from the clock is enabled by setting the SIE bit in the CCR. The LS125A's input is driven directly by the CLKIRQ signal. When the clock option is not present, the enable pin of the LS125A is pulled high by an 8.2K ohm resistor which forces the LS125A in its high-impedance mode.

5.6 ADDRESS DECODE LOGIC

The 16 address lines A0 through A15 on the PC channel connector are all used for determining the GPIB-PC2A address. A PAL16L2 is used along with some jumpers to decode the base address 2E1* and the interrupt arm signal 2FX*. The AEN signal from the processor is also used in the decode logic to ensure that the card is not inadvertantly selected when the 8088 processor does not have control of the bus.

The 2E1* signal is further conditioned to generate the signals MY BOARD, TLCSEL* and CLKSEL*. This is done using exclusive-or gates to compare the address lines A13 and A14 with on-board switches.

The binary base address 2E1 decoded by the PAL16L2 is as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
w	x	x	y	y	y	1	0	1	1	1	0	0	0	0	1

The three bits labeled y are used as register select signals for the uPD7210. These bits are also used to select the two clock registers, CCR and CDR. The two bits labeled x are used to select one of the four possible block numbers (1 through 4). The single bit, labled w, is used to select the NEC uPD7210 (w=0) or the NS MM58167A (w=1).

The interrupt arm signal 2FX* is decoded by the PAL16L2 as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	1	0	1	1	1	1	0	z	z	z

The three bits labeled z are user selectable as zeros or ones using switches on the card. The binary value of the three bits should be equivalent to the interrupt request level selected by the INTERRUPT CIRCUITRY to guarantee system integrity (i.e., if IRQ7 were selected, zzz should be configured as 111 using the switches on the board). The WR* and AEN* signals are also used in the decode logic to ensure that the 2FX* signal is generated only when the 8088 processor is writing to the decoded address.

5.7 INTERFACE CONTROL LOGIC

The PAL20R4 is the principal component in this group. The PAL monitors the internal data bus (GD0 through GD7), the uPD7210 select signal (TLCSEL*), and the uPD7210 register select signals (RS0 through RS3). These inputs allow the internal logic of the PAL to generate the following outputs:

* IR2*, or uPD7210 Interrupt Register 2, which is true (low) when either Interrupt Status Register 2 (ISR2) or Interrupt Mask Register 2 (IMR2) is being selected by the register select lines.

* The SC output is a latched signal, signifying that the GPIB-PC2A is the GPIB System Controller. SC is latched high whenever the uPD7210 receives an auxiliary command to set or clear the GPIB REN or IFC signals. SC is returned to a low level when a DSC (disable system control) or Chip Reset auxiliary command is sent to the uPD7210 or when RESET* is low.

* The DMA EN* signal is latched at a low level whenever either of the DMA enable bits in the uPD7210 IMR2 is set. DMA EN* is returned to a high level if both bits are cleared or if RESET* is low.

* The INT EN signal is latched at a high level if one or more of the interrupt enable bits in IMR1 or IMR2 are set. INT EN is returned to a low level if all of the interrupt enable bits are cleared or if RESET* is low.

The latched signals are all cleared when RESET* goes low. See the PAL20R4 fuse equations (Appendix A) for more information.

5.8 GPIB TALKER/LISTENER/CONTROLLER

The TLC is a large scale integrated circuit (NEC uPD7210) that contains most of the logic circuitry needed to program, control, and monitor the GPIB interface functions that are implemented by the GPIB-PC2A. Access to these functions is through 8 read-only registers and 13 write-only registers, of which 5 are indirectly addressed.

The TLC is enabled when the TLC SEL* signal is low and the register select signals RS0 through RS2 are decoded internally to access the appropriate register. Data on the internal data bus (GD0 through GD7) is strobed into write-only registers at the trailing edge of WR*. Data in the read-only registers is placed on the internal data bus a minimum access time after TLC SEL* and RD* are both low.

Most of the GPIB interface functions can be implemented or activated from either side of the TLC; that is, the interface may be programmed to do these functions from the IBM PC system or it may be addressed to do them from the GPIB. In terms of the IEEE-488 standard, the distinction between the two is generally that between local and remote interface messages, respectively.

Command and data to the GPIB and data from the GPIB are pipelined through the Command/Data Out Register (CDOR) and the Data In Register (DIR), respectively.

Interrupt Mask Registers 1 and 2 (IMR1/2) are used to enable or disable the generation of the TLC INT signal on the occurrence of 13 key GPIB conditions or events. IMR2 also has two bits to enable DMA transfers to or from the TLC. Interrupt Status Registers 1 and 2 (ISR1/2) record the occurrence of the 13 conditions or events.

NOTE

> ISR1 and ISR2 are not true status registers in that the bits are cleared whenever they are read. Appropriate programming steps must be taken to derive status information from the information provided by these registers.

The status bits function independently of the corresponding mask bits.

The Address Status Register (ADRS) reveals the current status of the Controller, Talker, and Listener functions. The Address Mode Register (ADMR) is used to program or enable the desired addressing mode as well as determine the use of the TLC's three T/R output signals.

Interface messages (commands) that are not automatically deciphered and implemented by the TLC can be read by the driver program through the Command Pass Through Register (CPTR). The program can then decide what action to take, usually by writing commands to the Auxiliary Mode Register (AUXMR).

The AUXMR is used both to issue special commands to the TLC and to write to the five hidden registers. The Internal Clock Register (ICR) is used to set critical timing parameters based on the frequency of the TLC input clock, which on the GPIB-PC2A is approximately 4.77 MHz. The Parallel Poll Register (PPR) is used to locally configure the TLC for polling. Auxiliary Registers A, B, and E (AUXRA/B/E) provide a means to control a variety of diverse functions, such as enabling handshake holdoffs, transmitting END when the EOS byte is sent, setting the END RX bit when EOS is received, enabling high speed transfers, and others.

The Address Register (ADR) is used to program two address registers, ADR0 and ADR1, which contain the base GPIB addresses recognized by the TLC as well as Talker and Listener disabling bits. The manner in which the TLC uses these registers depends on the address mode established in the ADMR. A bit in ADR1 indicates if END was set on the last byte received.

The TLC can automatically determine when an end of string character (EOS) is received by comparing each byte against the character stored in the End of String Register (EOSR).

5.9 GPIB TRANSCEIVERS

The TLC is interfaced to the GPIB through two special purpose transceivers, a DS75160A for the data signals and a DS75162A for the handshake and interface management signals. Signal direction routing through these transceivers is controlled by three signals from the TLC (T/R1 through T/R3) and the SC signal from the INTERFACE CONTROL LOGIC. T/R1 is high when the TLC is a Talker or Active Controller, and low when it is a Listener; it controls the direction of the data, handshake and EOI signals. T/R2 is inverted so that the DC signal is high when the TLC is Controller-In-Charge and low otherwise. It controls the direction of the ATN and SRQ signals. T/R3 is high when the three-state driver mode is active and low when the open collector mode is active. When the GPIB-PC2A is parallel polled, the transceiver switches to open collector mode during the poll. SC controls the direction of the IFC and REN signals, driving the GPIB when SC is high and receiving from the GPIB when it is low.

5.10 OPTIONAL TIME-OF-DAY CLOCK

The GPIB-PC2A can be purchased with an optional real-time, time-of-day clock, with alarm interval and battery back-up. The clock logic consists of a NS MM58167A Real-Time Clock chip, a 74LS273 8-bit register (CCR), and some discrete clock support components.

The MM58167A is a low threshold metal gate cmos circuit that functions as a real-time clock. The time base for the clock is generated with a 32.768 KHz crystal oscillator. A 3 V lithium cell and power-down circuitry are provided so that timekeeping may be maintained when power to the card is removed. See National Semiconductor Data Book for more information on the MM58167A.

To remove the battery for replacement, insert a blunt NON-METALLIC tool between the bottom face of the cell and holder. Five access slots are provided. Pry the battery up and at an angle. Remove with fingers or insulated tweezer.

To replace the battery, place the cell into position with the negative (-) face down and positive (+) face up. The positive top contact will snap close when properly inserted. Replace with BR 2325 3 V cell or equivalent.

The CCR (Clock Control Register) is a write only register made from a 74LS273 8-bit latch. It is used to address the MM58167A internal registers and to enable clock interrupts as described in section 3.3.1. The low order five bits of the register are connected directly to the MM58167A register select lines (A0 through A4) and must be programmed before any data transfers with the MM58167A can take place. Bit 5 (CLK IE) enables interrupts from the clock to either the separate clock interrupt circuitry or the shared interrupt logic as determined by bits 6 and 7. Bit 6 (SIE) allows interrupts from the clock by enabling the LS125A three-state driver which drives the CLKIRQ signal onto the wire-wrap array. Bit 7 (CIE) allows the clock to interrupt through the shared interrupt logic circuitry along with interrupts from the uPD7210 and DMA TC. All bits of the 74LS273 are cleared when RESET* goes low.

Section Six

IB2AIFT - Operation and Specification

This section contains the operation and specification of IB2AIFT, a standalone production test for the GPIB-PC2A. A copy of IB2AIFT.EXE is stored on the disk that accompanies this manual. IB2AIFT can test up to four GPIB-PC2As at one time.

The first part of this section, OPERATING INSTRUCTIONS, describes the procedure used to run IB2AIFT. The second part, the GPIB-PC2A DIAGNOSTIC SPECIFICATION, is the specification that IB2AIFT is based on and lists the steps contained in each individual test of the GPIB-PC2A executed by IB2AIFT.

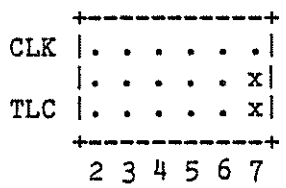
A wrap plug is needed to use IB2AIFT. To make a wrap plug, cut the end off a GPIB cable and connect the wires, as shown in Part II of the GPIB-PC2A DIAGNOSTIC SPECIFICATION.

6.1 OPERATING INSTRUCTIONS - GPIB-PC2A INCOMING FUNCTIONAL TEST

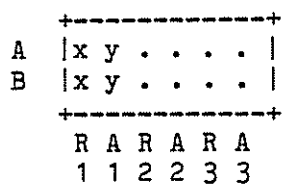
The incoming functional test (IB2AIFT) will be performed using an IBM PC equipped with a monitor, keyboard, and (optionally) an expansion chassis. A wrap plug is also required. Up to four boards at one time can be tested in either the PC chassis or an expansion chassis. Follow these test instructions:

1. Turn off power to the IBM PC and the expansion chassis if used).

2. All boards should be set to interrupt level 7. Select IRQ7 by placing the jumper such that the points marked "X" are connected.



3. All boards should be set to DMA channel 1. Select DMA channel 1 (DRQ1 and DACK1) by placing the two jumpers so that the points marked x are connected and the points marked y are connected.



4. Each board should have its own address, as follows: The x indicates where the switch is pressed. Always start with board 0. For example, to test three boards, use the switch settings for boards 0 through 2.

Board #	Address	Switch settings 1 2 3 4 5	
0	02E1	ON	x x
		OFF	x x x
1	22E1	ON	x
		OFF	x x x x
2	42E1	ON	x
		OFF	x x x x
3	62E1	ON	
		OFF	x x x x x

5. Turn on the expansion chassis, if used. The wrap plug should NOT be inserted on any board.

6. To start (or restart) the program from the DOS prompt A>, type IB2AIFT. Press return in response to the prompt.

7. The user is prompted for the number of boards installed and told when to insert the wrap plug.

6.2 DIAGNOSTIC PROGRAM SPECIFICATION FOR PRODUCTION TEST

1-16-85

Reference Documents:

8237A-5 Chip Description
8259A Chip Description
IBM PC Technical Reference

6.2.1 Board Configuration

The following configurations are used for the Diagnostic programs.

* DMA Channel - The boards may be configured for DMA Channel 1 or 3.

NOTE

IBIFT requires that boards be configured for DMA Channel 1 only.

* Interrupt Level - The boards may be configured for Interrupt Levels 2 through 7.

NOTE

IBIFT requires that boards be configured for Interrupt Level 7 only.

6.2.2 Board Tests

The hardware is tested by programming certain registers on the GPIB-PC2A board and within the PC and then reading other registers to determine if the operations were successful.

The Diagnostic tests are divided into two parts. Each part is made up of one or more routines. Each routine is divided into a series of independent tests.

Part I is completely standalone (uses no wrap plug) and is divided into four routines. The first routine (Tests 2 through 4) tests the interface between the GPIB interface hardware and the PC I/O Channel. The second routine (Tests 5 through 35) tests the GPIB functions of the interface. The third routine (Tests 36 and 37) tests the DMA transfer capability of the interface. The fourth routine (Tests 38 through 55) tests the interrupt capability of the interface, both for GPIB functions and DMA terminal count.

Part II consists of one routine (Test 56 through 66) that tests the GPIB interface using a wrap plug.

The board under test is initialized at the beginning of each test. Tests 2 through 55 are executed sequentially with the exception that Test 38 is executed before Tests 36 and 37. If the software includes the wrap plug tests, Tests 56 through 66 are executed after the wrap plug is attached.

A board will fail if there is a functional problem, a DMA or interrupt channel configuration problem, or the board is recognized at an illegal address.

The test of a board will stop on first step failure.

6.2.3 Test Specifications

The tests making up a GPIB-PC2A Diagnostic program are listed below.

Each test consists of a sequence of numbered steps. If a step fails, the test number, step number, board number and test error number are printed on the screen, along with the relevent error messages. The test number and step number can refer the user to the exact step that failed.

Most steps are a read from or write to an I/O register. For example:

write auxmr = 02

Write a 02 out to the 7210 Auxiliary Mode Register.

read isr2 = 00?

Read isr2, the 7210 Interrupt Status Register. If it is not a zero, the test step fails.

Other possible test steps are as follows:

read isr2 = Bit 5 set?

Read register isr2. If bit 5 is clear step passes, otherwise step fails.

read isr2 = Bit 5 clear?

Read register isr2. If bit 5 is clear step passes, otherwise step fails.

init (initialize)

Execute the initialization procedure

ckint (check interrupt)

Check that an interrupt occurred. If an interrupt did not occur, test step fails.

ckspint (check spurious interrupt)

Check that an interrupt did not occur. If an interrupt did occur, test step fails.

In addition, there are a few other symbols used in the tables, as follows:

04 | dmach

Represents the logical OR of the values 04 and dmach

stat * drqmask

Represents the logical AND of the values stat and drqmask

Some tests have additional steps that are unique to that test and are described in detail below the test.

6.2.4 Error Messages

The following is a list of error messages common to all tests.

"Test Pattern Error"

The value read from the indicated register did not match the expected value given in the test table. Both the expected and received values are printed after this message.

"Timed out waiting for valid test pattern"

The correct test pattern did not appear within the expected amount of time.

"Timed out waiting for DMA transfer"

A DMA operation did not begin within the expected amount of time.

"DMA transfer data error"

One or more bytes of data were incorrectly transferred in the DMA process.

"Expected interrupt did not occur"

No interrupt was detected when one should have appeared on the board's interrupt level (level 2 for IBIFT).

"Interrupt received when none expected"

An interrupt request was detected when all interrupts should have been clear.

In addition, there are one or more error messages unique to each test. These messages are described in the test tables section.

6.2.5 Registers

The following I/O registers are referred to in the test steps:

6.2.5.1 GPIB-PC2A Registers

7210 registers for base address 2E1

Read	Write	Address	RS2=A12	RS1=A11	RS0=A10
dir	cdor	02E1	0	0	0
isr1	imr1	06E1	0	0	1
isr2	imr2	0AE1	0	1	0
spsr	spmr	0EE1	0	1	1
adsr	admr	12E1	1	0	0
cptr	auxmr	16E1	1	0	1
adr0	adr	1AE1	1	1	0
adr1	eosr	1EE1	1	1	1

6.2.5.2 DMA Controller Registers

8237A-5 in IBM PC

Register	I/O Address
chaddr	2x(GPIB-PC2A Dma Channel #)
chcnt	chaddr + 2
stat	8
mask	A
mode	B
bpflop	C
page	83 if dma channel = 1 81 if dma channel = 2 82 if dma channel = 3

6.2.5.3 Interrupt Controller Registers

8259A in IBM PC

Register	I/O Address
OCW3	20
imr	21

6.2.6 Test Tables

All numbers in the tables that follow are hexadecimal.

6.2.6.1 INIT

The following initialization instructions occur at the beginning of each test so that the tests may be performed in any sequence. However, the sequence in which the tests are presented below is implemented in all the Diagnostic Programs.

At the start of each test, the following instructions (initialization procedure) are executed:

Write auxmr = 02	reset chip; clear isr1, isr2, spmr, parallel poll flag, EOI bit in asr; set clkr to 8; hold all functions in idle state
Write imr1 = 00	clear both mask registers
Write imr2 = 00	
Write admr = 00	clear mode register
Write auxmr = 00	release from idle state

PART I

Routine I: Test of GPIB Hardware Interface to the PC I/O Channel

Test 2

Purpose : Verify register locations

```
Step
  1  init
  2  Read isr1 = 00?   read mask registers
  3  Read isr2 = 00?
  4  Read adsr = 40?   check that ATN is set
```

Error numbers : 6300 "Base address incorrect"
6301 "Read registers incorrect following init"

Possible reasons for failure:

-board at wrong address
-cannot read from board
-7210 bad
-internal data bus problems
-register select lines bad
-ATN is on (Read adsr = 00) because 75162 or ATN line between it and 7210 is bad
-wrap plug attached to board
-cable and device attached to board

If passed:

-7210 is responding at right address
-can write to and read from 7210

TEST 3

Purpose - Verify data lines

```
Step
  1  init
  2  write spmr = ff
  3  read  spsr = ff?
  4  write spmr = 00
  5  read  spsr = 00?
```

Error number : 6302 "Write to SPMR failed"

Possible reasons for failure:

-cannot read or write interface
-7210 might be bad

-internal data bus problem
-register select lines bad

If passed:

-can read and write interface
-data lines probably ok
-address lines probably ok
-register select lines ok

TEST 4

Purpose : Verify board is an GPIB-PC2A board

Step
1 init
2 write adr = 55
3 write adr = aa
4 read adr0 = 55?
5 read adr1 = 2a?

Error number : 6303 "Write to ADR failed"

Possible reasons for failure:

-7210 might be bad
-internal data bus problem
-register select lines bad

If passed:

-interface definitely at right address
-data lines ok
-address lines ok
-interface I/O address decoding ok

Routine II: GPIB Functions

TEST 5

Purpose : Verify GPIB-PC2A can be programmed to listen

```
Step
  1  init
  2  write admr = 40    set listen only
  3  read  isr1 = 00?
  4  read  isr2 = 00?
  5  read  adsr = 44?   check listener active
```

Error number 6310: "Cannot be programmed to listen"

Possible reasons for failure:

- -7210 might be bad
- -register select lines bad
- -ATN line driven low external to 7210

If passed:

- -can program 7210 to be listener

TEST 6

Purpose : Verify GPIB-PC2A can be programmed to talk

```
Step
  1  init
  2  write admr = 80    set talker only
  3  read  isr1 = 02?   check DO bit set
  4  read  isr2 = 00?
  5  read  adsr = 42?   check for talker active
```

Error number : 6311 "Cannot be programmed to talk"

Possible reasons for failure:

- -7210 might be bad

If passed:

- -can program 7210 to be talker

TEST 7

Purpose : Verify GPIB-PC2A can take control and become controller

```
Step
  1  init
  2  write admr  = 31   t/r mode 3, address mode 1
  3  write auxmr = 1e   set IFC
  4  read  isr1  = 00?
  5  read  isr2  = 09?  check that CO and ADSC bits set
  6  read  admr  = 80?  check for controller-in-charge,
                           asserting ATN
```

Error number : 6312 "Cannot take control with IFC"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can take control

TEST 8

Purpose : Verify GPIB-PC2A can go to standby

```
Step
  1  init
  2  write admr  = 31   set t/r mode 3, address mode 1
  3  write auxmr = 1e   set IFC
  4  read  isr1  = 00?
  5  read  isr2  = 09?  check that CO and ADSC bits set
  6  write auxmr = 16   clear IFC
  7  write auxmr = 10   go to standby
  8  read  isr2  = 00?
  9  read  adsr  = c0?  check for controller-in-charge,
                           ATN unasserted
```

Error number : 6313 "cannot go to standby"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can go to standby

TEST 9

Purpose : Verify GPIB-PC2A can take control asynchronously

```
Step
  1  init
  2  write admr  = 31   set t/r mode 3, address mode 1
  3  write auxmr = 1e   set IFC
  4  read  isr1  = 00?
  5  read  isr2  = 09?  check that CO and ADSC bits set
  6  write auxmr = 16   clear IFC
  7  write auxmr = 10   go to standby
  8  read  isr2  = 00?
  9  read  adsr  = c0?  check controller-in-charge,
                           ATN unasserted
 10  write auxmr = 11   take control asynchronously
 11  read  isr2  = 08?  check CO bit set
 12  read  adsr  = 80?  check controller-in-charge,
                           ATN asserted
```

Error number : 6314 "Cannot take control asynchronously"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can take control asynchronously

TEST 10

Purpose : Verify GPIB-PC2A cannot take control when it shouldn't

```
Step
  1  init
  2  write admr  = 31   set t/r mode 3, address mode 1
  3  write auxmr = 1e   set IFC
  4  read  isr1  = 00?
  5  read  isr2  = 09?  check CO, ADSC bits set
  6  write auxmr = 16   clear IFC
  7  write auxmr = 10   go to standby
  8  read  isr2  = 00?
  9  read  adsr  = c0?  check controller-in-charge
 10  write auxmr = 12   take control synchronously
 11  read  isr2  = 00?
 12  read  adsr  = c0?  check controller-in-charge,
                           ATN unasserted
```

Error number : 6315 "Takes control synchronously incorrectly"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 does not take control when it shouldn't

TEST 11

Purpose : Verify GPIB-PC2A can pass system control

```
Step
  1  init
  2  write adr   = 00   major address = 0
  3  write adr   = e0   disable minor TL
  4  write admr  = 31   tr mode 3, address mode 1
  5  write auxmr = 1e   set IFC
  6  read  isr1  = 00?
  7  read  isr2  = 09?  check CO, ADSC bits set
  8  write auxmr = 16   clear IFC
  9  write cdor  = 41   send other talk address
 10  write cdor  = 09   send take control command
 11  read  isr2  = 01?  ADSC
 12  read  adsr  = 40?  check not controller-in-charge,
                           ATN unasserted
```

Error number : 6316 "Cannot pass control"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can pass control

TEST 12

Purpose : Verify GPIB-PC2A can listen at all 32 possible listen addresses

```
Step
  1  init
  2  write adr   = 00    program major address
  3  write adr   = e0    disable minor TL
  4  write admr  = 31    tr mode 3, address mode 1
  5  write auxmr = 1e    set IFC
  6  write auxmr = 16    clear IFC
  7  read  isr2  = 09?   check CO, ADSC bits set
  8  read  adsr  = 80?   check for controller-in-charge,
                            ATN asserted
  9  write cdor  = 20    send my listen address
 10  read  isr1  = 00?
 11  read  isr2  = 09?   check for CO, ADSC bits set
```

```
 12  read  adsr  = 94?  check for controller-in-charge,
                          ATN asserted, listener primary
                          address, listener active
```

Repeat for my listen addresses 21 through 3E.

Error number : 6317 "Cannot be addressed to listen"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be addressed to listen

TEST 13

Purpose : Verify GPIB-PC2A can be unaddressed as listener

```
Step
  1  init
  2  write adr   = 00   major address = 0
  3  write adr   = e0   disable minor TL
  4  write admr  = 31   tr mode 3, address mode 1
  5  write auxmr = 1e   set IFC
  6  write auxmr = 16   clear IFC
  7  read  isr2  = 09?  check CO, ADSC bits set
  8  read  adsr  = 80?  check controller-in-charge,
                          ATN asserted
  9  write cdor  = 20   send my listen address
 10  read  isr1  = 00?
 11  read  isr2  = 09?  check CO, ADSC set
 12  read  adsr  = 94?  check for controller-in-charge,
                          ATN asserted, listener primary
                          addressed, listener active
 13  write cdor  = 3f   send unlisten command
 14  read  isr2  = 09?  check CO, ADSC bits set
 15  read  adsr  = 80?  check controller-in-charge, ATN
                          asserted, not listener
```

Error number : 6318 "cannot be unaddressed to listen"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be unaddressed to listen

TEST 14

Purpose : Verify GPIB-PC2A can be addressed to talk at all 32 talk addresses

Step				
1	init			
2	write	adr	= 00	major address = 0
3	write	adr	= e0	disable minor TL
4	write	admr	= 31	t/r mode 1, address mode 3
5	write	auxmr	= 1e	set IFC
6	write	auxmr	= 16	clear IFC
7	read	isr2	= 09?	check CO, ADSC
8	read	adsr	= 80?	controller-in-charge, ATN
9	write	cdor	= 40	send my talk address
10	read	isr1	= 00?	
11	read	isr2	= 09?	check CO, ADSC
12	read	adsr	= 8a?	controller-in-charge, ATN,tpas, talker active
13	write	auxmr	= 10	go to standby
14	read	isr1	= 02?	check DO

repeat for my talk addresses 41 - 57

Error number : 6319 "cannot be addressed to talk"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be addressed to talk

TEST 15

Purpose : Verify GPIB-PC2A can be unaddressed as talker

Step				
1	init			
2	write	adr	= 00	major address = 0
3	write	adr	= e0	disable minor TL
4	write	admr	= 31	t/r mode 1, address mode 3
5	write	auxmr	= 1e	set IFC
6	write	auxmr	= 16	clear IFC
7	read	isr2	= 09?	check CO, ADSC
8	read	adsr	= 80?	controller-in-charge, ATN
9	write	cdor	= 40	send my talk address
10	read	isr1	= 00?	
11	read	isr2	= 09?	check CO, ADSC
12	read	adsr	= 8a?	controller-in-charge, ATN, tpas, talker active
13	write	cdor	= 5f	untalk
14	read	isr2	= 09?	check CO, ADSC
15	read	adsr	= 80?	controller-in-charge, ATN

Error number : 6320 "Cannot be unaddressed to talk"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be unaddressed to talk

TEST 16

Purpose : Verify GPIB-PC2A can be addressed to listen at all 960 extended addresses

```
Step
  1  init
  2  write adr   = 00    primary address = 0
  3  write adr   = 80    secondary address = 0
  4  write admr  = 32    t/r mode 1,address mode 2
  5  write auxmr = 1e    set IFC
  6  write auxmr = 16    clear IFC
  7  read  isr2  = 09?   check CO, ADSC
  8  read adsr   = 80?   controller-in-charge, ATN
  9  write cdor  = 20    send my listen address
 10  read  isr1  = 00?
 11  read  isr2  = 08?   check CO
 12  read  adsr  = 90?   controller-in-charge, ATN, listener
                             primary address state
 13  write cdor  = 60    my secondary address
 14  read  isr1  = 00?
 15  read  isr2  = 09?   check CO, ADSC
 16  read  adsr  = 94?   controller-in-charge, ATN, listener
                           primary address state, listener active
```

repeat for 960 other listen addresses

Error number : 6321 "Cannot be addressed to listen"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be addressed to listen

TEST 17

Purpose : Verify GPIB-PC2A can be unaddressed as listener with extended address

Step				
1	init			
2	write	adr	= 00	primary address = 0
3	write	adr	= 80	secondary address = 0
4	write	admr	= 32	tr mode 3, address mode 2
5	write	auxmr	= 1e	set IFC
6	write	auxmr	= 16	clear IFC
7	read	isr2	= 09?	check CO, ADSC
8	read	adsr	= 80?	controller-in-charge, ATN
9	write	cdor	= 20	send my listen address
10	read	isr1	= 00?	
11	read	isr2	= 08?	check CO bit
12	read	adsr	= 90?	controller-in-charge, ATN, listener primary address state
13	write	cdor	= 60	my secondary address
14	read	isr2	= 09?	check CO, ADSC
15	read	adsr	= 94?	controller-in-charge, ATN, listener primary address state, listener active
16	write	cdor	= 3F	unlisten
17	read	isr2	= 09?	check CO, ADSC
18	read	adsr	= 80?	controller-in-charge, ATN

Error number : 6322 "cannot be unaddressed to listen with extended addressing'

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be unaddressed to listen with extended addressing

TEST 18

Purpose : Verify GPIB-PC2A can be addressed to talk at all 960 extended addresses

Step				
1	init			
2	write	adr	= 00	primary address = 0
3	write	adr	= 80	secondary address = 0
4	write	admr	= 32	tr mode 3, address mode 2
5	write	auxmr	= 1e	set IFC
6	write	auxmr	= 16	clear IFC
7	read	isr2	= 09?	check CO, ADSC
8	read	adsr	= 80?	controller-in-charge, ATN
9	write	cdor	= 40	send my talk address
10	read	isr1	= 00?	
11	read	isr2	= 08?	check CO bit
12	read	adsr	= 88?	controller-in-charge, ATN, tpas

```
 13  write cdor  = 60    my secondary address
 14  read  isr1  = 00?
 15  read  isr2  = 09?   check CO, ADSC
 16  read  adsr  = 8a?   controller-in-charge, ATN, tpas,
                            talker active
 17  write auxmr = 10    write GTS (go to standby)
 18  read  isr1  = 02?   check DO bit
```

repeat for 960 other talk addresses

SError number : 6323 "Cannot be addressed to talk"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be addressed to talk with extended addressing

TEST 19

Purpose : Verify GPIB-PC2A can be unaddressed to talk with extended addressing

```
Step
  1  init
  2  write adr   = 00    primary address = 0
  3  write adr   = 80    secondary address = 60
  4  write admr  = 32    tr mode 3, address mode 2
  5  write auxmr = 1e    set IFC
  6  write auxmr = 16    clear IFC
  7  read  isr2  = 09?   check CO, ADSC
  8  read  adsr  = 80?   controller-in-charge, ATN
  9  write cdor  = 40    send my talk address
 10  read  isr1  = 00?
 11  read  isr2  = 08?   check CO
 12  read  adsr  = 88?   controller-in-charge, ATN, tpas
 13  write cdor  = 60    my secondary address
 14  read  isr1  = 00?
 15  read  isr2  = 09?   check CO, ADSC
 16  read  adsr  = 8a?   controller-in-charge, ATN, tpas,
                            talker active
 17  write cdor  = 5f    untalk
 18  read  isr2  = 09?   check CO, ADSC
 19  read  adsr  = 80?   controller-in-charge, ATN
```

Error number : 6324 "cannot be unaddressed to talk with extended addressing"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can be unaddressed to talk with extended addressing

TEST 20

Purpose : Verify GPIB-PC2A can recognize data in condition

```
Step
  1  init
  2  write admr  = c0    talk only, listen only
  3  read  isr1  = 02?   check DO bit set
  4  read  isr2  = 00?
  5  read  adsr  = 46?   listener active, talker active
  6  write cdor  = aa
  7  read  isr1  = 03?   check DO, DI bits
  8  read  dir   = aa?   data in
```

Error number : 6325 "Cannot write to self"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can write to self

TEST 21

Purpose : Verify GPIB-PC2A can recognize error condition

```
Step
  1  init
  2  write admr = b0    talk only, tr mode 3
  3  read  isr1 = 02?   check DO
  4  read  isr2 = 00?
  5  read  adsr = 42?   talker active
  6  write cdor = aa
  7  read  isr1 = 06?   ERR, DO
```

Error number : 6326 "Cannot generate handshake error"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can generate handshake error

TEST 22

Purpose : Verify GPIB-PC2A can recognize device clear command

Step				
1	init			
2	write	admr	= 70	listen only, tr mode 3
3	read	isr1	= 00?	
4	read	isr2	= 00?	
5	read	adsr	= 44?	listener active
6	write	auxmr	= 1e	set IFC
7	write	auxmr	= 16	clear IFC
8	read	isr2	= 08?	check CO bit
9	read	adsr	= 84?	check controller-in-charge, ATN, listener active
10	write	cdor	= 14	set DCL
11	read	isr1	= 08?	check DEC
12	read	isr2	= 08?	check CO bit

Error number : 6327 "Cannot detect DCL message"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect DCL message

TEST 23

Purpose : Verify GPIB-PC2A can recognize Selected Device Clear (SDC)

Step				
1	init			
2	write	admr	= 70	listen only, tr mode 3
3	read	isr1	= 00?	
4	read	isr2	= 00?	
5	read	adsr	= 44?	listener active
6	write	auxmr	= 1e	set IFC
7	write	auxmr	= 16	clear IFC
8	read	isr2	= 08?	check CO bit
9	read	adsr	= 84?	controller-in-charge, ATN, listener active
10	write	cdor	= 04	send SDC command
11	read	isr1	= 08?	check DEC bit
12	read	isr2	= 08?	check CO bit

Error number : 6328 "Cannot detect SDC message"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect SDC message

TEST 24

Purpose : Verify GPIB-PC2A can recognize EOI and set end

```
Step
  1  init
  2  write admr  = f0    talk only, listen only, tr mode 3
  3  read  isr1  = 02?   check DO
  4  read  isr2  = 00?
  5  read  adsr  = 46?   listener active, talker active
  6  write auxmr = 06    send END
  7  write cdor  = 55
  8  read  isr1  = 13?   check END, RX, DO, and DI
  9  read  dir   = 55?   data in
```

Error number : 6329 "Cannot detect END with EOI"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect END with EOI

TEST 25

Purpose : Verify GPIB-PC2A can recognize EOI and set REOI

```
Step
  1  init
  2  write admr  = f0    talk only, listen only, tr mode 3
  3  read  isr1  = 02?   check DO
  4  read  isr2  = 00?
  5  read  adsr  = 46?   listener active, talker active
  6  write auxmr = 06    send end
  7  write cdor  = 55
  8  read  isr1  = 13?   check END,RC, DO, DI
  9  read  adr1  = 80?   check EOI
 10  read  dir   = 55?   data in
```

Error number : 6330 "Cannot detect EOI with EOI"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect EOI with EOI

TEST 26

Purpose : Verify GPIB-PC2A can recognize 8-bit eos character and set end

```
Step
  1  init
  2  write admr  = f0    talk only, listen only, tr mode 3
  3  read  isr1  = 02?   check DO bit
  4  read  isr2  = 00?
  5  read  adsr  = 46?   listener active, talker active
  6  read  auxmr = 94?   set end on 8 bit eos
  7  write eosr  = aa
  8  write cdor  = 2a
  9  read  isr1  = 03?   check DO, DI
 10  read  dir   = 2a?   data in
 11  write cdor  = aa
 12  read  isr1  = 13?   check END, RX, DO, DI
```

Error number : 6331 "Cannot detect END with 8 bit EOS"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect END with 8 bit EOS

TEST 27

Purpose : Verify GPIB-PC2A can recognize 7-bit eos character and set end

```
Step
  1  init
  2  write admr  = f0    talk only, listen only, tr mode 3
  3  read  isr1  = 02?   check DO
  4  read  isr2  = 00?
  5  read  adsr  = 46?   listener active, talker active
  6  read  auxmr = 84?   set end on 7 bit eos
  7  write eosr  = aa
  8  write cdor  = 2a
  9  read  isr1  = 13?   check END, RX, DO, DI
 10  read  dir   = 2a?   data in
 11  write cdor  = aa
 12  read  isr1  = 13?   check END, RX, DO, DI
```

Error number : 6332 "Cannot detect END with 7 bit EOS.

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect END with 7 bit EOS.

TEST 28

Purpose : Verify GPIB-PC2A can recognize group execute trigger command

Step				
1	init			
2	write admr	= 70	listen only, tr mode 3
3	read isr1	= 00?	
4	read isr2	= 00?	
5	read adsr	= 44?	listener active
6	write auxmr	= 1e	set IFC
7	write auxmr	= 16	clear IFC
8	read isr2	= 08?	check DO bit
9	read adsr	= 84?	controller-in-charge, ATN, listener active
10	write cdor	= 08	write group execute trigger command
11	read isr1	= 20?	check DET bit
12	read isr2	= 08?	check CO

Error number : 6333 "Cannot detect GET"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 can detect GET

TEST 29

Purpose : Verify GPIB-PC2A can set APT bit on unrecognized command

Step			
1	init		
2	write adr	= 00	major primary address = 0
3	write adr	= e0	disable minor address
4	write admr	= 33	tr mode 3, address mode 3
5	write auxmr	= 1e	set IFC
6	write auxmr	= 16	clear IFC
7	read isr2	= 09?	check CO, ADSC

```
 8  read  adsr  = 80?   controller-in-charge, ATN
 9  write cdor  = 40    send my talk address
10  read  isr1  = 00?
11  read  isr2  = 08?   check CO
12  read  adsr  = 88?   controller-in-charge, ATN, talker
                           primary address
13  write cdor  = 60    send my secondary address
14  read  isr1  = 40?   check APT bit
15  read  isr2  = 00?
16  read  adsr  = 88?   controller-in-charge, ATN, talker
                           primary address
17  write auxmr = 0f    valid pass through
18  read  isr2  = 09?   check CO, ADSC
19  read  adsr  = 8a?   controller-in-charge, ATN, talker
                           primary address, talker active
```

Error number : 6334 "Mode 3 addressing not functioning"

Possible reasons for failure:

-7210 might be bad.

If passed:

-Mode 3 addressing functional

TEST 30

Purpose : Verify GPIB-PC2A can recognize undefined command and execute

```
Step
  1  init
  2  write admr  = 70    listen only, tr mode 3, address mode 0
  3  write auxmr = 1e    set IFC
  4  write auxmr = 16    clear IFC
  5  write auxmr = a1    cpt enable
  6  read  isr1  = 00?
  7  read  isr2  = 08?   check CO
  8  read  adsr  = 84?   controller-in-charge, ATN, listener
                           active
  9  write cdor  = 02    any undefined command
 10  read  isr1  = 80?   check CPT
 11  read  cptr  = 02?   undefined command
 12  write cdor  = 03    another one
 13  read  isr1  = 00?   have not acted on first one yet
 14  write auxmr = 07    invalid command
 15  read  cdor  = 06?   another one
 16  read  isr1  = 80?   check CPT
 17  read  cptr  = 06?   new undefined command
```

Error number : 6335 "Cannot recognize undefined command"

Possible reasons for failure:

-7210 might be bad.

If passed:

-7210 recognizes undefined command

TEST 31

Purpose : Verify GPIB-PC2A can set REM, REMC, LOK, and LOKC bits

Step				
1	init			
2	write	admr	= 31	tr mode 3, address mode 1
3	write	adr	= 00	major address = 0
4	write	adr	= e0	disable minor address
5	write	auxmr	= 1e	set IFC
6	write	auxmr	= 16	clear IFC
7	write	auxmr	= 1f	set REN
8	write	auxmr	= 14	release system control
9	read	isr2	= 09?	check CO, ADSC
10	write	cdor	= 20	send my listen address
11	read	isr2	= 09?	check CO, ADSC
12	write	auxmr	= 1e	set IFC
13	write	auxmr	= 16	clear IFC
14	write	auxmr	= 1f	set REN
15	read	isr1	= 00?	
16	read	isr2	= 01?	ADSC
17	read	adsr	= 90?	controller-in-charge, ATN and listener active
18	write	cdor	= 20	send my listen address
19	read	isr1	= 0?	
20	read	isr2	= 1b?	REM, CO, REMC, ADSC
21	read	adsr	= 94?	controller-in-charge, ATN, listener primary address, listener active
22	write	cdor	= 11	write local lockout
23	read	isr2	= 3c?	check LOK, REM, CO, LOKC

Error number : 6336 "Cannot detect REM, REMC, LOK, or LOKC"

Possible reasons for failure:

-REN line may be grounded
-7210 might be bad

If passed:

-7210 can detect REM, REMC, LOK, and LOKC

TEST 32

Purpose : Verify GPIB-PC2A can clear REM and LOK bits

Step				
1	init			
2	write	admr	= 31	tr mode 3, address mode 1
3	write	adr	= 00	major address = 0
4	write	adr	= e0	disable minor address
5	write	auxmr	= 1e	set IFC
6	write	auxmr	= 16	clear IFC
7	write	auxmr	= 1f	set REN
8	read	isr1	= 00?	
9	read	isr2	= 09?	check CO, ADSC
10	read	adsr	= 80?	controller-in-charge, ATN
11	write	cdor	= 20	send my listen address
12	read	isr1	= 0?	
13	read	isr2	= 1b?	REM, CO, REMC, ADSC
14	read	adsr	= 94?	controller-in-charge, ATN, listener primary address, listener active
15	write	cdor	= 11	write local lockout
16	read	isr2	= 3c?	check LOK, REM, CO, LOKC
17	write	auxmr	= 17	clear REN
18	read	isr2	= 06?	check LOKC, REMC

Error number : 6337 "Cannot clear REM or LOK"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can detect REM and LOK

TEST 33

Purpose : Verify GPIB-PC2A can set SRQI

Step				
1	init			
2	write	admr	= f0	talk only, listen only, tr mode 3
3	write	auxmr	= 1e	set IFC
4	write	auxmr	= 16	clear IFC
5	read	isr1	= 00?	
6	read	isr2	= 08?	CO
7	write	spmr	= 40	set SRQ
8	read	isr2	= 40?	SRQI

Error number : 6338 "Cannot detect SRQ"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can detect SRQ

TEST 34

Purpose : Verify GPIB-PC2A can perform serial poll as controller-in-charge and as a responding device

Step				
1	init			
2	write	admr	= f0	talk only, listen only, tr mode 3
3	write	auxmr	= 1e	set IFC
4	write	auxmr	= 16	clear IFC
5	read	isr1	= 00?	
6	read	isr2	= 08?	check CO
7	write	spmr	= 55	set SRQ and status byte
8	read	isr2	= 40?	SRQI
9	write	cdor	= 18	write Serial Poll Enable command
10	read	adsr	= a6?	controller-in-charge, ATN, SPMS, listener active, talker active
11	write	auxmr	= 10	write Go To Standby command
12	read	isr1	= 01?	check DI bit
13	read	dir	= 55?	status byte
14	write	auxmr	= 11	take control asynchronously
15	write	auxmr	= 10	Go to standbys
16	read	isr1	= 01?	check DI bit
17	read	dir	= 15?	status byte without RSQ

Error number : 6339 "Cannot conduct serial poll"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can conduct serial poll

TEST 35

Purpose : Verify GPIB-PC2A can perform a parallel poll both as controller-in- charge and as responding device

Step				
1	init			
2	write	admr	= 70	listen only, tr mode 3
3	write	auxmr	= 1e	set IFC
4	write	auxmr	= 16	clear IFC
5	write	auxmr	= 01	clear ist
6	write	auxmr	= 60	lpe/ppe on dio1

```
 7  read  isr1  = 00?
 8  read  isr2  = 08?   check CO
 9  read  adsr  = 84?   controller-in-charge, ATN, listener
                          active
10  write auxmr = 1d    parallel poll
11  read  cptr  = 01?   response
12  write auxmr = 6b    lpe/ppe  on dio5
13  write auxmr = 1d    parallel poll
14  read  cptr  = 00?   no response
15  write auxmr = 09    set IST
16  write auxmr = 1d    parallel poll
17  read  cptr  = 08?   new response
18  write auxmr = 70    lpe/ppd
19  write auxmr = 1d    parallel poll
20  read  cptr  = 00?   no response
```

Error number : 6340 "Cannot conduct parallel poll"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can conduct serial poll

Interrupt Test -- see Routine IV

TEST 38

Purpose : Verify unexpected interrupts have not occurred.

```
Step
  1  init
  2  ckspint
     re-enable shared interrupt line
  3  ckspint
```

Error number : 6358 "Unevoked interrupt received"

Possible reasons for failure:

-IRQ line may be malfunctioning
-7210 might be bad

If passed:

-no pending interrupts

Routine III: DMA Transfers

dmach = dma channel number
drqmask == 1 shifted left dmach + 4 times
dmamode = dmach ored with 48H

TEST 36

Purpose : Verify GPIB-PC2A can DMA correctly from GPIB to memory

Step			
1	init, write (2fX) = 00		
2	write bpflop = 00		set pointer
3	write mask	= drqmask	disable dmas
4	write mode	= dmamode	dma write
5	write imr2	= 20	enable dma out
6	read stat	= bit 5 clear?	no request pending, dreq driven low
7	write admr	= f0	make GPIB-PC2A talker only, listener only, t/r mode 3
8	read adsr	= 46?	verify talker active, listener active
9	read stat	= bit 5 set?	request pending, dreq driven high
10	write chadr	= low byte of address	
11	write chadr	= next byte of address	
12	write port	= top nibble of address	
13	write chcnt	= 1F	low byte of count-1
14	write chcnt	= 00	high byte of count-1
15	write mask	= dmach	start dma
16	check that DMA started on time		
17	check that data transferred correctly		
18	check for no DMA TC interrupt		

Error numbers:

Step 1-15	: 6350	"Cannot DMA to 7210"
Step 16-17	: 6351	"Data error on DMA to 7210"
Step 18	: 6358	"Unevoked interrupt received"

Possible reasons for failure:

All steps:
-DMA jumpers bad or improperly installed
-PAL possibly bad
-7210 possibly bad
Step 6:
-DREQ line high or floating
Step 9:
-DREQ line low
Step 16:
-DACK high or floating
-DMAACK high or floating
Step 17:
-data lines possibly bad

If passed:

-no pending interrupts
-DMA into memory functions correctly

TEST 37

Purpose : Verify GPIB-PC2A can DMA correctly from memory to GPIB

```
Step
  1  init, write (2fX) = 00
  2  write bpflop = 00        set pointer
  3  write mask = 04|dmach    disable DMA
  4  write mode = 04|dmach    dma write to from memory
  5  write imr2 = 10          enable DMA in
  6  read  stat * drqmask
        = Bit 5 clear ?       check DREQ driven low
  7  write admr = f0          talk only, listen only, tr mode 3
  8  read  adsr = 46?         check for talker active,
                                  listener active
  9  write chadr = low byte of address
 10  write chadr = next byte of address
 11  write page  = top nibble of address
 12  write chcnt = low byte of count-1
 13  write chcnt = high byte of count-1
 14  write mask = dmach       start dma
 15  check that DMA started in time
 16  check that data transferred correctly
 17  check for no DMA TC interrupt
```

Error numbers:

Step	1-15	: 6352	"Cannot DMA from 7210"
Step	16	: 6353	"Data error on DMA from 7210"
Step	17	: 6358	"Unevoked interrupt received"

Possible reasons for failure:

All steps:
-DMA jumpers bad or improperly installed
-PAL possibly bad
-7210 possibly bad
Step 6:
-DREQ high or floating
Step 15:
-DACK high or floating
-DMAACK high or floating
Step 16:
-Data lines possibly bad

If passed:

-DMA from memory to GPIB functions correctly
-no pending interrupts

Routine IV: Interrupts

TEST 39

Purpose : Verify GPIB-PC2A can interrupt on Address Status Change

Step			
1	init, write (2fX)	= 00	
2	write imr2	= 01	enable ADSC interrupt
3	write admr	= 31	set tr mode 3, address mode 1
4	write auxmr	= 1e	set IFC
5	ckint		check that int occurred
6	read isr2	= 89?	check INT, CO, and ADSC bits

Error number : 6359 "Cannot interrupt on ADSC"

Possible reasons for failure:

- -interrupt jumpers bad or incorrectly set
- -PAL might be bad
- -IBM interrupt controller might be bad
- -interrupt circuitry may be malfunctioning
- -7210 might be bad

If passed:

- -interrupt circuitry is functioning correctly
- -7210 can interrupt on ADSC

TEST 40

Purpose : Verify GPIB-PC2A can be interrupt on Address Status Change

Step			
1	init, write (2fX)	= 00	
2	write imr2	= 01	enable CO interrupt
3	write admr	= 31	setup tr mode 3, address mode 1
4	write auxmr	= 1e	set IFC
5	ckint		check that interrupt occurred
6	read isr2	= 89?	check int, CO, and ADSC bits

Error number : 6360 "Cannot interrupt on ADSC"

Possible reasons for failure:

- -7210 might be bad

If passed:

-7210 can interrupt on ADSC

TEST 41

Purpose : Verify GPIB-PC2A can interrupt on CO condition

```
Step
  1  init, write (2fX) = 00
  2  write imr2   = 08    enable CO interrupts
  3  write admr   = 31    setup t/r mode 3, address mode 1
  4  write auxmr  = 1e    set IFC
  5  ckint
  6  read  isr2   = 89?   check int, CO, and ADSC bits set
```

Error number : 6361 "Cannot interrupt on CO"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can can interrupt on CO

TEST 42

Purpose : Verify GPIB-PC2A can interrupt on DO condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 02    enable do interrupt
  3  write adr    = 0     major primary address = 0
  4  write adr    = e0    disable minor address
  5  write admr   = 31    setup t/r mode 3, address mode 1
  6  write auxmr  = 1e    set IFC
  7  write auxmr  = 16    clear IFC
  8  read  isr2   = 09?   Check that CO and ADSC bits are set
  9  read  adsr   = 80?   Check that GPIB-PC2A is controller-in-charge
                             and asserting ATN
 10  write cdor   = 40    send my talk address
 11  read  isr1   = 00?
 12  read  isr2   = 09?   check that CO, ADSC bits are set
 13  read  adsr   = 8a?   controller-in-charge, asserting ATN;
                            talker primary addressed, and talker
                            active
 14  write auxmr  = 10    go to standby
 15  ckint                check that interrupt occurred
 16  read  isr1   = 02?   check DO bit set
```

Error number : 6362 "Cannot interrupt on DO"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on DO

TEST 43

Purpose : Verify GPIB-PC2A can interrupt on DI condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 01    enable DI interrupt
  3  write admr   = c0    program GPIB-PC2A for talker only,
                            listener only
  4  read  isr1   = 02?   check DO bit set
  5  read  isr2   = 00?
  6  read  adsr   = 46?   check that GPIB-PC2A is talker only,listener only
  7  write cdor   = aa    write out data byte
  8  ckint
  9  read  isr1   = 03?   check DO and DI bits set
```

Error number : 6363 "Cannot interrupt on DI"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on DI

TEST 44

Purpose : Verify GPIB-PC2A can interrupt on ERR condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 04    enable ERR interrupt
  3  write admr   = b0    program talker only, t/r mode 3
  4  read  isr1   = 02?   check DO bit set
  5  read  isr2   = 00?
  6  read  adsr   = 42?   talker active, ATN unasserted
  7  write cdor   = aa    write out data byte
  8  ckint
  9  read  isr1   = 06?   check ERR, DO bits set
```

Error number : 6364 "Cannot interrupt on ERR"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on ERR

TEST 45

Purpose : Verify GPIB-PC2A can interrupt on DEC condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 08    enable DEC interrupt
  3  write admr   = 70    program listener only, t/r mode 3
  4  read  isr1   = 00?   check that interrupt registers
  5  read  isr2   = 00?     are cleared
  6  read  adsr   = 44?   check listener active, ATN unasserted
  7  write auxmr  = 1e    set IFC
  8  write auxmr  = 16    clear IFC
  9  read  isr2   = 08?   check CO bit set
 10  read  adsr   = 84?   check for controller-in-charge, ATN
                             asserted, and listener active
 11  write cdor   = 14    send device clear command
 12  write ckint
 13  read  isr1   = 08?   check GPIB-PC2A in Device Execute Clear
```

Error number : 6365 "Cannot interrupt on DEC"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on DEC

TEST 46

Purpose : Verify GPIB-PC2A can interrupt on END condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 10    enable interrupt on END IE
  3  write admr   = f0    set talker only, listener only,
                             t/r mode 3
  4  read  isr1   = 02?   check DO bit set
  5  read  isr2   = 00?
  6  read  adsr   = 46?   check listener and talker active
  7  read  auxmr  = 06?   send EOI with next byte
  8  write cdor   = 55    write data byte
  9  ckint
 10  read  isr1   = 13?   check END RX, DO, and DI bits set
```

Error number : 6366 "Cannot interrupt on END"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on END

TEST 47

Purpose : Verify GPIB-PC2A can interrupt on DET condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 20    enable interrupt on DET
  3  write admr   = 70    set listener only, t/r mode 3
  4  read  isr1   = 00?   check that interrupt registers are
  5  read  isr2   = 00?   clear
  6  read  adsr   = 44?   check listener active, ATN unasserted
  7  write auxmr  = 1e    set IFC
  8  write auxmr  = 16    clear IFC
  9  read  isr2   = 08?   check CO bit set
 10  read  adsr   = 84?   check for controller-in-charge, ATN
                            asserted, and  active listener
 11  write cdor   = 08    send group execute trigger command
 12  ckint
 13  read  isr1   = 20?   check that GPIB-PC2A is in device execute
                             trigger state
```

Error number : 6367 "Cannot interrupt on DET"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on DET

TEST 48

Purpose : Verify GPIB-PC2A can interrupt on APT condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 40     interrupt on APT
  3  write adr    = 00     set major primary address = 0
  4  write adr    = e0     disable minor address
  5  write admr   = 33     set address mode 3
  6  write auxmr  = 1e     set IFC
  7  write auxmr  = 16     clear IFC
  8  read  isr2   = 09?    check that CO and ADSC bits set
```

```
 9  read  adsr   = 80?    check for Controller-in-charge and ATN
10  write cdor   = 40     send my talk address
11  read  isr1   = 00?    check that
12  read  isr2   = 08?    CO bit is set
13  read  adsr   = 88?    check for controller-in-charge, ATN
                             asserted, talker primary addressed
14  write cdor   = 60     send my secondary address
15  ckint                 check that interrupt occurred
16  read  isr1   = 40?    check APT bit set
```

Error number : 6368 "Cannot interrupt on APT"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on APT

TEST 49

Purpose : Verify GPIB-PC2A can interrupt on CPT condition

```
Step
  1  init, write (2fX) = 00
  2  write imr1   = 80    enable cpt interrupt
  3  write admr   = 70    GPIB-PC2A listener only, t/r mode 3, mode 0
  4  write auxmr  = 1e    set IFC
  5  write auxmr  = 16    clear IFC
  6  write auxmr  = a1    enable command pass-through mode
  7  read  isr1   = 00?   check that only CO
  8  read  isr2   = 08?   bit is set
  9  read  adsr   = 84?   check for controller-in-charge, ATN
                            asserted, and active listener
 10  write cdor   = 02    or any other undefined command
 11  ckint                check that interrupt occurred
 12  read  isr1   = 80?   chect CPT bit set
 13  read  cptr   = 02?   read undefined command
```

Error number : 6369 "Cannot interrupt on CPT"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on CPT

TEST 50

Purpose : Verify GPIB-PC2A can interrupt on REMC condition

```
Step
  1  init, write (2fX) = 00
  2  write imr2   = 02    enable REMC interrupts
  3  write admr   = 31    set t/r mode 3, address mode 1
  4  write adr    = 00    set major address = 0
  5  write adr    = e0    disable minor address
  6  write auxmr  = 1e    set IFC
  7  write auxmr  = 16    clear IFC
  8  write auxmr  = 1f    set REN
  9  read  isr1   = 00?   check that CO and ADSC bits
 10  read  isr2   = 09?     are set
 11  read  adsr   = 80?   check controller-in-charge, ATN asserted
 12  write cdor   = 20    send my listen address
 13  ckint
 14  read  isr2   = 9b?   check INT, REM, REMC, and ADSC bits set
```

Error number : 6370 "Cannot interrupt on REMC"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on REMC

TEST 51

Purpose : Verify GPIB-PC2A can interrupt on LOKC condition

```
Step
  1  init, write (2fX) = 00
  2  write imr2   = 04    enable LOKC interrupts
  3  write admr   = 31    set t/r mode 3, address mode 1
  4  write adr    = 00    set major address = 0
  5  write adr    = e0    disable minor address
  6  write auxmr  = 1e    set IFC
  7  write auxmr  = 16    clear IFC
  8  write auxmr  = 1f    set REN
  9  read  isr1   = 00?   check CO and ADSC bits
 10  read  isr2   = 09?     are set
 11  read  adsr   = 80?   check controller-in charge, ATN asserted
 12  write cdor   = 20    send my listen address
 13  read  isr1   = 00?   check that REM, CO, REMC, and ADSC
 14  read  isr2   = 1b?     bits are set
 15  read  adsr   = 94?   check controller-in-charge, ATN asserted,
                          listener active, listener primary
                          addressed
 16  write cdor   = 11    send local lockout command
 17  ckint
 18  read  isr2   = bc?   check INT, LOK, and LOKC bits set
```

Error number : 6371 "Cannot interrupt on LOKC"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on LOKC

TEST 52

Purpose : Verify GPIB-PC2A can interrupt on SRQI condition

```
Step
  1  init, write (2fX) = 00
  2  write imr2   = 40    enable SRQI interrupts
  3  write admr   = f0    set talker only, listener only, mode 3
  4  write auxmr  = 1e    set IFC
  5  write auxmr  = 16    clear IFC
  6  read  isr1   = 00?   check CO bit set
  7  read  isr2   = 08?   check CO bit set
  8  write spmr   = 40    set SRQ
  9  ckint
 10  read  isr2   = c0?   check SRQI and INT bits set
```

Error number : 6372 "Cannot interrupt on SRQI"

Possible reasons for failure:

-7210 might be bad

If passed:

-7210 can interrupt on SRQI

TEST 53

Purpose : Verify GPIB-PC2A can DMA from GPIB to system memory and cause DMA TC interrupt

```
Step
  1  init, write (2fX) = 00
  2  write bpflop = 00     set byte pointer flipflop
  3  write mask   = 05     turn off DMA
  4  write mode   = 49     DMA read from memory
  5  write imr2   = 24     enable DMA out and DMA TC interrupt
  6  read  stat   = 20?    check DREQ driven low
  7  write admr   = f0     set talker only, listener only,
                              t/r mode 3
  8  read  adsr   = 46?    check talker active, listener active
  9  read  stat   = bit 5 set?  check DREQ driven high
 10  write chadr  = low byte of address
```

```
 11  write chadr  = next byte of address
 12  write page   = top nibble of address
 13  write chcnt  = low byte of count-1
 14  write chcnt  = high byte of count-1
 15  write mask   = 01       start dma
 16  check that no interrupt occurred
 17  check that DMA started on time
 18  check that data transferred correctly
 19  check that interrupt occurred after DMA completed
 20  check for no other interrupts

Error numbers:
  Step  1-15,
       17-19  : 6373     "Cannot interrupt on terminal count
                          on DMA to 7210"
  Step 16,20  : 6358     "Unevoked interrupt received"


Possible reasons for failure:

     -DMA terminal count circuitry not working
     -PAL might be bad
     -interrupt circuitry not working
     -7210 might be bad

If passed:

     -DMA terminal count circuitry working
```

TEST 54

```
Purpose : Verify GPIB-PC2A can DMA from system memory to GPIB and
     cause a DMA TC interrupt

     Step
       1  init, write (2fX) = 00
       2  write bpflop = 00      set byte pointer flipflop
       3  write mask   = 04|dmach  disable DMA
       4  write mode   = 04|dmach  dma write to memory
       5  write imr2   = 14   enable DMA in and DMA TC interrupts
       6  read  stat * drqmask = 00?  check DREQ driven low
       7  write admr   = f0   set talker only, listener only,
                              t/r mode 3
       8  read  adsr   = 46?  check active talker, active listener
       9  write chadr  = low byte of address
      10  write chadr  = next byte of address
      11  write page   = top nibble of address
      12  write chcnt  = 1f      low byte of count-1
      13  write chcnt  = 00      high byte of count-1
      14  write mask   = dmach   start DMA
      15  check that DMA began in the correct amount of time
      16  check that start of DMA did not cause interrupt
      17  check that data transferred correctly
      18  check that DMA caused an interrupt on completion
      19  check that interrupt was cleared
```

Error numbers :
  Step 1-15,
       17-18 : 6374   "Cannot interrupt on terminal count
                          on DMA from 7210"
  Step 16,19 : 6358   "Unevoked interrupt received"

Possible reasons for failure:

-7210 might be bad
-PAL might be bad

If passed:

-7210 can interrupt on terminal count on DMA from 7210

TEST 55

Purpose : Verify GPIB-PC2A does not cause DMA TC interrupt on other channel TC.

Step
  1 init, write (2fX) = 00
  2 write imr2   = 24   enable DMA and DMA TC interrupts
  3 wait 1 second and check that no DMA TC interrupt occurred

Error number : 6375   "Spurious DMA terminal count interrupt"

Possible reasons for failure:

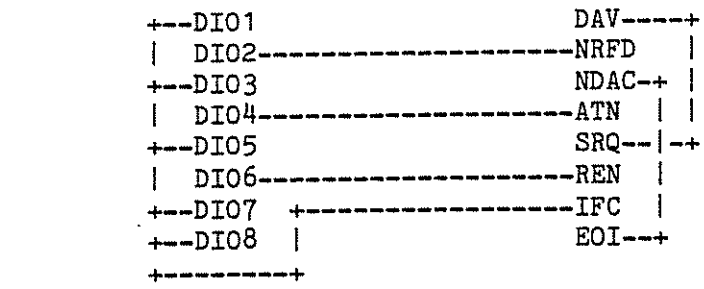
-TC line floating

If passed:

-TC line ok

PART II

This part requires a wraparound test plug with the following cross connections:



wire list:

DIO1 to DIO3 to DIO5 to DIO7 to DIO8 to IFC DIO2 to NRFD DIO4 to ATN DIO6 to REN DAV to SRQ NDAC to EOI

TEST 56

Purpose : See analysis section below

Step			
1	init		
2	write admr = 50		listen only, t/r mode 1, drv NDAC, rec EOI
3	write adr = 80		clear adr1
4	read adr1 = 00?		EOI bit not set yet
5	read cptr = 00?		drv rfd, rec ~dio2
6	write spmr = 40		drv SRQ, rec DAV, drv NRFD, rec dio2, drv dac
7	read cptr = 02?		drv NRFD, rec dio2
8	write adr1 = 80		rec EOI

Error number : 6380 "GPIB Signal Problem on EOI, DAV, SRQ, NRFD, OR DOI2"

analysis:

Programming the 7210 to be a listener asserts NDAC, which is tied to EOI. The EOI bit in adr1 is not set yet because DAV is not seen. The cptr shows that NRFD is false, as well as ATN, IFC, and REN. Setting the rsv bit in the spmr asserts SRQ, which is tied to DAV and simulates a data byte written from an external device. The 7210 responds by asserting NRFD, as seen in the cptr. The EOI bit in adr1 is now set as expected.

TEST 57

Purpose : See analysis section below

```
Step
  1  init
  2  write cdor  = 00
  3  write admr  = 10      t/r mode 1
  4  write auxmr = 1e      set IFC
  5  write auxmr = 1f      set REN
  6  write auxmr = 1d      rpp
  7  read  cptr  = fd?     drv IFC, rec dio8,7,5,3,1, drv REN,
                              rec dio6, drv ATN, REN dio4
```

Error number : 6381 "GPIB Signal Problem on IFC, REN, ATN, DIA8-3, or DIO1"

analysis:

Setting IFC makes the 7210 active controller and drives most data lines. Setting REN drives another. The data lines are set false to not interfere. The parallel poll response in the cptr shows that IFC, REN, and ATN are asserted correctly and the corresponding data lines are receiving correctly.

TEST 58

Purpose : See analysis section below

```
Step
  1  init
  2  write cdor  = 00
  3  write admr  = 10      t/r mode 1
  4  read  isr2  = 00?     no SRQ at this time
  5  write auxmr = 1e      set IFC
  6  write auxmr = 16      clear IFC
  7  write cdor  = 00      drv DAV
  8  read  isr2  = 49?     SRQI, rec SRQ, CO, ADSC
  9  write cdor  = 02      drv dio2, rec NRFD
 10  read  isr2  = 00?     no co
```

Error number : 6382 "GPIB Signal Problem on DAV, SRQ, NRFD, or DIO2"

analysis:

Writing to the codr after being made active controller asserts DAV which in turn simulates an SRQ input, which is noted in isr2. Thus DAV is asserted correctly and SRQ is received correctly. Another byte is written with dio2 set, which drives NRFD. The no co status indicates that NRFD is received correctly and dio2 is driven correctly, and the handshake lines are in a NRFD holdoff.

TEST 59

Purpose : See analysis section below

```
Step
  1  init
  2  write cdor  = 00
  3  write auxmr = 10     t/r mode 1
  4  write auxmr = 1e     set IFC
  5  write auxmr = 16     clear IFC
  6  write auxmr = 14     release control
  7  read  adsr  = 80?    controller-in-charge, ATN asserted
  8  write cdor  = 80     drv dio8, rec IFC
  9  read  adsr  = 40?    not controller-in-charge, ATN
                             unasserted
```

Error number : 6383 "GPIB Signal Problem on IFC or DIO8"

analysis:

By pulsing IFC and releasing control, the 7210 ends up as active controller and receiving IFC. By setting each data bit tied to IFC in turn, the controller function is reset to the idle state. Thus IFC is received correctly and the data lines are asserted correctly.

TEST 60

Purpose : See analysis section below

Same as TEST 59, except step 8 is "write cdor = 40"
Error number : 6384 "GPIB Signal Problem on IFC or DIO7"

TEST 61

Same as TEST 59, except step 8 is "write cdor = 10"
Error number : 6385 "GPIB Signal Problem on IFC or DIO5"

TEST 62

Same as TEST 59, except step 8 is "write cdor = 4"
Error number : 6386 "GPIB Signal Problem on IFC or DIO3"

TEST 63

Same as TEST 59 ,except step 8 is "write cdor = 1"
Error number : 6387 "GPIB Signal Problem on IFC or DIO1"

TEST 64

Purpose : See analysis section below

```
Step
  1  init
  2  write cdor = 00
```

```
3  write admr = 90        talk only, t/r mode 1
4  write cdor = 08        drv dio4, rec ATN
5  check adsr alternating between 02 and 42.
6  indicates adsr did not begin alternating.
```

Error number : 6388 "GPIB Signal Problem on ATN or DIO4"

analysis:

After programming the 7210 to be talker, dio4, which is tied to ATN, is asserted. This puts the t/r1 output of the 7210 into oscillation, as it first sees ATN set, switches from source to acceptor control of the 75160, which unasserts the data line and ATN, which allows the 7210 to switch back to source control, setting the data line and ATN, etc, etc. The ATN line oscillates at about 10 MHz with about a 50% duty cycle, and this can be seen by reading the adsr, which should read 42 and 02 with equal frequency.

TEST 65

Purpose : See analysis section below

```
Step
  1  init
  2  write cdor  = 00
  3  write admr  = 90       talk only, t/r mode 1
  4  read  isr1  = 06?      ERR, do
  5  write auxmr = 06       drv EOI on next byte, rec NDAC
  6  write cdor  = 00
  7  read  isr1  = 04       ERR
  8  write cdor  = 00
  9  read  isr1  = 02?      do
```

Error number : 6389 "GPIB Signal Problem on NDAC, EOI, NRFD, or DIO2"

analysis:

An error occurs by writing to the cdor when not a talker, but do is set because the bus is not in a holdoff. By driving EOI, which is tied to NDAC, on the next write, the 7210 will see either that an error exists when the cdor is loaded, but because EOI/NDAC is asserted, do is not set yet NDAC holdoff. The handshake lines are now in a legal state, NRFD unasserted and NDAC asserted. Writing to the cdor again clears EOI/NDAC, but because NDAC was seen when the cdor is loaded, no error occurs. The handshake finishes and a new do status occurs.

TEST 66

Purpose : See analysis section below

Step				
1	init			
2	write	cdor	= 00	
3	write	admr	= 11	t/r mode 1, address mode 1
4	write	adr	= 00	GPIB adr = 00
5	write	adr	= e0	disable secondary addressing
6	write	auxmr	= 1e	set IFC
7	write	auxmr	= 16	clear IFC
8	write	auxmr	= 14	release control
9	read	adsr	= 80?	controller-in-charge, ATN
10	write	cdor	= 24	drv dio6,dio4, rec REN
11	read	isr2	= 49?	SRQI, CO, ADSC
12	write	cdor	= 20	send my listen address, drive dio6, rec REN
13	read	isr2	= 5b?	SRQI, REM, CO, REMC, ADS

Error number : 6390 "GPIB Signal Problem on REN, DAV, SRQ, or DIO6"

analysis:

The cdor is first cleared to avoid conflict and the GPIB listen address is set to 20. Then the 7210 is made active controller and system control is released so that REN can be received. Writing 24 to the cdor sets the lines tied to REN and ATN. The latter is done so that another listen address is sent that does not cause a signal conflict. Next the 7210s listen address is sent, and since REN is set, a remote state change is detected. Thus REN is received correctly and dio6 is asserted correctly.

Appendix A

Parts List, PAL Drawings, and Schematic Diagram

This appendix contains the parts list, PAL drawings, and schematic diagram for the GPIB-PC2A.

ITEM NO	QTY REQD	MFR FSCM NO	PART/DWG STOCK NO	DESCRIPTION REFERENCE DESIGNATION
1	1	NI	180212-01	PWB, GPIB-PC/2A
		7U296	180212-01	
2	REF	NI	180211-01	SCHEMATIC, GPIB-PC/2A
		7U296	180211-01	
3	1	PANSNC	BR2325	BATTERY, 3V, 160MAH, LITHIUM COIN CELL
			724023-01	BT1
4	6	AVX	SA105E473ZAA	CAPACITOR, AXIAL LEAD, 50V, .047 UF
			715079-01	C1-6
5	1	CDE	CD6CA200D03	CAPACITOR, MICA, +-1/2PF, 100V, 20PF
			715047-01	C7
6	1	TUS	513011G7-40	CAP, VAR, 7-40PF
			715081-01	C8
7	4	SPG	199D106X0016CE2	CAPACITOR, RDL LEAD, 16 V, 20%, 10 UF
			715062-01	C9-12
8	1	SAN	D7-1-C-821-J-0	CAPACITOR, RDL LEAD, 100 V, 5%, 820 PF
		00853	715008-01	C13
9	4	GE	1N4148	DIODE, SILICON, SWITCHING
		15413	730001-01	D1-4
10	1	NI	180168-01	CONNECTOR, CHAMP, DISASSEMBLED
		7U296	180168-01	J1
11	1	GE	2N4403	TRANSISTOR, 40 V, 0.5 A, SWCHG, PNP
		15413	733001-01	Q1
12	1	GE	2N4401	TRANSISTOR, 40 V, 0.5 A, SWCHG, NPN
		15413	733000-01	Q2
13	3	-	RC05GF681J	RES, 680, 1/8W, 5%
			711149-01	R1, 2, 3
14	1	-	RC05GF103J	RES, 10K, 1/8W, 5%
			711150-01	R4
15	3	-	RC05GF333J	RES, 33K, 1/8W, 5%
			711151-01	R5, 8, 10
16	2	-	RC05GF274J	RES, 270K, 1/8W, 5%
			711152-01	R6, 9
17	1	-	RC05GF562J	RES, 5.6K, 1/8W, 5%
			711153-01	R7
18	1	-	RC05GF154J	RES, 150K, 1/8W, 5%
			711154-01	R11

TITLE
PARTS LIST- CCA, GPIB-PC/2A + CLOCK

FSCM NO DWG NO REV
PL 180210-02 A
Thu Oct 11 14:06:54 1984

ITEM NO	QTY REQD	MFR FSCM NO	PART/DWG STOCK NO	DESCRIPTION REFERENCE DESIGNATION
19	1	-	RC05GF206K	RES, 20M, 1/8W, 10%
			711155-01	R12
20	1	-	RC07GF103J	RESISTOR, 1/4 W, 5%, 10K OHM
		71785	711011-01	R13
21	1	CTS	750-101-R10K	RESISTOR, SIP, 10-PIN, 2%, 9 X 10K OHM
		00079	710006-01	R14
22	1	-	RC05GF470J	RES, 47, 1/8W, 5%
			711156-01	R15
23	1	TI	SN74LS02N	IC, QUAD 2-INPUT NOR GATE
		06668	700002-01	U1
24	1	TI	SN74LS08N	IC, QUAD 2-INPUT & GATE
		06668	700004-01	U2
25	1	TI	SN74LS51N	IC, 2-WIDE 2-INPUT &-OR-INVERT GATE
		06668	700012-01	U3
26	2	TI	SN74LS164N	IC, 8-BIT PARALLEL-OUT SER SHIFT REG
		06668	700097-01	U4, 12
27	1	TI	SN74LS74AN	IC, DUAL D-TYPE FF
		06668	700013-01	U5
28	2	TI	SN74LS00N	IC, QUAD 2-INPUT NAND GATE
		06668	700001-01	U6, 15
29	1	TI	SN74LS04N	IC, 14-PIN DIP, PLASTIC, HEX INVERTER
		06668	700003-01	U7
30	1	NI	700324-01	IC, PAL20R4CNS, 488 CTRL
		7U296	700324-01	U8
31	1	NS	MM58167AN	IC, 58167, REAL TIME CLOCK
			700322-01	U9
32	1	NEC	uPD7210	IC, GPIB INTERFACE CTRLR
			700200-01	U10
33	1	NS	DS75162AN	IC, OCTAL IEEE-488 CONTROL BUS XCVR
			700107-01	U11
34	2	TI	SN74LS32N	IC, QUAD 2-INPUT OR GATE
		06668	700011-01	U13, 14
35	1	NI	700323-01	IC, PAL16L2, ADDR DECODE
		7U296	700323-01	U16
36	1	AMP	435668-4	SWITCH, SPST, LOW PF, 5-POSITION, DIP
		00779	720001-01	U17

TITLE
PARTS LIST- CCA, GPIB-PC/2A + CLOCK

FSCM NO DWG NO REV
PL 180210-02 A
Thu Oct 11 14:06:54 1984

NATIONAL INSTRUMENTS SH 3 OF 5

ITEM NO	QTY REQD	MFR FSCM NO	PART/DWG STOCK NO	DESCRIPTION REFERENCE DESIGNATION
37	1	TI	SN74LS266N	IC, OPEN C OUT, QUAD 2-INPUT EXCL NOR
		06668	700263-01	U18
38	1	TI	SN74LS125AN	IC, QUAD BUS BFR GATES
		06668	700279-01	U19
39	1	TI	SN74LS541N	IC, OCTAL 3-STATE BFR/LINE DRVR
		06668	700077-01	U20
40	1	TI	SN74LS273N	IC, OCTAL D-TYPE FF W/ CLR
		06668	700237-01	U21
41	1	TI	SN74LS245N	IC, OCTAL 3-STATE BUS XCVR
		06668	700099-01	U22
42	1	NS	DS75160AN	IC, OCTAL IEEE-488 DATA BUS XCVR
			700106-01	U23
43	1	MPD	BH906	BATTERY HOLDER, LITHIUM COIN CELL
			745088-01	XBT1
44	1	STATEK	CX-1V 32.768KC	CRYSTAL, 0.1%, 32.768 KHZ
			724004-01	Y1
45	30	BERG	65500-101	HEADER, SGL ROW, STR, .1 CTR, 1 POSN
		22526	760013-01	
46	4	AMP	531220-3	CONNECTOR, 2 POSI, .1 CTR, MINI-JUMP
		00779	760014-02	
47	1	AMP	435238-2	COVER, PROT, SWITCH, 5 POSN, DIP
		00779	720008-01	
48	1	NI	180183-01	BRACKET, CONNECTOR MOUNTING, CHAMP
		7U296	180183-01	
49	1	NI	180188-01	EMI SHIELD, CONNECTOR, CHAMP
		7U296	180188-01	
50	2	ZIER	741	ANGLE BRACKET, 4-40 THD HOLE
			745100-01	
51	2	NI	180187-01	JACKSOCKET, CHAMP, METRIC, LONG
		7U296	180187-01	
52	2			WASHER, SPLIT LOCK, NO. 10, ZPS
			740406-01	
53	2			SCREW, 4-40 X 1/4, PNH, STAINLESS STL
			740911-01	
54	4			WASHER, INT TOOTH, NO. 4, STAINLESS STL
			740407-01	

ITEM NO	QTY REQD	MFR FSCM NO	PART/DWG STOCK NO	DESCRIPTION REFERENCE DESIGNATION
55	2		740010-01	NUT, HEX, 4-40 THD, STAINLESS STEEL

TITLE	FSCM NO	DWG NO	REV
PARTS LIST-, CCA, GPIB-PC/2A + CLOCK		PL 180210-02	A
		Thu Oct 11 14:06:54 1984	

NATIONAL INSTRUMENTS SH 5 OF 5

GPIB-PC2A
PAL DESCRIPTION
REV A

NI PART NO. 700324-01 made from PAL20R4. Manufacturer part no. MMI PAL20R4CN.

PROGRAM

PAL20R4 — PAL DESIGN SPECIFICATION
700324-01 REV A — Bob Summersett 7/18/84
TLC Instruction Decoder
NATIONAL INSTRUMENTS — AUSTIN — TEXAS

CK	RS2	GD7	GD6	GD5	GD4
GD3	GD2	GD1	GDO	/TLCSEL	GND
/OE	NC	/DSC, CRST	/RESET	/DMAEN	/SC
IE2	IE1	/IR2	RSO	RS1	VCC

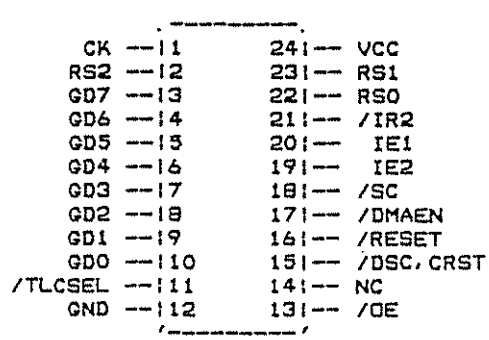
```
DSC, CRST = /GD7*/GD6*/GD5*GD4*/GD3*GD2*/GD1*/GDO
          + /GD7*/GD6*/GD5*/GD4*/DG3*/GD2*GD1*/GDO
DMAEN     = /RESET*IR2*GD5
          + /RESET*IR2*GD4
          + /RESET*/IR2*DMAEN
SC        = /RESET*RS2*/RS1*RSO*TLCSEL*/GD7*/GD6*/GD5*GD4*GD2*GD1
          + /RESET*RS2*/RS1*RSO*TLCSEL*/DSC, CRST*SC
          + /RESET*SC*/RS2
          + /RESET*SC*RS1
          + /RESET*SC*/RSO
          + /RESET*SC*/TLCSEL
IE2       = /IE2*/IR2
          + /IE2*/GD6*/GD3*/GD2*/GD1*/GDO
          + IR2*/GD6*/GD3*/GD2*/GD1*/GDO
          + RESET
IE1       = /IE1*RS2
          + /IE1*RS1
          + /IE1*/RSO
          + /IE1*/TLCSEL
          + /IE1*/GD7*/GD6*/GD5*/DG4*/GD3*/GD2*/GD1*/GDO
          + /RS2*/RS1*RSO*TLCSEL*/GD7*/GD6*/GD5*/GD4*/GD3*/GD2*/GD1*/GDO
          + RESET
IR2       = /RS2*RS1*/RSO*TLCSEL
```

DESCRIPTION

Programmed PAL must be stamped with NI part no. 700324-01A

TITLE	SIZE	FSCM NO	DWG NO	REV
PROGRAM, PAL20R4CNS-GPIB-PC/2A	A	7U296	700324-01	A
	NATIONAL INSTRUMENTS			SH 2 OF 3

Pinout for 700324-01



TITLE	SIZE	FSCM NO	DWG NO	REV
PROGRAM, PAL20R4CNS - GPIB-PC/2A	A	7U296	700324-01	A
	NATIONAL INSTRUMENTS			SH 3 OF 3

GPIB-PC2A
PAL DESCRIPTION
REV A

NI PART NO. 700323-01 made from PAL16L2. Manufacturer part no.
MMI PAL16L2CN.

PROGRAM

PAL16L2 — PAL DESIGN SPECIFICATION
700323-01 REV A — Bob Summersett 7/17/84
Address Decoder
NATIONAL INSTRUMENTS AUSTIN TEXAS

```
AEN   /WR   A6    A5    A4
A3    A2    A1    A0    GND
A8    A9    A7    NC    /2E1
/2FX  INT2  INT1  INT0  VCC
```

```
2E1 = /AEN*A9*/A8*A7*A6*A5*/A4*/A3*/A2*/A1*A0
2FX = /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*/INT2*/A2*/INT1*/A1*/INT0*/A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*/INT2*/A2*/INT1*/A1*INT0*A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*/INT2*/A2*INT1*A1*/INT0*/A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*/INT2*/A2*INT1*A1*INT0*A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*INT2*A2*/INT1*/A1*/INT0*/A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*INT2*A2*/INT1*/A1*INT0*A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*INT2*A2*INT1*A1*/INT0*/A0
    + /AEN*WR*A9*/A8*A7*A6*A5*A4*/A3*INT2*A2*INT1*A1*INT0*A0
```

DESCRIPTION

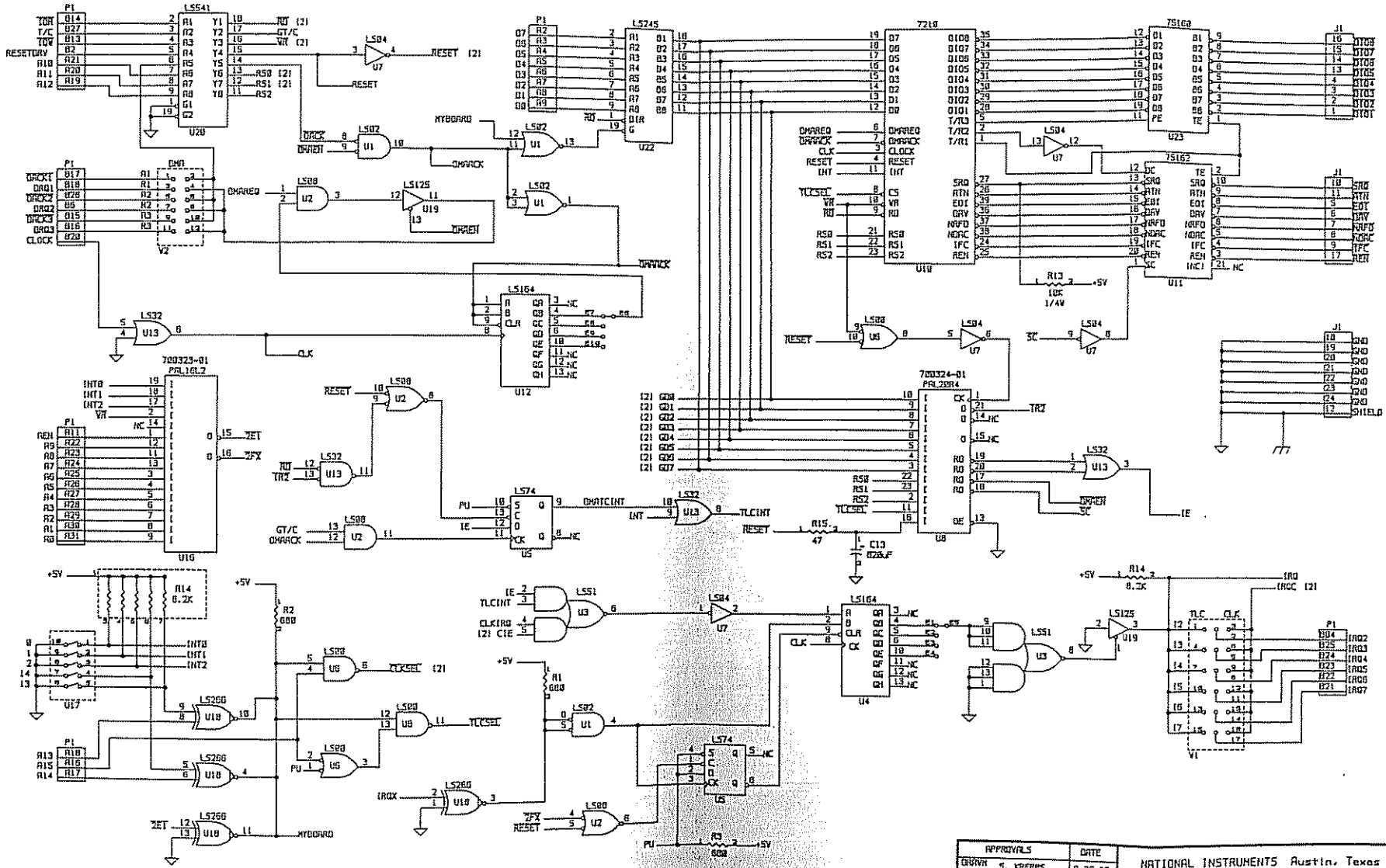
Programmed PAL must be stamped with NI part no. 700323-01A

Pinout for 700323-01

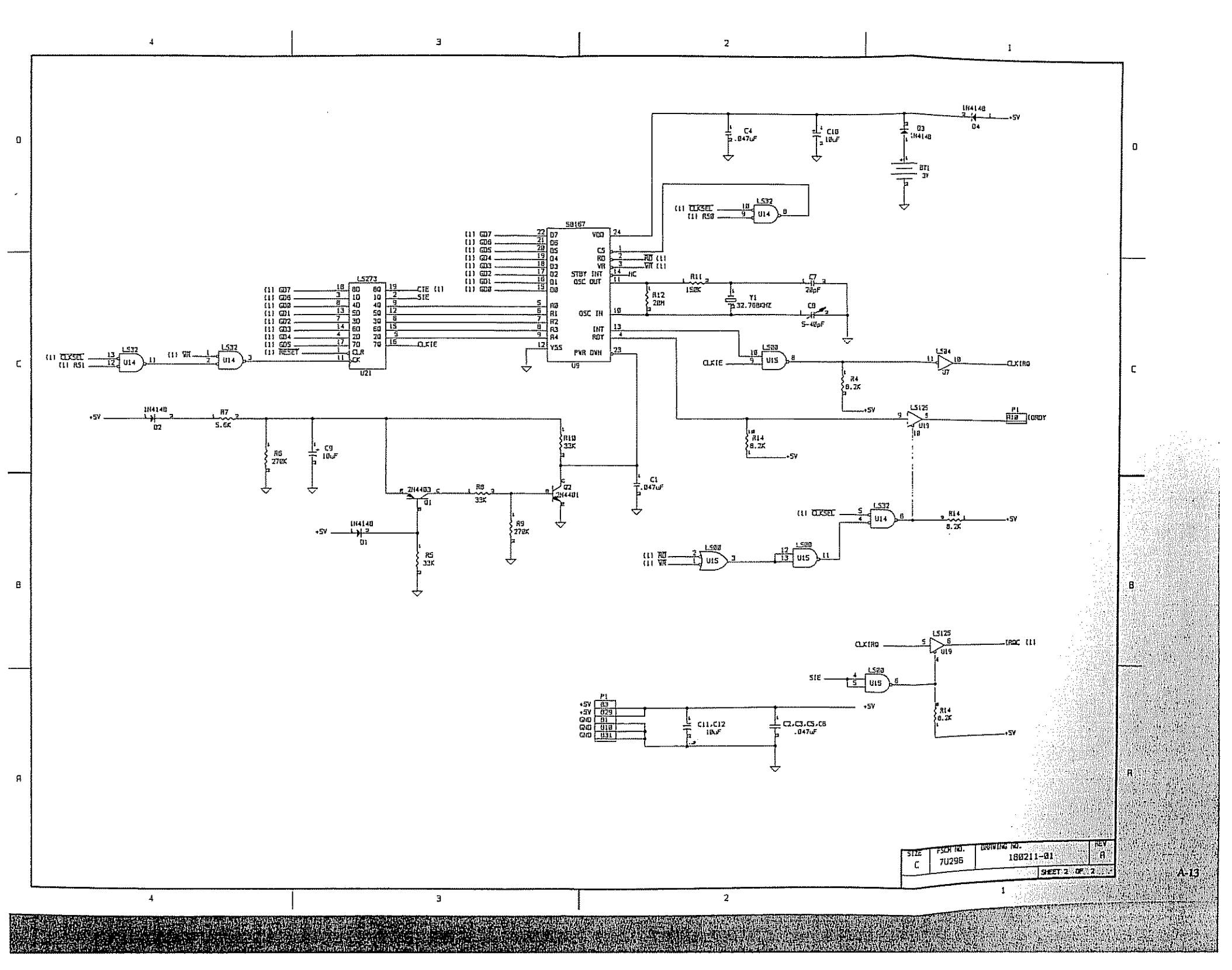
```
          .---------.
AEN --| 1       20 |-- VCC
/WR --| 2       19 |-- INT0
 A6 --| 3       18 |-- INT1
 A5 --| 4       17 |-- INT2
 A4 --| 5       16 |-- /2FX
 A3 --| 6       15 |-- /2E1
 A2 --| 7       14 |-- NC
 A1 --| 8       13 |-- A7
 A0 --| 9       12 |-- A9
GND --| 10      11 |-- A8
          '---------'
```

TITLE	SIZE	FSCM NO	DWG NO	REV
PROGRAM, PAL16L2CN-GPIB-PC/2A	A	7U296	700323-01	A
	NATIONAL INSTRUMENTS			SH 2 OF 2

REVISIONS				
ZONE	LTR	DESCRIPTION	DATE	APPROVED
	A	DRAWING RELEASED	6-27-84	V. HOWLIN
	A	REDRAWN WITHOUT CHANGE	6-27-84	B. Summerlin



1. ALL RESISTOR VALUES ARE IN OHMS, 1/8W
NOTES: (UNLESS OTHERWISE SPECIFIED)


4
3
2
1
0
C
B
A
1N4148
D4
+5V
C4
.047uF
C10
10uF
D3
1N4148
BT1
3V
(1) CLKSEL
(1) RS0
LS32
U14
58167
VDD
CS
RD
WR
STBY INT
OSC OUT
OSC IN
INT
RDY
PWR DWN
VSS
U9
(1) GD7
(1) GD6
(1) GD5
(1) GD4
(1) GD3
(1) GD2
(1) GD1
(1) GD0
LS273
U21
CLR
CK
(1) RESET
CLKIE
SIE
(1) RS1
(1) WR
R11
150K
R12
20M
Y1
32.768KHZ
C7
20pF
C8
5-40pF
LS08
U15
LS04
U7
CLKIRQ
R4
8.2K
LS125
U19
P1
(1)RDY
R14
8.2K
D2
R7
5.6K
R6
270K
C9
10uF
R10
33K
C1
.047uF
2N4403
Q1
R8
33K
Q2
2N4401
R9
270K
D1
R5
33K
(1) RD
(1) WR
(1) TRQC
C11,C12
10uF
C2,C3,C5,C6
.047uF
SIZE
C
FSCM NO.
7U296
DRAWING NO.
180211-01
REV
A
SHEET 2 OF 2
A-13

APPENDIX B

MULTILINE INTERFACE COMMAND MESSAGES

(Sent and Received with ATN TRUE)

The following tables are multiline interface messages (sent and received with ATN TRUE).

APPENDIX B

MULTILINE INTERFACE COMMAND MESSAGES

(Sent and Received with ATN TRUE)

Hex	Octal	Decimal	ASCII	Message	Hex	Octal	Decimal	ASCII	Message
00	000	0	NUL		20	040	32	SP	MLA
01	001	1	SOH	GTL	21	041	33	!	MLA
02	002	2	STX		22	042	34	"	MLA
03	003	3	ETX		23	043	35	#	MLA
04	004	4	EOT	SDC	24	044	36	$	MLA
05	005	5	ENQ	PPC	25	045	37	%	MLA
06	006	6	ACK		26	046	38	&	MLA
07	007	7	BEL		27	047	39	'	MLA
08	010	8	BS	GET	28	050	40	(	MLA
09	011	9	HT	TCT	29	051	41	)	MLA
0A	012	10	LF		2A	052	42	*	MLA
0B	013	11	VT		2B	053	43	+	MLA
0C	014	12	FF		2C	054	44	,	MLA
0D	015	13	CR		2D	055	45	-	MLA
0E	016	14	SO		2E	056	46	.	MLA
0F	017	15	SI		2F	057	47	/	MLA
10	020	16	DLE		30	060	48	0	MLA
11	021	17	DC1	LLO	31	061	49	1	MLA
12	022	18	DC2		32	062	50	2	MLA
13	023	19	DC3		33	063	51	3	MLA
14	024	20	DC4	DCL	34	064	52	4	MLA
15	025	21	NAK	PPU	35	065	53	5	MLA
16	026	22	SYN		36	066	54	6	MLA
17	027	23	ETB		37	067	55	7	MLA
18	030	24	CAN	SPE	38	070	56	8	MLA
19	031	25	EM	SPD	39	071	57	9	MLA
1A	032	26	SUB		3A	072	58	:	MLA
1B	033	27	ESC		3B	073	59	;	MLA
1C	034	28	FS		3C	074	60	<	MLA
1D	035	29	GS		3D	075	61	=	MLA
1E	036	30	RS		3E	076	62	>	MLA
1F	037	31	US		3F	077	63	?	UNL

MULTILINE INTERFACE COMMAND MESSAGES

(Sent and Received with ATN TRUE)

Hex	Octal	Decimal	ASCII	Message	Hex	Octal	Decimal	ASCII	Message
40	100	64	@	MTA	60	140	96	`	MSA,PPE
41	101	65	A	MTA	61	141	97	a	MSA,PPE
42	102	66	B	MTA	62	142	98	b	MSA,PPE
43	103	67	C	MTA	63	143	99	c	MSA,PPE
44	104	68	D	MTA	64	144	100	d	MSA,PPE
45	105	69	E	MTA	65	145	101	e	MSA,PPE
46	106	70	F	MTA	66	146	102	f	MSA,PPE
47	107	71	G	MTA	67	147	103	g	MSA,PPE
48	110	72	H	MTA	68	150	104	h	MSA,PPE
49	111	73	I	MTA	69	151	105	i	MSA,PPE
4A	112	74	J	MTA	6A	152	106	j	MSA,PPE
4B	113	75	K	MTA	6B	153	107	k	MSA,PPE
4C	114	76	L	MTA	6C	154	108	l	MSA,PPE
4D	115	77	M	MTA	6D	155	109	m	MSA,PPE
4E	116	78	N	MTA	6E	156	110	n	MSA,PPE
4F	117	79	O	MTA	6F	157	111	o	MSA,PPE
50	120	80	P	MTA	70	160	112	p	MSA,PPD
51	121	81	Q	MTA	71	161	113	q	MSA,PPD
52	122	82	R	MTA	72	162	114	r	MSA,PPD
53	123	83	S	MTA	73	163	115	s	MSA,PPD
54	124	84	T	MTA	74	164	116	t	MSA,PPD
55	125	85	U	MTA	75	165	117	u	MSA,PPD
56	126	86	V	MTA	76	166	118	v	MSA,PPD
57	127	87	W	MTA	77	167	119	w	MSA,PPD
58	130	88	X	MTA	78	170	120	x	MSA,PPD
59	131	89	Y	MTA	79	171	121	y	MSA,PPD
5A	132	90	Z	MTA	7A	172	122	z	MSA,PPD
5B	133	91	[	MTA	7B	173	123	{	MSA,PPD
5C	134	92	\	MTA	7C	174	124	|	MSA,PPD
5D	135	93	]	MTA	7D	175	125	}	MSA,PPD
5E	136	94	^	MTA	7E	176	126	~	MSA,PPD
5F	137	95	_	UNT	7F	177	127	DEL	

Appendix C

Mnemonics Key

Mnemonic	Type*	Full Name
ACDS	ST	Acceptor Data State (AH function)
ACG	C	Addressed Command Group
ACRS	ST	Acceptor Ready State
AD1-0	B	TLC GPIB Address Bit 1 (Mode 2)
AD1-1	B	TLC GPIB Secondary Address Bit 1 (Mode 2)
AD2-0	B	TLC GPIB Address Bit 2 (Mode 2)
AD2-1	B	TLC GPIB Secondary Address Bit 2 (Mode 2)
AD3-0	B	TLC GPIB Address Bit 3 (Mode 2)
AD3-1	B	TLC GPIB Secondary Address Bit 3 (Mode 2)
AD4-0	B	TLC GPIB Address Bit 4 (Mode 2)
AD4-1	B	TLC GPIB Secondary Address Bit 4 (Mode 2)
AD5-0	B	TLC GPIB Address Bit 5 (Mode 2)
AD5-1	B	TLC GPIB Secondary Address Bit 5 (Mode 2)
ADCS	B	Address Status Change
ADCS IE	B	Address Status Change Interrupt Enable
ADM0	B	Address Mode Bit 0
ADM1	B	Address Mode Bit 1
ADMR	R	Address Mode Register
ADR	R	Address Register
ADR0	R	Address Register 0
ADR1	R	Address Register 1
ADSR	R	Address Status Register
AH	ST	Acceptor Handshake
AIDS	ST	Acceptor Idle State
ANRS	ST	Acceptor Not Ready State
APRS	ST	Affirmative Poll Response State
APT	B	Address Pass Through
APT	B	Address Pass Through
APT IE	B	Address Pass Through Interrupt Enable
ARS	B	Address Register Select
ATN	SL	Attention
ATN*	B	Attention
AUXMR	R	Auxiliary Mode Register
AWNS	ST	Acceptor Wait for New cycle State
BIN	B	Binary
C	F	Controller
CACS	ST	Controller Active State (C function)
CADS	ST	Controller Addressed State
CAWS	ST	Controller Active Wait State
CDOR	R	Control/Data Out Register
CDO[0-7]	B	Control/Data Out Bits 0-7

*Key: F=Function, RM=Remote Message, LM=Local Message, ST=State, B=Bit, R=Register

Mnemonic	Type*	Full Name
CIC	B	Controller-In-Charge
CIDS	ST	Controller Idle State
CLK0	B	Clock Bit 0
CLK1	B	Clock Bit 1
CLK2	B	Clock Bit 2
CLK3	B	Clock Bit 3
CNT0	B	Control Code Bit 0
CNT1	B	Control Code Bit 1
CNT2	B	Control Code Bit 2
CO	B	Command Output
CO IE	B	Command Output Interrupt Enable
COM0	B	Command Code Bit 0
COM1	B	Command Code Bit 1
COM2	B	Command Code Bit 2
COM3	B	Command Code Bit 3
COM4	B	Command Code Bit 4
CPPS	ST	Controller Parallel Poll State
CPT	B	Command Pass Through
CPT ENAB	B	Command Pass Through Enable
CPT IE	B	Command Pass Through Interrupt Enable
CPTR	R	Command Pass Through Register
CPT[0-7]	B	Command Pass Through Bits 0-7
CPWS	ST	Controller Parallel Poll Wait State
CSBS	ST	Controller Standby State
CSHS	ST	Controller Standby Hold State
CSNS	ST	Controller Service Not Requested State
CSRS	ST	Controller Service Requested State
CSWS	ST	Controller Synchronous Wait State
CTRS	ST	Controller Transfer State (C function)
DAB	RM	Data Byte
DAC	RM	Data Accepted
dacr		DAC holdoff release
DAG		Device Address Group
DAV	SL	Data Valid
DC		Device Clear
DCAS	ST	Device Clear Active State
DCIS	ST	Device Clear Idle State
DCL	RM	Device Clear
DEC	B	Device Clear
DEC IE	B	Device Clear Interrupt Enable
DET	B	Device Execute Trigger
DET IE	B	Device Execute Trigger Interrupt Enable
DHDC	B	DAC Holdoff on DCAS
DHDT	B	DAC Holdoff on DTAS
DI	B	Data In
DI IE	B	Data In Interrupt Enable
DIR	R	Data In Register

*Key: F=Function, RM=Remote Message, LM=Local Message, ST=State, B=Bit, R=Register

Mnemonics Key (continued)

Mnemonic	Type*	Full Name
DI[0-7]	B	Data In Bits 0-7
DL	B	Disable Listener
DL0	B	Disable Listener 0
DL1	B	Disable Listener 1
DMAI	B	DMA Input Enable
DO	B	Data Out
DO IE	B	Data Out Interrupt Enable
DT		Device Trigger
DT	B	Disable Talker
DT0	B	Disable Talker 0
DT1	B	Disable Talker 1
DTAS	ST	Device Trigger Active State
DTIS	ST	Device Trigger Idle State
END	RM	End
END IE	B	END Received Interrupt Enable
END RX	B	END Received
EOI	B	END or Identify
EOI	SL	End Or Identify
EOS	RM	End Of String
EOSR	R	End of String Register
EOS[7-0]	B	End of String Bits 7 to 0
ERR	B	Error
ERR	RM	Error
ERR IE	B	Error Interrupt Enable
GET	RM	Group Execute Trigger
GND	SX	Ground
GTL	RM	Go To Local
gts	LM	Go To Standby
HLDA	B	Holdoff on All
HLDE	B	Holdoff on END
IDY	RM	Identify
IFC	RM	Interface Clear
IMR1	R	Interrupt Mask Register 1
IMR2	R	Interrupt Mask Register 2
INT	B	Interrupt
INV	B	Invert
IORD*	SX	I/O Read
IOWRT*	SX	I/O Write
ISR1	R	Interrupt Status Register 1
ISR2	R	Interrupt Status Register 2
ISS	B	Individual Status Select
ist		Individual Status
L	F	Listen
LA	B	Listener Active
LACS	ST	Listener Active State (L function)
LADS	ST	Listener Addressed State (L function)
LAG	RM	Listen Address Group

*Key: F=Function, RM=Remote Message, LM=Local Message, ST=State, B=Bit, R=Register

Mnemonic	Type*	Full Name
LE	F	Extended Listen
LIDS	ST	Listener Idle State
LLO	RM	Local Lockout
LOCS	ST	Local State
LOK	B	Lockout
LOKC	B	Lockout Change
LOKC IE	B	Lockout Change Interrupt Enable
lon	B	Listener Only
lon	LM	Listen Only
LPAS	B	Listener Primary Addressed State
LPAS	ST	Listener Primary Addressed State
lpe	LM	Local Poll Enabled
LPIS	ST	Listener Primary Idle State
ltn	LM	Listen
lun	LM	Local Unlisten
LWLS	ST	Local With Lockout State
MCLK*	SX	SBX Clock
MCS0A*	SX	Chip Select 0, Connector A
MCS0B*	SX	Chip Select 0, Connector B
MCS0C*	SX	Chip Select 0, Connector C
MCS1A*	SX	Chip Select 1, Connector A
MCS1B*	SX	Chip Select 1, Connector B
MCS1C*	SX	Chip Select 1, Connector C
MD7-MD0	SX	Data Lines
MDACK*	SX	DMA Acknowledge
MDRQT	SX	DMA Request
MINTRA0	SX	Interrupt 0, Connector A
MINTRA1	SX	Interrupt 1, Connector A
MINTRB0	SX	Interrupt 0, Connector B
MINTRB1	SX	Interrupt 1, Connector B
MINTRC0	SX	Interrupt 0, Connector C
MINTRC1	SX	Interrupt 1, Connector C
MJMN	B	Major-Minor
MLA	RM	My Listen Address
MPST*	SX	Present
MSA	RM	My Secondary Address
MTA	RM	My Talk Address
nba	LM	New Byte Available
NPRS	ST	Negative Poll Response State
NUL	RM	Null byte
OSA	RM	Other Secondary Address
OTA	RM	Other Talk Address
P1	B	PPR Response Bit 1
P2	B	PPR Response Bit 2
P3	B	PPR Response Bit 3
PACS	ST	Parallel Poll Addressed to Configure State
PCG	RM	Primary Command Group

*Key: F=Function, RM=Remote Message, LM=Local Message, ST=State, B=Bit, R=Register

Mnemonics Key (continued)

Mnemonic	Type*	Full Name
PEND	B	Pending
pof	LM	Power Off
pon	LM	Power On
PP		Parallel Poll (scan all status flags)
PPAS	ST	Parallel Poll Active State
PPC	RM	Parallel Poll Configure
PPD	RM	Parallel Poll Disable
PPE	RM	Parallel Poll Enable
PPIS	ST	Parallel Poll Idle State
PPR		Parallel Poll Response
PPSS	ST	Parallel Poll Standby Active
PPU	RM	Parallel Poll Unconfigure
PUCS	ST	Parallel Poll Unaddressed to Configure State
rdy	LM	Ready for next message
REM	B	Remote
REMC	B	Remote Change
REMC IE	B	Remote Change Interrupt Enable
REMS	ST	Remote State
REN	RM	Remote Enable
REOS	B	END on EOS Received
RFD	RM	Ready For Data
RL	F	Remote/Local
rpp	LM	Request Parallel Poll
RQS	RM	Request Service
rsc	LM	Request System Control
rsv	B	Request Service
rsv	LM	Request Service
rtl	LM	Return To Local
RWLS	ST	Remote With Lockout State
S	B	Sense
SACS	ST	System Control Active State
SCG		Secondary Command Group
SDC	RM	Selected Device Clear
SDYS	ST	Source Delay State
SGNS	ST	Source Generate State
SH	F	Source Handshake
SIAS	ST	System Control Interface Clear Active State
sic	LM	Send Interface Clear
SIDS	ST	Source Idle State
SIIS	ST	System Control Interface Clear Idle State
SINS	ST	System Control Interface Clear Not Active State
SIWS	ST	Source Idle Wait State
SNAS	ST	System Control Not Active State
SP	F	Serial Poll (scanning flags)
SPAS	ST	Serial Poll Active State (T function)
SPD	RM	Serial Poll Disable
SPE	RM	Serial Poll Enable

*Key: F=Function, RM=Remote Message, LM=Local Message, ST=State, B=Bit, R=Register

Mnemonic	Type*	Full Name
SPEOI	B	Send Serial Poll EOI
SPIS	ST	Serial Poll Idle State
SPMR	R	Serial Poll Mode Register
SPMS	B	Serial Poll Mode State
SPMS	ST	Serial Poll Mode State
SPSR	R	Serial Poll Status Register
SR	F	Service Request
SRAS	ST	System Control Remote enable Active State
sre	LM	Send Remote Enable
SRIS	ST	System Control Remote Enable Idle State
SRNS	ST	System Control Remote Enable Not Active State
SRQ	RM	Service Request
SRQI	B	Service Request Input
SRQI IE	B	Service Request Input Interrupt Enable
SRQS	ST	Service Request State
STB	RM	Status Byte
STRS	ST	Source Transfer State
SWNS	ST	Source Wait for New Cycle State
T	F	Talk
TA	B	Talker Active
TACS	ST	Talker Active State (T function)
TADS	ST	Talker Addressed State
TAG	RM	Talk Address Group
tca	LM	Talk Control Asynchronously
tcs	LM	Take Control Synchronously
TCT	RM	Take Control
TDMA	SX	Terminate DMA
TE	F	Extended Talk
TIDS	ST	Talker Idle State
TLC		Talker/Listener/Controller (GPIB Adapter)
ton	B	Talker Only
ton	LM	Talk Only
TPAS	B	Talker Primary Addressed State
TPAS	ST	Talker Primary Addressed State
TPIS	ST	Talker Primary Idle State
TRI	B	Three-State Timing
TRM0	B	Transmit/Receive Mode Bit 0
TRM1	B	Transmit/Receive Mode Bit 1
U	B	Unconfigure
UCG	RM	Universal Command Group
UNL	RM	Unlisten command
UNT	RM	Untalk command
XEOS	B	Transmit END with EOS

*Key: F=Function, RM=Remote Message, LM=Local Message, ST=State, B=Bit, R=Register

User Comment Form

National Instruments encourages you to give us your comments on the documentation supplied with its products. This information helps us provide quality products to meet your needs.

Title: **GPIB-PCIIA Technical Reference Manual**

Edition Date: **September 1989**

Part Number: **320045-01**

Please comment on the completeness, clarity, and organization of the manual.

If you find errors in the manual, please record the page numbers and describe the errors.

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Phone (_____)_____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039