MC68488UM(AD)

MOTOROLA
Semiconductor Products

MC68488
GPIA
USER'S MANUAL

Including an Introduction to the IEEE-Standard 488-1978 for Instrumentation Interface

FOREWORD

The adoption of IEEE Standard 488-1978 is a significant step toward creation of universal instrumentation systems approach. This standard allows easy systems configuration involving instruments produced by different manufacturers. As such, it greatly simplifies the design of production testing, systems control or scientific data recording systems.

Quite naturally, the IEEE-Standard has generated a tremendous amount of interest among both instrument and computer manufacturers and measurement equipment users. Among the restraining factors on 488-1978 implementation have been the complexity of the logic protocol and the inavailability of low-cost ICs to satisfy the required logic control needs.

A multifunctional intelligent LSI device (MC68488) has been developed by Motorola to provide the interface between the IEEE-488 Standard bus (GPIB — General Purpose Interface Bus) and a parallel computer bus. This complex IC chip functions as an entire block allowing easy interaction with the GPIB. It performs many of the interface tasks automatically, reducing the software burden of the processor and improving throughput.

This manual describes the IEEE-488 Standard, and discusses Motorola's MC68488 (GPIA — General Purpose Interface Adapter) for system implementation. The document is intended for the prospective user as well as the experienced instrumentation designer. It can be used as a tutorial presentation, a detailed user's manual, or a reference guide.

This manual does not cover the IEEE Standard 488-1978 in its entirety. Instead it supplements and highlights the standard rather than replacing it. It will be assumed that the reader has access to copies of the complete standard for a detailed technical reference on the subject. Portions of this document have been taken from IEEE STD-1978 Digital Interface for Programmable Instrumentation, with permission of the Institute of Electrical and Electronics Engineers, Inc. Copies of IEEE STD 488-1978 may be purchased from IEEE, 345 East 47th Street, New York, N.Y. 10017, or from the American National Standards Institute, 1430 Broadway, New York, N.Y. 10018.

TABLE OF CONTENTS

TABLE OF CONTENTS
(CONTINUED)

TABLE OF CONTENTS
(CONCLUDED)

LIST OF FIGURES

LIST OF FIGURES
(CONTINUED)

LIST OF FIGURES
(CONCLUDED)

LIST OF TABLES

CHAPTER 1
INTRODUCTION

Communications between instruments, terminals, mass storage devices, etc., in an instrumentation system is accomplished by interconnecting these devices to a common bus. This means each device needs to be interfaced so its logic convention, signal levels, timing protocols, etc. are compatible with those of the other instruments in the system.

1.1 TRADITIONAL APPROACH TO INTERFACE DESIGN

The traditional approach to interface design has been to provide each device (instrument) with a set of specialized controls, data, and status signals. A fairly easy straight-forward design was provided to achieve direct control of an instrument's programmable features; however, a problem resulted. The choice of signal lines, status lines, etc., which were to be made available, and the types of timing transitions between these lines were decided upon by the individual design engineer. Thus, there were about as many different interfacing techniques as design engineers. The net result was a dedicated interface structure for each instrument used in a system. Not only were interface structures for different manufacturers incompatible, but many instruments from the same manufacturer were incompatible. To accommodate all of the different structures (logic levels, signal levels, timing, etc.) many different adapters were built (Figure 1-1). This created a system with a mass of interconnecting cables and black boxes. As the sophistication of instrumentation systems increased — due mainly to the increase of highly programmable instruments — so did the complexity of the interface adapters. The resulting interface cost and complexity to achieve required performance levels, became unmanageable. The effort put into the wide variety of hardware design and the implementation of the necessary software was enormous — an interface standard was definitely needed.

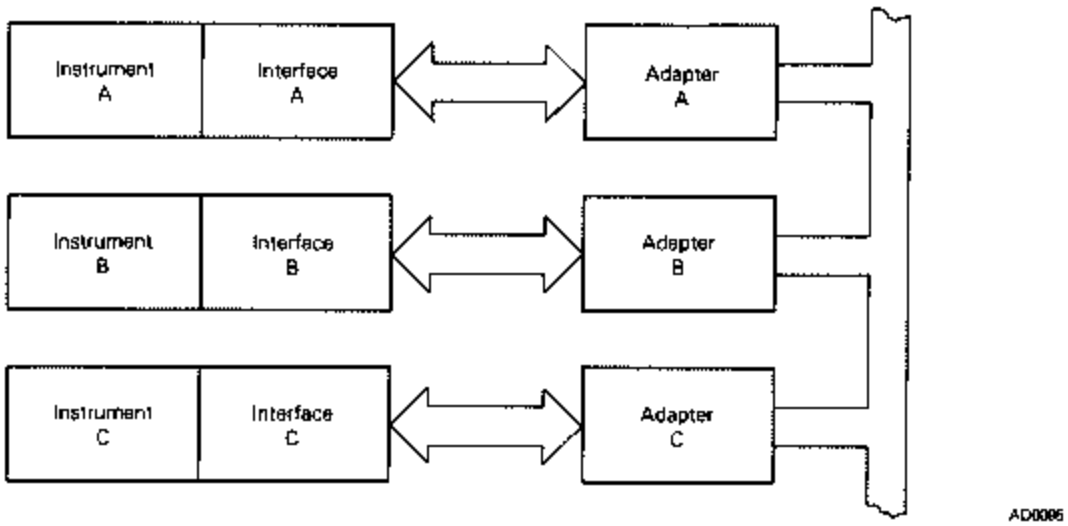


Figure 1-1. Instrumentation System Prior to Standardization

1.2 PURPOSE OF IEEE-488 STANDARD

The purpose of the IEEE-488 Standard is to allow for interconnection of programmable instruments with minimum engineering. Its intent is to remove the need for adapters and numerous types of patching cables. The IEEE-488 Standard allows system configurations using programmable instruments, calculators, and other types of peripheral devices produced by different manufacturers (Figure 1-2). The IEEE-488 Standard provides a set of rules for establishing an unambiguous communications link, producing a high degree of compatibility and yet maintaining flexibility between independently manufactured products.

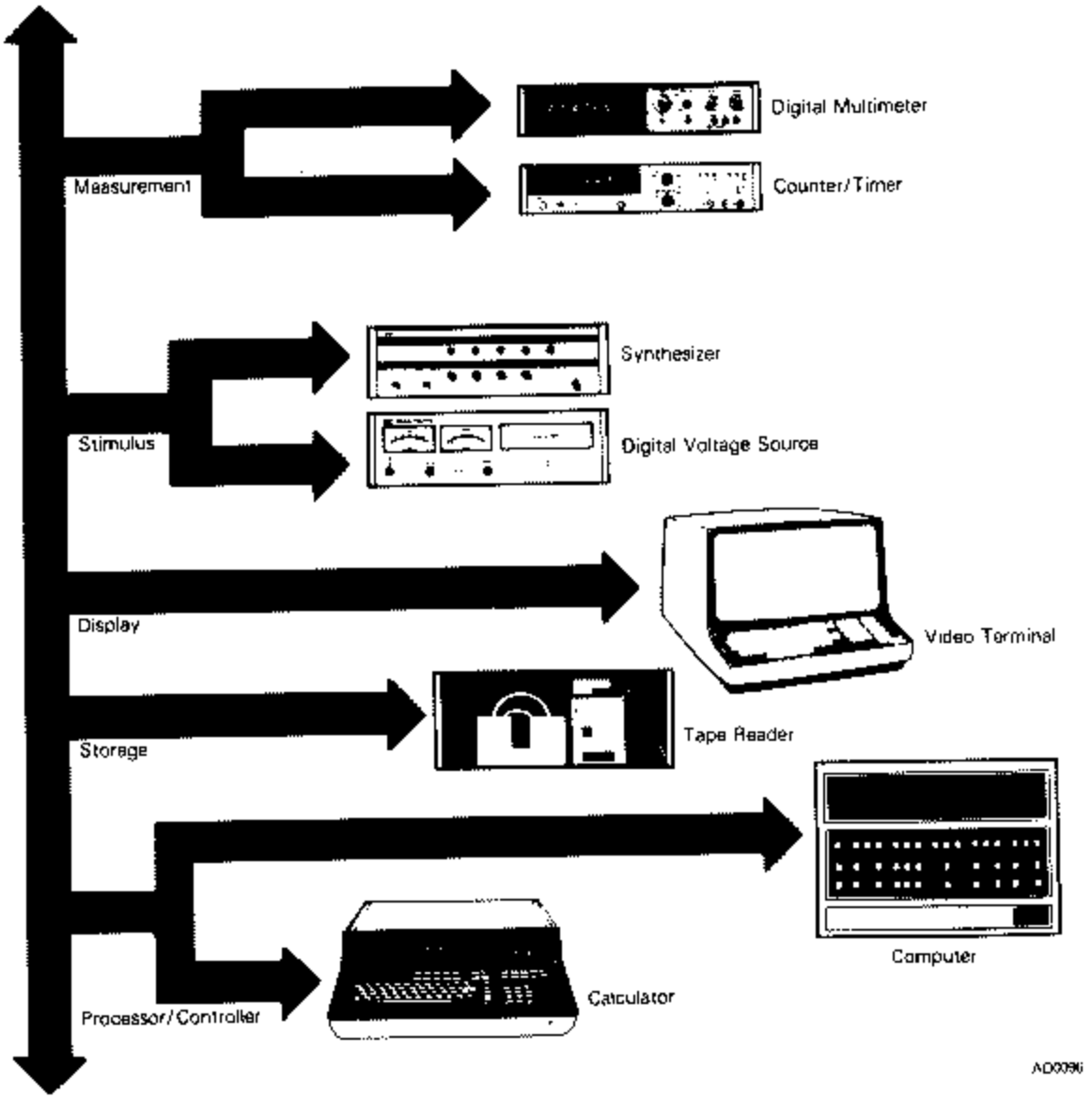


Figure 1-2. System Configuration

The IEEE-488 Standard defines the:

- Electrical Characteristics — driver and receiver circuit parameters, logical and electrical signal levels, loading requirements, and ground requirements;
- Mechanical Characteristics — connector type, contact assignments, and cable assembly;
- Functional Characteristics — the complete repertoire and precise definition of each of the signal lines, the protocol and timing relationships used to transfer all messages across the interface and the response expected as a result of receiving these messages. There are ten defined functions to be performed by the interface. These will be covered in Chapter 2.

The IEEE-488 Standard defines a special bus structure known as the GPIB (General Purpose Interface Bus). Any device meeting the specifications described in the IEEE-488 Standard will be directly compatible with the GPIB without the need of an adapter (Figure 1-3). Motorola's MC68488 General Purpose Interface Adapter (GPIA) meets these requirements.

The standard applies to interface systems used to interconnect both programmable and non-programmable electronic measuring apparatus with other apparatus and accessories necessary to assemble instrumentation systems.

It applies to the interface of instrumentation systems or portions of them, in which the:

1. Data exchange between the interconnecting apparatus is digital (as distinct from analog);
2. Number of devices that may be interconnected by any contiguous bus does not exceed 15;
3. Total transmission path length over the interconnecting cables does not exceed 20 meters;
4. Data rate across the interface on any signal line does not exceed one Mega byte/second.

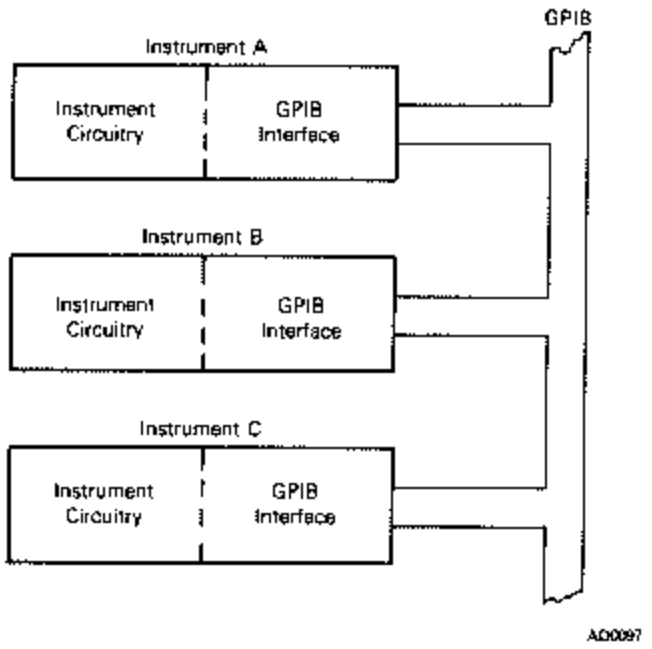


Figure 1-3. GPIB System (Adapters Not Needed)

The basic functional specifications of the IEEE-488 Standard may be used in digital interface applications which require longer distance, more devices, environments requiring increased noise immunity, or combinations of these parameters. Different electrical and mechanical specifications may be required (for example, symmetrical circuit configurations, high threshold logic, special connectors, or cable configurations) for those extended applications. However, applications of this type are not covered and are beyond the scope of the IEEE-488 Standard.

The IEEE-488 Standard is also applicable to other instrumentation system elements such as processors, stimulus units, display units, terminal units or mass storage devices. It applies generally to laboratory and production test or any other test environment which is both electrically quiet and restricted as to physical dimensions (distance between the system components).

The IEEE-488 Standard only specifies the operation of the GPIB and interface. It assures that data and control information will be transferred without error between the communication instruments. It does not specify the way this information will be interpreted by the instrument. It is up to the user to choose the type of data format and instrument control messages.

1.3 OVERVIEW OF THE IEEE-488 STANDARD

The GPIB can be thought of as the communications link between two or more instruments (Figure 1-4). This communications link is a parallel bus as opposed to the serial links associated with most other types of data communications. The transmission is that of a bit-parallel, byte-serial format. Bit-parallel refers to a set of concurrent data bits being transmitted simultaneously, and the byte-serial refers to consecutive bytes carried over the data link in a serial fashion. The GPIB consists of 16 transmission lines which are catagorized into:

1. eight data lines;
2. three data byte transfer control lines (handshake lines);
3. five general interface management signal lines.

The eight data lines are used to transfer data from the talkers to listeners. They are also used to transfer interface commands from the controller to various instruments. All transmission is asynchronous and all transmissions except parallel poll occurs according to a specialized 3-wire handshake (DAV, NRFD, and NDAC). It is with this handshake that the talker or controller synchronizes its readiness to transmit data to the listener's readiness to receive data.

At any point in time, an individual device on the bus will be idle monitoring the bus activity, a talker sending data to listeners on the bus, a listener receiving data from the talker on the bus, or a controller controlling the activity of the bus.

NOTE

Confusion sometimes arises in the meaning of the word controller. In the context of the IEEE-488 Standard and this manual, the word controller refers to a special device that connects to the GPIB. It is a complete unit in itself. The controller directs the flow of data by assigning devices to be either listeners or talkers. It can also interrupt the data flow and command specific actions within devices. The word controller does not refer to an MPU on the instrument side of the GPIB.

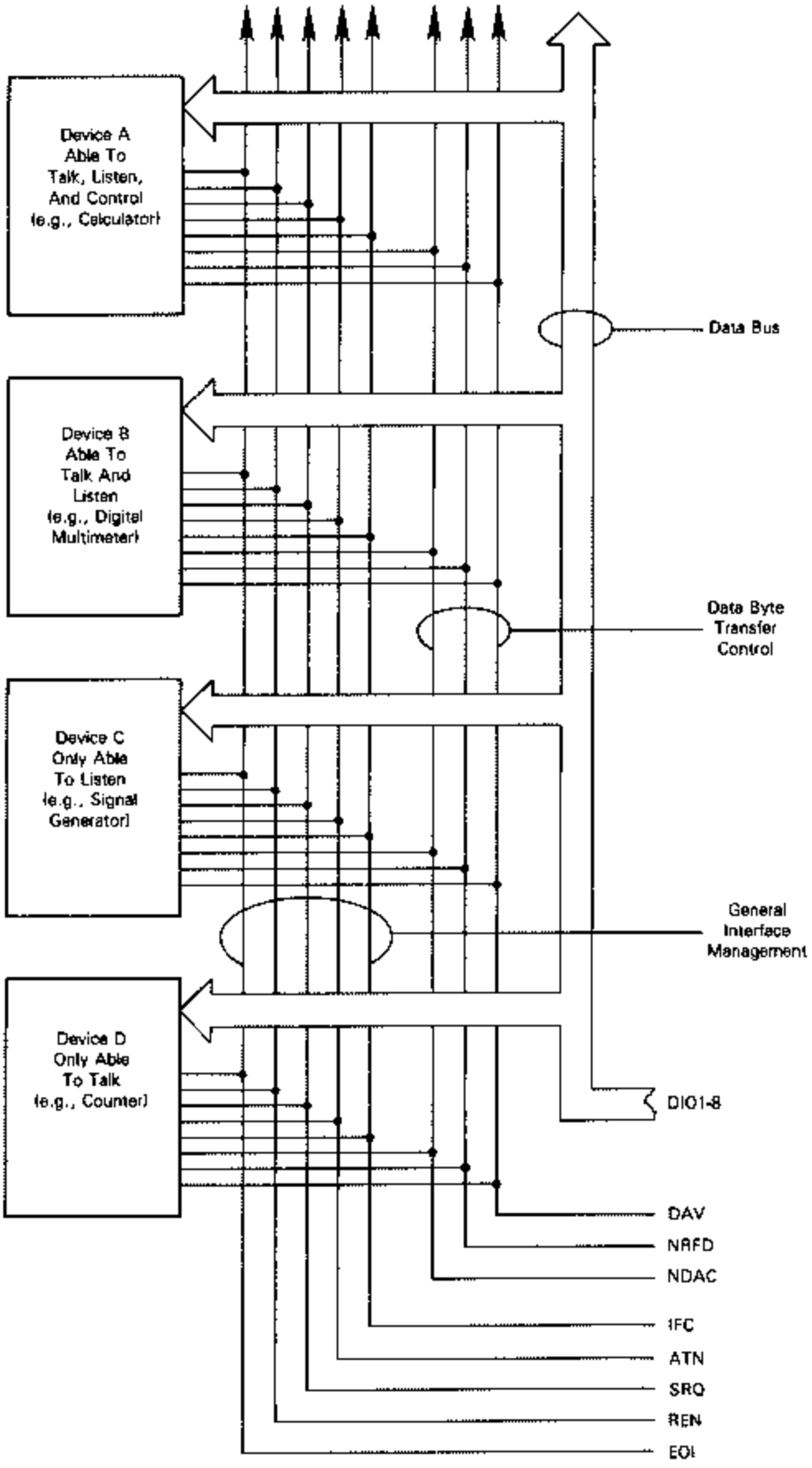


Figure 1-4. Interface Bus Structure

Many devices are both talkers and listeners. A programmable multimeter, for instance, will be a listener when receiving its programmed instructions and a talker when sending its data to another device such as a printer or disk. There can be more than one listener at one time, but only one talker. A minimum system need not contain a controller but may consist of just one talker and one listener. For example, a dedicated voltmeter could be outputting data to a dedicated printer. In such a system, it will be necessary for the two devices to have interfacing options that will allow local messages from the device MPU to assign them as either a talker or a listener. The assignment will most likely be made during power-up and will not change. Controllers can only be omitted from such dedicated systems, i.e., systems where the functions of talker and listener are not automatically reassigned.

The controller alters activity on the bus by sending interface commands. The controller (currently active controller if more than one exists on the bus at the same time) is the only device capable of sending commands. It does this in one of two ways:

1. Uniline Messages — The controller can send a command over one of the five management lines, e.g., Interface Clear (IFC) or Remote Enable (REN).
2. Multiline Messages — The controller can send a command over the eight data lines. It does this by first asserting the ATN (Attention) management line signifying to all instruments on the bus that the eight data lines contain a multiline command or address rather than data.

These messages are interface commands which do not interact directly with the measurement process of an instrument. They interact only with the interface logic within connected devices. The primary purpose of these messages is to carry out the proper protocol in setting up, maintaining, and terminating an orderly flow of device dependent messages (device dependent messages refer to the information being sent by the addressed talker device to the addressed listener devices and not the messages used to control the interface). The multiline and uniline messages are used to address devices as either talkers or listeners, to signal an instrument to ignore or not ignore front panel settings, to inquire about any problem the device has, to reset the interface circuitry, to begin making a measurement, etc. Commands can be categorized as Universal Commands, Addressed Commands, Universal Addressed Commands, and Secondary Address or Commands. The command repertoire of the IEEE-488 Standard is covered in detail in Chapter 2.

Addresses are assigned to each device so it can respond to addressed commands. Using this address the controller can pick out a specific device and instruct it to be a talker or listener. The controller does not assign addresses; this must come from an external means such as a set of switches attached to the back panel of an instrument or a subroutine resident in the device software. The address is placed in the instrument GPIB interface during an initialization sequence. Once resident in the interface circuitry, the device can respond to addressed commands. The address is a 5-bit binary number that allows the controller to talk to a particular instrument.

A talker sends a data byte over the GPIB to a listener or listeners using the asynchronous 3-wire handshake. The transfer begins when the talker makes data available (DAV is asserted) and is completed when the slowest listener accepts the data byte (DAC is asserted). The third handshake line, RFD (Ready For Data), is used to signal the talker that the listeners are ready for data. There actually are four states in a data transfer.

1. The talker generates a new byte;
2. The states of the data bus signal lines settle;

3. The listeners accept the data;
4. The listeners become ready for the next byte.

Since there can be many listeners (maximum of 14; 14 listeners + 1 talker = 15 devices) it is possible to have some that respond very quickly (e.g., a disk) and some that respond slowly (e.g., a mechanical teletype) to the same data byte. Thus, the overall speed of transmission over the bus will be governed by, and will not exceed, the response rate of the slowest listener.

There is no restriction to data format on the GPIB. The only functional restriction is that the data be transmitted in 8-bit byte lengths. The choice of code and convention is up to the system designer. It should be pointed out, however, that many commercially available calculators used as system controllers have established codes and data formats. The development task will, in many instances, be simpler if the data formatting convention of all devices in the system is compatible. The IEEE-488 Standard, even though it doesn't restrict itself to any particular convention, is set up up to accommodate standard 7-bit ASCII code with ease. Because of this, many controllers have adapted the ASCII code. A table is given in Appendix A correlating the IS0-7 code convention to the IEEE-488 Standard multiline command codes.

The objective of the IEEE-488 Standard is to provide a set of rules which insure compatibility and yet are flexible enough to allow a designer freedom to tailor an instrument to his particular needs. To do this, the IEEE-488 Standard was developed in terms of interface functions (as distinct from instrument functions), messages to and from the interface functions, and state diagrams describing the behavior of each of these functions. Thus, an instrument operation is divided into two sets of functions, interface functions (which insures the instrument behaves correctly with respect to the GPIB signal lines) and device functions (which are used for specific instrument control, e.g., performing a voltmeter function). The device functions are beyond the scope of the IEEE-488 Standard. Note that the theoretical separation of device and interface functions does not imply a physical separation. The interface is partitioned (see Chapter 2 for further details) into ten interface functions. Five of these — Source and Acceptor Handshake, Talker, Listener, and Controller — provide the primary communications capability. The other five interface functions, Device Clear, Device Trigger, Service Request, Parallel Poll, and Remote/Local add special purpose capabilities. In turn, these interface functions interact with the device functions to form a complete operational unit.

1.4 SUMMARY

The IEEE-488 Standard is a much needed and internationally recognized standard. It is universal and encompasses many types of applications. As such, it is also a very complex interface standard. Among the restraining factors causing its slow adoption has been the complexity of the logic protocol and the unavailability of low cost LSI ICs to satisfy the required logic needs. Implementing the interface standard discretely could involve up to 190 TTL small scale integrated circuit packages. The number of packages would of course depend upon the number of interface functions to be implemented: not all of the interface functions are required for every application.

Motorola's MC68488 General Purpose Interface Adapter (GPIA) provides an easy, low cost interface to the IEEE-488 Standard instrumentation bus. The GPIA completely implements all of the IEEE-488 Standard interface functions (except that of the controller) as well as provides extra special purpose features. When used in conjunction with the MC3448 or MC3447 high current bus drivers, a set is formed which interfaces any 8-bit MPU standard bus to the IEEE-488 Standard instrumention bus (Figure 1-5).

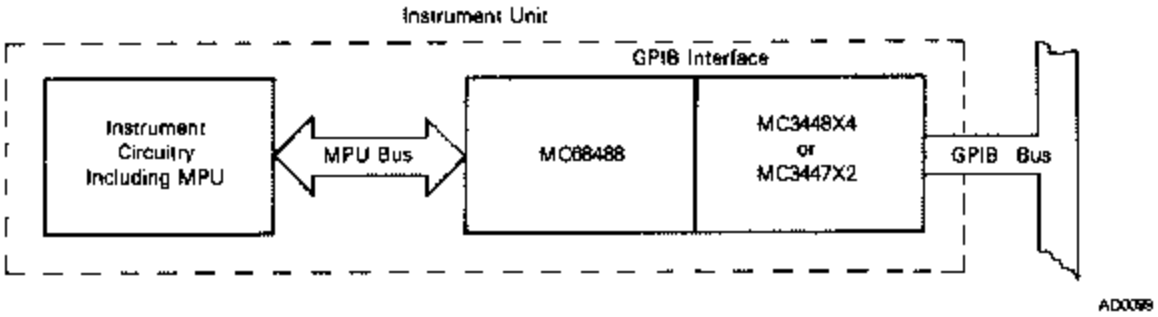


Figure 1-5. Typical Instrument Configuration

This manual describes the IEEE-488 Standard and shows how the MC68488 is used to implement this standard, including application examples.

CHAPTER 2
IEEE-488 STANDARD

The IEEE-488 Standard has become a recognized and accepted standard in the field of instrumentation. It is, however, not restricted to instrumentation and can be used in other areas that require an asynchronous parallel bus. This chapter highlights the standard, its system architecture and characteristics. The intent is to provide a working knowledge of the standard. For a more detailed description, a copy of the IEEE-488 Standard can be purchased from the IEEE headquarters in New York.

The general system defines all circuits, cables, connectors, control protocol, and message repertoire. Table 2-1 briefly outlines the general specifications for the interface system; the overall interface bus structure is illustrated in Figure 2-1.

Table 2-1. Interface Specification

Interconnected Devices	Up to 15 maximum on one Contiguous bus
Interconnection Path	Star or linear bus network under 20 meters total transmission path length
Signal Lines	Sixteen total; eight data lines and eight lines for critical control and status messages
Message Transfer Scheme	Byte-serial bit-parallel asynchronous data transfer using interlocked three-wide handshake technique
Data Rate	One megabyte per second maximum over limited distances; 250-500 kilobytes per second typical over full transmission path
Interface Function	Ten total; five primary communication functions and five special-purpose functions
Address Capability	Primary addresses, 31 talk and 31 listen; secondary (2-byte) addresses, 961 talk and 961 listen
Control Shift	May be delegated, never assumed, with a maximum of one talker (up to 14 listeners) at a time
Interface Circuits	Driver and receiver circuits TTL-compatible

2.1 BUS LINE CALLOUTS

The IEEE-488 Standard bus consists of a 24 line passive cable of which 8 lines are used for ground connection and the remaining 16 are signal lines categorized as:

1. eight data bus signal lines;
2. three handshake lines;
3. five management lines.

The following is a description of each of these lines. Since the bus has a negative logic convention, a less positive voltage level is referred to as TRUE (binary 1) and a more positive voltage level is FALSE (binary 0).

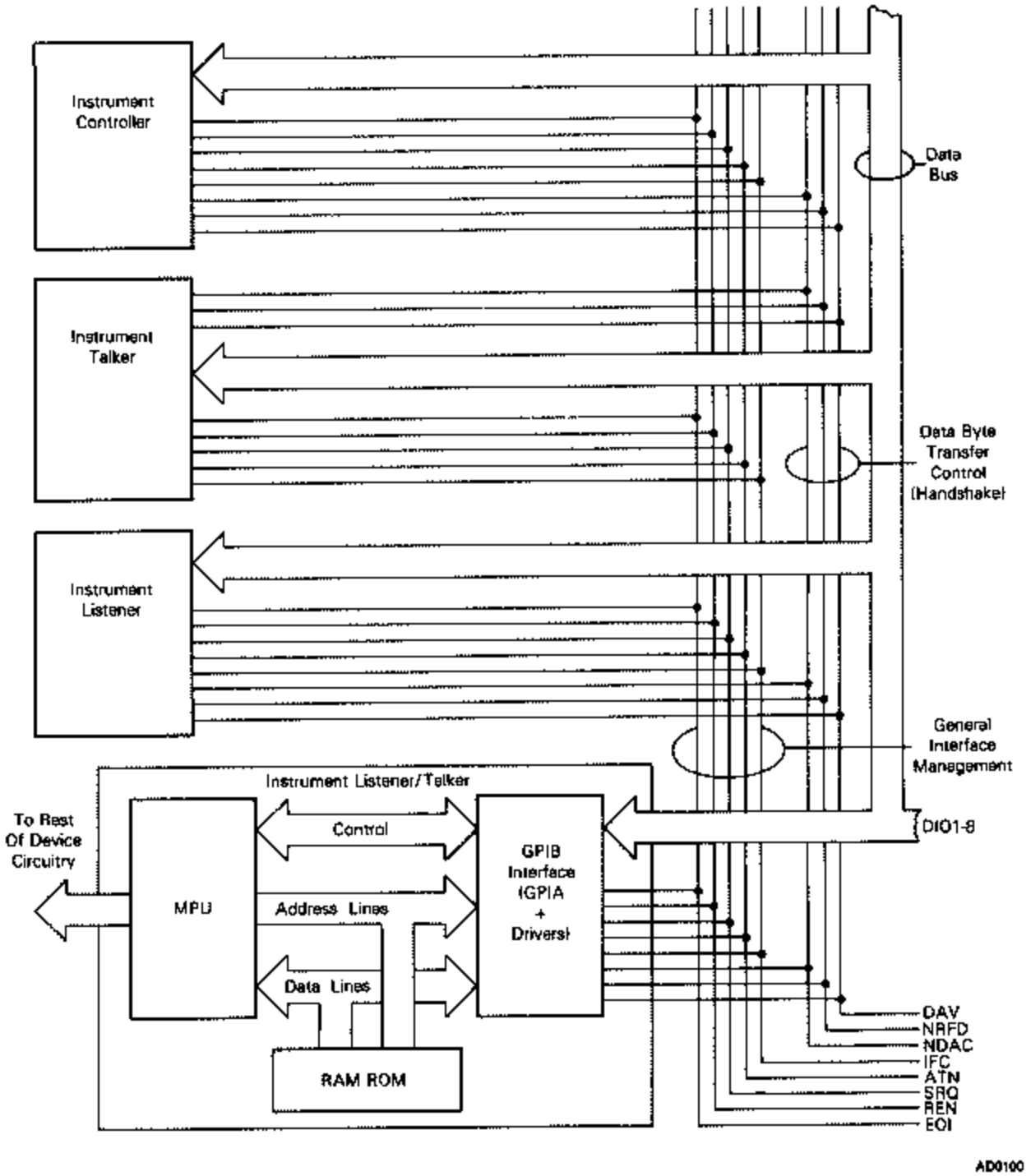


Figure 2-1. A Possible Interface Bus Structure

2.1.1 Data Lines

These are the message lines for carrying data in a bit-parallel, byte-serial form. Data is asynchronous and generally bidirectional. These lines carry either data or address/command information, depending upon the condition of the ATN management line.

2.1.2 Handshake Lines

The three handshake lines (NRFD, DAV and NDAC) form a unique three-wire handshake for transferring information on the GPIB. The handshake lines provide the means to asynchronously transfer data between devices.

NRFD — The NRFD (Not Ready for Data) line is used to indicate the condition of readiness of devices to accept data. All instruments must drive NRFD passively High when ATN is Low.The NRFD line and the NDAC line are monitored by the controller when ATN is Low and by the Addressed Talker when ATN is High. The NRFD line is Low when any listener is not ready for data.

DAV — The DAV line (Data Valid) is used to indicate the validity of data on the data lines. DAV is driven by the Controller when ATN is Low and by the Addressed Talker when ATN is High. The DAV line is monitored by all instruments if ATN is Low and by Addressed Listeners when ATN is High. The DAV line is Low when data is valid.

NDAC — The NDAC (Not Data Accepted) line is used to indicate acceptance of data by Addressed Listeners. Listeners indicate acceptance of data by setting NDAC passively High. When NDAC is Low, one or more Listeners have not accepted the data.

2.1.3 Management Lines

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

ATN — The ATN (Attention) line is used by the Active Controller to indicate how information on the data lines is to be interpreted. The ATN line must be monitored by all instruments other than the Active Controller at all times. When the ATN line is Low, the Active Controller can send interface messages and addresses to instruments on the bus. Data can be sent by the Active Talker to Active Listeners when the ATN line is high.

IFC — 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 low by the current System Controller and must be monitored by all other instruments. In order to clear an interface, the IFC line must be set low for at least 100 microseconds. IFC may be set by the current System Controller at any time. IFC puts talkers and listeners into their idle states.

REN — The REN (Remote Enable) line is used to operate an instrument under remote mode, e.g., interface control rather than front panel control. A Low level for the Remote Enable line is one of the conditions for operation of an instrument in remote mode. The REN line is driven by the current 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 High. The use of the remote function is optional.

SRQ — The SRQ (Service Request) line is used by an instrument to asynchronously request service from the Controller in charge of the GPIB. The SRQ line is sensed by the currently addressed Controller.

EOI — The EOI (End or Identify) line is used to indicate the end of a data string provided the ATN signal is High. When the ATN line is High, the Addressed Talker may indicate the end of its data by setting EOI Low at the same time it places the last byte on the data lines. The currently addressed Active Controller may initiate a Parallel Poll of all instruments with Parallel Polling capability by setting ATN and EOI Low simultaneously.

2.2 DEFINITIONS

A number of definitions or terminology are presented here to aid in understanding subsequent discussions of bus concepts. Unless otherwise noted with an asterisk (*), these definitions are directly from the IEEE-488 Standard.

Active Transfer — A technique for resolving conflicts between two devices simultaneously sending opposite remote message values, whereby, the interface is so structured that the active value overrides the passive value.

Bidirectional Bus — A bus used by an individual device for two-way transmission of messages, that is, both input and output.

Bit-Parallel — Refers to a set of concurrent data bits present on a like number of signal lines used to carry information. Bit-parallel data bits may be acted upon concurrently as a group (byte) or independently as individual data bits.

Bus — A signal line or a set of signal lines used by an interface system to which several devices can be connected and over which messages are carried.

Byte — A group of eight adjacent binary digits operated on as a unit.

Byte-Serial — A sequence of bit-parallel data bytes used to carry information over a common bus.

Compatibility — The degree to which devices may be interconnected and used, without modification, when designed as defined throughout Section 2.3 and 4 of the IEEE-488 Standard.

Controller — A device that can address other devices to listen or to talk. In addition, this device can send interface messages to command specified actions within other devices. A device with only this capability neither sends nor receives device dependent messages.

NOTE

The use of the word controller throughout the IEEE-488 Standard applies strictly to the management (control) of the interface system and does not imply the broad capabilities typically associated with the word in the data processing context. Further classification of the controller is made in Section 2 of the IEEE-488 Standard to distinguish between different types of controller capabilities related to the interface system. Listener, talkers, and controller capabilities occur individually and collectively in devices interconnected via the interface system, as shown in Figure 2-1.

Device Dependent Messages — Messages used by the devices interconnected via the interface system, that are carried by but not used or processed by, the interface system directly. Device dependent messages are passed between the device functions and the message coding logic via specified interface functions. These will cause no state transitions within the interface functions. Examples of device dependent messages include device programming data, device measurement data, and device status data (see Figure 2-7, message route 3).

An Expression — Consists of one or more local messages, remote messages, state linkages, or minimum time limits used in conjunction with the operators AND, OR, or NOT.

Handshake Cycle — The process whereby digital signals affect the transfer of each data byte across the interface by means of an interlocked sequence of status and control signals. Interlocked denotes a fixed sequence of events in which one event in the sequence must occur before the next event may occur.

High State — The relatively more positive signal level used to assert a specific message content associated with one of two binary logic states.

Interface — A shared boundary between a considered system and another system, or between parts of a system, through which information is conveyed.

Interface Messages — Messages used to manage the interface system itself. Each interface message is sent to cause a state transition within another interface function. An interface message will not be passed along to the device when received by an interface function (see Figure 2-7, message route 2).

Interface System — The device-independent mechanical, electrical and functional elements of an interface necessary to effect communication among a set of devices. Cables, connector, driver and receiver circuits, signal line descriptions, timing and control conventions, and functional logic circuits are typical interface system elements.

Listener — A device that can be addressed by an interface message to receive device dependent messages from another device connected to the interface system. Listener, talker, and controller capabilities occur individually and collectively in devices interconnected via the interface system, as shown in Figure 2-1.

Local Control — A method whereby a device is programmable by means of its local (front or rear panel) controls in order to enable the device to perform different tasks. (Also referred to as manual control.)

Local Messages — Local messages flow between device functions and interface functions (see Figure 2-7, message route 5).

NOTE

Certain local messages are conveyed as remote messages and vice versa.

The designer is not allowed to introduce new local messages to interface functions.

The designer is allowed to introduce a local message derived from any state of any interface functions to device function(s).

Local messages sent by device functions must exist for enough time to cause the required state transitions.

Low State — The relatively less positive signal level used to assert a specific message content associated with one of two binary logic states (a logic true).

Messages — Quantities of information carried by the interface system. A message will be received either true or false at any specific time.

All communication between an interface function and its environment is accomplished through message sent or received.

Message Coding — Message coding is the act of translating remote messages to or from interface signal lines values.

***Mnemonic** — Refers to a technique of abbreviations that has some easily remembered relationship to the name of the state or message; i.e., NY for New York, JFK for New York's John F. Kennedy Airport.

Multiline Message — A message that shares a group of signal lines with other messages, in some mutually exclusive set. Only one multiline message (message byte) can be sent at one time.

Passive Transfer — A technique for resolving conflicts between two devices simultaneously sending opposite remote message values. A passive value is not guaranteed to be the value received as it can be overridden by an active transfer.

Programmable — That characteristic of a device that makes it capable of accepting data to alter the state of its internal circuitry to perform two or more specific tasks.

Programmable Measuring Apparatus — A measuring apparatus that performs specified operations on command from the system and, if it is a measuring apparatus proper, may transmit the results of the measurement(s) to the system.

Remote Messages — Messages sent via the interface between interface functions of different devices are called remote messages.

Each remote message is either an interface message or a device dependent message.

Signal — The physical representation which conveys data from one point to another.

NOTE

For the purpose of the IEEE-488 Standard, this is a restricted definition of what is often called "signal" in more general sense, and is hereinafter referring to digital electrical signals only.

Signal Level — The magnitude of a signal when considered in relation to an arbitrary reference magnitude (voltage in the case of the IEEE-488 Standard).

Signal Line — One of a set of signal conductors in an interface system used to transfer messages among interconnected devices.

Signal Parameter — That parameter of an electrical quantity in which the values or sequence of values convey information.

State Linkage — The logical interconnection of two interface functions where the transition to an active state of one interface function is dependent upon the existence of a specified active state of another interface function (see Figure 2-7, message route 4).

System — A set of interconnected elements constituted to achieve a given objective by performing a specified function.

Talker — A device that can be addressed by an interface message to send device dependent messages to another device connected to the interface system.

Listener, talker, and controller capabilities occur individually and collectively in devices interconnected via the interface system, as shown in Figure 2-1.

Terminal Unit — An apparatus by means of which a connection (and translation, if required) is made between the considered interface system and another external interface system.

Unidirectional Bus — A bus used by any individual device for one-way transmission of messages only, that is, either input only or output only.

Uniline Message — A message sent over a single signal line. Two or more of these messages can be sent concurrently.

2.3 CONVENTIONS

Following the lead of IEEE-488 Standard, certain conventions will be followed throughout the remaining text and in all figures and diagrams.

Local Messages — All local messages to an interface function defined by the IEEE-488 Standard will be represented by three-letter lower case mnemonic. Example: rdy. Local messages are tabulated in Table 2-2.

Table 2-2. Local Message Mnemonics

Mnemonic	Message Title
gts	go to standby
ISR	individual service request (qual)
lon	listen only
lpe	local poll enable
lun	local unlisten
nba	new byte available
pon	power on
rdy	ready
rpp	request parallel poll
rsc	request system control
rsv	request service
rtl	return to local
sic	send interface clear
sre	send remote enable
tca	take control asynchronously
tcs	take control synchronously
ton	talk only

(a) IEEE-488 Standard specified local messages.

apte	address pass through enable
dacd	DAC disable (disables automatic command mode)
dacr	DAC release
dal	disable listener
dat	disable talker
dsai	disable interrupt for GET, UACG, UUCG, SDC, DLC
feoi	force EOI line true
fget	force group execute trigger
hlda	hold RFD on all data
hlde	hold RFD on end of data
lsbe	least significant bit enable
ma	my address
msa	my secondary address
rfdr	RFD release

(b) Non IEEE-488 Standard specified local messages. These 3 and 4 letter mnemonics represent additional local messages used by the MC68488 and are not part of the standard.

Remote Messages — All remote messages received via the interface will be represented by three-letter upper case mnemonic. Example: ATN. Remote messages are given in Table 2-3.

Table 2-3. Remote Message Mnemonics

Mnemonic	Message Title
ATN	Attention
DAB	Data Byte
DAC	Data Accepted
DAV	Data Valid
DCL	Device Clear
END	End
GET	Group Execute Trigger
GTL	Go To Local
IDY	Identify
IFC	Interface Clear
LLO	Local Lock Out
MLA or [MLA]	My Listen Address
MSA or [MSA]	My Secondary Address
MTA or [MTA]	My Talk Address
OSA	Other Secondary Address
OTA	Other Talk Address
PCG	Primary Command Group
PPC	Parallel Poll Configure
[PPD]	Parallel Poll Disable
[PPE]	Parallel Poll Enable
PPRn	Parallel Poll Response n
PPU	Parallel Poll Unconfigure
REN	Remote Enable
RFD	Ready For Data
RQS	Request Service
[SDC]	Selected Device Clear
SPD	Serial Poll Disable
SPE	Serial Poll Enable
SRQ	Service Request
STB	Status Byte
TCT or [TCT]	Take Control
UNL	Unlisten

State Notation — Each state that an interface function can assume will be represented by a four-letter upper case mnemonic with an S being the final letter. The mnemonic will be enclosed in a circle and all permissible transitions between states of the interface function will be represented by arrows between the circles. States are summarized in Table 2-4.

Linkage — A linkage from another state diagram will be represented by a four-letter mnemonic enclosed in an oval. Example: (LACS) . Linkages are listed in Table 2-5.

Table 2-4. Interface State Mnemonics

Mnemonic	Message Title
ACDS	Accept Data State
ACRS	Acceptor Ready State
AIDS	Acceptor Idle State
ANRS	Acceptor Not Ready State
APRS	Affirmative Poll Response State
AWNS	Acceptor Wait For New Cycle State
CACS	Controller Active State
CADS	Controller Addressed State
CAWS	Controller Active Wait State
CIDS	Controller Idle State
CPPS	Controller Parallel Poll State
CPWS	Controller Parallel Poll Wait State
CSBS	Controller Standby State
CSNS	Controller Service Not Requested State
CSRS	Controller Service Requested State
CSWS	Controller Synchronous Wait State
CTRS	Controller Transfer State
DCAS	Device Clear Active State
DCIS	Device Clear Idle State
DTAS	Device Trigger Active State
DTIS	Device Trigger Idle State
LACS	Listener Active State
LADS	Listener Addressed State
LIDS	Listener Idle State
LOCS	Local State
LPAS	Listener Primary Addressed State
LPIS	Listener Primary Idle State
LWLS	Local With Lockout State
NPRS	Negative Poll Response State
PACS	Parallel Poll Addressed To Configure State
PPAS	Parallel Poll Active State
PPIS	Parallel Poll Idle State
PPSS	Parallel Poll Standby State
PUCS	Parallel Poll Unaddressed To Configure State
REMS	Remote State
RWLS	Remote With Lockout State
SACS	System Control Active State
SDYS	Source Delay State
SGNS	Source Generate State
SIAS	System Control Interface Clear Active State
SIDS	Source Idle State
SIIS	System Control Interface Clear Idle State
SINS	System Control Interface Clear Not Active State
SIWS	Source Idle Wait State
SNAS	System Control Not Active State
SPAS	Serial Poll Active State
SPIS	Serial Poll Idle State
SPMS	Serial Poll Mode State
SRAS	System Control Remote Enable Active State
SRIS	System Control Remote Enable Idle State
SRNS	System Control Remote Enable Not Active State
SRQS	Service Request State
STRS	Source Transfer State
SWNS	Source Wait For New Cycle State
TACS	Talker Active State
TADS	Talker Addressed State
TIDS	Talker Idle State
TPAS	Talker Primary Addressed State
TPIS	Talker Primary Idle State

Table 2-5. State Linkages

State Diagram	Mnemonic	Interface State
SH	TACS	Talker Active State (T function)
SH	SPAS	Serial Poll Active State (T function)
SH	CACS	Controller Active State (C function)
SH	CTRS	Controller Transfer State (C function)
AH	LADS	Listener Addressed State (L function)
AH	LACS	Listener Active State (L function)
T	ACDS	Accept Data State (AH function)
TE	ACDS	Accept Data State (AH function)
TE	LPAS	Listener Primary Addressed State (L function)
L	ACDS	Accept Data State (AH function)
L	CACS	Controller Active State (C function)
LE	ACDS	Accept Data State (AH function)
LE	CACS	Controller Active State (C function)
LE	TPAS	Talker Primary Addressed State (T function)
SR	SPAS	Serial Poll Active State (T function)
RL	ACDS	Accept Data State (AH function)
RL	LADS	Listener Addressed State (L function)
PP	ACDS	Accept Data State (AH function)
PP	LADS	Listener Addressed State (L function)
DC	ACDS	Accept Data State (AH function)
DC	LADS	Listener Addressed State (L function)
DT	ACDS	Accept Data State (AH function)
DT	LADS	Listener Addressed State (L function)
C	ACDS	Accept Data State (AH function)
C	ANRS	Accept Not Ready State (AH function)
C	SDYS	Source Delay State (SH function)
C	STRS	Source Transfer State (SH function)
C	TADS	Talker Addressed State (T function)

Maximum Time — If a transition has a maximum time limit, it will be indicated by the symbol t_n. Thus the state pointed to, must be entered within a specified amount of time after the expression becomes true. See Table 2-6.

Minimum Time — A minimum time limit is represented by the symbol T_n. This symbol achieves a true value only after the interface has been in the state originating the transition for the time value specified. It will remain true until the state is exited. See Table 2-6 for these time limits.

AND — The AND operator is represented by the symbol ∧.

OR — The OR operator is represented by symbol ∨. (The AND operator takes precedence over the OR operator unless specified by parenthesis.)

NOT — The NOT operator is represented by a horizontal bar above the portion of the expression to be negated.

Active/Passive — The interface is so structured that conflicts, which may arise when opposite remote message values are simultaneously transmitted by two devices, are resolved. In such cases one message must always be made to override the other. Thus an active value will override a passive value when conflicts occur.

T — indicates active true.

F — indicates active false.

(T) — indicates passive true.

(F) — indicates passive false.

Optional True — If a portion of an expression is optional in that its true value is not required for the complete expression to be true, then it will be enclosed with square brackets [.....].

Table 2-6. Time Values

Time Value Identifier*	Function (Applies To)	Description	Value
T_1	SH	Setting time for multiline messages	≥ 2 μs†
t_2	SH, AH, T, L	Response to ATN	≤ 200 ns
T_3	AH	Interface message accept time‡	> 0 §
t_4	T, TE, L, LE, C	Response to IFC or REN false	< 100 μs
t_5	PP	Response to ATN ∧ EOI	≤ 200 ns
T_6	C	Parallel poll execution time	≥ 2 μs
T_7	C	Controller delay to allow current talker to see ATN message	≥ 500 ns
T_8	C	Length of IFC or REN false	> 100 μs
T_9	C	Delay for EOI**	≥ 1.5 μs††

*Time values specified by a lower case t indicate the maximum time allowed to make a state transition. Time values specified by an upper case T indicate the minimum time that a function must remain in a state before exiting.

†If three-state drivers are used on the DIO, DAV, and EOI lines, T_1 must be:
(1) ≥ 1100 ns
(2) Or ≥ 700 ns if it is known that within the controller ATN is driven by a three-state driver
(3) Or ≥ 500 ns for all subsequent bytes following the first sent after each false transition of ATN (the first byte must be sent in accordance with (1) or (2))

‡Time required for interface functions to accept, not necessarily respond to, interface messages.

§Implementation dependent.

**Delay required for EOI, NDAC, and NRFD signal lines to indicate valid states.

††600 ns for three-state drivers.

2.4 LOGIC LEVELS

Throughout the IEEE-488 Standard document the coded logical states 0 and 1 are utilized. A negative logic convention is defined as follows:

Coding Logical State	Electrical Signal Level
0 = False	≥ 2.0 V called the high state
1 = True	≤ 0.8 V called the low state

Either open-collector or three-state drivers may be utilized with the following constraints:

1. Open collector types are required on the following lines: SRQ, NRFD, and NDAC. In addition, if parallel polling is used, DIO1-DIO8 must also be open collector types.
2. Three-state drivers are recommended for higher speed systems and especially in a controller ATN signal line, if used with other devices employing three-state drivers on the DAV, EOI and DIO1-DIO8 lines.

Drivers of either open-collector or three-state configuration must be capable of sinking +48 mA without exceeding 0.5 V. Three-state drivers must maintain 2.4 V, or greater, when sourcing 5.2 mA. Typical specifications for both open-collector and three-state drivers and receivers are given in the IEEE-488 Standard. When implemented with standard TTL logical elements and with a 3 kΩ resistor to V_{CC} and a 6.2 kΩ resistor to ground at each common node, the typical suggested configuration is met (see Figure 2-2). The actual dc load requirement which must be met, however, is a function of driver, receiver, and resistive terminations. This load requirement is given in Figure 2-3.

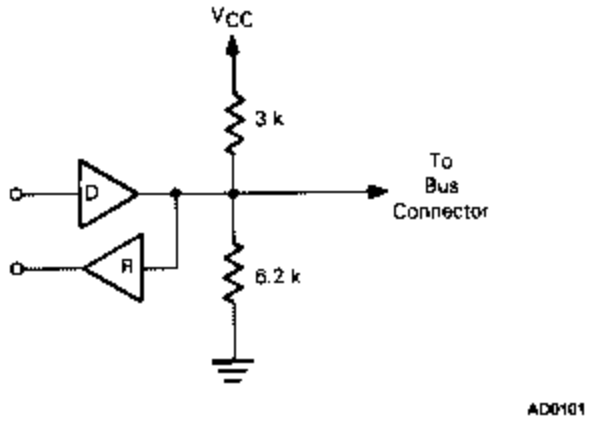


Figure 2-2. Typical TTL Implementation

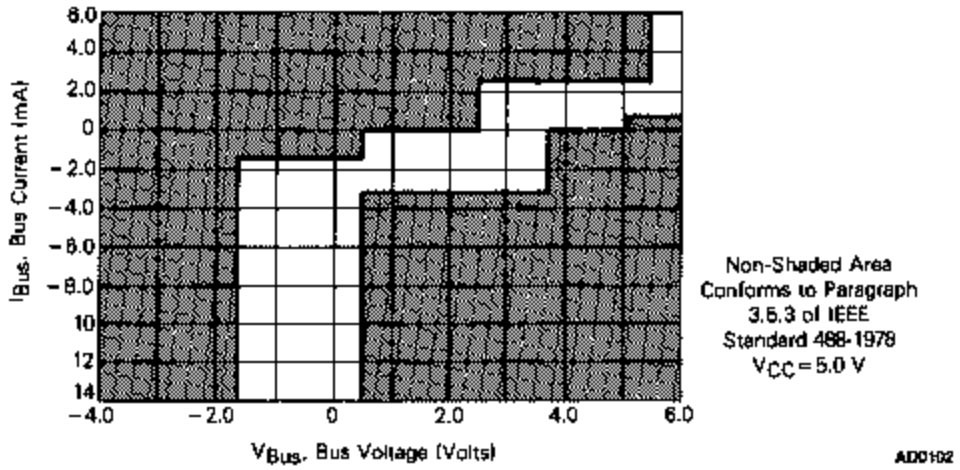


Figure 2-3. Typical Bus Load Line

The recievers may be standard TTL gates with 2.0 V high state input voltages or Schmitt trigger types employing hysteresis.

If hysteresis is utilized for improved noise immunity, it is recommended that at least 0.4 V be employed. It is also recommended that the lower threshold be greater than 0.8 V and the upper threshold less than +2.0 V. Negative voltage clamping is also required within the receiver.

2.5 TIMING

All specified timing constraints are summarized in Table 2-6.

2.6 THE CONNECTOR

Both the dimensions and the actual pin locations of the connector are prescribed in the IEEE-488 Standard. Recommended connectors include MICRORIBBON (Amphenol or Cinch Series 57) or CHAMP (AMP). Illustrations of the connector, cables and pin connections are presented in Table 2-7 and Figures 2-4 through 2-6.

Table 2-7. Connector Pin Assignments

Contact	Signal Line	Contact	Signal Line
1	DIO1	13	DIO5
2	DIO2	14	DIO6
3	DIO3	15	DIO7
4	DIO4	16	DIO8
5	EOI	17	REN
6	DAV	18	Gnd. (6)
7	NRFD	19	Gnd. (7)
8	NDAC	20	Gnd. (8)
9	IFC	21	Gnd. (9)
10	SRQ	22	Gnd. (10)
11	ATN	23	Gnd. (11)
12	SHIELD	24	Gnd. LOGIC

Note: Gnd. (n) refers to the signal ground return of the referenced contact.

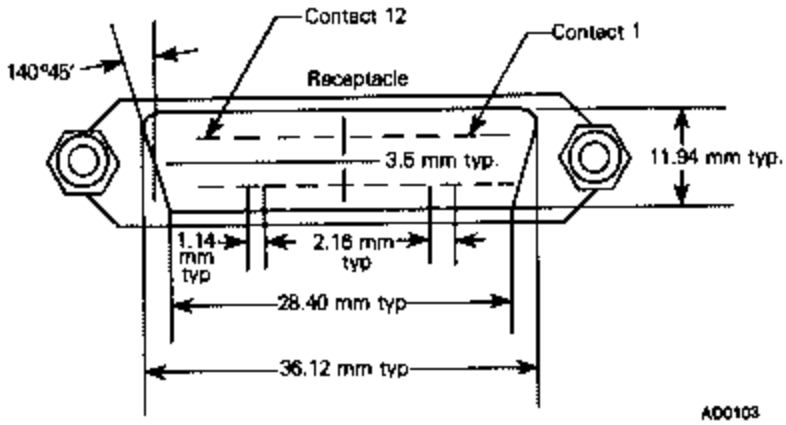


Figure 2-4. Device Connector Mounting

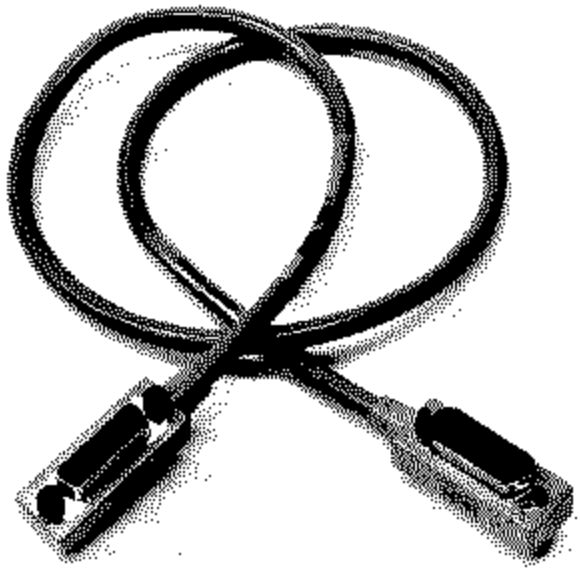


Figure 2-5. Cable and Connectors

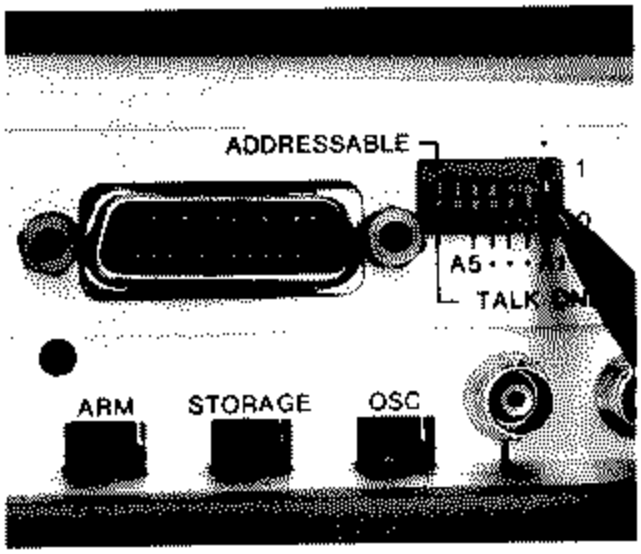


Figure 2-6. Device Mounted Connector

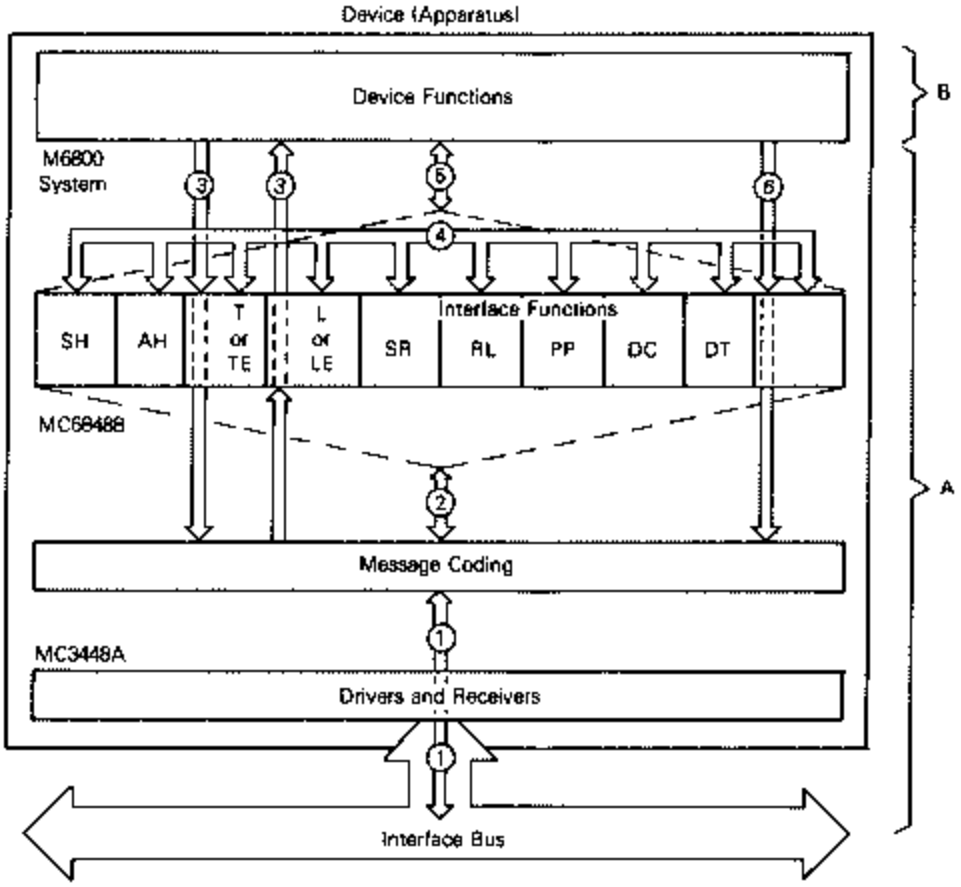
Photos courtesy of Hewlett-Packard.

2.7 PARTITIONING

An IEEE-488 Standard compatible device can be divided into three major functional parts:

1. The device function;
2. The interface functions (5 basic and 5 supplementary);
3. The message coding logic.

Figure 2-7 shows the way the IEEE-488 Standard system is partitioned. The device functions, as mentioned earlier, are simply the application which the device has been designed to perform, e.g., voltmeter, signal generator, logic analyzer, etc.



A — Capability defined by the IEEE-488 Standard
B — Capability defined by the designer
1 — Interface bus signal lines
2 — Remote interface messages to and from interface functions
3 — Device dependent messages to and from device functions
4 — State linkages between interface functions
5 — Local messages between device functions and interface functions (messages to interface functions are defined; messages from interface functions exist according to the designer's choice)
6 — Remote interface messages sent by device functions within a controller

AD0104

Figure 2-7. Functional Partition Within A Device

2.7.1 Interface Functions

The interface function is that part of the system which allows for the basic link through which a device can receive, process, and transmit messages. Table 2-8 and 2-9 list the ten interface functions defined by the standard.

Table 2-8. Basic Interface Functions

Function	Description
(1) TALKER	
Basic Talker	Allows an instrument to send data to another instrument.
Talk Only	Allows an instrument to operate in a system with a controller.
Unaddress if my listen address (MLA)	Prevents an instrument capable of functioning as both a talker and a listener from talking to itself.
Extended Talker (TE)	Same as talker function with added addressing capability to send a "status byte" to the
Serial Poll	controller and identify itself in the course of a service request.
(2) LISTENER	
Basic Listener	Allows an instrument to receive data from another instrument.
Listen Only	Allows an instrument to operate in a system without a controller.
Unaddress if my talk address (MTA)	To prevent an instrument capable of functioning as both talker and listener from listening to itself.
Extended Listener (LE)	Same as listener function with added addressing capability.
(3) SOURCE HANDSHAKE	Synchronize transmission of information on the data bus by the talker when sending instrument-generated data and by the controller when sending interface messages.
(4) ACCEPTOR HANDSHAKE	Synchronizes the receipt of information on the data bus for all interface functions when receiving interface messages and for the listener function when receiving instrument-generated data.
(5) CONTROLLER	
System Controller	Allows an instrument to send the interface clear (IFC) or remote enable (REN) messages.
Send Interface Clear (IFC)	Allows a system to take charge from another controller and/or initialize the bus.
Send Remote Enable (REN)	Allows a system controller to enable instruments to switch to remote control.
Respond to Service Request (SRQ)	Allows the controller to respond to service requests.
Send Interface Messages	Allows the controller to send multiline messages.
Receive Control	Allows the controller to accept control on the bus from another controller.
Pass Control	Allows the controller to pass control of the bus to another controller.
Parallel Poll	Allows the control to execute a parallel poll.
Take Control Synchronously	Allows the controller to take control of the bus without destroying a data transmission in progress.

Table 2-9. Supplementary Interface Functions

Function	Description
(6) SERVICE REQUEST	Allows an instrument to indicate to the controller that some event has occurred and request it to take some specific action asynchronously with respect to other bus operations (one SRQ function is required for each independent reason for requesting service).
(7) REMOTE-LOCAL	
Basic Remote-Local	Allows the control of the instrument to be switched between its local (manual) controls and remote controls (programming codes received while addressed as a listener).
Local Lock Out	Allows the local control "return to local" to be disabled.
(8) PARALLEL POLL	
Basic Parallel Poll	Allows instruments to return one-bit status to the controller. Up to eight instruments may respond simultaneously. More than one instrument may respond on the same status line so that logical operations (AND, OR) may be performed on a group of instruments.
Parallel Poll Configuration	Allows the instrument to be configured by the controller.
(9) DEVICE CLEAR	
Basic Device Clear	Provides a means by which an instrument (device) may be initialized to a predefined state. All instruments are cleared concurrently.
Selective Device Clear	Clears individual instruments (devices) selectively.
(10) DEVICE TRIGGER	Allows instruments (devices), either singly or in a group, to be triggered, or some action be started.

2.7.2 Message Coding

Message coding is the process of converting remote messages to or from interface signal line values.

As mentioned in Chapter 1, there are two types of messages: uniline, a message sent over a signal line; and multiline, a message that uses a group of lines. Only one multiline message may be sent at a time, however, more than one uniline message may be set at a time (on different lines). Uniline and multiline messages form the command repertoire defined by the IEEE-488 Standard. The commands serve several different purposes such as:

1. Address, or talk and listen commands select the instruments that will transmit and accept data. They are all multiline messages.
2. Universal commands cause every instrument so equipped to perform a specific interface question. They include multiline messages and three uniline commands; Interface Clear (IFC), Remote Enable (REN), and Attention (ATN).
3. Addressed commands are similar to universal commands, except that they affect only those devices that are addressed and are all multiline commands. An instrument responds to an addressed command, however, only after an address has already told it to be a talker or listener.
4. Secondary commands are multiline messages that are always used in series with primary commands (address, universal command, or addressed command) to form a longer version of the particular command, thus they extend the code space when necessary.

To send multiline messages, the controller uses seven of the eight data bus lines. This allows instruments using 7-bit ASCII code to act as a controller. The commands are catagorized into five distinct groups by the assignment of bits 5, 6 and 7 of the command word (Table 2-10). For example, the code format for any multiline message in the Listen Command Group is (X01XXXXX). Bit 6 being a 1 and bit 7 a 0 define this group. The five least significant bits define the address of the device. Thus the command (X0100011) is a multiline message that instructs device #3 to be a listener. All 1s (decimal 31) in the five least significant bits are not allowed as an address because the code (X0111111) is reserved for the unlisten command (see Table 2-10). The talk command (X10XXXXX), defined by bit 6 being a 0 and bit 7 a 1, is analogous to the listen command. Thus the command (X1000011) is a multiline message that instructs device #3 to talk. The command (X1011111) is the untalk command for all devices.

Table 2-10. Command Group Coding

Code								Function	ASCII
b8	b7	b6	b5	b4	b3	b2	b1		
X	0	0	0	A_4	A_3	A_2	A_1	Addressed Commands	NUL – SI
X	0	0	1	A_4	A_3	A_2	A_1	Universal Commands	DLE – US
X	0	1	A_5	A_4	A_3	A_2	A_1	Listen Address	SP – > (includes numerics)
X	0	1	1	1	1	1	1	Unlisten Command	?
X	1	0	A_5	A_4	A_3	A_2	A_1	Talk Address	@ – ↑(includes an alpha char.)
X	1	0	1	1	1	1	1	Untalk Command	_
X	1	1	A_5	A_4	A_3	A_2	A_1	Secondary Commands	` – ~
X	1	1	1	1	1	1	1	Ignored	DEL

For the multiline interface messages, Table 2-11 lists the remote message coding. Many devices use the IS0-7 bit code (or the equivalent code in American National Standard Code for Information Interchange ANSI X3.4-168) because it is convenient to both generate and interpret this code. A description of this code and the relationships between the IS0-7 code and the messages (binary bit pattern) defined in the IEEE-488 Standard can be found in Appendix A.

A uniline message value is valid as soon as the corresponding logic state is detected. However, multiline messages are valid only within the context of SH and AH functions. Thus the transmitted multiline message is valid while the SH function is in the STRS state. The received multiline message is valid while the AH function is in the ACDS state. All passive message values are transferred as 0 signal line states. This requires the logic OR of signal line states to be performed on the interface.

2.7.3 Drivers and Receivers

The drivers and receivers provide the necessary current levels and termination networks to satisfy the requirements in the preceding section on logic levels. It is generally assumed that either bipolar or MOS technology can be employed for the message coding and interface function sections. MOS LSI will probably be preferred due to the logic complexity and either gold-doped or Schottky bipolar devices can be utilized to implement the 48 mA drivers and provide the current drive at the actual bus.

The Motorola complement of ICs includes four Schottky bipolar quad transceivers, type MC3448A or two Schottky bipolar octal transceivers, type MC3447C for the driver/receiver function; and an NMOS LSI device, type MC68488, to provide the message coding and interface functions. The MC68488 is designed specifically to be an interface between the IEEE-488 Standard bus and the M6800 microprocessor bus. With some modification, however, the MC68488 can be made to function with other microprocessor systems.

2.8 THE HANDSHAKE*

2.8.1 General Description

When data is transferred from a source to one or more acceptors, a handshaking procedure is utilized to insure that the transfer is performed properly. The procedure is repeated for each byte transferred. Three signal lines are utilized for the handshake; the DAV, NRFD and NDAC line. The DAV line is controlled by the talker, while the NRFD and NDAC are under the control of the listeners.

The handshake procedure insures that each listener is ready to accept data, that the data on the DIO1-DIO8 lines is valid data and that the data has been accepted by all listeners. Data will be sent only as rapidly as it can be accepted by the slowest listener.

*The three-wire handshake described in this section is the subject of patents owned by Hewlett-Packard Co. Inquiries on license details should be directed to the legal department of Hewlett-Packard.

Table 2-11. Remote Message Coding

Mnemonic	Message Name	Bus Signal Line(s) and Coding That Assert Message True: Data I/O DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	Handshake DAV	NRFD	NDAC	Bus Management ATN	EOI	REN	IFC	SRQ	Message Type	Message Class	Notes
ACG	Addressed Command Group	X	0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	M	AC	10
ATN	Attention	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	X	U	UC	–
DAB	Data Byte	D8	D7	D6	D5	D4	D3	D2	D1	X	X	X	X	X	X	X	X	M	DD	1, 9
DAC	Data Accepted	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	U	HS	–
DAV	Data Valid	X	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	U	HS	–
DCL	Device Clear	X	0	0	1	0	1	0	0	X	X	X	X	X	X	X	X	M	UC	10
END	End	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	U	ST	9, 11
EOS	End of String	E8	E7	E6	E5	E4	E3	E2	E1	X	X	X	X	X	X	X	X	M	DD	2, 9
GET	Group Execute Trigger	X	0	0	0	1	0	0	0	X	X	X	X	X	X	X	X	M	AC	10
GTL	Go To Local	X	0	0	0	0	0	0	1	X	X	X	X	X	X	X	X	M	AC	10
IDY	Identify	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	X	U	UC	10, 11
IFC	Interface Clear	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X	U	UC	–
LAG	Listen Address Group	X	0	1	X	X	X	X	X	X	X	X	X	X	X	X	X	M	AD	10
LLO	Local Lock Out	X	0	0	1	0	0	0	1	X	X	X	X	X	X	X	X	M	UC	10
MLA	My Listen Address	X	0	1	L5	L4	L3	L2	L1	X	X	X	X	X	X	X	X	M	AD	3, 10
MTA	My Talk Address	X	1	0	T5	T4	T3	T2	T1	X	X	X	X	X	X	X	X	M	AD	4, 10
MSA	My Secondary	X	1	1	S5	S4	S3	S2	S1	X	X	X	X	X	X	X	X	M	SE	5, 10
NUL	Null Byte	1	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X	M	DD	–
OSA	Other Secondary Address	(OSA = SCG ∧ MSA)																M	SE	10
OTA	Other Talk Address	(OTA = TAG ∧ MTA)																M	AD	10
PCG	Primary Command Group	(PCG = ACG ∨ UCG ∨ LAG ∨ TAG)																M	–	10
PPC	Parallel Poll Configure	X	0	0	0	0	1	0	1	X	X	X	X	X	X	X	X	M	AC	10
PPE	Parallel Poll Enable	X	1	1	0	S	P3	P2	P1	X	X	X	X	X	X	X	X	M	SE	6, 10
PPD	Parallel Poll Disable	X	1	1	1	D4	D3	D2	D1	X	X	X	X	X	X	X	X	M	SE	7, 10
PPR1	Parallel Poll Response 1	X	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	U	ST	–
PPR2	Parallel Poll Response 2	X	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X	U	ST	–
PPR3	Parallel Poll Response 3	X	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X	U	ST	–
PPR4	Parallel Poll Response 4	X	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	U	ST	–
PPR5	Parallel Poll Response 5	X	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	U	ST	–
PPR6	Parallel Poll Response 6	X	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	U	ST	–
PPR7	Parallel Poll Response 7	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	U	ST	–
PPR8	Parallel Poll Response 8	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	U	ST	–
PPU	Parallel Poll Unconfigure	X	0	0	1	0	1	0	1	X	X	X	X	X	X	X	X	M	UC	10
REN	Remote Enable	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	1	U	UC	–
RFD	Ready for Data	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	U	HS	–
RQS	Request Service	X	1	X	X	X	X	X	X	X	X	X	X	X	X	X	X	U	ST	9
SCG	Secondary Command Group	X	1	1	X	X	X	X	X	X	X	X	X	X	X	X	X	M	SE	10
SDC	Selected Device Clear	X	0	0	0	0	1	0	0	X	X	X	X	X	X	X	X	M	AC	10
SPD	Serial Poll Disable	X	0	0	1	1	0	0	1	X	X	X	X	X	X	X	X	M	UC	10
SPE	Serial Poll Enable	X	0	0	1	1	0	0	0	X	X	X	X	X	X	X	X	M	UC	10
SRQ	Service Request	X	X	X	X	X	X	X	X	X	X	X	X	X	1	X	X	U	ST	–
STB	Status Byte	S8	X	S6	S5	S4	S3	S2	S1	X	X	X	X	X	X	X	X	M	ST	8, 9
TCT	Take Control	X	0	0	0	1	0	0	1	X	X	X	X	X	X	X	X	M	AC	10
TAG	Talk Address Group	X	1	0	X	X	X	X	X	X	X	X	X	X	X	X	X	M	AD	10
UCG	Universal Command Group	X	0	0	1	X	X	X	X	X	X	X	X	X	X	X	X	M	UC	10
UNL	Unlisten	X	0	1	1	1	1	1	1	X	X	X	X	X	X	X	X	M	AD	10
UNT	Untalk	X	1	0	1	1	1	1	1	X	X	X	X	X	X	X	X	M	AD	10

Symbols:

Type
- U = Unlike message
- M = Multiline message

Class
- AC = Addressed Command
- AD = Address (talk or listen)
- DD = Device Dependent
- HS = Handshake
- UC = Universal Command
- SE = Secondary
- ST = Status
- 0 = logical zero (HIGH Signal Level)
- 1 = logical one (LOW Signal Level)
- X = don't care (for the coding of a received message)
- X = must not drive (for the coding of a transmitted message)

Notes:

(1) D1-D8 specify the device dependent data byte.

(2) E1-E8 specify the device dependent code used to indicate the EOS message.

(3) L1-L5 specify the device dependent bits of the device listen address.

(4) T1-T5 specify the device dependent bits of the device talk address.

(5) S1-S5 specify the device dependent bits of the device secondary address.

(6) S specifies the sense of the PPR.

S	Response
0	0
1	1

P1-P3 specify the PPR message to be sent when a parallel poll is executed.

P3	P2	P1	PPR Message
0	0	0	PPR1
.	.	.	.
.	.	.	.
.	.	.	.
1	1	1	PPR8

(7) D1-D4 specify don't care bits that must be sent as zeroes, but do not need to be decoded by the receiving device.

(8) S1-S6, S8 specify the device dependent status. (DIO7 is used for the RQS message.)

(9) The true message value must be ignored when received if the LACS is inactive.

(10) The true message value must be ignored when received if the ATN message is false.

(11) Interface protocol specifies that the IDY message is sent true, whereas the End message is sent only when the ATN message is sent false.

The three-wire handshake has three important characteristics which give the interface system wide flexibility.

First, the data transfer is asynchronous, thus avoiding inherent timing restrictions. Data can be transferred at any rate up to 1 megabyte per second that is suitable to the device on the bus.

Second, the handshake allows the interconnection of devices which operate at different input/output speeds. Data is transferred automatically at the speed which can be handled by the slowest active device on the bus.

Third, more than one device can accept data simultaneously.

A handshake cycle is illustrated in a step-by-step manner in Figure 2-8.

Both the source and acceptor power-up, or start, in a known quiescent state; thus the source has DAV high indicating the data on the bus is not valid, and the acceptor sets both NRFD and NDAC passive true (low) indicating that data has neither been accepted nor is the listener ready to accept data. If both NRFD and NDAC are sensed high by the source, an error condition exists.

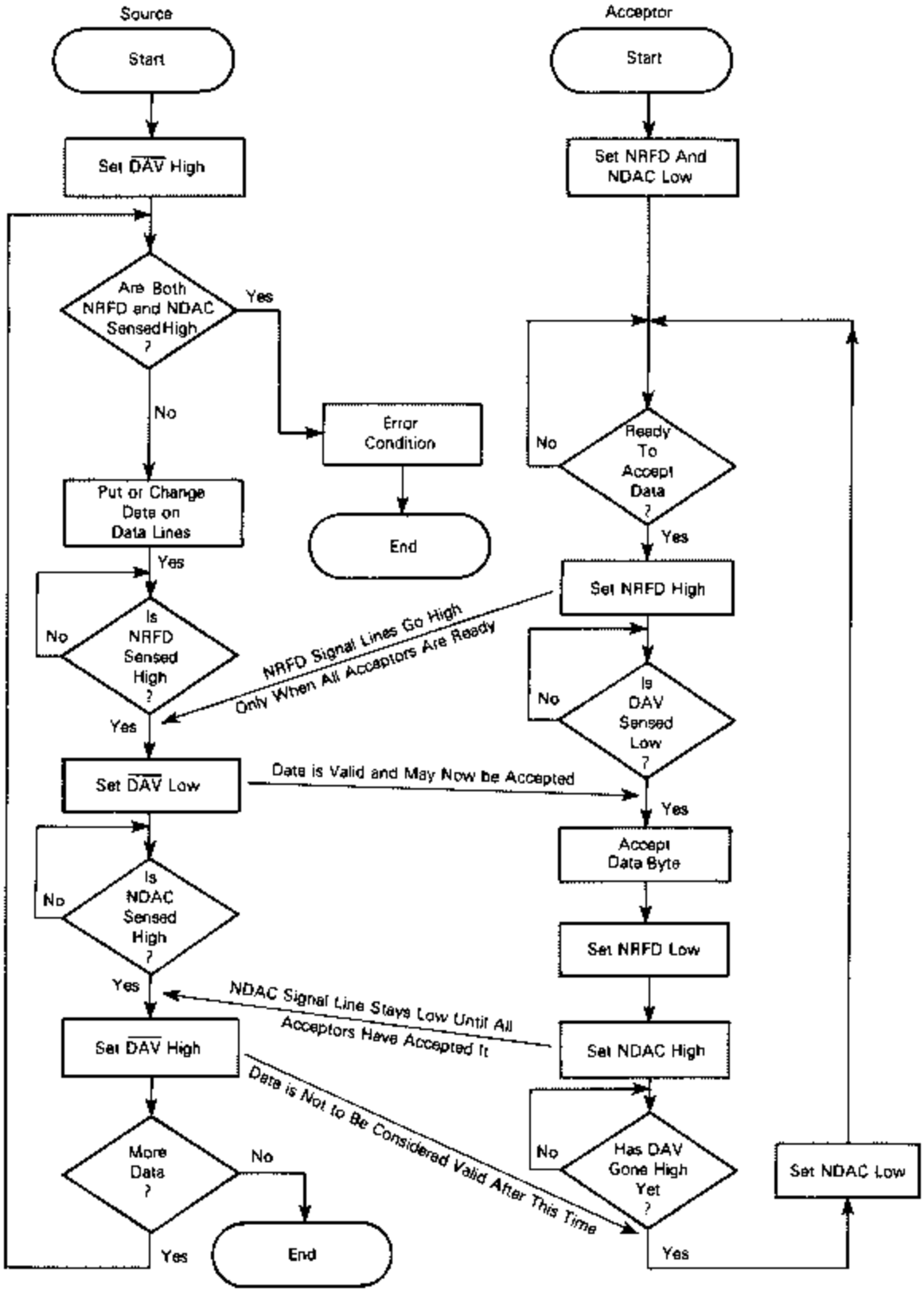
A source will indicate that valid data is available on the DIO lines by setting DAV low. When this event occurs, the acceptor(s) respond by pulling NRFD low. During this time data transfer takes place. Since each acceptor accepts data at different rates, their NDAC handshake lines are set high accordingly. The source will not sense NDAC as high until the slowest acceptor has responded. At this time NDAC goes high, indicating that all the acceptors have accepted the data. When the source senses NDAC high, the source will set DAV high, indicating that data on the DIO lines is no longer valid. Upon DAV going high, the acceptors will return the NDAC line low. As data transfer continues, the cycle repeats.

No step in this sequence can be initiated until the previous step is completed. Thus, information can proceed as fast as devices can respond, but no faster than the slowest device that is presently addressed as active.

2.8.2 Timing of Handshake

The sequences of timing for a handshake and byte transfer are illustrated in Figure 2-9.

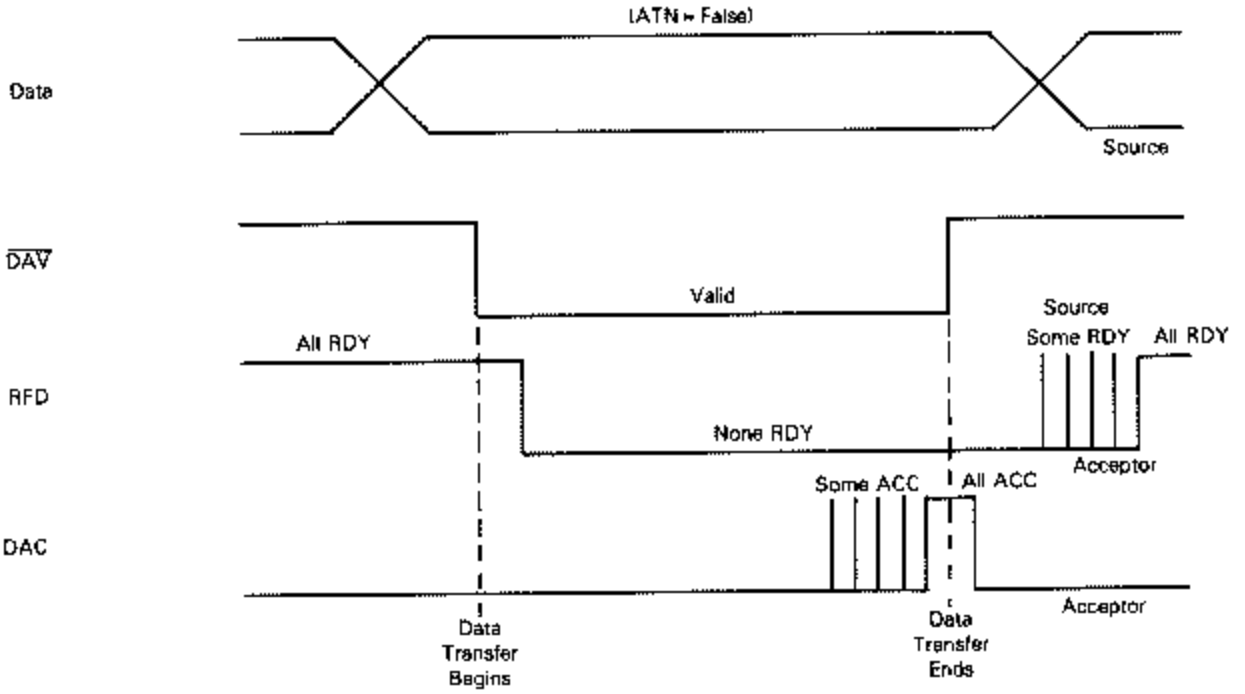
Details of the Source Handshake (SH) and Acceptor Handshake (AH) states are given in the State Diagram Section to follow.



Flow diagram outlines sequence of events during transfer of data byte. More than one listener at a time can accept data because of logical AND connection of NRFD and NDAC lines.

AD0105

Figure 2-8. Data Transfer



This diagram displays logical voltage levels on the MC68488 pins. The MC68488 pins are labeled as the complement of the specified 488 bus callout; i.e., $\overline{DAV}$ rather than DAV. RFD rather than NRFD and DAC rather than NDAC. This was done to stay with standard positive logic format, which is used with all M6800 family devices.

AD0106

Figure 2-9. Source and Acceptor Handshake Functions

2.9 STATE DIAGRAMS

2.9.1 Descriptions and Definitions

State diagrams are used throughout the IEEE-488 Standard to graphically illustrate the permissible transitions between states and the conditions required to effect such transitions. In the state diagram convention, each state is illustrated as a four-letter upper case mnemonic ending with an S, enclosed with a circle. Arrows connect the various states and the direction of the arrow indicates the path from one state to the next. Expressions beside the path define conditions which must be true to result in the transition along the path of the arrow.

A simple example will be given for illustration (see Figure 2-10). ABCS and XYZS are States. To cause a transition from State ABCS to XYZS, the logical OR of conditions R and S must be true; i.e., either R or S or both must be true.

NOTE

The logical OR function is represented by the symbol v and the logical AND function is represented by the symbol A.

A transition from ABCS to XYZS will occur if and only if R v S (logical OR) occurs. A transition from XYZS to ABCS will occur if and only if D A A (logical AND) occurs.

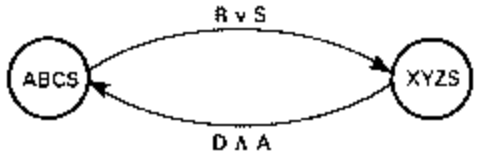


Figure 2-10. State Diagram

2.9.2 States

A state diagram is given in this section for each of the major interface functions. These state diagrams (Figures 2-11 through 2-21), represent each of the interface functions as implemented by the MC68488 (GPIA). As such, they are derived from and encompass the IEEE-488 Standard state diagrams with the following changes:

1. The controller state diagram has been omitted;
2. The additional addressing mode capabilities of the GPIA have been added to the talker and listener state diagrams;
3. The response to the additional local messages designed into the MC68488 are indicated in the appropriate state diagram.

A brief description of the basic talker function as implemented by the MC68488 is given to illustrate the use of the state diagrams. Only the talker function (not extended talker) is covered in this example. Refer to Figure 2-13. When power is turned on (pon), the talker function enters its initial (idle) state (TIDS Λ SPIS) in which the device cannot present device-dependent messages on the GPIB; i.e., the instrument cannot send data or status. This state is also entered when the controller issues an Interface Clear (IFC) message or when the dat (disable talker) local message is sent.

The assurance that there can only be one talker active per system is achieved by assigning each talker function its unique address (MTA — My Talk Address), which it can distinguish from talk addresses assigned to other devices (OTA — Other Talk Address). The presentation of the MTA message (specifically $\overline{\text{apte}}$ Λ $\overline{\text{IFC}}$ Λ MTA Λ $\overline{\text{ATN}}$) brings the function to the Talker Addressed State (TADS), a necessary condition to be able to place device-dependent messages on the bus. The apte local message must be false as this message, when true, activates the talker extended mode of operation. If lsbe local message is sent, the talker function will respond to either of two consecutive address values; i.e., the least significant address bit is a don't care. In the above transition expression, the attention message (ATN) is used by the controller to distinguish interface messages from device-dependent messages.

When a talker function is in the TADS state, it can enter either the Talker Active (TACS) or Serial Poll Active (SPAS) state. It will enter TACS where the device can actively send device-dependent messages on the data bus conditioned by the attention false ($\overline{\text{ATN}}$) message. The TADS state is

restored when the attention message becomes true again. Thus, it is possible for the controller (at the end of a handshake sequence) to interrupt the stream of device-dependent messages being sent by the talker without introducing errors.

In normal operation, the MTA and ATN messages place the device in the TACS state. A device is conditioned to present status data to the bus when the function also is placed in the SPAS state via the Serial Poll enable command (SPE). Note that two states of the talker function are active concurrently, one in each of the two sets of mutually exclusive states. Ambiguity within the device is precluded since it cannot transmit talker bytes when SPE is presented. The serial poll mode of operation is used to obtain status data sequentially from a number of devices. This will be covered in greater detail in Chapter 4. The arbitrary program order of the SPE and MTA messages makes a very flexible status polling sequence possible.

Some simple systems containing two devices (e.g., a voltmeter and printer) may require a talker function in one device and a listener function in the other device with no separate controller function. In systems where there is no controller function, ATN is always false and therefore when ton is made true (via a local switch) the talker will enter TACS via TADS at once.

Finally, some devices that have both the talker and listener functions may want to have the talker automatically unaddressed when they are addressed as a listener. This is done in order to prevent the device from listening to its own talker or status data. This is accomplished by considering the listen address of the device as an OTA address.

Source Handshake (SH) is the interface which controls the starting and stopping of multiline message byte transfer.

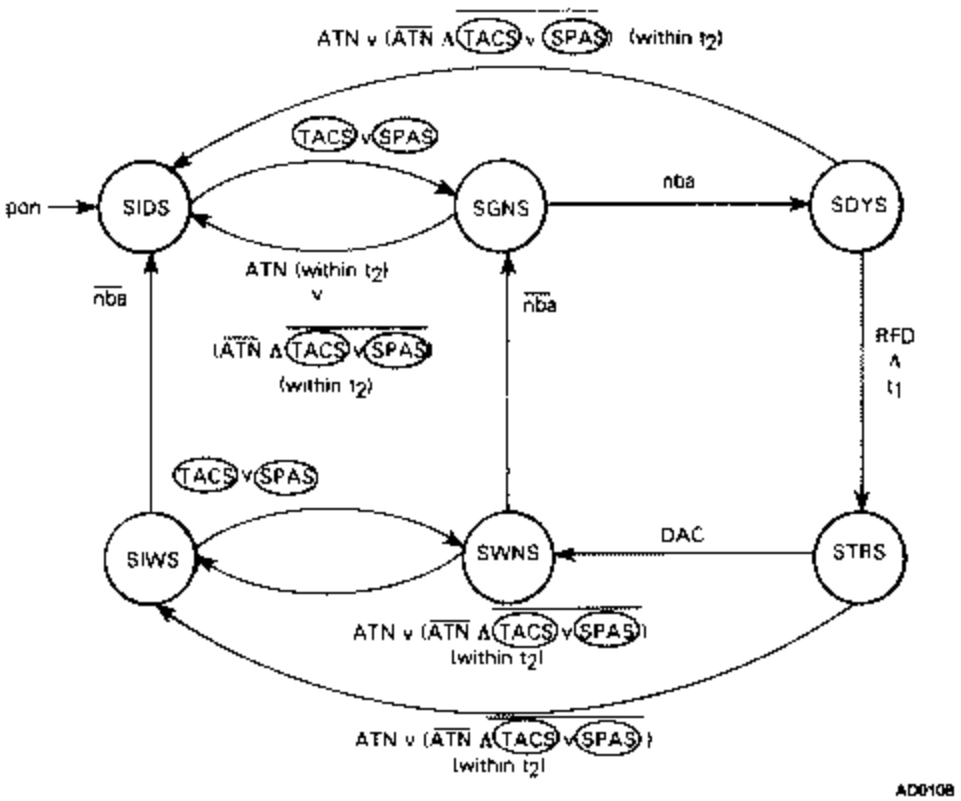
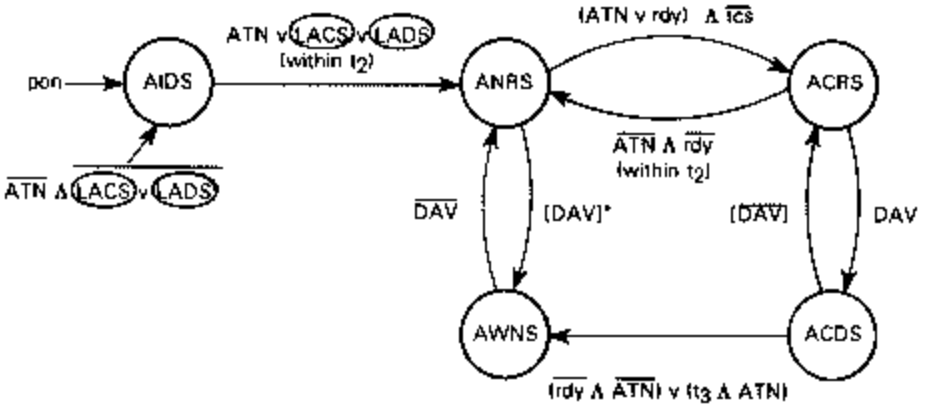


Figure 2-11. Source Handshake State Diagram

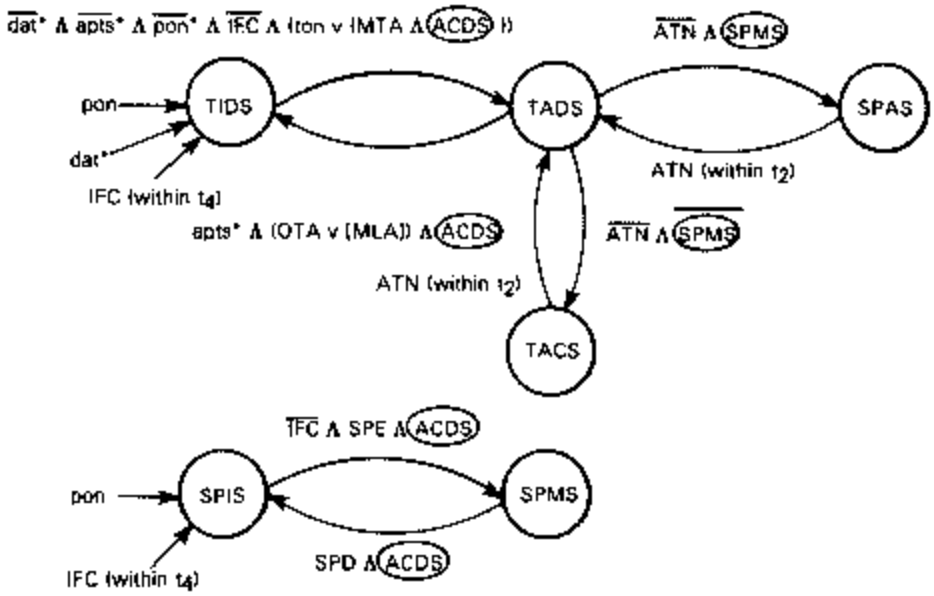
Acceptor Handshake (AH) allows for the proper reception of remote multiline messages with a device. The Acceptor Handshake function uses the DAV, NRFD, and NDAC lines to coordinate each message byte transfer.



Note: pon = $\overline{RESET}$ v software reset
:$\overline{RESET}$ = pin #19
:software reset = Bit 7, reg. 3
*This transition will never occur under normal interface operation; however, it may be implemented to simplify the interface function design.

Figure 2-12. Acceptor Handshake State Diagram AD0109

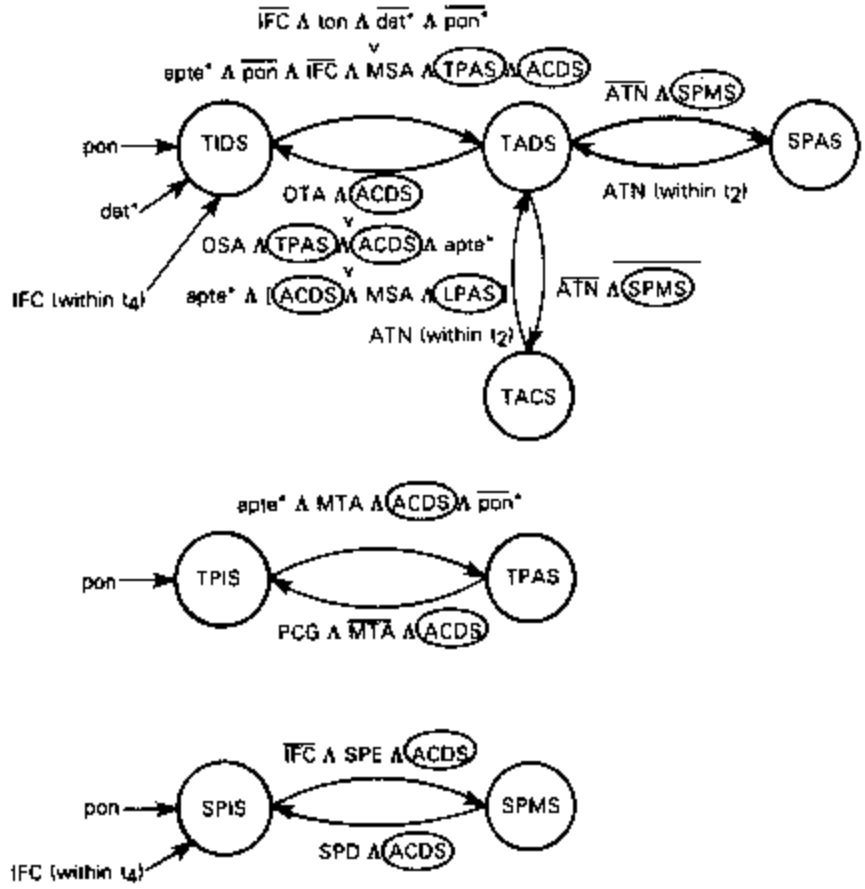
Talker (T)/Talker Extended (TE) allow the device to transmit data over the interface to other devices.



Note: If lsbe = 1 the LSB of talk address is a don't care.
*These local messages reflect additional mnemonics designed into the MC68488 and are not part of the IEEE-488 Standard.

AD0110

Figure 2-13. Talker State Diagram (apte = 0, dat = 0)

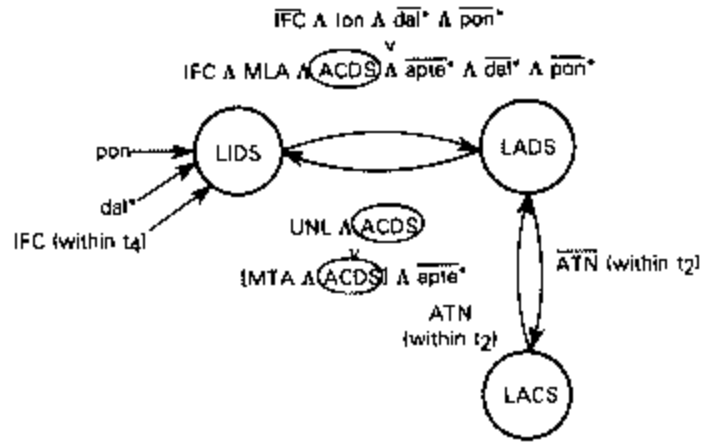


*These local messages reflect additional mnemonics designed into the MC68488 and are not part of the IEEE-488 Standard.

AD0111

Figure 2-14. Talker Extended Diagram (apte = 1, dat = 0)

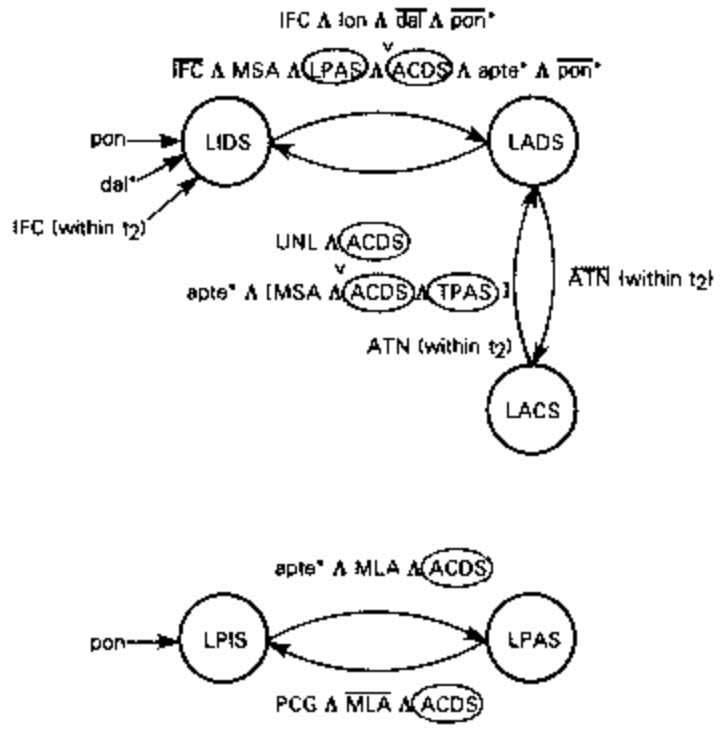
Listener (L)/Listener Extended (LE) provide the device with the ability to receive data over the interface bus from other devices.



Note: If lsbe = 1, LSB of listen address is a don't care.
*These local messages reflect additional mnemonics designed into the MC68488 and are not part of the IEEE-488 Standard.

AD0112

Figure 2-15. Listener State Diagram (apte = 0, dat = 0)

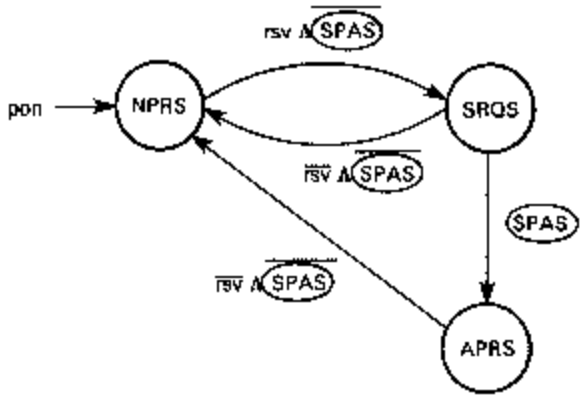


*These local messages reflect additional mnemonics designed into the MC68488 and are not part of the IEEE-488 Standard.

AD0113

Figure 2-16. Listener Extended State Diagram (apte = 1, dat = 0)

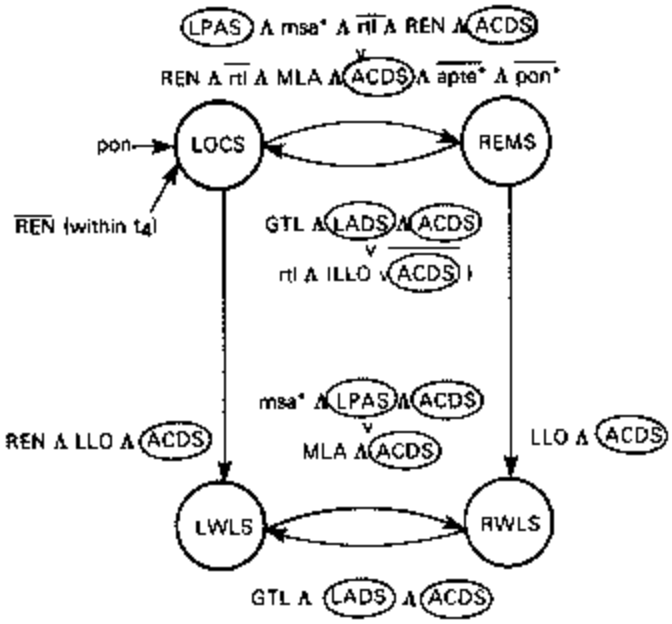
Service Request (SRQ) allows a device to request service from the interface controller.



AD0114

Figure 2-17. Service Request State Diagram

Remote Local (RL) allows a device to select from one of two sources of information. The source can be local (input information from the device front panel) or remote (input information from the interface).

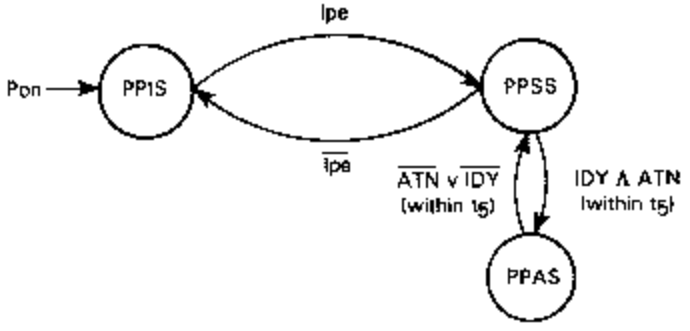


*These local messages reflect additional mnemonics designed into the MC68488 and are not part of the IEEE-488 Standard.

AD0115

Figure 2-18. Remote Local State Diagram

Parallel Poll (PP) allows a device to send one bit of status to the controller without having been previously addressed to be a talker.



lpe is accomplished by writing parallel poll word into Reg. #6.
lpe is accomplished by writing hex 00 into Reg. #6.

AD0116

Figure 2-19. Parallel Poll State Diagram

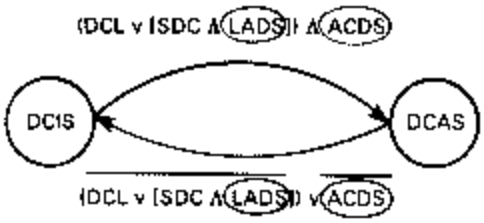


Figure 2-20. Device Clear State Diagram

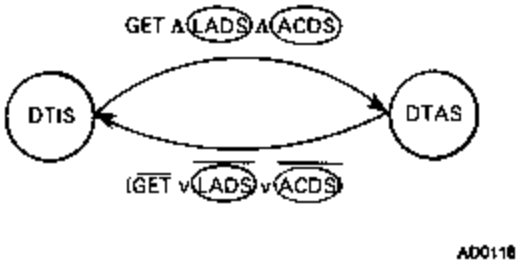


Figure 2-21. Device Trigger State Diagram

CHAPTER 3
GPIA FEATURES

3.1 ORGANIZATION

The MC68488 is an NMOS General Purpose Interface Adapter (GPIA) that provides the means to interface between the IEEE-488 Standard instrument bus and an 8-bit computer bus. It implements the necessary IEEE-488 Standard state diagrams for the Acceptor and Source Handshake, Talker, Talker Extended, Listener, Listener Extended, Service Request, Parallel Poll, Device Trigger, Remote Local, and Device Clear functions. In addition, the part implements other Non-IEEE-488 Standard defined functions such as: single and dual primary address recognition, RFD handshake "hold-off", DMA, talk only, listen only and programmable interrupts. Many instrument bus protocol features are automatically handled by the device and require minimal MPU response due to the state of the MC68488 and GPIB. The device is configured for the desired operation by programming it from the MPU data bus or the IEEE-488 Standard bus.

The GPIA is designed to be used with an MPU. The MPU is used to program the part (e.g., initialization or changing operations within the context of the IEEE-488 Standard protocol), to transfer device-dependent messages to and from the part (and hence the GPIB), and to monitor status of the GPIA and GPIB. The MC68488 responds to GPIB controller commands and conveys current status to an MPU through either an interrupt or polling sequence (see Chapter 4).

The MC68488 will automatically handle all handshake protocol on the GPIB. The device provides:

- MC6800 compatibility
- IEEE-488 Standard features:
  Talker, Talker Extend, Listener, Listener Extend
  Complete Source and Acceptor Handshake
  Serial and Parallel Poll
  Device Trigger, Device Clear, and Remote/Local
- Single or dual primary address recognition
- Secondary address capability
- RFD holdoff to prevent overrun
- Talk only or listen only capability
- DMA capability
- Complete static operation

The GPIA is contained in a 40-pin package and operates from a single 5-volt power supply. Figure 3-1 shows the pinout structure and Table 3-1 provides a description of the pinout signals.

Table 3-1. Description of MC68488 Pinouts

Symbol	Description
DB0-DB7	Allows Data Transfer Between the MPU and the MC68488
$\overline{IB0}$-$\overline{IB7}$	Allows Data Transfer Between the MC68488 and the 488 Bus
$\overline{CS}$	Chip Select
RS0-RS2	Register Selects
$\overline{IRQ}$	Interrupt Requests
$\overline{RESET}$	Used to Initialize the Chip During Power-On
DMA GRANT	Direct Memory Access Grant
DMA REQUEST	Direct Memory Access Request
$\overline{ASE}$	Address Switch Enable
DAC	Data Accepted
RFD	Ready for Data
$\overline{DAV}$	Data Valid
$\overline{ATN}$	Attention
$\overline{IFC}$	Interface Clear
SRQ	Service Request
$\overline{REN}$	Remote Enable
$\overline{EOI}$	End or Identify
T/$\overline{R}$1-2	Transmit/Receive Transceiver Control Outputs
E	Derivative of MPU ϕ2 Clock
R/$\overline{W}$	Read/Write Line from MPU
TRIG	Trigger Out Corresponds to GET and fget Command
V_{SS}	Ground
V_{DD}	+5 V Power Supply

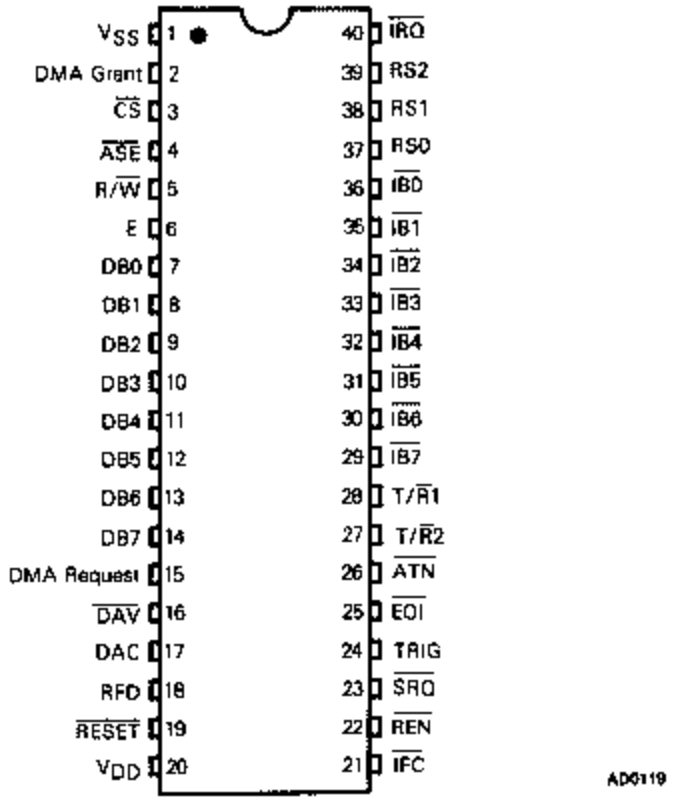


Figure 3-1. Pinouts for MC68488 GPIA

The MC3448A quad bus transceivers (Figure 3-2) or MC3447 Octal Bus Transceivers provide the electrical interface between the GPIB signal lines and the MC68488. It is with these transceivers that the electrical specifications (drive requirements, logic levels, receiver hysteresis, terminations, etc.) are met. The primary reason for using a separate IC for the electrical interface is to provide the 48 mA drive capability per signal line required by the IEEE-488 Standard. These driver ICs are coupled directly to the MC68488, and the proper direction of data flow is automatically selected by T/$\overline{R}$1 and T/$\overline{R}$2. The signals on T/$\overline{R}$1 and T/$\overline{R}$2 are generated by the MC68488.

3.1.1 Input/Output Functions

All inputs to the GPIA are high impedance and TTL compatible. All outputs from the GPIA are compatible with standard TTL. $\overline{IRQ}$ (Interrupt Request), however, is an open drain output (no internal pullup).

3.1.1.1 INTERFACE WITH MPU

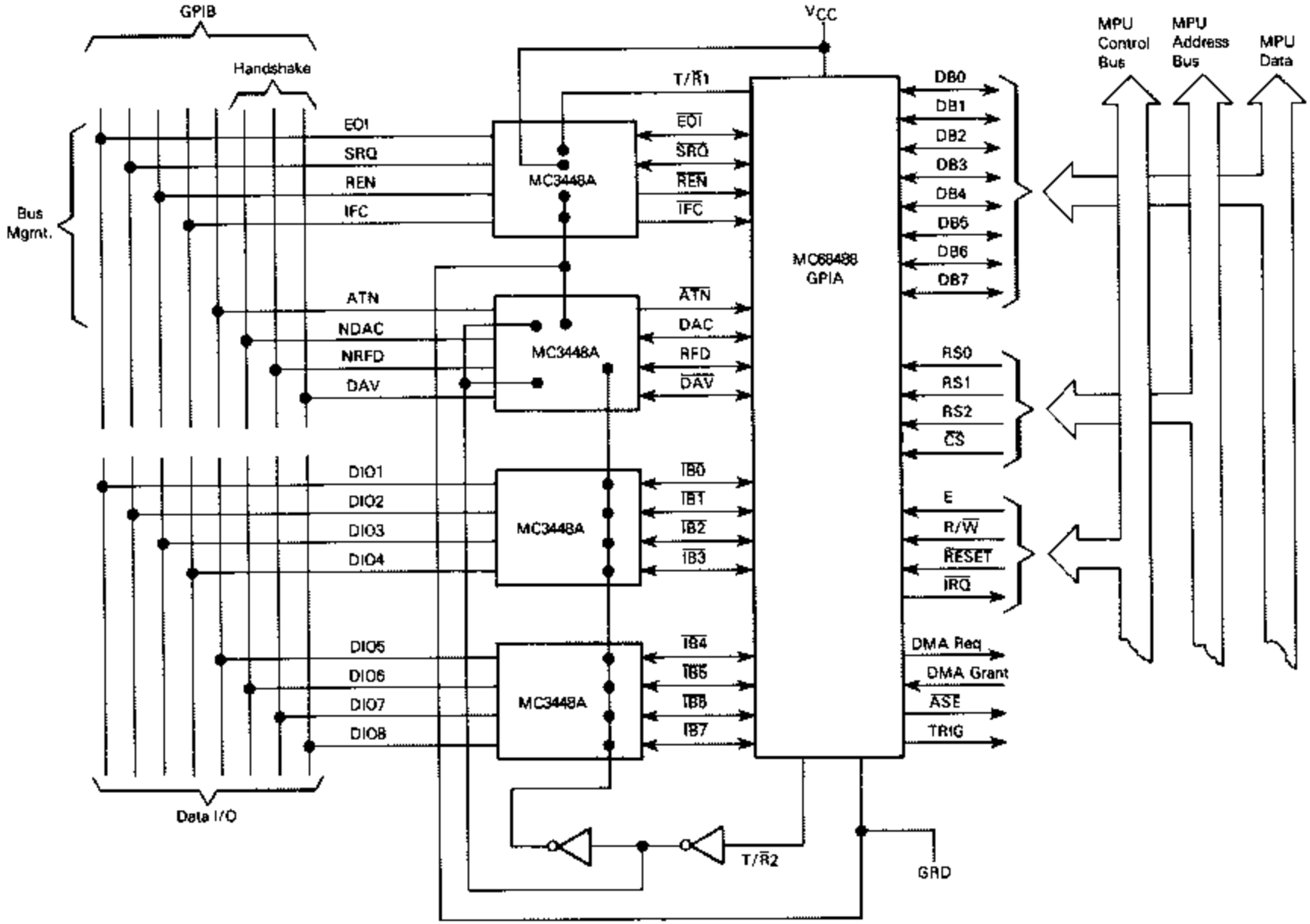
Bidirectional Data (DB0-DB7) — The bidirectional data lines allow the transfer of data between the MPU and GPIA. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs a GPIA read operation or the DMA controller performs a memory write operation. The Read/Write line is in the read high state when the GPIA is selected for a read operation.

Chip Select ($\overline{CS}$) — This input signal is used to select the GPIA. $\overline{CS}$ must be low for selection of the device. Chip Select decoding is normally accomplished with logic, external to the chip.

Read/Write Line (R/$\overline{W}$) — This signal is generated by the MPU or DMA controller to control register access and direction of data transfer on the data bus. A low state on the GPIA Read/Write and DMA Grant lines allows for the selection of one of seven write only registers when used in conjunction with register select lines RS0, RS1, and RS2. A high state on the GPIA Read/Write and low state on the DMA Grant line allows for the selection of one of eight read only registers when used in conjunction with register select lines RS0, RS1, and RS2.

Register Select (RS0, RS1, RS2) — The three register select lines are used to select the various registers inside the GPIA. These three lines are used in conjunction with the Read/Write line to select a particular register that is to be written or read. Table 3-2 shows the register select coding.

Interrupt Request ($\overline{IRQ}$) — The $\overline{IRQ}$ output goes to the common interrupt bus line for the MPU. This is an open drain output which is wire-ORed to the $\overline{IRQ}$ bus line. The $\overline{IRQ}$ is asserted Low when an enabled interrupt occurs and stays low until the MPU reads from the interrupt status register. Reading R0R will reset $\overline{IRQ}$ to the high state.



Note: The four MC3448A quad bus transceivers can be replaced by two MC3447 octal bus transceivers.

AD0120

Figure 3-2. MC68488 Interface Configuration

Table 3-2. Register Access

RS2	RS1	RS0	R/$\overline{W}$	Register Title	Register Symbol
0	0	0	1	Interrupt Status	R0R
0	0	0	0	Interrupt Mask	R0W
0	0	1	1	Command Status	R1R
0	0	1	0	Unused	—
0	1	0	1	Address Status	R2R
0	1	0	0	Address Mode	R2W
0	1	1	1	Auxiliary Command	R3R
0	1	1	0	Auxiliary Command	R3W
1	0	0	1	Address Switch*	R4R
1	0	0	0	Address	R4W
1	0	1	1	Serial Poll	R5R
1	0	1	0	Serial Poll	R5W
1	1	0	1	Command Pass-Through	R6R
1	1	0	0	Parallel Poll	R6W
1	1	1	1	Data In	R7R
1	1	1	0	Data Out	R7W

*External to MC68488

Reset — The $\overline{\text{RESET}}$ input provides a means of resetting the GPIA from a hardware source. In the Low state, the $\overline{\text{RESET}}$ input causes the following:

- The Interrupt "Mask" register is reset;
- All status conditions are reset;
- The GPIA is placed in the Untalk/Unlisten state;
- The Parallel Poll, Serial Poll, Data In, and Data Out registers are reset;
- The Address register and Address mode register are cleared;
- All stored conditions in the Auxiliary Command register except bit 7 are reset — (Bit 7 is to be set);
- T/$\overline{\text{R}}$1, 2 will go to the low state.

When $\overline{\text{RESET}}$ returns High (the inactive state) the GPIA will remain in the reset condition until the MPU writes bit 7 of the Auxiliary Command register (R3W) low. Prior to the release of the software reset bit, the only register that can be accessed is the Address Register. The conditions affected by the $\overline{\text{RESET}}$ pin cannot be changed while this pin is low.

E (Enable Clock) — E activates the address inputs (CS, RS0, RS1, and RS2) and R/$\overline{\text{W}}$ input and enables data transfer on the MPU data bus. It is also used internally as a state counter allowing the device to change interface states. E should be a free running clock such as the MC6800 ϕ2 clock.

3.1.1.2 GPIA/GPIB INTERFACE BUS SIGNALS. The GPIA provides a set of eighteen interface signal lines between the M6800 and the IEEE-488 Standard bus.

NOTE

The IEEE-488 Standard defines these signals as negative logic. In this document all MPU and MC68488 signals are defined as positive logic.

Signal Lines ($\overline{IB0}$-$\overline{IB7}$) — These bidirectional lines allow for the flow of seven bit ASCII interface messages and device dependent messages. Data appears on these lines in a bit-parallel byte-serial form. These lines are buffered by transceivers and applied to the IEEE-488 Standard bus (DIO1-DIO8).

Byte Transfer Lines (DAC, RFD, $\overline{DAV}$) — These lines allow for proper transfer of each data byte on the bus between sources and acceptors. RFD goes passively High indicating "Ready For Data". A source will indicate the "data is valid" by pulling $\overline{DAV}$ low. Upon the reception of valid data, DAC will go passively High indicating that the "data has been accepted" by all acceptors. The handshake lines have internal pullup resistors.

Bus Management Lines ($\overline{ATN}$, $\overline{IFC}$, $\overline{SRQ}$, $\overline{EOI}$, $\overline{REN}$) — These lines are used to manage an orderly flow of information across the interface lines.

Attention ($\overline{ATN}$) — is continuously monitored by the GPIA. The device resonds to any changes on this line in less than 200 ns by activating the transmit/receive control signals. if the $\overline{EOI}$ line and $\overline{ATN}$ are Low at the same time, GPIA will place the contents of the parallel poll register on the IEEE-488 Standard bus.

Interface Clear ($\overline{IFC}$) — is used by a system controller to put the GPIA in a known quiescent state. The occurance of $\overline{IFC}$ will place the GPIA in the Listener/Talker idle state (LIDS or TIDS). If the MC68488 is in a Listener Active state with a byte of data in the Data In Register (BI bit set) an $\overline{IFC}$ will place the part in LIDS but will not destroy the received byte nor the status indication (BI). Any interface function that requires the device to be in either the Listener or Talker Active state (e.g., a Serial Poll enable command) will be reset if an $\overline{IFC}$ occurs. A command that originates from the MPU (e.g., to, lo, fget, hlda) will only be affected during the occurance of an $\overline{IFC}$ (when $\overline{IFC}$ is low) and will return to its programmed state when $\overline{IFC}$ returns, high; i.e., $\overline{IFC}$ will not affect local messages. For example: if the GPIA is in TACS (Talker Active State) and has placed a byte in the Data-Out Register it has made a new byte available (nga). If $\overline{IFC}$ occurs while the source handshake is in SDYS, the talker function will be returned to its idle state but nba (a local message) will not be destroyed. When the GPIA is again made a talker, the byte in the Data-Out Register (placed there before $\overline{IFC}$) will be placed onto the GPIB, after completion of the handshake. The address register is not affected by an $\overline{IFC}$.

Service Request ($\overline{SRQ}$) — is used to indicate a need for attention in addition to requesting an interruption in the current sequence of events. This indicates to the controller that a device on the bus is in need of serivce. This output becomes active Low by setting the rsv bit (bit 6) or R5W. This line is an open drain and an external pullup resistor (3 kΩ) must be used.

Remote Enable (REN) — is used to select one of two alternate sources of device programming data — local and remote control. When this input is Low the GPIA is enabled to move to the REMS state. Note that REN being Low is a necessary but not a sufficient condition for moving to REMS. See Remote Local (RL) state diagram in Chapter 2.

End or Identify ($\overline{EOI}$) — serves a dual purpose. When the GPIA is in TACS and the MPU writes bit 5 or R3W (feoi) this pin becomes an output and signals the end of a multibyte transfer. If the system controller makes the $\overline{EOI}$ line true in conjunction with $\overline{ATN}$, the contents of the Prallel Poll register will be placed on the IEEE-488 Standard bus.

Transmit/Receive Control Signals (T/$\overline{R}$1, T/$\overline{R}$2) — These two signals are used to control the quad or octal transceivers which drive the interface bus. It is assumed that transceivers equivalent to the MC3447 or MC3448A will be used where each transceiver has a separate Transmit/Receive control pin. These pins can support one TTL load each. The outputs can then be grouped and the control for $\overline{SRQ}$ hardwired high to transmit. The Transmit/Receive inputs of $\overline{REN}$, $\overline{IFC}$, and $\overline{ATN}$ are hardwired low to receive. $\overline{EOI}$ is controlled by T/$\overline{R}$1 through the MC3447/MC3448A (or equivalents) allowing it to transmit or receive. T/$\overline{R}$1 operates exactly as T/$\overline{R}$2 except during the parallel polling sequence. During parallel poll, $\overline{EOI}$ will be made an input by T/$\overline{R}$1 while $\overline{DAV}$ and IB0/IB7 lines are outputs.

3.1.1.3 CONTROL SIGNAL

DMA Control Lines (DMA Grant, DMA Request) — The DMA request line is used to signal a DMA controller that a data transfer is pending. The DMA request line is set high if either the BI or BO status bits are set in the Interrupt Status Register (R0R). The DMA request line is cleared when the DMA Grant is true. The DMA Grant line is used to signal the GPIA that the DMA Grant is true. The DMA Grant line used to signal the GPIA that the DMA has control of the MPU data and address lines. The DMA GRANT, when set high, selects register 7. It also inhibits the RS0, RS1, and RS2 lines. During this time the $\overline{CS}$ input must be High. the DMA GRANT also inverts the function of the R/$\overline{W}$ line makeing it $\overline{R}$/W. Thus,if the DMAC supplies a write function to a memory location, this same line will perform a read of the GPIA (R7R) and vice versa.

NOTE

DMA GRANT MUST BE GROUNEDED WHEN NOT IN USE!

Trigger Output (TRIG) — The TRIG pin provides an output corresponding to the GET and fget commands. A hardware or software reset places this output at a low level. The trigger output can be programmed high by either of two methods:

1. Setting fget (bit 0 of R3W) by the MPU causes the trigger output to be set. It remains set until the fget bit is programmed low or until a reset occurs.
2. The Trigger Output is set upon reception of a GET command from the controller. It is reset when the GPIA moves out of DTAS (Device Trigger Active State); ie.., when GET, LADS, or ACDS occur.

Address Switch Enable (ASE) — The ASE output is used to enable three-state buffers that connect instrument address switches to the MPU data bus. This output pin is pulsed low when the Address Switch Register of the GPIA is read (R4R), i.e., a read of R4R will drive the ASE line low for the E clock that is used to read R4R.

3.1.2 Logic Conventions

The GPIB is a negative logic bus as described in Chapter 2. Thus, a binary 1 is represented by a low voltage (0 volt) and a binary 0 by a high voltage (5 volts). This is the inverse of the notation used on the MC6800 MPU data bus (Figure 3-3).

1 = high 0 = low	DB7 1	DB6 0	DB5 1	DB4 0	DB3 1	DB2 1	DB1 0	DB0 1	MPU Write To MC68488 (Positive Logic)
1 = high 0 = low	$\overline{IB7}$ 0	$\overline{IB6}$ 1	$\overline{IB5}$ 0	$\overline{IB4}$ 1	$\overline{IB3}$ 0	$\overline{IB2}$ 0	$\overline{IB1}$ 1	$\overline{IB0}$ 0	MC68488 Write To GPIB Transceivers (Positive Logic)
1 = low 0 = high	DIO8 1	DIO7 0	DIO6 1	DIO5 0	DIO4 1	DIO3 1	DIO2 0	DIO1 1	Data on GPIB (Negative Logic)

AD0121

Figure 3-3. Logic Notation Comparison Between MPU and GPIB

The GPIA handles the inversion of the data bus internally. Thus, each of the data bits on the GPIB will be the inverse of its corresponding MPU data bus bit. The GPIB being a negative logic bus will be completely transparent to the user. A data byte in the talker device will be sent by the MPU to the talker MC68488 as a positive logic data byte. The talker MC68488 will invert it and at the appropriate time place it on the GPIB. The listener MC68488 will receive the negative logic data byte, invert it and place it in the GPIA Data In Register. The listener device MPU will receive a positive logic byte from the MC68488 in the same format as sent by the talker MPU. The MC68488 inverts all GPIB data bytes before placing them on the MPU data bus, even those that are received through the Command Pass Through Register.

Also note that IEEE-488 Standard uses the notation (DIO1-DIO8) to represent the individual bits on the GPIB. Motorola's MC6800 family uses DB0-DB7 as its MPU Data bus notation. The interpretation is as follows: the least significant bit on the GPIB is DIO1 while the counterpart on the MPU bus is DB0 and so on up to the most significant bit DIO8 and DB7.

3.1.3 GPIA Registers

The MC68488 has fifteen register locations accessible to the MPU data bus which are used for reading and writing data to and from the IEEE-488 Standard bus, transferring data to implement various functions, and providing current status to the GPIA and GPIB. There are seven Write Only and eight Read Only registers accessed according to the three least significant bits of the MPU address bus and the status of the R/$\overline{W}$ line (Table 3-2). The write registers of the MC68488 are programmed by the MPU. The read registers reflect data commands, and addresses from the IEEE-488 Standard bus side of the GPIA. Many of the bits in the read registers are set by internally generated states of the GPIA. The major IEEE-488 Standard bus standards are realized using the Data Input

Register, Data Output Register, Parallel Poll Register, Serial Poll Register, and Address/Address Switch Register. The reset of the registers is used to set flags for commands, addresses and to indicate the condition of the IEEE-488 Standard bus. The data In/Out Registers are used to write data from the MPU to the IEEE-488 Standard bus (Out) or to receive data from the IEEE-488 Standard bus (In). Figure 3-4 shows a block diagram of the MC68488.

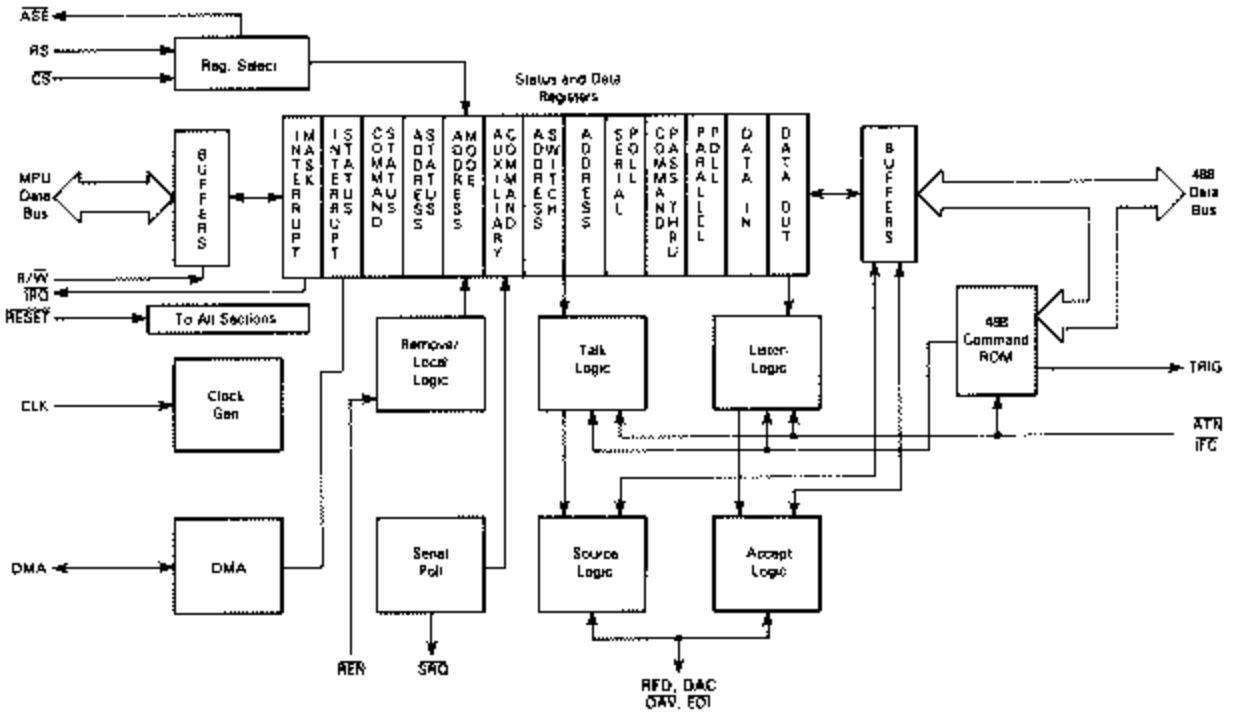


Figure 3-4. Block Diagram of MC68488

The Address Switch Register (R4R) is external to the part, but this address location is provided for reading a set of external address switches (see description of $\overline{ASE}$ line). Table 3-3 shows the actual bit contents of each of the registers.

3.1.3.1 INTERRUPT STATUS REGISTER. All of the bits in this register are "level sensitive."*

Interrupt Status Register (R0R)
(Read Only)

7	6	5	4	3	2	1	0
INT	BO	GET	1	APT	CMD	END	BI

*Level Sensitive status bit means that the bit being set is a result of the current levels of the appropriate conditions rather than a latched situation. When any of the conditions causing the set condition (e.g., $\overline{ATN}$ becoming false or a special command being removed from the data bus) are removed, the status bit will return to zero. Level sensitivity applies to certain bits in the Interrupt and Command Status Registers.

Table 3-3. GPIA Register Contents

	7	6	5	4	3	2	1	0	
R0W	IRQ	BO	GET		APT	CMD	END	BI	Interrupt "Mask Register"
R0R	INT	BO	GET		APT	CMD	END	BI	Interrupt Status Register
R1R	UACG	REM	LOK		RLC	SPAS	DCAS	UUCG	Command Status Register
R1W									Unused
R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS	Address Status Register
R2W	dsel	to	lo		hlde	hlda		apte	Address Mode Register
R3R	RESET	DAC	DAV	RFD	msa	rtl	ulpa	fget	Auxiliary Command Register
R3W		rfdr	feoi	dacr			dacd		Auxiliary Command Register
R4R	UD3	UD2	UD1	AD5	AD4	AD3	AD2	AD1	Address Switch Register
R4W	lsbe	del	dat	AD5	AD4	AD3	AD2	AD1	Address Register
R5R	S7	SRQS	S5	S4	S3	S2	S1	S0	Serial Poll Register
R5W		rsv							
R6R	B7	B6	B5	B4	B3	B2	B1	B0	Command Pass-Through Register
R6W	PPR8	PPR7	PPR6	PPR5	PPR4	PPR3	PPR2	PPR1	Parallel Poll Register
R7R	DI7	DI6	DI5	DI4	DI3	DI2	DI1	DI0	Data In Register
R7W	DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0	Data Out Register

Notes:
1. Upper case letters indicate a message resulting from the IEEE-488 Standard bus.
2. Lower case letters indicate a message resulting from the MPU data bus.
3. The bit terminology of the Data In and Data registers represent the numbering of the IEEE-488 Standard bus and not the 6800 MPU bus — see Section 3.1.2.

b0 Byte In (BI) — This bit indicates when data can be read from the GPIA Data In Register by the MPU. BI is set when the GPIA is addressed as an Active Listener and data from the IEEE-488 Standard bus is present in R7R. The DAC handshake is also held up. BI is reset and the handshake completed when R7R is read by the MPU. A release of the handshake (by dacr) is a reconfiguration of the GPIA by the controller (e.g., $\overline{\text{IFC}}$) will not reset the BI bit.

b1 END — This bit is used to indicate the reception of the final byte of data in a multibyte transfer. This bit is set when the $\overline{\text{EOI}}$ line is Low and $\overline{\text{ATN}}$ is High. It is reset when $\overline{\text{EOI}}$ is High or $\overline{\text{ATN}}$ is Low. This bit is not qualified with the handshake and will be set as soon as the $\overline{\text{EOI}}$ line is low. The reception of the final data byte occurs after END is set and causes the BI status bit to be set.

b2 Command (CMD) — When set, this bit indicates that the controller has sent a command which may require special recognition or action by the MPU. The category or special commands received can be obtained by reading the Command Status Register or Serial Poll Register. This bit is set when the following conditions are met: CMD = SPAS v RLC v dsel Λ (DCAS v UUCG v UACG). dsel is a bit in the Address Mode Register which is used by the MPU to deselect DCAS, UUCG, and UACG from causing a command interrupt and holding the handshake. CMD is reset when the condition in the Command Status Register that caused CMD to be set is reset.

b3 Address Pass Through (APT) — When set, this bit indicates, in the secondary address mode, that a secondary address can be read from the Command Pass-Through Register. This bit is set when the apte bit is set (placing the device in the secondary address mode), the GPIA is in LPAS or TPAS, and a seconary address has been received by the GPIA. This bit is reset when the handshake is completed in the secondary address protocol (writing dacr high after examining the secondary address).

b4 — This bit is not used in the status register and is always in the high state. This allows the MC6800 TAP instruction to be used in examining the status register without resetting the microprocessor interrupt mask.

b5 Group Execute Trigger (GET) — When set, this bit indicates the reception of the Group Execute Trigger command from the controller. When this bit is high, the Trigger output pin will be High. This bit is set as long as the GPIA is in DTAS (Device Trigger Active State). That is, the bit is set when dsel (bit 7, R2W) is low, the GPIA is in LADS (Listener Address State), the GET command is TRUE, and the GPIA acceptor handshake is in ACDS. The handshake will also be held-off. The GET bit will be reset and the handshake released if the MPU writes dacr high in the Auxiliary Command Register (R3W). If dsel (bit 7, R2W) is set, this bit is inhibited and does not cause an Interrupt Request ($\overline{IRQ}$). The handshake is automatically completed. The GET bit is not set by writing a one into fget (R3W).

b6 Byte Out (BO) — When set, this bit indicates to the MPU that another byte can be written to the Data Out Register (R7W). This bit indicates completion of the handshake when the GPIA is a talker. The BO bit is set upon initialization as a talker or upon reception of DAC by all the listeners when the GPIA is in TACS. This bit is reset when data is written to the Data Out Register (R7R). Byte Out is also Low when $\overline{ATN}$ is Low.

b7 Interrupt Bit (INT) — This bit is set to indicate the state of the status register when the appropriate bits are enabled in the Interrupt Mask Register (R0W). INT = [BI ∧ BI(M)] ∨ [END ∧ END (M)] ∨ [CMD ∧ CMD (M)] ∨ [APT ∧ APT (M)] ∨ [GET ∧ GET (M)] ∨ [BO ∧ BO (M)]. "M" represents the corresponding bit in the Interrupt Mask Register. Thus, the INT bit in R0R is the result of the logic ORing of bits 0, 1, 2, 3, 5 and 6 in R0R being ANDed with their respective bits in R0W. If IRQ (M) is set in R0W, the INT bit in R0R, when set, will cause the $\overline{IRQ}$ output pin to become active (low). The $\overline{IRQ}$ pin goes to the inactive state (high) by reading R0R.

The IRQ output (from R0W) is edge sensitive to occurrence of an INT status condition; i.e., if a second condition occurs to make the INT status high, but INT is already high, the second condition will not cause IRQ to go low. When servicing this register through a software interrupt routine, the final instructions should reset the interrupt mask register followed by programming the interrupt mask register to its original state. Should a second interrupt 2 condition be pending, this sequence causes the necessary edge transition and an $\overline{IRQ}$ will occur (see Section 3.3.2 for further details).

3.1.3.2 INTERRUPT MASK REGISTER. The Interrupt Mask Register is used to select which bit(s) in R0R will cause an interrupt. Each bit is ANDed with a corresponding bit in R0R. This provides an MPU controllable mask for the interrupt. Bit 4 is unused.

Interrupt Mask Register (R0W)
(Write Only)

7	6	5	4	3	2	1	0
IRQ	BO	GET	X	APT	CMD	END	BI

3.1.3.3 COMMAND STATUS REGISTER. The bits in this register reflect Special Command status. All the bits except the RLC bit are "level sensitive."* RLC is edge sensitive and will go high whenever a remote-to-local or local-to-remote transition is detected.

Command Status Register (R1R)
(Read)

7	6	5	4	3	2	1	0
UACG	REM	LOK	1	RLC	SPAS	DCAS	UUCG

b0 Undefined Universal Command Group (UUCG) — When this bit is set, the GPIA has received a command which is to be interpreted by the MPU. This bit will be set anytime the command X001XXXX is sent by the controller. The handshake will also be held off. The UUCG causes the CMD bit in the Interrupt Status Register to be set if dsel is zero (bit 7, R2W). The command byte can be read from the Command Pass-Through Register (R6R). The UUCG bit and its corresponding CMD bit if selected will be reset and the handshake released by writing dacr (bit 4, R3W) high. If dsel is set to a "1", an undefined Unviersal Command will not cause this bit to be set.

b1 Device Clear Active State (DCAS) — This bit is set whenever the GPIA enters the Device Clear Active State. This state will be entered if either of the two following conditions result, and the dsel is set Low:

1. $\overline{\text{ATN}}$ on the GPIB is Low and the Device Clear Command (X0010100) is sent (duration of command).
2. $\overline{\text{ATN}}$ on the GPIB is Low, the GPIA is in the LADS, and the Selective Device Clear Command (X0000100) is sent (duration of command).

These conditions will also set the CMD interrupt bit if dsel (bit 7, R2W) is zero. The DAC handshake will be held off when either of the above two conditions occur. The DCAS bit and CMD bit, if enabled, will be reset and the handshake released by writing dacr (bit 4, R3W) high.

b2 Serial Poll Active State (SPAS) — When set, this bit indicates that the GPIA is in the Serial Poll Active State. This bit is set when the GPIA is an addressed talker, the Serial Poll Enable Command (X0011000) is sent and then $\overline{\text{ATN}}$ is High. The SPAS bit will always set the CMD bit in the Interrupt Status Register. This bit is reset when the GPIA leaves SPAS.

b3 Remote Local Change (RLC) — This bit, when set, indicates a change has occurred in the REM bit (bit 6, R1R) unless the REM bit was reset via the rtl local message. The RLC bit sets the CMD bit in the Interrupt Status Register. Reading the Command Status Register resets the RLC bit. Only the REM bit affects RLC. The rtl local message will reset the REM bit if LOK is not set but this action will not set the RLC bit.

b4 - Bit 4 is unused, and is always a logical 1.

b5 Local Lockout Enable (LOK) — This bit is set when REN is made true and then the Local Lockout Command (X0010001) is sent. This bit is reset when REN is high.

*Level Sensitive status bit means that the bit being set is a result of the current levels of the appropriate conditions rather than a latched situation. When any of the conditions causing the set condition (e.g., $\overline{\text{ATN}}$ becoming false or a special command being removed from the data bus) are removed, the status bit will return to zero. Level sensitivity applies to certain bits in the Interrupt and Command Status Registers.

b6 Remote Enable (REM) — This bit, when set High, indicates to the device that the control function is to be performed remotely. This bit is set High when the REN uniline message is first made Low and then the GPIA is made a listener. The REM bit remains High until the REN uniline message goes High or until the rtl bit (bit 2, R3W) is written High, or the Go to Local Command (X0000001) is sent by the controller. Any change in this bit, unless it is caused by the rtl local message, causes the RLC bit to be set.

b7 Undefined Address Command Group (UACG) — When set, this bit indicates to the MPU that the GPIA has received an undefined address group command and that this byte is in the Address Pass-Through Register (R6R). This bit is set anytime dsel (bit 7, R2W) is low and the Addressd Command Group (X000XXXX) is received excluding the GTL, GET and SDC Commands. The handshake is also held off. If dsel (bit 7, R2W) is zero, the UACG causes the CMD Interrupt Status bit to be set. The UACG bit and corresponding CMD, if enabled, is reset and the handshake released by writing dacr (bit 4, R3W) high. If dsel is set to a "1", UACG will not set the UACG status bit. When dsel = 1, the UACG is ignored by the MC68488 and the handshake will automatically be completed.

3.1.3.4 ADDRESS STATUS REGISTER

Address Status Register (R2R)
(Read Only)

7	6	5	4	3	2	1	0
ma	to	lo	ATN	TACS	LACS	LPAS	TPAS

b0 Talker Primary State (TPAS) — This bit is only used in the secondary (extended) address mode (apte bit = 1) and indicates, when set, the reception of the primary talker address. It remains set until another primary command is sent over the IEEE-488 Standard bus or until the GPIA is removed from the talker mode. This bit is not set when in the primary address mode (apte bit = 0).

b1 Listener Primary Address State (LPAS) — This bit is only used in the secondary address mode (apte bit = 1) and indicates, when set, the reception of the primary listener address. It remains set until another primary command is sent over the IEEE-488 Standard bus or until the GPIA is removed from the listener mode. This bit is not set when in the primary address mode (apte bit = 0).

b2 Listener Active State (LACS) — This bit is set when the GPIA is in LACS. The GPIA can be placed in LACS by reception of MLA from the controller or by the MPU setting lo (bit 5, R2W) high and $\overline{ATN}$ being high. The GPIA is now ready to receive data. A received data byte is indicated by the BI bit in R0R being set. ($\overline{IFC}$ and $\overline{ATN}$ will overide lo for the duration of $\overline{IFC}$.) The LACS bit is reset from the IEEE-488 Standard bus by $\overline{ATN}$ becoming low, $\overline{IFC}$, and by the unlisten command (if lo is not set), or the GPIA being addressed as a Talker. The MPU can remove the GPIA from LACS by setting dal (bit 6, R4W) high (this will override lo for the duration of dal being high).

b3 Talker Active State (TACS) — This bit is set when the GPIA is in TACS. The GPIA can be placed in TACS by reception of MTA from the controller or by the MPU setting to (bit 6, R2W) high and $\overline{ATN}$ going high. The GPIA is now ready to transmit data. The BO-bit in R0R is set at this time. The TACS bit is reset from the IEEE-488 Standard bus for the duration of $\overline{ATN}$ being low, by

the occurrence of an IFC (if the to bit is set TACS will only be low for the duration of $\overline{IFC}$), and by the Untalk Command (if the to bit is not set). The GPIA being addressed to listen also resets the TACS bit. The MPU can remove the GPIA from TACS by setting dat (bit 6, R4W) High (this will override the to bit for the duration of dat being high).

b4 Attention (ATN) — This bit is the inverse of the $\overline{ATN}$ line of the IEEE-488 Standard bus. This bit is set when the controller asserts the $\overline{ATN}$ uniline message Low and is reset when the controller makes $\overline{ATN}$ High.

b5 listen only (lo) — When this bit is set, the GPIA is in the listen only mode. It is set and reset by the MPU programming bit 5 of R2W high or low, respectively.

b6 talker only (to) — When this bit is set, the GPIA is in the talk only mode. It is set and reset by the MPU programming bit 6 of R2W High and/or Low, respectively.

b7 my address has occurred (ma) — This bit is set automatically when the controller has addressed the GPIA or either the to or lo bit has been set. The ma bit will be set when the GPIA is in TACS, TADS, LACS, LADS or SPAS. This bit is reset from the IEEE-488 Standard bus by IFC, the Untalk Command (if previously in TACS or TADS), or the Unlisten Command (if previously in LACS or LADS).

3.1.3.5 ADDRESS MODE REGISTER

Address Mode Register (R2W)
(Write Only)

7	6	5	4	3	2	1	0
dsel	to	lo	X	hlde	hlda	x	apte

b0 address pass-through enable (apte) — This bit is set by the MPU to place the GPIA in the extended (secondary) addressing mode. This should be done prior to reception of the primary address. When apte = 1, the GPIA will expect a secondary address (X11XXXXX) to follow its primary address. The part responds to the secondary address by setting APT in R0R and holding off the DAC handshake. The apte bit is reset by writing it low or by a chip reset.

b1 — This bit is unused and is undefined.

b2 hold-off RFD on all data (hlda) — This bit, when set, holds off the RFD handshake on reception of each data byte when the GPIA is in the listener mode. The reception of data occurs in the normal sequence. However, the handshake is not released by simply reading the Data In register (DAC only is released). Rather, RFD is released by the MPU writing rfdr (bit 6 of R3W) high after R0R is read. The hlda bit is set and reset by the MPU writing this bit high or low, respectively. The hlda bit being set causes an RFD latched condition for the next received data byte. This latched condition is released by writing rfdr high, but is again latched as long as the hlda bit is set. To remove the hlda condition, the hlda bit must be written low.

b3 hold-off RFD on end (hlde) — This bit is used, when in the listener mode, to hold-off the RFD handshake when an EOI uniline message is received by the GPIA. If this bit is set and a data byte is received while EOI is Low, the RFD handshake line is not released until rfdr (bit 6, R3W) is written high by the MPU. The hlde bit is set and reset by the MPU writing this bit high or low, respectively. The hlde bit being set causes a latched condition in the GPIA for a data byte received when EOI is true. This latched condition is released by writing rfdr high but is again latched as long as the hlde bit is set. To remove the hlde condition, the hlde bit must be written low followed by writing the rfdr bit high, thus removing the stored condition. If rfdr is not written high, the next EOI received by the GPIA in the listener state, even if the device was made a talker and then returned, will holdoff an RFD.

b4 Unused — undefined.

b5 listener only (lo) — The GPIA will be in LACS when this bit is set, $\overline{\text{ATN}}$ is high and dal (bit 6, R4W) is low. Under these conditions R2R will show a hex A4 reflecting ma, lo, and LACS. The dal bit, when set, will force the GPIA into LIDS even if lo is set. The lo bit is set and reset by the MPU programming bit 5 of R2W high or low, respectively. Once set, a reset of the lo bit is not sufficient to place the GPIA back in LIDS. The device remains in LACS until the dal bit is set. Setting dal with lo low will place the GPIA in LIDS.

b6 (talker only) to — When this bit is set, ATN is high and dat (bit 5, R4W) is low, the GPIA will be in TACS. Under these conditions R2R shows a hex C8 reflecting ma, to, and TACS. The dat bit, when set, forces the GPIA into TIDS for duration of dat being high, even if the to bit is set. The to bit is set and reset by the MPU programming this bit either High or Low, respectively. Once set, a reset of the to bit is not sufficient to place the GPIA back in TIDS. The device remains in TACS until the dat bit is set. Setting dat with to low places the GPIA in TIDS.

b7 deselect (dsel) — When this bit is set, it prevents the DCAS, UUCG, or UACG status bits from being set in R1R and inhibits the status conditions and their respective interrupts on the GET, UACG, UUCG, SDC, and DCL commands. The handshake is also automatically completed. Thus, setting dsel high causes the GPIA to ignore these commands.

3.1.3.6 AUXILIARY COMMAND REGISTER

Auxiliary Command Register (R3)

	7	6	5	4	3	2	1	0
R3R	RESET	DAC	DAV	RFD	msa	rtl	alpha	fget
R3W		rfdr	feoi	dacr			dacd	

b0 (RW) forced group execute trigger (fget) — When this bit is set, the TRIG output pin will be high. When fget is reset, the TRIG output is controlled by the presence or absence of a GET command from the IEEE-488 Standard bus. The fget bit is set and reset by the MPU writing this bit either high or low. Setting fget high does not cause the GET bit to be set.

b1 (R) upper/lower primary address (ulpa) — Once the GPIA has received its primary address, this bit reflects the value of the least significant bit of that address. It is used in the Dual Primary Address mode to detect which of two primary addresses has occurred. This bit is set if the GPIA received primary address from the IEEE-488 Standard bus is of the form "XXXX1" and reset if this address is of the form "XXXX0".

b1 (W) data accept disable (dacd) — When set, this bit inhibits completion of the DAC handshake on addresses or commands. The handshake is released by writing dacd low or dacr high. As long as dacd is set, the handshake is held up on addresses or commands.

b2 (RW) return to local (rtl) — When set, this bit returns the GPIA from the remote state to the local state, if the local state is not locked out (bit 5, R1R set); i.e., if REM (bit 6 of R1R) is set and LOK (bit 5, R1R) is reset, setting rtl resets the REM bit. The rtl bit is set and reset by the MPU writing this bit either high or low.

b3 (RW) my secondary address (msa)— This bit is set to indicate the existence of a valid secondary (extended) address. When in the secondary address mode, setting this bit causes ma (bit 7, R2R) to be set and places the GPIA into either LADS or TADS if the GPIA is initially in the LPAS or TPAS state. The GPIA will then move to either LACS or TACS when ATN is made false. This bit should be set by the MPU during an SCG sequence and is automatically reset at the end of a Secondary Command Group (SCG) multiline message.

b4 (R) Ready For Data (RFD) — This bit monitors the state of the NRFD handshake line. When this bit is set, the handshake line is in the high state. If the RFD bit is a Low, the handshake line is in the low state.

b4 (W) DAC release handshake (dacr) — When set, this bit releases any handshake which was held off because of DAC not being released. The dacr bit is set by the MPU writing this bit high and is reset one E clock cycle after it was set.

b5 (R) Data Available (DAV) — This bit monitors the status of the $\overline{\text{DAV}}$ handshake line. It is set when the $\overline{\text{DAV}}$ line is high and it is low when the $\overline{\text{DAV}}$ line is low. This bit is low when data is valid, and high when data is not valid on the IEEE-488 Standard bus.

b5 (W) forced end or identify (feoi) — The GPIA will assert the $\overline{\text{EOI}}$ management line when the feoi bit is set and the device is in TACS. This bit is set by the MPU writing this bit high, and reset one E clock cycle after it was set. This bit can be used when the GPIA is a talker to indicate to the listener(s) on the IEEE-488 Standard bus, the end of a data string. Once feoi is written high, the END message will be sent with the next byte that the talker sends, unless a reset occurs. This is true even if feoi is written high while the device is not an active talker — the END message will be sent with the first byte the GPIA sends when it becomes a talker.

b6 (R) Data Accepted (DAC) — This bit monitors the state of the DAC handshake line. When this bit is a high, the DAC line is in the high state. When this bit is low, the DAC line is in the low state.

b6 (W) ready for data release (rfdr) — This bit releases the handshake caused by RFD hold-off, e.g., hlda or hlde. The rfdr bit is set by the MPU writing this bit high and reset one E clock cycle after it has been set.

b7 (RW) reset — When this bit is set, it clears the GPIA, except for the Address Register, placing the GPIA in an idle state. The device remains in this state until this bit is reset by the MPU writing it low. During the reset state, the only registers which can be changed are the Address and Address Switch Registers. All current interrupts will also be cleared. A hardware reset sets the reset bit. When removing the software reset condition, the only bit in R3W that can be written into is bit 7; i.e., this same write instruction that resets bit 7 cannot be used to alter any of the other bits in R3W.

3.1.3.7 ADDRESS SWITCH REGISTER

Address Switch Register (R4R)
(Read Only)

7	6	5	4	3	2	1	0
UD3	UD2	UD1	AD5	AD4	AD3	AD2	AD1

The Address Switch Register is not a register inside the GPIA. However, this addressing space is assigned to the GPIA and a read of R4R causes a negative going pulse to occur on the $\overline{ASE}$ output pin. This provides a means for using external switches on an instrument which is used to select the device address. The switches must be buffered onto the MPU data bus with $\overline{ASE}$ connected to the buffer enable pin. Thus, an R4R ready by the MPU places the address switch contents on the MPU data bus. Bits 0-4 are used for addresses and bits 5-7 are user defined. However, care should be used in defining bits 5-7 if these bits are to be placed in the Address Register of the GPIA.

3.1.3.8 ADDRESS REGISTER

Address Register (R4W)
(Write Only)

7	6	5	4	3	2	1	0
lsbe	dal	dat	AD5	AD4	AD3	AD2	AD1

b0-b4 Address — These bits establish the primary address of the GPIA. The GPIA responds to listener/talker commands if the address sent by the controller matches the address in bits 0-4. The MPU must write to this register to establish the correct address. This register is reset by a hardware reset. An all 1's address (hex 1F) cannot be used as this matches an untalk or unlisten command.

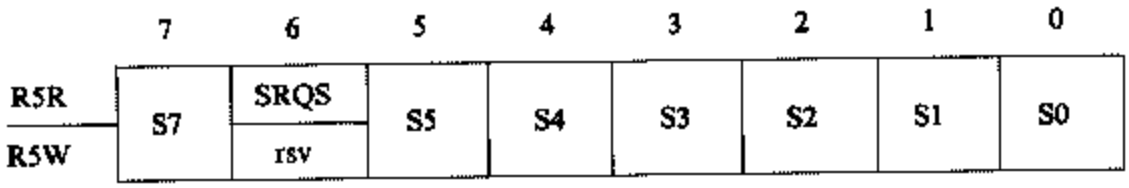
b5 disable the talker (dat) — When set, this bit disables and inhibits the talker function. The data bit is set and reset by the MPU writing it either high or low.

b6 disable the listener (dal) — When set, this bit disables and inhibits the listener function. The dal bit is set and reset by the MPU writing it either high or low.

b7 least significant bit enable (lsbe) — This bit enables the dual primary address mode. The lsbe bit, when set, causes the least significant bit of the primary address to be a don't care, i.e., the GPIA will respond to either of two addresses that differ by the LSB only. The MPU can determine which of the two addresses was actually sent by checking bit 1 of R3R. The lsbe bit is set and reset by the MPU writing it either high or low.

3.1.3.9 SERIAL POLL REGISTER

Serial Poll Register (R5R)
(Read/Write)



b0-b5, b7 (RW) Status Bits — These bits are used to give the device status to the controller during a Serial Poll. The status bits are set by the MPU at the same time the rsv bit is set. They are reset by the MPU writing them low.

b6 (W) service request (rsv) — Setting this bit causes the $\overline{SRQ}$ management line of the GPIA to become low. This bit is set and reset by the MPU writing it either high or low. The device initiates a service request by writing this bit high.

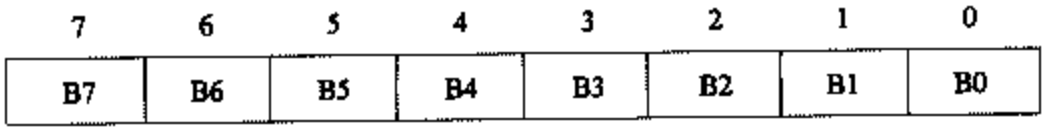
Note

Once written high and a Serial Poll conducted by the controller, the rsv bit must be written low prior to the initiation of a second service request. Unless rsv is reset in between consecutive service requests, the $\overline{SRQ}$ line cannot be made active. This bit, however, cannot be written low while a Serial Poll is being conducted.

b6 (R) Service Request Status State (SRQS) — This bit is set when rsv is written high. It is reset if rsv is written low or when the GPIA is in the Serial Poll Active state (SPAS — bit 2 of R1R is set).

3.1.3.10 COMMAND PASS-THROUGH REGISTER

Command Pass-Through Register (R6R)
(Read Only)



The Command Pass-Through Register reflects the status of the data line of the IEEE-488 Standard bus. This is not a latched register. The contents of the IEEE-488 Standard data bus are placed on the MPU data bus when the MPU reads R6R. Reading this register does not reset any status bits in the GPIA nor does it release the handshake if it had been held off.

3.1.3.11 PARALLEL POLL REGISTER

Parallel Poll Register (R6W)
(Write Only)

7	6	6	4	3	2	1	0
PPR8	PPR7	PPR6	PPR5	PPR4	PPR3	PPR2	PPR1

This register is written to by the MPU and its contents placed on the IEEE-488 Standard data bus during the PPAS (Parallel Poll Active State). A byte of data is placed into this register by the MPU writing to R6W. This does not set any status conditions nor does it flag the controller. The contents of this register are placed on the IEEE-488 Standard bus when the EOI and ATN management lines are both true. This register is reset by writing all bits low or setting the RESET bit (in the Auxiliary Command Register).

3.1.3.12 DATA-IN REGISTER

Data In Register (R7R)
(Read Only)

7	6	5	4	3	2	1	0
DI7	DI6	DI5	DI4	DI3	DI2	IDI1	DI0

This register contains the received data byte when the GPIA is an active listener. The BI status bit indicates to the MPU the presence of a data byte in the register. When in the listener mode and receiving data, the DAC handshake is held off until the MPU reads R7R. The DAC handshake line (any time it is held off) is released after the high-to-low transition of the same E pulse used to read data from the Data In Register.

3.1.3.13 DATA-OUT REGISTER

Data Out Register (R7W)
(Write Only)

7	6	5	4	3	2	1	0
DO7	DO6	DO5	DO4	DO3	DO2	DO1	DO0

Data is transferred from the MPU to the IEEE-488 Standard data bus through the Data-Out Register. When the data byte in R7W has been accepted by all the listeners, the BO bit in the Interrupt Status Register is set indicating to the MPU that another data byte can be written to the Data-Out Register. Writing to this register resets BO until all listeners have again accepted the data byte.

3.2 INTERFACE FUNCTION — GPIA RECOGNITION AND OPERATION

The GPIA implements the interface function state diagrams of the IEEE-488 Standard in hardware logic. The GPIA moves in and out of the states in these diagrams as a function of the conditions on either the GPIB or MPU data bus. The state of the MC68488 can be monitored by examining the various GPIA status registers. This section correlates the state diagrams to the contents of these status registers. For simplicity, the conditions for transition between states have been omitted from the state diagrams shown in this section. Only the relationship between the states are indicated. For more details the complete interface state diagrams as implemented by the GPIA can be found in Chapter 2. Also, the movement from state to state in this section assumes that the GPIA has previously been initialized and programmed properly (see Chapter 4 for initialization).

3.2.1 Listener State

The GPIA uses the Listener Function in conjunction with the Acceptor function to receive data from the GPIB. There are three states in the Listener State diagram (Figure 3-5). The GPIB indicates whether it is in LIDS or LACS by setting the appropriate bits in the Address Status Register (R2R). The LISTENER Extended Function defines two additional states (LPIS and LPAS) in a separate state diagram. There are four listener modes that allow movement into and out of the Listener states: 1) Listener Mode (Primary Address Mode), 2) Dual Primary Address Mode (Special Feature of GPIA), 3) Listener Extended Mode (requires the secondary addressing feature of the GPIA), and 4) listen only (programmed by the MPU).

Figure 3-5. Listen State Diagram

3.2.1.1 LISTENER MODE (Primary addressing – apte = 0). Once initialized, the GPIA is ready to become a listener. The controller sends MLA (My Listen Address) which is:

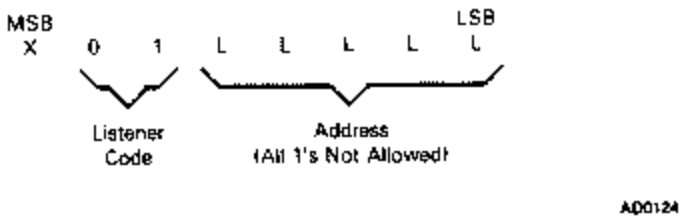


Figure 3-6. MLA From Controller

where the L's correspond to the address placed in the Address Register. An all 1's address is not allowed because this code (X0111111) represents the unlisten command. Using the acceptor handshake, the GPIA moves to LADS (Figure 3-7). When ATN goes inactive (high), the GPIA will move to LACS.

Bit	7	6	5	4	3	2	1	0
R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	1	0	0	0	0

ATN = True
(a) Listener Addressed State (hex 90)

Bit	7	6	5	4	3	2	1	0
R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	0	0	1	0	0

ATN = False
(b) Listener Active State (hex 84)

AD0125

Figure 3-7. Address Status Register (apte = 0)

The device is now ready to be a listener. A flag, BI (Byte Input), in the Interrupt Status Register (R0R) is set high when a data byte is sent by the addressed talker. The MPU obtains this data byte by reading the Data-In-Register (R7R).

Bit	7	6	5	4	3	2	1	0
R0R	INT	BO	GET		APT	CMD	END	BI
Read	0	0	0	1	0	0	0	1

No Interrupt Masks Set.

AD0126

Figure 3-8. Interrupt Status Register For Listener

BI is reset low when the MPU reads the Data-In Register. When the talking device sends EOI, the END flag in the Interrupt Status Register is set high to indicate this condition. END resets to zero when the $\overline{EOI}$ line goes high or $\overline{ATN}$ goes low. An option is provided to holdoff the RFD (Ready for Data) handshake when the END flag is set high allowing the listener to hold up the IEEE-488 Standard bus in order to accomplish any necessary processing of the data that was just sent. This holdoff on END option is selected by setting the hlde bit in the address mode register (R2W).

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo		hlde	hlda		apte
Write	0	0	0	X	1	0	X	0

AD0127

Figure 3-9. Hold On End Option

The hlda (hold on all) bit in the R2W, when set, holds off the RFD handshake on all data. The handshake is released by writing rfdr (ready for data release) high in the Auxiliary Command Register (R3R).

Bit	7	6	5	4	3	2	1	0
R3R	Reset	DAC	DAV	RFD	rmsa	rtl	ulpa	fget
R3W		rfdr	feoi	dacr			dacd	
Write	0	1	0	0	0	0	0	0

AD0128

Figure 3-10. Release of RFD Hold Off (Caused by hide or hlda)

3.2.1.2 DUAL PRIMARY ADDRESS MODE (apte = 0). The dual primary address mode is selected by writing lsbe (least significant bit enable) high in R4W. This bit can be set with the same write operation that places the device address in R4W.

Bit	7	6	5	4	3	2	1	0
R4W	lsbe	dal	dat	A4	A3	A2	A1	A0
Write	1	0	0	A4	A3	A2	A1	A0

Write Hex 80

AD0129

Figure 3-11. Dual Primary Address Enable

The Dual Primary Addressing mode is similar to the previously discussed listener mode (Primary Address Mode) except that, as its name implies, the device responds to two adjacent addresses; i.e., the least significant bit of the address is a don't care. A software routine must be used to determine which of the two possible addresses was actually sent over the GPIB. For example, if the address in the Address Register of the GPIA is 00010, then the MC68488 responds by going into LACS if the controller sends either of the MLAs shown in Figure 3-12.

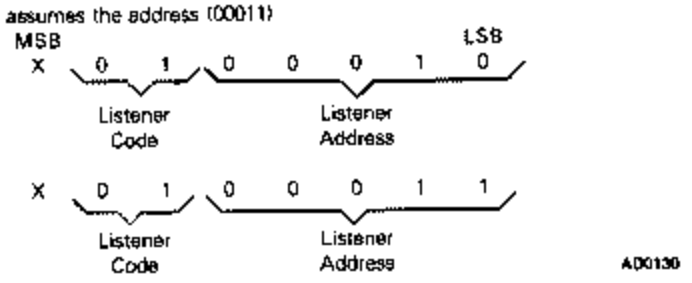


Figure 3-12. Acceptor MLAs from Controller in Dual Address Mode (R4W) Address = 00010

The state of the least significant bit of the address sent by the controller can be read in the Auxiliary Command Register (Figure 3-13).

Bit	7	6	5	4	3	2	1	0
R3R	Reset	DAC	DAV	RFD	msa	rtl	ulpa*	fget
R3W		rfdr	feoi	decr			dcd	

*ulpa is a 1 for binary data and a 0 for ASCII data

AD0131

Figure 3-13. Dual Primary Addressing (LSB Recognition)

A possible application for the dual primary address mode is to provide an easy way for a device to distinguish two types of data being sent on the IEEE-488 Standard data lines. For example, when in the Listen Active State, a read of the ulpa bit might alert the MPU that either ASCII data (ulpa = 0) or binary data (ulpa = 1) is forthcoming. The ulpa bit could be used to make a software decision concerning the type of data handling routine to be used.

NOTE

This mode can also be used in conjunction with the listener extended mode.

3.2.1.3 LISTENER EXTENDED MODE (apte = 1). Refer to the state diagram in Figure 3-5 for the relation between LIDS, LADS, and LACS. However, some of the transitions between these states are different in the extended mode from those of the nonextended mode; see Chapter 2 for details. Two additional states (LPIS and LPAS) in a separate state diagram are defined for the Listener Extended mode (Figure 3-14).

AD0132

Figure 3-14. LPAS State Diagram

To be in the listener extended mode, the apte bit (bit 0, R2W) must be set. This allows the GPIA to respond to secondary commands. If apte = 1, the GPIA moves to the Listener Primary Addressed State when it receives its primary address. This will cause LPAS (Listener Primary Address State) in R2R to be set. Note that ma (my address) is not set at this time. That is because full address has not been received.

The primary address sequence is implemented as described in Section 3.2.1.1. When the controller sends MLA (My Listen Address) and makes ATN (Attention) false, the Address Status Register (R2R) contents is as shown in Figure 3-18b.

Following the primary address, the controller sends the SCG (Second Command Group) of the form shown in Figure 3-15. If the controller does not send this, but rather it sends another PCG (Primary Command Group), the GPIA goes to the listener idle state (LIDS) and MLA must be sent again to establish secondary addressing. When the SCG is received, the DAC handshake signal is held-off (not allowing the handshake to complete) until the MPU has had a chance to examine the command.

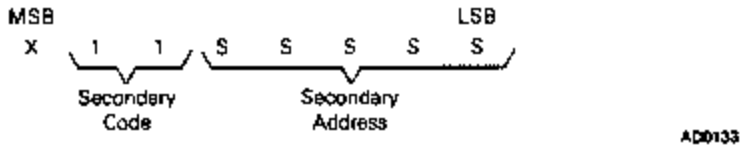


Figure 3-15. Secondary Address Command

The APT bit is set high in the Interrupt Status register (R0R, Figure 3-16) indicating the Secondary Address is available for examination in the Command Pass-Through Register (R6R). The Command Pass-Through Register contains the secondary address (Figure 3-17).

Bit	7	6	5	4	3	2	1	0
R0R	IRQ	BO	GET		APT	CMD	END	BI
Read	0	0	0	1	1	0	0	0

No Interrupt Mask Bits Set

AD0134

Figure 3-16. Interrupt Status For APT

Bit	7	6	5	4	3	2	1	0
R6R	X	1	1	S	S	S	S	S

AD0135

Figure 3-17. Contents of Command Pass Through Register

Once the secondary address is read into the MPU and a compare is made to determine if the secondary address is valid, the following procedure should be implemented (Figure 3-18).

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo	X	X	X	X	1
Write	X	0	0	X	X	X	X	1

(a) Address Mode Register (apte = 1 selects secondary addressing)

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	0	0	0	0	0	0	X*	0

Primary Address sent and ATN = false

(b) Address Status Register when in LPAS (hex 02)

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	0	0	1	X*	0

MPU writes msa (R3W) high along with dacr if secondary command is valid.

(c) Address Status Register when in LACS (hex 06)

*LPAS will be a "1" after reception of the primary command group but will be reset to a "0" when another primary command is sent over the GPIB.

AD0136

Figure 3-18. Listener Extended Mode

1. If the SCG is valid, the MPU should write a hex 18 into R3W releasing the handshake and allowing the GPIA to go to the Listener Active State when ATN is released. Once the secondary command sequence is initiated, the LPAS status bit remains set until another PCG is sent over the bus. See Section 4.2.4.1 for details on LPAS.
2. If the SCG is invalid, the MPU should write a hex 10 into R3W releasing the handshake. The GPIA does not go to the Listener Active State when ATN is released. LPAS will be a "1" after the reception of the primary command group but is reset to a "0" when another primary command is sent over the GPIB.

Writing the dacr bit high releases the DAC handshake, and writing msa high places the MC68488 into the Listener Active State when ATN is released (Figure 3-19).

Bit	7	6	5	4	3	2	1	0
R3F	Reset	DAC	DAV	RFD	msa	rtl	ulpa	fget
R3W		rfdr	feoi	dacr			dacd	
Write If Valid	0	0	0	1	1	0	0	0
Write If Invalid	0	0	0	1	0	0	0	0

AD0137

Figure 3-19. MPU Response to Secondary Address

3.2.1.4 LISTEN ONLY MODE (lo). The GPIA can become an active listener by programming it to listen only (lo). Setting bit 5 of R2W high (with dal low, R4W) places the GPIA in the Listener Active State through the listen only mode (Figure 3-20) if ATN is not asserted. The talk only (to) bit must not be set high when lo is high.

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo	X	hlde	hlda	X	apte
Write	X	0	1	X	X	X	X	X

Address Mode Register

(a) Selecting listen only mode

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	1	0	0	1	0	0

Address Status Register

(b) Listener Active State (Listen Only — Hex A4)

AD0138

Figure 3-20. Listener Only Mode

The lo bit is reset by writing it low via the MPU. The GPIA remains in LACS after the lo bit is reset. To place the device in LIDS, the lo bit is reset followed by setting the dal bit (Figure 3-21). The dal bit need only be toggled high to place the GPIA in LIDS. Once dal is set placing the device in LIDS, it can be immediately reset and the GPIA will remain in LIDS.

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo	X	hlde	hlde	X	apte
Write	X	0	1	X	X	X	X	X

Address Mode Register

(a) device in listen only mode

R2W	dsel	to	lo	X	hlde	hlda	X	apte
Write	X	0	0	X	X	X	X	X

Address Mode Register

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	0	0	1	0	0

Address Status Register
GPIA remains in LACS after lo reset.

(b) Reset lo bit.

R4W	lsbe	del	dat	AD5	AD4	AD3	AD2	AD1
Write	X	1	0	A	A	A	A	A

Address Register
Set del bit

R4W	lsbe	del	dat	AD5	AD4	AD3	AD2	AD1
Write	X	0	0	A	A	A	A	A

Address Register
Reset del bit

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	0	0	0	0	0	0	0	0

Address Status Register
device now in LIDS

(c) Set/Reset dal bit places device in Listener Idle State.

AD0139

Figure 3-21. Removing GPIA From Listen Only Mode

3.2.2 Talker State Diagrams

The GPIA uses the TALKER functions in conjunction with the source function to send data on the GPIB.

There are four states in the Talker State Diagram (Figure 3-22). The GPIA indicates whether it is in TIDS or TACS by setting the appropriate bits in the address Status Register (R2R). The Serial Poll Active State (SPAS) is recognized in the Command Status Register. This function is described in the Service Request State/Serial Poll Section. The Talker Extended function defines two additional states (TPIS and TPAS). As with the Listener function, there are four talker address modes that

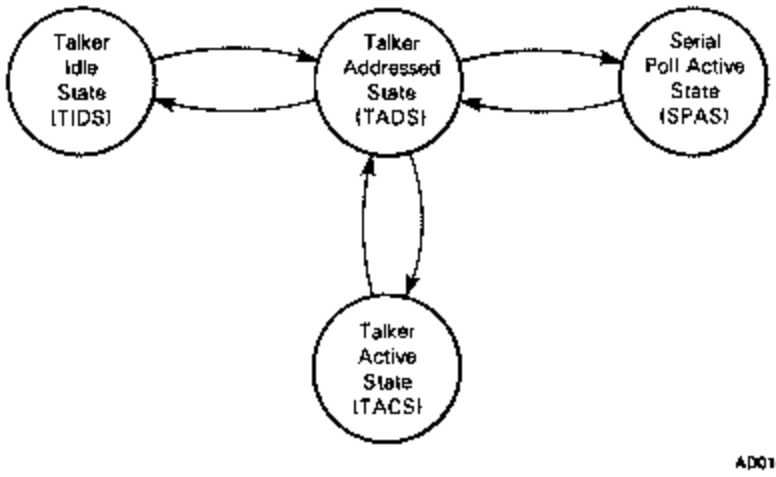


Figure 3-22. Talker State Diagram

allow movement into and out of the talker state: 1) Talker Mode (Primary Address Mode), 2) Dual Primary Talker Mode (special feature of the GPIA), 3) Talker Extended Mode (requires the secondary addressing feature of the GPIA) and 4) talk only (programmed by MPU). The operation of the GPIA in the various talker address modes is analogous to that of the listener address modes discussed in the previous sections.

3.2.2.1 TALKER MODE (Primary Addressing – apte = 0). Once initialized, the GPIA is ready to become a talker on the GPIA. The controller sends MTA (My Talker Address) which is:

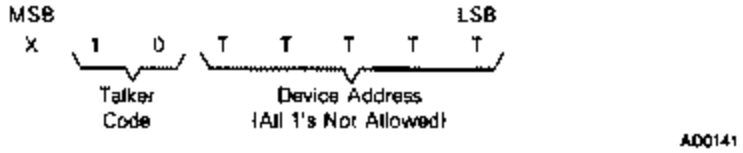


Figure 3-23. MTA Sent by Controller

where the T's correspond to the address placed in the Address Register. An all 1's address is not allowed because this code (X1011111) represents the Untalk command. The GPIA responds as shown in Figure 3-24.

Bit	7	6	5	4	3	2	1	0
R3R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	1	0	0	0	0

ATN = True

(a) Talker Address State (Hex 90)

R3R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	0	1	0	0	0

ATN = False

(b) Talker Active State (Hex 88)

AD0142

Figure 3-24. Address Status Register After MTA Placed on GPIB (apte = 0)

After entering TACS, the device is ready to talk. The BO (Byte Output) status bit in the Interrupt Status Register (R0R) is set high (Figure 3-25) indicating that the GPIA is in TACS and the Data-Out Register (R7W) is empty.

Bit	7	6	5	4	3	2	1	0
R0R	INT	BO	GET	1	APT	CMD	END	BI
Read	0	1	0	1	0	0	0	0

The BO interrupt mask bit (R0W) is not set.

AD0143

Figure 3-25. Interrupt Status Register for BO

Writing to the Data-Out Register (R7W) places the byte on the IEEE-488 Standard data lines DIO1-DIO8. Writing the byte to the Data-Out Register resets BO status (R0R). BO is again set high when the GPIA receives a DAC back for the listener(s). $\overline{EOI}$ (End or Identify) being made true by the talker indicates the end of a multiple byte transfer. This is accomplished by writing feoi (forced end or identify) high in the Auxiliary Command Register (Bit 5, R3W) prior to sending the last byte (Figure 3-26). This will drive the $\overline{EOI}$ management line low. The $\overline{EOI}$ line is made false (high) when the listener(s) accept the data. The feoi bit is automatically reset one E clock cycle after it was set.

NOTE

Once feoi has been written high, the END message is sent with the next byte the talker sends unless a hardware (pin 19) or software (bit 7, R3W) reset occurs. This is true even if feoi is written high while the device is not an active talker — the END message is sent with the first byte the GPIA sends when it becomes a talker.

Bit	7	6	5	4	3	2	1	0
R3R	Reset	DAC	DAV	RFD	msa	rtl	ulpa	fget
R3W		rfdr	feoi	dacr			dacd	
Write	0	0	1	0	0	0	0	0

AD0144

Figure 3-26. EOI Activation By Setting feoi

3.2.2.2 DUAL PRIMARY ADDRESS MODE (apte = 0). The dual primary address mode is selected by writing lsbe (least significant bit enable) high in R4W. As previously mentioned for the listener mode, setting lsbe high makes the least significant bit of the device address a don't care; i.e., the device responds to either of two adjacent addresses (see Figure 3-27). A software routine must be used to determine which of the two possible addresses was actually sent over the GPIB. The software routine can check the LSB of the address by monitoring bit 1 of R3W (see Figure 3-13).

NOTE

This mode can also be used in conjunction with the talker extended mode.

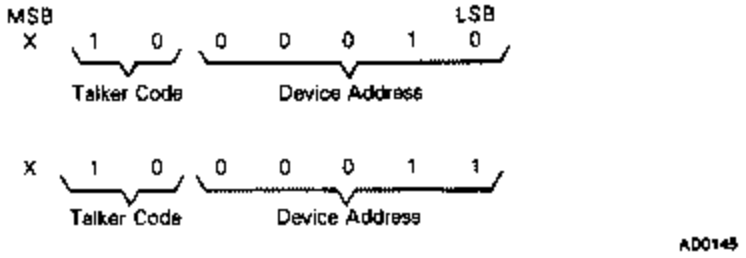


Figure 3-27. Accepted MTA(s) from Controller In Dual Address Mode (R4W Address = 00010)

3.2.2.3 TALKER EXTENDED MODE (apte = 1). Refer to the state diagram in Figure 3-22 for the relationship between TIDS, TADS, and TACS. However, some of the transitions between these states are different in the extended mode from those of the nonextended mode; see Chapter 2 for details. Another state diagram (Figure 3-28) defines two additional states (TIPS and TPAS) for the Talker Extended mode.

Figure 3-28. TPAS State Diagram (Talker Extended)

The Talker Extended mode requires that the MPU program the apte bit (bit 0, R2W) high. This allows the GPIA to respond to secondary commands. In the secondary address mode, the controller first sends a primary address and then a secondary address, giving two bytes of address capability. Both bytes are limited to the five least significant bits for addressing, giving 961 possible addresses. The first byte received is the primary address and the primary address sequence is described in Section 3.2.2.1. The addressed state of the GPIA can be obtained by reading R2R (Figure 3-29).

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo	X	hide	hlda	X	apte
Write	X	0	0	X	X	X	X	1

(a) Address Mode Register (apte = 1 selects secondary addressing)

R2R	rna	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	0	0	0	0	0	0	0	X*

(b) Address Status Register when in TPAS (Hex 01)

R2R	rna	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	0	1	0	0	X*

TACS occurs after MPU writes rnsa indicating valid secondary address has been received.

(c) Address Status Register when in TACS (Hex 90)

*TPAS will be a "1" after the reception of the primary command group, but will be reset to a "0" when another primary command (e.g., another device's primary address) is sent over the GPIB.

AD0147

Figure 3-29. Talker Extended Mode

Following the primary address, the controller sends the SCG (Secondary Command Group) of the form shown in Figure 3-15. If the controller does not send this but rather it sends another RPCG (Primary Command Group), the GPIA goes to the Talker Idle State (TIDS) and the primary talk address has to be sent again by the controller. When the SCG is received, the DAC handshake signal is held off (not allowing the handshake to complete) until the MPU has had a chance to examine the command. The APT bit is set high in the Interrupt Status Register (see Figure 3-16) indicating a secondary command is available for examination in the Command Pass-Through Register (R6R). Once the Secondary Command is read into the MPU and a compare is made to determine if the Secondary Command is valid, the following procedure should be implemented (see Figure 3-19).

1. If the SCG is valid, the MPU should write a hex 18 into R3W. This releases the handshake and allows the GPIA to go to the Talker Active State when ATN is released.
2. If the SCG is invalid, the MPU should write a hex 10 into R3W, releasing the handshake but keeping the GPIA in the Talker Primary Address State.

Once the secondary command sequence is initiated, the LPAS status bit remains set until a PCG is sent over the bus. See Section 4.2.4.1 for details on LPAS.

3.2.2.4 TALKER ONLY (to). The GPIA can become an active talker by programming it to talk only (to). Setting bit 6 of R2W high (with dat low, R4W) will place the GPIA in the Talker Active State through the listen only mode (Figure 3-30) if ATN is high. The listen only bit (lo) must not be set high when talk only is high.

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo	X	hide	hida	X	apte
Write	X	1	0	X	X	X	X	X

Address Mode Register

(a) Selecting listen only mode

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	1	0	0	1	0	0	0

Address Status Register

(b) Talker Active State (talk only — Hex C8)

AD0148

Figure 3-30. Talker Only Mode

The to bit is reset by writing this bit low via the MPU. The GPIA remains in TCAS after the to bit is reset. To place the device in TIDS, the to bit is reset followed by setting the data bit (Figure 3-31). The data bit only need be toggled high to place the GPIA in TIDS; i.e., once data is set placing the device in TIDS, it can be immediately reset and the GPIA remains in TIDS.

3.2.3 Service Request State/Serial Poll

The GPIA uses the Service Request function to request service from the controller. The state diagram for service request (Figure 3-32) has three states; NPRS (Negative Poll Response State), SRQS (Service Request State), and APRS (Affirmative Poll Response State). When the GPIA issues

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo	X	hide	hida	X	apte
Write	X	1	0	X	X	X	X	X

Address Mode Register

(a) device in talk only mode

R2W	dsel	to	lo	X	hide	hida	X	apte
Write	X	0	0	X	X	X	X	X

Address Mode Register

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	1	0	0	0	1	0	0	0

Address Status Register
GPIA remains in TACS after to reset.

(b) Reset to bit

R4W	lsbe	dal	det	AD5	AD4	AD3	AD2	AD1
Write	X	0	1	A	A	A	A	A

Address Register
Reset dal bit

R2R	ma	to	lo	ATN	TACS	LACS	LPAS	TPAS
Read	0	0	0	0	0	0	0	0

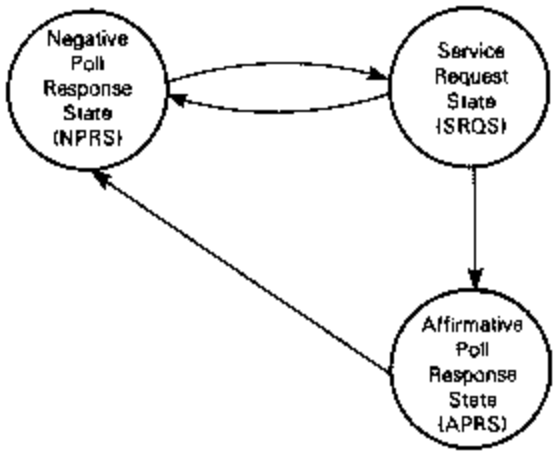
Address Status Register
Setting of dat will place device in TIDS

(c) set dat

AD0149

Figure 3-31. Removing GPIA From the Talk Only Mode

a service request, it is requesting that the controller conduct a Serial Poll to obtain the service request information placed in the GPIA Serial Poll Register by the MPU. Since a Serial Poll can only be conducted on a GPIA when it is in the talker mode, the Serial Poll Active State (SPAS) is part of the Talker State Diagram (Figure 3-22). For state diagram details see Chapter 2.



AD0150

Figure 3-32. Service Request State Diagram

The device issues a service request to the controller by enabling the $\overline{SRQ}$ management line (bringing this line low). To request service, a "1" is written into the rsv bit (bit 6) of the Serial Poll Register (R5W — Figure 3-33). This action enables the $\overline{SRQ}$ line indicating to the controller that one of the devices in the system is requesting service. The appropriate status bits (indicating what service is requested) should be written into R5W at the same time rsv is set. Seven bits are allowed for coding the status.

Bit	7	6	5	4	3	2	1	0
R5R	S7	SRQS	S5	S4	S3	S2	S1	S0
R5W		rsv						
Write	S7	1	S5	S4	S3	S2	S1	S0

AD0151

Figure 3-33. Request for Service

The controller must then poll each device to determine which device is requesting service. The poll is initiated by the controller sending the remote message SPE (Serial Poll Enable — Figure 3-34) which places all devices in the Serial Poll Mode State (SPMS).

MSB							LSB
X	0	0	1	1	0	0	0
Controller Sends							

AD0152

Figure 3-34. Serial Poll Enable

In order for a device to be Serial Poll Active it must be a talker. The controller then sequentially addresses each device to be a talker. As each GPIA is addressed to talk, it responds by going to the Serial Poll Active State (SPAS). The contents of the Serial Poll Registers are placed on the GPIB. If bit 6 (DIO7) is high (DIO7 will be low on the GPIB since it is a negative logic bus, but will be a "1" on the MPU side of the GPIA) the controller knows this is the device requesting service. If more than one device is requesting service the first device polled has highest priority. When the GPIA is in SPAS, bit 2 of the command status register is set (Figure 3-35).

Bit	7	6	5	4	3	2	1	0
R1R	UACG	REM	LOK	X	RLC	SPAS	DCAS	UUCG
Read	0	0	0	1	0	1	0	0

AD0153

Figure 3-35. Serial Poll Active State

Bit 2 being set causes the CMD status bit (R0R) to be set, which will in turn issue an interrupt if the appropriate mask bits are set in the Interrupt Mask Register. The CMD bit and SPAS bit remains set as long as the GPIA is in SPAS. When $\overline{ATN}$ is asserted the GPIA leaves SPAS. It returns to SPAS if $\overline{ATN}$ becomes false unless SPD (Serial Poll Disable) or OTA (Other Talk Address) has been sent. After sending SPD, the GPIA is in TACS and the SPAS and CMD status bits are reset. The SRQS bit in R5R is also reset. The rsv bit, however, remains set. Once the rsv bit is set and a Serial Poll conducted, this bit must be reset by the MPU before it can be set again for a second service request.

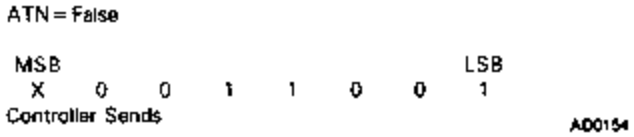


Figure 3-36. Serial Poll Disable

3.2.4 Parallel Poll State

Parallel Poll is a means by which the controller may request one bit of status from each device. The GPIA uses the Parallel Poll State Diagram (Figure 3-37) to place the contents of the Parallel Poll Register (R6W) on the GPIB. The states implemented by the GPIA are PPIS (Parallel Poll Idle State), PPSS (Parallel Poll Standby State), and PPAS (Parallel Poll Active State). The PUCS and PACS states defined in the IEEE-488 Standard as a separate state diagram are not implemented by the GPIA. This omits the capability to be directly configured by the controller. The configuration must occur from the controller via the GPIA MPU. The procedure to configure for Parallel Poll is given in Chapter 4.

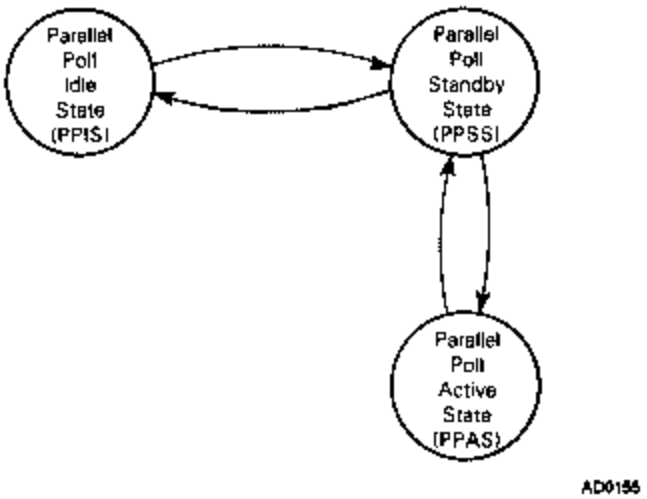


Figure 3-37. Parallel Poll State Diagram

Parallel Poll is initiated with the controller requesting that all devices place their status bit on the bus simultaneously (Figure 3-38).

Bit	7	6	5	4	3	2	1	0
R6W	PPR8	PPR7	PPR6	PPR5	PPR4	PPR3	PPR2	PPR1
Write	1	0	0	0	0	0	0	0

Bit 7 assigned to this device.

AD0156

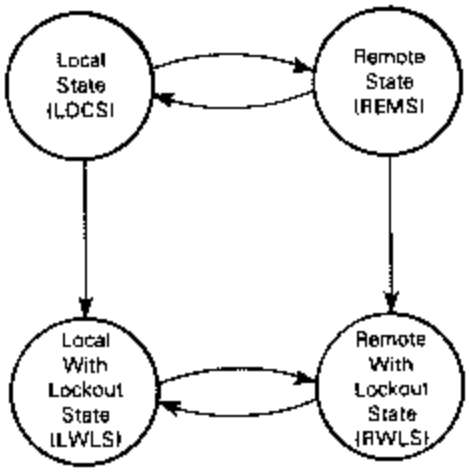
Figure 3-38. Parallel Poll Register

Each device is assigned one bit to be used for its Parallel Poll. If more than eight devices are connected to the GPIB, then some devices must share one of the bits. To initiate a Parallel Poll, the controller asserts both the EOI and ATN management lines simultaneously. This procedure causes all the GPIAs connected to the bus to place their Parallel Poll Register content on the GPIB.

Unlike Serial Poll, which is device initiated, Parallel Poll is controller initiated. Also, whereas Serial Poll provides seven bits of status information to a device, Parallel Poll provides only one. However, the Parallel Polling procedure is much less complicated and is more quickly executed than that of Serial Polling. In many cases, the Parallel and Serial Poll functions are used together to obtain the most efficient status information transfer (see Chapter 4).

3.2.5 Remote/Local State

The Remote/Local modes provide a protocol for allowing the device to be configured from either its front panel controls or from the controller. Thus, such instrument parameters as voltage ranges, frequency setting amplitude settings, etc., may be set either by switches on the front panel of the device or set internally by commands sent over the interface. An additional feature of the Remote/Local Mode is a Local Lockout. It not only prohibits the front panel controls from controlling the device, but, also prevents the device from being placed in the Local Mode from the front panel. The Remote/Local State diagram (Figure 3-39) has four states; LOCS (Local state), REMS (Remote State), LWLS (Local Without Lockout State), and RWLS (Remote With Lockout State).



AD0157

Figure 3-39. Remote/Local State Diagram

The Remote/Local Mode uses the REM (Remote), LOK (Lockout), and RLC (Remote/Local Change) bits in the Command Status Register (R1R) and the rtl (return to local) bit in the Auxiliary Command Register (R3W). The REM and LOK status bits are used to uniquely indicate the four states of the Remote/Local State diagram (Table 3-4).

Table 3-4. Remote/Local State Indication

REM	LOK	State
0	0	LOCS: no lockout
0	1	LWLS: rtl disabled
1	0	REMS: no lockout
1	1	RWLS: rtl disabled

REM = bit 6, R1R
LOK = bit 5, R1R

The RLC bit (bit 3, R1R) of the Command Status Register indicates, when set, that the GPIA has changed states in the Remote/Local state diagram; i.e., anytime the REM bit changes from a 1-to-0 or 0-to-1, the RLC bit is set. This bit and not LOK or REM causes the CMD interrupt bit to be set. The RLC bit is reset by reading the Command Status register.

The GPIA enters REMS (Remote State) when the $\overline{\text{REN}}$ uniline message is true and MLA is received (Figure 3-40). Note that $\overline{\text{REN}}$ must be sent first, followed by MLA before the GPIA enters REMS.

Bit	7	6	5	4	3	2	1	0
R1R	UACG	REM	LOK	—	RLC	SPAS	DCAS	UUCG
Read	0	1	0	1	1	0	0	0

RLC will be set on REM transition and reset on read of R1R.

AD0158

Figure 3-40. REMS Indication

The REM bit remains high until the $\overline{\text{REN}}$ uniline message goes false (high) or the rtl bit in the Auxiliary Command Register is written high (if the GPIA is in Local Lockout, the rtl will be disabled). The GTL (Go-To-Local) command from the controller also resets the REM bit.

The GPIA will go from REMS to RWLS (Remote with Lockout State) if the REN uniline message is true and the LLO (Local Lockout) universal command is sent by the controller (Figure 3-41).

Bit	7	6	5	4	3	2	1	0
R1R	UACG	REM	LOK	—	RLC	SPAS	DCAS	UUCG
Read	0	1	1	1	0	0	0	0

RLC will be set on REM transition and reset on read of R1R, but RLC is not set on a LOK transition.

AD0159

Figure 3-41. RWLS Indication

The LWLS (Local With Lockout State) is activated when the LLO command is sent from the controller with the $\overline{REN}$ uniline message true and MLA not active (Figure 3-42). LWLS can also be entered from RWLS via the GTL command.

Bit	7	6	5	4	3	2	1	0
R1R	UAC	REM	LOK	–	RLC	SPAS	DCAS	UUCG
Read	0	1	0	1	0	0	0	0

RLC is not set on a LOK transition.

AD0180

Figure 3-42. LWLS Indication

3.2.6 Device Clear State

The Device Clear Mode allows a device to be cleared, setting it to some initial state. This capability could be used to place an instrument in its power on (pon) state. The Device Clear State Diagram has two states: Device Clear IDLE State (DCIS) and Device Clear Active State (DCAS) as shown in Figure 3-43.

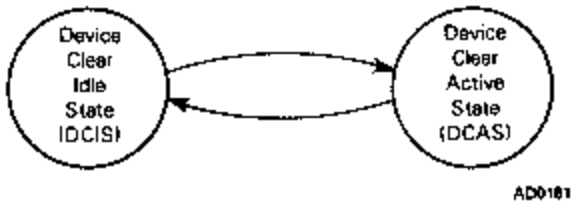


Figure 3-43. Device Clear State Diagram

The DCAS is activated by the controller sending either the Device Clear (DCL command) or the Selected Device Clear (SDC) command. The GPIA enters DCAS if:

1. The Device Clear Command (X0010100) is sent.
2. The GPIA is in LADS (Listener Address State) and the Selected Device Clear (X0000100) is sent, i.e., if the device has first been programmed to be a listener.

When the GPIA enters DCAS it responds by setting the DCAS bit in the Command Status Register (Figure 3-44).

The DCAS bit remains set as long as the device is in DCAS. This condition causes the CMD interrupt bit to be set if the dsel bit (R2W) is not set. The DAC handshake is also held off allowing the MPU time to respond to the Device Clear before releasing the GPIB. The GPIB moves from DCAS to DCIS if (a) the controller removes the Device Clear Command in condition 1 above; or (b) for condition 2 if either the GPIA leaves LADS or the controller removes the selected Device Clear message.

Bit	7	6	5	4	3	2	1	0
R1R	UACG	REM	LOK	–	RLC	SPAS	DCAS	UUCG
Read	0	0	0	1	0	0	1	0

AD0162

Figure 3-44. DCAS Indication

3.2.7 Device Trigger State

The Device Trigger Mode allows one or more devices to be triggered from the same signal (command). This could be used to instruct instruments to begin making measurements at the same time. The Device Trigger State Diagram consists of two states, Device Trigger Idle State (DTIS) and Device Trigger Active State (DTAS) as shown in Figure 3-45.

AD0163

Figure 3-45. Device Trigger State Diagram

The DTAS is activated by the controllor sending the Group Execute Trigger (GET) command (X0001000). The GPIA must be in LADS (Listener Address State) and the dsel bit (R2W) must be low in order to respond to GET. The device responds by setting the GET bit (bit 5, R0R, see Figure 3-46). The DAC handshake is also held off. If dsel is set, the GPIA completely ignores the GET command; i.e., the GET status bit is not set, the TRIG pin remains low and the handshake is automatically completed.

Bit	7	6	5	4	3	2	1	0
R0R	INT	BO	GET	–	APT	CMD	EMD	BI
Read	0	0	1	1	0	0	0	0

dsel=0
GET interrupt mask bit not set.

AD0164

Figure 3-46. DTAS Indication

When the GET bit is set, the TRIG output pin (pin 24 on the MC68588) goes high and remains high while the GET bit is high or until the dacr (DAC release) bit in R3W is set. (The dacr bit returns low one MPU clock cycle after it is set true.)

The TRIG output pin can also be set high by the MPU programming the fget (force group execute trigger) bit high in the Auxiliary Command Register (R3W). See Figure 3-47. The TRIG pin remains high until the MPU writes fget low. Programming the fget bit high will not set the GET bit.

Bit	7	6	5	4	3	2	1	0
R3R	Reset	DAC	DAV	RFD	mse	rtl	ulpa	fget
R3W		rtldr	feoi	dacr			dscd	
Write	0	X	X	X	X	X	X	1

TRIG pin will remain high until fget is programmed low by MPU.

AD0165

Figure 3-47. fget Local Command

3.3 SPECIAL FEATURES

The previous section describes the features of the GPIA that relate to the implementation of the IEEE-488 Standard state diagrams. This section describes the special purpose features that are not described by the IEEE-488 Standard. These features allow an MPU an easy straightforward way of monitoring and interfacing with the GPIB.

3.3.1 Handshake Holdoff

Some commands are not handled by the GPIA. In many cases it is necessary for the MPU to receive the command (e.g., Undefined commands or Secondary commands) and halt the GPIB temporarily while it interprets the command (what does it mean, is it meant for this device, etc.). In other cases some commands (e.g., Device Clear) require that the MPU perform a task. If the handshake is held up, not only does the device have time to complete the task, but the controller can monitor the handshake. When released, the controller knows that all required actions have been completed. In still other cases, some devices, when receiving data, may want to work with the data on a per byte basis (e.g., perform some computation on each byte as it is received). These devices can hold up the handshake not allowing the talker to send the next byte until the listener is ready. This lists but a few of the reasons for temporarily halting the GPIB. Since the GPIB is an asynchronous bus, the most convenient way of halting the flow of data is to hold up the handshake signals. There are two ways the GPIA can hold up the handshake: 1) Not allowing the DAC handshake line to return high, indicating to the talker that not all listeners have accepted the current data byte on the GPIB, 2) Not allowing the RFD handshake line to return high, indicating to the talker that not all addressed listeners are ready for data.

3.3.1.1 DAC HOLDOFF. The DAC holdoff feature is used to hold up the handshake on the reception of commands from the controller. Certain commands such as GET, DCL, UUCG, UACG, to name a few, causes an automatic DAC holdoff. As mentioned earlier, this gives the MPU time to take any needed action. For example: in the secondary address mode a controller might be using a secondary command to set the range on a digital voltmeter. The reception of a Secondary Command Group (SCG) automatically holds off the DAC handshake. The MPU can then receive the command, compare it to a lookup table in software and determine which range is being requested. After taking the necessary action, the MPU can release the handshake by writing bit 4 of the Auxiliary Command Register (R3W) high (Figure 3-48).

Bit	7	6	5	4	3	2	1	0
R3R	Chip	DAC	DAV	RFD	msa	rtl	ulpa	fget
R3W	reset	rtldr	leoi	dacr			dacd	
Write	0	0	0	1	0	0	0	0

dacr reset one $\phi 2$ clock cycle after set.

AD0168

Figure 3-48. DAC Handshake Release (hex 10)

In addition to the automatic DAC holdoff on certain commands mentioned in the previous paragraph, the GPIA provides a programmable control bit that allows holding off the DAC on all remote multiline commands. The dacd (data accept disable) local message provides a means of holding off the DAC handshake on the reception of all Remote Multiline Messages, including MLA (My Listen Address), MTA (My Talk Address), and the Untalk Command. Note that neither MTA nor MLA cause $\overline{IRQ}$ interrupts. Thus, dacd should not be used on these commands unless a polling routine is used by the MPU to monitor R2R. The MPU activates the DAC holdoff feature by writing dacd (bit 1, R3W) high. After each command is received the MPU can release the handshake by writing dacr high. Note that to keep dacd activated, a hex 12 must be written into the Auxiliary Command Register (Figure 3-49). This releases DAC for the current command and keeps the dacd bit high, holding up the handshake on every remote multiline message until it is programmed low.

Bit	7	6	5	4	3	2	1	0
R3R	Chip	DAC	DAV	RFD	msa	rtl	ulpa	fget
R3W	Reset	rtldr	leoi	dacr			dacd	
Write	0	0	0	1	0	0	1	0

AD0167

Figure 3-49. DAC Release with dacd Activated (hex 12)

3.3.1.2 RFD HOLDOFF. The RFD holdoff feature is used in the listener mode to hold up the handshake on the reception of data from the talker. An RFD holdup indicates to the talker the unreadiness for more data by the listener(s). This is interpreted differently from a DAC holdoff. In this case, the talker receives a DAC from the listener and can place the next data byte in the Data Output register of the talker GPIA (making new byte available — nba), preparing itself for the next transfer. This byte is immediately placed on the GPIB, but the handshake does not begin until the

talker makes DAV true. When the talker receives RFD it will make DAV (Data Available) handshake signal true and the byte will automatically initiate a handshake to the listener(s). This provides a much faster data transfer rate than would be possible if the DAC holdoff were implemented for data transfers.

There are two features provided for RFD holdoff (Figure 3-50): 1) hlda (hold on all data) and 2) hlde (hold on end).

Bit	7	6	5	4	3	2	1	0
R2W	dsel	to	lo		hlde	hlda		apte
Write	0	0	0	X	0	1	X	0

(a) RFD hold-off on each data byte received.

R2W	dsel	to	lo		hlde	hlda		apte
Write	0	0	0	X	1	0	X	0

(b) RFD hold-off on End = EOI management line true while ATN is false.
Note that any other desired features in this register must be kept at the appropriate state.

AD0188

Figure 3-50. RFD Holdoff

To hold the RFD handshake on every data byte the MPU writes a 1 into bit 2 of the Address Mode Register (R2W). This halts the data transfer on the GPIB for each data byte until the MPU releases the handshake by writing rfdr (bit 6, R3W)high (Figure 3-51). The rfdr bit is automatically reset on the next E-pulse.

Bit	7	6	5	4	3	2	1	0
R3R	Reset	DAC	DAV	RFD	msa	rtl	ulpa	lget
R3W		rfdr	feoi	dacr			dacd	
Write	0	1	0	0	0	0	0	0

AD0189

Figure 3-51. Ready for Data Release (rfdr — hex 40)

Another feature is provided to hold RFD on the last data byte sent by the talker. The talker, when sending a string of data makes the EOI uniline management line low (by writing feoi high) just before the last data byte. The hlde bit (bit 3, R2W) of the listener, when high, will holdoff the RFD handshake on the data byte received when the $\overline{\text{EOI}}$ line is low.

The RFD handshake is released by sending the rfdr local message (setting bit 6, R3W).

3.3.2 Interrupt Status

Conditions in either the Interrupt Status Register (R0R) or the Command Status Register (R1R) can cause an interrupt; i.e., these conditions will cause the interrupt signal line ($\overline{IRQ}$) to the MPU to go low if the appropriate interrupt masks are set. There are three registers involved, the two just mentioned and the Interrupt Mask Register (R0W). There is another status register, the Address Status Register (R2R), but this register status cannot cause an interrupt and thus is not discussed in this section.

The main status register is the Interrupt Status Register. It is the contents of this register, and only this register, that can cause an interrupt (see Section 3.1.3.1 for details). An $\overline{IRQ}$ interrupt is generated if INT (bit 7, R0R) is set along with its corresponding mask bit (IRQ) in R0W. The $\overline{IRQ}$ output line goes low and remains low until the MPU reads R0R. The $\overline{IRQ}$ line returns high when the MPU reads R0R.

The Interrupt Status Register and Command Status Register are linked together by the CMD bit (bit 2, R0R). The UUCG, DCAS, SPAC, RLC and UACG (bits 0, 1, 2, 3 and 7 respectively) in the Command Status register causes, if dsel (bit 7, R2W) is low, the CMD bit in the Interrupt Status Register to be set. Thus the Command Status Register bits, through an indirect means, can cause an $\overline{IRQ}$ interrupt.

The dsel bit (bit 7, R2W) provides a means of deselecting the GET (R0R), UACG (R1R), DCAS (R1R) and UUCG (R1R) from causing an interrupt (see Section 3.1.3.5 for more details).

Consider only the effects of an Undefined Address Command Group (UACG): Figure 3-52 shows the possible conditions that could occur in the Interrupt and Command Status Registers.

Bit	7	6	5	4	3	2	1	0
R0R	INT	BO	GET	–	APT	CMD	END	BI
Read	0	0	0	1	0	0	0	0
R1R	UACG	REM	LOK	–	RLC	SPAS	DCAS	UUCG
Read	1	0	0	1	0	0	0	0

(a) dsel (bit 7 – R2W) = 1; no $\overline{IRQ}$ generated.

R0R	INT	BO	GET	–	APT	CMD	END	BI
Read	0	0	0	1	0	1	0	0
R1R	UACG	REM	LOK	–	RLC	SPAS	DCAS	UUCG
Read	1	0	0	1	0	0	0	0

(b) dset = 0. No mask bits set; no $\overline{IRQ}$ generated.

R0R	INT	BO	GET	–	APT	CMD	END	BI
Read	1	0	0	1	0	1	0	0
R1R	UACG	REM	LOK	–	RLC	SPAS	DCAS	UUCG
Read	1	0	0	1	0	0	0	0

(c) dsel = 0. Only CMD mask bit set; no $\overline{IRQ}$ generated.

R0R	INT	BO	GET	–	APT	CMD	END	BI
Read	1	0	0	1	0	1	0	0
R1R	UACG	REM	LOK	–	RLC	SPAS	DCAS	UUCG
Read	1	0	0	1	0	0	0	0

(d) dsel = 0. CMD and IRQ mask bits set; $\overline{IRQ}$ generated.

AD0170

Figure 3-52. VACG Interrupt (Effects of dsel and Mask Bits)

The bits in the Command and Interrupt Status Registers are level sensitive. This means that a bit being set is a result of the current condition rather than a latched condition. When any of the conditions causing the set condition are removed, that status bit will return to a low.

For example: if the controller sends a Device Clear message, the GPIA enters the Device Clear Active State and the DCAS bit in R1R is set. If dsel = 0 and the CMD and IRQ interrupt mask bits are set, an $\overline{IRQ}$ will be generated. If the controller mistakenly makes the $\overline{ATN}$ line inactive (high), the GPIA leaves DCAS and all the status bits resulting from the condition are reset. The $\overline{IRQ}$ line however, remains low producing a "ghost interrupt" to the MPU; i.e., the $\overline{IRQ}$ line is low, signaling an interrupt but no Interrupt status bit will be set.

The $\overline{IRQ}$ signal line responds and is set low as a function of a low-to-high transition on its internal input latch. This prevents extraneous interrupts from occurring. Thus, if a status bit is set and causes an interrupt, the $\overline{IRQ}$ signal line goes low. This signals the MPU to enter an interrupt handler. Once in the interrupt handler, the MPU reads R0R and tests the appropriate bits for the origin of the interrupt. If, for example, the BI bit is the only set condition, the MPU detects this and reads the Data-In register. After handling this byte of data, the MPU is ready to leave the routine. Care should be taken when exiting the interrupt handler. After R0R in the above example is read and before the Data-in Register is read, the BI bit is still set. Should another status condition in R0R be set (this is an unlikely event and in most applications will never occur), this condition does not produce an $\overline{IRQ}$. This second condition does not cause an $\overline{IRQ}$ because the first condition kept the level of the INT status bit high while the second condition occurred and hence a low-to-high transition is not detected. The occurrence of such a set of circumstances is rare, but to avoid missing an $\overline{IRQ}$, the last instruction sequence before leaving the interrupt handler, should be a reset of the Interrupt Mask Register followed by programming it to its original state. If a second interrupt went undetected, this sequence will produce the needed low-to-high transition and an $\overline{IRQ}$ does result. The most likely sequence where a second interrupt condition could occur before the first one is serviced is in relation to the Remote Enable Uniline Message. $\overline{REN}$ is an asynchronous occurrence to other activity on the GPIB and can occur at any time. It is possible for $\overline{REN}$ to be sent by the controller causing an RLC status interrupt while another interrupt situation is being serviced. The change in the $\overline{REN}$ line in this sequence does not cause an $\overline{IRQ}$ until the interrupt mask register is reset and then set.

3.3.3 DMA (Direct Memory Access)

The GPIA provides the necessary handshake line to allow its use in DMA mode. These control lines (DMA Grant and DMA Request) are used to handshake data bytes to and from memory with the aid of a DMA Controller (see Chapter 4). The DMA control lines, as well as the specialized operation of the R/$\overline{W}$ line and register select lines (RS0, RS1, RS2), in this mode allow a DMA controller such as Motorola's MC6844 to connect directly to the GPIA without any additional gating circuitry (see Section 3.1.1.3 for description of the control line).

1. DMA REQUEST — This line is an output signal line to the DMA controller used by the GPIA to request a DMA transfer. It is used in either the talker or listener modes and is set high if either BI or BO in the interrupt status register is set. This line always follows the status of BI or BO even if the DMA feature is not used.

2. DMA GRANT — The DMA Grant line is an input signal line to the GPIA from a DMA controller. Once the DMA controller, after receiving a DMA request, has obtained control of the MPU bus it issues a DMA Grant instructing the GPIA to begin the transfer.

NOTE

If the DMA feature of the GPIA is not used, the DMA Grant line must be grounded.

3.3.4 Primary Address Recognition

The Address Register (R4W) gives the GPIA automatic primary address recognition capability (Figure 3-53).

Bits	7	6	5	4	3	2	1	0
R4W	lsbe	da	dat	AD5	AD4	AD3	AD2	AD1
Read	X	X	X	A	A	A	A	A

AD1-AD5 can assume any value except hex 1F as this is an invalid primary address.

A00171

Figure 3-53. Address Register

The value in the five least significant bits of this register must be programmed by the MPU with the address to which the device is to respond to, upon receipt of command from the controller. When the controller sends addressed commands such as MLA and MTA, the address in these commands must correspond to that in the Address Register (R4W) in order to be recognized by the GPIA.

The GPIA provides a means of selecting this address through a set of external switches attached to the device (Figure 3-54). The $\overline{ASE}$ (Address Select Enable) signal output goes low for one E2 clock pulse coincident with the read of the Address Switch Register (R4R). This register does not exist in the GPIA but its MPU address selection is provided. Thus, if a circuit such as is shown in Figure 3-54 is used, when the MPU reads R4R it actually reads the contents of the external switches which have been manually set to the correct address. The MPU then simply performs a write operation to the same address location* and the GPIA is programmed with its IEEE-488 Standard device address. This provides an easy way of placing a device in a system without changing software to change the device address. Care should be taken in the selection of the three most significant bits of the 8-bit word written to R4W. These bit locations, although undefined for addressing, are used in the Address Switch Register (see Chapter 4 for more details).

*This address location is not part of the IEEE-488 Standard system addressing, rather it refers to the location of the GPIA in a memory mapped peripheral arrangement for the MPU. It pertains to MPU addressing and not GPIB addressing.

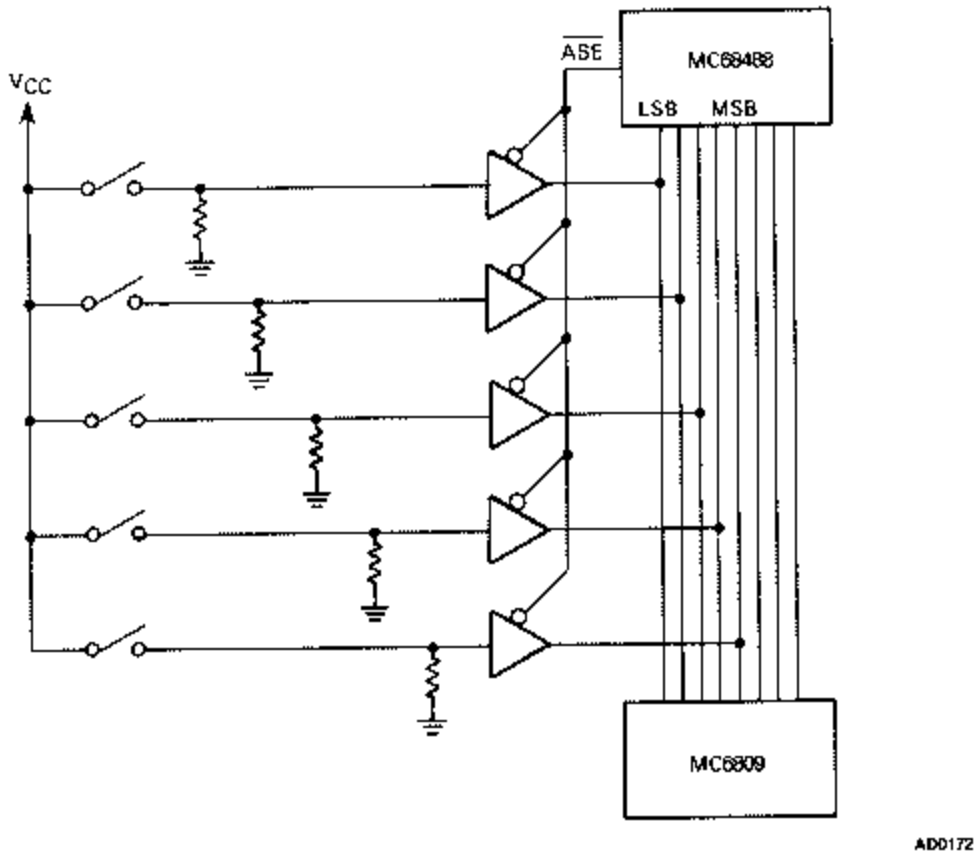


Figure 3-54. Address Switches for Device Addresses

3.4 RESPONSE TO REMOTE MULTILINE MESSAGES

The IEEE-488 Standard defines a set of remote multiline messages that can be sent by the controller. Some of these messages direct specific actions by the interface and thus are handled automatically. Others are better handled through software. Table 3-5 lists all of the IEEE-488 Standard defined remote multiline messages and how each is handled by the GPIA

Table 3-5. MC68488 Response to IEEE-488 Standard Multiline Messages

Multiline Message	MC68488 Response
Addressed Command Group (ACG)[2]	(a) dsel (bit 7, R2W) = 0 UACG (bit 7, R1R) will be set; DAC handshake held off; CMD (bit 2, R0R) will be set; ACG available to MPU through R6R. (b) dsel (bit 7, R2W) = 1 Command ignored.
Device Clear (DCL)	(a) dsel (bit 7, R2W) = 0 DCAC (bit 2, R1R) is set when GPIA is in Device Clear Active State (when $\overline{ATN}$ is asserted and DCL received); DAC handshake held off; CMD (bit 2, R0R) will be set. (b) dsel (bit 7, R2W) = 1 Command ignored
Group Execute Trigger (GET)	(a) dsel (bit 7, R2W) = 0 GET (bit 5, R0R) is set when MC68488 is in Device Trigger Active State; Trigger output (pin 24) is set high; DAC held off. (b) dsel (bit 7, R2W) = 1 Command ignored.
Go To Local (GTL)	Will reset REM (bit 6, R1R).
Local Lock Out (LLO)	LOK (bit 5, R1R is set when MC68488 is in Local State (LOCS) or Remote State (REMS) and $\overline{REN}$ input (pin 22) is low.
Listener Address Group (LAG)	(a) apte (bit 0, R2W) = 0 1. My Listen Address (MLA) — if dal (bit 6, R4W) is not set, the MC68488 will enter LACS after $\overline{ATN}$ goes high. LACS status bit (bit 2, R2R) and ma (bit 7, R2R) will be set. 2. Other Listen Address (OLA) — ignored 3. Secondary Command Group (SCG) — ignored (b) apte (bit 0, R2W) = 1 1. My Listen Address (MLA) — if dal (bit 6, R4W) is not set, the MC68488 will enter LPAS; LPAS status bit (bit 1, R2R) will be set. 2. Other listen address (OLA) (a) if in LPAS the MC68488 will go to LIDS. LPAS (bit 1, R2R) will be reset. (b) if not in LPAS the OLA will be ignored 3. Secondary Command Group (SCG) — APT (bit 3, R0R) will be set; DAC held off; Secondary Command available in R6R.
Parallel Poll Configure (PPC)[1]	Treated as ACG — See Address Command Group; PPC code recognition in MPU software.
Parallel Poll Enable[2]	Treated as SCG — See Secondary Command Group recognition: Specific code assignment recognition in MPU software.
Parallel Poll Disable (PPD)[1]	Treated as SCG — See Secondary Command Group recognition: specific code assignment recognition in MPU software.
Parallel Poll Response 1-8 (PPR1-PPR8)	Appropriate code placed in parallel poll register (R6W) by MPU; contents of R6W placed on GPIB when controller makes EOI and $\overline{ATN}$ true.
Parallel Poll Unconfigure (PPU)[1]	Treated as UCG — See Universal Command Group: Specific Command code recognition in MPU software.
Secondary Command Group[2]	(a) apte (bit 0, R2W) = 0 Command ignored. (b) apte (bit 0, R2W) = 1 APT (bit 3, R0R) will be set; DAC held off; Secondary Command available in R6R.
Selected Device Clear (SDC)[2]	Response is the same as Device Clear (DCL) except that the GPIA must be in LADS to respond to SDC.
Serial Poll Disable (SPD)	This command will remove the GPIA from the Serial Poll Mode. 1. If the MC68488 is in SPAS the device will be placed back in TADS: SPAS status bit (bit 2, R1R) and CMD status bit (bit 2, R0R) will be reset. 2. If MC68488 is not in SPAS, no status bits will be affected. However, since the latched Serial Poll Mode is removed, the SPAS state cannot be entered unless SPE is again received.

— Continued

Table 3-5. MC68488 Response to IEEE-488 Standard Multiline Messages (Continued)

Multiline Message	MC68488 Response
Serial Poll Enable (SPE)	This command will place the GPIA in a latched Serial Poll Mode 1. If the GPIA is in TADS when SPE is received, the MC68488 will move to the Serial Poll Active State when $\overline{ATN}$ is released. The SPAS status bit (bit 2, R1R) and CMD status bit (bit 2, R0R) will be set. 2. If the MC68488 is not initially in TADS, the GPIA will remain in the Serial Poll Mode but no status bits will be set. If the controller places the GPIA in TADS after sending SPE by sending MTA, the device will move to SPAS when $\overline{ATN}$ is released, and the appropriate status bits as described in 1 will be set.
Take Control (TCT)	Treated as ACG – See Address Command Group: Specific Command Code recognition in MPU software.
Talk Address Group (TAG)	(a) apte (bit 0, R2W) = 0 1. My Talk Address (MTA) – if dat (bit 5, R4W) is not set, the MC68488 will enter TACS. TACS Status bit (bit 3, R2R) and ma (bit 7, R2R) will be set. 2. Other Talk Address (OTA) – will put MC68488 in TIDS; if MC68488 has been in TACS, the TACS status (bit 3, R2R) and ma status (bit 7, R2R) will be reset. 3. Secondary Command Group (SCG) – ignored. (b) apte (bit 0, R2W) = 1 1. My Talk Address (MTA) – if dat (bit 5, R4W) is not set the MC68488 will enter TPAS; TPAS status (bit 0, R2R) will be set. 2. Other Talk Address (OTA) – will put MC68488 in TIDS; if MC68488 has been in TACS or TPAS their respective status bits will be reset. 3. Secondary Command Group – APT status (bit 3, R0R) will be set; DAC held off; Secondary Command available in R6R.
Universal Command Group[2]	(a) dsel (bit 7, R2W) = 0 UUCG (bit 0, R1R) will be set; DAC handshake held off; CMD (bit 2, R0R) will be set. UUCG available to MPU through R6R. (b) dsel (bit 7, R2W) = 1 Command ignored
Unlisten (UNL)[2]	Places MC68488 in LIDS[2]
Untalk (UNT)[2]	Places MC68488 in TIDS[2]

[1]The responses to these commands will remain as long as the command remains on the GPIB. However, should $\overline{ATN}$ be made false or the command code removed, the conditions indicated will also be removed.

[2]The responses to these commands will remain as long as the command remains on the GPIB. However, should $\overline{ATN}$ be made false or the command code removed, the conditions indicated will also be removed.

CHAPTER 4
PROGRAMMING CONSIDERATIONS

4.1 HARDWARE CONFIGURATION

The MC68488 is fully compatible with Motorola's 6800 family and needs no special circuitry. It can also be used with any MPU based system as long as the bus characteristics of the GPIA are met. Figure 4-1 contains a block diagram of an expanded GPIA/MPU system.

4.1.1 MPU Bus

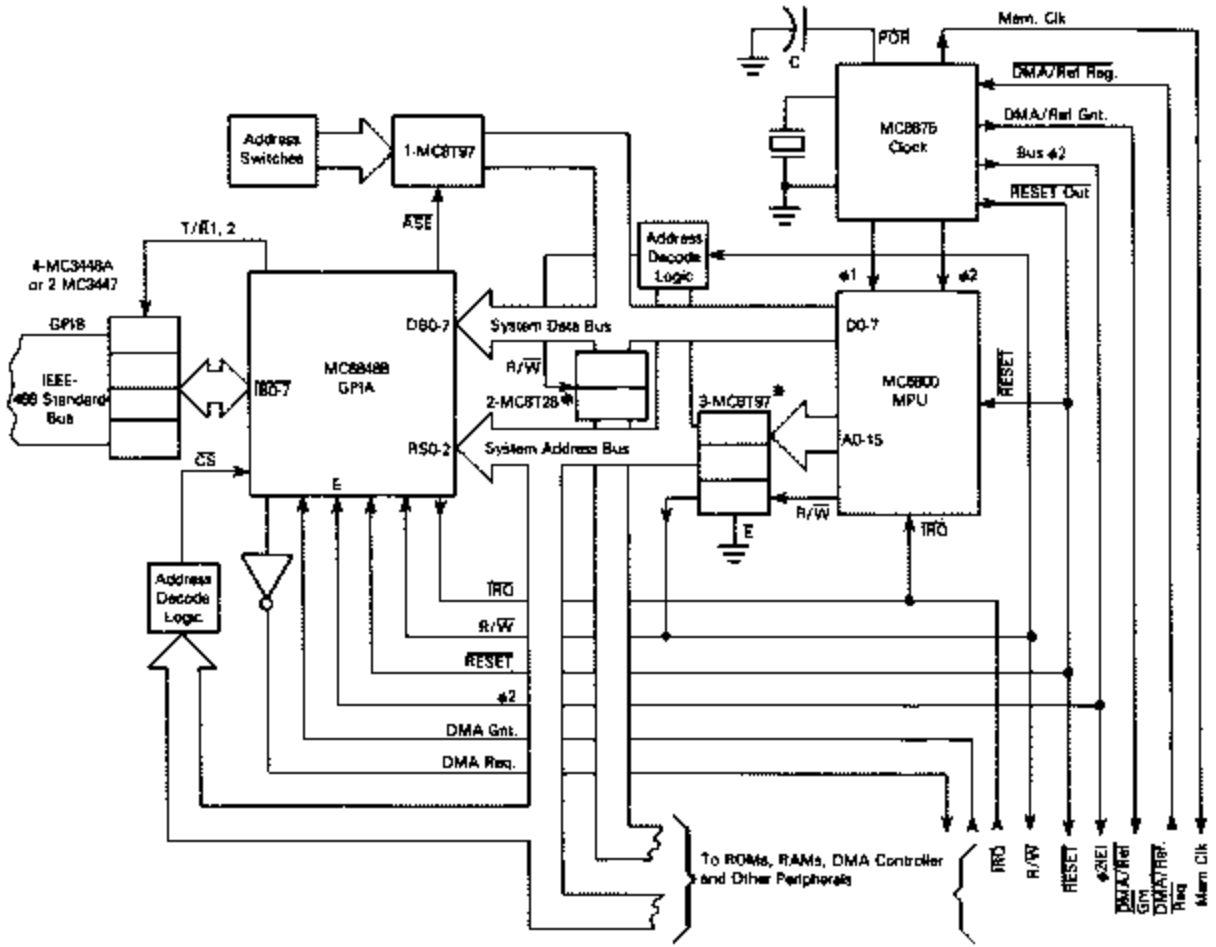
The eight data lines applied to the MPU are bidirectional and are in the high impedance condition until the MPU performs a read operation. During a read, they are capable of driving one TTL load. In a multiboard system it may be necessary to buffer these lines for added drive capability.

There are 15 internal registers in the GPIA that are accessed by the 3 least significant bits of the MPU address bus in conjunction with the Read/Write line. Address lines A0, A1 and A2 go to register select lines RS0, RS1, and RS2, respectively. See Table 3-2 for details. The GPIA uses eight address locations in the MPU address structure. These eight address locations can be placed anywhere in the address map by decoding the appropriate address lines used to activate Chip Select ($\overline{CS}$).

Remember that there are two separate address categories on the GPIA. The address system just described represents the way the MPU addresses, through its address bus, the GPIA internal registers. This is distinct from the device (instrument) address which is used by the GPIB controller when it sends addressed commands to a specific device. The device address is the five least significant bits of an 8-bit word that appears on the address switch configuration of an instrument and is placed in the address Switch Register (R4W) of the GPIA. Thus, the MPU uses its address bus to select specific registers in the GPIA (e.g., reading the contents of the Data-In Register — R7R) and the GPIB controller uses the device address (contents of bits 0-5 of R4W) to send addressed commands, (e.g., talk to listen commands).

The $\overline{ASE}$ (Address Switch Enable) signal line on the MC68488 provides a convenient method of using switches on the back panel of an instrument to place the device address in the GPIA. The $\overline{ASE}$ line goes low when the MPU performs a read of register 4 (R4R). If the $\overline{ASE}$ line is connected as shown in Figure 4-2, the MPU will read the contents of the switch register when it tries to read R4R. A write operation to the register 4 location following the

LDAA	R4R
STAA	R4W

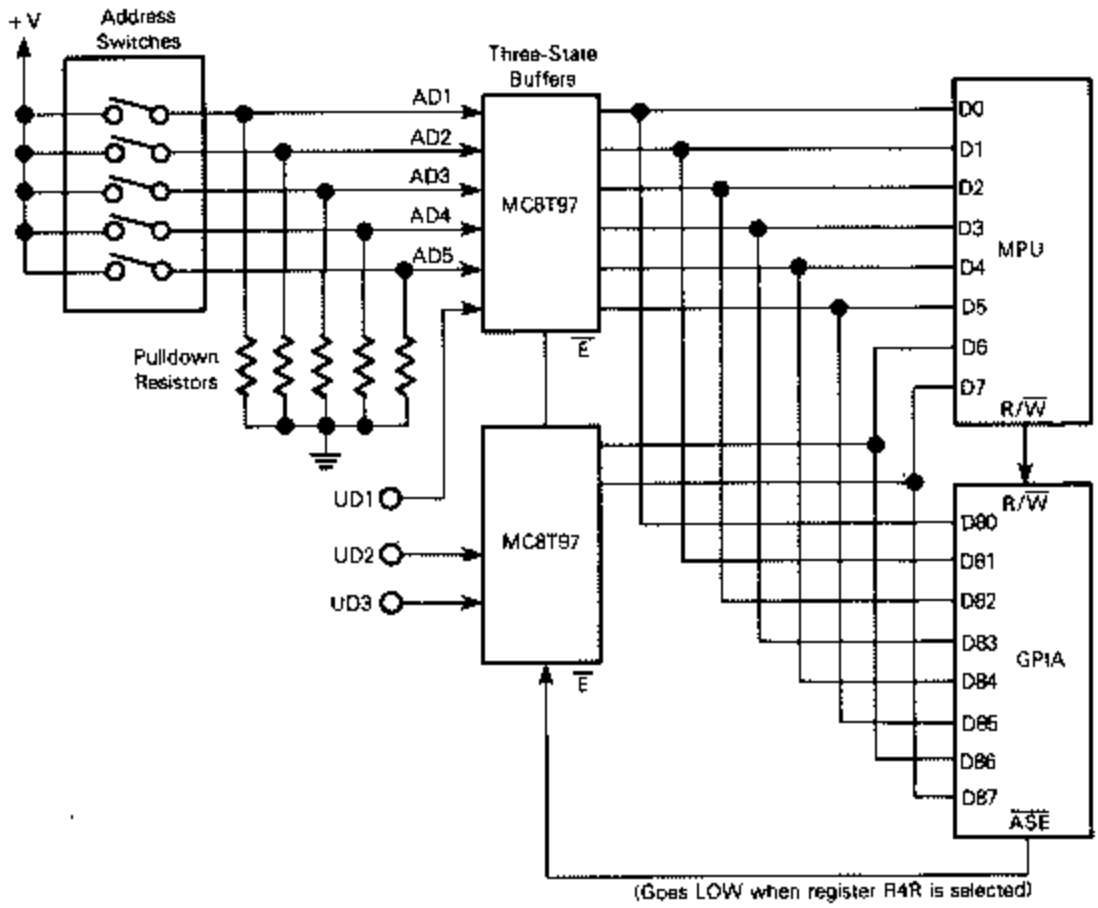


*Required in larger systems to provide buffering of the data and address buses for increased drive capability.

AD0173

Figure 4-1. Expanded GPIA/MPU System

read operation will place the contents of the switch register into the MC68488. With this procedure, the device address can be changed by simply changing the position of the address switches. The MPU software does not have to be altered.



Notes:
1. Tie UD1 HIGH for LISTENER ONLY device, otherwise tie LOW.
2. Tie UD2 HIGH for TALKER ONLY device, otherwise tie LOW.
3. With UD3 tied LOW, Dual Primary Mode can be enabled with subsequent software.

AD0174

Figure 4-2. Device Address Connections

The ENABLE (E) input is a clock signal input that is used:

1. to enable data transfers on the MPU bus. During a Read operation, data is made available to the MPU during the high time of the Enable pulse. During a write operation, data is latched into the addressed internal register of the MC68488 on the high-to-low transition of the E-pulse.
2. to release the DAC handshake line when in the listener mode and data byte has been received. When the MC68488 is a listener and receiving data from the addressed talker, the DAC handshake line is held off until the MPU reads the Data-In Register (R7R). The high-to-low transition of the E-pulse when R7R is addressed by the MPU initiates the release of DAC indicating to the addressed talker the acceptance of the data byte.
3. as a state counter. The E-pulse is used to move the MC68488 through the state of the source and acceptor function.
4. to enable or release many of the status conditions.

The MC68488 is completely static, and as such, the removal of the E-pulse as a clock does not destroy any of the stored conditions in the chip. Since the E-pulse is used as a state counter, however, it is important to maintain a clock allowing state transitions in the IEEE-488 Standard state diagrams. Because of this, the removal of the E clock prevents state transitions and could cause the MC68488 to hang-up the GPIB and not allow data transfer.

The E clock signal can be operated at a lower frequency than its specified maximum value and need not be symmetrical as long as the timing constraints, as listed in the data sheet, are adhered to.

If a 6800 Family microprocessor is used, the E signal should be the same as the MPU $\phi 2$ clock signal. If a microprocessor without a $\phi 2$ (E) clock is used, a free running clock must be provided that can be synchronized to the Read/Write operation when the internal registers of the MC68488 are being accessed.

4.1.2 Bus Transceiver

4.1.2.1 MC3448A. The MC3448A is a quad bidirectional transceiver for mating MOS or bipolar logic systems to the IEEE-488 Standard bus. Each channel provides back-to-back driver and receiver elements plus the required bus terminations. Direction of data flow is controlled by three-state disabling of the undesired direction element, i.e., driver or receiver; Schottky technology assures high speed while PNP buffered input structures guarantee low input loading for MOS compatibility. Both driver and receiver elements are non-inverting.

A pullup enable input is provided on each pair of drivers which allows selection of an open-collector or three-state driver configuration.

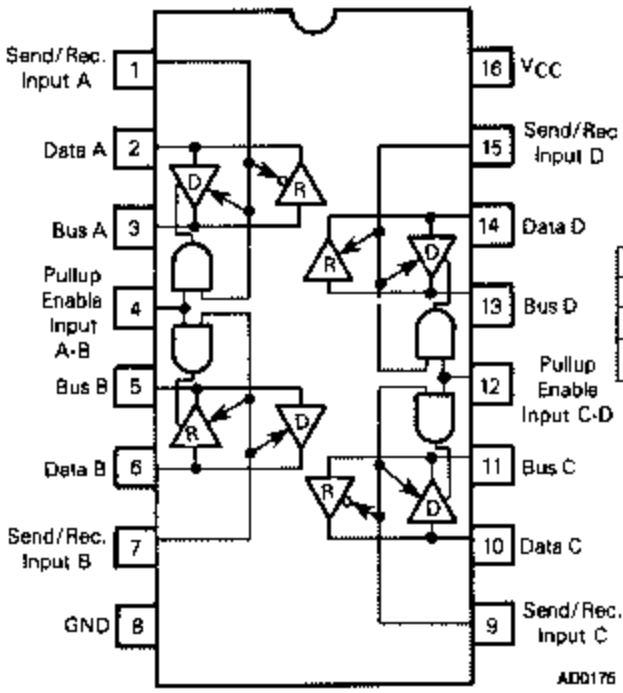
Additional features include:

- Minimum receiver hysteresis of 400 mV for improved noise immunity
- Power up/down protection to assure that no invalid information is transmitted to the bus during these time periods
- No bus loading (including terminations when power is removed from the device)
- Fast propagation delay times
- Selection of a three-state or open-collector configuration
- 48 mA drive capability.

The pinouts and truth table for the MC3448A are given in Figure 4-3.

Four MC3448A transceivers are required to buffer the 16 bus lines. Figure 4-4 shows the connections of the transceiver Send/Receiver and Pullup enable inputs.

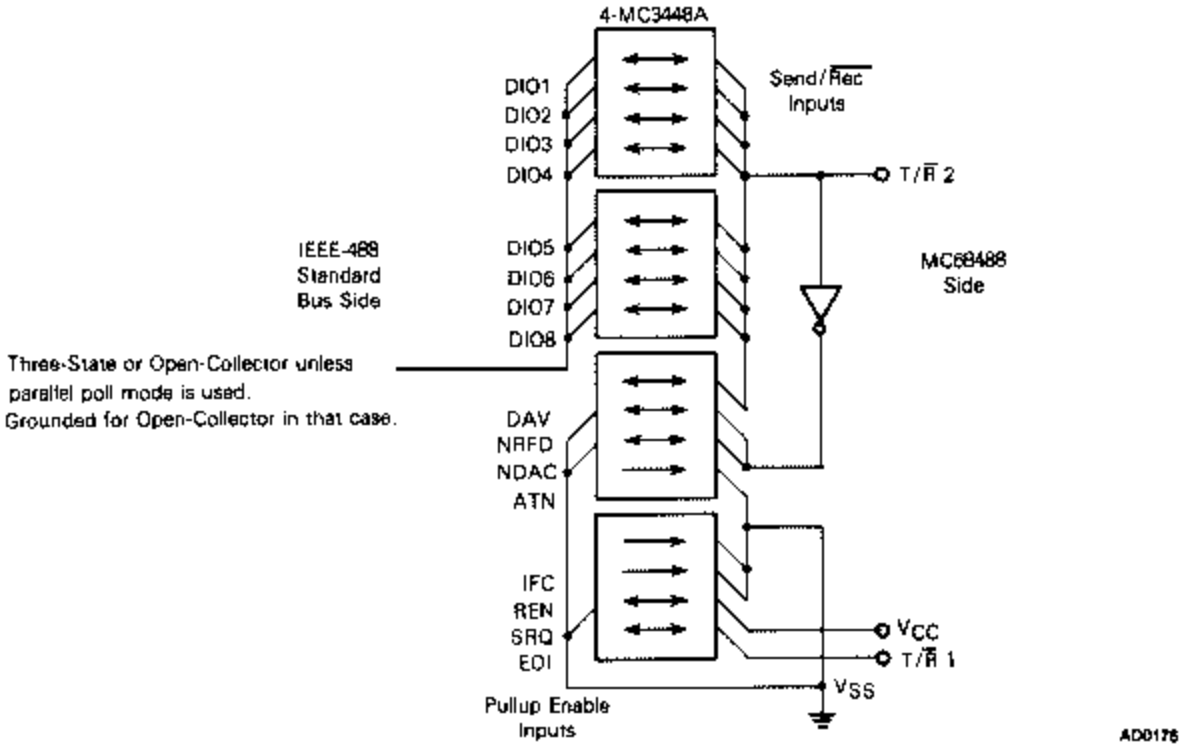
Figure 4-5 shows the proposed Motorola IEEE-488 Standard system along with the required interconnections.



TRUTH TABLE

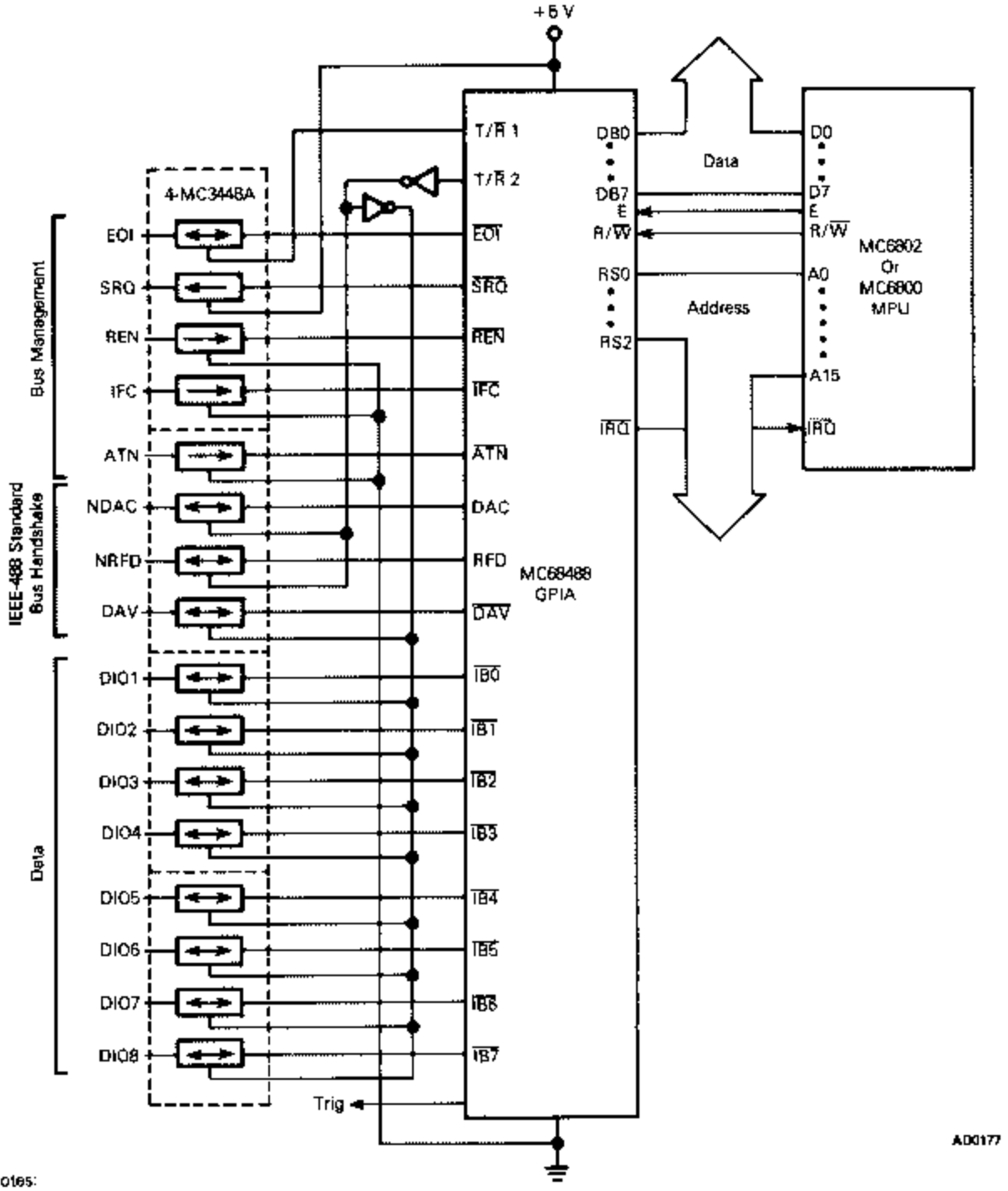
Send/Rec.	Enable	Info. Flow	Comments
0	X	Bus – Data	–
1	1	Data – Bus	Active Pullup
1	0	Data – Bus	Open Collector

Figure 4-3. MC3448A Pinouts and Truth Tables



Note: Although each Pullup Enable Input controls a pair of drivers, individual control is shown here to illustrate which lines must be tied low for open-collector operation.

Figure 4-4. MC3448A Direction Control and Pullup Enable



Notes:

1. Although the MC3448A transceivers are non-inverting, the bus callouts appear inverted with respect to the MC68488 pin designations. This is because the IEEE-488 Standard is defined for negative logic while all M6800 MPU components make use of positive logic format.
2. Unless proper considerations are provided, it is recommended that the pullup enable pins on the MC3448A be grounded selecting the open-collector mode.

Figure 4-5. Drivers/GPIA Configuration Using MC3448A Devices

4.1.2.2 MC3447. The MC3447 is an octal bidirectional transceiver for mating MOS or bipolar logic systems to the IEEE-488 Standard bus. Only two MC3447 devices are required for interfacing the GPIA to the IEEE-488 Standard bus. In addition, the T/$\overline{R}$ 1, 2 control lines from the GPIA can be directly connected to these drivers without the need of an inverter.

4.2 GPIA OPERATIONS

4.2.1 Initialization

A hardware $\overline{\text{RESET}}$ (applied to pin 19) provides a means of resetting the GPIA. It places the MC68488 in the idle state for each of the interface functions. Section 3.1.1.1 describes the effects of a hardware reset. An internal reset condition is latched in the GPIA (bit 7, R3W is set high) to prevent erroneous output transitions. This internal reset condition remains until bit 7, R3W is programmed low by the MPU. During the reset condition, the only accessable register locations are the Addrss Switch Register (R4R), the Address Register (R4W), and Auxiliary Command Register bit 7 (R3W, bit 7). Thus, it is possible to place the device address in R4W before writing the reset bit low which allows the controller to have access to the device. After a hardware reset and prior to programming R2W, the contents of the Address Register is hex 00. Should the GPIA be taken out of reset (bit 7, R3W written low) at this time, it is possible for the device to respond to an addressed command with a 00000 as the address. To prevent this, it is recommended that the correct address be placed in R4W prior to writing the reset bit low.

The RESET bit (bit 7, R3W) when high prevents the MPU from writing to any of the GPIA registers except R4W/R4R and itself. Bits 0-6 of R3W are not accessible to the MPU until the RESET bit has been programmed low; i.e., the same write instruction that write bit 7, R3W low cannot be used to program the other 7 bits.

If back panel address switches are used for the device address, an example initialization procedure is as follows:

1. Set address switches to desired Device Address.
2. Turn on power and provide a hardware reset.
3. Load the contents of the GPIA register 4 address location (R4R) into the accumulator. This enables the three-state address switch buffers and applies the contents to the MPU data bus.
4. Store the contents of the accumulator in the GPIA register 4 address location (R4W). This places the contents of the address switches into the GPIA Address Register (R4W).
5. Remove the latched reset condition (program bit 7, R3W low).
6. Set desired GPIA features, e.g., interrupt masks, local messages, etc.

In some cases, back panel address switches are not used. In such a case, the correct device address must be resident in a software routine and this routine places the address in the GPIA.

4.2.2 Interrupt Status Register Monitoring

The Interrupt Status Register (R0R) is an 8-bit register that contains the primary status information of the GPIA. Only the contents of this register are directly capable of generating an interrupt. It is through this register that the MPU determines when to output another byte; when, in the listener mode, a byte is available to the MPU; when the last data byte in a long string has been received, etc. Bit 2 in R0R (CMD) links the Command Status Register (R1R) to the Interrupt Status Register. The CMD bit reports the presence of a status condition in R1R to the Interrupt Status Register. Thus, through an indirect means (CMD bit) the contents of the Command Status Register (R1R) can be made to cause an $\overline{IRQ}$ interrupt. These two registers form the complete interrupting capability of the GPIA.

The Interrupt Status Register can be monitored through either an interrupt sequence or a polling sequence. In a polling sequence, the MPU checks the register on a routine basis. The data rate on the GPIB is in part governed by the MPU. For high data rates, a polling sequence may require full time monitoring by the MPU. If a polling operation is desired, the appropriate Interrupt Mask Register bits should still be set causing bit 7 in R0R (INT) to be set if any of the desired status conditions occur. Thus, the polling routine only need check bit 7 and if bit 7 = 0 then no further action is required. If bit 7 = 1 then the desired bits need to be checked. The IRQ Interrupt Mask Bit (bit 7, R0W) must not be set high for a polling operation as this bit, when high enables the $\overline{IRQ}$ output line.

To free the MPU for other activities, the GPIA status conditions can be monitored in an interrupt mode by setting bit 7 of R0W along with the other appropriate interrupt mask bits. In this mode, the MPU can be engaged in other tasks and when a status condition requiring service occurs, the MPU is interrupted. After an $\overline{IRQ}$, the MPU interrupt handler can check bit 7 of R0R to be sure the interrupt is a result of the GPIA and not some other device in the system. If this bit is set, the handler routine can then check the other status bits in the order set up in the program. Section 3.2.2 further describes the use of the interrupt structure of the GPIA. Note that status conditions in the Address Status Register (R2R) do not cause interrupts. This register is used to indicate when the GPIA is an active talker or listener but these conditions do not directly cause interrupts. However, the occurrence of being in either the Talker Active State (TACS) or Listener Active State (LACS) can be obtained when the BO (Byte Out) or BI (Byte In) status bits are set. As soon as the GPIA is placed in TACS, the BO status bit is set indicating to the MPU that the GPIA is in TACS and the Data-Out Register is empty. When in LACS and a data byte is received in the Data-In Register, the BI status bit is set.

4.2.3 Data Handling

Once the device has been addressed, data is passed through the MC68488 via either the Data-In Register (R7R) when a listener, or the Data-Out Register (R7W) when a talker.

4.2.3.1 SENDING DATA. When the device is addressed as a talker, the GPIA enters TACS (Talker Active State) when the controller makes (asserts) $\overline{ATN}$ false. The BO (Byte-Out) Status bit in R0R is set, flagging the MPU to write a data byte to R7W. After writing to R7W, the BO status bit goes

low. The 3-line handshake is set to the appropriate state and the data byte placed on the GPIB data bus. (The Talker controls $\overline{DAV}$ and asserts the line to make data available to the listeners. The listeners control DAC and RFD, sending these signals back to the talker indicating when the listeners have accepted data and when they are ready for more data.) After all listeners have been accepted the data (DAC becomes true), BO is again set, indicating to the MPU that the data has been accepted and another data byte can be placed in the Data-Out Register. The next data byte is placed on the GPIB as soon as it is written to R7W. The $\overline{DAV}$ (Data Available) line is made true when the talker receives an RFD (Ready for Data) from the listeners. All listeners must be ready for data before the RFD line is asserted true. An example of a basic talker routine flow chart is shown in Figure 4-6. In this routine a string of data is sent by the talker outputting the required number of data bytes to the addressed listener(s). The end of the data transmission is indicated by making the EOI management line true just prior to sending the final byte. When a listener detects a true state on the EOI management line, it knows the next data byte received in R7R will be the final one. The use of EOI is, of course, optional.

The procedure used for sending the last byte is described below. When using the EOI management line, the MPU software must first set the feoi control bit (asserting $\overline{EOI}$) and then send the last byte. When the last byte is accepted by all listeners, the BO status bit of the talker device is set. The BO status bit is not qualified with the $\overline{EOI}$ line but is set whenever the current data byte is accepted by all listeners and the device is in the Talker Active State (TACS). (Note that when the controller asserts $\overline{ATN}$ to send commands, the GPIA moves out of TACS causing BO to reset, and remains out of TACS as long as $\overline{ATN}$ is asserted.) After the data block transfer the controller takes control of the bus (asserts $\overline{ATN}$) and reconfigures the GPIB system. In performing this task, the controller sends command(s) that untalk the device (MLA, OTA, UNT) or reassigns it as a Talker (MTA) asking for further data transfers. Since the GPIB operates asynchronously with respect to the device MPU bus it is possible for the controller to take control of the GPIB and cause actions that change the state of the BO status bit in the middle of the MPU interrupt routine. As a result, care needs to be exercised when responding to the BO status bit interrupt occurring after transferring the last byte. Any of the following conditions can occur.

1. Device Untalked — If either My Listen Address, Other Talk Address, or the Untalk command is sent the device is placed in the Talker Idle State (TIDS) — the device is Untalked. In this case the BO status bit is set as soon as the last data byte is accepted, reset when the controller asserts $\overline{ATN}$, and BO will remain reset after $\overline{ATN}$ is released. (a) The BO status bit indicates a set condition if the MPU reads the Interrupt Status Register before the controller asserts $\overline{ATN}$. This status indication, however, is misleading as another byte transfer is not intended. The device is soon to be Untalked. (b) The BO status bit indicates a reset condition if the MPU reads the Interrupt Status Register after $\overline{ATN}$ has been asserted — a "ghost interrupt" is produced. This BO status bit remains reset after $\overline{ATN}$ is released.
2. Device Reassigned as a Talker — The controller reassigns the device to talk by sending My Talk Address. In this case the BO status bit is set as soon as the last data byte is accepted, reset when the controller asserts $\overline{ATN}$ to send MTA, and is again set when $\overline{ATN}$ is released by the controller. (a) The BO status bit indicates a set condition if the MPU reads the Interrupt Status Register before the controller asserts $\overline{ATN}$. This case is identical to part (a) for "Device Untalked" shown above. (b) The BO status bit indicates a set condition if the MPU reads the In-

terrupt Status Register while $\overline{ATN}$ is asserted — a "ghost interrupt" is produced. (c) The BO status bit indicates a set condition if the MPU reads the Interrupt Status Register after $\overline{ATN}$ is released. This status indication is requesting a byte transfer and should be acted upon accordingly.

To alleviate the above ambiguity and ghost interrupt situation, the GPIB handshake must be synchronized with action by the device MPU. The following step-by-step procedure provides this needed synchronization and eliminates the ambiguity when servicing the BO status bit after sending the last byte.

1. Before sending the last byte of a block transfer, the feoi bit (if used) should be set. In addition, the dacd bit in R3W should be set holding off the handshake upon reception of any command (establishes the required synchronization between MPU and controller).
2. If operating under interrupts, the BO interrupt mask bit should be reset. This prevents generations of a BO status interrupt when the last byte is received.
3. Send the last data byte.
4. The MPU now monitors the $\overline{ATN}$ bit in the Address Status Register (R2R). When the ATN bit is set, the $\overline{ATN}$ line has been asserted and it will remain asserted until completion of the handshake. The procedure described herein, assumes that $\overline{ATN}$ is asserted between block transfers and at least one command sent. The fact that $\overline{ATN}$ is asserted indicates that the device is no longer in TACS and, thus, the BO status bit is reset.
5. The dacd bit in R3W can now be written low removing the manual handshake hold-off on subsequent commands. With the same write instruction, the dacr bit should be set releasing the handshake on the current command (write a hex 10 to R3W).
6. The BO interrupt mask bit can now be set, enabling interrupts for another block transfer.

After following this procedure, a BO status condition will occur only if a second block of a data is requested by the controller. In addition, the possibility of a BO "ghost interrupt" is eliminated.

A data exchange normally takes place without controller intervention. There are, however, many instances where the controller breaks the flow of data. The controller can take control of the bus by asserting $\overline{ATN}$ (low). When this occurs, all active GPIAs on the bus become listeners and monitor the GPIB for commands. The controller might intervene if it detects an undefined state on the GPIB, etc. The controller can interrupt the data transfer asynchronously or synchronously.

An asynchronous bus take-over is usually the result of a major problem on the GPIB that requires immediate attention (e.g., the bus being in an undefined state). If a device dependent message is true (e.g., data being transferred from talker to listeners) and $\overline{ATN}$ becomes true (low), the interrupted byte could be misinterpreted by other devices as an interface message (for example, commands or addresses) and produce unintended state transitions. As a result an asynchronous take-over (asserting the Attention line, $\overline{ATN}$, during the handshake sequence) might place the GPIA in an undefined state. This type of intervention must be immediately followed by an $\overline{IFC}$ (with $\overline{ATN}$ high) from the controller, placing the GPIA(s) in a known state, e.g.; after the controller asserts $\overline{ATN}$ asynchronously, it must then remove $\overline{ATN}$ and send the $\overline{IFC}$ message before allowing another handshake to take place. Once in a known state, the controller can reconfigure the bus (see Question 8 — Appendix B).

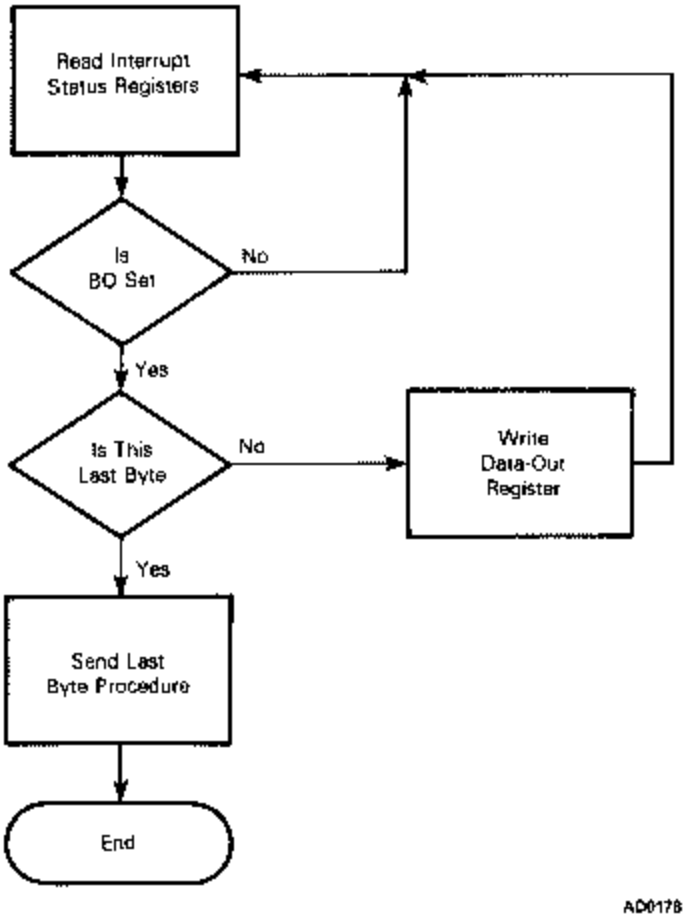


Figure 4-6. Basic Talker Routine

A synchronous bus take-over is often used with only a temporary interruption of the data flow. To take over the bus synchronously, the controller simply monitors the GPIB conditions and asserts the $\overline{\text{ATN}}$ line during an idle condition in the handshake process (between handshakes). To do this the controller must wait until all listeners have accepted the current data byte and before the next data byte has been made available. Under these conditions the controller can intervene and use the data bus without causing ambiguity in the interrupted data transfer.

Anytime the controller takes over the bus in a synchronous fashion, the GPIA is not adversely affected. In the talker mode, it is conceivable that the take-over could occur between the time the MPU has written a byte of data to the Data-Out Register (made nba true in the source state diagram) and before this byte becomes valid on the GPIB. If this occurs (for example, the controller conducts a Serial Poll of the device) and then the controller gets off the bus, placing the original talker back in TACS, the data byte previously placed in R7W is automatically transferred (after handshake) to the listeners. An IFC being sent by the controller at any time does not destroy nba.

4.2.3.2 RECEIVING DATA. When the device is addressed as a listener, the GPIA will enter LACS (Listener Active State). The GPIA does not interrupt the MPU until the first data byte is received from the addressed talker. When received, the BI (Byte-In) status bit in R0R is set, flagging the MPU to read the Data-In Register (R7R).

The DAC handshake is held-off on the reception of each byte. DAC is released and the BI status bit reset when the MPU reads the Data-In Register. The handshake is automatically completed after the read of R7R. A Basic Listener Routine is shown in Figure 4-7. When the talker sends the last data byte in a string it sets the EOI management line. When this occurs, the END status bit in R0R of the listener GPIA is set, flagging the MPU that the final data byte has just been received. The END status bit, like the BI status bit, is reset when the MPU reads the Data-In Register.

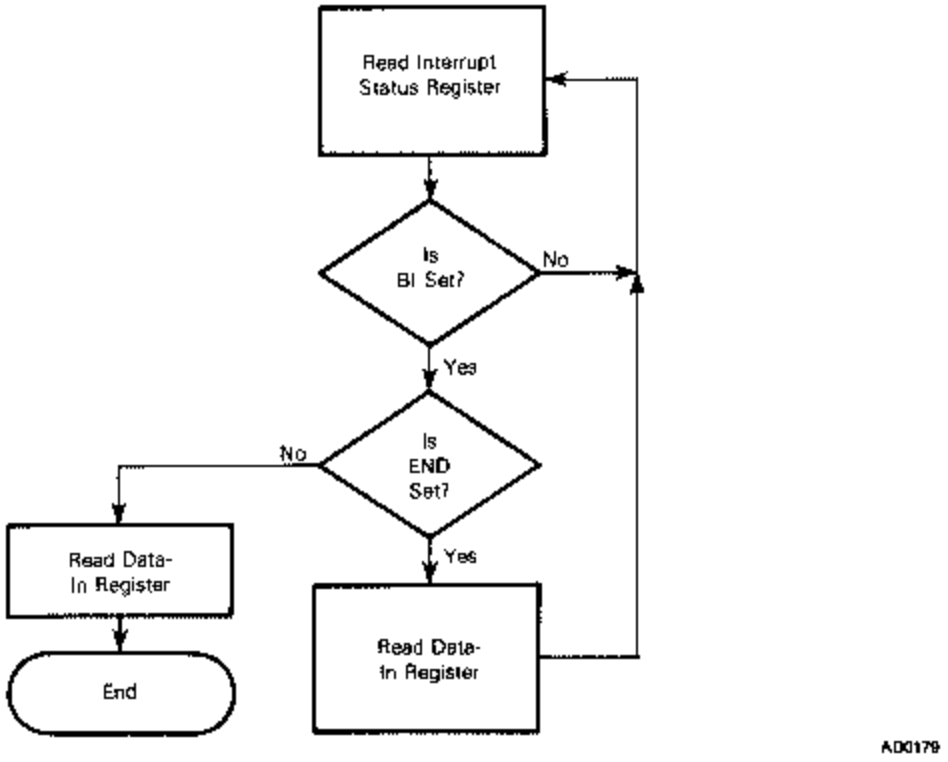


Figure 4-7. Basic Listener Routine

The listener GPIA, unless programmed differently, automatically completes the handshake when R7R is read. Reading R7R releases DAC which is followed by an automatic release of RFD completing the handshake sequence. Once RFD indicates a readiness for more data, the talker can send another data byte (make $\overline{DAV}$ true). In some situations, however, it is desirable to hold off the handshake to temporarily halt the beginning of the next data byte transfer. For instance, the MPU may need to perform some task after each data byte (e.g., perform data calculations on a per byte basis). The GPIA provides two features for holding off the handshake (see Section 3.3.1.2 for details). The hlda and hlde features prevent the handshake completion by holding off the handshake sequence. The MPU allows the handshake to proceed by writing rfdr high in R3W. Thus, the MPU

can read the data byte from R7R without allowing the next handshake sequence. The hlda feature holds off the RFD handshake on each data byte received. This gives the MPU the option of looking at each data byte on a per byte basis. The other feature, hlde, holds off the RFD handshake on the data byte received when the EOI management line is true (low).

If the controller takes over the GPIB asynchronously (asserting $\overline{ATN}$ when the handshake is not in an idle state) it cannot be assumed that the listener GPIA will go back to LACS when $\overline{ATN}$ becomes false (high). An asynchronous take-over must be immediately followed by an $\overline{IFC}$ (with $\overline{ATN}$ high) and then the controller can reconfigure the GPIA, as discussed in Section 4.2.3.1.

If the controller takes over the bus synchronously (forcing $\overline{ATN}$ low during the idle time in the handshake) and then makes $\overline{ATN}$ high without reconfiguring the listener GPIA, the MC68488 will return to LACS.

The BI status bit in R0R reports to the MPU the presence of a data byte in R7R. Asserting $\overline{IFC}$ does not reset BI. The BI status bit can only be reset by a hardware reset (pin 19), software reset (bit 7, R3W), or by reading R7R.

4.2.4 GPIA Interface Operation Modes

Each of the interface functions of the GPIA is entered through a programming protocol and the interaction of both the GPIB controller and the GPIA MPU. Before any of the interface functions can be established, the MC68488 must first be initialized as mentioned earlier. A flow diagram of the Primary initialization is given in Figure 4-8. A read of R4R is not necessary if the MPU software is to supply the device address.

The initialization routine described here is used when an active controller is to place the GPIA in one of its operating modes. If a controller is not used and the GPIA is placed in the talker or listener mode through the talk only (to) or listen only (lo) local messages, then the address registers (R4R and R4W) are not used and these steps can be omitted from the initialization routine.

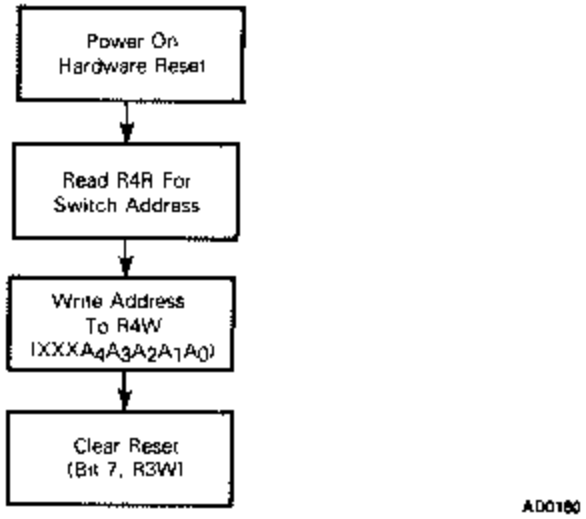


Figure 4-8. Primary Initialization Routine

4.2.4.1 TALKER/LISTENER ADDRESSING MODES. There are three additional addressing modes available to the GPIA user. The controller can command the GPIA to be a talker or listener through either the 1) Primary Addressing Mode (Basic listener/talker) 2) Dual Primary Addressing Mode, or 3) Secondary Addressing Mode. In configuring the system, the controller should either keep ATN asserted through the entire system initialization or make all listener assignments before sending the talk command. Once the talker assignment has been made and ATN is set false (high), this device can begin sending data. If all listener assignments have not been made at this time, confusion and missed data could result.

Primary Address Mode

After the Primary Initialization Routine has been completed, the controller can address the GPIA to be either a talker or listener in the Primary Address Mode. At this point, the GPIA remains in the idle state until MTA (My Talk Address) or MLA (My Listen Address) is sent (Figure 4-9).

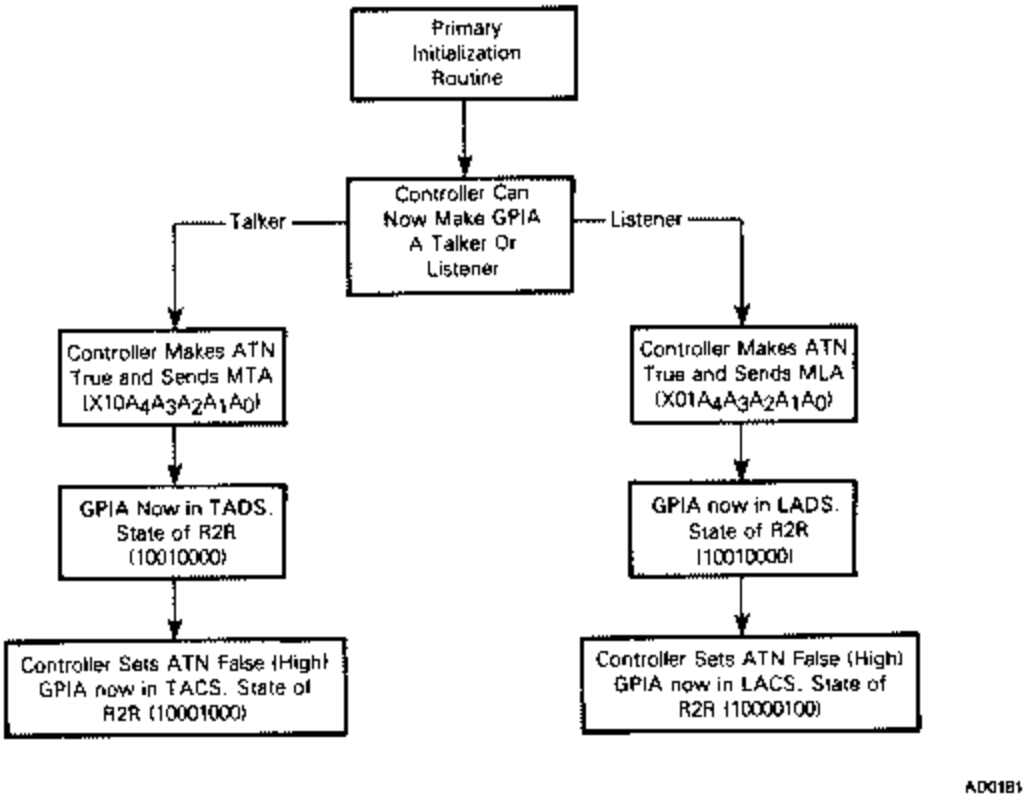


Figure 4-9. Primary Talker/Listener Sequence

If MTA is sent, the GPIA will go into TACS (Talker Active State) as soon as the ATN line is set false (high). The presence of LACS or TACS states can be monitored in the Address Status Register (R2R). These conditions do not cause an MPU interrupt.

Dual Primary Address Mode

The Dual Primary Address Mode is similar to the previously mentioned primary address mode in that only one command need be sent from the controller to configure the GPIA as either a talker or listener (this mode can also be used with secondary addressing). It differs, however, in that this mode, as the name implies, allows the GPIA to respond in the same way to two adjacent addresses. For example, if the address programmed into the MC68488 is 01010, then the device responds to addressed commands with either 01010 or 01011 as the address portion of the command. A software routine must be used to determine which of the two possible addresses was actually received.

The GPIA is placed in the Dual Primary Address Mode by writing lsbe (bit 7, R4W) high. Thus, one additional step is required in the Primary Initialization Routine (Figure 4-10). Once the appropriate address is in GPIA, an OR operation of R4W information with hex 80 is one method of establishing the Dual Primary Address Mode.

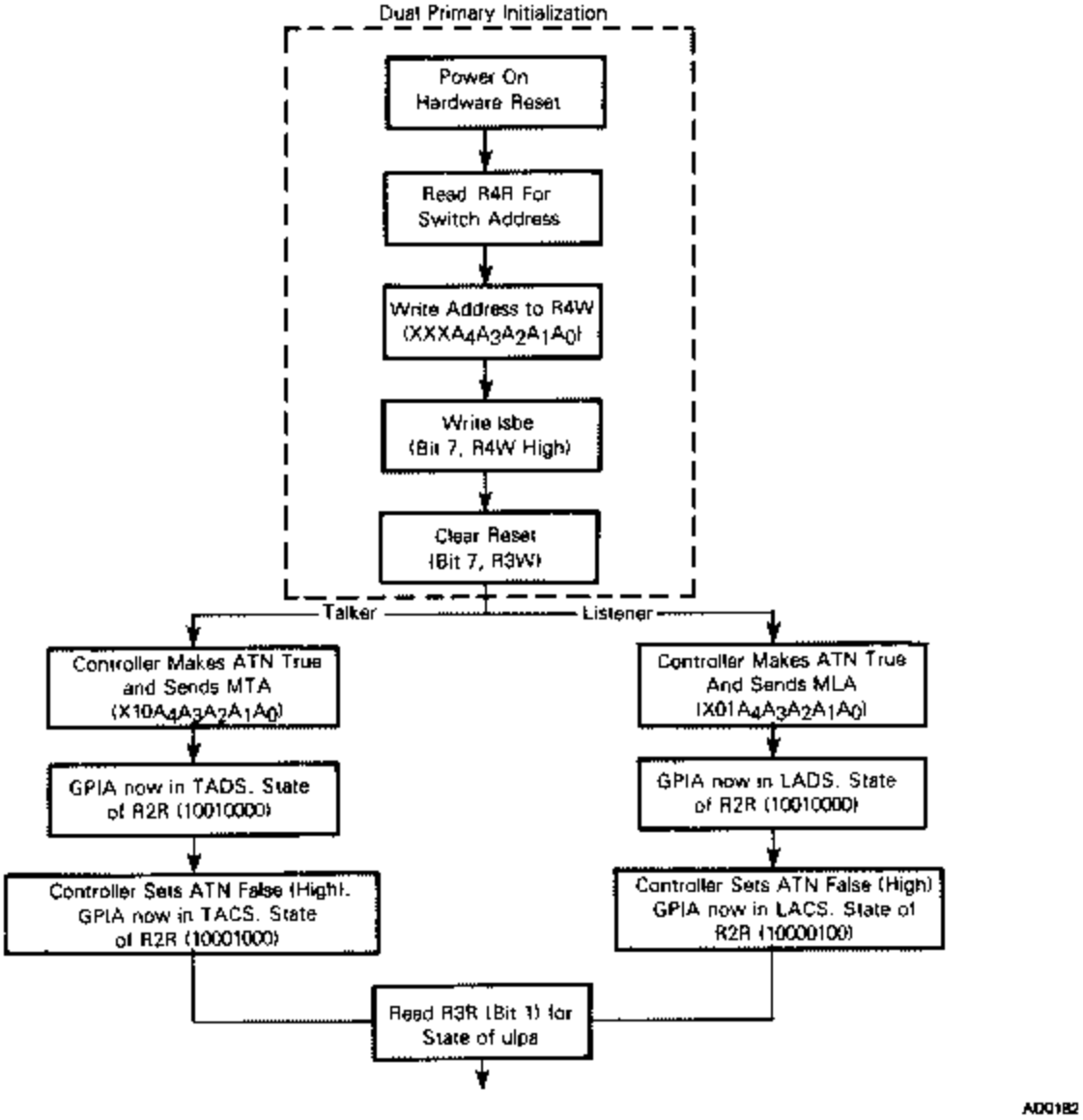


Figure 4-10. Dual Primary Talker/Listener Sequence

After the GPIA has been addressed as either a talker or listener, the MPU can determine which of the two addresses was sent by reading ulpa (bit 1, R3R). The ulpa bit always monitors the least significant bit of the addressed command that was sent to the GPIA by the controller. In the example mentioned earlier, if 0101[0] had been sent, the ulpa bit will be a 0 and if 0101[1] has been sent, ulpa is a 1.

Dual Primary Addressing provides a convenient method of partitioning a device into two sections. For instance, a voltmeter and ammeter could use the same GPIA (be considered one device on the GPIB) and yet respond to two different primary addresses.

Dual Addressing implies the use of two adjacent primary addresses and, as such, care should be taken when selecting the primary addresses for this mode. Address 30 (11110) should not be used because the Dual Address counterpart of decimal 30 is decimal 31 (11111). Since address 31 has the same bit code as that of either the Untalk or Unlisten Command this value is an invalid primary address for the IEEE-488 Standard system.

Secondary Addressing Mode

Secondary Addressing may be used to extend the number of individual instrument addresses in the system or allow individual devices to have multiple addresses for data routing. In this mode, the controller addresses a device to be a talker or listener by sending at least two commands. The first is the Primary Address followed by the Secondary Address(es). A software routine must be used to determine the validity of each secondary address.

The GPIA is placed in the Secondary Address Mode by writing apte (bit 0, R2W) high. This is accomplished in the initialization routine after software reset. Thus, the Secondary Initialization Routine uses the Primary Initialization Routine followed by one additional step (Figure 4-11).

For a talker device, the controller sends MTA just as for the Primary Address Mode or Dual Primary Address mode. Upon reception of this, the GPIA enters TPAS (Talker Primary Address State) and no MPU interrupt is generated. The handshake is automatically completed. The controller then sends a Secondary Command Group (SCG). This causes the APT bit in R0R to be set, flagging the MPU that a secondary command is to be read through the Command Pass-Through Register (R6R). The DAC handshake is also held off giving the MPU time to read the command and determine if it is proper, i.e., does the secondary address belong to this device (most likely a comparison is made in a "look-up table"). After the comparison is made, the MPU should do one of two things:

1. If the secondary address is not proper, the MPU should write a hex 10 to R3W releasing the handshake leaving the GPIA in TPAS. The device waits for another SCG.
2. If the Secondary address is proper, the MPU should write a hex 18 to R3W releasing the handshake. The device will now be ready to talk when ATN is released.

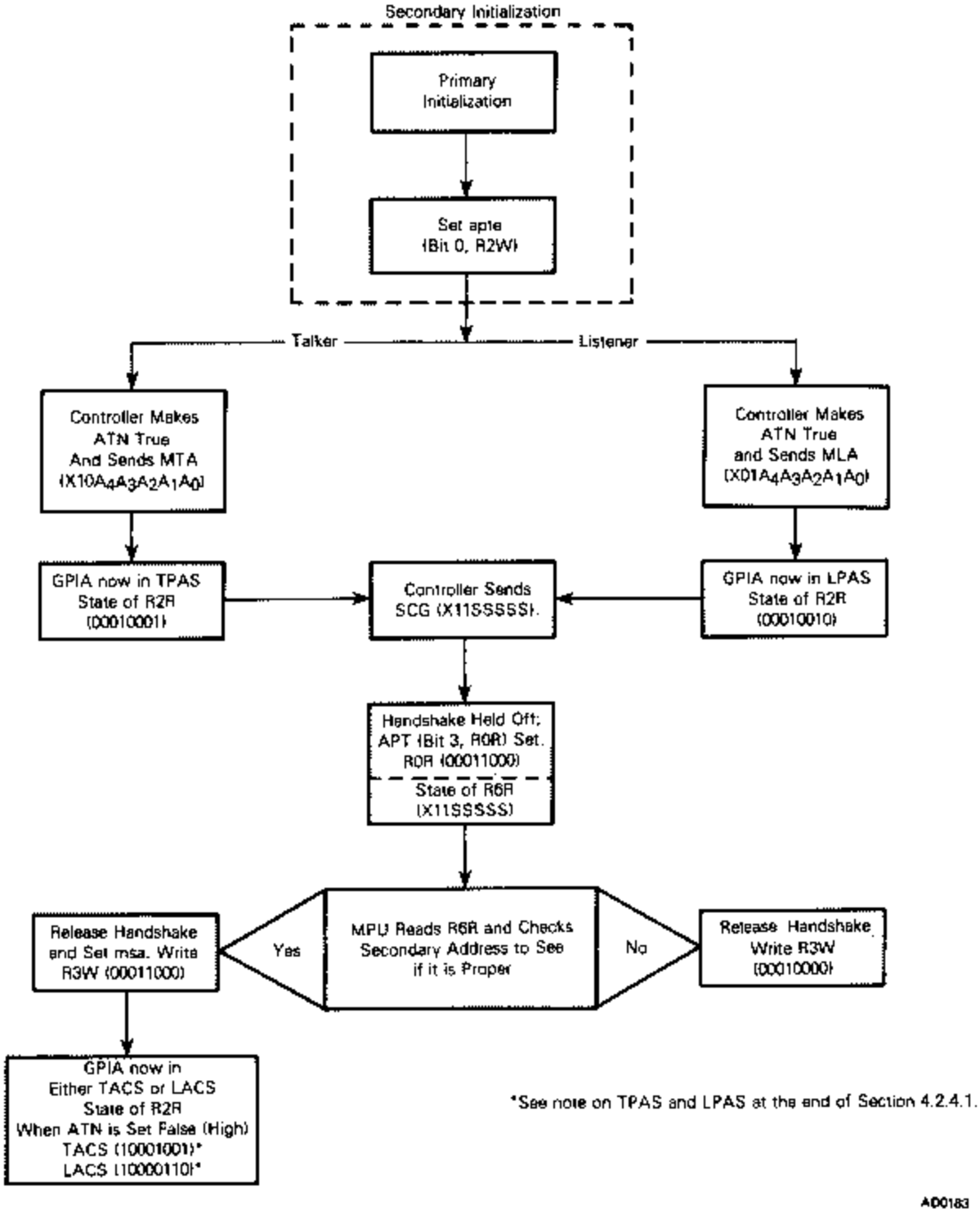


Figure 4-11. Secondary Addressing Sequence

The procedure for the listener mode is much the same. See Figure 4-11 for the Secondary Addressing procedures.

The IEEE-488 Standard protocol requires, in extended addressing, that the Secondary address follow the primary address. Should a GPIA (with apte = 1) receive its primary address, it enters either TPAS or LPAS. If while in either of these states the controller sends something other than the SCG, the GPIA returns back to its idle state and the entire process needs to be initiated again.

Secondary Addressing can be used for many purposes. A multimeter can receive specialized instructions through secondary addresses (e.g., function to be performed or range settings). Secondary addresses can be used to cluster instruments around one GPIA. In this configuration, all of the instruments would be treated as one device on the GPIB and have one primary address. The secondary address would then address a specific instrument or part of an instrument. The secondary address could also be used for routing specific data arrays.

NOTE

TPAS/LPAS status bits once set remains set until another Primary Command Group (PCG) message is sent over the GPIB. Thus, for example, if device #1 receives its secondary sequence for a listener state, the contents of R2R will be 10000[1]0 — LPAS is set. Should a Primary Command (e.g., another device listen address) be sent over the GPIB, the state of R2R in device #1 is 100001[0]0 — LPAS is reset.

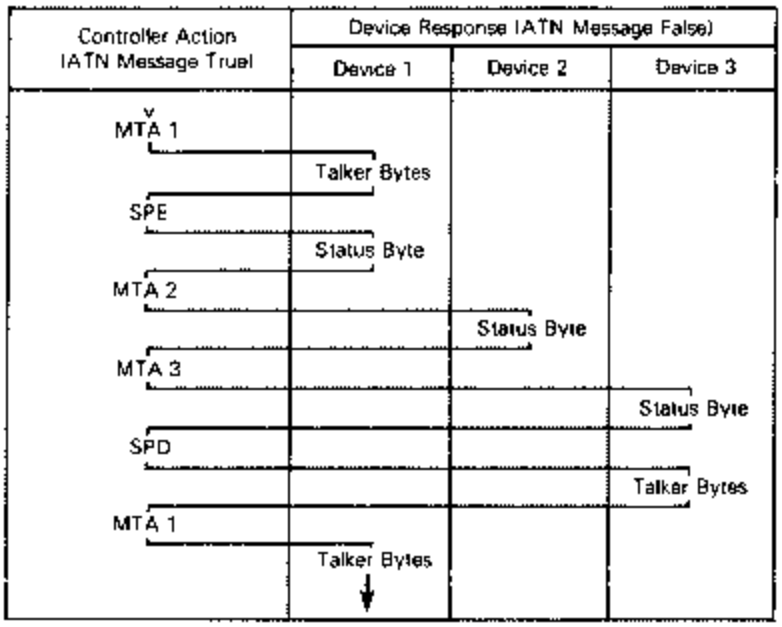
4.2.4.2 SERIAL POLL WITH SERVICE REQUEST. The service request (rsv) provides a device with the capability to alert the system controller that special status information is available from the device. The request for service may be asserted asynchronously from other activity on the interface bus. A device sends a service request by making the SRQ management line true (low). Devices can use this feature for reporting specific status conditions, e.g., out of paper, improper measuring connection, critical measurement results, etc., or any other time a device would like to alert the controller of a status situation. The Serial Poll Register (R5W) in the GPIA is used for sending a service request and reporting the status. Seven of the bits in this register are used for coding the specific status conditions while the remaining bit (bit 6) serves a dual role. First, when set it forces the SRQ line low. Second, this bit indicates to the controller that this is the device requesting service.

After the service request is detected, the controller obtains the status information by sending a Serial Poll Enable (SPE) command followed by the device talk address. This causes the GPIA to enter the Serial Poll Active State and the GPIA to place the contents of R5W on the GPIB data bus. It also causes the GPIA to release the $\overline{\text{SRQ}}$ line. If another device is also requesting service at the same time, the SRQ management line remains low. After the status information is obtained, the controller sends the Serial Poll Disable (SPD) command completing the Serial Poll procedure. However, there is only one service request line (SRQ), and there may be as many as 14 devices all capable of driving this line low. The controller must systematically determine which device is requesting service. It can do this in one of two ways.

1. The controller can send SPE and then sequentially make each device a talker. When made a talker, the addressed device places the contents of R5W (Serial Poll Register) on the GPIB data

bus. The controller can then check the status byte of each device. When it finds a status byte with bit 6 (DIO7 in GPIB convention) set, it knows that this device had requested service. If the SRQ line is not high at this time the controller knows that a second device is requesting service. Figure 4-12 shows the device/controller interaction. If device #1 is made a talker and the controller does not find bit 6 set, it will know that this device did not request service. It then needs to check the next device in its polling routine. The controller need not send the Untalk Command before proceding to device #2, since device #1 automatically returns to the Talker Idle State (TIDS) when it detects another device primary talk address on the GPIB. This prevents more than one talker from being on the bus at any one time. After the controller receives the status byte of the requesting device and the SRQ line is high, it can send SPD and reconfigure the bus.

2. The Serial Poll Procedure, as described above, can be a lengthy process. It could require the controller to poll 14 devices before finding the one requesting service. If the devices in the system have Parallel Poll capability, the efficiency of the Serial Poll procedure can be greatly increased. Both the Serial Poll Function (initiated through register R5W) and the Parallel Poll Function (established through register R6W) can be used together. If before initiating a service request (setting bit 6, R5W) the device set assigned Parallel Poll bit in R6W, the controller can narrow the possible choice of devices requesting service to 1 or 2. Thus, on detection of an SRQ, the controller would first perform a Parallel Poll, isolating which devices might be requesting service, and then perform a Serial Poll on this smaller group. If, for example, a Parallel Poll was conducted and the Parallel Poll byte showed bit 4 set high, the controller need only perform the Serial Poll procedure on the device(s) assigned bit 4. Since there are 8 Parallel Poll status bits and a maximum of 14 devices (15 total; 14+controller), there need only be a maximum of 2 devices assigned to any one Parallel Poll bit.

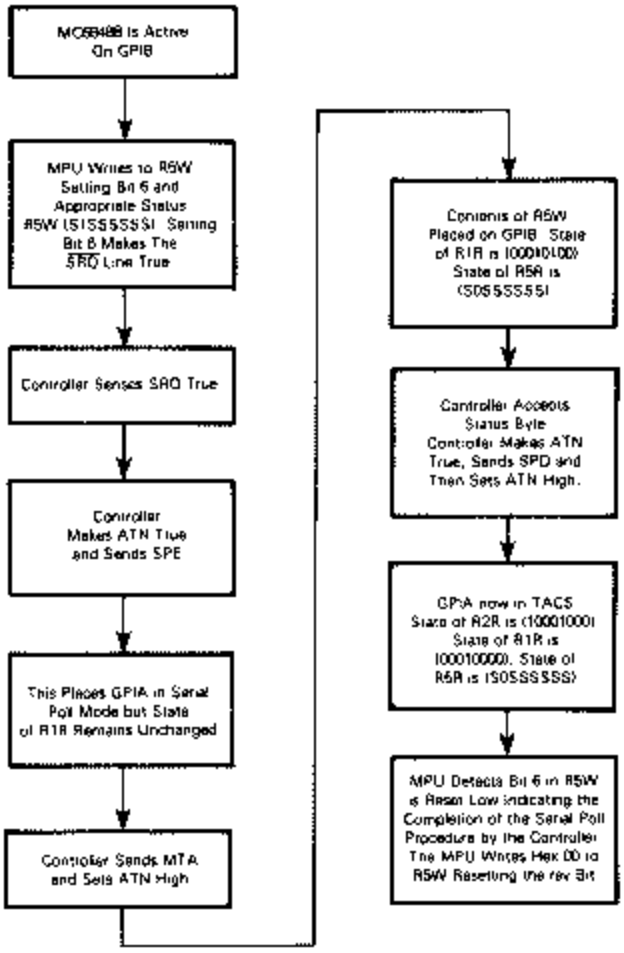


Note: At the end of the Serial Poll sequence if SPD is send and ATN set false (high), the last addressed talker will enter the Talker Active State (TACS). This may be undesirable in some applications. To avoid this situation the controller should send Untalk Command following SPD.

AD0184

Figure 4-12. Serial Poll Procedure

The MPU initiates a service request by writing rsv (bit 6, R5W) high in the GPIA. At the same time the appropriate code should be placed in the other 6 bits. Bit 6 being set causes the $\overline{SRQ}$ management line to go low. The GPIA will enter the Serial Poll Active State (SPAS) when it receives SPE and is an active talker. When it enters SPAS, the following occurs: the SPAS status bit (bit 2, R1R) is set, the CMD status bit (bit 2, R0R) is set, the $\overline{SRQ}$ line is asserted (low), the SRQ status bit (bit 6, R5R) is reset, and the contents of R5R are placed on the GPIB data bus. Figure 4-13 shows the GPIA Serial Poll Sequence.



AD0185

Figure 4-13. Serial Poll Sequence

When the GPIA enters SPAS, the SPAS status bit (R1R) is set. This in turn causes the CMD status bit in R0R to be set. In an interrupt driven system with the CMD and IRQ mask bits set, this causes an MPU interrupt. These status bits are not latched conditions and only monitor the current state of the GPIA. If the controller places the GPIA in SPAS (sends SPE and MTA), receives the Serial Poll status byte and removes the GPIA from SPAS (sends SPD) before the MPU reads the Interrupt Status register, the contents of this register shows hex 00. Since the MPU knows that this device issued the service request, it should check bit 6 of R5W if an MPU interrupt is generated but no status bit is set. If this bit is reset, the MPU will know the controller has performed a Serial Poll on it. However, the SRQ status bit being reset does not indicate that the status byte was accepted by the controller — that is, the handshake was completed. Rather it indicates that the GPIA has been placed in SPAS and that the status byte has been placed on the GPIB. To determine when the status byte was accepted, the MPU can monitor SPAS status bit. This bit is reset when the controller has removed the GPIA from the SPAS. Once in SPAS the controller must accept the Serial Poll byte before moving the device from SPAS unless it interrupts the handshake asynchronously. The rsv bit cannot be written low until the status byte has been accepted. The rsv bit should be written low as soon as the status byte has been accepted by the controller.

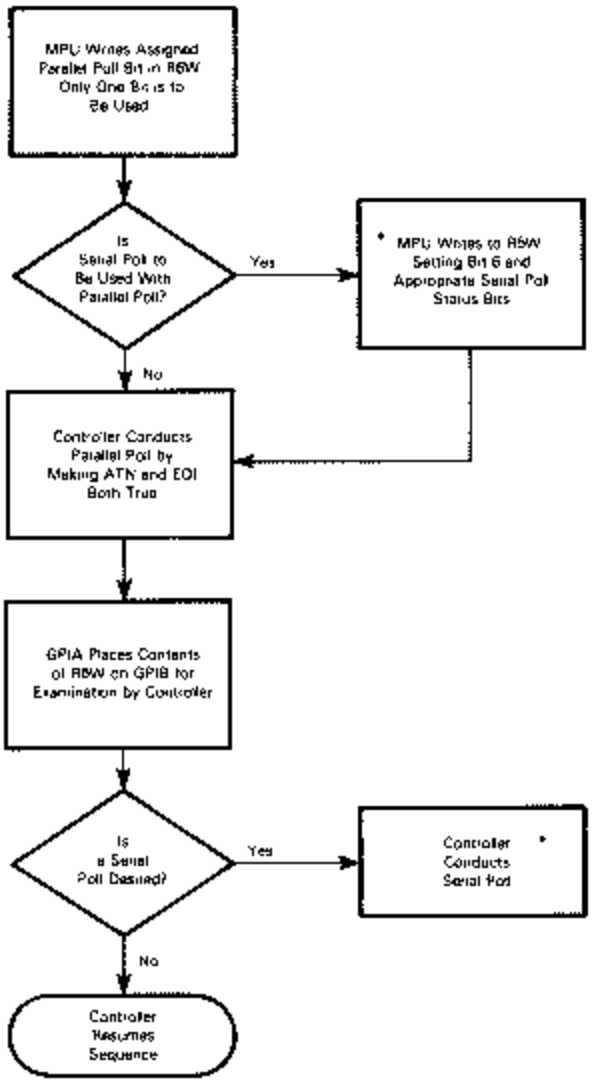
The GPIA uses the source handshake to send the Serial Poll status byte to the controller. The GPIA does this by placing the status byte on the GPIB, and when the controller makes RFD high, the GPIA will assert DAV true (low) and the handshake will take place according to the IEEE-488 Standard handshake protocol. If nba for the GPIA TACS function is false at this time, the GPIA will send this byte only once; i.e., the GPIA does not assert DAV true (low) a second time. If nba for the GPIA TACS function is true at this time, the GPIA sends this byte over and over provided the controller continually makes RFD true at the end of the handshake without reconfiguring the device, i.e., the GPIA in this situation deasserts $\overline{\text{DAV}}$ true each time it receives an RFD true from the controller. The only time nba can be true for TACS is if the device was an active talker prior to the Serial Poll sequence and the GPIA MPU had loaded a byte in R7W. Now if the controller synchronously takes over the bus before this byte is placed on the GPIB, the nba for TACS will be true.

NOTE

After a Serial poll has been conducted on the GPIA and the SRQ bit (bit 6, R5W = 0) is reset, the MPU must write the rsv (bit 6, R5W) low before another service request can be initiated.

4.2.4.3 PARALLEL POLL. Parallel Poll is a function where the controller simultaneously seeks one bit of status information from each device. Each device is assigned one of eight bit locations by the controller during a Parallel Poll configure initialization. The MPU can set (or reset depending on the convention used) the appropriate GPIA status bit by writing the assigned bit in the Parallel Poll Register (R6W). The Parallel Poll sequence is shown in Figure 4-14. Unlike Serial Poll, the device shown does not request service through Parallel polling. This is a controller initiated sequence and it is beyond the scope of th IEEE-488 Standard to specify when a Parallel Poll is optional. In some systems the controller might conduct a Parallel Poll each time it reconfigures the bus, and in others the Parallel Poll procedure might not be used at all. As mentioned in Section 4.2.4.2, a Parallel Poll sequence can be used in conjunction with a service request to increase the efficiency of

the Serial Poll procedure. The controller conducts a Parallel Poll by asserting both the ATN and EOI lines low (true). When this occurs, each active device (the GPIA not in reset) places the contents of R6W on the GPIB data bus.

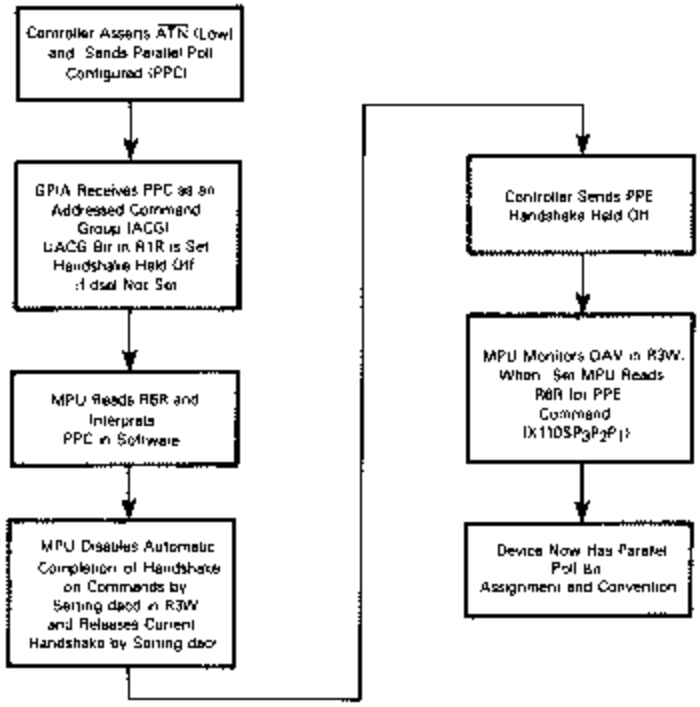


*These steps will be included when a parallel poll and serial poll are used together.

Figure 4-14. Parallel Poll Sequence

The controller configures a device by first sending the Parallel Poll Configure (PPC) command. The GPIA treats this as an unidentified Address Command Group and sets the UACG bit in R1R. The MPU reads the command through R6R and interprets it in software. In normal IEEE-488 Standard protocol, the controller follows this command by sending Parallel Poll Enable (PPE) or Parallel Poll Disable (PPD). (The PPC command alerts a device to establish Parallel Poll capability and PPE/PPD provides bit assignment and convention). Both PPE and PPD are part of the Secondary Command Group. The Parallel Poll configure procedure involves the reception of two commands. Since software routine must be used to analyze these commands, the handshake must be held off on the reception of both. The handshake is automatically held off on the first one (PPC) but the GPIA

must be programmed after receiving PPC to hold off the handshake on the second (PPE/PPD). After receiving PPC, the MPU must set the dacd bit in R3W holding off the handshake on reception of all commands. Now the MPU can monitor the DAV status bit in R3W for the occurrence of valid data (in this case PPE or PPD). When this bit is set, the MPU should read R6R and interpret the received data byte in software. The handshake is released by setting dacr in R3W (see Section 3.3.1.1 for details). Figure 4-15 shows an example of how the GPIA receives its Parallel Poll bit assignments. Note that the IEEE-488 Standard does not require the PPE or PPD command to immediately follow PPC and thus it is possible for the controller to send other commands in between. In such a case, the MPU may need to read more than one byte through R6R (releasing the handshake after each byte) before receiving the appropriate bit assignment. It is recommended, however, that the controller send either PPE or PPD immediately after sending PPC. Otherwise, circumstances could arise where PPE/PPD would be mistakenly interpreted as a secondary address. For example, suppose PPC is sent to device #1 and this is followed by the controller sending device #2 primary address and device #2 in the extended address mode. Since PPE and PPD are part of the Secondary Address Group, confusion could arise when the secondary address is sent to device #2. Device #1 might interpret this secondary address as its Parallel Poll bit assignment.



AD0187

Figure 4-15. Parallel Poll Assignment Sequence

The controller sends PPC followed by PPE/PPD to establish Parallel Poll Capability for a device. To remove this capability, the controller sends Parallel Poll Unconfigure (PPU). The GPIA will treat this as an Unidentified Address Command Group and set the UACG bit in R1R. As with PPC, the MPU will read this command through R6R. The PPU command does not have another command associated with it and thus dacd does not need to be set.

4.2.4.4 REMOTE/LOCAL. In many systems, the option is provided to configure an instrument (e.g. voltage range, operating frequency, amplitude settings, etc.) from either the instrument front panel (local mode) or the controller (remote mode). The controller can place a device (instrument) in either of these two modes by sending commands like Remote Enable (REN) or Go-To-Local (GTL). In addition, the Local Lockout Command (LLO) is used to place the GPIA in RWLS once the interface function is in REMS. The GPIA can move from RWLS to LWLS if the GTL command is received. However, the GPIA does not respond to the Return To Local (rtl) local command while in either RLWS or LWLS. Thus LLO prevents the GPIA from responding to rtl as long as the REN line remains low.

The GPIA provides three status bits (bits 3, 5, and 6-R1R) for monitoring the Remote/Local modes. It also provides the return to local (rtl) local message for placing the device in the local state. Section 3.2.5 describes the interaction of these bits. Figure 4-16 shows the sequence for placing the GPIA in either the Remote (REN) or Remote with Local Lock (LWLS) states.

4.2.4.5 DEVICE CLEAR. The controller can instruct a device to clear itself by sending either the Device Clear (DCL) or selected Device Clear (SDC) commands. A controller might use commands like these to place devices in their power-on state. If DCL is sent, all active GPIAs (GPIAs not in software reset) will receive this command. Only addressed listeners will receive the SDC command.

When the GPIA is placed in the Device Clear Active State by these commands, it sets bit 1 in R1R, which in turn flags the MPU through the command interrupt bit (CMD). The MPU detects the presence of this state by reading R1R and takes appropriate action according to its Device Clear software routine. The GPIA response to these commands can be found in Section 3.2.6. Figure 4-17 shows the Device Clear sequence.

4.2.4.6 DEVICE TRIGGER. The Device Trigger allows an instrument or group of instruments to be triggered from the GPIB. The controller can trigger all addressed instruments simultaneously by sending the Group Execute Trigger (GET) command. The MPU can, in addition, trigger its own instrument(s) by sending the fget local command to its GPIA. Section 3.2.7 explains the GPIA response to these commands. Figure 4-18 shows the Device Trigger Sequence.

4.2.5 DMA Operation

DMA is normally used to transfer blocks of data at high data rates. The high data rates are achieved by removing the MPU (and its associated software overhead) from the system and allowing a DMA controller (which is much more efficient at moving data between peripheral devices and memory) to control the data transfer. Note in the context of the IEEE-488 Standard that DMA does not refer to the transfer on the GPIB as data is always transferred on the GPIB according to the 3-wire asynchronous handshake previously described. DMA here refers to how data is transferred from a talker device memory to its GPIA or from a listener device GPIA to its memory. If a device can transfer data between its GPIA and its memory at a faster rate, then it has increased its data rate capability on the GPIB. However, increasing a device capability does not insure that in a system configuration data is transferred at a faster rate. The GPIB data rate is always governed by the slowest listener in

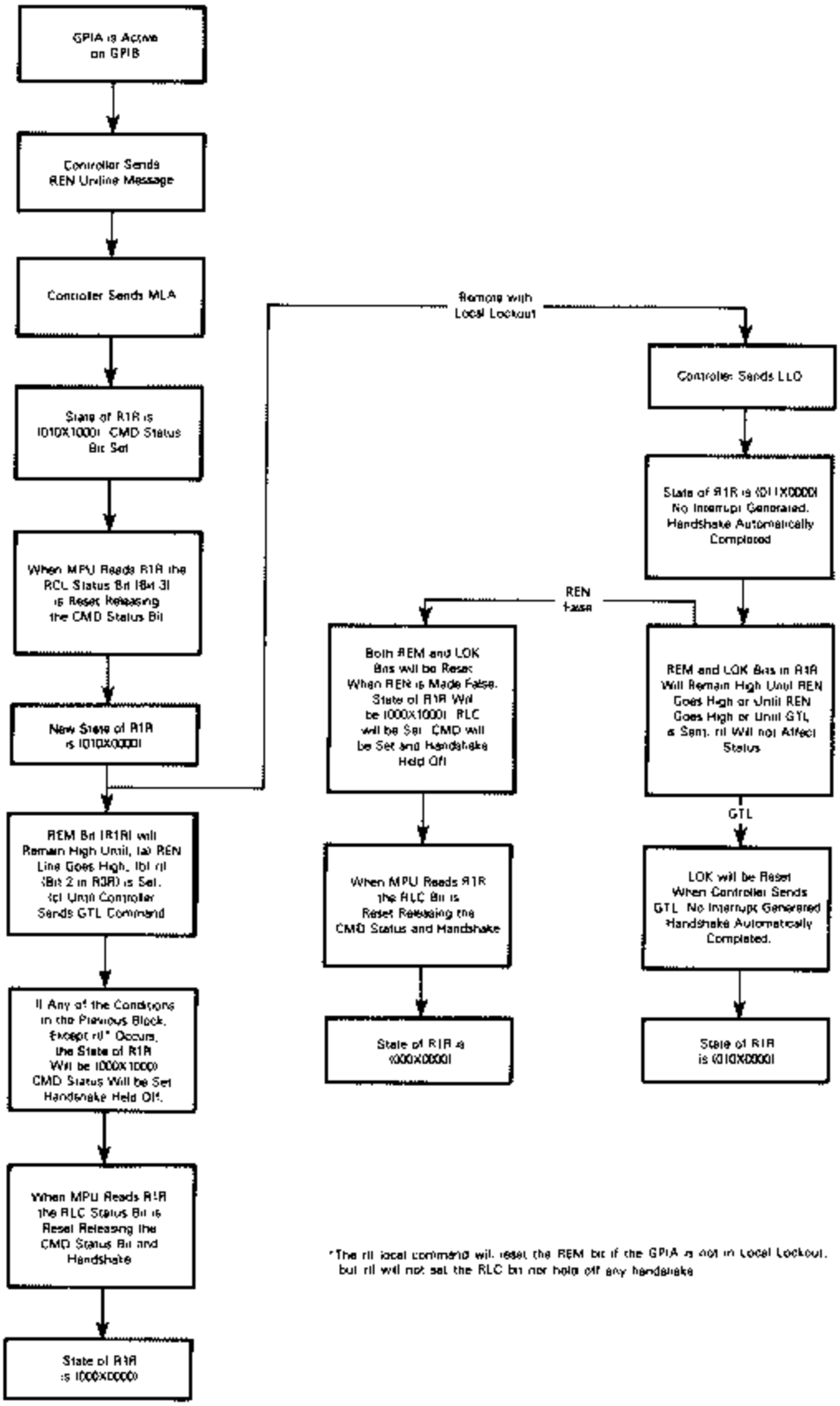


Figure 4-16. Remote/Remote-With-Local-Lockout Sequence

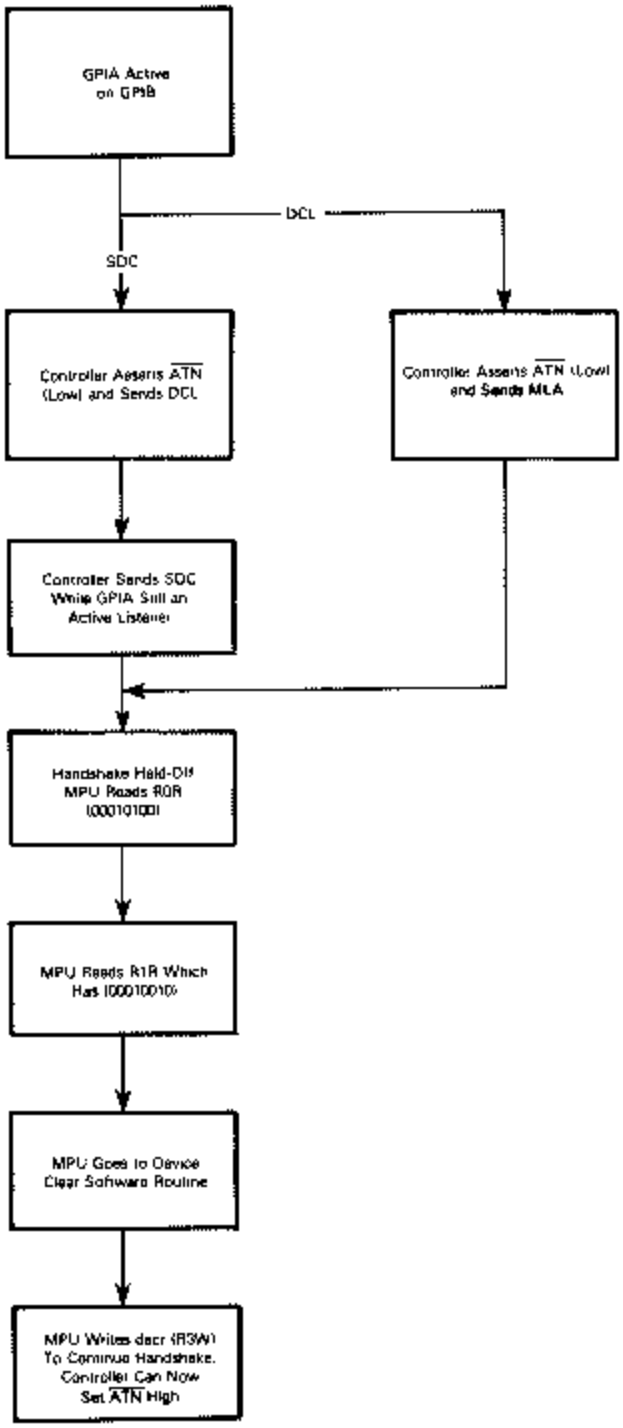
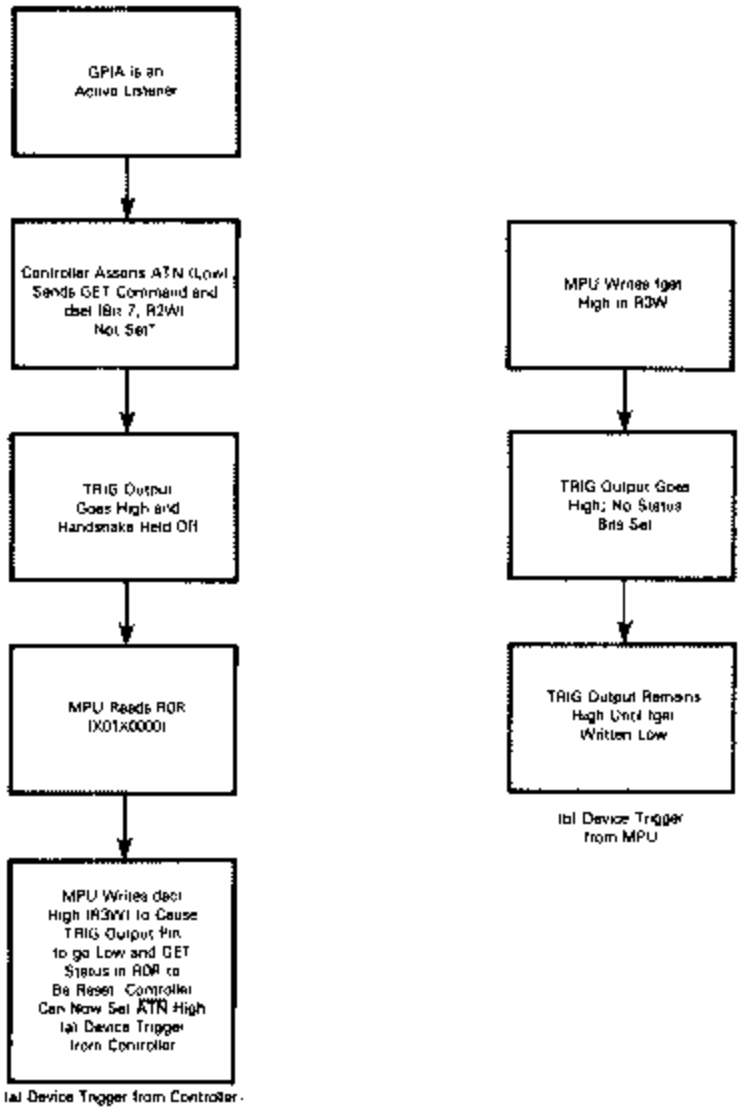


Figure 4-17. Device Clear Sequence



AD0190

Figure 4-18. Device Trigger Sequence

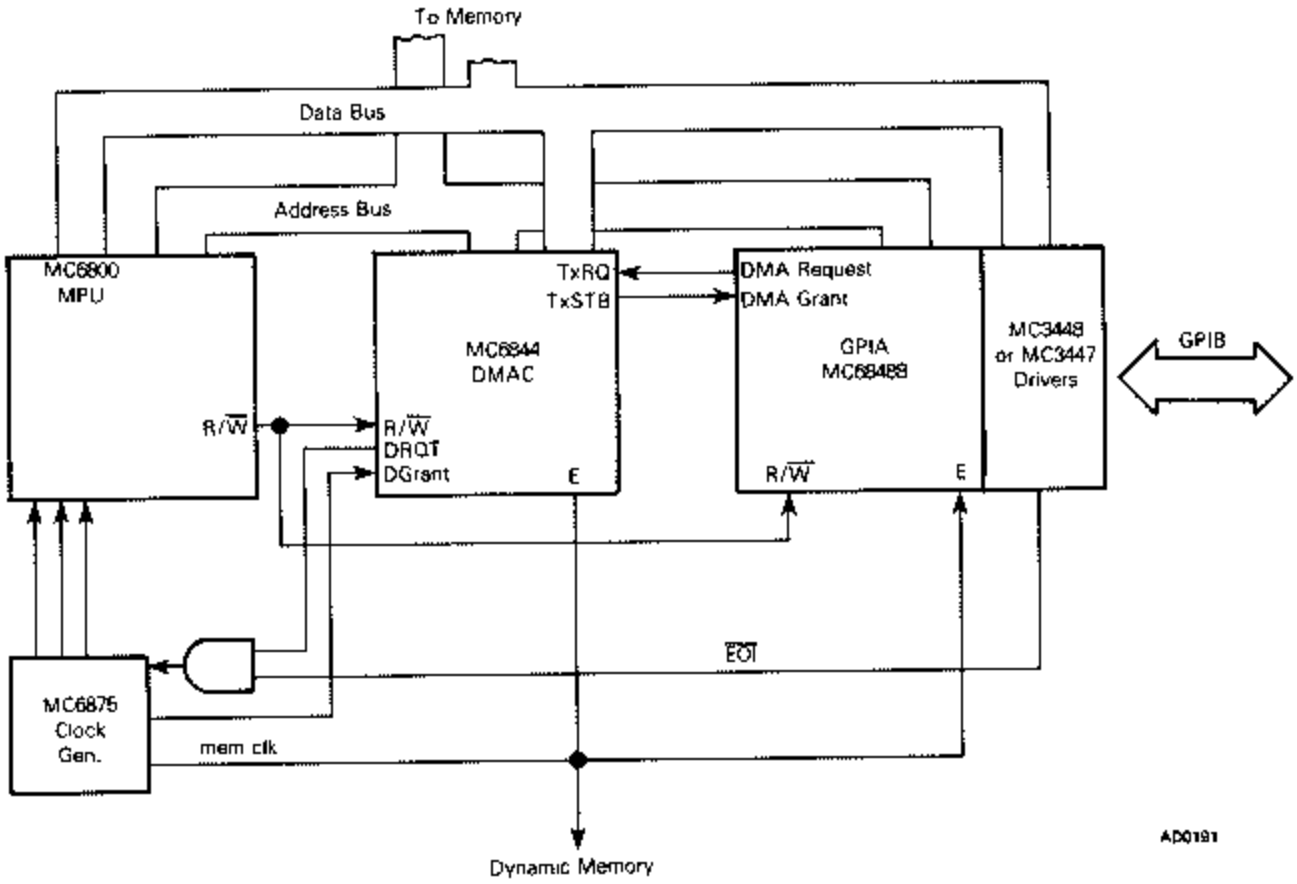
the system. High data rate capability listeners can only receive data as fast as the slowest active listener in the system.

There are basically two methods used for transferring data in DMA mode. They are:

1. Halt Burst Mode — In this mode, the MPU is halted and removed from the device system while a block of data is transferred between memory and the GPIA. The DMA controller manages the control lines (e.g., R/W, address lines, etc.) and keeps track of how many bytes have been transferred, returning control back to the MPU when the last byte has been sent. Thus, if the DMA controller has been programmed for a 16K byte transfer, the MPU is removed from the system at the beginning and is not brought back on line until all 16K bytes are transferred. The highest data rates are achieved in this mode of operation at the expense of altering the MPU for long periods of time while large blocks of data are transferred.

2. Three-state steal (cycle stealing) — In this mode, the MPU is not halted nor removed from the system for an extended length of time. Rather, the MPU clock is stretched holding the MPU temporarily suspended while the DMA controller transfers one byte of data. At the end of this one byte transfer, control is given back to the MPU. If a block of data is being transferred, the MPU is placed back in the system between each transfer for at least one MPU clock cycle. This method of DMA operation is slower than the halt-burst mode but does not cause the MPU to relinquish control for long periods of time.

Figure 4-19 shows a block diagram of a DMA system using Motorola's DMA Controller (MC6844) in the three-state steal (cycle steal) mode. After a power-on reset, the MPU must initialize both the DMAC and the GPIA. Initialization of the GPIA is as previously discussed with one exception. If DMA is to be the mode of transfer, the GPIA should not be initialized to cause an $\overline{IRQ}$ when BO or BI occur in R0R (these mask bits must be low). Rather than causing an $\overline{IRQ}$ to the MPU, when data is to be read from R7R (listener mode) or written to R7W (talker mode), the GPIA flags the DMAC through its DMA Request line (pin 15). The DMA request line from the GPIA will produce a request signal if either the BO status bit or BI status bit is set in R0R. See Section 3.1.1.3 for details.



Note: Only these signal lines unique to DMA transfer are indicated in this diagram.

Figure 4-19. DMA TSC Steal Block Diagram

In some cases, a device might use DMA when addressed as a listener, but not as a talker or vice versa. In such a case, only the interrupt mask bit pertaining to the DMA mode of operation need be inhibited.

During the initialization process, the MPU programs the DMA controller for the desired operation. Internal registers on the DMA controller need to contain such information as direction of data transfer (memory-to-GPIA for talker mode and GPIA-to-memory for listener mode), which one of the four channels is to be used for the transfer, what memory address is to be accessed, number of bytes to be transferred, method of DMA transfer, etc. See Motorola's MC6844 data sheet for details.

The GPIA inititates a DMA transfer request. In the listener mode, when a byte has been received from the addressed talker, the BI bit will be set and the DMA request line goes high. In the talker mode, when the Data Out Register is empty, the BO bit is set and the DMA Request line goes high. The DMA Request line, when high, issues a request to the DMA controller. The DMAC in turn issues a transfer request (DRQT) to the MPU system (in this case the MC6875 clock). The MC6875 clock responds with a DMA Grant (DGRANT) back to the DMAC. While the MC6875 is issuing DGRANT, it is holding its clock outputs to the MPU in an idle state (stretching the clock to the MPU). The Memory Clock output (memclk) from the MC6875 continues to run at its normal rate during the DMA stretch period and is the clock used by the DMAC and GPIA, thus, the clock to the DMAC and GPIA is never interfered with. When the DMAC receives a grant back from the clock, it in turn issues a DMA Grant (high state) to the GPIA. The DMA Grant line being high results in the following to the GPIA:

1. The DMA Request line is reset.
2. Register 7 of the GPIA is automatically selected irrespective of the conditions on RS0, RS1, and RS2. The $\overline{CS}$ line must be high.
3. The R/$\overline{W}$ line is inverted and becomes $\overline{R}$/W. This eliminates the need of an inverter when using DMA. The DMAC drives R/$\overline{W}$ high for a read of memory and low for a write of memory. But in DMA mode a read of memory means a write to the GPIA, and a write to memory means a read of the GPIA. This condition is automatically processed by the GPIB.

The steps in the three-state mode are summarized as follows:

1. GPIA initiates transfer request to DMAC.
2. DMAC in turn issues a request to MPU clock.
3. MPU clock stretches clock and then issues grant to DMAC.
4. DMAC issues grant to GPIA which causes transfer on the concurrent E clock cycle.
5. DMAC removes request to clock.
6. Clock removes grant to DMAC and releases stretch to MPU.

After the byte has been transferred, the DMA Grant line to the DMAC and, in turn, to the GPIA is brought low and the clock to the MPU begins running. Each succeeding byte is transferred by the same method. The MPU is brought back on line after each byte transfer. Thus, should the GPIB system controller interrupt the GPIB (e.g., sending IFC) during a block transfer, the MPU can be immediately notified.

When in the listener mode and receiving a data string, the $\overline{EOI}$ management line can be used by the Talker to indicate, to all active listeners, that the next byte received is the final byte. The MPU in the listener device is notified of the final byte by the END status bit in R0R. This bit and the $\overline{EOI}$ line is reset when all listeners have accepted the final data byte. If the $\overline{EOI}$ option is used, it is important not to read the final data byte by DMA means; because once this data byte is read, the $\overline{EOI}$ line can go inactive, causing the END status bit to be reset. To avoid this problem, the $\overline{EOI}$ management line should be allowed to gate off the DRQT signal (transfer request) to the MPU system. The END status bit can then be used to interrupt the MPU, and the system interrupt handler can receive the final byte through a non DMA mode. An advantage of using the $\overline{EOI}$ line is that the Listener System need not know the exact number of data bytes to be transferred. The DMAC byte count register (BCR) need only be loaded with some large number (must be larger than the actual number of bytes to be transferred) and then allow the $\overline{EOI}$ line to signal the end of the transfer.

The halt-burst mode of transfer requires that the DRQH line from the DMAC be connected to the MC6800 halt line. The DGRANT line back to the DMAC line comes from the BA (Bus Available) line of the MC6800. Care should be taken when using a device connected to the GPIB in the halt-burst mode. As mentioned earlier, once a halt-burst mode begins, the MPU is halted (in this mode the MPU is completely removed from the system and will not even respond to an IRQ) for the entire block transfer. If during the transfer, the GPIB system controller sends an IFC, the GPIA is placed in an idle state, but the MPU will remain halted since the full block has not been transferred (DMAC byte count register is not zero). To prevent a device from hanging-up in this mode, extra gating circuitry should be used in the MPU system, providing a means to release the halt line to the MPU from the GPIB system. For example, the IFC and/or ATN lines from the GPIB could be gated to release the halt condition on the MPU. Thus, if the GPIB controller stops the block data transfer it can also bring the MPU out of the halt state. Also, if the $\overline{EOI}$ line were able to interrupt the MPU halt condition (as described for the three-state steal mode above), the DMAC, when the device is a listener, would not have to be programmed for the exact number of bytes to be received, since in many cases, this is not known.

4.3 EXAMPLE PROGRAMS

This section shows two programs for the talker and listener routines. These programs are for illustrative purposes only and are not intended to necessarily be the most efficient or practical software for any given application. Table 4-1 shows the MPU address memory map for the GPIA used for these programs.

Table 4-1. MPU Address Map for GPIA

Hexadecimal Address	MC68488 Registers (R/$\overline{W}$)
$5000	Interrupt Status/Interrupt Mask
$5001	Command Status/ –
$5002	Address Status/Address Mode
$5003	Auxiliary Command/Auxiliary Command
$5004	Address Switch/Address
$5005	Serial Poll/Serial Poll
$5006	Command Pass-Through/Parallel Poll
$5007	Data In/Data Out

4.3.1 Basic Talker Software

```
                                  Assume the MC68488 is at address location $5000.
        LDAA    $5004     READ the device address on the ADDRESS SWITCHES.
        STAA    $5004     WRITE the address into the ADDRESS REGISTER.
                          NOTE: Assume the device address was $0A. (AD5-AD1 corresponds
                          to 01010, respectively)
        LDAA    #$00      LOAD ACC A with zeros.
        STAA    $5003     This clears the reset bit.
        STAA    $5000     Mask all interrupts (if desired) in the INTERRUPT MASK
                          REGISTER.
        STAA    $5002     Select no special features in the ADDRESS MODE REGISTER.
                          NOTE: At this time the controller will address the device to TALK in
                          the following manner: ENABLE ATN and send MTA (My Talk Ad-
                          dress) on the DIO1-8 lines which would be X1001010 ($4A). Now
                          DISABLE ATN, A READ of $5002 ADDRESS STATUS
                          REGISTER will show: $88 ma (bit 7) and TACS (bit 3) will be set
                          HIGH. At this time the device is ready to TALK, BO (bit 6) of the IN-
                          TERRUPT STATUS REGISTER will be HIGH. Writing a byte to
                          $5007 DATA OUT REGISTER will reset BO to ZERO. BO is set
                          HIGH when the device(s) listening accept(s) the data. A possible soft-
                          ware approach to output five bytes of data from memory pointed to
                          by the INDEX REGISTER to the IEEE-488 Standard bus is as
                          follows:
        LDAB    #$05      Counter
LOOP    LDAA    $5000     LOAD ACC A with contents of INTERRUPT STATUS REGISTER
        CMPA    #$50      Check for Bit (BO) to be set.
        BNE     LOOP      If BO is zero, keep checking.
        DECB              Decrement counter.
        BEQ     LOOP 1    If last byte go set feoi control bit
        LDAA    0, X      Get a byte of data.
        STAA    $5007     Output to DATA OUT REGISTER and into bus.
        INX               increment pointer
        BRA     LOOP      Go check INTERRUPT STATUS REGISTER for another BO.
LOOP 1  LDAA    #$20      LOAD ACCA A with hex 20
        STAA    $5003     Set feoi (this precedes last byte of data).
        LDAA    0, X      Remove last byte from buffer.
        STAA    $5007     Write last byte to DATA OUT REGISTER and onto bus.
        RTS               End of Subroutine.
```

4.3.2 Basic Listener Software

```
                                  Assume the MC68488 is at address location $5000.
        LDAA    $5004     READ the device address on the ADDRESS SWITCHES.
        STAA    $5004     WRITE the address into the ADDRESS REGISTER.
                          NOTE: Assume the device address is $06 (AD5-AD1 corresponds to
                          00110, respectively).
        LDAA    #$00      LOAD ACC A with zeros.
        STAA    $5003     This clears the reset bit.
        STAA    $5000     Mask all interrupts (if desired) in the INTERRUPT MASK
                          REGISTER.
        STAA    $5002     Select no special features in the ADDRESS MODE REGISTER.
                          NOTE: At this time controller will address the device to LISTEN in
                          the following manner:
                            ENABLE ATN and send MLA (My Listen Address) on the DIO1-8
                          lines which would be X0100110 ($26). Now DISABLE ATN. A
                          READ of $5002 ADDRESS STATUS REGISTER will show $84 ma
                          (bit 7) and LACS (bit 2) will be set high. At this time the device is
                          ready to LISTEN.
                            BI (Bit 0) of the INTERRUPT STATUS REGISTER will be
                          LOW. BI will go HIGH to indicate that a data byte is available in the
                          DATA-IN REGISTER at $5007. Reading the DATA-IN REGISTER
                          will reset BI (Bit 0). A possible software approach could be as
                          follows: Accept data from the IEEE-488 Standard bus to a memory
                          buffer pointed to by INDEX REGISTER.
LOOP 1  LDAA    $5000     Load ACC A with contents of INTERRUPT STATUS REGISTER.
        TAPP              Transfers ACC A contents to CONDITION CODE REGISTER.
        BCC     LOOP 1    LOOP until carry bit is set. This indicates BI is set in R0R.
        BVS     LOOP 2    BRANCH to LOOP 2 if overflow is set, indicating END, bit 1 , of
                          R0R has set (i.e., TALKER has sent EOI).
        LDAA    $5007     LOAD DATA-IN REGISTER into ACC A. This resets bit BI.
        STAA    0, X      STORE the data byte in the buffer.
        INX               Increment pointer.
        BRA     LOOP 1    BRANCH back to LOOP 1 and check to see if BI is set.
        INX               Increment pointer.
LOOP 2  LDAA    $5007     Get the last byte from the DATA-IN REGISTER.
        STAA    0, X      Put last byte in the buffer.
        RTS               End of Subroutine.
```

4.4 A/D-D/A SYSTEM EXAMPLE

Two devices are discussed showing how to use the MC68488 in a GPIB compatible instrument. An Analog-to-Digital (A/D) system describes a device configured as a talker, and a Digital-to-Analog (D/A) sytem describes a device configured as a listener.

In both of these systems, the MPU initializes the MC68488 as discussed earlier in this chapter. Once initialized, the controller can configure the GPIB. Figure 4-20 shows an example of the controller configuration procedure.

The system descriptions and software flow charts are for a minimum system. As such, they do not handle universal commands or special features like Parallel or Serial Poll. They are only concerned with the basic talker or listener functions as pertain to the respective circuitry.

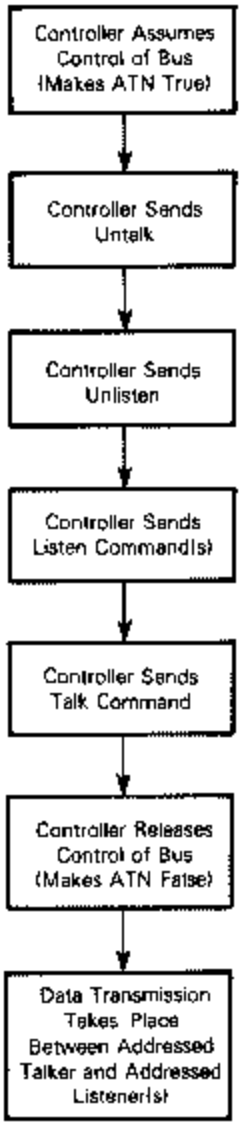


Figure 4-20. System Configuration By Controller

4.4.1 A/D System

The A/D system converts the analog signal at any one of six inputs to an 8-bit digital word and sends it over the GPIB. The six-channel design in Figure 4-21 is built around an MC6802 acting as the MPU, along with other ICs that are responsible for controlling the A/D converter and placing data obtained from the A/D to the bus. The interface handles all the handshaking protocol associated with the IEEE-488 Standard and places the desired data on the bus.

Two types of data are sent to the addressed listeners. First, the A/D channel number is sent; i.e., the first data byte received indicates the channel with which the following data byte is associated. Second, a data byte representing the digitized analog signal is sent. Thus, the addressed listeners will receive data formatted in pairs (address of A/D channel followed by that channel's converted analog data).

The A/D converter is a CMOS unit (MC14443), which receives the appropriate multiplexed address from a Peripheral Interface Adapter (PIA). Three output lines from a PIA in turn supply the analog multiplexer address. For simplicity, assume that the input voltages to the A/D are constant during the conversion cycle.

The A/D converts voltages into pulses using a single-slope technique. Pulse width is determined by the unknown voltage plus an offset voltage (which is considered constant during conversion).

The rate at which the ramp goes toward zero is fixed by the reference current and the holding capacitor (slope $= I_R/C$). So the time it takes to reach zero is proportional to the unknown voltage. After conversion, the A/D stops the timing process with an output to the PIA, which in turn interrupts the MPU.

With the MPU set up in a software timing loop, timing starts when the MPU gives the A/D starting pulse and stops when the A/D sends back an interrupt request ($\overline{IRQ}$). The equivalent converted voltage then resides in the MPU counting registers.

To initialize the A/D system as a talker, the MPU must assign an address to the MC68488. This address may be read from switches or may be assigned in the software. The controller follows procedures indicated in Figure 4-20 to assign the A/D GPIA as a talker function. To start the conversion process, the MPU must set up registers which are used to time out the A/D processor. The MPU then supplies the PIA with a code to issue a ramp start pulse to the A/D. The block diagram of the software necessary for the conversion process is indicated in Figure 4-22. The blocks numbered 1 through 7 are used for the actual conversion process. Blocks 10 through 18 are used to transmit the converted data to the bus and assign the appropriate address associated with the data.

In Figure 4-22, block 1 indicates that the counter registers in the MPU and the registers in the GPIA and PIA must be initialized. The PIA must be initialized so that four lines are output lines and one line can be used as an interrupt. Blocks 2, 3, 4, and 5 are the procedures used to establish ground and voltage references in equivalent times T1 and T2. To measure each time, in the conversion sequence, a software counting loop is set up in the MPU. After each time measurement, the counting registers are reset. The times T1 and T2 need only be measured once and stored, assuming the

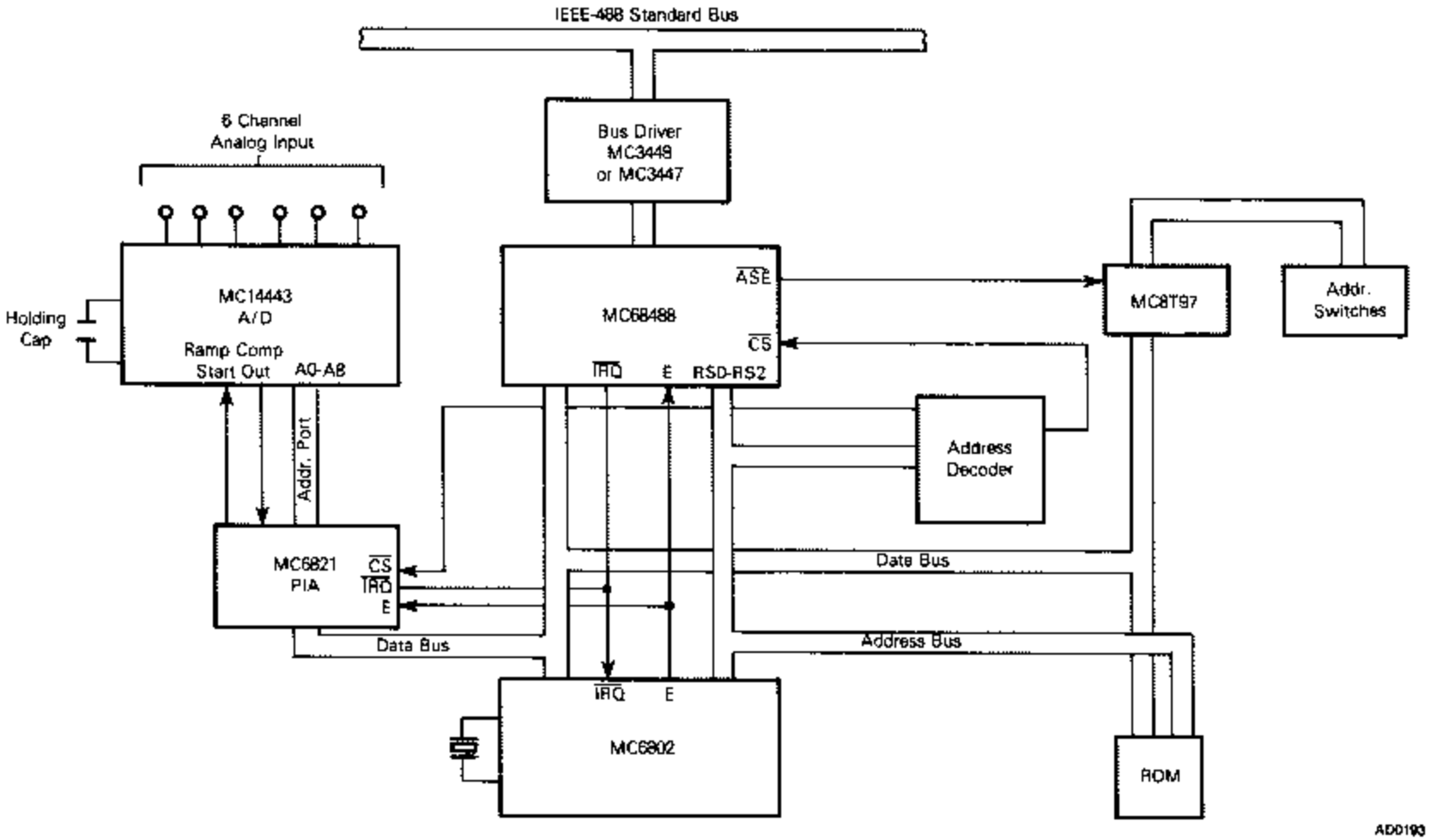
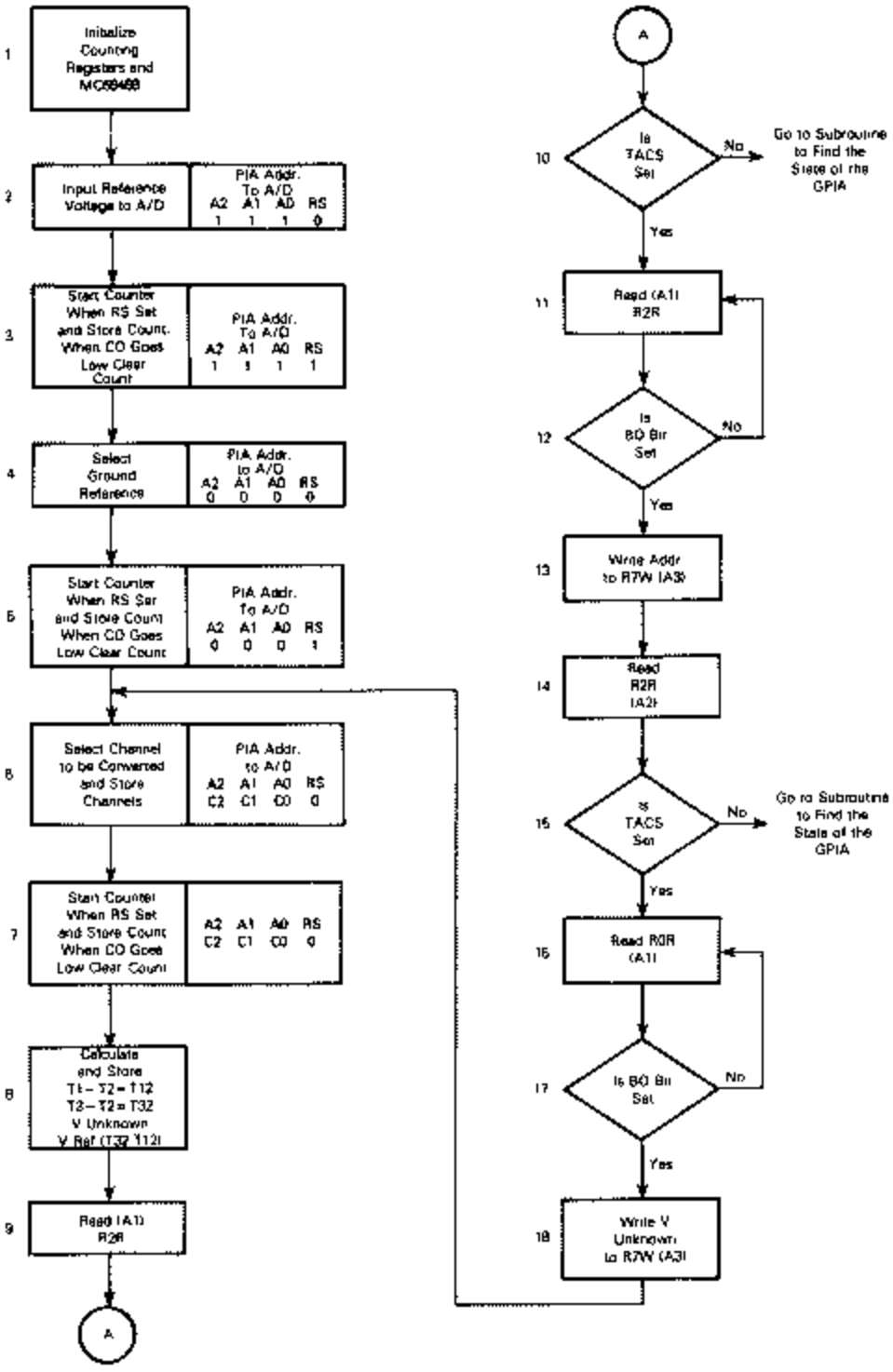


Figure 4-21. Block Diagram of A/D-GPIB System

references are stable and there are no temperature drifts over the measurement sequence. The conversion sequence of the unknown voltage occurs in blocks 6, 7 and 8. The analog channel is selected by selecting the proper channel address C0, C1, C2 and the time T3 is measured. The channel address is stored for later use. Block 8 of the conversion sequence is the scaling of the unknown voltage. It should be noted that the scaling can be greatly simplified by properly choosing T12 and V_{ref}. Then it is only necessary to calculate T32. Blocks 11, 12, 13, 16, 17 and 18 are used in conjunction with the GPIA to apply the data and the channel address to the IEEE-488 Standard bus. The analog channel number is written to the GPIB first and after it is accepted by the listener the data is sent. When the data is accepted, the process is repeated (block 18). Blocks 9, 10, 14 and 15 are used to check on the status of the GPIA.



AD0184

Figure 4-22. Software Flow Chart for A/D System

4.4.2 D/A System

The D/A system is a 6-channel Digital-to-Analog converter as shown in Figure 4-23. This device is designed to be a listener on the GPIB in the inverse manner of the A/D system. As a listener, it receives data in pairs from the addressed talker. The first data byte is the channel number (address) that the following data byte is intended for. The second data byte of each is the 8-bit digital word that is to be converted to an analog value. The flow chart which can be used for development of the necessary software to drive the D/A converter is given in Figure 4-24. The MPU must initialize the PIA in such a manner that eight lines are used for the data transfer to the D/A (output lines) and three lines are used to decode the channel number address to the analog multiplexer (block 1). Reading R0R of the GPIA and testing BI will indicate if the byte in register R7R is valid data; BI is reset by reading R7R (blocks 2 and 3). The first byte received should be a D/A channel number address. This byte can be compared with addresses previously stored in memory (blocks 4, 5 and 6). If the address does not compare, a flag is set, or a routine is entered to identify the next valid channel address on the bus. Once a proper address is established and written into memory, the R0R register is read to see if the BI bit is set and if it is set, R7R can be read for the data (blocks 9 and 10). The data and multiplexer address can then be written to the PIA for conversion to an analog voltage. As in the case of the Talker, the status of the chip must be checked to see if it is still an active listener. This is done by blocks 7, 8, 11 and 12. The D/A system uses a polling routine to check the status of the GPIA. The system, rather than using a polling sequence, could be interrupt driven by enabling the appropriate interrupt mask bits in R0W. A software interrupt handler would also need to be placed in the MPU system memory.

4.5 GPIA AS A CONTROLLER

4.5.1 Hardware Configuration

With additional circuitry (Figure 4-25) the GPIA can be used to implement the controller function. The three basic components of this circuit are:

1. The MC68488 — The GPIA is used to monitor the status of the GPIB. The state of the handshake lines can be monitored through R3W. In addition, the GPIA is used to send and receive data and to send commands; i.e., the source and acceptor handshake functions are used by the controller much like they would be by a talker or listener device.
2. The MC6821 (Parallel Port) — The PIA is a dual 8-bit parallel interface adapter that is used to manage the five GPIB management lines.
3. The MPU — The controller state diagram is implemented in the MPU software. In microcomputer systems like the MC6801, the software is stored in the "on chip" ROM. The software architecture could be such that the MPU shown in Figure 4-25 would respond to a set of macroinstructions from a larger system or the MPU shown could encompass the entire controller device.

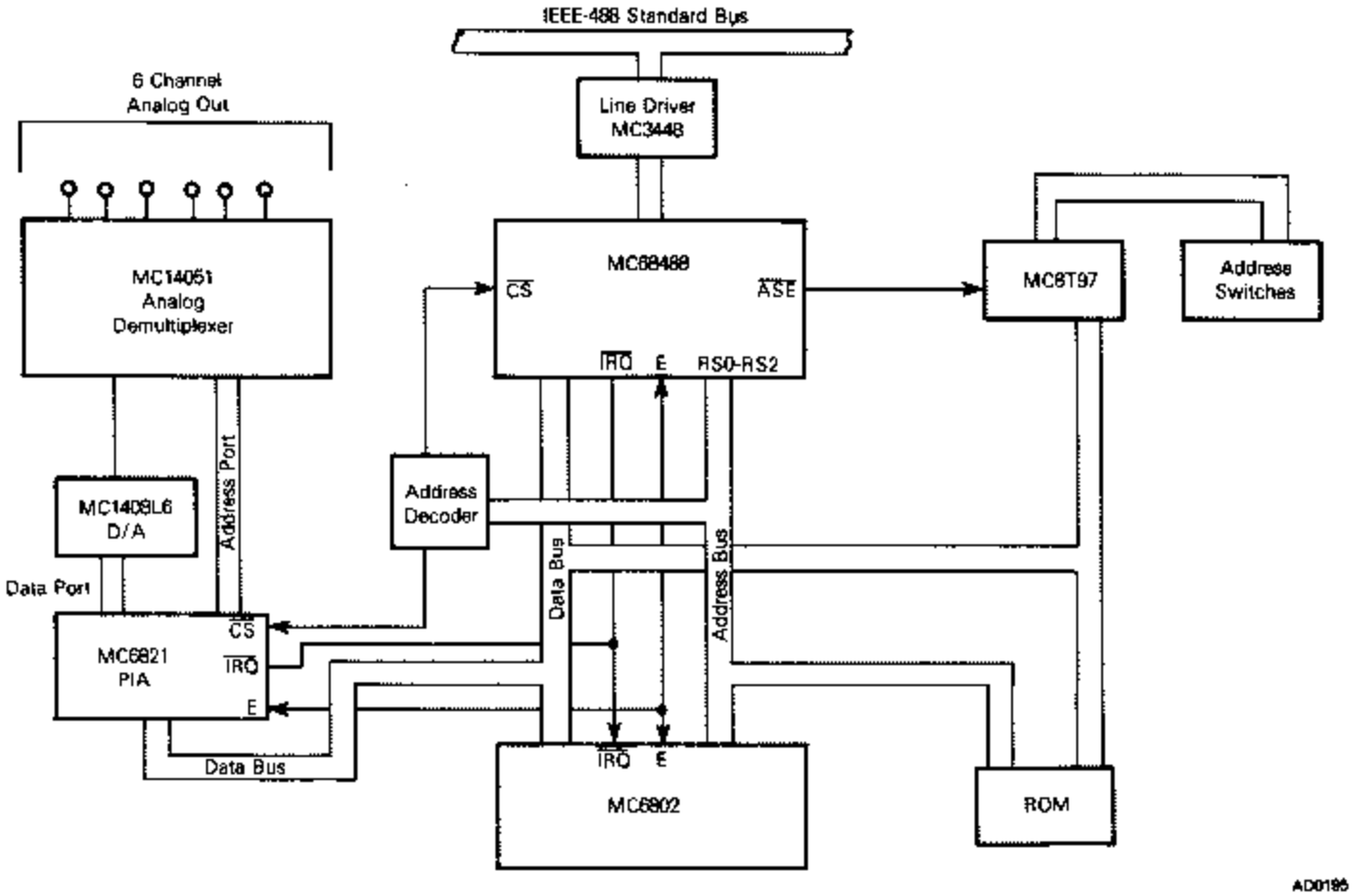
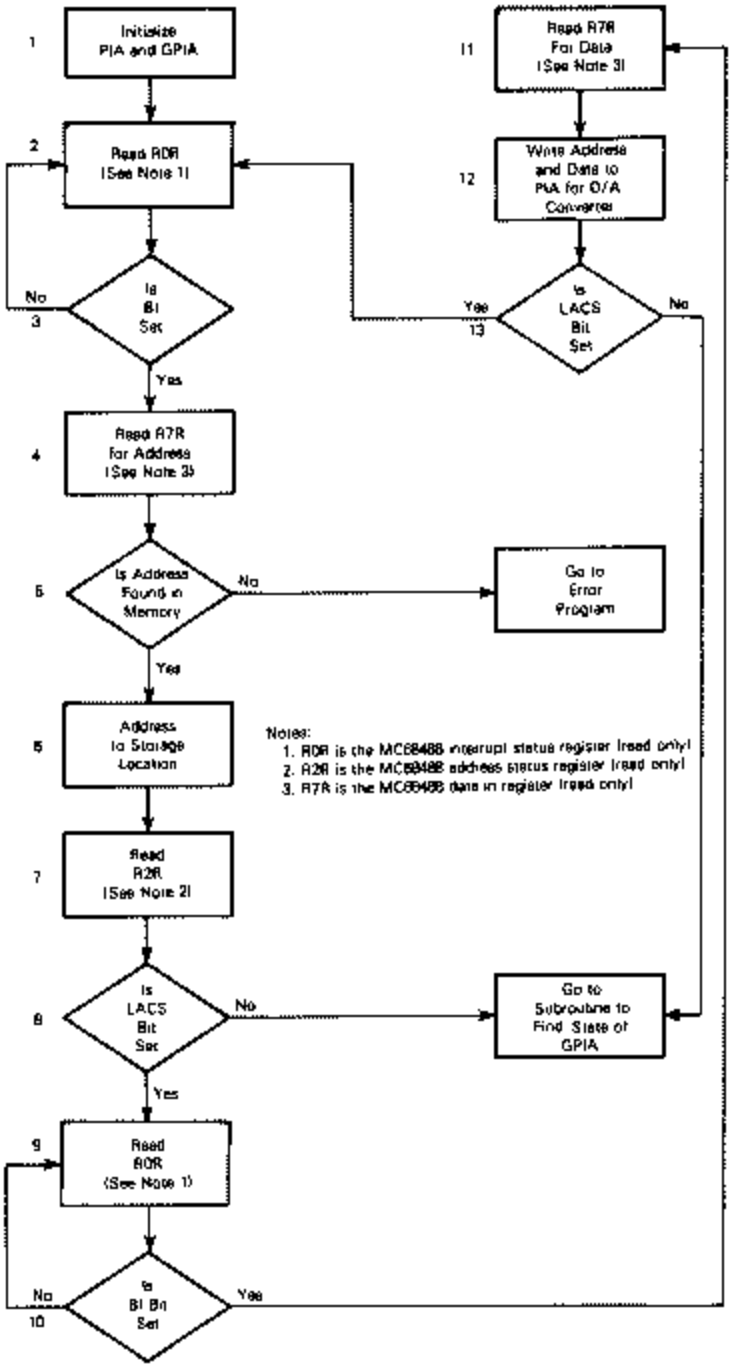


Figure 4-23. Block Diagram of D/A GPIA System



AD0198

Figure 4-24. Software Flow Chart for D/A System

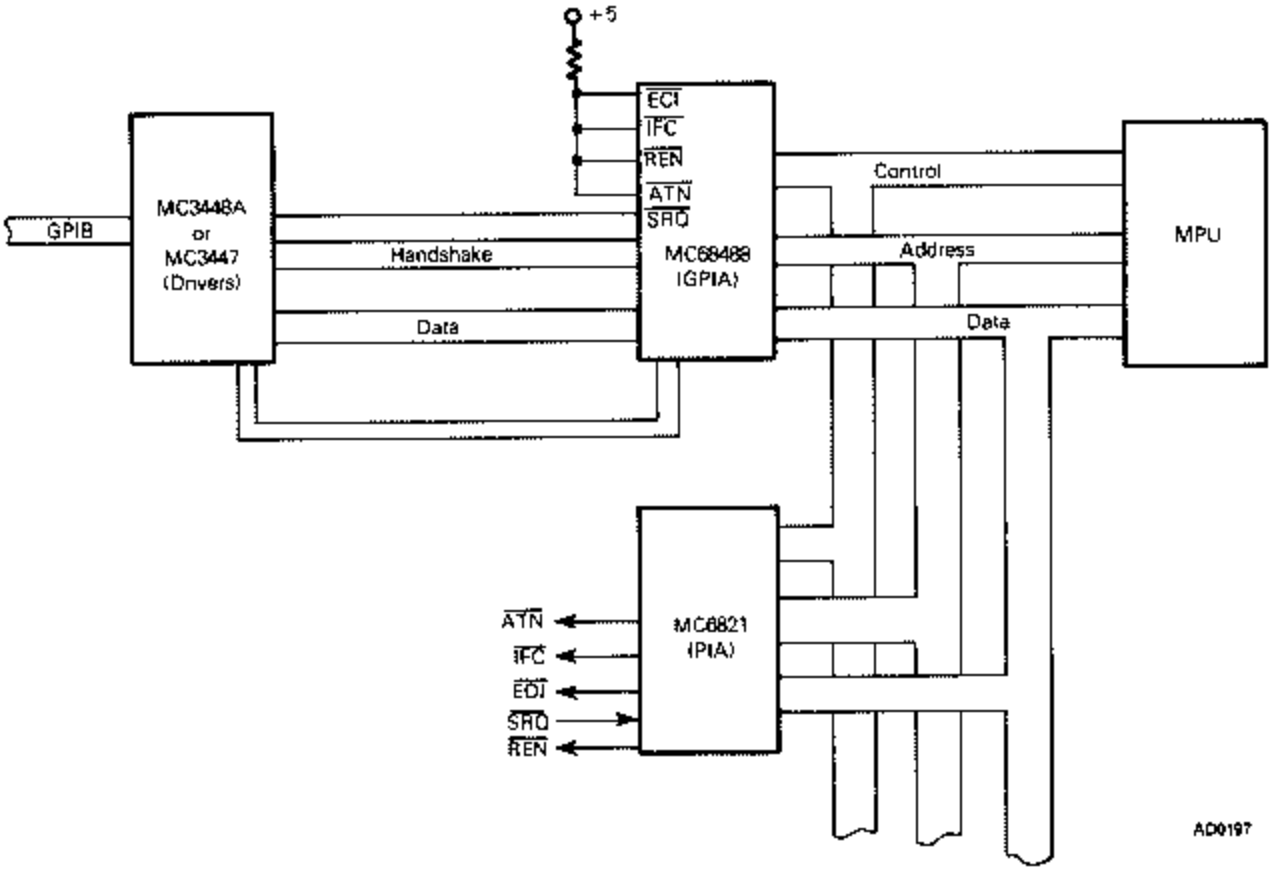


Figure 4-25. GPIA In a Controller Configuration

The purpose of Figure 4-25 is to show a possible controller circuit configuration and should not be considered a complete system. Gating and decoding circuitry are not included as well as the driver circuitry for the management lines. All of the management lines of the GPIA except $\overline{SRQ}$, are shown tied high (inactive state). In a multiple controller system, these lines need to be gated to the proper management lines for controller-to-controller interaction.

The MC6821 has two 8-bit parallel ports. Five of the 8-bits of one of the ports are used for the management lines. Since each bit of the PIA can be independently programmed as either an input or output, the proper signal flow is easily established. The handling of the $\overline{SRQ}$ line is, to a large degree, a function of the application. In many cases immediate recognition of an active $\overline{SRQ}$ is desired. In such a case, the $\overline{SRQ}$ line should be connected to one of the PIA special interrupt lines and cause an immediate interrupt of MPU. In other applications, this line will be one of the inputs on the PIA port and be routinely polled by the MPU.

4.5.2 Operation

To transmit either data or commands, the GPIA is configured as a talker through the talk only (to) local command. The controller sends commands by first having the MPU make the designated ATN bit on the PIA True (low) and then, using a handshake, transfer the command onto the GPIB

through the Data-Out-Register (R7W) of the MC68488. The only distinction between commands and data on the GPIB data bus is the state of the $\overline{ATN}$ line. Once the appropriate PIA bit has been asserted, the controller uses the GPIA to send commands much like a device programmed as a talker would use it to send data. Each time the BO status bit in R0R is set, the MPU knows the command currently on the bus has been accepted (a true on the DAC line) by all active devices. The state of the handshake lines can be monitored, if desired, through either R3R on the GPIA or by connecting these lines to the three unused lines on the PIA. Once the IEEE-488 Standard system has been configured, the controller releases the bus by programming the designated $\overline{ATN}$ line high on the PIA.

The controller can take over the bus asynchronously by asserting ATN during the middle of a handshake. As discussed in Section 4.2.3.1, this type of take over will produce undefined states on the GPIB and should be avoided except where absolutely necessary. If the controller takes over the bus asynchronously, it should immediately follow the take over by sending IFC with ATN false.

The controller normally takes over the bus by taking control synchronously. To do this, the controller monitors the handshake lines for an idle state before asserting ATN. The following procedure can be used by the controller to take-over the bus synchronously.

1. Make the GPIA a listener by setting the listen only (lo) bit.
2. Set the hlda bit (R2W) to hold off the RFD handshake on all received data bits or set hlde (R2W) instead of hlda if the take-over is to occur after the addressed listener has received the last data byte of a string. See following note for take over when data is not being transmitted.
3. When the BI bit is set, the MPU should read the Data-In Register (R7R). This will release the DAC handshake (DAC can also be released by writing dacr in the Auxiliary Command Register).
4. Now the MPU should monitor the DAC handshake line. When this line returns high, the controller will know that all active listeners have accepted the current data byte. Since the controller GPIA is holding off RFD, the handshake will be held up in an idle state. Another handshake sequence will not begin until RFD is released.
5. During this idle state, the MPU can assert $\overline{ATN}$, taking over the bus synchronously. The MPU should write rfdr (R3W) releasing the RFD hold off. Otherwise, when the GPIA is returned to the listener state, RFD will remain low (not-ready-for-data).
6. The MPU can now make the GPIA a talker (to) and commands can be sent. If the controller wants to continue holding off RFD on the reception of data from the addressed talker when the GPIA returns to the listener state, then the hlda or hlde bit need not be written low. Otherwise, these bits should be written low followed by writing rfdr (R3W) high. (See Section 3.3.1.2 for details.)

NOTE

The synchronous take over procedure just described assumes that the GPIB is actively transferring data (using handshake) from talkers to listeners. If there is no data being transmitted, the hlde/hlda hold off procedure will not work since there isn't a handshake to hold off. A software routine should be added to the above procedure to monitor and time out the handshake sequence. Thus, if after a predetermined time as a listener a BI interrupt has not occurred, the controller should poll the handshake lines for an idle state.

This section has described an approach for using the GPIA in a controller application. For details concerning the controller state diagram function and timing details, the reader should consult the IEEE-488 Standard.

APPENDIX A
IS0-7 BIT CODE REPRESENTATION

Many devices use the IS0-7 bit code because it is convenient to both generate and interpret this code. The relationship between this code and the IEEE-488 Standard interface messages are identified in this appendix.

The IEEE-488 Standard interface system utilizes message coding as defined in Table 2-11 to transfer interface messages between devices when the ATN message is true. This coding may be correlated to the IS0-7 bit code by relating DIO1 through DIO7 on the GPIB data line to bit 1 through 7 in Table A-1. The IS0-7 bit code does not contain the equivalent of the dedicated $\overline{\text{ATN}}$ line.

When the interface system defined in the IEEE-488 Standard is interconnected via a terminal unit to other environments, then protocol beyond the scope of the IEEE-488 Standard must be used to enable proper communication and avoid possible contradictions with other assigned meanings for the IS0-7 bit code.

Table A-1. Multiline Interface Messages: IS0-7 Bit Code Representation

b7 / b6 / b5 / Bits (2)					0 0 0	(1) MSG	0 0 1	MSG	0 1 0	MSG	0 1 1	MSG	1 0 0	MSG	1 0 1	MSG	1 1 0	MSG	1 1 1	MSG
b4 ↓	b3 ↓	b2 ↓	b1 ↓	Column / Row ↓	0		1		2		3		4		5		6		7	
0	0	0	0	0	NUL		DLE		SP		0		@		P		`		p	
0	0	0	1	1	SOH	GTL	DC1	LLO	!		1		A		Q		a		q	
0	0	1	0	2	STX		DC2		"		2		B		R		b		r	
0	0	1	1	3	ETX		DC3		#		3		C		S		c		s	
0	1	0	0	4	EOT	SDC	DC4	DCL	$		4		D		T		d		t	
0	1	0	1	5	ENQ	PPC (3)	NAK	PPU	%		5		E		U		e		u	
0	1	1	0	6	ACK		SYN		&		6		F		V		f		v	
0	1	1	1	7	BEL		ETB		'		7		G		W		g		w	
1	0	0	0	8	BS	GET	CAN	SPE	(		8		H		X		h		x	
1	0	0	1	9	HT	TCT	EM	SPD	)		9		I		Y		i		y	
1	0	1	0	10	LF		SUB		*		:		J		Z		j		z	
1	0	1	1	11	VT		ESC		+		;		K		[		k		{	
1	1	0	0	12	FF		FS		,		<		L		\		l		|	
1	1	0	1	13	CR		GS		-		=		M		]		m		}	
1	1	1	0	14	SO		RS		.		>		N		^		n		~	
1	1	1	1	15	SI		US		/	UNL	?		O		_	UNT	o		DEL	

MLA Assigned to Device

MLA Assigned to Device

MTA Assigned to Device

MTA Assigned to Device

Meaning Defined by PCG Code

Meaning Defined by PCG Code

Addressed Command Group (ACG)

Universal Command Group (UCG)

Listen Address Group (LAG)

Talk Address Group (TAG)

Primary Command Group (PCG)

Secondary Command Group (SCG)

Notes:
1. MSG = Interface Message
2. b1 = DIO1...b7 = DIO7
3. Requires Secondary Command
4. Dense Subset (Column 2 Through 5)
5. Sent and Received with ATN = 1

Table A-2. Command and Address Formats

	IB7...IB0
Addressed Commands	X000XXXX
Universal Commands	X001XXXX
Listen Addresses	X01XXXXX
Talk Addresses	X01XXXXX
Secondary Commands	X11XXXXX

Table A-3. ASCII and Hexadecimal Equivalents

Command Mnemonic	ASCII Code	Hex Equivalent (0)
GTL	SOH	01
SDC	EOT	04
PPC	ENQ	05
GET	BS	08
TCT	HT	09
LLO	DC1	11
DCL	DC4	14
PPU	NAK	15
SPE	CAN	18
SPD	EM	19
Secondary Commands	Blank through ←	60 through 7E

ATN = 1

Table A-4. ASCII Codes for Talk/Listen Commands

Address	ASCII Code	Hex Equivalent (0)
Listen Address	SP through >	20 through 3E
Unlisten	?	3F
Talk Address	@ through ^	40 through 5E
Untalk	_	5F

ATN = 1

APPENDIX B
QUESTIONS AND ANSWERS

Q1. What is the difference between $\phi 2$, E, clock and Enable?

A1. There is no difference. These terms can be used interchangeably when referring to the input at pin 6 of the MC68488. This input is often times called $\phi 2$ since in a 6800 system the $\phi 2$ clock applied to the MPU is the same signal that goes to the peripheral devices. It is also called Enable (or simply E) since this signal enables data transfers between MPU and peripherals (e.g., the MC68488) during Read/Write operations.

Q2. Can more than one pulse occur during a read or write operation of the internal registers?

A2. Problems can arise if this occurs. A double E pulse during a read cycle has the same effect as reading that register twice and likewise a double E pulse during a write cycle has the same effect as writing to that register twice. Remember, the E pulse that occurs during a read of the Interrupt Status Register (R0R) will reset the $\overline{IRQ}$ line if it is currently active. Should a second interrupt occur between these two E pulses, the second pulse resets the $\overline{IRQ}$ line before the MPU has a chance to acknowledge it. In the Command Status Register (R1R) a read resets the RLC bit if set. If the status byte is placed in the MPU accumulator by the second E-pulse, the RLC bits, even if it was set before the read sequence, will always be a "0". Also, when in the listener mode, the E pulse that occurs during a read of the Data-In Register (R7R) releases the DAC handshake. If the talker is able to place a second byte in R7R between these two E pulses, the second E pulse will release the DAC for the second byte. The result will be missed data bytes.

Q3. Do bits 4, 5, and 6 of R3R monitor the handshake lines while software reset (bit 7, R3W) is high?

A3. Yes.

Q4. Is there any way to destroy new-byte-available (nba)? For example, suppose the GPIA in a multi-range instrument is a talker and outputting data. The controller is monitoring the data and "sees" that a multi-range instrument is not outputting the correct data and a range scale needs to be changed. The controller takes over the bus synchronously, makes the device a listener and instructs it to go to a different range, and then returns the device back to the talk mode. Now, just prior to the controller taking over the bus, the MPU wrote a byte of data to R7W (made nba true), but the GPIA had not made this byte available (asserted DAV) to the listeners. What happens to this byte?

A4. There is no way to destroy nba except for a chip reset. In the example presented, the byte, that was placed in R7W before the synchronous take over, is passed (via handshake) to the listeners as soon as the device is returned to the talker state and the controller deasserts $\overline{ATN}$ high. This situation can, however, be handled external to the GPIA. For instance, the controller, after taking over the bus, can unlisten all active listeners and then become a listener accepting this data byte. Another approach would be for the talker to write over the current data byte before asserting $\overline{DAV}$ low. This would require the DAV handshake to be held up. Since DAV is automatic, external circuitry would be required for this situation.

Q5. Will an interface clear message destroy nba?

A5. No. The purpose of the IFC message is to place the GPIB in a known state. It does this by acting upon all messages that are GPIB originated. For example, if a device is made a talker by the controller sending MTA, a subsequent IFC message will reset the GPIA to the idle state (TIDS). However, local messages are a function of the device and not the GPIB management and as such an IFC message does not interface with these conditions. Since nba is local message, it is not affected by an IFC message. An IFC message will, in addition, only remove a device from TACS or LACS for the duration of the IFC message if the local message to or lo caused them.

NOTE

Care should be taken when using to or lo with a controller in the system. These local messages are meant to be used when there is not a GPIB controller.

Q6. Will an IFC message reset the BI status bit?

A6. No, BI indicates the presence of a data byte in the Data-In register (R7R). IFC does not affect this data byte nor its associated interrupt. However, the data byte in R7R is only reliable if the GPIB data lines retain the value of the last data byte until the MPU reads R7R. See question 7 for more details.

Q7. Is the data in the Data-In Register a latched condition?

A7. The inverse of the values on the GPIB data lines are placed into R7R on the falling edge of $\overline{\text{DAV}}$. While $\overline{\text{DAV}}$ is low, R7R follows the GPIB data bus. Data cannot be placed in R7R when $\overline{\text{DAV}}$ is high. During a data transfer, the talker must maintain the correct data on the GPIB until it receives an NDAC back from the addressed listeners. If the GPIB is interrupted and the data bus changes (e.g., going to a high-impedance condition) before the device MPU reads R7R, this information may be incorrect even though the BI status bit is set.

Q8. During a handshake sequence, can the $\overline{\text{DAV}}$ line to the GPIA be made high (false) prior to the DAC line being made true (high)?

A8. Yes. This is an allowed condition but is an interruption of a normal handshake sequence. It can place the GPIA(s) in a lock up state unless the procedure for an asynchronous take control (Section 4.2.3.1) is properly followed. The GPIA uses the acceptor handshake as defined in the IEEE-488 Standard to receive data and commands (Figure B-1) and expects the following sequence:

1. GPIB asserts RFD (high) alerting the sending device that the listening device is ready-for-data (a new handshake sequence can now begin).
2. The sending device asserts $\overline{\text{DAV}}$ (low) indicating valid data or commands on the data bus. The high-to-low transition of $\overline{\text{DAV}}$ causes the RFD line to become low. The $\overline{\text{DAV}}$ line must be held low until the sending device receives a high on the NDAC line.
3. When all listening GPIAs have accepted the byte on the GPIB, the DAC line becomes true (high) indicating this condition. This low-to-high transition causes the sending device to deassert $\overline{\text{DAV}}$. DAV becoming high in turn causes the NDAC line to become low (end of handshake sequence).

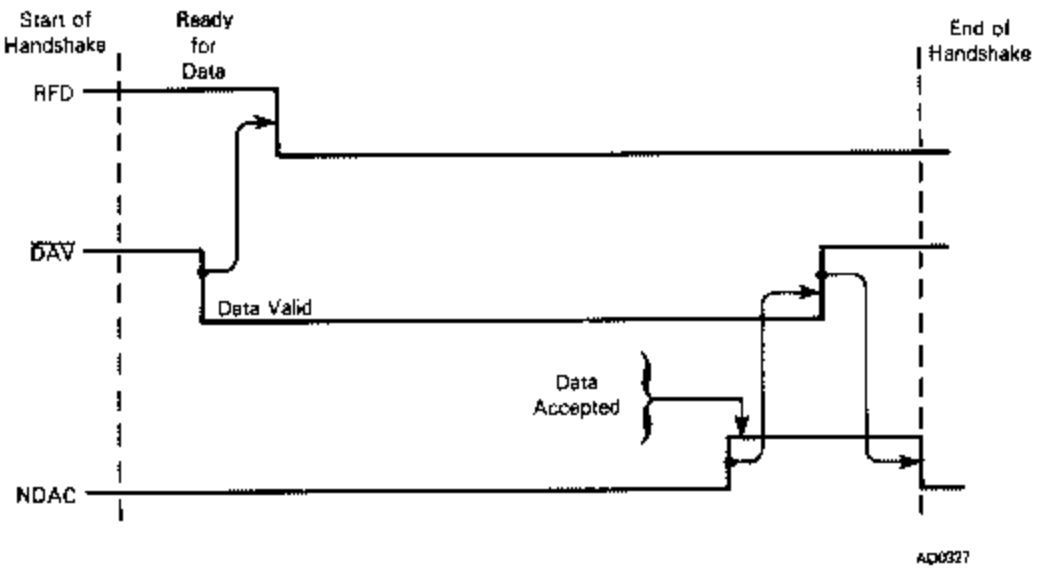


Figure B-1. GPIA Handshake Sequence

The handshake sequence shown herein must be adhered to for proper handling. Any violations can place the GPIA in an unintended state.

NOTE

If both the sending device and receiving device(s) use the MC68488, the handshake sequence will always be completed in the proper manner.

If the DAV handshake line is made false (high) before NDAC is made false (high) during a handshake sequence, the listener GPIA(s) responds as follows:

1. IFC sent by the controller with ATN false before another handshake sequence is initiated. If this occurs the GPIA responds to IFC as described in Section 3.1.1.2. The BI status bit is set; but since the handshake was interrupted, the data byte in R7R not valid.

NOTE

If IFC is sent with ATN true, it must be sent again with ATN false.

2. Another handshake is allowed to be initiated before IFC is sent with ATN false. If this occurs the GPIA will not generate interrupts for subsequent data bytes received by the listener GPIA(s). The device will respond to commands and move into and out of TACS, LACS, etc. but no further BI interrupts will be generated. The only solutions to this situation are to reset the MC68488 or have the MPU perform a dummy read of R7R. In most systems, however, the MPU will not know it is to perform either of these tasks and a power-on reset will be required.

Q9. Does the MC68488 interface to other manufacturers' microprocessors?

A9. Yes. As long as the electrical specifications are met and the Read/Write timing conditions are adhered to (as indicated in the MC68488 Data Sheet), the MC68488 will easily interface with any 8-bit microprocessor.

Q10. What is the state of the handshake lines during reset?

A10. During either a hardware or software reset all the handshake lines will be passively high, i.e., DAV, NRFD and NDAC are sent passively false. In addition, the EOI and SRQ lines are set passively false (high). This insures the GPIA does not disturb the GPIB on power-up or chip reset.

Q11. Does the REM bit in R1R always monitor the REN management line?

A11. No. REM only monitors the REMS or RWLS states. See state diagrams in Chapter 2.

Q12. An MPU interrupt occurs ($\overline{IRQ}$ line goes low) and then the status bit that caused the interrupt is reset before the MPU acknowledges the in interrupt condition. Will the $\overline{IRQ}$ line return high?

A12. No. The $\overline{IRQ}$ line only returns high by reading R0R or by a chip reset.

Q13. Can the controller place the GPIA in both TACS and LACS simultaneously by sending both MTA and MLA?

A13. No. The GPIA can never be in both LACS and TACS at the same time. If in LACS and the controller sends MTA, the GPIA will move to LIDS and TACS. If in TACS and the controller sends MLA, the GPIA will move to TIDS and LACS. The GPIA will also move to TIDS from TACS if the controller sends another device talk address (OTA).

Q14. Is there a status bit for an interface clear message?

A14. No. The GPIA does not report the occurrence of an IFC message to the MPU. If it is important to know if an IFC message occurred, this line should be monitored external to the MC68488.

Q15. Does the MC68488 require an MPU on the device side?

A15. Yes. The MPU is used to access the 15 internal registers of the GPIA. The MC68488 is an MPU peripheral part and is directly compatible with all of Motorola's 6800 Family microprocessors. In addition, it can be interfaced with any computer bus as long as the Read/Write timing characteristics indicated in the MC68488 Data Sheet are met.

Q16. How does a GPIA, addressed as a talker, function if there are no active listeners on the bus?

A16. If there are no active listeners, the RFD and DAC handshake line float and assume the passively true state. The talker BO status bit is set, instructing the MPU to write a byte of data to the GPIA. The MPU writes a byte to R7W resetting BO. Since RFD and DAC appear at a high level, the talker GPIA interprets this as the listener(s) having accepted the current data byte and are ready for more data; the BO bit is again set, instructing the MPU to output another byte. This sequence will be repeated continuously.

Q17. If an active talker writes to R7W when there are no active listeners, does $\overline{DAV}$ go low?

A17. Yes. RFD being high causes DAV to go low indicating valid data. Because DAC is high, the talker assumes this data byte has been accepted. $\overline{DAV}$ will immediately return high and another BO status bit will be produced.

Q18. How do I clear the BO status bit without writing to R7W or resetting the GPIA? For example, the talker is sending data, BO is set when each byte is accepted. With the last byte, the EOI line is set. Another BO interrupt occurs when this last byte is accepted.

A18. This is true. Another BO interrupt is generated. In an interrupt driven system the MPU checks R0R and sees BO set. Since it knows there is no more data to be sent, it should ignore the interrupt. Reading R0R releases the $\overline{IRQ}$ line and this line will not be set until another interrupt edge is detected (see Section 3.3.2). The BO bit remains set, however, until the GPIA is removed from TACS. Since the BO bit remains set during this sequence, another MPU interrupt does not occur. If the controller asserts ATN true (low), the GPIA will move out of TACS into TADS and the BO will be reset. If the GPIA is not removed from the talker state, another BO status interrupt will occur when ATN is deasserted high.

Q19. What happens to the BO bit if the Untalk Command is sent?

A19. The BO status bit, if set, is reset when the GPIA receives an Untalk Command. In an interrupt driven system it is possible for the controller to send an Untalk Command, resetting BO, before the MPU interrupt handler is able to check the Interrupt Status Register. This could happen when the Talker sends the last byte of a string and: (1) the controller detects this, (2) synchronously takes control of the bus, and (3) sends an Untalk Command to the Talker. Under these conditions a "ghost interrupt" might occur. If the "ghost interrupt" occurs, the MPU software should check R2R to see if the GPIA is still in the Talker State. (Refer to Question 27 for more details concerning the "ghost interrupt.")

Q20. What is meant by overlapping interrupts?

A20. Considering the expected protocol on the IEEE-488 Standard bus the Interrupt Status register has been specifically designed to produce one status condition at a time. It is possible, however, in some situations for more than one status bit to be set (excluding the INT status bit). When this occurs, the set status bits are said to be overlapping.

Q21. In an interrupt driven system, if a second interrupt condition occurs after the MPU reads R0R for the first condition (deasserting the $\overline{IRQ}$ line) and before the MPU has acted on the first condition, what happens to this overlapping interrupt condition and when is it likely to occur?

A21. The Interrupt Status Register has been specifically designed to efficiently monitor the needed GPIB status conditions and avoid overlapping interrupts. It is possible, however, for an overlapping interrupt to occur when in the secondary address mode. If the controller sends REN and then addresses a device to listen, the following could occur. When the Secondary Command Group is received, the APT status bit will be set. This bit is reset when the handshake is released (the MPU writes dacr high in R3W). If during the same write instruction the MPU writes msa high, indicating the complete listen address has been received, the GPIA moves to the Listener Active State (LACS). Thus, the APT bit will be reset and the device will move to LACS simultaneously. Since REN has

been asserted prior to the GPIA receiving its listener address, an impending RLC status condition and hence CMD interrupt (if enabled) will be resident in the GPIA. As soon as the device moves to LACS, both the RLC and CMD bits will be set. If this situation occurs, the APT bit is reset when the CMD bit is set and it is possible that the CMD low-to-high transition needed by the $\overline{IRQ}$ Line could be missed. (Remember, the $\overline{IRQ}$ line responds to the leading edge of the INT status bit. If a second status bit is set before the first is reset, the INT status bit remains set and a low-to-high transition for the second status does not occur.) Thus, the $\overline{IRQ}$ line is not asserted (low) by the CMD bit.

If the last sequence of instructions in the interrupt handler is a reset of the Interrupt Mask Register, followed by programming it to its original state, all overlapping interrupt conditions will be allowed to cause an MPU interrupt. See Section 3.3.2 for details.

Q22. What is the difference between a hardware and software reset?

A22. The only difference is that a hardware reset resets the Address Register (R4W) and a software reset (writing bit 7, R3W high) does not affect R4W.

Q23. When using an EXORcisor can the MAID System program be used to interrogate the internal registers of the GPIA?

A23. MAID can be used to examine all read register bits except the RLC bit in R1R. MAID automatically performs a double read on each location it examined and it is the second read that is displayed by MAID. Since the RLC bit is reset by a read of R1R the first read in the double read sequence will reset RLC. Thus, second read always indicates that this bit is low.

Q24. What does the SRQS bit in R6R monitor?

A24. There are two cases:

a. Prior to the GPIA being placed in the APRS state, the SRQS bit monitors the rsv bit in R6W. If rsv is set, the SRQS will be set. If rsv is reset, the SRQS bit is reset.

b. If the rsv bit is set, it causes the SRQS bit to be set and the $\overline{SRQ}$ management line to go low. When the controller initiates a Serial Poll, the SRQS bit in R6R is reset when the GPIA enters SPAS.

Thus if rsv is set, causing the GPIA to actively pull the $\overline{SRQ}$ management line low, the SRQS bit is set. When the GPIA enters SPAS, releasing the $\overline{SRQ}$ management line, the SRQS status bit will return low even if another device is actively keeping the $\overline{SRQ}$ management line low.

Q25. If the rsv bit in R6W is not reset after a Serial Poll procedure, will subsequent Serial Polls by the controller show bit 6 of the Serial Poll status byte to be set?

A25. Yes. Since the controller checks bit 6 of each device status byte during a Serial Poll procedure, it is important to reset rsv after being Serial Polled so that subsequent Serial Polls do not indicate the device is still requesting a service.

Q26. In the dual addressing mode of operation what happens if address 30 is placed in R4W as the primary address, i.e., hex 9E is written to R4W?

A26. The Dual addressing mode allows the GPIA to recognize two adjacent addresses, separated by the least significant bit of the address, as valid primary addresses. However, since address 31 is an invalid primary address (address 31 has the same code as that of the Untalk or Unlisten Commands) the address 30 also becomes an invalid primary address in the Dual addressing mode. The GPIA responds to a programmed address of 30 as follows for dual addressing.

1. In the listener mode, the Unlisten command (same as sending the listen command with address 31) will unlisten the GPIA. If address 30 is sent by the control the GPIA is addressed to listen.
2. In the Talker mode, if the Untalk command is sent (same as sending the talk command with address 31) the GPIA does not respond to this as an Untalk command and is addressed to talk. If the talk address 30 is sent by the controller, the GPIA is addressed to talk.

Q27. What causes a "ghost interrupt?"

A27. A "ghost interrupt" results when the MC68488 issues an interrupt request ($\overline{IRQ}$) to the MPU, but, when the software routine services the interrupt, it no longer exists (all bits in the Interrupt Status Register are reset). There are two conditions that can legitimately cause a "ghost interrupt." They are:

a. **SPAS status bit** — If the controller conducts a serial poll by sending Serial Poll Enable (SPE) and then sends the GPIA talk address, the SPAS status bit is set and can cause an interrupt. When the controller receives the Serial Poll Status byte it sends Serial Poll Disable (SPD) which resets the SPAS status bit. If the controller can perform this sequence of events before the interrupt handler can check the SPAS status bit, the MPU will not find any status bits set ("ghost interrupt"). The possibilities are twofold:
   1) If this device had actually requested the service, then the MPU, after receiving the interrupt ("ghost" or not) should check bit 6 of the Serial Poll register. If this bit is reset the MPU knows that a Serial Poll was conducted and can reset the rsv as per normal Serial poll handling procedures.
   2) If this device did not request the service request and SPAS is not set the software should detect this as a "ghost interrupt" and proceed with normal operations.

b. **BO Status bit** — The BO Status bit is set whenever the MC68488 is in the Talker Active State and the output register (R7W) is empty. After the listener(s) accept the current data byte on the IEEE-488 Standard bus, the BO status bit is again set and, with the appropriate mask bits set, causes an interrupt. When the talker sends the last byte of a string it is possible for the controller to detect this, synchronously take control of the bus, and untalk the talker; however, when the last byte is accepted the BO status bit is again set and if so programmed, causes another interrupt. It is possible for the controller to untalk the device and reset BO before the MPU interrupt handler is able to check the status register. Under these conditions a "ghost interrupt" occurs. In many applications the "ghost interrupt" can be ignored and the software proceed with other operations. The BO "ghost interrupt" can be avoided by having the MPU software disable the BO status bit (reset the mask bit) before sending the final data byte. If the BO status bit mask is not disabled, the MPU software should check the address status register to see if the GPIA has been unaddressed to talk. If it has, the interrupt was most likely caused by a BO "ghost interrupt." If the GPIA is still addressed to talk, the interrupt was most likely caused by a serial poll "ghost interrupt." Note that in some applications the controller might

send the Untalk Command and immediately follow it with My Talk Address (MTA). If this happens the TACS bit could go from the set condition to the reset condition and back to the set condition before the MPU can check the status of TACS. In those applications, where it is important to monitor all address changes, the MC68488 should be programmed to hold off the handshake on all commands (the dacd bit in R3W should be set). If this approach is taken, the MPU must use a polling routine to monitor R3W for the occurrence of Data Valid ($\overline{DAV}$). If $\overline{DAV}$ is detected, the software should then check R2R for $\overline{ATN}$ asserted. If both are true, a command is present on the GPIB. See Section 3.3.1.1 for details on using dacd.