Tektronix®
COMMITTED TO EXCELLENCE

Please Check for
CHANGE INFORMATION
at the Rear of this Manual

4051 GPIB
HARDWARE SUPPORT

REFERENCE MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

MANUAL PART NO.
070-2270-00

First Printing MAR 1977
Revised JUL 1981

DISCLAIMER

The interface circuits in this manual are for educational purposes only. It is our way of presenting practical 4051 GPIB interface designs to help you get started. All efforts have been made to keep the designs simple, practical, and economical. The designs serve nicely as a starting point—a basis for learning the fundamentals of 4051 GPIB interfacing. In most applications you'll find it necessary to modify parts of a design, add more sophisticated components, and arrive at more innovative approaches to meet the interfacing requirements for a particular peripheral device.

Each interface circuit in this manual has been built and tested with a 4051 to make sure that it works; and we feel reasonably sure that the interface circuits will work in most applications, provided the circuits are built by a competent electrical engineer or electronic technician with a good working knowledge of digital logic design. However, since some of the interface designs have not undergone the rigorous Tektronix evaluation and environmental testing procedures that are normally required of most Tektronix instruments, Tektronix cannot assume any liability for a customer-built interface based on these designs. Furthermore, Tektronix cannot assume liability for any interface circuitry that has not been designed, built, tested, and marketed by Tektronix. Any damage to Tektronix equipment which results from interface circuiting not designed, built, tested and marketed by Tektronix may nullify the warranty on the damaged Tektronix equipment.

PREFACE

The purpose of this manual is to provide hardware support information to Tektronix customers and personnel who are interested in designing interfaces for the Tektronix 4051 Graphic System and its General Purpose Interface Bus (GPIB). Although the discussions and designs in this manual are directed specifically toward the 4051 GPIB, the information contained herein is valuable to any design effort involving the implementation of IEEE Standard 488-1975.

This manual clarifies the guidelines set forth in the IEEE Standard and at the same time provides practical working models of 4051 GPIB interfaces. The manual is organized in the following manner:

Section 1 contains background information on IEEE Standard 488-1975 for those of you not familiar with the document. The GPIB organization is discussed first, followed by the bus signal line definitions, protocol definitions, and the 4051's compatibility with the IEEE Standard. If you have a clear understanding of the IEEE Standard, most of this material may be skipped without loss of continuity.

Section 2 discusses a model of a general purpose TTL interface that works on the 4051 GPIB. Starting with the GPIB connector, this section shows how to make the proper electrical connections to the bus and how to arrange bus receivers and transmitters in an effective listen/talk configuration. Typical TTL circuit designs are then discussed, starting with Listen Handshake circuitry. Talk Handshake circuitry, is covered next, then Address Decoding circuitry, and finally Serial Poll circuitry. A schematic diagram and an Integrated Circuit list is provided for each circuit. These circuits can be combined into a complete interface that listens, talks, and responds to serial polls over the 4051 GPIB. The complete block diagram and schematic diagram for this interface are found in the diagram section at the back of this manual.

Section 3 presents a working model of a microprocessor-based GPIB interface. The interface design is based on the MC6800 microprocessor and is taken directly from a production model TEKTRONIX 4924 Digital Cartridge Tape Unit. The written text in this section originates from an application note written by Mr. Steve Baunach, the Tektronix firmware engineer responsible for the design and implementation of both the 4051 GPIB interface controller and the 4924 GPIB interface.

Section 4 in this manual presents detailed timing information on the 4051 GPIB. Timing diagrams are used to illustrate GPIB data transfers characteristics for the BASIC I/O statements PRINT, INPUT, WRITE, READ, WBYTE, RBYTE, and POLL. A written explanation of the timing events accompanies each diagram to assist those of you who are unfamiliar with timing diagram symbology. The timing events in this manual are based on Level 2 4051 firmware and may differ slightly from other releases of 4051 firmware.

Section 5 in this manual discusses methods that can be used to calculate maximum effective data rates for 4051 GPIB data transfers. The information in this section may be used to determine if the 4051 GPIB data rates are compatible with your system application.

The back of this manual contains an Appendix with information on the 4051 internal floating point format for numeric data. This information is provided for those who are interested in transferring numeric data over the 4051 GPIB with READ and WRITE statements. A diagram section containing a block diagram for the general purpose interface, the schematic diagram for the general purpose interface, a schematic diagram for the 4051 GPIB Binary Header Generator can also be found in the back.

CONTENTS

CONTENTS (cont)

CONTENTS (cont)

Fig. 1-1. The 4051 GPIB can be interfaced to anything.

Section 1

BACKGROUND INFORMATION ON THE 4051 GPIB

INTRODUCTION

This section provides background information on IEEE Standard 488-1975 and the 4051 GPIB. The development of the IEEE Standard is discussed first, followed by an explanation of the 4051 General Purpose Interface Bus organization, signal line definitions, and protocol definitions. This material is most helpful to people who are unfamiliar with GPIB concepts. This section may be skipped by more experienced personel without loss of continuity.

WHY WAS A STANDARD INSTRUMENTATION INTERFACE NEEDED ?

PROGRAMMABLE INSTRUMENTATION DEVELOPMENT

Technological advances have brought an increase in the sophistication of the computer industry over the last ten years. These technological advances have had a parallel effect on measurement devices and instrumentation. The instrumentation of today is very versatile and highly programmable; this justifies the need for a general purpose interface standard.

Not long ago measurement devices were almost all analog, using panel meters to indicate measurement values. Some devices which were inherently digital (counters, for example) began appearing with easy-to-read digital displays. Once displays become digital, manufacturers started supplying output-only interfaces to allow the display readings to be transferred to recording devices, such as digital strip printers. In this way, the "numbers" from the instruments were permanently recorded without the need for anyone to write them down. To perform computations with these numbers still required someone to key them into a terminal or key punch; a direct transfer of the data from the instrumentation to the computer (sometimes via paper tape) was only a step away. A number of custom interfaces appeared which allowed specific instruments to output data to specific computers.

A device such as the frequency counter just discussed has an analog input and digital output. From the standpoint of the digital information, the frequency counter may then be referred to as a "talk only" device.

"Listen only" devices also exist. Just as measurement devices can talk to a computer, those devices capable of generating stimuli or other signals can be programmed by a computer and are classified as listeners. A waveform generator is an example of such an instrument. A number of waveform generators exist having output frequencies and amplitudes which are programmable digitally via interfaces specific to the brand and model of instrument.

With the growing popularity of instruments with digital interfaces came an even greater number of interfaces with differing mechanical, electrical, and functional specifications. Some devices required two interfaces, one for input and one for output. A multimeter could be such a listener/talker capable of being programmed by a computer-controller to a measurement setting of volts DC, and range of 100.00 full scale. The multimeter could then make a measurement and output the reading in a digital format for use by the computer.

EVENTS LEADING TO A STANDARDIZED INTERFACE

The traditional approach to interfacing such instrumentation has been to provide each device with specialized control, data, and status signal lines. This works well, but has resulted in just about as many interface solution techniques as there are design engineers. The net result was a dedicated interface structure for each device or instrument integrated into a system. This then led to the design of many interface adapters to accommodate the ever-increasing variety of codes, formats, signal levels, logic conventions, and timing protocols, to name only a few factors.

Initial attempts were made to standardize the interfaces from the perspective of the computer or controller. Such a solution proved too costly, as more sophisticated instrumentation had many programmable input and output lines, and began to require as much as 100 interface signal lines. At this point, both European and American instrumentation manufacturers began an all-out effort toward a proper solution.

HOW IS IEEE STANDARD 488-1975 DEFINED ?

DESIGN OBJECTIVE FOR THE STANDARD INTERFACE

The needs of the entire instrumentation community are varied and one interface solution cannot satisfy all of everyone's requirements. It is the most-frequent needs which require standardization and which fall within the objectives of IEEE Standard 488-1975. In order to better define those systems for which the standard would be suited, the following limitations were set:

1. Data rates of up to one megabyte (one million characters) per second would be supported. This would handle all but the fastest analog-to-digital converters and mass storage devices.

2. Distances up to a total of twenty meters would be supported. This would handle instrument setups close to the controller, but not remote terminals and displays.

3. Up to about fifteen devices running simultaneously would be supported. Most systems fall into this category.

4. Communications methods would be optimized for devices with typical message lengths of ten to twenty characters (digital), as most programmable instruments fall into this category.

Taking full account of cost, flexibility, and compatibility as major factors to be considered, the objectives of the IEEE 488 Standard are to:

1. Define a general-purpose system for use in limited distance applications.

2. Specify the device-independent mechanical, electrical, and functional interface requirements.

3. Specify the terminology and definitions related to the system.

4. Enable the interconnection of independently manufactured apparatus into a single functional system.

5. Permit apparatus with a wide range of capability—from the simple to the complex—to be interconnected to the system simultaneously.

6. Permit direct communication between devices without requiring all messages to be routed through a control unit.

7. Define a system with a minimum of restrictions on the performance characteristics of the devices.

8. Permit asynchronous communication over a wide range of data rates.

9. Define a system that, of itself, may be relatively low cost and permits the interconnection of low cost devices.

10. Define a system that is easy to use.

WHAT IS INCLUDED IN IEEE STANDARD 488-1975 ?

The primary focus of the General Purpose Interface Bus (IEEE Standard 488-1975) is to define an interface system to interconnect self-contained devices to other devices by external means. This means that the GPIB is a *device-independent* interface system. In this manner, existing programmable apparatus should be able to connect to GPIB-compatible devices by adding another module to the present interface. New instruments need not be designed with the GPIB in mind to eventually be compatible with the GPIB.

There are four elements to any interface system. These elements are:

1. Mechanical Elements
2. Electrical Elements
3. Functional Elements
4. Operational Elements

Of these four, only the fourth is truly device-dependent. Operational elements state the way in which any one device reacts to a signal on the bus. These reactions tend to be device-dependent characteristics and state the way in which the devices use the interface via application software. As the operational characteristics of all present and future devices and systems can not be foretold, the interface standard does not include operational elements. The characteristics of the 4051 are, however, discussed in this manual.

Mechanical elements, the physical connectors and cables, are defined by the standard. Building mechanical specifications into the standard ensures that interconnecting GPIB compatible devices will never require more than a standard interfacing cable. It should never be necessary to wire connectors or route signals to appropriate pins by studying the manuals of each of the devices concerned. The connectors have 24 pins with trapezoidal shells for ease in interconnecting devices. The cables are provided with a plug and a receptacle at each end to allow rigid stacking of connectors at any cable intersection or device. This allows both star and bus structure configurations. Sixteen of the 24 pins are defined for signals.

Electrical elements, the voltage and current values required at connector nodes, are well defined by the interface standard. All specifications are based on the use of TTL technology. The logical states are defined as follows:

Coding Logical State	Electrical Signals Levels
Ø	corresponds to 2.0 V ≤volts ≤5.2V (called high state)
1	corresponds to 0 ≤volts ≤0.8 V (called low state)

Messages can be set as either active or passive true signals. Passive true signals occur in the high state and must be carried on a signal line using open collector devices. (Driver requirements are expanded upon in IEEE Standard 488 Section 3.3)

Functional elements are well defined by the interface standard and determine the ease with which one can interconnect independently designed devices and have them interact appropriately. Functional elements cover:

1. Interface functions which define the use of specific signal lines so that a device can receive, process, and send messages.

2. The specific protocol by which interface functions send and receive their limited sets of messages.

3. Logical and timing relationships between the allowable states of interface signal lines.

4. The repertoire of interface functions from which the design engineer may choose for his particular device application area.

5. The total processing capability and communications capability that the system is capable of supporting.

Ten interface functions provide the system with complete communications and control capabilities. These are discussed in the sections on compatibility with the interface standard.

Therefore, the IEEE Standard 488-1975 interface definition encompasses the device independent elements of mechanical, electrical, and functional nature to leave only the device-dependent operational elements to the design engineer, thus insuring system compatibility.

COMPATIBILITY WITH IEEE STANDARD 488-1975

There are ten interface functions, some with as many as 28 allowable subsets, supported by and included in the interface standard. Only those functions important to a particular product's applications need be implemented.

A device need only be able to handshake on data to be compatible with the standard. In order to be addressed, it must also acknowledge the call from the controller and be able to recognize its own address. The least number of signal lines, then, that must be implemented in order for a device to respond on the interface bus are:

1. The eight data lines D101-D108

2. The three data transfer handshake lines:
   a. NRFD—not ready for data
   b. NDAC—data not accepted
   c. DAV—data valid

3. CONTROLLER's management line:
   a. ATN—attention

4051 GPIB TO IEEE 488 COMPATIBILITY

In general, the 4051 Graphic System acts as a standard talker, listener, and controller. The controller function does not have the ability to conduct a parallel poll; it does however, have the ability to conduct a serial poll. Serial polls are taken each time the POLL statement is executed in BASIC.

The Graphic System does not have the ability to transfer control to another device on the GPIB with controller capability. Therefore, the Graphic System assumes that it is the only controller on the GPIB. This assumption is made at all times.

4051 GPIB Support of IEEE 488 Interface Functions

The degree to which the 4051 GPIB supports each of the ten interface functions is described below. The referenced sections are in the IEEE Standard 488-1975 document.

1. SH (Source Handshake Function, Section 2.3)

   The SH function provides a device with the ability to initiate and terminate the transfer of messages on the data bus.

   The 4051 conforms to subset SH1 meaning that it is completely compatible with no states omitted.

2. AH (Acceptor Handshake Function, Section 2.4)

   The AH function provides a device with the capability to guarantee proper reception of messages on the data bus as well as the capability of delaying initiation or termination of such messages.

   The 4051 conforms to subset AH1 meaning that it is completely compatible with no states omitted.

3. T (Talker Function, Section 2.5)

   The T function provides the device with the capability of sending device dependent data over the bus to other devices.

   The 4051 conforms to subset TE3 meaning that it is a basic extended talker, honoring secondary addresses. The 4051 addresses itself internally and not over the GPIB. The only talker substates not implemented would allow other devices to poll the status of the 4051. This isn't required because the 4051 is the system controller at all times.

4. L (Listener Function, Section 2.6)

   The L function provides the device with the capability of receiving device dependent data over the bus from other devices. This capability exists only when the function is addressed to listen.

   The 4051 conforms to subset LE1 meaning that it is a basic extended listener, honoring secondary addresses. The 4051 addresses itself internally and not over the GPIB, but is also capable of removing itself from the bus as a listener.

5. SR (Service Request Function, Section 2.7)

   The SR function provides the device with the capability to asynchronously request service from the controller in charge of the interface. The 4051 is always the controller and therefore has no need for this capability.

   The 4051 conforms to subset SRØ meaning that it has no capability to issue an SRQ. This has no consequence on the 4051's capability to honor SRQs, which is quite complete.

6. RL (Remote Local Function, Seciton 2.8)

   The RL function provides the device with the capability to select between two sources of input information: programmed or manual (remote and local). This is most important with stand-alone measurement devices, and is not used with the 4051.

   The 4051 conforms to subset RLØ, meaning that it has no compatibility here, although it can control the RL function of other devices.

7. PP (Parallel Poll Function, Section 2.9)

   The PP function provides the device with the capability to present one bit of status to the controller in charge, without being addressed to talk. The 4051 is the controller in charge and does not require this capability.

   The 4051 conforms to subset PPØ meaning that it has no capability here.

8. DC (Device Clear Function, Section 2.1Ø)

   The DC function provides the device with the capability of being cleared (initialized) either individually or as part of a group of devices.

   The 4051 conforms to subset DCØ meaning that it does not have the capability of being cleared by an external device. The BASIC keyword INIT does inititalize the 4051 as well as all devices on the GPIB.

9. DT (Device Trigger Function, Section 2.11)

   The DT function provides the device with the capability of having its basic operation started either individually or as part of a group of devices.

   The 4051 conforms to subset DTØ which means that it has no capability of being "started" remotely. This is a logical consequence of the 4051 being the controller of the system.

10. C. (Controller Function, Section 2.12)

    The C function provides the device with the capability to send device addresses, universal commands, and addressed commands to other devices over the interface. The device with an active C function is called the system controller (of the interface system).

    The 4051 conforms to subsets C1, C2, C3, C4 and C28. These have the following meanings:

    a. C1—the 4051 is the system controller.

    b. C2—the 4051 can send IFC (interface clear) with the BASIC keyword INIT to clear all other devices on the bus and thereby takes charge of the bus.

    c. C3—the 4051 can send REN (remote enable) to lock out the front panels of devices on the bus and put them in a remotely programmable mode. This occurs automatically whenever the 4051 is running a BASIC program.

d. C4—the 4051 can respond to SRQs (service requests) issued asynchronously by devices on the bus. The 4051 then undertakes a serial poll and determines which device originated the SRQ and the status of the device.

e. C28—the 4051 can send interface messages.

f. As mentioned previously, the 4051 is the only controller in the interface system. Therefore it cannot receive control from another controller, pass control to another controller, pass control to itself, or take control synchronously.

In summary, the following set of interface functions contained in the IEEE Standard 488-1975 document completely describe the 4051 GPIB specifications:

SH1	—source handshake
AH1	—acceptor handshake
TE3	—talker
LE1	—listener
SRØ	—service request
RLØ	—remote local
PPØ	—parallel poll
DCØ	—device clear
DTØ	—device trigger
C1, 2, 3, 4, 28	—controller

WHAT HARDWARE AND LOGIC DESCRIBE THE INTERFACE ?

GPIB DEVICES

The General Purpose Interface Bus (GPIB) of the 4051 Graphic System was designed as a convenient, easy to implement, and powerful communications link between the 4051 and compatible devices, as well as a link between the devices themselves. The interface contains all of the mechanical, electrical and functional specifications required by IEEE Standard 488-1975 which describes a standard digital interface for programmable instrumentation. The compatibility of the 4051 GPIB with IEEE 488 was previously discussed. GPIB devices can take on three status designations: controllers, talkers, and listeners.

Controllers

The Graphic System acts as the controller and is the device which assigns who is going to transmit data (talker) and who will receive data (listeners). There can be only one controller at a time and it services all interrupts from the other devices. The Graphic System assumes that it is the only controller on the bus and it has complete control over the direction of all data transfers. There is no provision in the Graphic System for other devices on the GPIB to take turns as controller-in-charge.

Peripheral devices on the GPIB are designated as talkers and listeners. The Graphic System acts as the controller to assign peripheral devices on the bus as listeners and talkers. Asynchronous communications of device addresses and data occur on an eight-line data bus at the rate of the slowest assigned listener.

Talkers

A talker is a device capable of transmitting information on the Data Bus. There can be only one talker at a time. The Graphic System has the ability to assume the role of the talker when it is programmed to do so. The talking rate is usually governed by the listeners which "handshake" on every byte (character) that is transmitted by the talker. Clearly, the net communications rate can not exceed the maximum transmission rate of the talker.

Listeners

A listener is a device capable of receiving information transmitted over the Data Bus. There may be up to fourteen listeners taking part in an I/O operation at any one time. The Graphic System has the ability to assume the role of a listener anytime it is programmed to do so. The communications rate is generally limited by the maximum receive rate of the slowest assigned listener.

Idle Devices

A device need not be a talker or a listener at all times. It may be idle. Any device which has not been addressed since the last untalk (UNT) and unlisten (UNL) commands is in an idle state and has no effect on data communications rates. Therefore, devices need not be powered down in order to maximize transfer rates. In fact, it should be remembered that more than half of the devices on the bus must be powered up for the system to operate.

THE GPIB CONNECTOR

The GPIB connector is located on the rear panel of the Graphic System main chassis. This connector allows external peripheral devices to be connected to the system. The devices must conform to IEEE Standard 488-1975. The GPIB connector is a standard 24-pin connector such as an Amphenol Micro-Ribbon connector, with sixteen active signal lines and eight interlaced grounds. The cable attached to the GPIB connector must be no longer than 20 meters maximum with no more than fifteen peripheral devices connected at one time. The connector pin arrangement and signal line nomenclature is shown in Fig. 1-2.

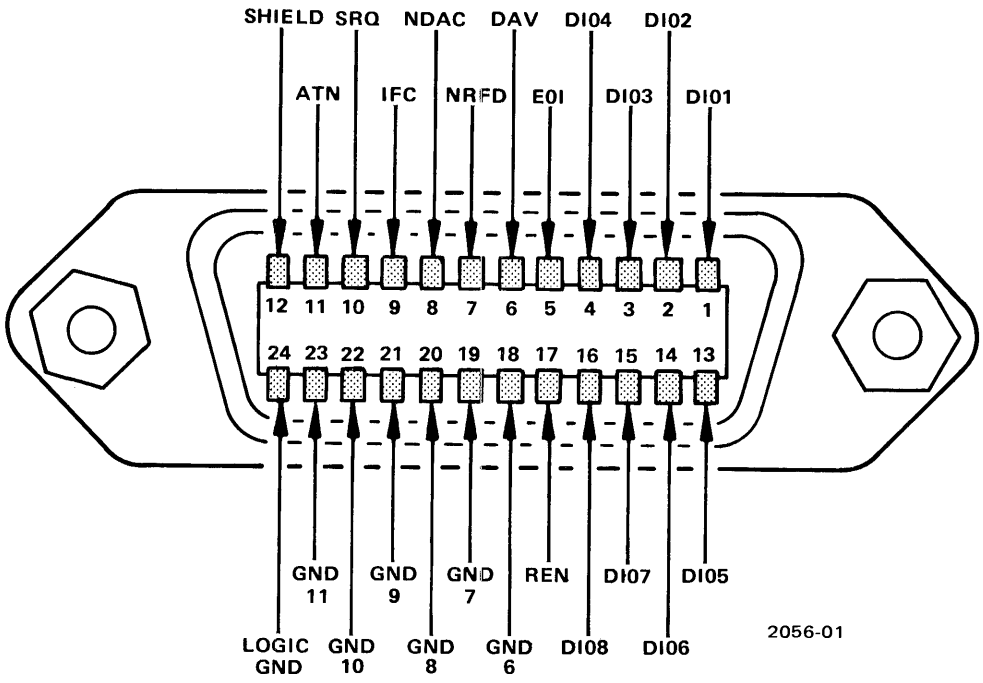


Fig. 1-2. GPIB connector.

Interconnecting cable assemblies are provided with both a plug and a receptable connector type at each end of the cable to allow either star or bus-structured systems. Connectors may be stacked rigidly using standard counterbored captive screws.

THE GPIB INTERFACING CONCEPT

The GPIB is functionally divided into three component buses; an eight-line Data Bus, a three-line Transfer Bus, and a five-line Management Bus for a total of sixteen active signal lines. This bus structure is shown in Fig. 1-3.

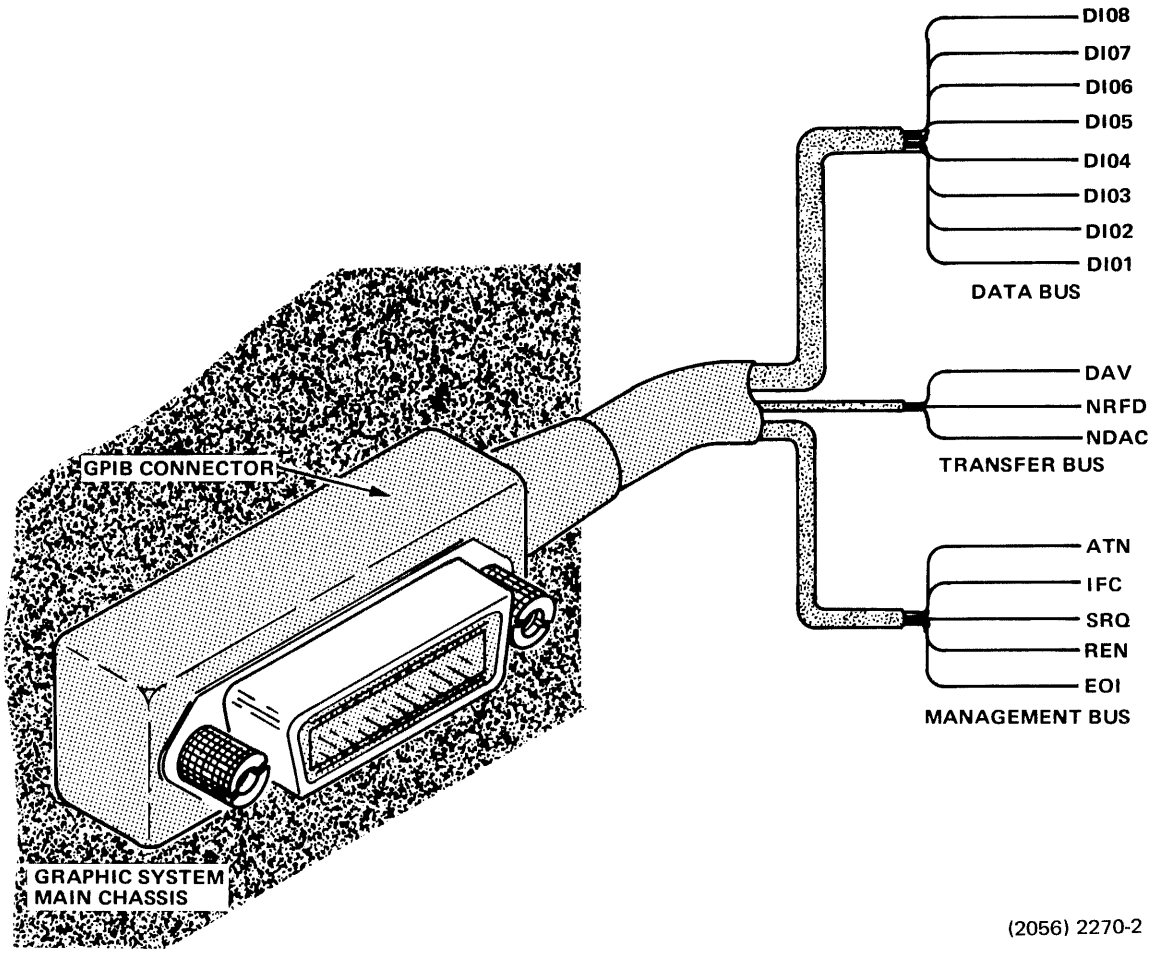


Fig. 1-3. GPIB component buses.

The transfer rate over the Data Bus is a function of the slowest peripheral device taking part in a transfer at any one time. The bus operates asynchronously. Both peripheral addresses and data are sent sequentially over the Data Bus. Once peripheral addresses are established for a particular transfer, successive data bytes may be transmitted in a burst for higher effective data rates. Within the 4051 Graphic System, data rates are dependent on the operation being performed and on the conversion time from the ASCII code on the GPIB to the machine dependent binary coding within the 4051. Effective data transfer rates are discussed in detail in Section 5 of this manual.

GPIB SIGNAL DEFINITIONS

Data Bus

The Data Bus contains eight bidirectional active-low signal lines, D101 through D108. One byte of information (eight bits) is transferred over the bus at a time. D101 represents the least significant bit in the byte; D108 represents the most significant bit in the byte. Each byte represents a peripheral address (either primary or secondary), a control word, or a data byte. Data bytes can be formatted in ASCII code, with or without parity (the Graphic System assumes no parity), or they can be formatted in machine dependent binary code.

Management Bus

The Management Bus is a group of five signal lines which are used to control data transfers over the Data Bus. The signal definitions for the Management Bus are as follows:

Attention (ATN) — This signal line is activated by the controller when peripheral devices are being assigned as listeners and talkers. Only peripheral addresses and control messages can be transferred over the Data Bus when ATN is active low. After ATN goes high, only those peripheral devices which are assigned as listeners and talkers can take part in the data transfer. The Graphic System assumes it is the only source of this signal. The use of the attention line is governed by user-written programs.

Service Request (SRQ) — Any peripheral device on the GPIB can request the attention of the controller by setting SRQ active low. The controller responds by setting ATN active low and executing a serial poll to see which device is requesting service.

This response is generated by an ON SRQ THEN statement which is executed in the BASIC program. The serial poll is taken when a POLL statement is executed in the BASIC program. After the peripheral device requesting service is found, BASIC program control is transferred to a service subroutine for that device. When the service subroutine is finished executing, program control returns to the main program. The SRQ signal line is reset to an inactive state when the device requesting service is polled. The Graphic System is not interrupted if another service request (SRQ active low) occurs during the service subroutine. This is not an error condition and the controller responds to the second SRQ on returning to the main program. Interruptable routines may be generated in software by resetting the interrupt feature with an ON SRQ THEN statement. See the section on handling interrupts in the 4051 Graphic System Reference manual.

Interface Clear (IFC) — The IFC signal line is activated by the 4051 when it wants to place all interface circuitry in a predetermined quiescent state. The Graphic System assumes that it is the only source of this signal. IFC is activated each time the INIT statement is executed in a BASIC program.

Remote Enable (REN) — The REN signal line is activated whenever the system is operating under program control. REN causes all peripheral devices on GPIB to ignore their front panel controls and operate under remote control via signals and control messages received over the GPIB.

End or Identify (EOI) — The EOI signal can be used by the talker to indicate the end of a data transfer sequence. The talker activates EOI as the last byte of data is transmitted. When the 4051 is listening, the 4051 assumes that a data byte received is the last byte in the transmission, if EOI is activated. When the 4051 is talking, it always activates EOI as the last byte is transferred except during a WRITE operation. The 4051 issues EOI at the end of WBYTE transmissions when it is user set. All transmissions other than WBYTE also end transmissions with the UNTalk and UNListen commands for use by devices which do not honor EOI.

Transfer Bus

A handshake sequence is executed by the talker and the listeners over the Transfer Bus each time a byte is transferred over the Data Bus. The Transfer Bus lines are defined as follows:

Not Ready for Data (NRFD) — An active low NRFD signal line indicates that one or more assigned listeners are not ready to receive the next data byte. When all of the assigned listeners for a particular data transfer have released NRFD, the NRFD line goes inactive high. This tells the talker to place the next data byte on the Data Bus.

Data Valid (DAV) — The DAV signal line is activated by the talker shortly after the talker places a valid data byte on the Data Bus. An active low DAV signal tells each listener to capture the data byte presently on the Data Bus. The talker is inhibited from activating DAV when a listener holds NRFD active low.

Data Not Accepted (NDAC) — The NDAC signal line is held active low by each listener until the listener captures the data byte currently being transmitted over the Data Bus. When all listeners have captured the data byte, NDAC goes inactive high. This tells the talker to take the byte off the Data Bus.

HOW DOES THE THREE WIRE HANDSHAKE WORK ?

INTRODUCTION

Consider an elegant system comprising hardware and software and a shared COMMON area that all devices can not only see, but monitor constantly. This COMMON area contains three types of information: data, control logic variables, and data transfer logic variables. The data constitutes the information to be transferred between the hardware components in the system in some well understood coding format, perhaps ASCII code.

We have up to 15 devices on the common bus at one time and must somehow keep them communicating. We address each device with numbers set manually on each device. We allow the devices to have one or more of three possible status capabilities. All devices will not necessarily want to receive all information on the bus at all times. Therefore we have:

TALKERS — Those devices which are the source of data at any given time and which have the capability of putting data out onto the bus and into the DATA area. (Only one device is allowed to talk at one time in order to eliminate possible confusion.)

LISTENERS — Those devices that accept or read the information in the DATA area. Any number of devices can be listeners at any one time, each of which behave as an input device.

CONTROLLER — The device that assigns TALK and LISTEN status to the other devices on the bus. We should also note that this is a relatively unique use of the word "controller" as this device is only assigning status and is never programming devices or reading their measurements. The controller here cannot make decisions and should not be confused with "process controllers" popular in industrial environments.

Examples of CONTROLLERS, LISTENERS, and TALKERS:

Devices that can Control, Talk and Listen, such as a programmable calculator or 4051. These devices control, perhaps by telling a 4924 Digital Cartridge Tape Drive to FIND file 8 and to start TALKING its stored information out onto the bus as DATA. The 4051 can also tell another device like the 4662 Digital Plotter to simultaneously LISTEN to the 4924 Tape Unit and plot the data that the Tape Unit is Talking. The 4051 is also capable of being a talker and has perhaps output that data which is now stored by the 4924 Tape Unit. The 4051 can also adopt listener status and input data from the same tape.

Some devices can't control, but can both Talk and Listen. The 4924 Digital Cartridge Tape Unit is one such device. In the world of instrumentation, so is a digital multimeter (such as a HP 3490A, Dana 5900, or Fluke 8500). Such a multimeter can adopt listener status and input a new change of measurement function (voltage, current, resistance) and range (100 mV full scale) and then adopt TALKER status to send the measurement value into the common area as data to be used by other devices.

Some devices can only be Listeners. A paper tape punch listens to data on the bus, accepts the data and punches it out on tape. A signal generator (such as a HP 330B, Fluke 6010 or 6011, or Wavetek 152 or 159) is also only a Listener. The waveform, frequency, and amplitude are dictated on the controller which lells the signal generator to listen. The information is talked by the controller and listened to by the signal generator which changes function, frequency, and amplitude and outputs these as an analog signal to some outside device.

Some devices can only be talkers. A paper tape reader is one such device. It reads a paper tape, then outputs or talks it onto the bus as data to be used by the assigned listeners. A digital counter (such as the HP 5345A or Dana 9000) inputs an analog frequency and displays this frequency as a number. It can also talk by putting this same number on the bus as data for use by the listeners.

THE COMMON AREA

The devices constantly scan the contents of the COMMON area and are always aware of the status of the following logic variables.

Data

The data is present as an ASCII character (8 parallel bits) and is placed there at the maximum rate of the talker. The talker is not allowed to put a new character in its place until the slowest listener has read it. This is in contrast to typical data communications systems where the bits come through serially at a clocked "Baud Rate". More on this later.

Control

We have need for 5 CONTROL or "General Interface Management" logic variables. These variables are either a "1" for "on" or a "Ø" for "off". In the routine transfers of data, the variables are relatiavely unimportant. What they do lend the system, though, is a considerable amount of flexibility. The variables are as follows:

IFC — "Interface Clear" usually remains off (Ø) and bothers no one. On system power-up or initialization, IFC is set to one (1) by the CONTROLLER to tell all the devices on the GPIB to set their status to a predetermined quiescent mode. Once everyone is initialized (and has said so) IFC goes into its usual Ø state. In the 4051, IFC is set to "1" every time the INIT command is executed in BASIC.

ATN

"Attention" is the most important variable. Without some special signal or flag, how can a LISTENER and TALKER know to cease its current activity and listen to the CONTROLLER for new TALKER/LISTENER assignments? ATN is usually "Ø," but whenever the CONTROLLER wants to get in its say, the CONTROLLER sets ATN to "1" and all of the devices cease their operation and listen. Besides a flag for assigning LISTENER and TALKER status, ATN is used by the 4051 when INIT is executed. First, the INIT command is recognized by the 4051. Next the value of IFC is set to "1" by the controller. All devices constantly observe the COMMON area and note this call from the CONTROLLER. All devices see IFC become a "1", so they reset themselves to a predefined state.

SRQ

"Service Request" is the opposite of ATN, that is, SRQ lets the CONTROLLER know that some device wants to talk. Not every device can set SRQ and allow the present processes to be interrupted. The 4051 CONTROLLER constantly monitor the status of SRQ when an ON SRQ THEN statement is executed in BASIC and honors the device request when SRQ is set it equal to "1." The 4051 is said to "support single level interrupts" or "honor interrupts". SRQ is a single variable equal to "Ø" or "1" and does not remember who set SRQ to "1". The capacity for honoring service requests is generally present only in sophisticated CONTROLLERS. Briefly, the CONTROLLER must:

1. Constantly monitor SRQ.
2. On finding SRQ equal to "1", stop all present activity.
3. Store all current device status.
4. POLL each device, according to a software command to see who called for its attention.
5. Honor the device's request via a user-written service request subroutine.
6. Reload device status from before interrupt (SRQ) condition.
7. Continue with previous process as if nothing had occurred.

The 4051 is one such sophisticated GPIB CONTROLLER.

REN

"Remote Enable" is a signal which acts as a safety latch for many devices. Some devices can be operated either manually or remotely from a distant station by program control. It is sometimes advisable to "lock out" the front panel of equipment if it is to be used solely in a programmable fashion. REN is set to "1" when GPIB bus is active. The devices which have this lock-out feature see that they are supposed to be under the control of some system component and not the whims of mortal man and summarily ignore their front panel switches and knobs. REN is automatically set to "1" by the 4051 CONTROLLER when the 4051 is placed under program control. It is not uncommon for a front panel lamp, such as "REMOTE", to light up on the GPIB device when it is in this mode.

EOI

"End or Identify" is one of the nicer things about the GPIB. Although not always implemented, those devices that use EOI are a joy to have in the system. Because EOI is a separately monitored variable, the TALKER can set EOI equal to "1" when the last byte of data is sent and tell the listener that there is no more data. This is the same as sending a CR delimiter to mark the end of the logical record. The 4051 terminates the data acquisition and continues with the next program line when: the current variables have assigned values and EOI is detected.

Another device that honors EOI is the TEKTRONIX 4662 Digital Plotter. The byte of data sent from the 4051 is accompanied by setting EOI equal to "1". The 4662 then knows that it has received the last byte of information.

Thus, with the five presently implemented control variables: IFC, ATN, SRQ, REN, and EOI we have the signals (or flags) required for the CONTROLLER to call (ATN) devices, reset (IFC) devices, to be interrupted (SRQ) asynchronously by devices, and to lock-out (REN) manual operation of devices. We also give TALKERS the ability to delimit transmissions (EOI) without having to send a delimiter character or character string.

DATA TRANSFER VARIABLES

The beauty of the GPIB design is its facility for devices to communicate effortlessly regardless of their transmitting and receiving rates. The process that allows this is the now famous "Three Line Handshake" for which the Hewlett-Packard Company holds the patent.

During a data transfer, the TALKER controls only one logic variable DAV, or "Data Valid". Whenever the TALKER places data in the common DATA area, for all LISTENERS to use, the TALKER sets DAV equal to "1". If the data isn't valid, or if the TALKER is in the process of updating the data, DAV is set equal to "Ø".

The LISTENERS have control over two logic variables, NRFD ("Not ready for Data") and NDAC ("Data Not Accepted"). These two signals let the TALKER know when new data bytes can be supplied and when the talker may erase the previous data, respectively.

Each LISTENER has to OK the transfer of each data byte. Because of this, NRFD and NDAC have inputs from every LISTENER and are thus not simple variables.

In fact, let us consider NRFD and NDAC as two arrays each with 31 elements:

DIM NRFD (31), NDAC (31)
where NRFD goes from NRFD (Ø) thru NRFD (30)
and NDAC goes from NDAC (Ø) thru NDAC (30)

Furthermore, when we speak of the value of NRFD or NDAC, we refer to NRFD (Ø) and NDAC (Ø) which summarize the status of the entire array. Remember that valid GPIB addresses go from 1 to 30 and that we may have no more than 15 devices on the bus at one time, each with a different address. We now assign NRFD (n) and NDAC (n) to device number "n". We always have extra array elements not being used and set these elements equal to "Ø". In fact, all values of the arrays NRFD and NDAC not pertaining to active GPIB LISTENERS are set equal to "Ø". The summary values NRFD (Ø) and NDAC (Ø) are the logical OR of the array members:

NRFD (Ø) = NRFD (1) OR NRFD (2) OR ... OR NRFD (30)
and NDAC (Ø) = NDAC (1) OR NDAC (2) OR ... OR NDAC (30)

Rather than signifying something to do, NRFD and NDAC hold up the communications process to wait for slowpoke LISTENERS on the bus. Using this technique, the TALKER can put an 8-bit byte in the DATA area at which time the LISTENERS can start reading it. As long as a LISTENER has not finished reading the data (has not accepted it, NDAC (n) = 1) the value of NDAC (Ø) remains 1 and the TALKER cannot update or modify the data.

(An alternative learning technique could involve using the terms RFD, Ready For Data, and DAC, Data Accepted, as well as inverting the meaning of the Ø to 1 designations. RFD (Ø) and DAC (Ø) would then be the Logical AND of the RFD (n) and DAC (n) terms, respectively. Although less obvious, the NRFD and NDAC terms are used because they are part of the IEEE-488 standard and have more trivial counterparts in the actual hardware than would RFD and DAC.)

HANDSHAKE PARAMETER STATES

Once the CONTROLLER has assigned the TALKER and LISTENERS the data transfer may commence. The only activity on the bus should be the changing of DATA accompanied by the Data Valid variable (DAV) going from Ø to 1 and the alternate switching of the NRFD (Ø) and NDAC (Ø) parameters from Ø to 1 and back to Ø again with the LISTENERS request for data (using NRFD) and, summarily, acceptance of data (using NDAC).

Two common errors may occur during the transfer. If a LISTENER requests data (NRFD = Ø, NDAC = 1) and the TALKER never responds, which means not admitting that valid data is on the bus (DAV = Ø), the bus hangs up. As there is no defined clocking rate for data, the LISTENERS think that they have a slow TALKER on the line and just sit there waiting. The difference between a slow talker and a dead one is a mater of degree. Thus, the controller cannot check for this error. This occurs during an incorrect INPUT, READ, OLD, or APPEND from an existing device or a request to TALK addressed to a LISTEN-only device.

The second common error is more insidious. NRFD and NDAC are binary and the two can thus have a total of four different configurations (11, 10, 01, 0,0). Three are legal states. The "0,0" state is a LISTENER error.

The "1,1" State

In the "1,1" state the LISTENER is Not Ready For Data (NRFD=1) and has Not Accepted Data (NDAC=1). This is a legal state for the LISTENER as it is indicating to the interface that it is not yet prepared internally to continue with the handshake cycle. If any LISTENER has the 1,1 configuration (NRFD (n)=1 and NDAC (n)=1), then communication on the bus is temporarily held up. All LISTENERS enter and leave communications modes in the 1,1 state.

The "Ø,1" State

In the "Ø,1" state the LISTENER is Ready For Data (NRFD=Ø) and therefore Not Accepting Data (NDAC=1). This is a legal state for any and all LISTENERS as it indicates to the interface and TALKER that the LISTENERS are prepared to receive messages.

The "1,Ø" State

In the "1,Ø" state, the LISTENER is Not Ready For Data (NRFD=1) because it is in the process of Accepting Data (NDAC=Ø). This is certainly a valid state as the LISTENER is indicating to the TALKER to maintain a valid byte of data. In the "1,Ø" state the LISTENER is indicating to the TALKER that it has received a data byte and is processing it.

The "Ø,Ø" State

The "Ø,Ø" state is always present in at least one device (the TALKER) but is not valid in an *assigned* LISTENER. In the "Ø,Ø" state the LISTENER is Ready For Data (NRFD=Ø) and is in the process of Accepting Data (NDAC=Ø). The first signifies to the TALKER to get rid of the present data and the second says to retain it as the data is still being read. The TALKER should recognize this as an error.

Remember that the TALKER does not see the status of each LISTENER, but only the logical OR of all the NRFD (n) and all the NDAC (n) elements. If any LISTENER has its NRFD or NDAC parameter set to "1", the TALKER will not recognize the presence of a "Ø,Ø" state in another LISTENER. Sophisticated GPIB LISTENERS will generate an EOI or SRQ if both its NRFD and NDAC are "Ø" when the device is in the LISTENER mode. Many do not.

HANDSHAKE SEQUENCE

The concept of assigning TALKERS and LISTENERS will be handled later. Let us now consider the actual handshake sequence involving a TALKER (in control of parameter DAV) and some LISTENERS (controlling the communications rate with parameters NRFD and NDAC.).

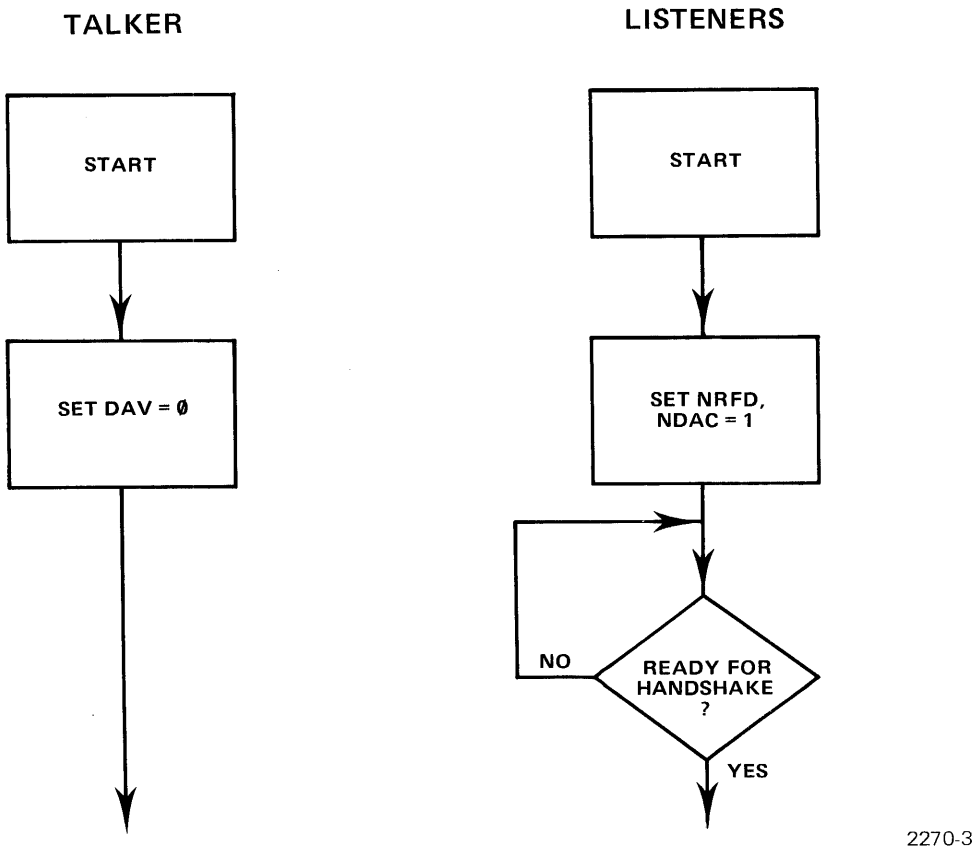


Fig. 1-4(a). Talker sets DAV inactive (Ø) to start the sequence. Listeners set NRFD and NDAC to 1.

1. On initialization and just after assignment of TALKER and LISTENER status, the TALKER initializes DAV = Ø (data not valid). Fig. 1-4(a).

2. The LISTENERS initialize NRFD = 1 (none are ready for data) and set NDAC = 1 (none have accepted the data). The LISTENERS will hold up the system until they feel that they are able to handshake and respond to data. Fig. 1-4(a).

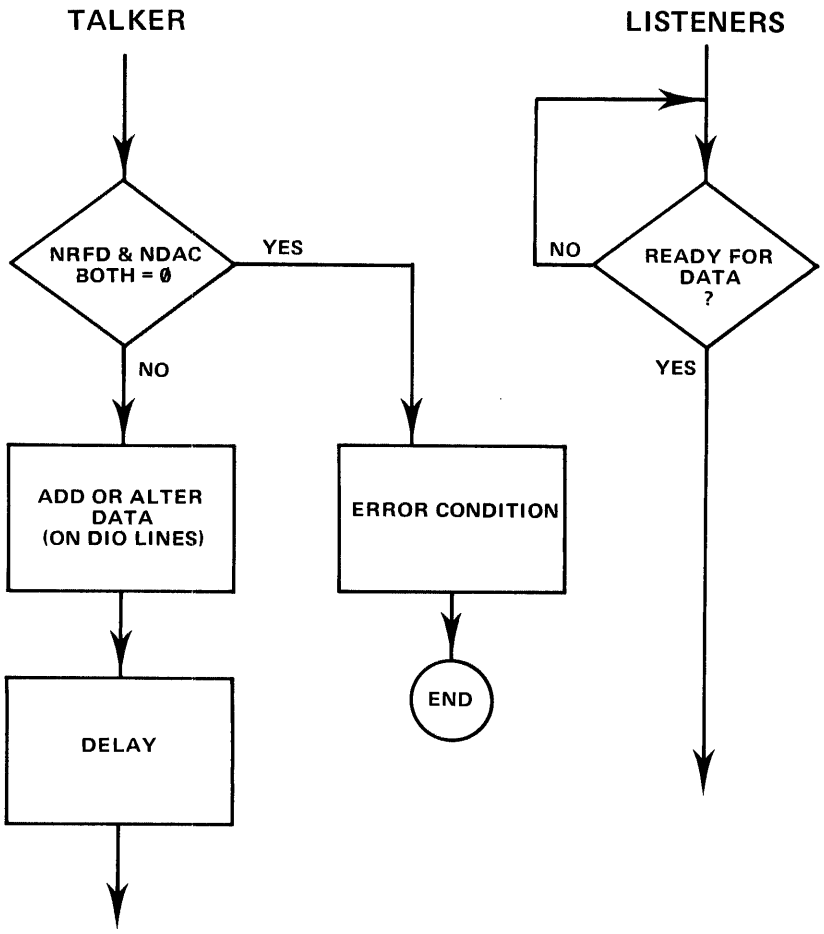


Fig. 1-4(b). Talker checks for an error, then places data on the data bus and waits.

3. The TALKER checks for the "Ø,Ø" status error condition (both NRFD and NDAC = Ø), then places the DATA in the common area. In reality, the data is placed on 8 parallel lines known as the DIO (Data Input/Output) lines. Fig. 1-4(b).

4. The TALKER then delays to allow the data to "settle" on the DIO lines. Meanwhile, the LISTENERS have initialized themselves and are capable of handshaking. They now wait until they are ready to accept data. Fig. 1-4(b).

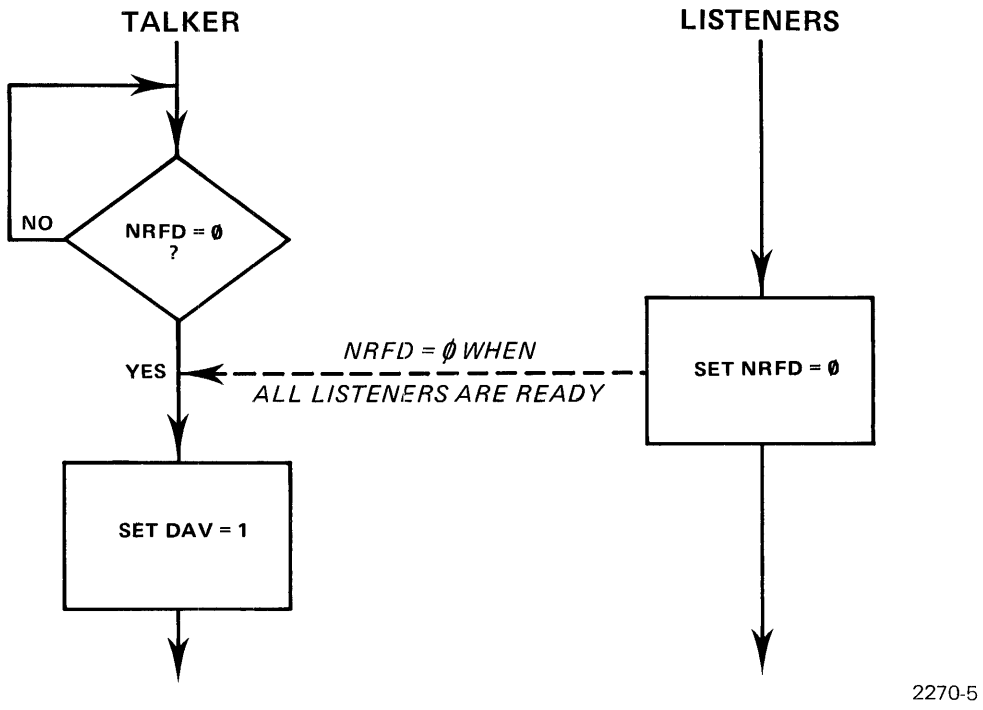


Fig. 1-4(c). Talker sets DAV active when the listeners are ready.

5. All LISTENERS have now indicated readiness to accept the first data byte (all NRFD (n) = Ø, therefore) NRFD = Ø. Fig. 1-4(c).

6. The TALKER, upon sensing NRFD = Ø, sets DAV = 1 to indicate that data is settled and valid. Fig. 1-4(c).

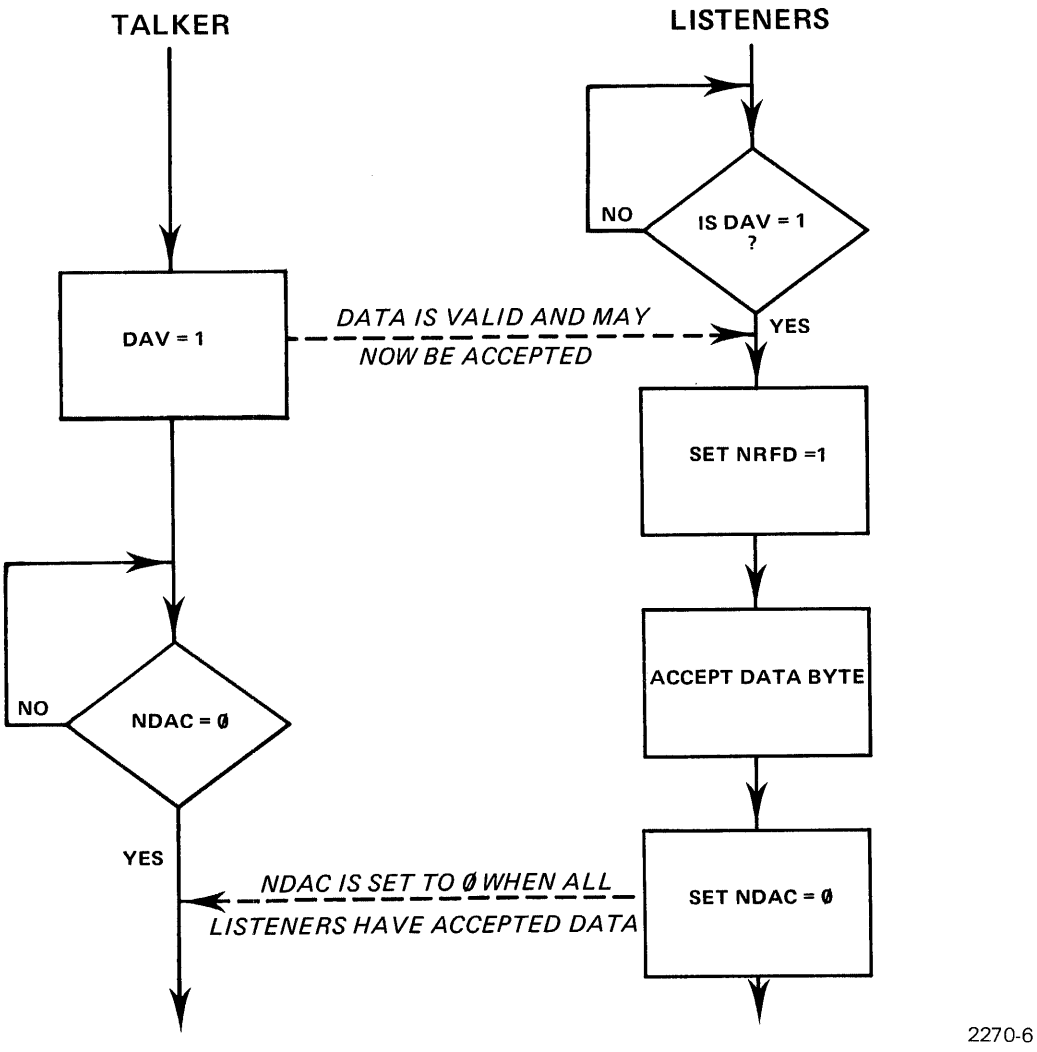


Fig. 1-4(d). Listeners set NRFD active (1), accept data, then set NDAC inactive (Ø).

7. The first LISTENER sets NRFD =1 to indicate that it is no longer ready, then accepts the data. The other LISTENERS follow at their own rates. Fig. 1-4(d).

8. The first LISTENER sets NDAC = Ø to indicate that it has accepted the data. (NDAC remains = 1 because the other LISTENERS still have NDAC (n) = 1.) Fig. 1-4(d).

9. The last LISTENER sets NDAC (n) = Ø to indicate that it has accepted the data; all have now accepted and NDAC = Ø. Fig. 1-4(d).

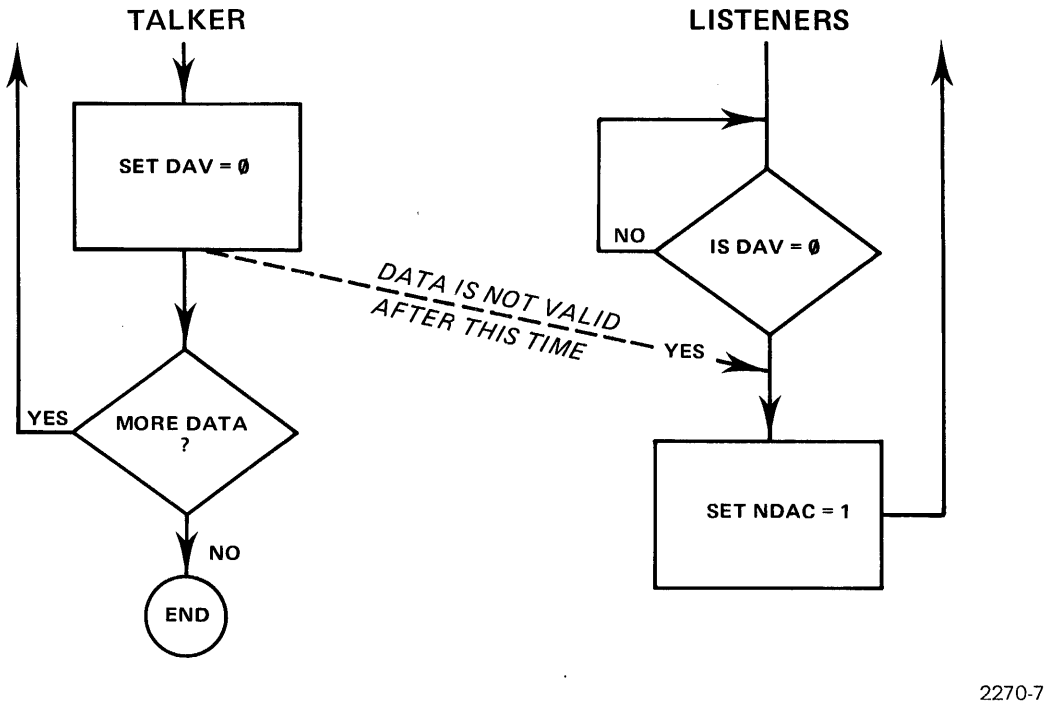


Fig. 1-4(e). Talker sets DAV inactive (Ø) and the handshake cycle is repeated or ended.

10. The TALKER, having sensed that NDAC = Ø, sets DAV = Ø. This indicates to the LISTENERS that the data (on the DIO lines) must now be considered not valid. Fig. 1-4(e).
11. TALKER changes data (on the DIO lines).
12. TALKER delays to allow data to settle (on DIO lines).
13. LISTENERS, upon sensing DAV = Ø, set NDAC = 1 in preparation for next cycle. NDAC = 1 as soon as the first LISTENER sets NDAC (n) = 1. Fig. 1-5.
14. The first LISTENER indicates that it is ready for the next data byte (character) by setting NRFD (n) = Ø. (NRFD remains = 1 due to other LISTENERS causing NRFD (Ø) = 1.) Fig. 1-5.
15. The last LISTENER indicates that it is ready for the next data byte by setting NRFD = Ø. NRFD (Ø) is now equal to Ø. Fig. 1-5.
16. The TALKER upon sensing NRFD = Ø, sets DAV = 1 to indicated that data on the DIO lines is settled and valid. Fig. 1-5.
17. The first LISTENER sets NRFD = 1 to indicate that it is no longer ready, then accepts the data. Fig. 1-5.
18. The first LISTENER sets NDAC (n) = Ø to indicate that it has accepted the data as in (8). Fig. 1-5.
19. The last LISTENER sets NDAC = Ø to indicate that it has accepted the data. Fig. 1-5.
20. The TALKER, having sensed that NDAC (Ø) = Ø, sets DAV = Ø. Fig. 1-5.
21. The TALKER removes the data byte from the DIO signal lines after setting DAV = Ø.
22. The LISTENERS, upon sensing DAV = Ø, set NDAC = 1 in preparation for the next cycle. Fig. 1-5.
23. Note that all three handshake signals, DAV, NRFD and NDAC are at their initialized states as in (1) and (2). Fig. 1-5.

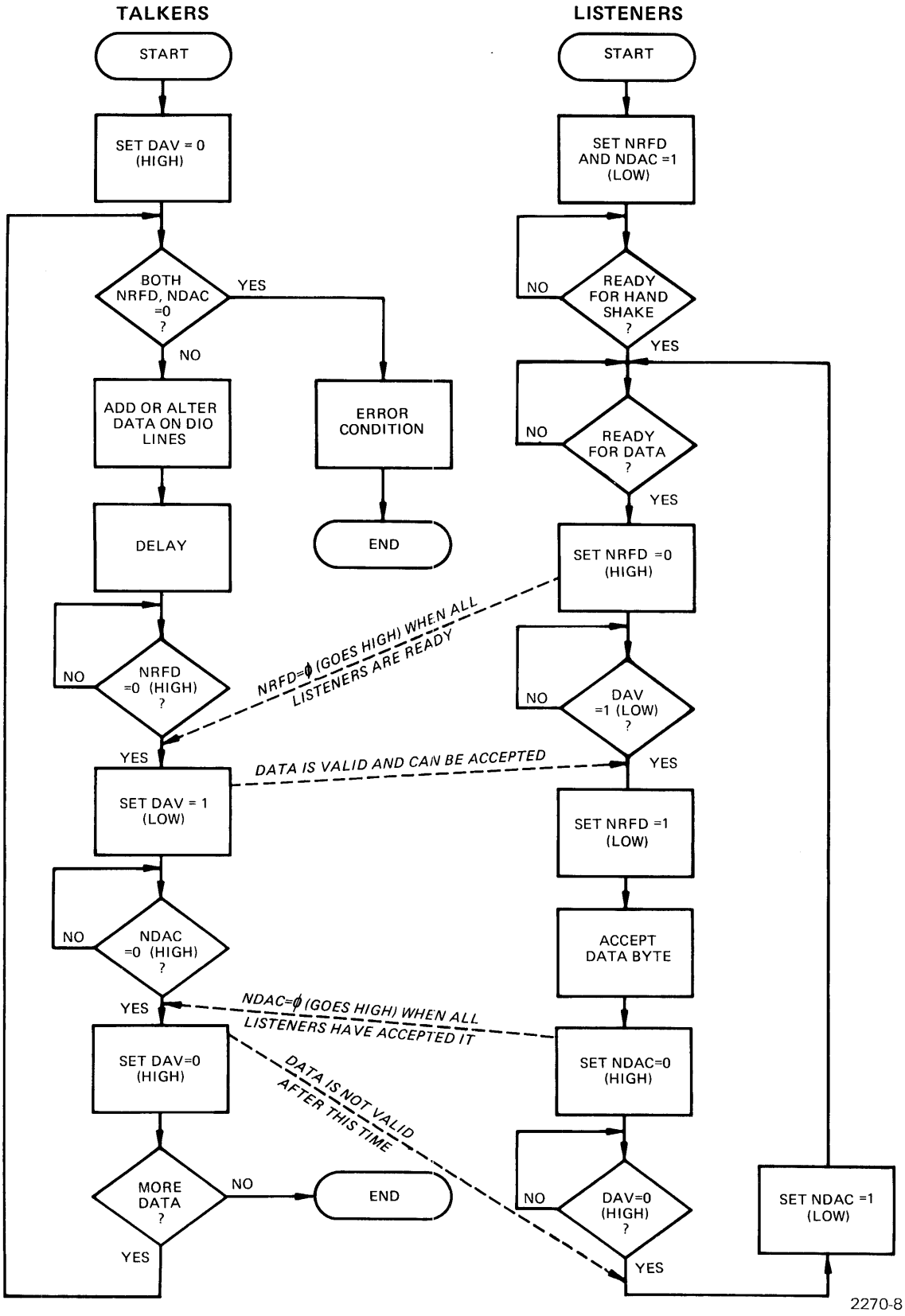


Fig. 1-5. Complete GPIB handshake flow chart.

HOW THE 4051 GPIB INTERFACE CONTROLLER IS IMPLEMENTATED

The implementation of the GPIB protocal in the 4051 is done in the firmware. Firmware is the 4051's internal microprocessor program that has been coded and "locked" into a silicon chip. User access to the GPIB is done through software, that is, programming the 4051 in the BASIC language.

Because the 4051 GPIB interface is programmable, the interface is discussed in software terminology. One of the most important concepts is the Three Wire Handshake, which was just discussed at length.

Section 2

4051 GPIB HARDWARE INTERFACE DESIGNS

INTRODUCTION

The information in this section is directed toward individuals who want to design their own 4051 GPIB interface and don't know where to start. This material picks up where IEEE Standard 488-1975 leaves off. A conceptual block diagram of a typical GPIB interface is discussed first. This interface is straightforward and allows a peripheral device to listen, talk, and respond to serial polls on the 4051 GPIB. After the block diagram discussion, the individual circuits in the interface are discussed in detail. A workable schematic of each circuit is provided along with an Integrated Circuit list to help you if you want to build the circuit. We hope that the material in this section is beneficial to you and that it will help you gain a better understanding of GPIB interfacing design concepts.

INTERFACE BLOCK DIAGRAM DESCRIPTION

INTRODUCTION

Diagram A is located in the Diagrams section at the back of this manual. This diagram presents a conceptual view of a general purpose interface that listens, talks, and responds to a serial poll on the 4051 GPIB. (Pull out this diagram now and look at it.) The text which follows describes how the interface works starting with the listen handshake circuitry and ending with the serial poll circuitry. Keep in mind that the term "interface" refers to the hardware components that allow a peripheral device to talk and listen to the 4051 over the GPIB. The purpose of the interface is to act as a go-between that matches the peripheral devices data transfer protocol with the protocol used on the 4051 GPIB.

THE LISTEN FUNCTION

The LISTEN function of the interface in Diagram A operates in one of two modes. In the first mode, the interface listens only to the controller; in the second mode, the interface listens only to the current data transfer.

Listening to the 4051 Controller

The interface is designed to respond almost instantly to the controller when ATN is set active low, regardless of the state of the peripheral device. This design feature allows the controller to clear the interface at the end of a data transfer and set up a new transfer with other devices, even though the peripheral device itself may be busy digesting data and unable to respond.

When the 4051 sets ATN low, the interface immediately comes on the bus in a ready state and is prepared to receive and decode addresses. The interface listen handshake circuitry handshakes on every data byte from the 4051 and the address from the 4051 is taken off the GPIB data bus and passed to the address decoder circuits. If the address is a valid address for the interface, the address decoding circuits send a signal to the interface memory where the address is "remembered." When ATN is released by the 4051, the interface enters the state currently recorded in the memory. This particular interface enters an IDLE state on system power-up, a TALK state when a preassigned primary talk address is received from the 4051, a LISTEN state when a pre-assigned primary listen address is received, and a POLL state when the controller command SPE (serial poll enable) is received from the 4051.

Once the interface is in one of these states, the interface participates in the bus activity until the 4051 controller sets ATN active low and issues UNTALK if the interface is talking, UNLISTEN if the interface is listening, or SPD (serial poll disable) if the interface is in a POLL state. Anytime IFC (interface clear) is set active low by the 4051, the interface returns to the IDLE state, regardless of the present state of the interface. (IFC is set low when an INIT statement is executed in BASIC.)

Listening to a 4051 Data Transfer

When a preassigned primary listen address is issued to the interface and ATN is released (made inactive) by the 4051, the interface stays on the bus ready to receive data bytes and send them to the peripheral's data input bus. At this time, the signal LISTEN TO ME is made active to tell the peripheral device's handshake signal lines GRAB IT, GOT IT, and I'M BUSY, and makes these handshake signals part of the GPIB handshake cycle. Here's how the interface listen handshake cycle works:

1. The interface listen handshake circuitry looks at the peripheral signal I'M BUSY to see if the peripheral device is busy. If I'M BUSY is true (active), the interface waits. If I'M BUSY is false (inactive), the interface sets NRFD high on the 4051 GPIB to tell the talker (the 4051 in this case) to send a data byte.

2. The 4051 responds by placing the data byte on the GPIB Data Bus. The data byte also appears on the peripheral's input data bus via the intefaces's Data Bus receivers. The talker (4051) then sets DAV (Data Valid) low—indicating that the data byte is valid and can be captured.

3. The interface listen handshake circuitry sees DAV go low and responds by setting GRAB IT active (true). This signal tells the peripheral device to latch the data byte off its input data bus and place the byte into a peripheral input buffer (a temporary storage location).

4. When the data byte has been successfully captured, the peripheral device sets the signal line GOT IT to an active true state. The interface handshake circuitry then responds by setting NDAC (Not Data Accepted) high on the GPIB. This tells the 4051 that the data byte has been successfully captured by the peripheral device.

5. The 4051 responds to NDAC by setting DAV high (inactive) and takes the data byte off the Data Bus.

6. In the meantime, the interface handshake circuits set NDAC low and prepare for the next handshake cycle.

7. The above action is repeated over and over as each data byte is transferred from the 4051 to the interface. At the end of the transfer, the 4051 activates ATN and issues UNLISTEN to the interface. When this happens, the address decoder in the interface sends a signal to the interface memory and the interface memory returns to the IDLE state. At the same time, the signal line LISTEN TO ME goes inactive and the peripheral device is free to go about its business.

8. Note that at any time during the data transfer, the peripheral device can freeze the activity on the GPIB by setting the signal I'M BUSY active true. This usually occurs when the peripheral's input data buffer gets full, or when the peripheral device stops to process information just received, or when the peripheral device honors a higher priority interrupt request within its own architecture. Note also that while ATN is down, the I'M BUSY signal has no effect on the interface. In fact, all the peripheral handshake signals are ignored and the peripheral device is effectively "cut-off" from the data transfer. This allows the interface to respond instantly to 4051 controller commands (no matter what the state of the peripheral) and allows the 4051 to clear the interface from the GPIB anytime the 4051 elects to do so.

THE TALK FUNCTION

The general purpose interface enters the TALK state when the 4051 issues a preassigned primary talk address to the interface with ATN down. At this time, the interface address decoder sends a signal to the interface memory telling the memory to "remember" to TALK. When ATN is released, the TALK TO ME signal goes active true and the interface enters the TALK state.

The TALK TO ME signal has three functions:

1. The signal tells the peripheral device to start sending data to the interface.

2. The signal enables the interface GPIB Data Bus drivers to work, and thus enables the interface to place data bytes on the GPIB Data Bus.

3. The signal enables the interface talk handshake circuits to work.

Once in the TALK state, the peripheral device starts sending data bytes to the interface; the interface, in turn, passes the data bytes to the assigned listener (in this case the 4051) over the GPIB. Here's how it happens (refer to Diagram A):

1. When ATN is released by the 4051, TALK TO ME goes active true. Assuming that the peripheral device is ready to talk immediately, the peripheral device places the first data byte on the peripheral output data bus. Since TALK TO ME also enables the interface Data Bus drivers, the data byte appears on the GPIB Data Bus at the same time.

2. After waiting the required minimum time for the data lines to settle (2 μs), the peripheral device sets the signal SHIP IT active true. This tells the interface that it's O.K. to transfer the data byte.

3. In the meantime, the interface talk handshake circuits look for the 4051 to set NRFD high. When NRFD goes high and when SHIP IT is true, the interface circuitry sets DAV (Data Valid) low on the 4051 GPIB.

4. The 4051 responds to DAV by setting NRFD low. The 4051 captures the data byte, then sets NDAC high.

5. The interface responds to the high going NDAC signal by setting DAV high (inactive) to tell the 4051 that the data byte is no longer valid. The interface then makes the signal IT'S GONE active true. This tells the peripheral device that the data byte has been successfully transferred and that it can get ready to transfer the next byte.

6. The peripheral device sees that IT'S GONE, then takes the data byte off the peripheral output data bus. The peripheral device then places the next data byte on the bus, waits for the lines to settle, then tells the interface to SHIP IT.

7. The interface responds as before and the handshake cycle is repeated.

8. This action continues until the 4051 sets ATN active low and issues UNTALK to the interface over the GPIB. If the 4051 is the listener, the UNTALK command is issued automatically at the end of the transfer (except for RBYTE operations). If the 4051 is not a listener, the talking peripheral device can set EOI or SRQ low on the GPIB for 350 μs (minimum). This causes the 4051 to branch to an ON EOI THEN statement or an ON SRQ THEN statement in the BASIC program, then to a WBYTE @95: statement that issues the UNTALK command to the interface. When the UNTALK command is received, the interface returns to the IDLE state and the signal TALK TO ME goes inactive. This tells the peripheral device that the transfer is ended.

RESPONDING TO A SERIAL POLL

If the peripheral device wants service from the 4051, the peripheral device must set the signal SERVICE PLEASE to an active true state. This causes the interface to set SRQ low on the 4051 GPIB. Since the SRQ line is shared by all peripheral devices on the bus, the 4051 has no way of knowing which device is holding SRQ low. The SRQ situation is normally handled in the current BASIC program in the following way.

When SRQ is set low, the 4051 BASIC interpreter looks at a previously executed ON SRQ THEN statement in the BASIC program to find out where to branch in the BASIC program. Program control is then transferred to the line number specified in the ON SRQ THEN statement. The line number is usually the line number of a peripheral device service routine (if there is only one device on the bus) or a POLL statement (if there are several devices on the bus).

If the ON SRQ THEN statement transfers control to a POLL statement, the 4051 executes a serial poll on the GPIB to find out which device is pulling down on SRQ. Each peripheral device specified in the POLL statement address list must respond to the 4051 during the serial poll. The interface in Diagram A responds to a serial poll as follows:

1. The 4051 starts a serial poll by setting ATN active low. The 4051 then issues an UNLISTEN command, followed by a SPE (serial poll enable) command.

2. The interface address decoder sees SPE and sends a signal to the interface memory to tell the circuits to "remember" that a serial poll is in progress. The interface responds by setting the signal POLL IN PROGRESS to an active true state.

3. The peripheral device responds to POLL IN PROGRESS by placing its status byte on the output data bus (lines D1-D8). If the peripheral device is requesting service, the device must indicate this condition by setting bit 7 in the status byte to an active (true) state. If the device is not requesting service, bit 7 in the status byte must be inactive (false).

4. Notice at this point that the status byte is not placed on the GPIB Data Bus. All activity stops here and the interface and the peripheral device wait until the interface is addressed as a talker by the 4051.

5. The point at which the 4051 addresses the interface as a talker depends on the position of the peripheral address in the POLL statement address list. (See the 4051 Graphic System Reference Manual for more information). If the address is first in the list, the interface will be addressed first, if the address is second in the list, the interface will be addressed second, and so on. When the interface's primary talk address is issued by the 4051, the interface enters the TALK state, as previously described, and TALK TO ME is made active true.

6. The signal TALK TO ME enables the interface GPIB Data Bus drivers to work. This places the peripheral status byte on the GPIB Data Bus. TALK TO ME also activates the interface talk handshake circuitry and the status byte is ready for transfer.

7. Soon after the 4051 issues the primary talk address, the 4051 assigns itself as a listener and releases ATN. The TALK handshake sequence then occurs (as previously described) and the status byte is transferred to the 4051.

8. After the status byte is transferred, the signal IT'S GONE is made active true by the interface. At this time, the peripheral device must set SERVICE PLEASE to an inactive state; this causes the interface to release SRQ.

9. If bit 7 in the status byte is set to a true state, the 4051 assigns the position number of the device to the first variable in the POLL statement and the decimal equivalent of the status byte to the second variable in the POLL statement. (Refer to the 4051 Reference Manual for details.) The 4051 then ends the polling sequence by activating ATN, and issues UNTALK and SPD (serial poll disable), in that order. This sequence returns the interface to an IDLE state and frees the peripheral device for further operations.

10. If bit 7 in the status byte is not set, the 4051 assumes that the peripheral device is not requesting service. The 4051 then sets ATN low and issues the primary talk address for the next peripheral device in the POLL statement address list.

11. The interface must interpret this new talk address as an "implied" UNTALK command and clear the interface from the talk state. When the 4051 finds the peripheral device that is requesting service, the interface is returned to an IDLE state by the commands UNTALK and SPD (serial poll disable) at the end of the polling sequence.

CLOSING REMARKS

This block diagram description presents an overview of the interface circuits which are about to be described in detail in the following paragraphs. An attempt has been made to keep the schematic diagram layout in Diagram B similar to the block diagram layout in Diagram A. If you feel that you are getting lost in the detail of the schematic—that is "not able to see the forest because of the trees", then take time to review the block diagram and its layout; get a "feel" for the overall picture of the interface, before diving back into the detail.

THE FIRST STEP IS HOOKING UP TO THE BUS

GPIB CONNECTOR REQUIREMENTS

Each peripheral device must be connected to the 4051 GPIB via a 24-pin female connector as shown in Fig. 2-1. The most common (and preferred) mounting position for the connector is the horizontal position on the rear panel of the peripheral device with pin 1 positioned in the upper right corner. The mechanical specifications for a GPIB connector are found on page 95 of the IEEE Standard 488-1975.

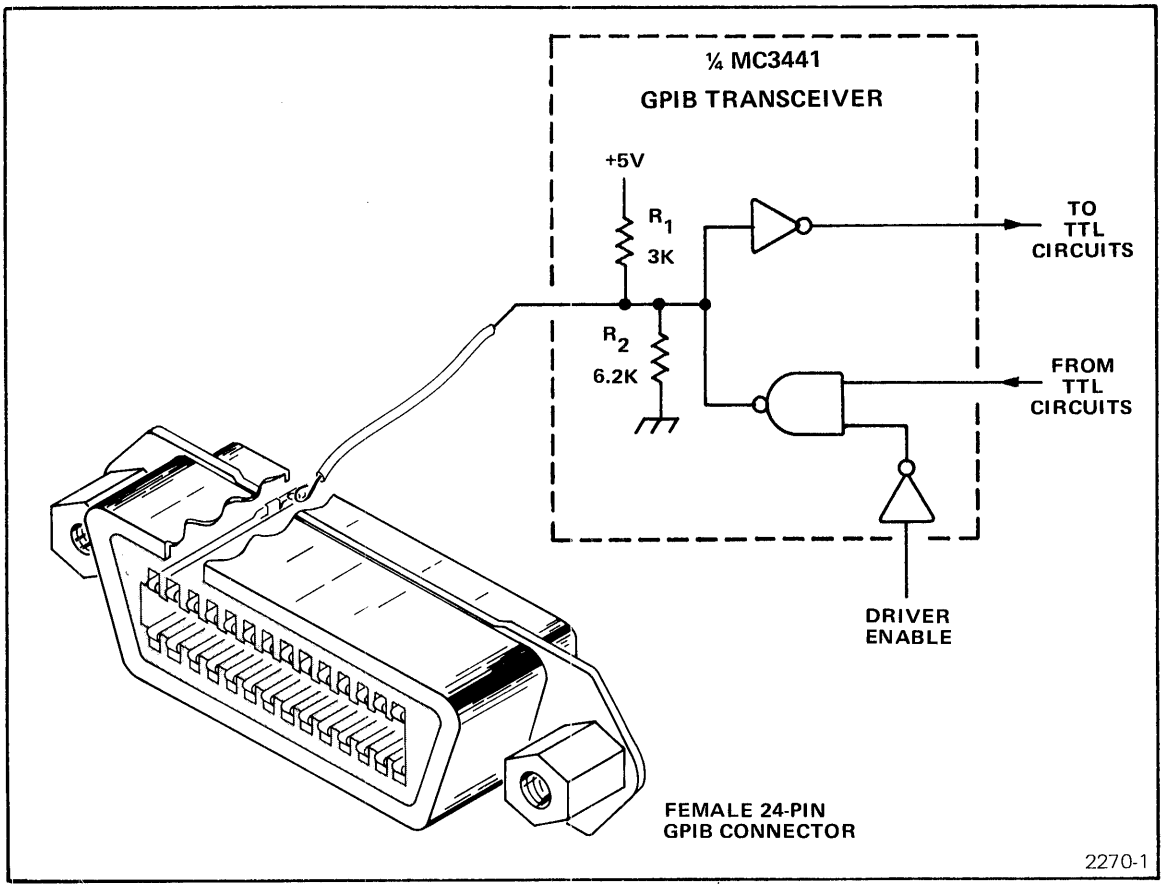


Fig. 2-1. GPIB connector and termination requirements.

GPIB TERMINATION REQUIREMENTS

If you are connecting a peripheral device to the GPIB, every signal line on the 24-pin connector must be terminated, regardless of whether the line is used or not. The standard line load configuration is shown in Fig. 2-1. Each bus line must be suspended between two resistors connected between ±5 vdc and ground. The top resistor (R_1) must be a 3K ohm resistor (±5%) and the bottom resistor must be 6.2K ohm resistor (±5%).

The most awkward way to handle the termination requirement is to use sixteen 3K ohm resistors and sixteen 6.2K ohm resistors, connecting them up two at a time to each bus line. This method takes a lot of room on the interface circuit board and we don't recommend it.

Another way to handle the problem is to use "resistor packs," which mount in 14-pin or 16-pin integrated circuit receptacles, and contain up to 16 individual resistors in each package. This method is acceptable, but there's a better way yet.

The best way to handle the termination problem is to use ready-made GPIB tranceiver chips which not only contain the termination resistors for each bus line, but also contain a bus driver and a bus receiver for each line. Besides saving a considerable amount of space on the interface circuit board, these chips provide a good electrical match between the GPIB connector and the TTL logic in the peripheral interface.

IF YOU DECIDE TO USE READY-MADE GPIB TRANSCEIVERS

Several companies manufacture transceiver integrated circuits which are specifically designed for GPIB applications. The transceiver chips used in this interface manual happen to be manufactured by Motorola Inc. and are used because they are available to us at the moment. There are undoubtedly other GPIB transceivers on the market that work just as well. We are not recommending any one transceiver over another. The choice is yours. Pick a transceiver that best suits your needs.

MAKING EFFICIENT USE OF GPIB HANDSHAKE TRANSCEIVERS

There are several GPIB transceiver chips in the MC3440 family and each chip has slightly different characteristics. Fig. 2-2 shows a GPIB handshake configuration using a MC3440 transceiver chip and a MC3441 transceiver chip.

There are four drivers and four receivers in each chip. The termination resistors are also in each chip, but are not shown. The only difference in the two chips is that the MC3441 (U2) has one driver which is independent from the common enable line controlling the rest of the drivers. The transceivers are designed so that the four receivers are always active (listening), but the four drivers are not enabled until a low is applied to pin 12 on each chip (except for one driver in U2).

Since the interface must drive part of the GPIB handshake signal lines and listen to the others during a LISTEN handshake, then reverse the role of each signal line during a TALK handshake, the trick is to arrange the transceiver output in such a way that the talk and listen state of the transceivers can be controlled with one signal line; and at the same time not have a receiver in the chip listening to its associated driver.

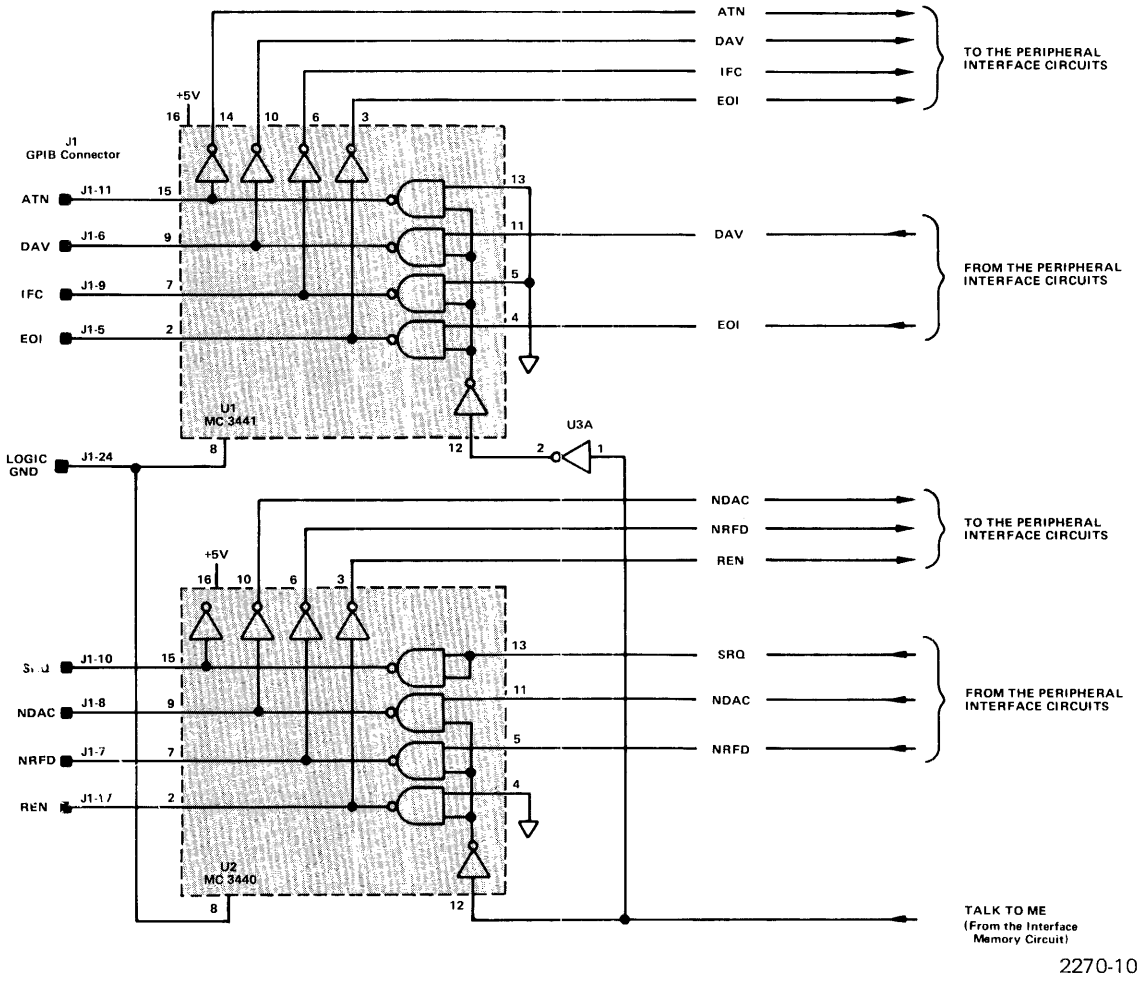


Fig. 2-2. Typical GPIB handshake transceiver configuration.

The transceiver arrangement in Fig. 2-2 works nicely and at the same time accommodates leftover control lines on the GPIB management bus. Two other MC3440 transceivers can be used to connect the GPIB Data Bus to the interface. This means that it only takes four chips to connect the entire interface to the GPIB connector. (The Data Bus arrangement will be shown in the schematics that follow.)

The following is a brief discussion on the transceiver arrangement in Fig. 2-2 and explains why signals are connected as they are.

TALK and LISTEN. During any GPIB handshake, the interface is driving some lines and listening to others. This means that one tranceiver must be in a receive mode while the other is in a transmit mode. It's not generally wise to use one chip for both purposes simultaneously, unless extra precautions are taken. Since the receivers are always receiving, the interface might end up listening to itself talk. Pin 12 on each transceiver turns the drivers on and off. These two inputs are controlled by one signal line called TALK TO ME. An inverter U3A is placed in-between the two inputs so they are always in the opposite state. When TALK TO ME is low (inactive), the drivers in U2 are enabled and the drivers in U1 are disabled. When TALK TO ME is high (active), the drivers in U2 are disabled and the drivers in U1 are enabled.

ATN (Attention). This signal comes in from pin 11 on the GPIB connector and connects to pin 15 on U1. Since the 4051 is the only device allowed to drive this signal line, pin 13 on U1 is grounded and the driver is permanently disabled. The interface can only listen to ATN at pin 14 on U1. Because the receiver is an inverter, pin 14 goes high when ATN is active low on the GPIB.

DAV (Data Valid). This signal comes from pin 6 on the GPIB connector and connects to pin 9 on U1. During a listen handshake sequence, the interface must listen to this line at pin 10 of U1. Pin 10 goes high when data is valid on the GPIB Data Bus and low when data is invalid. The low-to-high transition at pin 10 is normally used to clock a latch in the interface which captures the data byte.

When the interface is placed in the TALK mode, the signal TALK TO ME goes active and the DAV driver is enabled. The interface then drives the DAV signal line from pin 11 on U1 when the peripheral device places valid data on the GPIB Data Bus.

IFC (Interface Clear). This signal comes from pin 9 on the GPIB connector and connects to pin 7 on U1. Like ATN, this signal can only be driven by the 4051, so the U1 driver is grounded and permanently disabled. The interface listens to IFC at pin 6 on U1 and enters an IDLE state whenever pin 6 goes high.

EOI (End or Identify). This signal comes from pin 5 on the GPIB connector and connects to pin 2 on the U1 chip. Generally, this signal is undefined and can be used as the designer sees fit. However, in a 4051 system, EOI means END OF TRANSFER when the signal is activated as the last data byte in a transfer is placed on the bus. While the interface is listening, the interface can listen to EOI at pin 3 of U1. When the interface is talking, the peripheral device can activate EOI as the last data byte is transferred by setting pin 4 on U1 high. (This has the same effect as sending a CR delimiter to the 4051.).

SRQ (Service Request). This signal comes from pin 10 on the GPIB connector and connects to pin 15 on U2. Since only the 4051 can listen to this signal line, the bus receiver output for SRQ is not connected. The independently-enabled bus driver in the MC3440 chip is used to drive SRQ because the peripheral device must be able to set SRQ low and request service at any time, regardless of the state of the other interface tranceivers.

NOTE

According to the IEEE GPIB Standard: If several devices are connected to the GPIB bus, one more than 50% of the devices must be turned on (regardless of whether they are actually used), or the bus may be loaded down causing a spurious SRQ signal on the bus.

NDAC (Not Data Accepted). This signal comes from pin 8 on the GPIB connector and connects to pin 9 on U2. In the listen state, the interface must set this signal line low when a data byte is captured from the Data Bus. Since the drivers in U2 are enabled during a listen operation, the interface operates the NDAC signal line by setting pin 11 on U2 in a low state.

During a talk handshake operation, the interface must listen to NDAC to find out when the listeners on the bus have received the data byte being transferred. The interface listens at pin 10 on U2 and knows the data byte is transferred when pin 10 goes low.

NRFD (Not Ready for Data). This signal comes from pin 7 on the GPIB connector and connects to pin 7 on U2. During a listen operation, the interface sets NRFD high to tell the talker (or the 4051 controller) that the interface is ready for the next data byte. The interface sets NRFD high by placing a low on pin 5 on U2.

In the talk state, the interface must listen to NRFD to find out when the listener at the other end of the GPIB is ready for the next data byte. The interface listens at pin 6 of U2 and knows the listener is ready when pin 6 goes low.

REN (Remote Enable). This signal comes from pin 17 on the GPIB connector and connects to pin 2 on U2. Since this signal can only be driven by the 4051, the input to the driver is grounded and permanently disabled. The 4051 sets REN low when the 4051 is operating under program control. This makes pin 3 on U2 go high and tells the peripheral device to ignore its front panel controls and operate solely under the directions received from the 4051 GPIB.

Logic Ground. Pin 24 on the GPIB connector is the return path for all GPIB signals and should be connected to pin eight on all GPIB interface transceivers. This includes the transceivers for the GPIB Data Bus as well. A source of trouble can develope if the logic grounds on the transceivers are not properly connected.

Voltage to the Chips. A DC voltage of +5 volts must be connected to pin 16 on each transceiver chip. Each chip draws approximately 50 milliamps under normal operating conditions. Since the interface cannot draw power from the GPIB, the power must come from the peripheral device or a special power supply built specifically for the interface.

A Word of Caution. It is important to ground pins 13 and 5 on U1 and pin 4 on U2. If the input to one of these drivers is allowed to float high while the drive is enabled, a signal line (such as ATN) will be set low by the interface (instead of the 4051) and will cause havoc on the bus.

INTERFACE CIRCUIT DESCRIPTION

INTRODUCTION

A complete schematic diagram for a general purpose interface for the 4051 GPIB is located on the second pull-out in the Diagram section at the back of this manual; pull it out now. This schematic diagram gives a complete picture of the interface components along with an IC list. This diagram is provided as a convenience for you and you may remove it from the manual if you wish to do so.

For ease of illustration, the interface is broken into blocks and each block is discussed separately, starting with the listen handshake circuitry. The portion of the schematic on the pull-out to which the discussion pertains is repeated within this text; the schematic blocks within the text are slightly rearranged in some cases, with some components masked out to emphasize the active components in the circuit. The "U" numbers and the pin numbers of the components in these smaller schematics match the U numbers on the components in the complete schematic in Diagram B.

Circuit descriptions are sometimes tedious to read—especially if you are an experienced logic designer and can easily figure out how a circuit works by looking at it. The circuit descriptions do serve other purposes, however. It might be helpful to read through them at least once. The design criteria for each circuit is covered first, then an explanation follows on how the circuit design meets the design criteria. There are many subtleties in a GPIB interface design and many pitfalls to avoid. These subtleties and pitfalls are mentioned in the circuit descriptions when they apply. In addition, the purpose of logic components may not be obvious at first glance (like the "implied UNTALK" circuitry) and the circuit descriptions help remove the mystery surrounding these components.

INTERFACE LISTEN HANDSHAKE CIRCUIT DESIGN CRITERIA

The most important circuit in the interface is the LISTEN handshake circuit. Without the LISTEN handshake circuit, the 4051 can't communicate with the interface; the 4051 can't assign the peripheral device to be a listener or a talker and the interface can't respond to a serial poll.

The listen handshake circuit in an interface can be designed in several ways, but no matter how the circuit is designed, the circuit must meet the following design criteria:

1. **Response to ATN High (Inactive) when My Listen Address has not been Received.** The listen handshake circuitry in the interface should view this situation as an idle condition and get off the GPIB completely; that is, let all signal lines float high (inactive), just as though no device were present.

2. **Response to ATN Low when My Listen Address has not been Received.** This condition occurs on the GPIB when the interface is in an idle state and the 4051 starts a controller addressing sequence to assign peripheral devices as listeners and talkers. While ATN goes active low, the interface listen handshake circuit must get on the bus and handshake on every data byte transferred from the 4051. The address decoder in the interface must also leap into action and decode every address from the 4051—regardless of whether the address is meaningful to the interface or not. When the interface receives a meaningful address (with ATN low), the interface address decoder must continue to evaluate the additional address bytes coming in from the 4051 and not take action on the meaningful address until after the 4051 sets ATN high. This is important and the interface must not violate this rule!

The interface's initial response to a low-going ATN signal must be to set NDAC low and NRFD either high or low, depending whether or not the interface is ready to receive and decode addresses from the 4051. Whether the interface is ready to decode addresses or not, the interface must set NDAC low within 200 ns after ATN goes low to comply with the IEEE Standard.

3. **Response to ATN Being Low after My Listen Address has been Received.** This condition can occur two different times on the GPIB: (1) during the initial addressing sequence when ATN is low (true) and the primary listen address has been received and (2) at the end of a data transfer when the 4051 sets ATN low and prepares to issue an UNLISTEN command to the interface. In both cases, the interface must not allow the peripheral device to listen to the GPIB Data Bus while ATN is low. In the first case, the 4051 may still have addresses to issue (either primary or secondary) and the peripheral device might pick up these addresses as data bytes. In the second case, the interface may receive the UNLISTEN command from the 4051, but the peripheral device might also capture the UNLISTEN command and treat the command byte as the last data byte in the transfer. It is important to design the interface so that only the interface can listen to the GPIB when ATN is active low. The peripheral device must be effectively "cutoff" from the GPIB during this time.

4. **Response to ATN High when My Listen Address has been Received.** This condition on the GPIB tells the interface to capture all data bytes being transmitted over the GPIB and to pass the data bytes to the peripheral device as valid data. It is important for the interface at this point to honor the peripheral's handshake signal lines and the peripheral's busy signal. If the interface does not do this, the interface may start receiving data bytes faster than the peripheral device can take them and data will be lost. The peripheral's BUSY signal should be connected to the interface handshake circuitry in such a way that the peripheral device can freeze the activity on the GPIB anytime it elects to do so.

5. **Response to Interface Clear and an UNLISTEN Command from the 4051.** Anytime the 4051 sets IFC low on the GPIB, the interface must return to an IDLE state and get off the bus. And, if the interface is in a LISTEN state, and the 4051 issues an UNLISTEN command (decimal 63) with ATN active low, the interface must get off the bus.

A CIRCUIT THAT MEETS THE LISTEN HANDSHAKE DESIGN CRITERIA

Fig. 2-3 is a schematic diagram for a typical TTL listener handshake circuit. This circuit is identical to the listen handshake circuit block on the larger interface schematic in Diagram B. Although a circuit of this type can be designed in any one of a number of ways, the important thing is that this circuit be passive while other data transfers are taking place on the GPIB, and at the same time be ready to leap into action and listen to the bus as soon as the controller sets ATN active low. Here's how this response is built into the circuit.

Remaining Idle

As previously stated, the interface should be in an IDLE state when ATN is high and the interface's primary listen address has not been issued by the 4051. The listen handshake circuitry looks at the state of ATN at pin 14 on GPIB transceiver U1. When ATN is high (inactive) on the GPIB, pin 14 on U1 is low. The rest of the logic in the circuit is set up as follows.

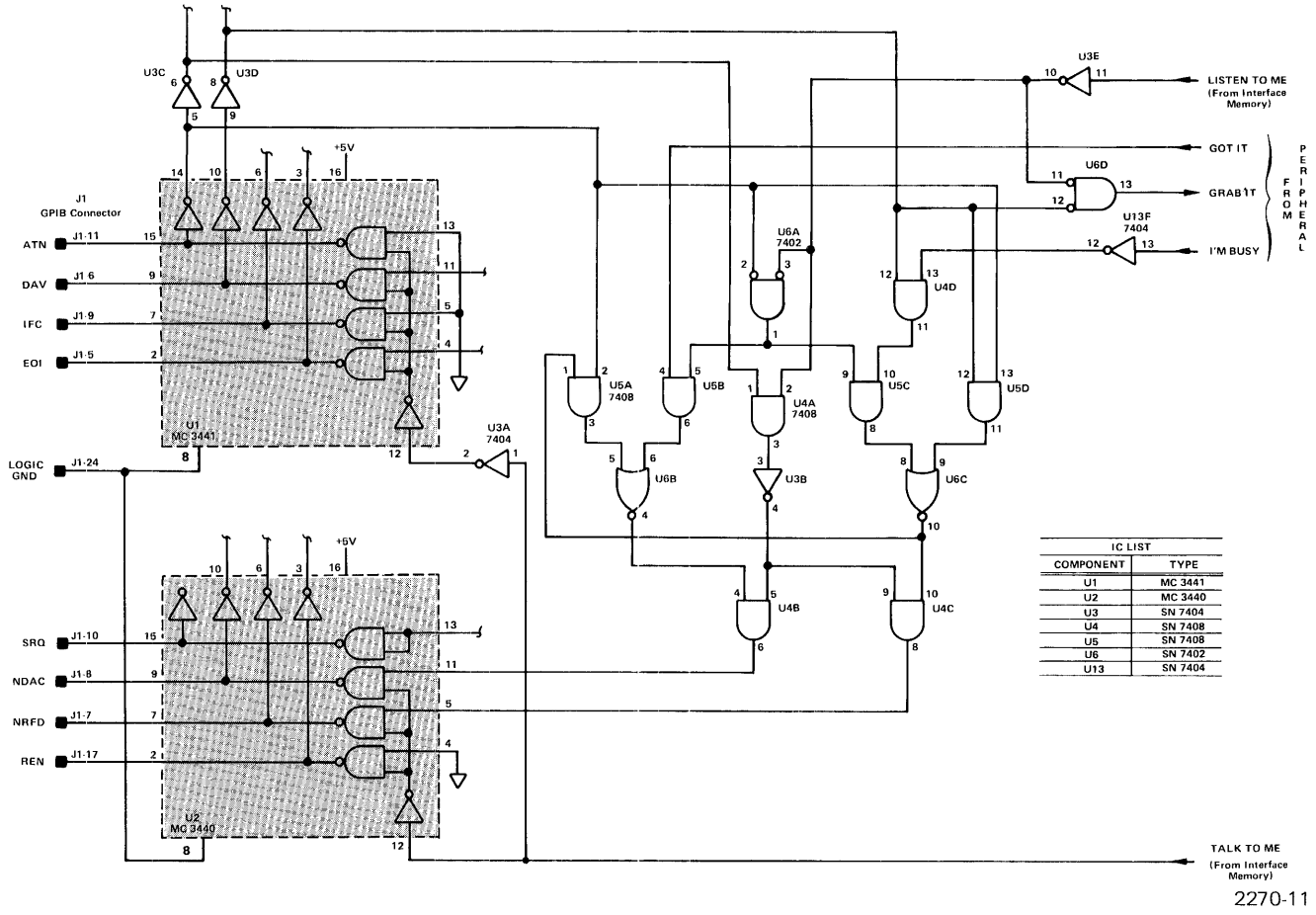


Fig. 2-3. Typical 4051 GPIB listen handshake circuit design.

The low at pin 14 on U1 makes the output of inverter U3C high which in turn makes pin 1 on AND gate U4A high (U4A is located in the middle of the circuit).

AND gate U4A is the key element which controls the active and idle state of this listen handshake circuit. Since the inactive state of ATN makes pin 1 on U4A high, and the inactive state of the LISTEN TO ME signal makes pin 2 on U4A high, the output of U4A goes high. (The LISTEN TO ME signal comes from the interface memory.) This LISTEN TO ME signal goes active only when the primary listen address for the interface is issued by the 4051). The high output of U4A makes the output of inverter U3B go low and keeps U4B and U4C disabled. (Notice that U4A and U3B together perform a NAND function. They were selected over a NAND gate in this case to make the most efficient use of leftover gates in the IC packages.)

U4B and U4C act as enable/disable gates for the signal lines NDAC and NRFD, respectively. With the output of U3B keeping one pin on each gate low, the output of U4B and U4C remain low which makes pin 7 and pin 9 on the GPIB transceiver U2 float high. The high condition of both NRFD and NDAC on the GPIB tells the 4051 that the interface is off the bus and essentially not present.

Addressing the Interface

The 4051 sets ATN low on the GPIB to issue addresses over the bus. The interface must respond immediately (within 200 ns) by setting NDAC low. Since this interface is designed to decode addresses anytime ATN is low (regardless of the state of the peripheral device), the interface sets NRFD high at the same time NDAC is set low. This tells the 4051 that the interface is immediately ready to decode addresses (provided another device on the GPIB is not keeping NRFD low). Before going into the details of how NRFD is set high and NDAC is set low, a brief overview of the listen handshake circuit is in order.

A Quick Schematic Overview

The interface listen handshake circuit is functionally divided into two groups of gates. The gates on the left of U4A control the interface's NDAC response. The gates on the right of U4A control the interface's NRFD response. Since the listen handshake circuit operates in two modes—one handshake mode with ATN active low and the other handshake mode during a data transfer—there are two AND gates in each group, one gate for each mode. When ATN is low and the 4051 is issuing address, U5A on the left controls the NDAC signal line; U5D on the right controls the NRFD signal line. During a data transfer when the peripheral device is listening, U5B on the left controls the NDAC signal line; U5C on the right controls the NRFD signal line.

Getting Ready to Handshake when ATN goes Low

Now back to the present situation. The 4051 sets ATN active low on the GPIB, and the following four actions occur in the interface listen handshake circuitry to get the interface ready to receive and decode addresses:

1. **Gates U6A, U5B, and U5C are immediately disabled and effectively removed from the circuit.** U6A is in the circuit to detect when the primary listen address is issued and ATN is high. This condition tells the interface to start listening to the GPIB Data Bus for valid data. Since ATN is low at this time and the controller is issuing peripheral addresses, this gate must be disabled. It is disabled in the following way. When ATN goes low, pin 14 on U1 goes high which makes pin 2 on U6A go high. This keeps the output of U6A low (disabled) which, in turn, keeps U5B and U5C disabled. With U5C disabled, U4D has no effect on the circuit either. (Later it will be shown how this circuitry handles the handshake during the data transfer with the assigned talker).

2. **Gates U4B and U4C are enabled which allows the interface to drive NRFD and NDAC.** When ATN goes low, pin 14 on U1 goes high, the output of inverter U3C goes low, and AND gate U4A is disabled. This makes pin 4 on U3B go high, which enables both U4B and U4C to pass the NRFD and NDAC signals to GPIB transceiver U2. The source of the NRFD signal is pin 10 on U6C and is passed by U4C to the input of the NRFD driver (U2, pin 5). The source of the NDAC signal is pin 4 on U6B and is passed to the input of the NDAC driver (U2, pin 11) by U4B.

3. **NRFD Remains Inactive High.** With U4D and U5C effectively out of the picture at this time, the NRFD signal is controlled by U5D. Pin 13 on U5D acts as the gate enable which goes active high when ATN is made active (low) on the GPIB. Therefore, NRFD is controlled directly by DAV (Data Valid) on the GPIB. At this time, data is not valid, so DAV is high on the GPIB. This causes pin 10 on U1 to go low and pin 8 on U3D to go high, thus making pin 12 of U5D go high. As a result, pin 11 of U5D goes high, pin 10 on U6C goes low, pin 8 on U4C follows along by staying low, and the output of the NRFD driver (U2, pin 7) stays high. This tells the 4051 controller that the interface is ready to receive the first address over the Data Bus.

4. **NDAC goes low.** At the same time U5D is enabled by the ATN signal, U5A is also enabled. The source of the NDAC signal is actually pin 10 on U6C. When U6C goes low, pin 1 on U5A goes low. This makes the output of U5A go low, which makes the output of U6B go high. Since U4B is enabled, the output of U4B follows the output of U6B and also goes high. This causes the NDAC driver output to go low, telling the 4051 controller that the interface has not yet accepted a data byte. The interface is now ready to capture and decode the first address from the 4051 controller.

Timing Delays Assuming a 15 ns delay occurs for signal transitions to pass through each logic gate in the circuit, it takes approximately 105 ns for the handshake circuitry to respond to ATN. This time delay is well within the 200 ns maximum response time which is specified in the IEEE Standard.

Handshaking with the 4051 While ATN is Active Low

With NRFD high and NDAC low on the GPIB, the interface effectively tells the 4051 controller "I am ready to receive the first address over the data bus". When all peripheral devices on the GPIB Data Bus are ready, the 4051 places the first address on the GPIB Data Bus and sets DAV low. When DAV goes low, pin 10 on U1 goes high; this indicates to the interface that the address on the Data Bus is valid and is ready to be captured and decoded.

Receiving and Decoding the Address Byte

Normally, a byte on the Data Bus is captured and decoded by an address decoder which is not part of the handshake circuitry. The active ATN signal is used to enable the address decoder to look at the Data Bus, and at the same time, the leading edge of DAV is used to clock the decoder into capturing (or reading) the address byte. This is usually set-up as a separate action, independent of the handshake sequence. Since the address decoder is discussed in detail later in this section, we'll keep our attention on the handshake circuitry for now. The point is this: due to the inherent delays in the handshake cycle (with the 4051 as the controller), the fastest time in which the handshake cycle can be completed is 18 μs. This provides an 18 μs "time window" for the address decoder to read and decode address bytes.

Taking advantage of this knowledge, the handshake circuitry is designed to proceed with the handshake as quickly as possible, even though the circuit doesn't know for sure that the address decoder is actively decoding the address. The handshake circuitry "assumes" that 18 μs is more than enough time for the address decoder to "do it's thing," so the handshake circuitry responds immediately to DAV by setting NRFD low, followed by setting NDAC high. It works nicely in this case and here's how it happens.

Setting NRFD Low

The interface circuit's first response to DAV must be to set NRFD low, as specified in the IEEE handshake flow diagram. This action is accomplished as follows. The low going DAV signal line causes pin 10 on U1 to go high. The output of inverter U3D responds by going low which causes the output of U5D to go low; the output of U6C goes high. This high is applied to pin 10 of U4C, and in combination with the high on pin 9 on U4C, makes the output of U4C go high. The high U4C output is fed to the input of the NRFD driver, and the output of the driver (U2, pin 7) goes low to make NRFD go low on the GPIB. All this happens in about 90 ns.

Setting NDAC High

The next step in the interface response to DAV is to capture the address byte on the Data Bus, then set NDAC high. Since the Address Decoding circuitry does this on the leading edge of DAV, enough time has already passed for the decoding circuits to complete the decoding operation. Therefore, the handshake circuit continues without stopping by setting NDAC high. This tells the 4051 that the data byte has been accepted.

The source of the NDAC signal is the NRFD signal with a few gate delay periods thrown in. When the NRFD signal goes high on pin 10 of U6C, pin 1 on U5A goes high. Since U5A is already enabled by the ATN signal on pin 2, the output of U5A goes high, causing the output of U6B to go low. The output of U4B follows by going low, which causes the output of the NDAC drive to go high on the GPIB.

This action happens quickly and NDAC is set high approximately 30 ns after NRFD goes low. NDAC going high tells the 4051 controller that the interface has received the address byte.

The 4051 Sets DAV High and Takes the Address Byte Off the Data Bus

As soon as NDAC goes high, the 4051 knows that all listeners have accepted the address byte; the 4051 now returns DAV to a high state (inactive) and takes the address byte off the Data Bus. In spite of the interface's fast response in setting NDAC high, it takes the 4051 18 μs to set DAV high and take the address byte off the bus. Because of this delay, there is an 18 μs "time window" for the address decoding circuits to look at the information on the Data Bus before the data is removed.

Resetting NRFD and NDAC

At this point in the handshake cycle, NRFD is low and NDAC is high on the GPIB. Since it was the low going DAV signal that set NRFD low and NDAC high in the first place, these two signals revert back to their original states when the 4051 sets DAV high again. NRFD is reset first by going high; NDAC follows 30 ns later and goes low.

But Wait Just a Minute...

It is interesting to note here that technically this action violates the IEEE Standard. NDAC should be set low first, then NRFD should be set high. For a split second (30 ns) both NRFD and NDAC are both high which indicates a possible error condition. But, in the interest of keeping this circuit simple for educational purposes, we have allowed it. And, because the 4051 is a microprocessor controlled device, and isn't fast enough to detect this 30 ns period when NRFD and NDAC are both high, we have allowed it. If this circuit were being drive by a controller which was able to detect a 30 ns time period when both NDAC and NRFD were high, then the controller might be justified to terminate the operation at this point; in which case, it would be necessary to add a few more logic gates and redesign the circuit so the NDAC goes low first, followed by NRFD going high.

Doing It Again and Again

With the 4051 as the controller, the 4051 keeps ATN held low until all the peripheral addresses in the current BASIC statement are issued over the GPIB. As far as the interface circuitry is concerned, this can be for an indefinite period, so the handshake circuitry keeps handshaking on every address byte, and the address decoder keeps looking for addresses that it recognizes. When the last address is issued and the 4051 releases ATN, the interface handshake circuits revert back to an idle state (NRFD and NDAC both high); that is, if the interface primary listen address was not received during the addressing operation.

When the Primary Listen Address is Received

If during the addressing operation, the 4051 issues the primary listen address for this interface, the address decoding circuits set LISTEN TO ME active high. This makes the output of inverter U3E go low and places a low on pin 3 of U6A and pin 2 of U4A.

It is important here that the interface circuits keep listening to the GPIB Data Bus and that they keep interpreting the data bytes as peripheral addresses and controller commands. If the interface allows the peripheral device to start listening to the bus after receiving the primary listen address, the peripheral might start interpreting the remaining address bytes as ASCII code (for example). The interface, therefore, must wait until after ATN goes high before passing the data bytes to the peripheral device as data.

AND gate U6A is in the circuit to make sure the interface waits for ATN to go high before capturing bytes on the Data Bus and treating them as data. U6A does this by disabling the data transfer handshake circuitry U5B and U5C when LISTEN TO ME is high (true) and ATN is low (true). Here's how it's done.

While the interface is in an idle state, the inactive ATN signal line on the GPIB keeps pin 2 on U6A low. Since the primary listen address has not been received, LISTEN TO ME remains inactive low which keeps the output of U3E high. This disables U6A and keeps pin 1 on U6A low. This low disables U5B and U5C and prevents these gates from taking part in the circuit action. During an addressing sequence, ATN goes low on the GPIB making pin 2 on U6A go high. This keeps the output of U6A low throughout the addressing operation, even if the interface's primary listen address is issued by the 4051 controller and the addressing decoding circuits set LISTEN TO ME high. When the addressing operation is over, ATN goes high again on the GPIB and pin 2 on U6A returns to a low state. The condition of LISTEN TO ME at this time then determines if the output of U6A goes high or low. If LISTEN TO ME is set high (true), then the output on U6A goes high which enables gates U5B and U5C to operate.

Handshaking During a Data Transfer

Introduction

When the interface is addressed to listen to a data transfer, the interface must keep handshaking after ATN goes inactive high on the GPIB. The interface must also honor the peripheral's busy signal and stop handshaking if the peripheral device gets too busy to handle additional data. The following paragraphs describe how the interface switches to a different set of gates to control the handshake during the data transfer and how the interface honors the peripheral handshake signals GRAB IT, GOT IT, and I'M BUSY.

Keeping U4B and U4C Enabled

If the interface's primary listen address is received while ATN is low on the GPIB, the address decoder and interface memory circuits set LISTEN TO ME high, as previously described. This high is inverted to a low by U3E and keeps pin 2 of U4A low. This disables U4A, even after pin 1 on U4A goes high when ATN goes high. Therefore, the output of U4A stays low and keeps the output of inverter U3B high; as a result, U4B and U4C remain enabled. This action allows the interface to keep driving NRFD and NDAC, even after ATN goes high on the bus and the assigned Talker starts sending data.

Keeping NDAC Low After ATN Goes High

The source of the NDAC signal during the data transfer is a signal called GOT IT which comes from the peripheral device. When GOT IT goes high true, it indicates that the peripheral device has successfully captured the data byte on the peripheral data input bus.

When ATN goes high after the initial addressing sequence, the first data byte isn't on the GPIB data bus yet, so the GOT IT signal is low (inactive). Even though U5B in the listen handshake circuit is enabled at this time by the high from U6A, the GOT IT signal keeps pin 4 on U5B low. This keeps the output of U5B (pin 6) low.

Both U5A and U5B in the NDAC circuitry are effectively disabled at this time, so the output of U6B remains high, the output of U4B remains high, and the NDAC driver in U2 keeps the NDAC signal line low on the GPIB. This tells the talker that the interface hasn't received data yet.

The end result of all this is that the interface switches from U5A and U5D to U5B and U5C as the NDAC and NRFD signal sources, respectively. The interface does this to bring GOT IT and I'M BUSY into the handshake cycle and it does this while at the same time maintaining a ready state on the GPIB by keeping NDAC low and NRFD high.

The Talker Places the First Data Byte on the Data Bus

After ATN goes inactive high at the end of the addressing sequence, the assigned talker checks to see if NRFD and NDAC are both high. If they are both high, it means nobody is out there listening; this is an error condition. If NDAC is low, the talker is allowed to place the first data byte on the GPIB Data Bus, let the data lines settle for at least 2 μs, then check to see if NRFD is high.

Placing the Data Byte on the Peripheral's Data Input Bus

Since the interface's Data Bus transceivers are always in a LISTEN state, (unless they are told to TALK), the data byte on the GPIB Data Bus automatically appears on the peripheral device's input Data Bus. (Refer to Diagrams A and B.)

If the Peripheral Device is Not Busy

If the peripheral device is not busy when ATN goes high, the peripheral device keeps I'M BUSY low (inactive), which keeps pin 13 on U4D high. At the same time, DAV is inactive high, which keeps pin 10 on U1 low, pin 8 on U3D high, and pin 12 on U4D high. With both inputs on U4D high, the output on U4D goes high. This high signal, along with the high signal from U6A keeps pin 8 on U5C high and the output of U6C low. The output of U4C goes low as a result and the output of the NRFD driver goes high (U2, pin 7). This, of course, tells the talker that the interface is ready to receive data immediately after ATN goes high.

If the Peripheral Device is Busy

If the peripheral device should get busy during the period when the primary listen address is received and ATN goes high, then I'M BUSY goes high. This makes pin 12 on U13F go low, which disables U4D. This in turn disables U5C which makes the output of U6C go high. U4C responds to this high by making its output go high which makes pin 7 of the NRFD driver go low. This tells the talker that the interface is not ready to receive data yet. When the peripheral is free to receive data, I'M BUSY goes low to make NRFD on the GPIB go high.

The Talker Sets DAV Low

After waiting at least 2 μs for the data lines on the GPIB to settle, and after checking to see that the NRFD signal line is in a high state, the talker sets DAV (Data Valid) low. This tells the interface that the data on the GPIB Data Bus is valid and can be captured.

The Interface Responds By Setting NRFD Low

The low going DAV signal on the GPIB triggers the listen handshake circuits to set NRFD low. Here's how it happens. Pin 10 on U1 goes high when DAV goes low. This causes pin 8 on U3D to go low which disables AND gate U4D. The low output of U4D disables U5C which causes the output of U6C to go high. The output of U4C also goes high which makes the output of the NRFD driver go low at pin 7 on U2. Setting NRFD low at this time is the first step in the listeners response to DAV (as specified in the IEEE Standard).

The Interface Tells The Peripheral Device to "GRAB IT"

When DAV goes low on the GPIB, pin 10 on U1 goes high as previously described. This makes the output of inverter U3D go low which makes pin 12 on U6D go low. (U6D is located on the right side of the listen handshake circuit.) Since the active LISTEN TO ME signal from the interface memory circuit is holding pin 11 on U6D low, the combination of two low inputs makes the output of U6D go high. This tells the peripheral device that the data byte on the peripheral's data input bus is valid and to GRAB IT.

The Peripheral Device Tells The Interface "I GOT IT"

When the peripheral device has successfully captured (or "latched") the data byte on its input bus, the peripheral device sets the signal GOT IT to an active (high) true state. This tells the interface that it's O.K. to proceed with the handshake sequence and set NDAC (Not Data Accepted) high on the GPIB.

The Interface Sets NDAC High

When the interface sees GOT IT go high, the interface relays the message to the GPIB talker by setting NDAC to a high state. Here's how it happens:

1. With the high output of U6A keeping pin 5 on U5B high, the GOT IT signal makes the output on AND gate U5B go high.

2. The high U5B output makes the output pin 4 on U6B go low.

3. This low is passed to pin 11 on GPIB transceiver U2 via AND gate U4B.

4. The low pin 11 on U2 turns off the bus driver and makes the NDAC signal on the GPIB go high. Assuming that another device on the GPIB is not holding NDAC low, the high-going NDAC signal tells the talker that the peripheral device has successfully captured the data byte.

The Talker Sets DAV High

As soon as the interface sets NDAC high, the talker responds by setting DAV high (inactive) on the GPIB. This means that the data byte on the GPIB Data Bus is no longer valid. The talker takes the data byte off the bus and places the next data byte on the bus.

The Interface Resets NRFD and NDAC And Prepares For The Next Cycle

When the talker sets DAV high, the interface listen handshake circuit reverts back to a ready state by setting NRFD high and NDAC low. Since the low DAV signal causes the handshake circuit to set NRFD low and NDAC high in the first place, the high going DAV signal causes a reverse action and returns NRFD and NDAC to their original states.

Here again, the reverse action of returning NRFD high before returning NDAC low causes a moment when both signals are high. This split-second action technically violates the IEEE Standard and the talker has the right to terminate the data transfer at this point. But, since the 4051 is unable to pick up this split-second violation and since this circuit is for educational purposes only, we have allowed it (primarily in the interest of keeping the circuit as simple and straight-forward as possible.)

Doing It Again And Again And . . . Again

If the talker is not busy, the talker is free to set DAV low as soon as the interface returns NRFD to a high state. This starts the next handshake cycle and the transfer sequence continues until the 4051, acting as the GPIB controller, sets ATN low and issues the UNLISTEN command. It is appropriate to point out here that anytime the peripheral device gets too busy to handle the data coming in, the peripheral device can set I'M BUSY active high and freeze the activity on the GPIB. Then, when the peripheral device is ready again, the peripheral can set I'M BUSY to an inactive low state and the interface can continue handshaking with the talker over the GPIB.

TALK HANDSHAKE CIRCUIT DESIGN CRITERIA

The Talk handshake circuitry is much simpler than the listen handshake circuitry and the design criteria is very simple—"talk only when you're told to." Fig. 2-4 illustrates a very simple talk handshake circuit design. This design is tailored specifically to take advantage of the 4051 timing characteristics on the GPIB and may not work when the interface is talking to other peripheral devices over the GPIB. An understanding of the timing sequence during a READ or INPUT operation with the 4051 is required, so it may help to refer to the READ and INPUT timing diagrams in Section 4 as you read this text.

A CIRCUIT THAT TALKS TO THE 4051

The circuit in Fig. 2-4 consists of two GPIB transceivers, one inverter, one D-latch, and one AND gate. The two transceivers are the same transceivers (U1 and U2) shown in the listen handshake circuit Fig. 2-3. The listen handshake circuitry has been masked out to emphasize the "talk" components of this handshake circuit.

The circuit is entirely controlled by the TALK TO ME signal that originates from the interface memory. (Refer to the interface block diagram to get a clear picture of where this TALK TO ME signal comes from.) When TALK TO ME is inactive (low), the D-input to U19A is low, pin 12 on U2 is low and pin 12 on U1 is high. This places the transceivers in the listen handshake configuration and keeps the outputs on D-latch U19A in a fixed, reset condition; Q output low, $\overline{Q}$ output high. It is interesting to note that any activity on the GPIB causes the clock input (pin 3) on the D-latch to pulse and the clear input (pin 1) to pulse. Nothing happens though, because the D input (pin 2) is low.

When the 4051 addresses this interface during a READ or INPUT operation, the 4051 issues the primary talk address, the secondary address for READ or INPUT, then holds down on NRFD and NDAC while it assigns itself as a listener. When the 4051 releases ATN, the 4051 keeps NRFD and NDAC low for 38 μs, then sets NRFD high.

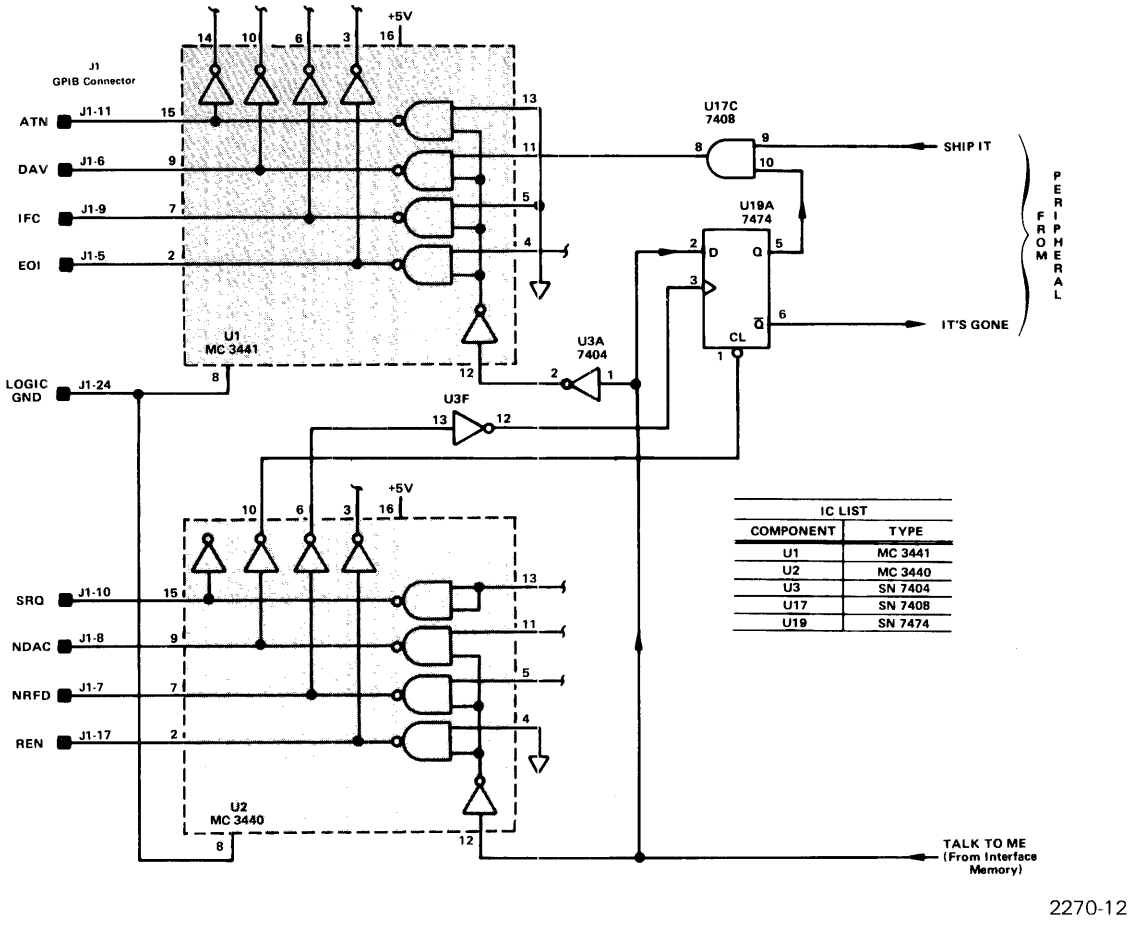


Fig. 2-4. Typical 4051 GPIB talk handshake circuit design.

This circuit takes advantage of this timing characteristic in the following way. When the 4051 sets ATN to a high (inactive) state, the interface memory circuits set the TALK TO ME signal high (true). This turns the bus drivers "on" in U2 and turns the drivers "off" in U1, thus placing the handshake circuitry in a TALK state. A high TALK TO ME signal is also placed on the D-input (pin 2) on U19A at this time, to "arm" the latch. The TALK TO ME signal also turns on the GPIB Data Bus drivers (see the block diagram) and tells the peripheral device to place the first data byte on its output data bus.

The Peripheral Device Talks To The Interface

The peripheral device responds to the TALK TO ME signal by placing the first data byte on the peripheral output bus; the peripheral waits 2 μs for the lines to settle, then sets the SHIP IT signal active (high). The SHIP IT signal enables AND gate U17C to pass the state of pin 5 on D-latch U19A to the DAV bus driver on the GPIB.

NOTE

This interface places the 2 μs waiting period burden on the peripheral device. If the peripheral device is unable to accomplish this, a 2 μs delay can be placed in the SHIP IT signal path by adding a flip-flop and a one-shop multivibrator.

The Interface Talks To The 4051—Setting DAV Low

When 38 μs have elapsed after ATN goes high, the 4051 sets NRFD high to indicate that it is ready to receive the first data byte. This low to high transition causes pin 6 on U2 to go low, which causes pin 12 on U3F to go high and clock D-latch U19A. Since a high is on pin 2 of U19A, the outputs on U19A change state; the Q output goes high and the $\overline{Q}$ output goes low.

Assuming that the peripheral device is on the ball and has the data byte on the bus and has SHIP IT set high, the high going Q output on U19A causes the output of U17C to go high. This high causes the output of the U1 driver to set DAV (Data Valid) on the GPIB to a low state, telling the 4051 that the data byte on the GPIB Data Bus is valid and can be captured.

The 4051 Sets NDAC High

When DAV goes low, the 4051 sets NRFD low, captures the data byte, places the byte in its input buffer, then sets NDAC high to indicate to the interface that the data byte has been received. This activity takes approximately 21 μs (minimum).

The Interface Sets DAV High And Tells The Peripheral Device To Continue

The high going NDAC signal on the GPIB causes pin 10 on U2 to go low. This makes pin 1 on D-latch U19A go low and clears the latch; the Q output goes low and the $\overline{Q}$ output goes high.

The low Q output on U19A causes the output of U17C to go low and the DAV signal line of the GPIB goes high (inactive). This tells the 4051 that the data byte is no longer valid.

At the same time, the $\overline{Q}$ output on U19A goes high making the IT'S GONE signal active true. This tells the peripheral device that the 4051 has received the data byte; it also means that the peripheral should place the next data byte on the data bus. The peripheral device places the next byte on the bus, then sets SHIP IT high and the handshake cycle is repeated.

The Transfer Ends When ATN Goes Low

The transfer continues in this fashion until the peripheral sets END OF TRANSFER active true with the last data byte on the bus or until the 4051 decides that it doesn't want any more data. When this happens, the 4051 sets ATN low, and the TALK TO ME signal immediately goes to a low state. This returns the handshake transceivers to a listen configuration, allows the interface to receive and decode the UNTALK addresses, and effectively "kills" the talk circuit. After the 4051 issues the UNTALK command with ATN down and releases ATN, the interface memory circuits return to an IDLE state and the TALK TO ME signal remains inactive (low).

ADDRESS DECODER DESIGN CRITERIA

The purpose of the interface address decoder is to evaluate each peripheral address and controller command issued by the 4051, and when the 4051 issues a meaningful address to the interface, the address decoder must recognize the meaningful address and immediately send a signal to the interface memory so the address can be remembered.

The interface address decoder must meet the following design criteria:

1. The address decoder must remain inactive when ATN on the GPIB is inactive (high).

2. Anytime ATN is active (low), the address decoder must evaluate **ALL** peripheral addresses and controller commands issued by the 4051.

3. And finally, the address decoder must send a signal to the interface memory circuits when a meaningful address or controller command is received over the GPIB.

A Conceptual View Of The Address Decoding Operation

It is appropriate to take a look at the concept on which the address decoder circuit is based before discussing the schematic diagram. The concept is not immediately obvious by looking at the schematic. Fig. 2-5 illustrates the concept on which the address decoding circuit is based. Each peripheral address coming over the GPIB Data Bus is issued as an eight-bit binary number. The GPIB data lines are separated into two groups with four lines in each group. Each group of four lines is fed into a four-line to sixteen-line decoder.

Each decoder has sixteen outputs—one output for each of the sixteen possible binary bit patterns on the four input lines. For every address issued over the GPIB, one output on each decoder goes active true. In order to detect a meaningful address, the two active outputs for the meaningful address are combined into one active true signal with an AND gate.

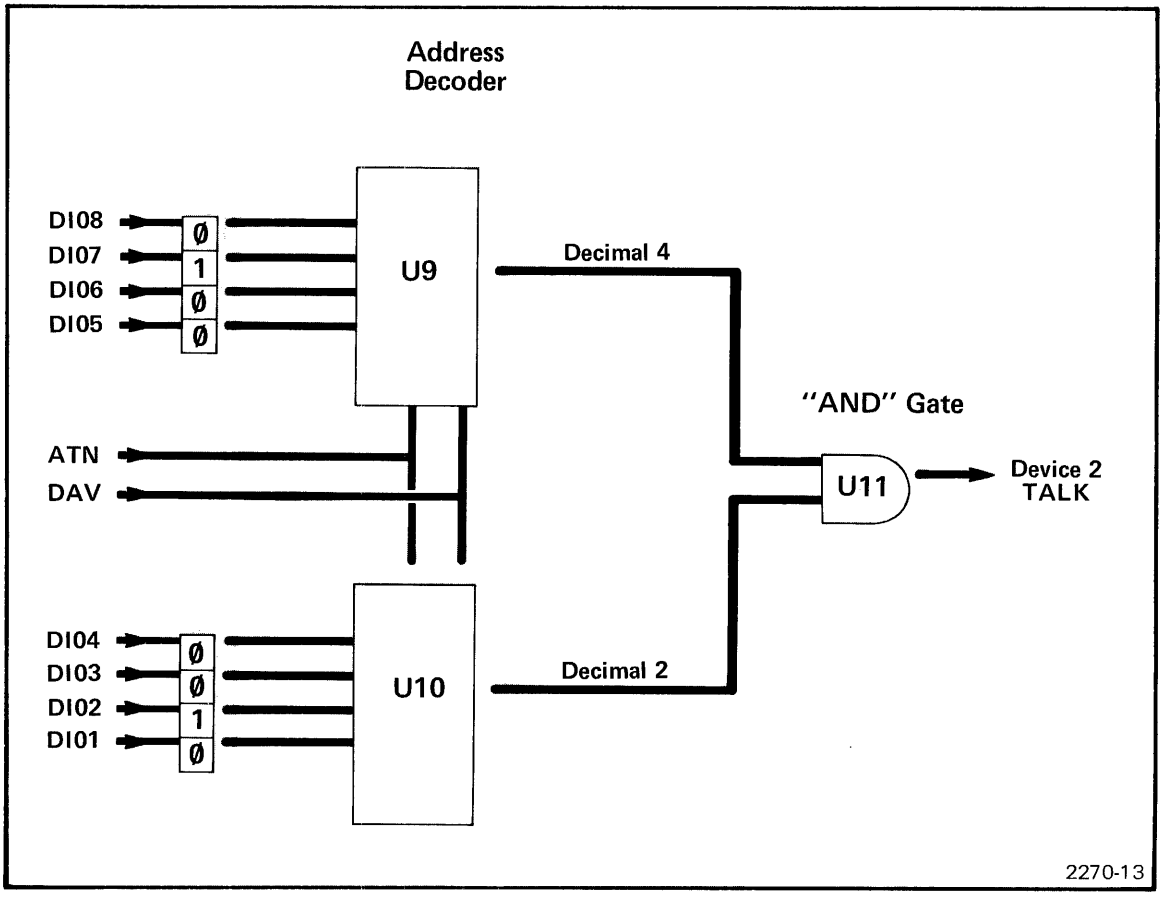


Fig. 2-5. Conceptual view of GPIB address decoder.

The binary bit pattern for the device 2 primary talk address is shown in Fig. 2-5. It is interesting to look at the meaning of each data line in this pattern.

The upper four data lines indicate whether the address is a universal controller command, a primary listen address, a primary talk address, or a secondary address. DIO8 is always inactive (false) and represents a binary 0. DIO7 and DIO6 determine the address type. If both lines are inactive false (00) the address is a universal controller command (like Serial Poll Enable). If DIO7 is false and DIO6 is true (01), then the address is a primary listen address. If DIO7 is true and DIO6 is false (10), as shown in the illustration, then the address is a primary talk address. And, if both lines are true (11), the address is a secondary address.

DIO5 determines whether the address is for devices 0-15 or 16-30. If DIO5 is false (0), then the address is for a device within the range 0-15. If DIO5 is true (1), then the address is for a device within the range 16-30.

The lower data lines DIO1-DIO4 indicate the device number within the upper or lower address group. In Fig. 2-5, DIO5 is false, so the address is for a device in the lower group (0-15). Since the binary number on DIO1-DIO4 represents a decimal 2, this primary talk address is for device 2.

The address decoder doesn't recognize the individual meaning of each data line, but instead treats the address as a two digit decimal number. The binary number on the upper four lines is equivalent to decimal 4, in this case, so the output representing decimal 4 on decoder U9 goes active true. At the same time, the binary number on the lower four lines represents decimal 2, so the output representing decimal 2 on decoder U10 goes active true. The combination of both these outputs makes the output of the AND gate U11 go active true and tells the interface memory that the primary talk address 2 has been received. Usually, the active transition of the AND gate output is used to clock a D-latch in the interface memory. This places the latch in a "set" condition and triggers other interface circuits into a TALK state.

It can be seen from this diagram that each address on the GPIB causes a different two-pin combination on the output of the address decoders. Therefore, a separate AND gate is required for each address that must be decoded. It can also be seen that the peripheral device number assignment is made at the output of the address decoder. In this case, the interface responds to the primary talk address for device 2. If we want this peripheral device to be device 3, then the lower input to the AND gate has to be changed to the output on U10 which represents decimal 3.

Enabling the Address Decoder

The address decoder must only be allowed to operate while ATN is low on the 4051 GPIB. To ensure that this requirement is met, a decoder with two enable inputs is used. Both inputs must be active for the decoder to operate. One input is made active when ATN goes low on the GPIB. The other goes active when DAV goes low on the GPIB. The ATN/DAV combination ensures that the address decoder only decodes valid peripheral address and controller commands. Due to 4051 transfer characteristics, the ATN/DAV combination lasts for at least 18 μs; this may be longer if a peripheral device slows the handshake down. This means that the output of the AND gate will be a positive-going pulse with a duration of 18 μs (minimum).

The Address Decoder And The Interface Memory

Since the interface address decoder and the interface memory are closely related, it is appropriate to talk about both circuits at the same time. Turn your attention now to the schematic diagram in Fig. 2-6.

This particular interface circuit is designed to respond to 4051 PRINT statements, INPUT statements, and POLL statements. Each of these operations are now covered in detail starting with PRINT operations.

4051 PRINT OPERATIONS

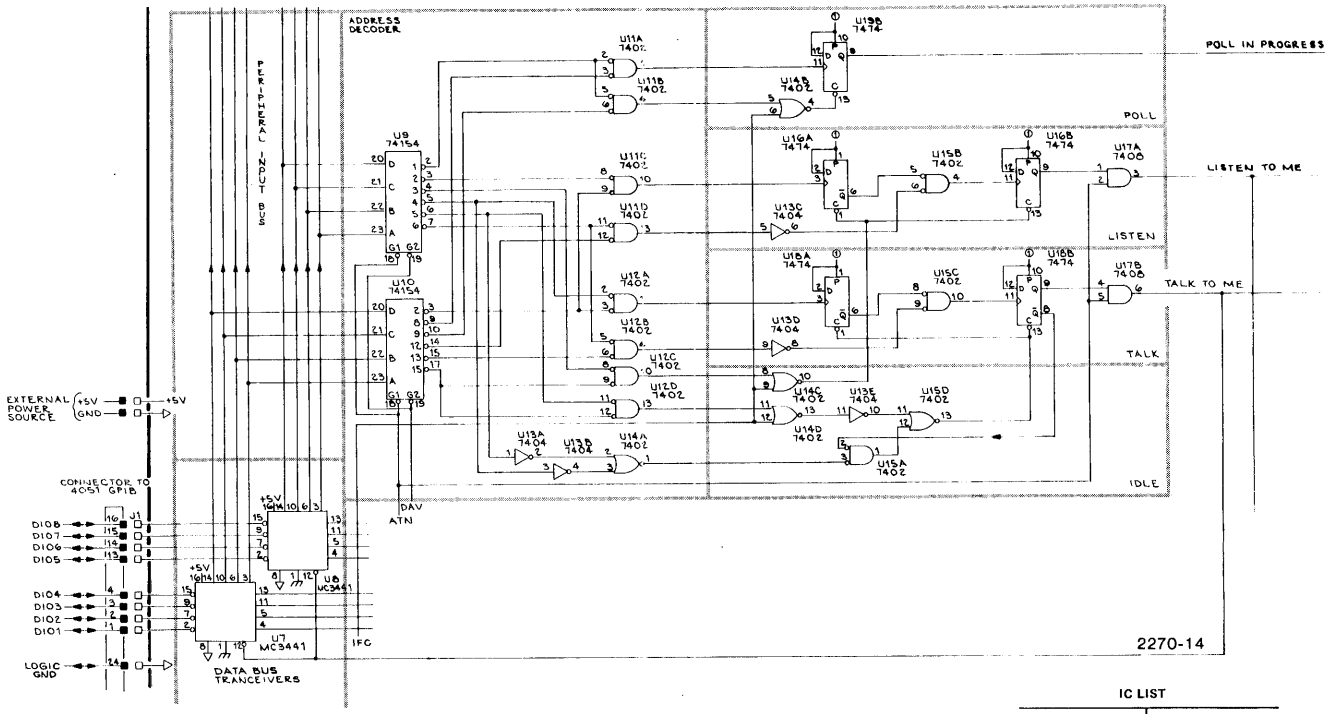
The PRINT operation was chosen in this case to illustrate how an interface should respond to a typical output transfer from the 4051. Usually a PRINT operation implies that the peripheral device is able to receive and handle data formatted in ASCII code. Line printers, mass storage devices, and programmable universal counters are examples of such devices.

In all PRINT statements, the 4051 issues the specified primary listen address followed by the default secondary address 12 (if another secondary address is not specified in the statement). The interface address decoder and memory circuit must be set up to recognize this sequence. Assume that the statement PRINT @2: "ABC" is executed. Here's how the address decoder responds during the print operation:

1. The 4051 starts the PRINT operation by setting ATN low on the GPIB. This makes pin 14 on GPIB transceiver U1 go high and the output of inverter U3C go low (see Diagram B). This low sets pin 18 on both address decoders (U9 and U10) low and fullfills half of the enable requirements for the decoders.

2. While ATN is low, the interface listen handshake circuit gets on the bus (as previously described) and tells the 4051 that the interface is ready to receive the first address byte.

3. The 4051 responds by placing the primary listen address for device 2 on the GPIB Data Bus and waits for the data lines to settle.



IC LIST

COMPONENT	TYPE
U7	MC 3441
U8	MC 3441
U9	SN 74154
U10	SN 74154
U11	SN 7402
U12	SN 7402
U13	SN 7404
U14	SN 7402
U15	SN 7402
U16	SN 7474
U17	SN 7408
U18	SN 7474
U19	SN 7474

Fig. 2-6. Typical 4051 GPIB address decoder and memory design.

4. Since the interface GPIB Data Bus receivers are active at this time, the primary listen address appears at the input the address decoder. Since the GPIB Data Bus receivers invert the signal states, the address decoder must be designed to accept positive logic; that is, the higher voltage state represents a binary 1.

5. After the 4051 waits for the lines to settle, the 4051 sets DAV low on the GPIB.

6. The low DAV signal makes pin 10 on the GPIB transceiver go high. This causes the output of inverter U3D to go low and triggers the address decoders (U9 and U10) to decode the address.

7. Since the binary number on DIO5-DIO8 represents decimal 2, pin 3 on U9 goes active low. The binary number on data lines DIO1-DIO4 also represents decimal 2, so pin 3 goes active low. These two low outputs make the output of AND gate U11C go high.

8. The low-to-high transition at the output of U11C clocks D-latch U16A. Since the D input is tied high to +5 vdc through resistor R1, the latch enters a "set" state where the $\overline{Q}$ output (pin 6) goes low. The D-latch acts as a memory element, recording the fact that the interface primary listen address has been received.

9. In this case, the low $\overline{Q}$ output on U16A is used to enable AND gate U15B. This prepares the interface memory to receive secondary address 12 which follows the primary listen address.

10. While the above action is taking place, the interface listen handshake circuitry completes the handshake with the 4051. The 4051 then takes the primary listen address off the GPIB and places the default secondary address 12 on the GPIB. The interface listen handshake circuit starts the second handshake sequence and the address decoder responds as follows.

11. When DAV goes low on the GPIB for the second time, the secondary address is at the input of the address decoder. The binary number on data lines DIO5-DIO8 represents decimal 6 so pin 7 on U9 goes low. The binary number on data lines DIO1-DIO4 represents decimal 12 so pin 14 on U10 goes low. These two low outputs make pin 13 on U11D go high which in turn makes the output of U13C go low. Since pin 5 on U15B is already low, the low in pin 6 causes the output of U15B to go high and clock D-latch U16B.

12. When U16B is clocked, the Q outputs enter a "set" state making the Q output (pin 9) go high. This indicates that the secondary address for PRINT has been received. It is important at this time that the interface not allow the peripheral device to listen to the GPIB Data Bus until after ATN goes high. The Q output on U16B is therefore fed into AND gate U17A. This AND gate is held in a disabled state until ATN is set high by the 4051.

13. While the above action is taking place, the interface listen handshake circuit finishes the handshake sequence for the secondary address byte. The 4051 then releases ATN and prepares to issue the ASCII character string "ABC" (CR).
14. When ATN goes high, pin 14 on GPIB transceiver U1 goes low (see Diagram B). This causes pin 6 on inverter U3C to go high which enables U17A to set LISTEN TO ME active high.
15. The LISTEN TO ME signal tells the peripheral device to prepare to receive ASCII information over its input data bus. LISTEN TO ME also switches the interface listen handshake circuit into a mode where the peripheral listen handshake signals GRAB IT, GOT IT, and I'M BUSY are incorporated into the GPIB handshake.

The Peripheral Device Receives The ASCII String

The 4051 at this time sends the three ASCII characters A, B, and C followed by a carriage return. As each character is placed on the GPIB Data Bus, the character also appears on the peripheral input data bus via GPIB transceivers U7 and U8 (located in the interface circuitry). The interface listen handshake circuit then goes through the handshake with the 4051 (as previously described) and the peripheral device receives the ASCII string "ABC".

After the fourth handshake, the CR is received by the peripheral device and the 4051 sets ATN low on the GPIB. This immediately sets LISTEN TO ME low because AND gate U17A is disabled. When LISTEN TO ME goes low, the interface handshake circuit returns to its original operating mode and ignores the peripheral listen handshake signal lines GRAB IT, GOT IT, and I'M BUSY. The interface address decoder is enabled at this time and the interface prepares to receive and decode additional address from the 4051.

The 4051 Issues UNLISTEN To The Interface

With ATN low, the 4051 issues the controller command UNLISTEN (decimal 63) to the interface. This makes pin 6 on U9 go low and pin 15 on U10 go low. The two low outputs on U9 and U10 cause pin 10 on AND gate U12C go high. This forces the output of U14C to go low which makes the CLEAR inputs on U16A and U16B go active low. As a result, both D-latches return to a "reset" state and the interface enters the IDLE state.

The 4051 finishes up by releasing ATN, then leaves the bus. The interface also leaves the bus.

INTERFACE CLEAR

It is important that the interface be in idle condition before the PRINT operation is executed. The interface doesn't have power-up circuitry to place each component in a predetermined state, so an IFC (interface clear) function is built-in to place the interface into an IDLE state. The 4051 sets IFC low when an INIT statement is executed in BASIC. This is one reason why it is good practice to place an INIT statement at the beginning of every BASIC program. INIT "resets" the 4051 to an initial state and at the same time pulses IFC on the GPIB to "reset" the interface circuitry on the GPIB. Here's how the interface responds to IFC (see Diagram B).

The low going IFC signal on the GPIB causes pin 6 on GPIB transceiver U1 to go high. This makes the output of NOR gates U14C, U14B, and U14D to go low. (These gates are located in the interface memory block.) It can be seen in the schematic diagram in Fig. 2-6 that these low outputs activate the CLEAR inputs on all D-latches in the interface memory. Since each D-latch represents a "memory cell," the entire interface memory is cleared and the interface enters an IDLE state. The three memory output signals POLL IN PROGRESS, LISTEN TO ME, and TALK TO ME go inactive low, if they are not already low.

A WORD ABOUT 4051 WRITE OPERATIONS

Two D-latches are used in the interface memory circuit to record the transmission of both the primary listen address and the secondary address for PRINT. If this peripheral device could only listen to ASCII data, then only one D-latch would be necessary. The secondary address for PRINT could be ignored and the Q output on U16A could be tied directly to pin 1 on AND gate U17A to activate the LISTEN TO ME signal.

But two D-latches are used simply to illustrate how to set up a listen function based on the secondary address.

Assume for the moment that the peripheral device is a mass storage device capable of receiving data in 4051 internal binary format as well as ASCII code format. To make the distinction between the two formats, the interface memory needs to send two signals to the peripheral device. One signal called LISTEN TO ME PRINT for ASCII data transfers and one signal called LISTEN TO ME WRITE for 4051 internal binary code transfers.

Since the 4051 issues the same primary listen address in both a PRINT and WRITE operation, the interface must make the distinction between the two operations by looking at the secondary address. The 4051 issues secondary address 12 for PRINT operations and secondary address 15 for WRITE operations.

The decoding circuitry presently in the interface doesn't need to be changed; but additions need to be made. Another decoding circuit similar to the logic chain U11D, U13C, U15B, U16B, and U17A needs to be added to decode the WRITE secondary address. Then, when the primary listen address is issued for the interface, D-latch U16A is set as previously described. But, when the secondary address is issued, only one of the two secondary address D-latches are set depending on whether the secondary address is 12 or 15. If secondary address 12 is issued, a latch is set which makes the signal LISTEN TO ME PRINT go active. If secondary address 15 is issued, a latch is set which makes the signal LISTEN TO ME WRITE go active. This tells the peripheral device what the data format is and the peripheral device can take the appropriate steps in preparing to receive the information.

4051 INPUT OPERATIONS

Input operations require the interface to assume a talker role and transmit data to the 4051. The circuitry in this interface is set up to respond to INPUT operations when INPUT statements are executed in BASIC. Like the listen operations just described, the interface can easily be set up to operate in a multitude of talk operations, like INPUT, READ, OLD, etc. The operation selected depends on the capabilities of the peripheral device. In this case INPUT is selected because ASCII data transfers are typical of most GPIB peripheral devices. The following text describes the interface response when the statement INPUT @2:A$ is executed in 4051 BASIC.

1. The 4051 starts the INPUT operation by setting ATN low on the GPIB. This causes the interface listen handshake circuitry to set NDAC low and NRFD high, as previously described.

2. The 4051 then places the primary talk address for device 2 on the GPIB and the address appears at the input of the interface address decoder. The 4051 then sets DAV active low.

3. When DAV goes low, pin 5 on address decoder U9 goes low and pin 3 on address decoder U10 goes low. This makes the output of AND gate U12A go high and clock D-latch U18A.

4. D-latch U18A serves as the "memory element" for the primary talk address. Since the D-input is always high, the latch enters a "set" state when clocked by U12A. The $\overline{Q}$ output of U18A goes low and tells the interface that the primary talk address has been received.

5. The low on pin 6 of U18 enables AND gate U15C to pass the secondary address signal from the address decoder to D-latch U18B. This signal comes from the address decoder as soon as the secondary address for INPUT is transferred over the GPIB.

6. While the above action is taking place, the interface listen handshake circuitry completes the handshake with the 4051. The 4051 then takes the primary talk address off the bus, places the secondary address for INPUT (decimal 109) on the bus, and sets DAV low. This causes pin 7 on U9 to go low and pin 15 on U10 to go low; the output of AND gate U12B goes high.

7. The high U12B output makes the output of inverter U13D go low. This low, in combination with the low from U18A pin 6, makes the output of AND gate U15C go high and clock D-latch U18B.

8. D-latch U18B serves as the "memory element" for the INPUT secondary address. The latch enters a "set" state when clocked by U15C; pin 9 on U18B goes high and pin 8 goes low. This tells the interface to start talking in ASCII code, but only after ATN goes high.

At this point, the interface must continue to receive and decode addresses until ATN goes high. And, the interface must prevent the peripheral device from "seeing" the talk signal until ATN goes high.

AND gate U17B is in the circuit to fulfill this requirement. With ATN in a low state, pin 5 on U17B stays low to keep the gate disabled. This prevents the peripheral device from seeing the high true TALK TO ME signal at pin 9 on U18B. After the 4051 sets ATN high, the output of U17B goes high which sets TALK TO ME to a true state. This tells the peripheral device to start talking in ASCII code.

The Implied UNTALK

If a peripheral device is assigned to be a talker with ATN low, and the controller issues another primary talk address without first issuing the UNTALK command, the first talker must interpret this action as an implied UNTALK and get off the bus. It is important that the implied UNTALK facility be built into the GPIB interface because a serial poll sequence depends on the presence of this facility. In addition, the 4051 may be programmed to issue one or more primary talk addresses in sequence (over a period of time) with the WBYTE statement and each talker must have the implied UNTALK facility in operation; if not, several talkers might be on the bus transfer data at the same time. This situation produces pure chaos!

The implied UNTALK requirement is built into the interface circuitry as follows:

When D-latch U18B enters a "set" state, the $\overline{Q}$ output (pin 8) goes low which enables AND gate U15A to pass a primary talk signal from the address decoder to NOR gate U15D. With the interface in this state, any additional primary talk addresses coming over the GPIB causes pin 1 on U14A to go low. This makes the output of U15A go high which makes the output of U15D go low. Since the output of U15D is tied to both clear inputs on U18A and U18B, the low going U15D output clears both D-latches and returns the interface to an UNTALK state.

Transmitting Data Over The GPIB

After the 4051 issues the secondary address for INPUT, the 4051 assigns itself as a listener, then releases ATN. This makes the TALK TO ME signal on the interface circuit board go active high. TALK TO ME tells the peripheral device to place the first data byte on the peripheral output data bus. At the same time, TALK TO ME activates the interface talk handshake circuitry and places the GPIB Data Bus drivers in a talk state.

The peripheral device transmits data bytes over the GPIB (as previously described) and the 4051 receives the data bytes and assigns them to the variables specified in the INPUT statement. The interface talk handshake circuit goes through the handshake sequence with each data byte transferred. This action continues until all of the specified variables have assigned values. The 4051 then sets ATN active low and issues the UNTALK command over the GPIB.

Responding To The UNTALK Command

When the 4051 sets ATN low, AND gate U17B is immediately disabled. This makes TALK TO ME inactive low which returns the GPIB Data Bus transceivers to a listen state and also activates the interface listen handshake circuitry. When the 4051 issues the UNTALK command, pin 6 on U9 goes low and Pin 17 on U10 goes low. This causes the following chain reaction: the output of AND gate U12D goes high; the output of NOR gate U14D goes low; the output of inverter U13E goes high; and the output of U15D goes low. The high to low transition at the output of U15D makes both CLEAR inputs to D-latches U18A and U18B go low. This places both latches in a "reset" state. After that, the 4051 releases ATN and the interface returns to an idle state.

4051 SERIAL POLL OPERATIONS

Serial poll operations occur on the GPIB whenever the 4051 executes a POLL statement in BASIC. This usually occurs when a peripheral device sets SRQ (service request) on the GPIB active low and an ON SRQ THEN statement in the BASIC program transfers program control to a POLL statement. All peripheral devices with SRQ capability are listed in the POLL statement with the highest priority device listed first. The 4051 initiates the serial poll on the GPIB and the interface circuitry responds as described below. (Refer to the serial poll timing diagram in Section 4 for additional information.)

1. The 4051 starts the serial poll by setting ATN low. The 4051 then issues the commands UNLISTEN (decimal 63) and SERIAL POLL ENABLE (decimal 24). If the interface is in a LISTEN state, the UNLISTEN command clears the interface (as previously described). The SPE command is then received by the interface and decoded.

2. The SPE data byte causes pin 2 on address decoder U9 to go low. Pin 9 on U10 also goes low. These two low outputs cause the output on AND gate U11A to go high and clock D-latch U19B. Since the D-input on U19B is always high, the latch enters a "set" state and pin 9 goes high. This high makes the POLL IN PROGRESS signal active true which tells the peripheral device that the 4051 is executing a serial poll and to place the peripheral status byte on the peripheral output data bus. The status byte is not placed on the GPIB Data Bus at this time, however, because the interface is not addressed as a talker and the GPIB Data Bus transceivers are in a listen state.

3. Next, the 4051 issues the primary talk address (and the secondary address, if one is specified) for the first device in the POLL statement address list. If the address is for another device on the bus, the interface address decoder decodes the address sequence, but the TALK latches (U18A and U18B) remain in a "reset" state. The 4051 then turns the bus around, assigns itself as a listener and receives the status byte and checks bit 7 for a binary 1 state.

   If bit 7 is set (true), the 4051 terminates the serial poll by setting ATN low and issues UNTALK, followed by SPD (serial poll disable). The SPD command byte makes the output of AND gate U11B go high, which makes the output of NOR gate U14B go low. This action clears U19B and returns the POLL IN PROGRESS signal (pin 9) to an inactive (low) state.

If bit seven in the first status byte is not set, the 4051 continues with the serial poll by setting ATN low and issues the primary talk address for the next device in the POLL statement address list. This tells the second device in the list to send its status byte and at the same time issues an "implied" UNTALK command to the first device. This action continues until the device address for this interface is issued.

Transferring The Status Byte

If this interface is holding SRQ active low, and the 4051 issues the primary talk address and the secondary address for INPUT with ATN low, then the interface enters a TALK state and TALK TO ME goes active (high). This activates the talk function in the interface and the interface transfers the peripheral status byte to the 4051. The peripheral device must issue the status byte at this time, because POLL IN PROGRESS is active true.

The 4051 examines the status byte and sees bit 7 set to a logic 1.

The 4051 then sets ATN low, issues UNTALK and SPD (serial poll disable) then releases ATN. This action clears D-latches, U18A, U18B, and U19B, and returns the interface to an idle state.

At this point the serial poll is over. Assuming that the 4051 is programmed to service this peripheral device, the 4051 branches to the service routine for this device via a GOTO OF statement, then readdresses this peripheral device according to directions in the service routine.

In cases where this interface is not requesting service, the 4051 issues the next primary talk address in the POLL statement address list after receiving the status byte. This action is an automatic "implied" UNTALK command to our interface and the "implied" UNTALK circuitry returns the TALK latches (U18A and U18B) to a "reset" state, as previously described. The SPD command at the end of the serial poll then resets U19B and the POLL IN PROGRESS signal goes inactive. This tells the peripheral device that the serial poll is over.

OBTAINING THE MAXIMUM SUSTAINED DATA RATE FOR ASCII INPUT

INTRODUCTION

In many applications, peripheral devices must collect large amounts of data in a short period of time and pass the data as quickly as possible to the 4051, before the data is lost. Time is of the essence; any delays in the transfer can cause valuable data samples to be lost (forever). If a peripheral device outputs data in ASCII code format, the INPUT statement in 4051 BASIC is a logical choice for receiving the data and assigning the data to variables in memory. The INPUT statement is very flexible—allowing a multitude of different ASCII formats to be transmitted. In an INPUT operation, the ASCII data is sorted as it comes into the 4051 I/O buffer; all non-valid characters are thrown out and only valid data is retained and assigned to the specified variables. This INPUT sorting operation does take time, however, and is not ideal for operations where the data format is known and fixed, and sustained data bursts over a long period of time are critical to the success of an operation.

USING THE READ STATEMENT TO INPUT ASCII DATA

The READ and WRITE statements are in the 4051 BASIC language to speed up data transfers between system components. The normal data formatting routines are bypassed and the data is taken directly from the 4051 memory and passed back and forth between peripheral devices. Since very few peripheral devices can correctly interpret the 4051 internal format, READ and WRITE data transfers are usually directed toward mass storage devices which store the data rather than interpret the data.

A special format is used to speed up the transfer. The 4051 generates a two-byte header for each data item. The header tells the receiving device the data item type (string or numeric) and the data item length in bytes. After the header is issued to the receiving device in a WRITE operation, the data item is lifted directly from the 4051 internal memory and transferred to the receiving device. Since the length of the data item is specified in the header, the receiving device does not have to "look" at the data as it is being received. The only action required by the receiving device is to count the number of bytes as they come in and delimit the data item after the specified number of bytes are received.

The 4051 follows the same rules when receiving data via a READ operation. The 4051 finds out the data item length from the header, then inputs the specified number of bytes without "looking" at the data as it is being received. Once the specified number of bytes are received, the 4051 delimits the data item and assigns the data to a variable. The 4051 **assumes** the data item is in the correct 4051 internal binary format.

4051 INTERNAL DATA FORMATS

NUMERIC DATA

Each numeric value is stored in the 4051 internal memory as an eight-byte floating-point number. This format is difficult to generate from scratch—therefore it is usually not practical to design a peripheral interface that converts ASCII numbers to 4051 internal binary floating point. In those cases where the conversion is practical, however, it is usually best to use a microprocessor based interface that is driven by firmware rather than try to handle the conversion with TTL logic. (A complete description of the 4051 internal floating point format is given in Appendix A.)

STRING DATA

ASCII string data is stored in the 4051 memory using the same binary numbers as the external format. That is, an ASCII "A" is represented internally by a binary bit pattern equivalent to decimal 65—the same as the external format.

When character strings are transferred from the 4051 memory using READ and WRITE, the 4051 generates the two-byte header describing the data item type and length, just like it does when transferring numeric data. The string is then transferred in ASCII code after the header is issued, the same as in a PRINT statement, only without the formatting delays.

When the 4051 executes a READ statement with a string variable as a parameter, such as READ @2:A$, the 4051 **assumes** that peripheral device (2) is sending an ASCII character string preceded by the proper two-byte header. Therefore, the 4051 receives and checks the first two bytes transferred from device 2. If the 8-bit bytes are in the correct header format and specify "character string X bytes long," then the 4051 inputs X bytes in a sustained burst and assigns the bytes to the specified string variable (A$ in this case). Since the 4051 **assumes** that the data is in the correct ASCII format, the character string is transferred at the maximum rate and the 4051 does not look at the data as it is being received. This method of transfer is much faster than an INPUT operation because the data scanning routines are bypassed. Data bursts of up to 8192 ASCII characters (maximum) can be achieved in this way as a sustained rate of 5.2K bytes/second.

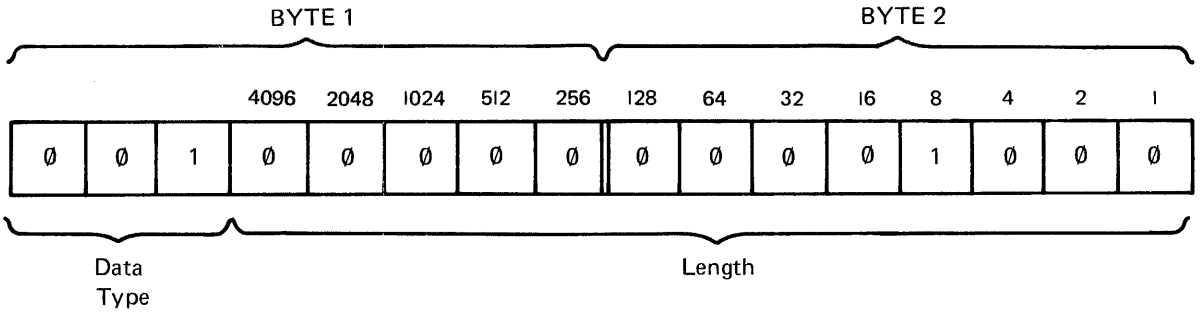
PROGRAMMING A PERIPHERAL DEVICE TO CREATE A TWO-BYTE HEADER

If an ASCII peripheral device can be programmed to disregard the first two bytes received, then the WRITE statement can be used to transfer ASCII character strings at a 8K byte/sec sustained burst to the peripheral. Likewise, if an ASCII peripheral device can be programmed to issue a two-byte header prior to transferring an ASCII character string, the transfer can be done with a READ statement.

The 4051 HEADER FORMAT

Each data item transferred in 4051 machine dependent binary code with READ and WRITE is preceded by a two byte (16 bit) header. This is true whether the data item is numeric data or character string data. The two byte header contains information which tells the receiving device the data item type (numeric, string, or End of File mark) and the data item length (in bytes). The Fig. 2-7 illustrates how information is stored in the two byte header.

The three high order bits of the first byte represent a binary number from Ø through 7. This number indicates the data item type. A binary 1 (ØØ1) means the data item is a numeric value. Binary 2 (Ø1Ø) means the data item is a character string. And binary 7 (111) means the data item is an End of File Character. All other bit combinations are undefined.



DATA TYPE

CODE	MEANING
Ø Ø Ø	Undefined
Ø Ø 1	Numeric Value
Ø 1 Ø	Character String
Ø 1 1	Undefined
1 Ø Ø	Undefined
1 Ø 1	Undefined
1 1 Ø	Undefined
1 1 1	End of File

2270-15

Fig. 2-7. 4051 binary header format for READ and WRITE.

The five remaining low order bits in the first byte of the header and the eight bits of the second byte of the header form a binary number which tells how many bytes are in the data item. Each bit carries a decimal weight as shown in the illustration. In this case, the data type is numeric (ØØ1) and the body of the data item (which follows the header) is eight bytes long.

As a general rule, each numeric data item is eight bytes long with a two byte header for a total length of 1Ø bytes. Each character string requires one byte for each character plus the two byte header for a total length of LEN+2 (where LEN is the Length function). The maximum length of a string in this format is 8192 bytes.

BUILDING A HEADER GENERATOR FOR AN ASCII PERIPHERAL DEVICE

If the output of an ASCII peripheral device cannot be programmed to issue a two-byte header, the only alternative is to build a header generator out of TTL logic and attach the device to the peripheral's GPIB connector. The rest of this section discusses how a header generator can be built. A practical design for a header generator is used for illustration. This design has been tested and works satisfactorily. However, remember that it is presented for educational purposes only and we can not guarantee that it will work in your particular application.

AN OVERVIEW ON HOW THE HEADER GENERATOR WORKS

The header generator in Fig. 2-8 is a little black box that has an IEEE Standard 488-1975 plug on each end—one male and one female. The box is designed to be connected in **series** with the ASCII peripheral device and the 4051 GPIB cable. Power to the box must be supplied by the peripheral device or an external power supply. Data transferred back and forth between the 4051 and the peripheral device passes through the box.

Normally, the 4051 uses PRINT statements to send messages to the peripheral device and INPUT statements to receive data from the peripheral device. With the header generator in the data path, PRINT statements can still be used to send messages to the peripheral device. However, both INPUT and READ statements can be used to receive messages from the peripheral device. The header generator contains GPIB handshake circuitry (both listen and talk), an address decoder which responds to the same addresses as the address decoder in the peripheral device, and two data registers that hold predefined header bytes.

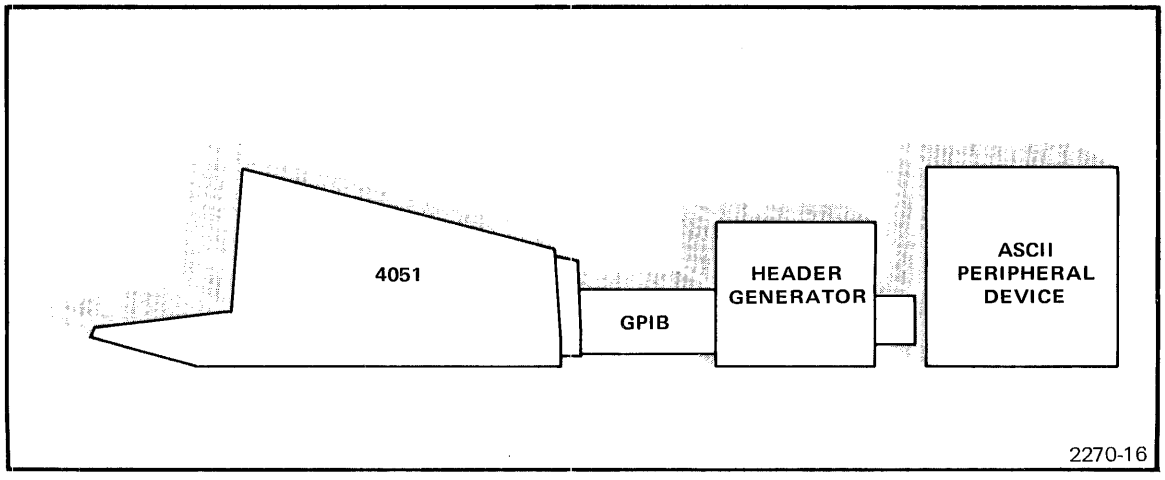


Fig. 2-8. 4051 binary header configuration for an ASCII peripheral device.

Operation

During PRINT operations, the address decoder in the header generator and the address decoder in the peripheral device decode the primary listen address and the secondary address for PRINT at the same time. Because the operation is a PRINT operation, the header generator remains passive and allows the data to pass through the box to the peripheral device "untouched."

On INPUT and READ operations, the header generator is set-up to respond to secondary address 14 (for READ). A two-byte header is then issued.

During a READ operation, the 4051 issues the peripheral's primary talk address and the secondary address 14 for READ. Both the header generator and the peripheral device receive these addresses at the same time. The header generator takes over and keeps the peripheral device "in-check" while a two byte header is issued to the 4051.

The 4051 receives the header bytes and, while the 4051 is examining the bytes, the header generator gets off the bus and allows the peripheral device to look at the 4051 handshake lines. When the 4051 says it's ready to receive the data, the peripheral device places the first ASCII data byte on the GPIB Data Bus and there're off and running. The 4051 starts receiving the ASCII data bytes as fast as it can without looking at the data. As soon as the 4051 receives the number of bytes specified in the header, the 4051 terminates the transfer by issuing UNTALK to the header generator and the peripheral device.

INPUT operations can also be performed, but in a slightly different manner. First, the secondary address 14 must be specified in the INPUT statement, such as INPUT @2,14:A$. This makes the operation appear to be a READ operation to the header generator. The header generator issues the two-byte header, then lets the peripheral device transfer the character string. The difference in this case is that the header is treated like the first two characters in the ASCII string. The INPUT data scanning and sorting routines are active and the data transfer is handled like a normal INPUT operation. When an CR delimiter is received, the 4051 terminates the operation. Because the first two bytes in the string are known to be the header, they are deleted using the REP function in BASIC.

HEADER GENERATOR CIRCUIT DESCRIPTION

Introduction

Diagram C in the back of this manual is the schematic diagram for the header generator just described. The 4051 GPIB interconnecting cable plugs into the female connector J1 on the left. The male connector J2 on the right can be plugged directly into the GPIB connector on the peripheral device or is connected to the peripheral device with a standard GPIB interconnecting cable.

Data traveling to and from the peripheral device over the 4051 GPIB Data Bus passes through the bus regulation network U19, U20, U21, and U22. This network allows data to travel over the bus in one direction at a time. The direction of the data flow is controlled by D-latch U17A in the top center position on the schematic.

Peripheral addresses are taken off the 4051 GPIB Data Bus by transceivers U1 and U2. The addresses are placed on the input terminals of address decoder U6 and U7 where they are decoded and passed on to the memory elements U9A and U9B, (if they are valid). The address decoder is set-up to respond to primary talk address 2 (decimal 66), which clocks D-latch U9A into a set state. Secondary address 14 for READ (decimal 110) clocks D-latch U9B into a set state.

The TALK and LISTEN handshake circuits, located just below the address decoder, are similar to the handshake circuits described in the General Purpose Interface description. The details on how this circuit works will be discussed in a moment.

When the header and the peripheral device are addressed simultaneously during a READ operation, the header generator inhibits the peripheral device from talking by holding the NRFD signal line on connector J2 in a low state.

At the same time, header generator uses the bus regulation network U19, U20, U21, and U22 to keep the peripheral data off the 4051 GPIB Data Bus.

While inhibiting the peripheral device from talking, the header generator handshakes with the 4051 twice and sends two predefined header bytes from data registers U13-U14 and U15-U16. After these bytes are transferred, the header generator allows the peripheral data to pass through the bus regulation network. The handshake signals are also reconnected so that the peripheral device can freely handshake with the 4051 until the data transfer is over. The following text describes in detail how the above circuit action takes place.

A Few Words about the Handshake Circuitry

The listen and talk handshake circuitry is a simplified version of the handshake circuit described in the beginning of this section. Components have been eliminated and the circuit has been designed to respond specifically to 4051 handshake characteristics. The major difference in this handshake circuit over the one previously described is that this circuit always stays on the GPIB handshaking in a listen mode, even if the header generator is not addressed. The header generator, however, does not listen to the data unless the 4051 activates ATN. The header generator handshake is so fast that is can never slow down a transfer between the 4051 and another peripheral device. Therefore, its presence on the GPIB is not detrimental.

Initializing the Header Generator Circuits

It is important that D-latches U9A and U9B be placed in a "reset" state before a READ or INPUT operation is executed by the 4051. Normally, an INIT state is executed by the 4051 at the beginning of every BASIC program. During an INIT operation, the 4051 pulses the IFC (Interface Clear) signal line on the GPIB. The header generator responds to IFC as follows:

The low-going IFC signal makes pin 6 on GPIB transceiver U4 go high. This makes the output of NOR gate U23B go low and set the CLEAR inputs to D-latches U9A and U9B low. This resets both latches (Q output low and $\overline{Q}$ output high).

The Circuit Action During a READ Operation

Responding to ATN. The 4051 starts the READ operation by setting ATN active low. The header generator circuitry responds to ATN as follows:

1. The low ATN signal makes pin 15 on Transceiver U4 go low. Pin 14 on U4 goes high making the output of inverter U5A go low. This low signal enables addresses decoders U6 and U7 to function.

2. Since data is not valid at this time, the DAV signal on the GPIB is high (inactive). This makes pin 9 on transceiver U4 go high and pin 10 go low. Pin 10 on U4 is the source of the header generators NRFD and NDAC signal response.

3. The low on U4 pin 10 causes pin 2 on NOR gate U23A to go low. Assuming at this point that the peripheral device is ready to receive addresses, pin 3 on U23A is low. This makes the output of U23A go high, the output of inverter U18D go low, and the output of the NRFD bus driver (U3, pin 9) go high. This high NRFD signal tells the 4051 that the header generator (and the peripheral device) are both ready to receive addresses over the GPIB.

4. If the peripheral device is not ready to receive addresses when ATN goes low, the peripheral indicates its unready state by setting pin 7 on GPIB plug J2 to a low state (right side of schematic). This causes the following chain reaction: The output of bus receiver U18A goes high making pin 13 on AND gate U10D go high. Since D-latch U9A is reset at this time, the high on pin 6 makes pin 12 on U10D go high. The output of U10D also goes high making the output of U23A go low, and the output of inverter U18D go high. Pin 9 on GPIB transceiver U3 goes low as a result and tells the 4051 that the peripheral device is not ready to receive addresses. When the peripheral device is ready, pin 7 on J2 goes high, the chain reaction occurs in reverse, and NRFD on the GPIB goes high.

5. At the same time the NRFD response is being generated, the header generator sets NDAC low as follows. The low at pin 10 on transceiver U4 causes the output of inverter U5B to go high. This high is fed directly to the input of the NDAC driver (U3, pin 13) and causes the output of the driver (U3, pin 15) to go low. This tells the 4051 that the header generator (and the peripheral device) have not yet received a data byte.

Receiving and Decoding the Primary Talk Address. After NDAC is low and NRFD is high on the GPIB, the 4051 continues the addressing operation. First, the 4051 places the primary talk address for the header generator (and the peripheral device) on the GPIB Data Bus. In this example, both devices are set to respond to primary talk address number two (decimal 66). Since the GPIB Data Bus transceivers U1 and U2 are in a listen state (drivers disabled by a high pin 12), the primary talk address is passed to the input of address decoders U6 and U7. The address is also presented to the Data Bus regulation network U19, U20, U21, and U22.

The Data Bus Regulator Network allows the header generator to control the data transmissions to and from the peripheral device. In the present state, the network is in a listen mode—allowing data to travel from the 4051 GPIB to the peripheral device and preventing data from traveling in the opposite direction.

With the primary talk address at the input ports to both address decoders, the 4051 sets DAV low. The header generator responds as follows:

1. Pin 10 on transceiver U4 goes high. This makes the output of NOR gate U23A go low, the output of inverter U18D go high, and the output of the NRFD driver (U3 pin 9) go low. (Setting NRFD low is the first step in receiving an address byte.)

2. The high on U4 pin 10 also causes the output of inverter U5B to go low, making the output of the NDAC driver (U3, pin 15) go high. This tells the 4051 that the header generator has accepted the address byte.

3. Although the header generator's handshake response is practically instantaneous, the peripheral's handshake is also occurring at the same time. The peripheral's NRFD response is ORed with the header generators response at U23A. The peripheral's NDAC response occurs independently on pin J2-8 on GPIB plug J2 and keeps the NDAC line low for as long as it takes the peripheral device to accept and decode the address.

4. While the handshake is being completed, the header generator's address decoder examines the address on the GPIB Data Bus. Since decimal 66 is on the lines, pin 5 on U6 and pin 3 on U7 go low. This makes the output of AND gate U8A go high and clock D-latch U9A. D-latch U9A enters a set state making pin 5 go high and pin 6 go low.

Receiving the Secondary Address. After the primary talk address is transferred, the 4051 places the secondary address for READ (decimal 110) on the GPIB Data Bus and sets DAV low. Both the header generator and the peripheral device complete the handshake as before. The address decoder in the header generator passes the address to pin 11 on D-latch U9B as a low to high transition. This clocks the latch into a "set" state—making pin 9 on U9B go high.

Ending the Address Sequence. After the handshake on the secondary address is completed, the 4051 assigns itself as a listener, sets NRFD and NDAC low, then releases ATN to a high state. This causes pin 3 on AND gate U10A (located just to the right of D-latch U9B) to go high and places the header generator in a talk state.

The Header Generator Keeps the Peripheral From Talking. Since the header generator must issue two header bytes before the peripheral device is allowed to transfer its ASCII string, the header generator must immediately hold the peripheral device "in check." This is how the generator does it.

1. D-latch U17A is the key element controlling the action of the peripheral device. Normally U17A is "reset" with the Q output (pin 5) low and the $\overline{Q}$ output (pin 6) high. These two outputs control the direction of data to and from the peripheral device via the Data Bus Regulation network U19, U20, U21, and U22. With U17A in a reset state, pin 12 on U19 and U20 is high and pin 12 on U21 and U22 is low. This allows data to travel from the 4051 to the peripheral device over the Data Bus. Data traveling in the opposite direction is blocked. Only after U17A is clocked into a set state can data reach the 4051 Data Bus from the peripheral device. (This does not happen until after the header generator issues the two-byte header to the 4051).

2. In addition to keeping peripheral data off the 4051 Data Bus, the outputs on U17A keep the peripheral's NRFD signal line low. When the primary talk address is received by the header generator and D-latch U9A is in a "reset" state, pin 9 on U12C (upper right corner) is also high. These two high inputs make the open-collector output of U12C go low and stay low. This tells the peripheral device that the "4051" is not ready to accept data. The peripheral device, therefore, does not set DAV low. Instead, the peripheral, thinking that NRFD is being controlled by the 4051, waits for NRFD to go high.

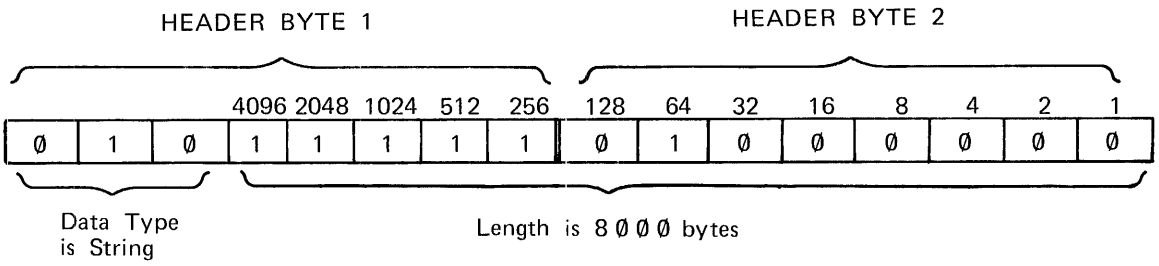
The Generator Prepares to Transfer the First Header Byte. In addition to keeping the peripheral device "in-check," the header generator prepares to transfer the first data byte as follows:

1. The output of AND gate U10A goes high after D-latch U9B is "set" and ATN goes inactive. This action "arms" the talk handshake circuit and places the first header byte on the 4051 GPIB Data Bus.

2. The talk handshake circuit consists of D-latch U11A and inverter U5D. Like the talk handshake circuit in the General Purpose Interface previously described, the D-latch remains inactive until the D-input to U11A also goes high and the latch is allowed to respond to the clock pulses from U3 pin 10.

3. In addition to "arming" the talk D-latch U11A, the high from U10A reverses the state of pin 12 on GPIB transceivers U3 and U4—making them enter a "talk" configuration. The high from U10A also "arms" D-latch U11B by releasing the clear input (pin 13) and setting the D-input (pin 12) high. The drivers in the GPIB Data Bus transceivers U1 and U2 are also enabled. These drivers place the first header byte from U15 and U16 on the 4051 GPIB Data Bus at this time.

The Header Byte Configuration. The 16-bit header is generated from a series of open-collector AND gates connected in parallel. The eight AND gates located in chips U15 and U16 make up the first header byte. These gates are controlled by a common enable coming from pin 9 on D-latch U11B. The second byte is generated by the eight AND gates in chips U13 and U14. These gates are controlled by a common enable from pin 8 on U11B. The second input on each AND gate is tied high or low, making the output of the gate represent a binary 1 or a binary Ø, respectively. The configuration shown in the schematic diagram generates the following header on the 4051 GPIB Data Bus and tells the 4051 that the peripheral device is about to send a character string 8ØØØ bytes long.



Transferring the First Header Byte. A short time after the 4051 releases ATN (37 μs to be exact), the 4051 sets NRFD high to indicate that it is ready to receive the first header byte. The header generator responds as follows:

1. The high NRFD signal line makes pin 10 on transceiver U3 go low. This makes the output of inverter U5D go high and clock D-latch U11A into a "set" state.

2. Pin 5 on D-latch U11A goes high making the output of the DAV driver (pin 9 on U4) go low. This tells the 4051 that the data on the 4051 GPIB Data Bus is valid.

3. The 4051 responds by setting NRFD low. The 4051 captures the header byte, then set NDAC high.

4. The high NDAC signal makes pin 14 on transceiver U3 go low. This low transition clocks the CLEAR input to D-latch U11A and returns the latch to a "reset" state.

5. Pin 5 on U11A goes low and returns the DAV signal line to a high (inactive) state. Pin 6 on U11A goes high and clocks D-latch U11B into a "set" state. This action disables the AND gates in U15 and U16, and enables the AND gates in U13 and U14. As a result, the second header byte is placed on the 4051 GPIB Data Bus.

6. While the 4051 is examining the first header byte, the second header byte is settling on the GPIB Data Bus. As soon as the 4051 is ready, the 4051 sets NRFD high to indicate its willingness to receive the second header byte.

7. The handshake on the second header byte takes place the same as the first. The high going NRFD signal clocks D-latch U11A into a "set" state and DAV goes low on the GPIB. The 4051 captures the second header byte, then sets NDAC high. This activates the clear input on U11A, returns the latch to a "reset" state, and sets DAV on the GPIB to a high (inactive) state.

Turning the Data Transfer Over to the Peripheral Device. At this point, the header generator has finished its function and turns the Data Transfer over to the peripheral device. The turn-over operation occurs as follows:

1. As the second header byte is placed on the GPIB Data Bus, the enable pulse from pin 9 on U11B make the D-input to U17A go high. When the 4051 completes the second handshake by setting NDAC high, D-latch U11A is "reset" and the low to high transition at pin 6 on U11A clocks D-latch U17A into a "set" condition.

2. With U17A in a set condition, the Q outputs (pins 5 and 6) reverse state. This causes the Data Bus Regulation Network (U19, U20, U21, and U22) to switch direction and place the first ASCII data byte from the peripheral device on to the 4051 GPIB Data Bus.

3. At the same time, open-collector AND gate U12D is enabled. This allows the peripheral device to "see" the true state of the NRFD signal line on the 4051 GPIB. In addition, the low at pin 6 on D-latch U17A causes AND gate U10B to be disabled. Pin 12 on GPIB Data Bus transceivers U1 and U2 goes high and the second header byte from the header generator is taken off the 4051 GPIB Data Bus. U17A pin 6 in the low state disqualifies U10C, which inhibits the header generator from issuing NRFD, turning control of the bus over to the peripheral.

From this point on, the header generator is effectively out of the picture and the peripheral device transfers ASCII data to the 4051 until the specified number of bytes (8000 in this case) are transferred. At that time, the 4051 ends the transfer by setting ATN low, issues the UNTALK command, then released ATN.

Ending the Transfer and Clearing the Bus

The header generators response to ATN occurs as follows:

1. The low ATN signal makes the output of AND gate U10A go low. This returns handshake transceivers U3 and U4 to a listen configuration; the address decoder (U6 and U7) is enabled and the talk handshake D-latches U11A and U11B are disabled (the D-inputs go low). The header generator is now ready to receive and decode addresses from the 4051 controller.

2. The peripheral device responds in a similar manner. When ATN goes low, the peripheral device prepares to receive and decode addresses from the 4051 controller. At the same time, the CLEAR input (pin 1) on D-latch U17A goes low; this "resets" U17A making the Q outputs (pins 5 and 6) reverse state and returns the Data Bus Regulation Network to a listen configuration (allows data to pass from the 4051 to the peripheral device).

3. After ATN goes low, the 4051 issues the command UNTALK (decimal 95) over the GPIB Data Bus. Both the header generator and the peripheral device receive the command at the same time.

4. The header generator address decoder passes the UNTALK command to NOR gate U23B and makes the output go low. This low is applied to the CLEAR inputs on D-latches U9A and U9B and returns both latches to a "reset" state, returning the header generator to an IDLE condition.

5. The peripheral device also decodes the UNTALK command and returns to an IDLE state.

6. After the UNTALK command is transferred, the 4051 issues the UNLISTEN command (for good measure), then releases ATN and the transfer is over.

Allowing the 4051 to Pass Data to the Peripheral Device. It is important that the header generator allow ASCII data to be transferred from the 4051 to the peripheral device with a PRINT statement, a WRITE statement, a SAVE statement, a DRAW statement, or any other output statement in 4051 BASIC. This header generator allows the data to pass because of the following design considerations.

1. When the 4051 goes through an addressing sequence, both the header generator and the peripheral device are allowed to "see" and decode the addresses. Although the header generator only responds to a primary talk address 2, the peripheral device can receive and respond to primary listen addresses when they are issued by the 4051.

2. When the header generator is not in a talk mode of operation, data is allowed to pass on the peripheral device via the Data Bus Separation Network U19, U20, U21, and U22. In addition, the peripheral device is allowed to handshake on the data because the header generators listen handshake circuitry is always active when the rest of the header generator is idle.

Section 3

A MICROPROCESSOR-BASED GPIB INTERFACE

INTRODUCTION

This section describes a working implementation of a microprocessor controlled IEEE 488 peripheral interface. The hardware and firmware described within are used to interface a peripheral mass storage device (the TEKTRONIX 4924 Digital Cartridge Tape Drive) to the IEEE 488 general purpose interface bus (GPIB). This implementation is suitable for medium speed applications (about 5k bytes/second) which contain a Motorola M68ØØ microprocessor. The design goal was to achieve a reasonable trade-off among transfer speed, cost, and firmware complexity.

Before starting the description of this particular interface, a brief review of the IEEE 488-1975 standard's characteristics will be presented. The IEEE 488-1975 standard is also called the ANSI MC1.1-1975 standard, the proposed IEC bus, the ASCII bus, the HP-IB, the GPIB and other names.

BACKGROUND

The IEEE 488 interface bus provides an internationally-standardized communication link between several instruments whether they be measurement devices, computers, mass storage devices, or other peripherals limited only by one's imagination. The charter of the IEEE 488 standard is meant to provide the logistics for signal management along the bus connecting the system devices. The standard describes the means of addressing, handling interrupt conditions and data interchange between devices as well as the electrical and mechanical specifications for the bus. As with many other standardization documents, this standard suffers from lack of readability and ease of understanding. However, it is complete and thorough. It is the objective of this document to aid in the understanding of the GPIB via a specific example.

With the advent of inexpensive microprocessors, reasonably sophisticated instruments and systems will become cost-effective and popular. Since microprocessors allow for greater flexibility and device local processing, the GPIB will likely provide a sufficient means for local date interchange for all but the highest speed applications.

Devices on the bus are grouped into three functional categories:

1. Listeners—devices which can receive data from the bus.

2. Talkers—devices which can send data along the bus.

3. Controllers—devices which provide the management function of assigning who talks to whom and when.

A device may have the capability to assume the role of any, or all, of the above three functions. Additionally, there may be only one active controller and one active talker at any one time along with any number of listeners, (up to 14, as limited by the electrical bus loading). When the controller is giving commands, it is the talker. All other instruments are forced to be listeners, listening to the commands. After the controller has finished giving commands, the bus is free for data interchange among the devices which have been instructed to communicate by the controller. Although the standard does specifically describe the logistics of addressing and data transfer, it does not make any attempt to describe the content of data messages which are passed along the bus. Although uncompatible messages may seem like a major point, it has proved to be only a minor inconvenience. This stems from two related phenomena:

1. Devices are somewhat "intelligent" in that they can perform some local data processing to format data into a palatable state, and . . .

2. Most manufacturers are using the ASCII character code for use in communicating data and status information.

The GPIB is a byte serial—8 bit parallel communication bus. In addition to the 8 bi-directional data lines there are 3 handshake control lines and 5 bus management lines.

The 3 control lines are used to provide a fully synchronized byte transfer on the 8 data lines (DIO lines). Thus, byte transfers occur under the control of a three wire handshake. The control lines, names and functions, are described below:

DAV —Data Valid. Asserted by the current talker when data is valid.

NRFD —Not Ready For Data. Asserted when one or more listening devices are not ready to receive data.

NDAC—Not Data Accepted. Asserted when the current data has not been read by all listening devices.

The five bus management lines provide the required functions to manage the usage of the bus. In some cases they augment information that appears on the data lines. The names and functions of these management lines are:

ATN —Attention. Asserted by the controller when it is issuing commands on the data lines.

IFC —Interface Clear. Asserted by the controller to reset all devices to the idle state.

SRQ —Service Request. Asserted by devices to request service. This is an asynchronous interrupt request line.

REN —Remote Enable. Asserted by the controller to activate the bus.

EOI —End or Identity. Optionally asserted by the talker to end messages. Also defined for use in a high speed parallel poll of interrupting devices.

THE DESIGN

The M68ØØ microprocessor, and associated firmware, is used to drive the GPIB interface hardware shown in Diagram D, and is also used to control other functions within the peripheral. The interface hardware consists of 1 and 1/2 PIA's (M682Ø—peripheral interface adapter), a handful of SSI and TTL devices and the required bus transceivers. The interface firmware uses about 75Ø bytes of ROM out of a total of 6K bytes of ROM contained in the 4924. The interface also requires some of the 768 bytes of R/W memory.

The hardware has three modes of operation:

1. IDLE—This is the unaddressed state where the bus drivers are disabled and the hardware is only "listening" to attention (ATN) and interface clear (IFC—system reset).

2. LISTEN—In this mode, the hardware is driving the NRFD and NDAC handshake lines while listening to the DAV handshake line as well as the associated data and management lines. The management lines used are ATN and EOI (end or identify) although REN (remote enable) is available for use if necessary.

3. TALK—In this mode, the hardware is driving the DAV handshake line and the appropriate data lines in addition to the EOI bus management line while listening to the NRFD and NDAC handshake lines. The hardware also has the capability of driving SRQ to request service.

The microprocessor is interrupted by the GPIB hardware under the following conditions:

1. IFC is asserted—Asserting IFC informs the firmware that a reset function is required.

2. ATN transitions—Both the assertion and unassertion of ATN causes the M68ØØ to be interrupted because some intervening action is required.

3. A handshake cycle relevant to the device. Revelant means that the microprocessor is not interrupted for handshake cycles for data transfers that occur on the bus when the microprocessor has NOT been addressed as a listener or talker. (See IDLE above.) If in the LISTEN mode, the hardware generates an interrupt when DAV is asserted. If in TALK mode, the hardware generates an interrupt when the last (slowest) current listener indicates a readiness to receive data by allowing NRFD to go high.

TWO HANDSHAKE EXAMPLES

The commands in these examples are given to the system controller which is assumed to be a TEKTRONIX 4051. Furthermore, the 4051 provides the other necessary complementary function, i.e., the talker function in the first sequence and the listener function in the second sequence.

The Listener

For the first example we execute an ASCII transfer of a locial record, i.e., the four character data sequence ABC<cr>, where <cr> represents the ASCII character CARRIAGE RETURN. The command is:

PRINT @1,12: "ABC"

which tells device #1 on the bus (our mass storage device) to receive (and store) the four data characters A, B, C and <cr>. The packets of information presented on the bus are shown in Fig. 3-1. A detailed handshake sequence is shown in Fig. 3-2.

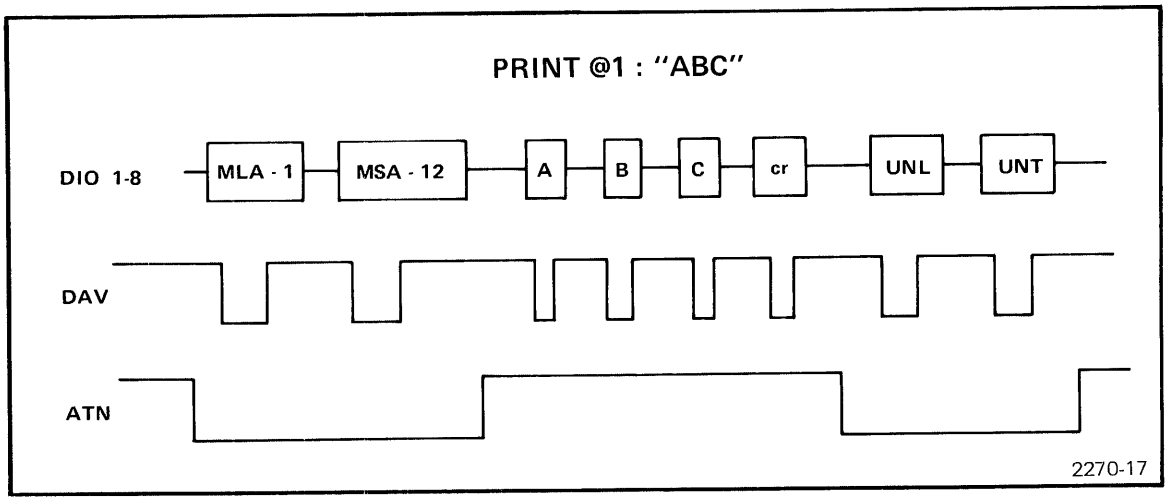


Fig. 3-1. PRINT @1,12: "ABC".

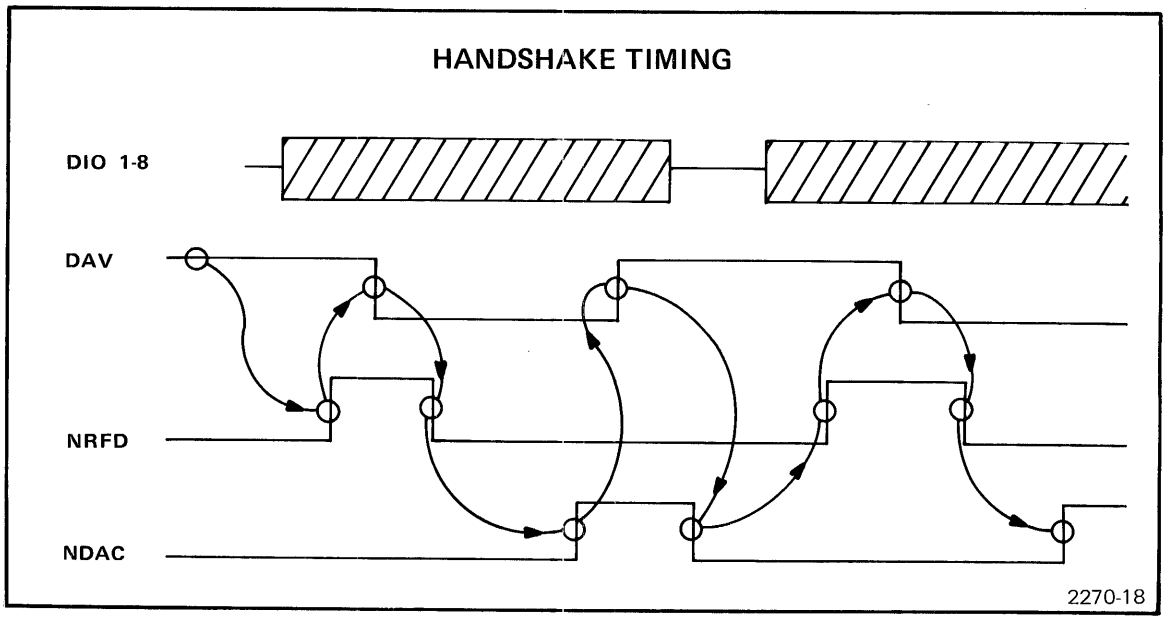


Fig. 3-2. Detailed Handshake Timing.

MLA-1 . . . The 4051 controller is setting up peripheral device #1 as a listener.

MSA-12 . . . The 4051 controller is further telling device #1 to perform the PRINT function. This is interpreted as My Secondary Address (MSA) #12 or alternatly interpreted as My Command. This byte tells the peripheral device that an ASCII transfer is comming from a talker on the bus.

ABC<cr> . . . The controller has now changed roles and is now an ordinary talker transmitting data. The data sent is the ASCII data to be stored in the buffer and eventually stored on the magnetic media.

UNL. . . The controller has finished the command and is telling the peripheral to get off the bus (stop listening). The UNT (UNTalk command) is ignored by the listener.

The text which follows, along with Figs. 3-1 and 3-2 and Diagram D, describe in detail the hardware/firmware/bus interactions.

1. ATN is asserted—This tells all peripherals on the GPIB that the controller is going to send a command (or address) to all peripherals and they must listen. The assertion of ATN causes an interrupt in the M68ØØ to inform the firmware of the event. ATN also causes the NRFD and NDAC handshake drivers to be placed on the bus in preparation for receiving the address byte.

2. DAV (data valid) is asserted—This informs the devices connected to the GPIB that the data appearing on the DIO (data) lines is valid and ready for action. Since ATN is also asserted, the peripherals interpret the data byte as a command (the primary address—MLA-1 in Fig. 3-1). The assertion of DAV causes an interrupt in the M68ØØ via the handshake line CA1—U315 pin 4Ø in Diagram D (note: this line is alternately used in TALK mode when NRFD goes high). The information presented on the data lines is interpreted by the peripheral to be its primary listen address. This is determined by the firmware looking at the backpanel hardware address switches and comparing them with the address received over the bus. Upon determining that the peripheral has been addressed, the firmware proceeds to enable the addressed bit (P86—U315 pin 16 in Diagram D) so that the hardware will be in the proper listen state after the controller is through talking, e.g. ATN goes high. Then the firmware advances to the next state. Meanwhile the hardware has caused NRFD (Not Ready For Data) to go low, indicating that the device is not ready to receive data.

3. A SHAKE pulse is issued by the firmware. This shake pulse causes the R-S flip-flop (U5a—U5b in Diagram D) to enter the reset state which in turn allows NDAC (Not Data Accept) to go high indicating to the controller that the data has been accepted. Upon seeing NDAC high, the controller proceeds to un-assert DAV (set high) while it prepares the next byte. DAV going high causes the hardware to set NRFD high as well as setting the R-S flip-flop allowing NDAC to go low. The hardware is now ready to receive the next byte.

4. The controller now issues the secondary address 12 which the peripheral accepts as before. The information passed across the GPIB thus far has set device #1 (our devices) as a listener (MLA-1), and has told it via the secondary address (MSA-12) to prepare for a PRINT operation, where the data following shall be interpreted as data to be stored in the current open file.

5. ATN goes high. Attention going high informs the peripherals on the bus that the controller is finished giving orders and the bus is now free for data interchange. This second transition of attention again causes an interrupt in the M68ØØ to inform the firmware of the change of state and the hardware acts accordingly. Since this peripheral was instructed to enter the listen state, the hardware stays on the bus in the LISTEN mode as determined by the ADDRESSED, and TALK/LISTEN lines which were set appropriately in step 2. If the peripheral had not been addressed, the ADDRESSED line would have been left un-asserted and the drivers would have been off the bus so data interchange among other peripherals could proceed at a rate unrelated to the speed of this device.

6. The talker (also the controller in this example) proceeds to send the four characters of information as data. Once again, DAV is asserted which causes an interrupt in the M68ØØ and the data to be read. This process is repeated until all appropriate data is sent and received, at which point the controller steps in again and sends the unlisten (UNL) or unaddress command.

7. Upon receiving the unaddress command the peripheral gets off the bus and begins executing the command requested which in this case was to store four characters of data.

Note that normal data transfers, those without ATN asserted, caused the firmware to store the data received in a buffer rather than being acted upon one byte at a time. This buffering is done for three reasons:

1. To allow data transfers at a maximum rate so that the talker can get about other business. This also maximizes the availability of the GPIB.

2. It is somewhat simpler to write conversion routines and scanners when a whole buffer is available at a time.

3. This peripheral is a magnetic tape drive and requires data to be buffered into physical blocks before accessing the actual recording media.

When looking at the firmware listing that follows this discussion, note that occasionally the hardware can be placed into a suspended state. These suspended states are envoked when a buffer becomes full or when the monitor (overall peripheral control program) is off doing something more important, like completing the execution of the last command. While in one of these suspended states, the hardware is left "sitting" on the bus refusing to handshake which effectively holds everything. When the monitor has completed the condition that caused the suspended state to be envoked, it can subsequently restart the handshake. One aspect of this buffer management scheme is that it is typical to receive an unaddress command (UNL or UNT) as a prerequisite to start the peripheral active as specified by the secondary address (MSA). The unaddress forces the peripheral monitor to come out from the idle state and process the current buffer even though it is not full. This generates a clean solution in that each command is explicitly delimited.

The Talker

A peripheral talk sequence proceeds in much the same way as the listen sequence previously discussed and will be illustrated by executing an input data request. The command being performed is:

```
INPUT @1:A$
```

This command requests device #1 to send a data stream from the currently open file. The bus information packets are shown in Fig. 3-3. The handshake detail is shown in Fig. 3-2.

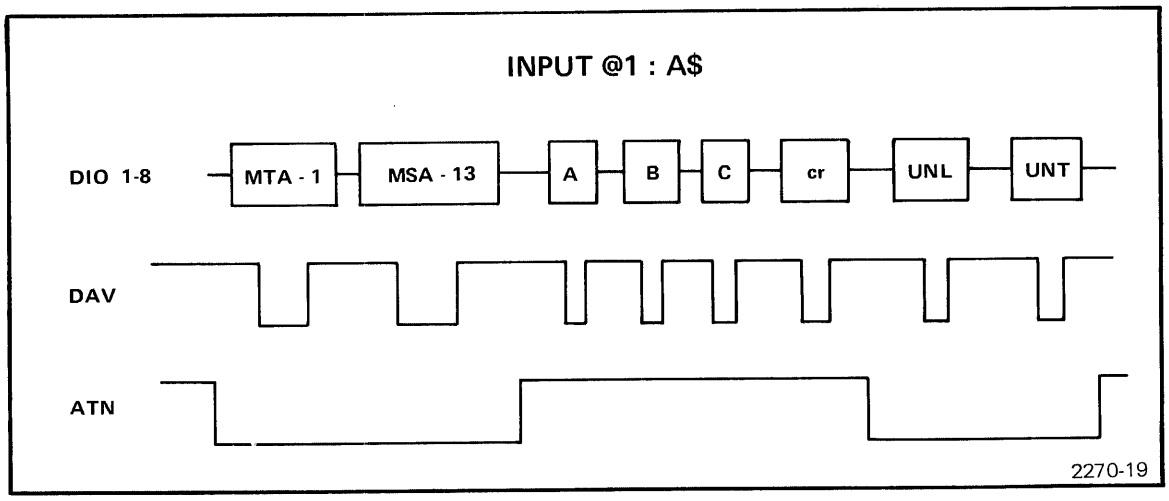


Fig. 3-3. INPUT @1:A$.

A detailed discussion of Fig. 3-3:

1. The first two command bytes, those sent with ATN asserted, are handled by the peripheral's hardware/firmware in much the same way as the previous example. However, since the first byte instructed the peripheral device to enter the talk mode, communicated by the designation MTA-1 (My Talk Address), the firmware asserts the ADDRESSED line and sets the TALK/LISTEN line to the TALK mode. This allows the hardware to enter the TALK state after ATN goes away.

2. When ATN does go away, the processor is interrupted, at which point it reverses the data registers so that they can be used for output.

3. As the last (slowest) listening device becomes ready to receive data, the NRFD line goes high. NRFD going high causes an interrupt via the HAND line. This in turn, tells the firmware to put a data byte on the DIO (data) lines and to issue the SHAKE pulse.

4. While in the TALK state, a SHAKE pulse controls a second R-S flip-flop (U5c—U5d in Diagram D). SHAKE places the flip-flop into the Set state which in turn asserts the DAV line on the bus, indicating that the data is valid. As the slowest listener accepts the data, the NDAC line is driven high. This action Resets the flip-flop, taking away DAV, and allows the listeners to once again enter the RFD (Ready For Data) state.

5. As the RFD state of the listener is reached, an interrupt is once again received on the HAND line and the process repeats itself until the controller is satisfied that enough data has been passed, at which point attention (ATN) is reasserted.

6. This example is a special case where the controller and listener are the same device. In general, the TALKER asserts EOI with the last byte of the transfer, thereby signalling the CONTROLLER that the data transfer is complete.

7. ATN is asserted by the CONTROLLER. At this point, the hardware is forced into a listen state by some appropriate gating. The firmware is informed of this action by an interrupt on the attention line and proceeds to interpret further interrupts of the HAND line as being data to be received (note: the data register must be turned around to receive data). Since the message received was an Untalk (UNT), the firmware informs the hardware of this change in state by clearing the ADDRESSED line and setting the TALK/LISTEN bit back to the LISTEN mode. (The UNL command was received and ignored.)

While in the TALK mode, the bus can be suspended by running out of data in a buffer. This generally calls on the monitor to get another physical buffer from the tape. After getting more data to work with, the handshake is continued.

MISCELLANEOUS COMMENTS

The following are some general implementation comments and/or suggestions.

The EOI (End or Identify) line is presented as a level and can be received or transmitted by the firmware when required. In this particular implementation, the EOI line is used during some operations to indicate the end of information or end of file.

The SRQ line may be activated by the firmware for requesting service. In the 4924, it is used primarily to indicate that an error condition exists. For example, SRQ is activated when a write operation is commanded with the media in a write-protected state.

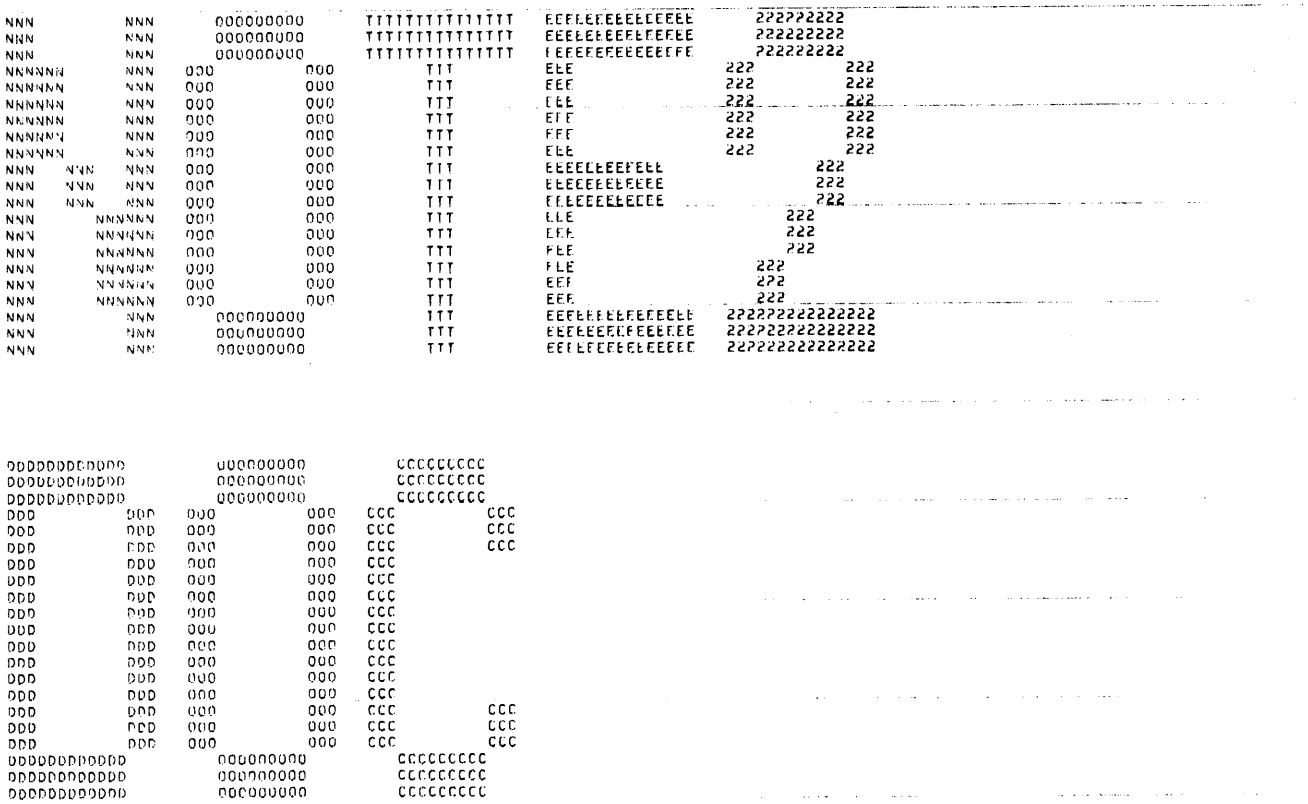
A digital debounce chip (U115 M14490 in Diagram D) is included on the control lines to help control noise problems. However, the delay also slows down the bus and it may be eliminated.

The careful observer will note that the hardware does not really get off the bus when IFC is asserted. The IFC function is executed by the firmware and as such may take a while to execute. This is not considered too serious, but can be corrected with the addition of a latch and some appropriate gates and a clearing mechanism.

NOBODY—This line is asserted anytime both NRFD and NDAC are sensed high. This condition means that nobody is listening to the bus which is considered an error state if someone is talking. It is appropriate that the talking device either exit from its talk state or wait until someone is listening. In this implementation, it was decided that the device would wait if nobody was on the bus and it was operating in the normal on-line mode. However, if it was operating in the off-line data logging mode (with no controller, it uses front panel controls), the device would exit from the current command.

MACROASSEMBLER LISTING

The following is a macroassembler listing which is substantially the same as the interface driver program used in the TEKTRONIX 4924. This list is believed to be accurate and correct, however, Tektronix can not guarantee that it is. This list is presented here for educational purposes only.



```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32
TABLE OF CONTENTS

    2-  1       HWISRV  INTERFACE INTERRUPT HANDLER
    3-  1       UTILS--- IFC, UNDRES, SETSRQ
    4-  1       ATNTRU--- ATN, HAS BECOME TRUE
    5-  1       ATNFAL--- ATN, HAS BECOME FALSE
    6-  1       HANDSK---- DAV, OR NRFD CONTROL
    7-  1       ATNBYT--- ACCEPT ATNENTION COMMAND BYTE
    8-  1       MLASUB, MTASUB --- PRIMARY ADDRESS REQUEST
    9-  1       SPE------ SERIAL POLL CONTROL
   10-  1       RCVBYT--- ENTER A DATA BYTE INTO BUFFER
   11-  1       SNDBYT--- SEND A DATA BYTE FROM THE BUFFER
   13-  1       TLKTBL, LSNTBL --- SEC. ADR. TABLES
   14-  1       IFCRST--- RESTARTS HANDSHAKES AFTER A SUSPENSION
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 1

                                        .TITLE  GPIB INTERFACE AN EXAMPLE
                                        .ENABLE LC
                                ;
                                ; Some globals and constants.
                                ;
                                        .GLOBL  HWMODE          ; Hardware mode byte
                                ;       Hwlsn   =       1       ; listen mode
                                ;       Hwtlk   =       128.    ; talk mode
                                ;       Hwlsns  =       4       ; listen suspended
                                ;       Hwtlks  =       8       ; talk suspended
                                ;       Hwmas   =       16      ; pri. adrs. suspended
                                ;       Hwunas  =       2       ; unaddress suspended
                                ;       HwIFCs  =       32      ; IFC suspended
                                ;       Hwsusp  =       hwlsns+hwtlks+hwmas+hwunas+hwIFCs
                                ;       Hwsusn  =       255.-hwsusp
                                        .GLOBL  EOIPTR          ; Pointer to EOI byte
                                        .GLOBL  SPEFG           ; Flag for serial poll enabled
                                        .GLOBL  ATNFG           ; Flag for ATN true
                                        .GLOBL  CNREGA          ; Cont reg and out reg used with idx
                                        .GLOBL  DTREGA          ; Clears control reg and stores
                                                                ; Cont reg and out reg
                                        .GLOBL  CNREGB          ; Same except its attention
                                        .GLOBL  DTREGB
                                        .GLOBL  ALTFLG          ; Set if in alternate format mode
                                        .GLOBL  NEWCMD          ; Pointer to new command block
                                                                ;  --- used by monitor
                                                                ; And is the main communication between
                                                                ; Interupt level and program level

                                        .GLOBL  PIAGPA          ; Iec data PIA
                                        .GLOBL  PIAGPB          ; Iec control line PIA
                                        .GLOBL  PIAADR          ; Iec address switch PIA
                                        .GLOBL  SNDBYT          ; Routine to send a byte from
                                                                ; The current output buffer
                                        .GLOBL  RCVBYT          ; Routine to stuff new char. into
                                                                ; Current input buffer.
                                        .GLOBL  OFFLIN          ; Set by monitor if in offline mode
                                ;
                0010                    ATNLVL  = 16.           ; Atn. dc level input
                0008                    HNDLVL  = 8.            ; Hand dc level input
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 2
HWISRV  INTERFACE INTERRUPT HANDLER

                                        .SBTTL  HWISRV  INTERFACE INTERRUPT HANDLER
                                ;
                                ; This is a "recursive" interrupt service routine
                                ; After servicing one interrupt request control is passed back to
                                ; The beginning <hwisrv> UNTil no more interrupts are
                                ; Pending in the iec bus registers.
                                ; Cnrega,cnregb save the contents of control regs. A and B
                                ; Respectively before resetting the interrupt bits in the PIA.
                                ; The saved control regs. and "recursivness" make it such that
                                ; No interrupts can be missed.
                                ;
                                ;
                                ; Note: the hardware is configured such that
                                ; An --- LDX PIAXXX-1 --- will read the control reg. then
                                ; The data reg. <which clears the hardware interrupt req.>
                                ;
                                ;
                                        .GLOBL  HWISRV
0000    96      00G     HWISRV: LDA A   CNREGA,D        ; Maintain homoginiety
0002    F6      FFFFG           LDA B   PIAGPA-1        ; Lets close off that window!!!!
0005    C5      C0              BIT B   #H0C0,I
0007    27      09              BEQ     3$              ; Go around if no interrupt there
0009    FE      FFFFG           LDX     PIAGPA-1        ; Get A side
000C    DF      00G             STX     CNREGA,D
000E    9A      00G             ORA A   CNREGA,D
0010    97      00G             STA A   CNREGA,D
0012    96      00G     3$:     LDA A   CNREGB,D
0014    F6      FFFFG           LDA B   PIAGPB-1        ; Go around if no interrupt there
0017    C5      C0              BIT B   #H0C0,I
0019    27      09              BEQ     4$
001B    FE      FFFFG           LDX     PIAGPB-1        ; Get B side
001E    DF      00G             STX     CNREGB,D
0020    9A      00G             ORA A   CNREGB,D
0022    97      00G             STA A   CNREGB,D
0024    96      00G     4$:     LDA A   CNREGA,D
0026    85      40              BIT A   #H40,I          ; Ifc asserted?
0028    26      14              BNE     IFC             ; Yes-service it
002A    96      00G             LDA A   CNREGB,D        ; Now ATN asserted?
002C    85      40              BIT A   #H40,I          ; No-ATN going away?
002E    27      03              BEQ     2$
0030    7E      00D3'           JMP     ATNFAL          ; Service ATN, false
0033    4D              2$:     TST A
0034    2B      72              BMI     ATNTRU          ; Service ATN true
0036    96      00G             LDA A   CNREGA,D        ; Handshake interrupt
0038    2B      01              BMI     1$              ; Yes-service interrupt
003A    3B                      RTI
003B    7E      0119'   1$:     JMP     HANDSK
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 3
UTILS--- IFC, UNDRFS, SETSRQ

                                        .SBTTL  UTILS--- IFC, UNDRFS, SETSRQ
                                ;
                                ;
                                ; Service IFC-reset interface functions,
                                ; If SRQ was asserted,reset it too,
                                ; When 4924 is addressed again,SRQ will be asserted
                                ; Again,(.......or??????)
                                ;
                                        .GLOBL  IECPIA, PIASET
                                        .GLOBL  HWSUSP
                                        .GLOBL  DTREGB
                                ;
003E    8D      02              IFC:    BSR     IFC1
0040    20      BE                      BRA     HWISRV
                                ;
                                ;
                                        .GLOBL  IFC1, HWIFCS
0042    96      00G             IFC1:   LDA A   CNREGB,D        ; Save ATN status
0044    36                              PSH A
0045    96      00G                     LDA A   CNREGA,D        ; Save other status
0047    36                              PSH A
0048    CE    0000G                     LDX     IECPIA,I        ; Reset PIA's and edge's
004B    BD    0000G                     JSR     PIASET
004E    96      00G                     LDA A   HWMODE,D        ; See if have anything suspended
0050    85      00G                     BIT A   HWSUSP,I
0052    27      05                      BEQ     1$
0054    B7    0000G                     STA A   PIAADR          ; If suspended clean up hardware
0057    4F                      1$:     CLR A                   ; Reset some flags
0058    97      00G                     STA A   HWMODE,D
005A    97      00G                     STA A   SPEFG,D
005C    32                              PUL A                   ; Restore byte status
005D    84      80                      AND A   #H80,I
005F    9A      00G                     ORA A   CNREGA,D
0061    97      00G                     STA A   CNREGA,D
0063    32                              PUL A                   ; Reset ATN status
0064    84      C0                      AND A   #H0C0,I
0066    9A      00G                     ORA A   CNREGB,D
0068    97      00G                     STA A   CNREGB,D
006A    20      09                      BRA     IFCDON
                                ;
                                ; This is the program/interrupt level routine to clear the
                                ; Monitor status.
                                        .GLOBL  IFCCLR
                                ;
006C    DE      00G             IFCCLR: LDX     NEWCMD,D        ; See if can set up clear command
006E    27      05                      BEQ     IFCDON
0070    86      00G                     LDA A   HWIFCS,I        ; Set IFC suspended
0072    97      00G                     STA A   HWMODE,D
0074    39                              RTS
0075    96      00G             IFCDON: LDA A   CMMODE,D        ; Allready idle?
0077    27      05                      BEQ     IFC,,
0079    CE    0000G                     LDX     CLRCMD,I
007C    DF      00G                     STX     NEWCMD,D
007E    39                      IFC,,:  RTS
                                ;
                                ;
                                ; Unaddress function
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 3+
UTILS--- IFC, UNDRES, SETSRQ

                                        ; If interface enabled --- disable the hardware and tell
                                        ; The monitor that an unaddress occured,
                                        ;
                                                .GLOBL  CLRCMD, NEWCMD
                                                .GLOBL  UNDRES, HWUNAS
 007F    96    00G       UNDRES: LDA A   HWMODE,D        ; Check if enabled
 0081    7F  0000G               CLR     HWMODE
 0084    85    81                BIT A   #H81,I          ; Lsn or Tlk?
 0086    27    0B                BEQ     2$
 0088    8D    E2                BSR     IFCCLR
 008A    96    00G               LDA A   DTREGB,D
 008C    8A    60                ORA A   #H60,I          ; Ln=>1
 008E    97    00G               STA A   DTREGB,D
 0090    B7  0000G               STA A   PIAGPB
 0093    B7  0000G       2$:     STA A   PIAADR          ; Toggle shake
 0096    39                      RTS
                                        ;
                                        ; Assert SRQ on the bus
                                        ; Note: this routine raises, then lowers the SRQ
                                        ; Line in order to fix an early production 4051 bug,
                                        ;
                                                .GLOBL  SETSRQ
                                        ;
 0097    96    00G       SETSRQ: LDA A   DTREGB,D
 0099    8A    02                ORA A   2,I             ; This is to fix 4051 bug
 009B    97    00G               STA A   DTREGB,D
 009D    B7  0000G               STA A   PIAGPB
 00A0    84    FD                AND A   375,I           ; Assert SRQ
 00A2    97    00G               STA A   DTREGB,D
 00A4    B7  0000G               STA A   PIAGPB
 00A7    39                      RTS
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 4
ATNTRU--- ATN, HAS BECOME TRUE

                                                .SBTTL  ATNTRU--- ATN, HAS BECOME TRUE
                                        ;
                                        ;
                                        ; Controller has asserted attention
                                        ; So arm handshake interrupt and
                                        ; Get ready to listen,
                                        ;
 00A8    8D    03        ATNTRU: BSR     ATNTR
 00AA    7E  0000'               JMP     HWISRV
                                        ;
                                        ;
                                                .GLOBL  ATNTR
                                        ;
 00AD    96    00G       ATNTR:  LDA A   CNREGB,D
 00AF    84    7F                AND A   #1-128.,I
 00B1    97    00G               STA A   CNREGB,D
 00B3    86    FF        1$:     LDA A   #H0FF,I         ; Set ATN flg
 00B5    97    00G               STA A   ATNFG,D
 00B7    96    00G               LDA A   CNREGA,D        ; Arm handshake interrupt
 00B9    8A    01                ORA A   1,I
 00BB    97    00G               STA A   CNREGA,D
 00BD    B7  FFFFG               STA A   PIAGPA+1
 00C0    96    00G               LDA A   DTREGB,D
                                                        ; Set up for listen
 00C2    8A    20                ORA A   #H20,I
 00C4    97    00G               STA A   DTREGB,D
 00C6    B7  0000G               STA A   PIAGPB
 00C9    2A    07                BPL     3$
 00CB    84    FE                AND A   #H0FE,I         ; Clear EOI
 00CD    97    00G               STA A   DTREGB,D
 00CF    B7  0000G               STA A   PIAGPB
 00D2    39        3$:     RTS
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MACRO  VM02-10   8-Oct-76 14:21:32 PAGE 5
ATNFAL--- ATN, HAS BECOME FALSE

                                                .SBTTL  ATNFAL--- ATN, HAS BECOME FALSE
                                        ;
                                        ;
                                        ; Controller has released attention,
                                        ; So check whether 4924 has
                                        ; Been assigned as talker or listener,
                                        ; If so set up the interface accordingly,
                                        ; If not, disarm the handshake interrupt
                                        ; And remain in idle state,
                                        ;
                                        ;
                                        ;
                                        .GLOBL  TEVEN,TLODD,LEVEN,LSEVEN,TKEVEN,LSODD,CMDODD,TKODD
                                        .GLOBL  MSAOK,ALTFLG,ALTCMD,CMDACT
                                        ;
                                        ;
00D3    8D      03              ATNFAL: BSR     ATNFLS
00D5    7E      0000'                   JMP     HWISRV
                                        ;
                                        ;
                                                .GLOBL  ATNFLS
                                        ;
00D8    96      00G             ATNFLS: LDA A   CNREGB,D
00DA    84      BF                      AND A   #-1-64,,I       ; 127 or 3f the hard way
00DC    97      00G                     STA A   CNREGB,D
00DE    96      00G                     LDA A   ATNFG,D         ; Atn flag set?
00E0    2A      1B                      BPL     4$
00E2    7F      0000G                   CLR     ATNFG           ; Clear flag
00E5    96      00G                     LDA A   HWMODE,D        ; Am I in talk?
00E7    2B      15                      BMI     2$              ; Yeswset up to talk
00E9    26      12                      BNE     4$              ; If idle, disarm handshake int
00EB    96      00G                     LDA A   CNREGA,D
00ED    84      FE                      AND A   #H0FE,I
00EF    97      00G                     STA A   CNREGA,D
00F1    B7      FFFFG                   STA A   PIAGPA=1
00F4    96      00G                     LDA A   DTREGB,D        ; Reset address line
00F6    8A      40                      ORA A   100,I
00F8    97      00G                     STA A   DTREGB,D
00FA    B7      0000G                   STA A   PIAGPB
00FD    39                      4$:     RTS
                                        ;
                                        ;
00FE    D6      00G             2$:     LDA B   DTREGB,D        ; Go to talk state
0100    C4      1E                      AND B   #H1E,I          ; Make sure EOI is not true
0102    D7      00G                     STA B   DTREGB,D
0104    F7      0000G                   STA B   PIAGPB
0107    96      00G                     LDA A   CNREGA,D        ; Address direction reg
0109    84      FB                      AND A   #H0FB,I
010B    B7      FFFFG                   STA A   PIAGPA=1
010E    C6      FF                      LDA B   #H0FF,I         ; Set up all outputs
0110    F7      0000G                   STA B   PIAGPA
0113    8A      04                      ORA A   4,I
0115    B7      FFFFG                   STA A   PIAGPA=1
0118    39                              RTS
```

```
GPIB INTERFACE AN EXAMPLE      RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 6
HANDSK---- DAV, OR NRFD CONTROL

                                        .SBTTL  HANDSK---- DAV, OR NRFD CONTROL
                                ;
                                ;
                                ; This routine may be servicing DAV or NRFD
                                ; Depending whether the 4924 is in listen or
                                ; Talk state.
                                ;
                                        .GLOBL  DTREGA
                                ;
0119    96      00G     HANDSK: LDA A   CNREGA,D        ; Get present control reg,
011B    84      7F              AND A   #H7F,I
011D    97      00G             STA A   CNREGA,D
011F    D6      00G             LDA B   DTREGA,D        ; Retrieve data reg,
0121    96      00G             LDA A   ATNFG,D         ; Attntion on?
0123    27      04              BEQ     1$              ; No=branch
0125    8D      18              BSR     ATNBYT          ; Decode command
0127    20      13              BRA     HANDXT
0129    96      00G     1$:     LDA A   HWMODE,D        ; Am I in talk?
012B    2B      05              BMI     TALK            ; Yes=get talk byte
012D    8D      0216'           JSR     RCVBYT          ; Get byte and try to stash it
0130    20      0A              BRA     HANDXT
0132    96      00G     TALK:   LDA A   SPEFG,D         ; Was serial polling enabled?
0134    27      03              BEQ     3$              ; No=unload byte from data buffer
0136    7E      01D1'           JMP     SPE             ; Service serial polling
0139    8D      0263'   3$:     JSR     SNDBYT          ; Try to send byte
013C    7E      0000'   HANDXT: JMP     HWISRV
```

```
GPIB INTERFACE AN EXAMPLE      RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 7
ATNBYT--- ACCEPT ATNENTION COMMAND BYTE

                                        .SBTTL  ATNBYT--- ACCEPT ATNENTION COMMAND BYTE
                                ; Atnbyt-routine to decode attention command groups
                                ; And dispatches accordingly.
                                        .GLOBL  MSACMD
                                        .GLOBL  ATNBYT
                                ;
013F    D6      00G     ATNBYT: LDA B   DTREGA,D
0141    17                      TBA                     ; Get data byte
0142    C4      1F              AND B   #H1F,I
0144    84      E0              AND A   #H0E0,I
0146    27      19              BEQ     3$              ; Do control functions
0148    81      60              CMP A   #H60,I          ; MSA?
014A    26      09              BNE     1$              ; No=keep checking
014C    96      00G             LDA A   HWMODE,D
014E    85      81              BIT A   #H81,I          ; Check if addressed
0150    27      20              BEQ     SHAKER
0152    7E      02C8'           JMP     MSACMD          ; Do it if addressed
                                ;
0155    81      40      1$:     CMP A   #H40,I          ; MTA?
0157    26      02              BNE     2$              ; No=continue
0159    20      3C              BRA     MTASUB          ; Yes=service it
                                ;
015B    81      20      2$:     CMP A   #H20,I          ; MLA?
015D    26      13              BNE     SHAKER          ; No=continue
015F    20      15              BRA     MLASUB
                                ;
0161    C1      18      3$:     CMP B   #H18,I          ; Serial poll enable?
0163    26      06              BNE     4$
0165    86      FF              LDA A   377,I
0167    97      00G             STA A   SPEFG,D         ; Yes=set spe flag
0169    20      07              BRA     SHAKER
016B    C1      19      4$:     CMP B   31,I            ; Serial poll disable?
016D    26      03              BNE     SHAKER
016F    7F      0000G           CLR     SPEFG           ; Yes=clear spe flag
0172    B7      0000G   SHAKER: STA A   PIAADR          ; Toggle shake
0175    39                      RTS
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:52 PAGE 8
MLASUB, MTASUB --- PRIMARY ADDRESS REQUEST

                                        .SBTTL  MLASUB, MTASUB --- PRIMARY ADDRESS REQUEST
                                ;
                                ; Look at listen address
                                ;
0176    C1      1F      MLASUB: CMP B   #H1F,I          ; Unlisten command
0178    26      03              BNE     1$              ; No-return
017A    7E      007F'           JMP     UNDRES
017D    8D      43      1$:     BSR     RDADDR          ; Check address switch
017F    26      F1              BNE     SHAKER          ; If not mine then just shake
0181    DE      00G             LDX     NEWCMD,D        ; Else: if command pending set susp. bit
0183    26      45              BNE     MASUSP
0185    8D      02              BSR     MLASET          ; Else: set listen mode and shake
0187    20      E9              BRA     SHAKER
                                ;
                                ;
                                ; Routine to set hardware into listen mode
                                ;
                                        .GLOBL  MLASET
0189    96      00G     MLASET: LDA A   DTREGB,D
018B    84      BF              AND A   #H0BF,I         ; Set hardware into addressed mode
018D    97      00G             STA A   DTREGB,D
018F    B7      0000G           STA A   PIACPB
0192    86      01              LDA A   1,I             ; Flag listen mode
0194    97      00G             STA A   HWMODE,D
0196    39                      RTS
                                ;
                                ; Look at talk address
                                ;
0197    C1      1F      MTASUB: CMP B   #H1F,I          ; Is this UNTalk?
0199    26      13              BNE     1$              ; No-check for MTA
019B    BD      007F'   3$:     JSR     UNDRES          ; Untalk: really clean up hardware also
019E    96      00G             LDA A   CNREGA,D        ; Address direction reg
01A0    84      FB              AND A   #H0FB,I
01A2    B7      FFFFG           STA A   PIAGPA-1
01A5    7F      0000G           CLR     PIAGPA          ; Set up all inputs
01A8    8A      04              ORA A   4,I
01AA    B7      FFFFG           STA A   PIAGPA-1
01AD    39                      RTS
01AE    8D      12      1$:     BSR     RDADDR          ; Read address switches
01B0    26      0A              BNE     2$              ; Not mine so do implied UNT
01B2    DE      00G             LDX     NEWCMD,D        ; Else: if command pending set susp. bit
01B4    26      14              BNE     MASUSP
01B6    86      80              LDA A   #H80,I          ; Else: set talk flag
01B8    97      00G             STA A   HWMODE,D
01BA    20      B6              BRA     SHAKER          ; And shake
                                ;
01BC    96      00G     2$:     LDA A   HWMODE,D        ; Am I talker?
01BE    2A      B2              BPL     SHAKER          ; No so no action
01C0    20      D9              BRA     3$              ; Else do UNTalk
                                ;
                                ;
                                ; This is the place where we read the address switch
                                ;
                                ;
01C2    B6      0000G   RDADDR: LDA A   PIAADR          ; Get address
01C5    43                      COM A                   ; Complement to get true data
01C6    84      1F              AND A   37,I            ; Turn off extra bits
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 8+
MLASUB, MTASUB --- PRIMARY ADDRESS REQUEST

    58   01C8    11                          CBA
    59   01C9    39                          RTS
    60                           ;
    61                           ;
    62                           ; Suspend primary address handshake
    63                           ;
    64                           ;
    65                                       .GLOBL  HWMAS           ; Suspend primary adrs.
    66   01CA    86    00G      MASUSP: LDA A   HWMAS,I
    67   01CC    9A    00G              ORA A   HWMODE,D
    68   01CE    97    00G              STA A   HWMODE,D
    69   01D0    39                     RTS
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 9
SPE------ SERIAL POLL CONTROL

     1                                       .SBTTL  SPE------ SERIAL POLL CONTROL
     2                           ;
     3                           ;
     4                           ; Service serial poll
     5                           ; Send status byte
     6                           ;
     7                           ;
     8                                       .GLOBL  ERRCD, CMMODE, ALTFLG, PIAKYB, ONLIN
     9   01D1    96    00G      SPE:    LDA A   DTREGB,D
    10   01D3    36                     PSH A                   ; Save for later
    11   01D4    8A    02               ORA A   2,I             ; Reset SRQ
    12   01D6    B7    0000G            STA A   PIAGPB
    13   01D9    97    00G              STA A   DTREGB,D
    14   01DB    8D    10               BSR     POLSTT          ; Go get status byte
    15   01DD    32                     PUL A                   ; See if I was one who caused SRQ
    16   01DE    85    02               BIT A   2,I
    17   01E0    26    02               BNE     5$
    18   01E2    CA    40               ORA B   64.,I
    19   01E4    F7    0000G    5$:     STA B   PIAGPA          ; Store byte
    20   01E7    B7    0000G            STA A   PIAADR          ; Like shaker
    21   01EA    7E    0000'            JMP     HWISRV
    22                           ;
    23                           ;
    24                           ;       Establish poll status byte
    25                           ;
    26                                       .GLOBL  EREOF, EREOM
    27                                       .GLOBL  POLSTT
    28   01ED    5F               POLSTT: CLR B                   ; Form status byte
    29   01EE    96    00G              LDA A   ERRCD,D
    30   01F0    27    0E               BEQ     1$
    31   01F2    CA    20               ORA B   32.,I           ; Set error bit
    32   01F4    81    00G              CMP A   EREOF,I
    33   01F6    26    02               BNE     5$
    34   01F8    CA    01               ORA B   1,I             ; Set EOF bit
    35   01FA    81    00G      5$:     CMP A   EREOM,I
    36   01FC    26    02               BNE     1$
    37   01FE    CA    02               ORA B   2,I             ; Set EOM bit
    38   0200    96    00G      1$:     LDA A   CMMODE,D
    39   0202    27    02               BEQ     2$
    40   0204    CA    10               ORA B   16.,I           ; Set busy bit
    41   0206    96    00G      2$:     LDA A   ALTFLG,D
    42   0208    27    02               BEQ     3$
    43   020A    CA    08               ORA B   8.,I           ; Set alternate mode bit
    44   020C    B6    0000G    3$:     LDA A   PIAKYB
    45   020F    85    00G              BIT A   ONLIN,I
    46   0211    26    02               BNE     4$
    47   0213    CA    04               ORA B   4,I             ; Set online bit
    48   0215    39               4$:     RTS
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 10
RCVBYT--- ENTER A DATA BYTE INTO BUFFER

                                        .SBTTL  RCVBYT--- ENTER A DATA BYTE INTO BUFFER
                                ; This is the interrupt level routine to enter a byte into the
                                ; Present output buffer
                                ;
                                        .GLOBL  A.IN, A.OUT, A.MAX      ; Buffer pointers
                                ;
                                        .GLOBL  RCVBYT
                                        .GLOBL  ERRCD,PIAADR, BFRFUL, BFRSTT, NEWCMD, DTREGA
                                        .GLOBL  DTREGB,CMDACT
                                ;
                                ;
                                ;
                        0040            CMDVRR=^H40
                                ;
                                ;
                        0080            CMDVAL=^H80
0216    DE      00G             RCVBYT: LDX     NEWCMD,D
0218    26      42                      BNE     NOGOOD
021A    96      00G                     LDA A   ERRCD,D
021C    27      03                      BEQ     4$
021E    7E    02C4'             5$:     JMP     SHAKEY          ; If error set then just shake
0221    96      00G             4$:     LDA A   CMMODE,D        ; Look at monitor status"
0223    27      F9                      BEQ     5$              ; If no command processing then throw out byte
0225    96      00G                     LDA A   BFRSTT,D        ; Test if buffer available
0227    26      33                      BNE     NOGOOD          ; If buffers full then set susp. bit
0229    DE      00G                     LDX     A.IN,D          ; Get pointer
022B    96      00G                     LDA A   DTREGA,D        ; Get byte
022D    A7      00                      STA A   0,X             ; Save the byte
022F    08                              INX
0230    DF      00G                     STX     A.IN,D
0232    96      00G                     LDA A   ASCFNC,D        ; Need to look at EOI?
0234    27      0B                      BEQ     6$
0236    8D      16                      BSR     DTBSAV          ; Save current hardware status
0238    2A      07                      BPL     6$
023A    DE      00G                     LDX     A.IN,D          ; Mark EOI byte if EOI set
023C    09                              DEX
023D    DF      00G                     STX     EOIPTR,D
023F    20      07                      BRA     3$              ; Also issue end so monitor can act on it
0241    DE      00G             6$:     LDX     A.IN,D
0243    09                              DEX
0244    9C      00G                     CPX     A.MAX,D         ; Buffer full?
0246    26      7C                      BNE     SHAKEY
0248    86      00G             3$:     LDA A   BFRFUL,I        ; Set buffer full
024A    97      00G                     STA A   BFRSTT,D
024C    20      76                      BRA     SHAKEY
                                ;
                                ;
                                ; Routine to carefully look at iec bus control line
                                ; Register without losing interrupts.
                                ;
024E    D6      00G             DTBSAV: LDA B   CNREGB,D        ; Carefully so don't lose interrupts
0250    FE    FFFFG                     LDX     PIAGPB=I
0253    DF      00G                     STX     CNREGB,D
0255    DA      00G                     ORA B   CNREGB,D
0257    D7      00G                     STA B   CNREGB,D
0259    D6      00G                     LDA B   DTREGB,D        ; See if EOI asserted
025B    39                              RTS
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 11
SNDBYT--- SEND A DATA BYTE FROM THE BUFFER

                                        .SBTTL  SNDBYT--- SEND A DATA BYTE FROM THE BUFFER
                                ; This is the interrupt level routine to send a byte from the present
                                ; Data buffer
                                ;
                                        .GLOBL  SNDBYT, BFREMP, PIAGPA
                                        .GLOBL  EREOF, ASCFNC
                                        .GLOBL  CLNUP
0004                                    HELLO   = 4             ; Bit to see if anybody listening on
                                                                ; The bus.
                                                                ; Only used in offline mode.

                                ;
                                ;
                                ; These routines are used to put the hardware in a suspended
                                ; State so that a return from the interrupt level can be effected.
                                ; A suspended state will be envoked if buffers are not available
                                ; Or past commands haven't finished cleaning up.
                                ;
025C    86      00G     NOGOOD: LDA A   HWLSNS,I        ; Set listen suspended
025E    9A      00G     EXSUSP: ORA A   HWMODE,D        ; Shared suspension exit
0260    97      00G             STA A   HWMODE,D
0262    39                      RTS
                                ;
                                ;
                                ;
0263    DE      00G     SNDBYT: LDX     NEWCMD,D        ; See if can send a byte
0265    26      0C              BNE     1$              ; Nope so set suspended
0267    96      00G             LDA A   ERRCD,D         ; If error set send back "h0ff"
0269    26      49              BNE     3$
026B    96      00G             LDA A   BFRST1,D        ; Check if buffer available
026D    27      04              BEQ     1$
026F    86      00G             LDA A   HWTLKS,I        ; Set talk suspended is encounter empty buffer,
0271    20      EB              BRA     EXSUSP
0273    96      00G     1$:     LDA A   CMMODE,D        ; If no command then send dummy byte
0275    27      3D              BEQ     3$
0277    DE      00G             LDX     A.OUT,D         ; Get char pointer
0279    A6      00              LDA A   0,X             ; Get char
027B    2A      12              BPL     4$              ; Don't worry about it if high bit off
027D    D6      00G             LDA B   ASCFNC,D        ; Need to check for ASCII logical EOF
027F    27      0E              BEQ     4$
0281    81      FF              CMP A   255.,I          ; Really ASCII EOF
0283    26      0A              BNE     4$
0285    C6      00G             LDA B   EREOF,I         ; Set EOF
0287    D7      00G             STA B   ERRCD,D
0289    86      00G             LDA A   BFREMP,I        ; Set buffer empty to force action by dispatcher
028B    97      00G             STA A   BFRST1,D
028D    20      25              BRA     3$
                                ;
                                ; Work on sending the data byte
                                ;
028F    D6      00G     4$:     LDA B   OFFLIN,D
0291    2A      0B              BPL     5$
0293    8D      B9              BSR     DTBSAV          ; If offline check hello bit
0295    C5      04              BIT B   HELLO,I
0297    27      05              BEQ     5$              ; If nobody there then abort command
0299    86      00G             LDA A   CLNUP,I
029B    97      00G             STA A   CMMODE,D        ; Set abort
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 11+
SNDBYT--- SEND A DATA BYTE FROM THE BUFFER

029D    39                      RTS
029E    97      00G     5$:     STA A   DTREGA,D        ; Set data byte on bus
02A0    B7    0000G             STA A   PIAGPA
02A3    DE      00G             LDX     A.OUT,D         ; At end?
02A5    9C      00G             CPX     A.MAX,D
02A7    26      06              BNE     2$
02A9    86      00G             LDA A   BFREMP,I
02AB    97      00G             STA A   BFRST1,D        ; If end: set buffer empty flag
02AD    20      15              BRA     SHAKEY
02AF    08              2$:     INX                     ; Update pointer
02B0    DF      00G             STX     A.OUT,D
02B2    20      10              BRA     SHAKEY          ; Finish handshake
                                ;
                                ; Send EOF with EOI
                                ; This is also used for the dummy byte if haven't anything better
                                ; To send.
                                ; This causes the 4051 to abort the current input condition
                                ; Because it looks like an EOF.
                                ;
02B4    86      FF      3$:     LDA A   255.,I
02B6    97      00G             STA A   DTREGA,D
02B8    B7    0000G             STA A   PIAGPA
02BB    96      00G             LDA A   DTREGB,D        ; Set EOI
02BD    8A      01              ORA A   1,I
02BF    97      00G             STA A   DTREGB,D
02C1    B7    0000G             STA A   PIAGPB
02C4    B7    0000G     SHAKEY: STA A   PIAADR          ; Finish handshake
02C7    39                      RTS
                                ;
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 12
SNDBYT--- SEND A DATA BYTE FROM THE BUFFER

                                        .GLOBL  OFFLIN, IOFUNK
                                        .GLOBL  MSACMD
                                        .GLOBL  DTREGA
                                ;
                                ; Look at secondary address to set up new command for the monitor.
                                ;
                                ;
                                ;
 02C8    96     00G     MSACMD: LDA A   OFFLIN,D        ; If offline
 02CA    2B     F8              BMI     SHAKEY          ; Then ignore sec, adr,
 02CC    CE   02F2'              LDX     LSNTBL,I        ; Assume listen address
 02CF    96     00G             LDA A   DTREGA,D        ; Get the MSA byte
 02D1    84     1F              AND A   #H1F,I          ; Strip off excess
 02D3    D6     00G             LDA B   HWMODE,D        ; Is it really talk?
 02D5    2A     03              BPL     1$
 02D7    CE   0323'              LDX     TLKTBL,I        ; Then get talk table
 02DA    E6     00      1$:     LDA B   0,X             ; Done with table?
 02DC    2B     E6              BMI     SHAKEY          ; Then illegal sec, addr, so just shake
 02DE    11                     CBA                     ; Have we found command
 02DF    27     07              BEQ     MSAOK
 02E1    08                     INX
 02E2    08                     INX
 02E3    08                     INX
 02E4    08                     INX
 02E5    08                     INX
 02E6    20     F2              BRA     1$
 02E8    A6     03      MSAOK:  LDA A   3,X             ; Get iofunc code
 02EA    97     00G             STA A   IOFUNK,D
 02EC    EE     01              LDX     1,X             ; Get command address
 02EE    DF     00G             STX     NEWCMD,D
 02F0    20     D2              BRA     SHAKEY
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 13
TLKTBL, LSNTBL --- SEC, ADR, TABLES

                                        .SBTTL  TLKTBL, LSNTBL --- SEC, ADR, TABLES
                                        .MACRO  MSA     A,, B,, C,
                                        .GLOBL  B,
                                        .BYTE   A,              ; Secondary address
                                        .WORD   B,              ; Command location
                                        .BYTE   C,              ; Iofunc
                                        .ENDM
                                ;
          000A                          .RADIX  10
                                        .GLOBL  LSNTBL
 02F2                   LSNTBL: MSA     12,PRINT,12     ; Print
 02F6                           MSA     15,PRINT,15     ; Write
 02FA                           MSA     1,PRINT,1       ; Save
 02FE                           MSA     27,FIND,0       ; Find
 0302                           MSA     28,MARK,0       ; Mark
 0306                           MSA     7,KILL,0        ; Kill
 030A                           MSA     29,SECRET,0     ; Secret
 030E                           MSA     2,CLOSE,0       ; Close
 0312                           MSA     0,STATIN,0      ; Status write
 0316                           MSA     17,PRINT,18     ; Binary save
 031A                           MSA     25,PRINT,17     ; Listen
 031E                           MSA     16,PRINT,20     ; Binary mem save
 0322     FF                    .BYTE   255             ; End of table
                                ;
                                        .GLOBL  TLKTBL
 0323                   TLKTBL: MSA     13,INPUT,13     ; Input
 0327                           MSA     14,INPUT,14     ; Read
 032B                           MSA     4,INPUT,4       ; Old
 032F                           MSA     0,STATOI,0      ; Status read
 0333                           MSA     6,TYPE,0        ; Type
 0337                           MSA     30,ERROR,0      ; Error
 033B                           MSA     9,HEADER,0      ; Header
 033F                           MSA     26,INPUT,16     ; Talk
 0343                           MSA     17,INPUT,19     ; Binary old
 0347                           MSA     24,TERR,0       ; Send magtape read errors
 034B     FF                    .BYTE   255             ; End of table
          0008                  .RADIX  8
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 14
IECRST--- RESTARTS HANDSHAKES AFTER A SUSPENSION

                                        .SBTTL  IECRST--- RESTARTS HANDSHAKES AFTER A SUSPENSION
                                        .GLOBL  IECRST
                                ;
                                ; This routine is called by the monitor to restart the handshake
                                ; After the monitor has cleaned up the reason for suspension.
                                ; This routine attempts to restart the hardware in an orderly fashion.
                                ;
                                        .GLOBL  IFCCLR
                                        .GLOBL  HWTLKS, HWLSNS, HWMAS, ATNBYT
                                        .GLOBL  ALTCMD, CLRCMD
                                        .GLOBL  UNDRES, HWSUSP, HWSUSN, HWUNAS, HWIFCS
034C    D6      00G     IECRST: LDA B   HWMODE,D        ; Check for suspended handshakes
034E    C5      00G             BIT B   HWSUSP,I        ; Check if anything suspended
0350    27      1A              BEQ     RSTXIT
0352    07                      TPA                     ; Disallow interrupts
0353    0F                      SEI
0354    36                      PSH A
0355    17                      TBA
0356    84      00G             AND A   HWSUSN,I        ; Clear suspended flags
0358    97      00G             STA A   HWMODE,D
035A    CE    036D'     RSTAGN: LDX     RSTTBL,I        ; Go find suspended function
035D    E5      00      1$:     BIT B   0,X             ; This bit?
035F    26      05              BNE     RSTSRV
0361    08                      INX
0362    08                      INX
0363    08                      INX
0364    20      F7              BRA     1$
                                ;
0366    EE      01      RSTSRV: LDX     1,X             ; Grab appropriate service routine
0368    AD      00              JSR     0,X             ; Go service it
                                ;
036A                    IECXIT:                         ; Clean up and exit
036A    32                      PUL A
036B    06                      TAP
036C    39              RSTXIT: RTS
                                ;
                                ;
                                ;
                                ;
                                        .MACRO  RT      A,,B,   ; Reset table entry
                                        .BYTE   A,
                                        .WORD   B,
                                        .ENDM
                                ;
                                ;
036D                    RSTTBL: RT      32.+2,,IFCCLR   ; Both UNListen and IFC
0370                            RT      HWMAS,ATNBYT    ; My primary address
0373                            RT      HWTLKS,SNDBYT   ; Talk suspension
0376                            RT      HWLSNS,RCVBYT   ; Listen suspension
0379    00                      .BYTE   0
        0001'                   .END
```

```
GPIB INTERFACE AN EXAMPLE       RT-11 MMAC  VM02-10   8-Oct-76 14:21:32 PAGE 14+
SYMBOL TABLE

ALTCMD= ****** G        ALTFLG= ****** G        ASCFNC= ****** G        ATNBYT   013FRG         ATNFAL   00D5R
ATNFG = ****** G        ATNFLS   00D8RG         ATNLVL=   0010          ATNTR    00ADRG         ATNTRU   00A8R
A.IN  = ****** G        A.MAX = ****** G        A.OUT = ****** G        BFREMP= ****** G        BFRFUL= ****** G
BFRSTT= ****** G        CLNUP = ****** G        CLOSE = ****** G        CLRCMD= ****** G        CMDACT= ****** G
CMDODD= ****** G        CMDVAL=   0080          CMDVRB=   0040          CMMODE= ****** G        CNREGA= ****** G
CNREGB= ****** G        DTBSAV   024ER          DTREGA= ****** G        DTREGB= ****** G        EOIPTR= ****** G
EREOF = ****** G        EREOM = ****** G        ERRCD = ****** G        ERROR = ****** G        EXSUSP   025ER
FIND  = ****** G        HANDSK   0119R          HANDXT   013CR          HEADER= ****** G        HELLO =  0004
HNDLVL=   0008          HWIFCS= ****** G        HWISRV   0000RG         HWLSNS= ****** G        HWMAS = ****** G
HWMODE= ****** G        HWSUSN= ****** G        HWSUSP= ****** G        HWTLKS= ****** G        HWUNAS= ****** G
IECPIA= ****** G        IECRST   034CRG         IECXIT   036AR          IFC      003ER          IFCCLR   006CRG
IFCDON   0075R          IFC..    007ER          IFC1     0042RG         INPUT = ****** G        IOFONK= ****** G
KILL  = ****** G        LEVEN = ****** G        LSEVEN= ****** G        LSNTBL   02F2RG         LSODD = ****** G
MARK  = ****** G        MASUSP   01CAR          MLASET   0189RG         MLASUB   0176R          MSACMD   02C6RG
MSAOK    02E8RG         MTASUB   0197R          NEWCMD= ****** G        NOGOOD   025CR          OFFLIN= ****** G
ONLIN = ****** G        PIAADR= ****** G        PIAGPA= ****** G        PIAGPB= ****** G        PIAKYB= ****** G
PIASET= ****** G        POLSTT   01EDRG         PRINT = ****** G        RCVBYT   0216RG         RDADDR   01C2R
RSTAGN   035AR          RSTSRV   0366R          RSTTBL   036DR          RSTXIT   036CR          SECRET= ****** G
SETSRQ   0097RG         SHAKER   0172R          SHAKEY   02C4R          SNDBYT   0263RG         SPE      01D1R
SPEFG = ****** G        STATIN= ****** G        STATOT= ****** G        TALK     0132R          TERR  = ****** G
TEVEN = ****** G        TKEVEN= ****** G        TKODD = ****** G        TLKTBL   0323RG         TLODD = ****** G
TYPE  = ****** G        UNDRES   007FRG
. ABS.    0000    00
          037A    01
ERRORS DETECTED: 0      Warnings Posted: 0      FREE CORE: 3093. WORDS
.note2,doc<ieee,app
```

Section 4

4051 GPIB TIMING DETAILS

INTRODUCTION

The timing events on the GPIB are discussed in detail in this section. First, ASCII data transfers are examined starting with the PRINT statement and the INPUT statement. Internal binary transfers are covered next, looking first at WRITE operations for binary output and READ operations for binary input. Finally, the lower order commands WBYTE and RTYTE are examined. These commands are used in special cases to individually transfer eight-bit binary numbers, one at a time, over the GPIB.

Timing diagrams are used extensively as visual aids in the discussions. Because each I/O transfer takes several milliseconds, a single timing diagram cannot cover the entire operation. Therefore, a complete transfer is broken into segments called "events." The details of an event are covered in one timing diagram. By mentally placing these timing diagrams side by side, the activity on the GPIB for an entire transfer can be visualized.

The timing diagrams are usually sufficient in themselves to tell an experienced electrical engineer the entire story about the bus operation. To a novice, however, the timing diagrams may be confusing and meaningless. Therefore, each diagram is followed by a step-by-step explanation of the events as they occur on the bus. The events are explained in terms of how a peripheral device responds to the 4051 and how the 4051, in turn, responds to a peripheral device.

The importance of the timing diagrams can not be over-emphasized. These diagrams act as a foundation for the discussions about every phase of the GPIB operation; from the discussions about the maximum data transfer rates, to the discussions about practical interface circuits designs. If you're not too sure how the GPIB works, how the signal lines are defined and how the handshake sequence works, we suggest you review the material in Section 1 of this manual and Section 1 of the IEEE Standard 488-1975. In addition, read the material in Appendix C of the 4051 Graphic System Reference Manual, starting on page C-13 and ending on page C-18.

TIMING DETAILS FOR THE PRINT STATEMENT

INTRODUCTION

The following text and illustrations describe the detailed timing events that occur on the GPIB during the execution of a PRINT statement. Three events occur on the bus; the peripheral addressing sequence, the data burst, and the peripheral unaddressing sequence.

Since the numeric data and the character string data in the PRINT statement are combined into one ASCII character string and transferred, no distinction is made between numerical data and string data. Therefore, the handshake timing for transferring numbers is the same as the handshake timing for transferring letters.

If the data to be transferred contains more than 72 characters, then periods of inactivity occur on the bus during the data burst. These periods are caused by the statement "set up" period, in which the 4051 prepares the next 72 characters for transfer. The set up time can range from 17.5 ms to 1080 ms, depending on the data to be transferred.

ADDRESS TIMING FOR PRINT WITH SECONDARY ADDRESS NOT SUPPRESSED.

If the secondary address 32 (which says "don't send a secondary address") is not specified in a PRINT statement, a default secondary address (12) is issued after the primary listen address during the initial addressing sequence. Fig. 4-1 illustrates the timing events that occur on the GPIB during this address sequence. The statement PRINT @2:"A" is used as an example.

The following is a step-by-step description of the events as they occur on the GPIB during the PRINT addressing sequence.

Responding to Attention

1. The GPIB is normally in an idle state prior to the execution of the PRINT statement. All signal lines are high (inactive) except NRFD and NDAC which are held low by the 4051.

2. At the beginning of the PRINT statement, the 4051 analyzes the parameter data list and prepares the first 72 characters for transfer. (Less characters are prepared, of course, if there are not 72 in the parameter list.) The GPIB stays in an idle state during this time. When the 4051 is ready, the 4051 sets ATN (Attention) active low. This tells the peripheral devices on the GPIB that addresses and controller commands are about to be issued by the 4051.

3. According to the IEEE Standard, each peripheral device on the bus must respond to ATN within 200 ns by setting NDAC low and NRFD either high or low. (IEEE Standard, page 93). The 4051, however, allows a peripheral device up to 45 μs to respond.

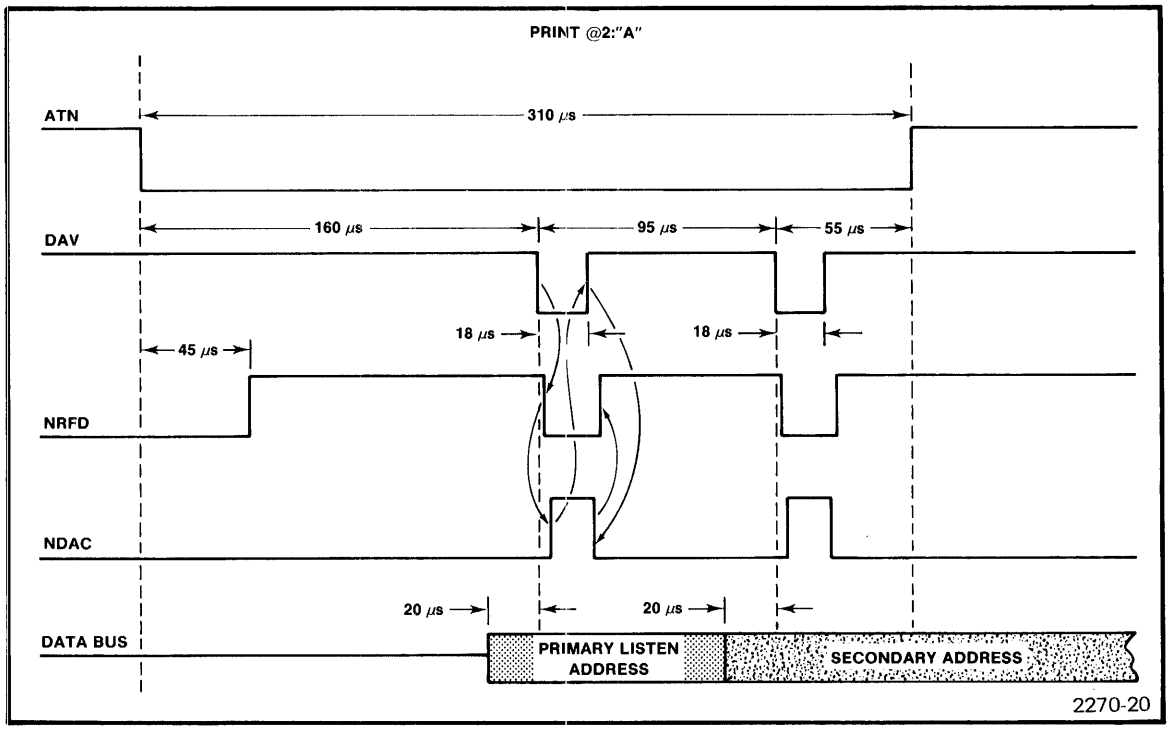


Fig. 4-1. Addressing Sequence for the PRINT Statement with the Secondary Address Not Suppressed.

4. After 45 μs, the 4051 releases NRFD and checks to see if NRFD and NDCA are not both high. If they are both high, the 4051 assumes that there is an error and the PRINT operation is aborted. The 4051 prints a GPIB interface error message on the screen (message 69). If NDAC is low and NRFD is low, the 4051 waits for NRFD to go high before continuing with the address operation. If NDAC is low and NRFD is high, the 4051 assumes that all devices are ready to receive the first peripheral address and the 4051 prepares to place the primary listen address on the Data Bus.

Transferring the Primary Listen Address

1. When all the devices are ready, the 4051 places the primary listen address for the specified device on the Data Bus. Normally this occurs 140 μs after ATN goes low. After waiting 20 μs for the bus lines to settle, the 4051 sets DAV (Data Valid) to an active (low) state. This tells each peripheral device on the GPIB that the address on the Data Bus is valid and can be captured.

2. The peripheral devices respond individually by setting NRFD low; they capture the address byte, then set NDAC high. The total time to complete this part of the handshake depends on the slowest peripheral device on the bus. If NDAC never goes high due to a peripheral interface failure, the 4051 will wait forever in a "hung" state due to the asynchronous nature of the GPIB protocal.

3. When NDAC goes high, the 4051 assumes that all devices on the bus have received the data byte; 18 μs later, the 4051 sets DAV high to tell the devices that the address is no longer valid.

4. When DAV goes high, the peripheral devices reset NDAC to a low (active) state. Each device then sets NRFD high, when the device is ready to receive the next address byte. (The NRFD signal line goes high, only after the last peripheral device sets it high.)

Transferring the Secondary Address

1. After DAV goes high (inactive) on the primary listen address, the 4051 prepares to transfer the secondary address. This preparation takes 57 μs. The 4051 then places the secondary address on the Data Bus, waits 20μs for the bus lines to settle, and checks to see if NRFD is high. If NRFD is high, the 4051 sets DAV low. If NRFD is low, the 4051 assumes that a peripheral device is still busy digesting the primary listen address, so the 4051 waits. When NRFD goes high, the 4051 sets DAV active low.

2. The handshake sequence to transfer the secondary address occurs exactly the same as the handshake sequence to transfer the primary listen address. Again, the total time to complete the handshake depends on the slowest peripheral device on the bus.

3. After DAV goes high on the secondary address, the 4051 waits 37 μs, then sets ATN high (inactive). At this time, only the device who received the primary listen address while ATN was low can listen to the data transfer. The 4051 executes the entire addressing sequence in 310 μs (minimum).

SUPPRESSING THE PRINT SECONDARY ADDRESS

If 32 is specified as the secondary address in a PRINT statement (PRINT @2,32: for example), the 4051 suppresses the secondary address and issues only the primary listen address. Although this suppression adds time to the statement overhead (approximately .5 ms), the addressing activity on the GPIB is cut by 86 μs.

Fig. 4-2 illustrates the address timing events on the GPIB when the secondary address in a PRINT statement is suppressed. The events are identical to the events just described, except that the secondary address isn't issued. The minimum time to execute this sequence is 224 μs.

HANDSHAKE TIMING DURING A PRINT DATA BURST

When the addressing sequence is over, the 4051 prepares to issue the data specified in the PRINT statment parameter list. Since only 72 characters can be prepared for transfer at a time, the data burst lasts for 72 characters maximum. The 4051 then stops and prepares the next 72 characters for transfer. If the data to be transferred contains more than 72 characters, then the data stream will be intermixed with periods of maximum activity, followed by periods of total inactivity.

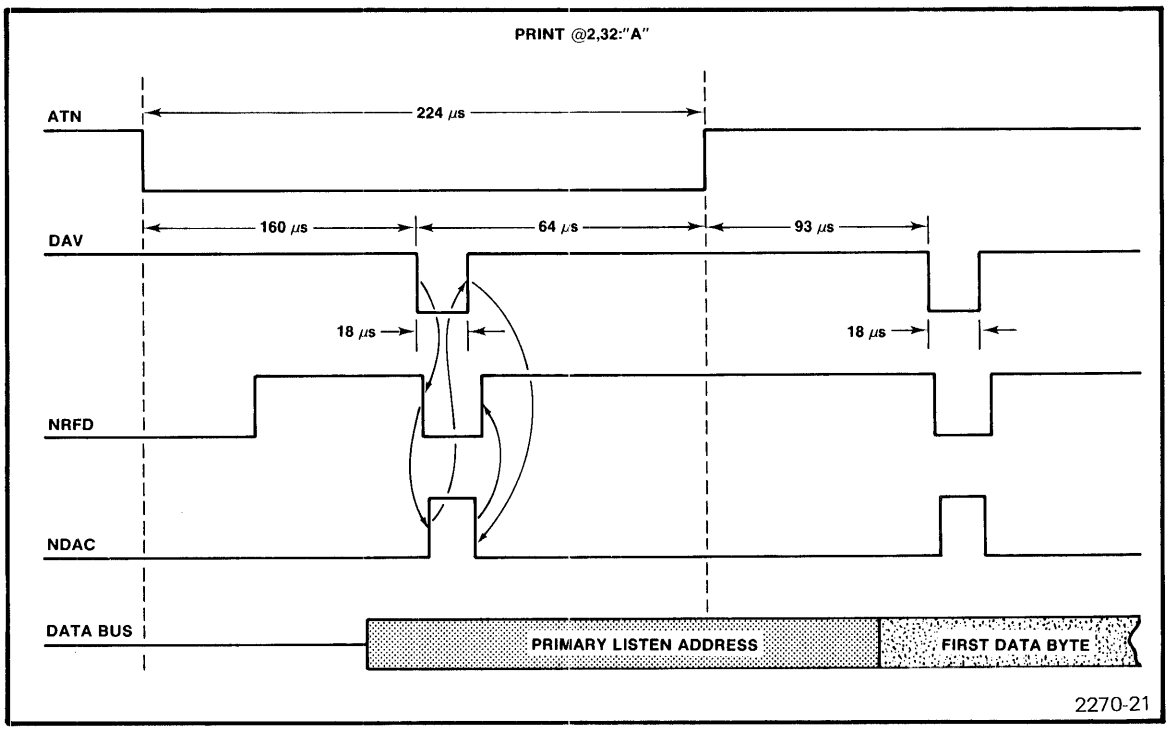


Fig. 4-2. Addressing Sequence for the PRINT Statement with the Secondary Address Suppressed.

Fig. 4-3 illustrates the activity on the GPIB when data bytes are being transferred at the maximum rate. This activity starts approximately 73 μs after ATN goes high to end the addressing sequence.

The following is a step-by-stpe description of the events as they occur on the GPIB during a PRINT data burst.

1. If the handshake sequence occurred properly when the primary listen address was issued, the 4051 assumes that the correct device received the address and is prepared to receive the ASCII data specified in the PRINT statement.

2. After ATN goes high to end the addressing sequence, the 4051 prepares the first data byte for transfer (an ASCII "A" in Fig. 4-3) and places the data byte on the Data Bus. This preparation takes 73 μs. The 4051 waits 20 μs for the bus lines to settle, then checks NRFD. If NRFD is high, the 4051 sets DAV low to start the handshake sequence. If NRFD is low it indicates that the peripheral is not yet ready to receive the byte, so the 4051 waits for NRFD to go high before setting DAV low.

Timing Details for PRINT

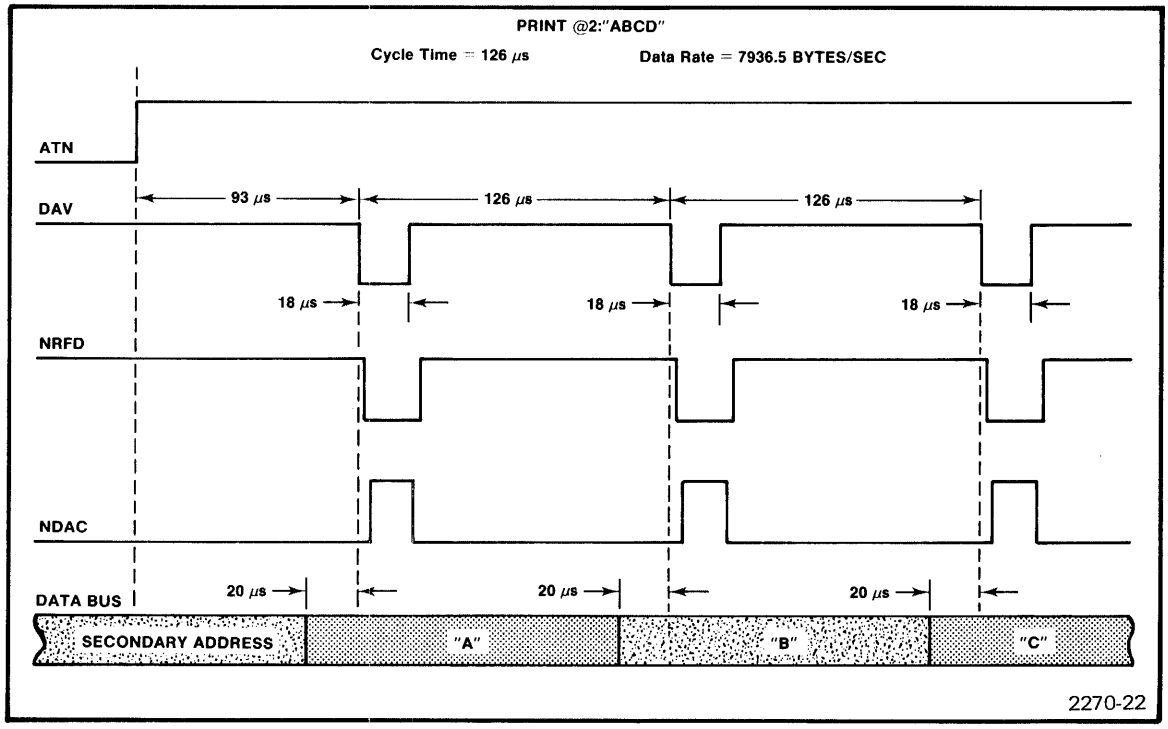


Fig. 4-3. Handshake Sequence During a PRINT Data Burst.

3. The handshake sequence occurs the same as the handshake during the addressing sequence, except that only one peripheral device is taking part. The peripheral device respondes to DAV by setting NRFD low; it captures the data byte, then sets NDAC high. The 4051 responds to NDAC by setting DAV high. This sequence takes a minimum of 18 μs to complete (plus a few nanoseconds).

4. Once DAV goes high, the 4051 prepares the next data byte for transfer. The peripheral device, in the meantime, sets NDAC low, then sets NRFD high when it is ready to receive the next data byte.

5. The 4051 places the next data byte on the Data Bus 88 μs after DAV goes high, waits 20 μs, then checks NRFD. If the peripheral device has set NRFD high by this time, the 4051 sets DAV low, and the next handshake cycle begins. If NRFD is low, the 4051 waits.

The minimum time to complete the handshake cycle is 127 μs. This provides a maximum data transfer rate of 7936.5 bytes/sec during a PRINT Data Burst.

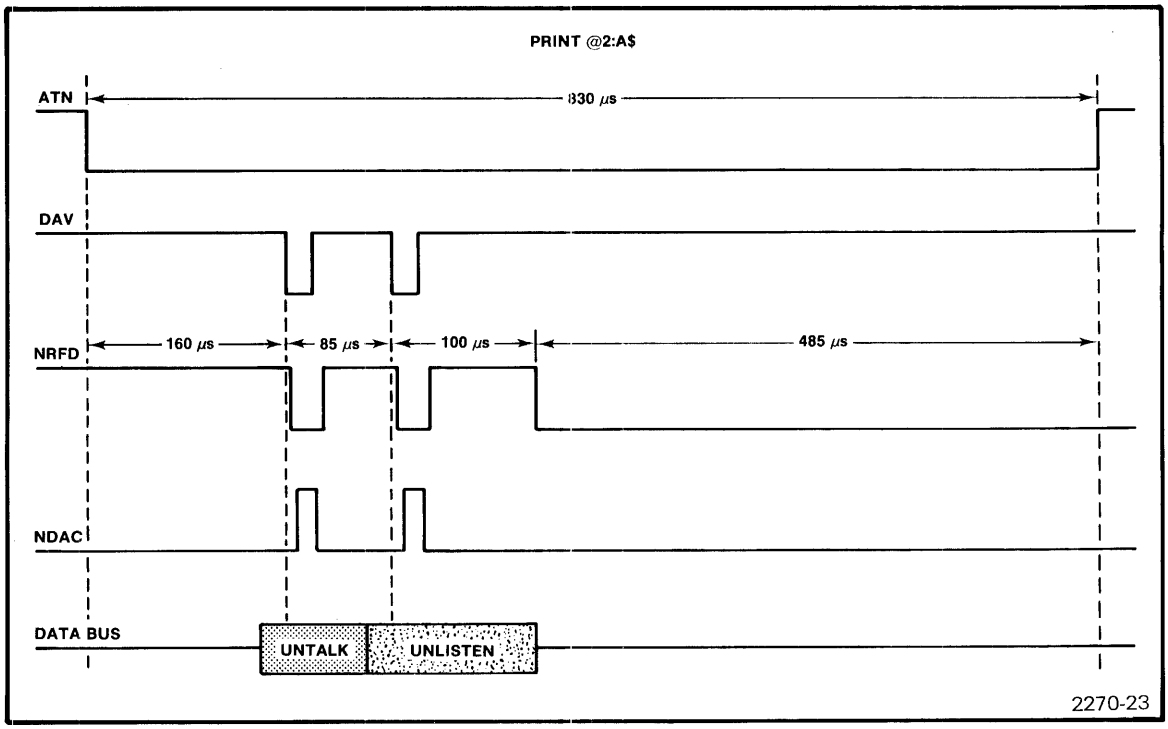


Fig. 4-4. Unaddressing Sequence for the PRINT Statement.

THE UNADDRESSING SEQUENCE FOR THE PRINT STATEMENT

After the ASCII data in the PRINT statement is transferred, the 4051 returns the GPIB to an idle state by activating ATN, issuing UNTALK and UNLISTEN, then releasing ATN. This unaddressing sequence takes 830 μs (minimum) and frees the peripheral device to go on about its business. The timing events that occur on the GPIB during the unaddressing sequence are illustrated in Fig. 4-4.

TIMING DETAILS FOR THE INPUT STATEMENT

INTRODUCTION

The following text and illustrations describe the detailed timing events that occur on the GPIB during the execution of an INPUT statement. Three events which occur on the bus: the peripheral addressing sequence, the data burst, and the peripheral unaddressing sequence.

Since numeric data and the character string data is received as one ASCII character string, no distinction is made between numeric data and string data until the data is analyzed in the I/O buffer. Therefore, the handshake timing for transferring numbers is the same as the handshake timing for transferring letters.

When the 4051 finds a delimiter (CR for example) in the data stream, or after 72 characters are input (whichever comes first), the 4051 stops to analyze and dump the contents of the buffer. During this buffer dump, the 4051 picks out the valid data and throws away the rest, then converts the data to internal binary format and stores it away in the RAM under the specified variable name.

Since the I/O buffer is dumped after every delimiter is received, the input data burst is sprinkled with periods of inactivity (the buffer overhead periods). These periods cannot be illustrated in the INPUT timing diagrams, however, they must be considered when computing the effective data rates for INPUT operations. (Refer to Section 5 for details.)

ADDRESS TIMING FOR INPUT WITH THE SECONDARY ADDRESS NOT SUPPRESSED.

If the secondary address 32 is not specified in an INPUT statement, a default secondary address (13) is issued after the primary talk address during the initial addressing sequence. Fig. 4-5 illustrates the timing events that occur on the GPIB during this address sequence. The statement INPUT @2:A$ is used as an example. A detailed step-by-step description of these events follows the figure.

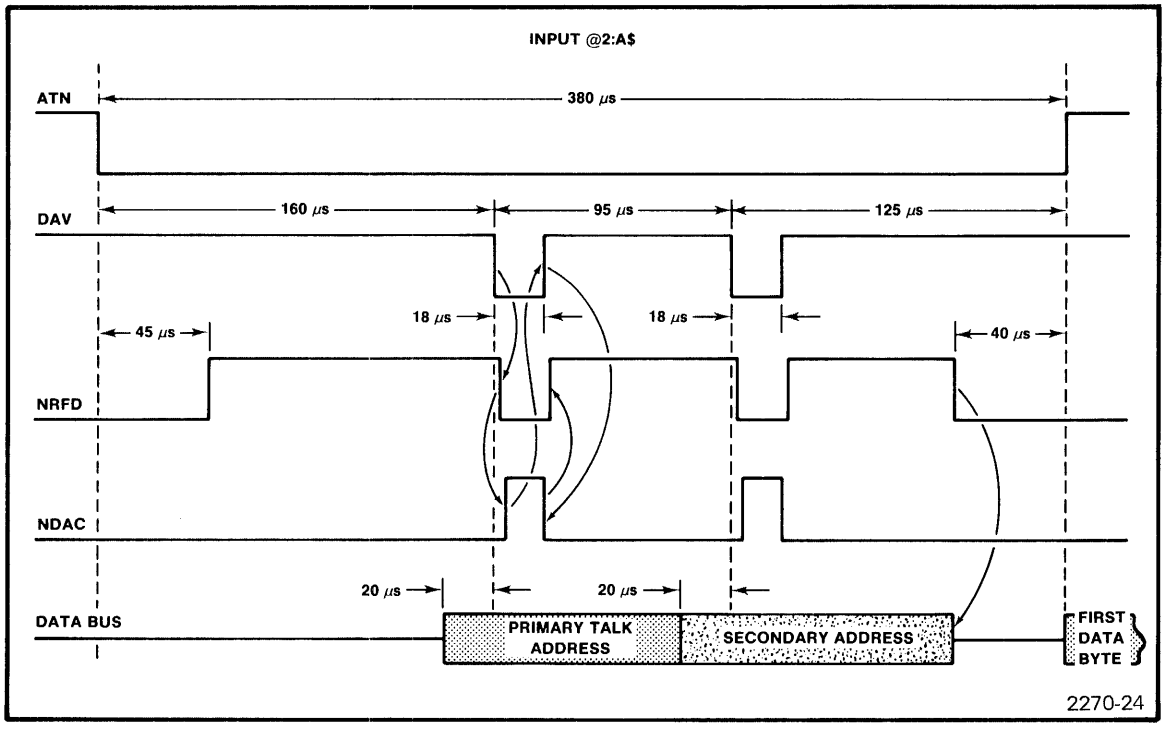


Fig. 4-5. Addressing Sequence for the INPUT Statement with the Secondary Address Not Suppressed.

Responding to Attention

1. The GPIB is normally in an idle state prior to the execution of an INPUT statement. All signal lines are high (inactive), except NRFD and NDAC which are held low by the 4051.

2. At the beginning of an INPUT statement, the 4051 analyzes the parameter variable list and prepares to input the specified data. The GPIB stays in an idle state during this time. When the 4051 is ready, the 4051 sets ATN (Attention) active low. This tells the peripheral devices on the GPIB that addresses and controller commands are about to be issued by the 4051.

3. According to the IEEE Standard, each peripheral device on the bus must respond to ATN within 200 ns by setting NDAC low and NRFD either high or low. (IEEE Standard, page 93). The 4051, however, allows a peripheral device up to 45 μs to respond.

4. After 45 μs, the 4051 releases NRFD and checks to see if NRFD and NDAC are not both high. If they are both high, the 4051 assumes that there is an error and the INPUT operation is aborted. The 4051 prints a GP Interface Error message on the screen (message no. 69). If NDAC is low and NRFD is low, the 4051 waits for NRFD to go high before continuing with the address operation. If NDAC is low and NRFD is high, the 4051 assumes that all devices are ready to receive the first peripheral address and the 4051 prepares to place the primary talk address on the Data Bus.

Transferring the Primary Talk Address

1. When all the devices are ready, the 4051 places the primary talk address for the specified device on the Data Bus. Normally this occurs 140 μs after ATN goes low. After waiting 20 μs for the bus lines to settle, the 4051 sets DAV (Data Valid) to an active (low) state. This tells each peripheral device on theGPIB that the address on theData Bus is valid and can be captured.

2. The peripheral devices respond individually by setting NRFD low; they capture the address byte, then set NDAC high. The total time to complete this part of the handshake depends on the slowest peripheral device on the bus.

3. When NDAC goes high, the 4051 assumes that all devices on the bus have received the data byte; 18 μs later, the 4051 sets DAV high to tell the devices that the address is no longer valid.

4. When DAV goes high, the peripheral devices reset NDAC to low (active) state. Each device then sets NRFD high, when the device is ready to receive the next address byte. (The NRFD signal line goes high, only after the last peripheral device sets it high.)

Transferring the Secondary Address

1. After DAV goes high (inactive) on the primary talk address, the 4051 prepares to transfer the secondary address. This preparation takes 57 μs. The 4051 then places the secondary address on the Data Bus, waits 20 μs for the bus lines to settle, and checks to see if NRFD is high. If NRFD is high, the 4051 sets DAV low. If NRFD is low, the 4051 assumes that a peripheral device is still busy digesting the primary talk address, so the 4051 waits. When NRFD goes high, the 4051 sets DAV active low.

2. The handshake sequence to transfer the secondary address occurs exactly the same as the handshake sequence to transfer the primary talk address. Again, the total time to complete the handshake depends on the slowest peripheral device on the bus.

3. Since the 4051 must listen to the GPIB during an INPUT operation, the 4051 must at this time assign itself as a listener while ATN is still down. It does this 85 μs after the secondary address is issued, The 4051 takes the secondary address off the Data Bus, assigns itself as a listener, then pulls both the NRFD and NDAC signal lines low; 40 μs later, the 4051 releases ATN and prepares to receive data bytes from the talker over the GPIB.

4. The entire address sequence takes a minimum of 380 μs to execute. Immediatly after ATN goes high, the addressed peripheral device is free to place the first data byte on the Data Bus and wait for the 4051 to set NRFD high.

INTERFACE DESIGN NOTE

It is important for the peripheral device to wait for ATN to go high before it starts transmitting data over the GPIB. If the peripheral device starts talking as soon as it receives it s talk address, it will interfere with the transmission of the secondary address.

SUPPRESSING THE INPUT SECONDARY ADDRESS

If 32 is specified as the secondary address in an INPUT statement (INPUT @2,32:A$ for example), the 4051 suppresses the secondary address and issues only the primary talk address. Although this suppression adds time to the statement overhead (approximately 1.7 ms), the addressing activity on the GPIB is cut by 86 μs.

Fig. 4-6 illustrates the address timing events on the GPIB when the secondary address in an INPUT statement is suppressed. The events are idential to the events just described, except that the secondary address isn't issued. The minimum time to execute this sequence is 224 μs.

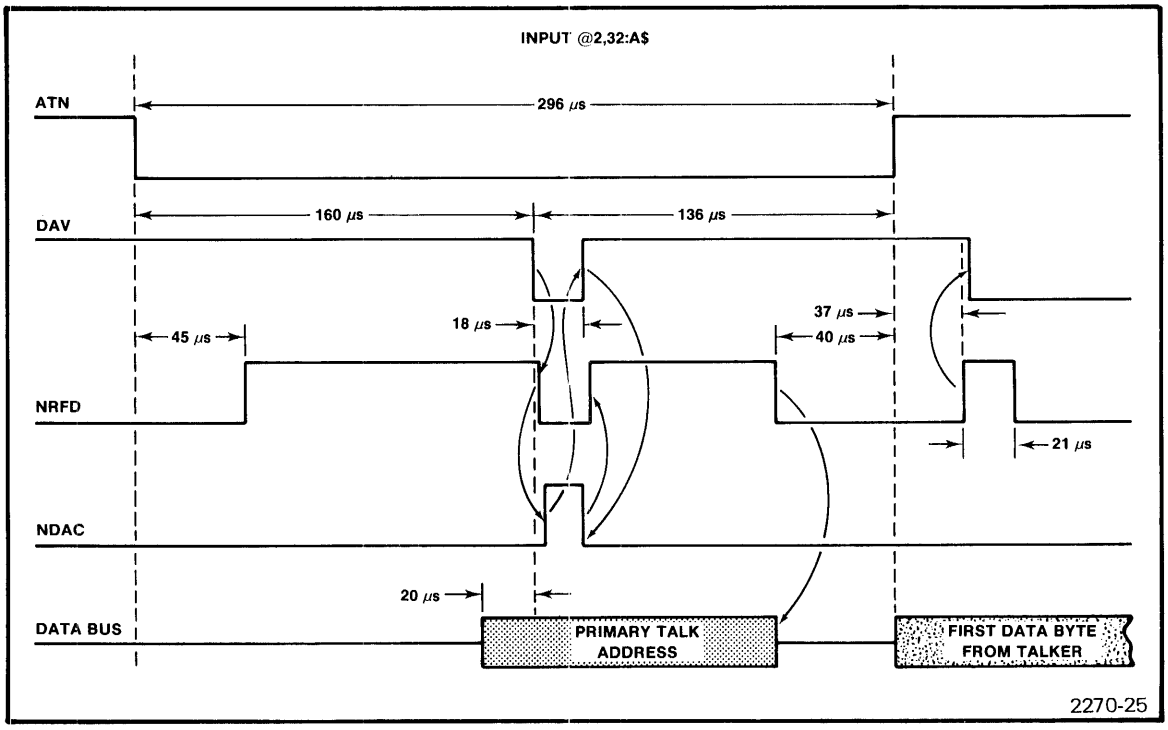


Fig. 4-6. Addressing Sequence for the INPUT Statement with the Secondary Address Suppressed.

THE INPUT DATA BURST

After the 4051 addresses a peripheral device during an INPUT operation, the 4051 assigns itself as a listener, pulls both NRFD and NDAC low, then releases ATN. The 4051 is now ready to receive data bytes from the addressed peripheral device. Fig. 4-7 illustrates the events that occur after ATN is released. The following is a step-by-step description of these events as they occur on the bus.

1. After the 4051 releases ATN, the peripheral device is free to place the first data byte on the Data Bus, but must wait until the 4051 sets NRFD high.

2. 38 μs after ATN goes high, the 4051 sets NRFD high. If the peripheral device has placed the first byte on the Data Bus by this time and waited at least 2 μs, then it is free to set DAV low.

3. When DAV goes low, the 4051 responds by setting NRFD low. This takes at least 21 μs. Next, the 4051 captures the data byte and places the byte in the I/O buffer. As soon as that is accomplished, the 4051 sets NDAC high which tells the peripheral device that the data byte has been received. It takes the 4051 a minimum of 84 μs to set NDAC high after DAV goes low.

Timing Details for INPUT

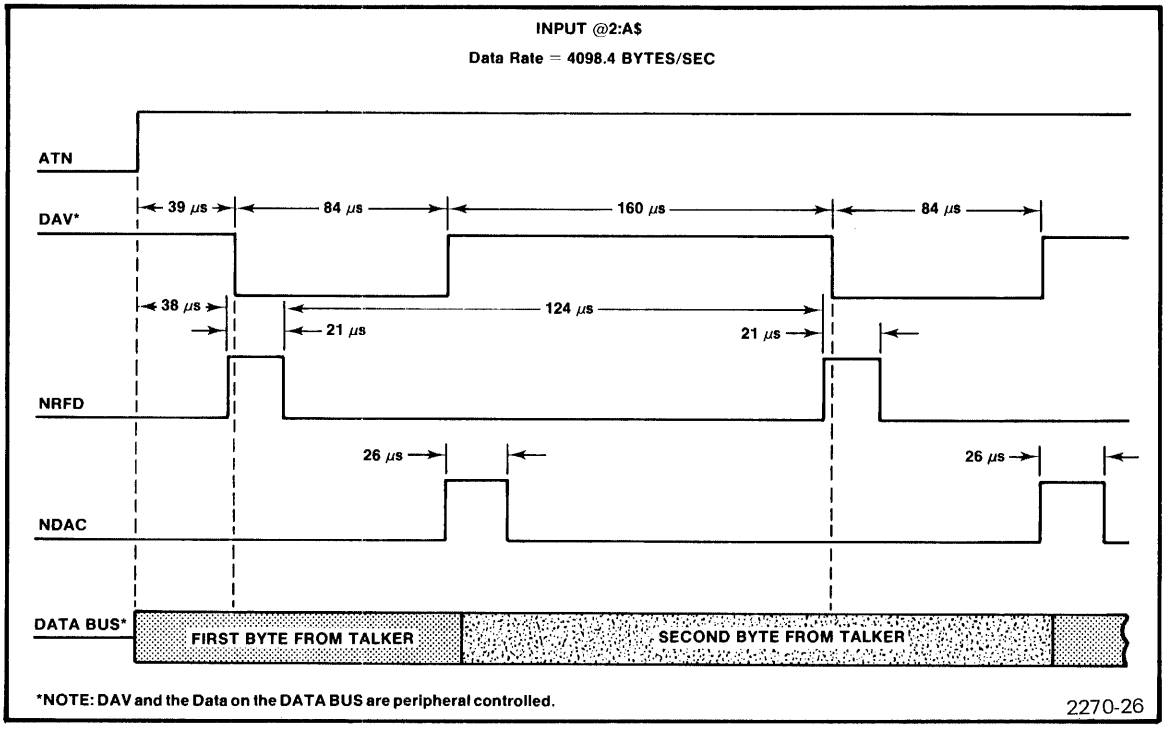


Fig. 4-7. Handshake Sequence During an INPUT Data Burst.

4. The peripheral device responds to the high NDAC signal by setting DAV high to indicate that the data byte is no longer valid. The peripheral device then prepares to place the next byte of data on the Data Bus.

5. In the meantime, the 4051 sets NDAC low, then examines the data byte to see if it is a CR delimiter. This takes a minimum of 160 µs. If an alternate delimiter is specified with a % sign in the I/O address instead of an @ sign, then the 4051 must also check to see if the data byte matches the alternate delimiter. This check takes an additional 7 µs.

6. After the 4051 is finished examining the data byte, the 4051 sets NRFD high, and prepares to receive the next data byte. Data bytes are received in this fashion until a CR delimiter is received, an alternate delimiter is received, or until 72 characters are received, whichever occurs first.

7. After one of the above occurs, the 4051 stops the data transfer by holding NRFD low, then processes the information in the I/O buffer. If a numeric variable or an array variable is specified as the target variable in the INPUT statement, the 4051 searches the contents of the I/O buffer for valid numeric data. When a numeric data item is found, the 4051 converts the data item from the ASCII format to internal floating-point format and assigns the data item to the specified variable in RAM. If a string variable is specified in the INPUT statement, then all of the characters in the I/O buffer up to the first CR (or alternate delimiter) are converted to internal ASCII format and are assigned to the string variable. If a valid delimiter is not found, the 4051 resumes the data transfer until a delimiter is found, then assigns all the characters up to the first delimiter to the string variable.

The time it takes the 4051 to process the information in the I/O buffer and get back to the GPIB and input more data is variable, depending on the contents of the I/O buffer and the kind of data the 4051 is looking for at the time. Normally, it takes 4 ms to 12 ms to dump the buffer.

It can be seen from Fig. 4-7 that the handshake cycle during the normal INPUT data burst is 244 μs. This sets the maximum data input rate at 4098.4 bytes/second during the data burst. In the "%" mode, the minimum handshake cycle is 251 μs. This sets the maximum INPUT rate to 3984.1 bytes/second during the data burst.

8. After the information in the I/O buffer is processed, the 4051 returns to the bus to receive more data from the talker. After more data is received, followed by a delimiter, or after 72 more characters are received, the 4051 stops again and processes the contents of the I/O buffer. This action repeats itself until all the target variables in the INPUT statement have assigned values.

THE UNADDRESSING SEQUENCE

After the 4051 has assigned data to each variable specified in the INPUT parameter list, the 4051 terminates the opertion by activating ATN and issuing an UNTALK/UNLISTEN sequence over the GPIB. Fig. 4-8 illustrates the events on the GPIB during the unaddressing sequence. A step-by-step description of these events follows the figure.

1. The 4051 starts the unaddressing sequence after it checks to make sure that all the specified variables have assigned data. During this check, the 4051 keeps the talker held up by holding down on NDAC.

2. When the 4051 is ready to terminate, the 4051 activates ATN as shown in Fig. 4-8.

3. The talker must respond to ATN within 45 μs (or 200 ns as per IEEE 488-1975) by setting NDAC low and NRFD either high or low; after 45 μs, the 4051 releases NRFD and NDAC and makes a check of the lines. If NRFD is low, the 4051 waits until the peripheral devices on the bus set NRFD high. If NRFD is high, the 4051 proceeds with the unaddressing sequence.

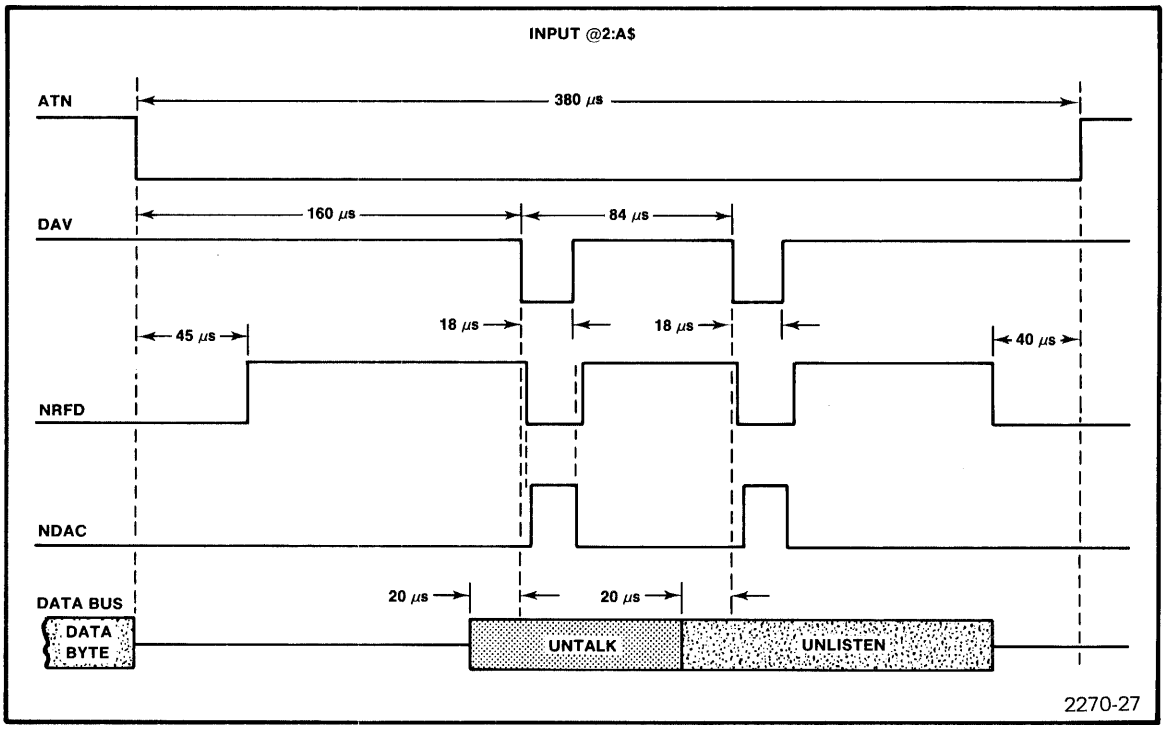


Fig. 4-8. Unaddressing Sequence for the INPUT Statement.

4. 140 μs after ATN goes down, the 4051 places the data byte for UNTALK (decimal 95) on the Data Bus, waits 20 μs, then checks the condition of NRFD. If NRFD is low, the 4051 waits until NRFD goes high. If NRFD is high, the 4051 sets DAV low and the handshake for transferring the UNTALK data byte begins.

5. The handshake cycle proceeds the same as any other handshake. Each peripheral device on the GPIB individually sets NRFD low, captures the UNTALK command byte, then sets NDAC high.

6. After NDAC goes high, the 4051 sets DAV high which indicates that the UNTALK command is no longer valid. The peripheral devices respond by setting NDAC low. After they are ready to receive the next data byte, the peripheral devices individually set NRFD high. When all are ready, the NRFD signal line goes high.

7. In the meantime, the 4051 prepares to transfer an UNLISTEN command; 65 μs after DAV goes high (invalid), the 4051 places the UNLISTEN byte on the Data Bus, waits 20 μs for the lines to settle, then checks NRFD. If NRFD is high (all are ready), the 4051 sets DAV low which tells the peripheral devices that the address on the Data Bus is valid. If NRFD is still low, indicating that some device is not yet ready for the next byte, the 4051 waits before setting DAV low. As soon an NRFD goes high, the 4051 sets DAV low, and the next handshake cycle begins.

Timing Details for INPUT

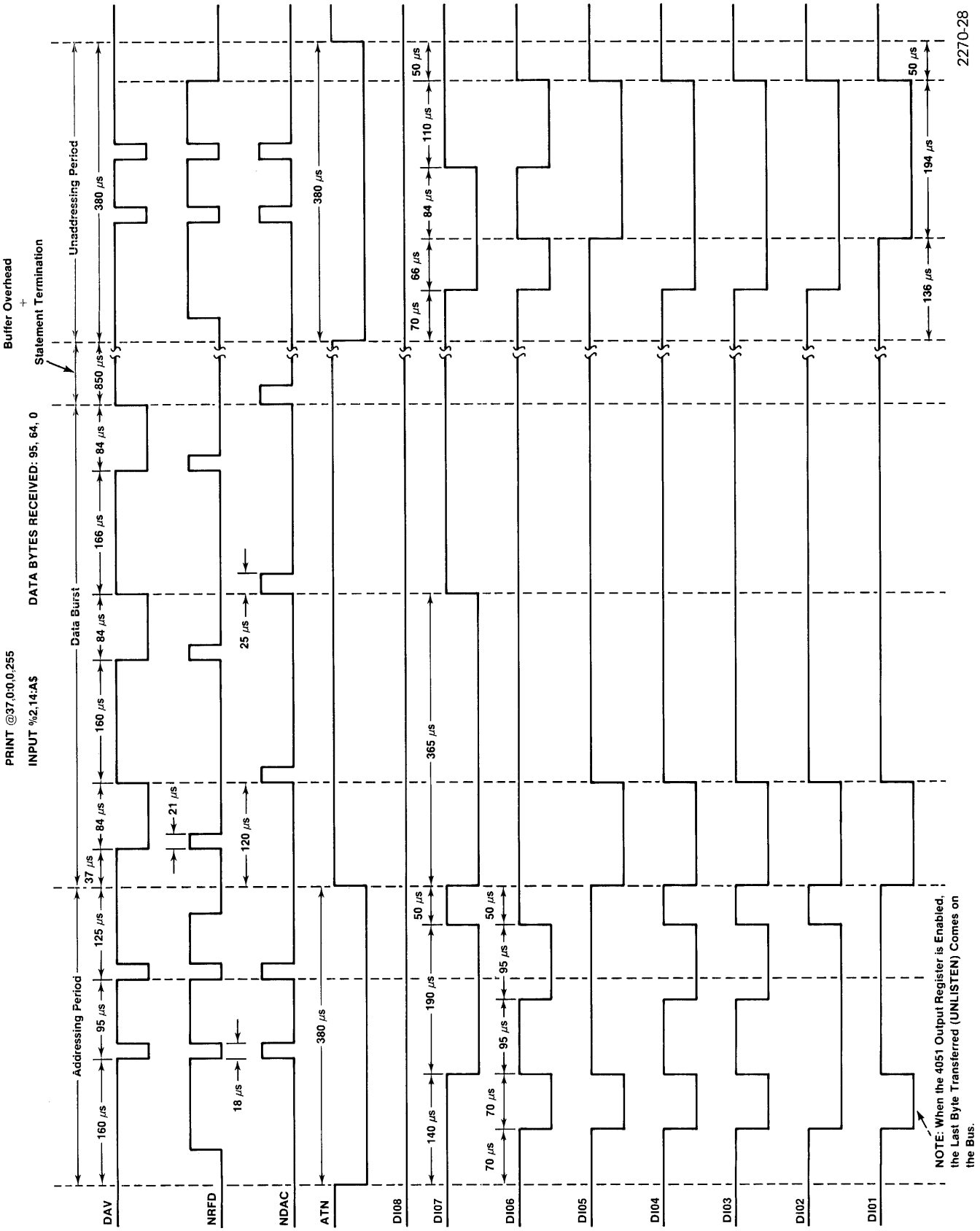


Fig. 4-9. The Complete Timing Events for an INPUT Operation.

8. The second handshake cycle proceeds the same as the first and the total time for completion depends on the slowest peripheral device on the bus.

9. 82 μs after NDAC goes high on the UNLISTEN command, the 4051 sets NRFD and NDAC low. At the same time, the 4051 takes the UNLISTEN data byte off the Data Bus; 485 μs later, the 4051 sets ATN high, and the GPIB returns again to an idle state.

A COMPLETE INPUT OPERATION

Fig. 4-9 shows the timing events for a complete INPUT operation over the 4051 GPIB. In this diagram, the alternate delimiter mode for INPUT is used to transfer a character string from peripheral device 2. The alternate delimiter is first set to decimal 0 with aa PRINT @37,0:0,79,255 statement. The statement INPUT %2:A$ is then executed to start the transfer. The talker (device 2) sends decimal 94 (an ASCII _ character) as the first data byte, decimal 64 (an ASCII @ character) as the second data byte, and decimal 0 (an ASCII NULL) as the delimiter. The first two characters are assigned to A$, then the INPUT operation is terminated.

TIMING DETAILS FOR THE WRITE STATEMENT

INTRODUCTION

The following text and illustrations describe the detailed timing events that occur on the GPIB during the execution of a WRITE statement. Three events occur on the bus: the peripheral addressing sequence, the data burst, and the peripheral unaddressing sequence.

Since the data transferred in a WRITE statement is formatted in 4051 internal binary code, the number of data bytes transferred during the data burst is predictable. Each numeric value in the data stream consists of a two-byte header plus eight bytes of internal floating-point notation. Each character string contains a two-byte header plus one byte for each character in the string (i.e. LEN A$). The maximum length of any one string is 8192 bytes plus the header.

In addition to the predictable length of each data item, the gaps in the data burst are considerable reduced when compared with the PRINT statement. Since the conversion from internal binary format to ASCII code format is not necessary, the data is taken directly from the internal RAM memory and is placed on the GPIB Data Bus. The statement setup period is virtually eliminated, so the total time to complete the data transfer is considerably less than a PRINT statement data transfer.

ADDRESS TIMING FOR WRITE WITH THE SECONDARY ADDRESS NOT SUPPRESSED.

If the secondary address 32 is not specified in a WRITE statement, a default secondary address (15) is issued after the primary listen address during the addressing sequence. Fig. 4-10 illustrates the timing events that occur on the GPIB during this address sequence. The statement WRITE @2:"A" is used as an example.

The following is a step-by-step description of the events as they occur on the GPIB during the WRITE addressing sequence.

Responding to Attention

1. The GPIB is normally in an idle state prior to the execution of the WRITE statement. All signal lines are high (inactive) except NRFD and NDAC which are held low by the 4051.

2. At the beginning of the WRITE statement, the 4051 analyzes the parameter data list and prepares to transmit the data. The GPIB stays in an idle state during this time. When the 4051 is ready, the 4051 sets ATN (Attention) active low. This tells the peripheral devices on the bus that addresses and controller commands are about to be issued by the 4051.

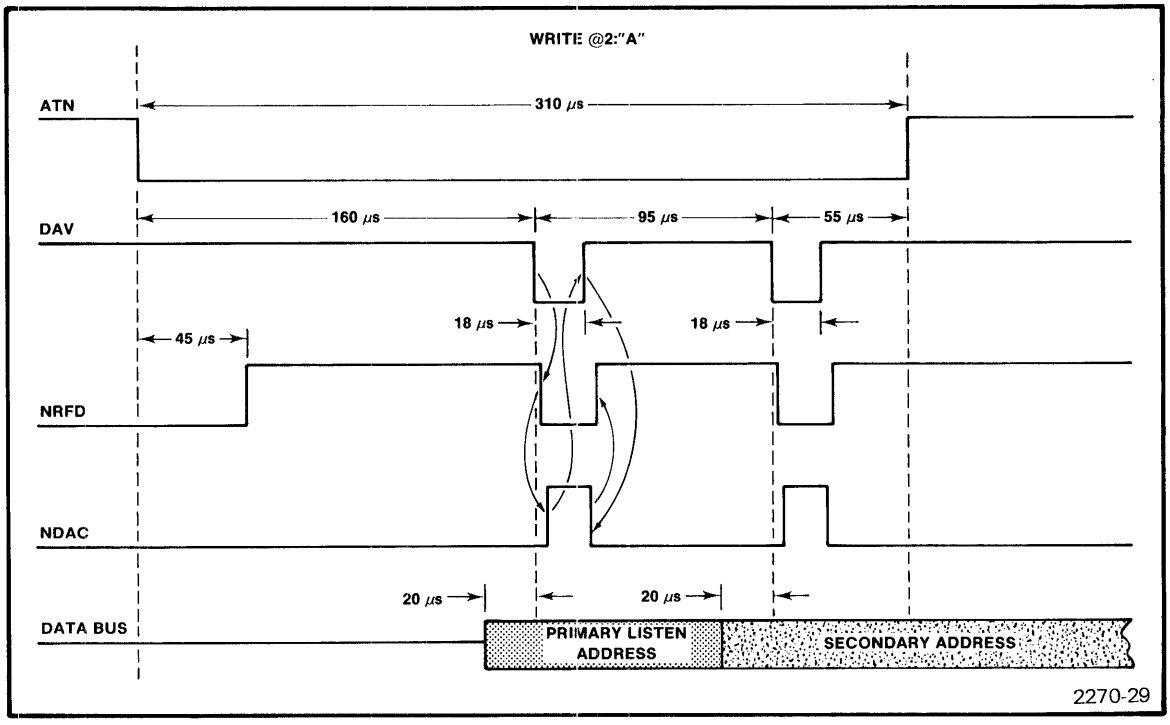


Fig. 4-10. Addressing Sequence for the WRITE Statement with the Secondary Address Not Suppressed.

3. According to the IEEE Standard, each peripheral device on the bus must respond to ATN within 200 ns by setting NDAC low and NRFD either high or low. (IEEE Standard, page 93). The 4051, however, allows a peripheral device up to 45 μs to respond.

4. After 45 μs, the 4051 releases NRFD and checks to see if NRFD and NDAC are not both high. If they are both high, the 4051 assumes that there is an error and the WRITE operation is aborted. The 4051 prints a GPIB Interface Error Message on the screen (message 69). If NDAC is low and NRFD is low, the 4051 waits for NRFD to go high before continuing with the address operation. If NDAC is low and NRFD is high, the 4051 assumes that all devices are ready to receive the first peripheral address and the 4051 prepares to place the primary listen address on the Data Bus.

Transferring the Primary Listen Address

1. When all the devices are ready, the 4051 places the primary listen address for the specified device on the Data Bus. Normally this occurs 140 μs after ATN goes low. After waiting 20 μs for the bus lines to settle, the 4051 sets DAV (Data Valid) to an active (low) state. This tells each peripheral device on the GPIB that the address on the Data Bus is valid and can be captured.

2. The peripheral devices respond individually by setting NRFD low; they capture the address byte, then set NDAC high. The total time to complete this part of the handshake is determined by the slowest peripheral device on the bus.

3. When NDAC goes high, the 4051 assumes that all devices on the bus have received the data byte; 18 μs later, the 4051 sets DAV high to tell the devices that the address is no longer valid.

4. When DAV goes high, the peripheral devices reset NDAC to a low (active) state. Each device then sets NRFD high, when the device is ready to receive the next address byte. (The NRFD signal goes high, only after the last peripheral device sets it high.)

Transferring the Secondary Address

1. After DAV goes high (inactive), the 4051 prepares to transfer the secondary address. This preparation takes 57 μs. The 4051 then places the secondary address on the Data Bus, waits 20 μs for the bus lines to settle, and checks to see if NRFD is high. If NRFD is high, the 4051 sets DAV low. If NRFD is low, the 4051 assumes that a peripheral device is still busy digesting the primary listen address, so the 4051 waits. When NRFD goes high, the 4051 sets DAV active low.

2. The handshake sequence to transfer the secondary address occurs exactly the same as the handshake sequence to transfer the primary listen address. Again, the total time to complete the handshake depends on the slowest peripheral device on the bus.

3. After DAV goes high on the secondary address, the 4051 waits 37 μs, then sets ATN high (inactive). At this time, only the device that received the primary listen address while ATN was low can listen to the data transfer. The 4051 executes this addressing sequence in 310 μs (minimum).

SUPPRESSING THE WRITE SECONDARY ADDRESS

If 32 is specified as the secondary address in a WRITE statement (WRITE @2,32:"A" for example), the 4051 suppresses the secondary address and issues only the primary listen address. Although this suppression adds time to the statement overhead (approximately .5 ms), the addressing activity on theGPIB is cut by 86 μs to 224 μs.

Fig. 4-11 illustrates the address timing events on the GPIB when the secondary address in a WRITE statement is suppressed. The events are identical to the events just described, except that the secondary address isn't issued. The minimum time to execute this sequence is 224 μs.

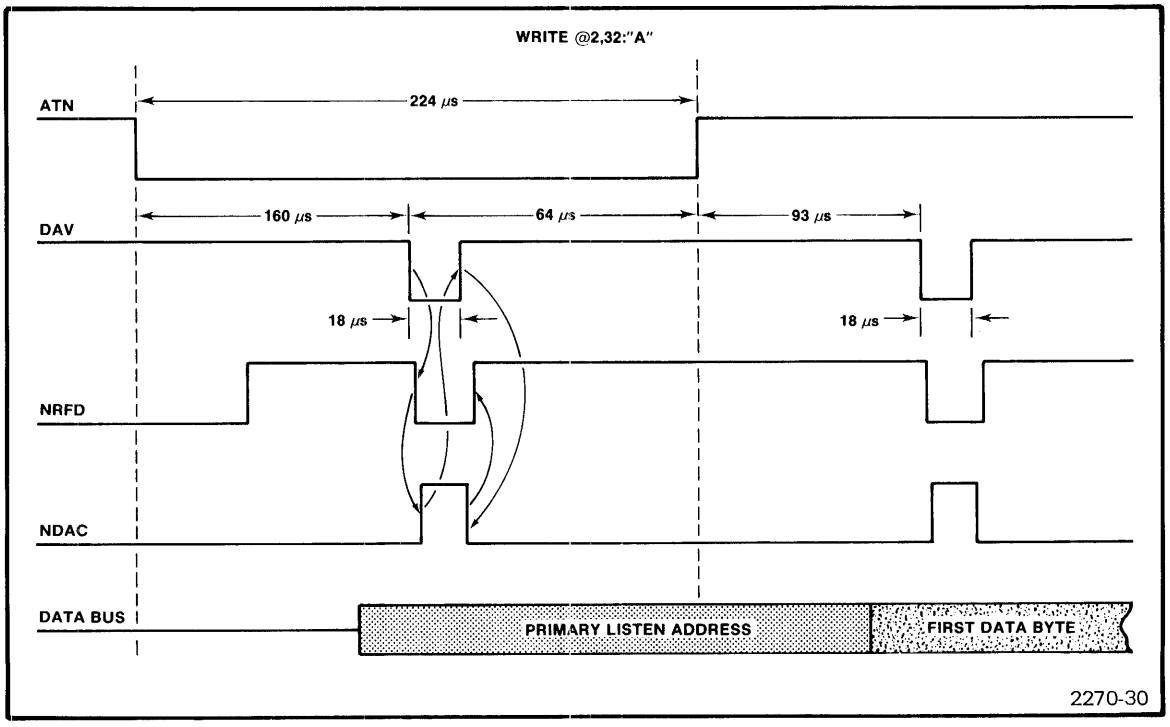


Fig. 4-11. Addressing Sequence for the WRITE Statement with the Secondary Address Suppressed.

THE WRITE DATA BURST

Starting the Data Transfer

If the handshake sequence occurred normally while the primary listen address was on the bus, the 4051 assumes that the correct peripheral ddevice received the listen address and is prepared to receive the binary data. The 4051, therefore, prepares the header for the first data item in the parameter list and places the first byte of the header on the Data Bus. This takes 73 μs after ATN goes high, as shown in Fig. 4-11.

Transferring Numeric Data

Each numeric data item specified in a WRITE statement is transferred as an eight byte floating-point number preceded by a two-byte header. (This format is described in Appendix A.) If a numeric array is specified, each array element is transferred as an eight-byte floating-point number with a two-byte header, one after another in row major order. Fig. 4-12 illustrates the timing events that occur on the GPIB when a number is transferred in floating-point using the WRITE statement.

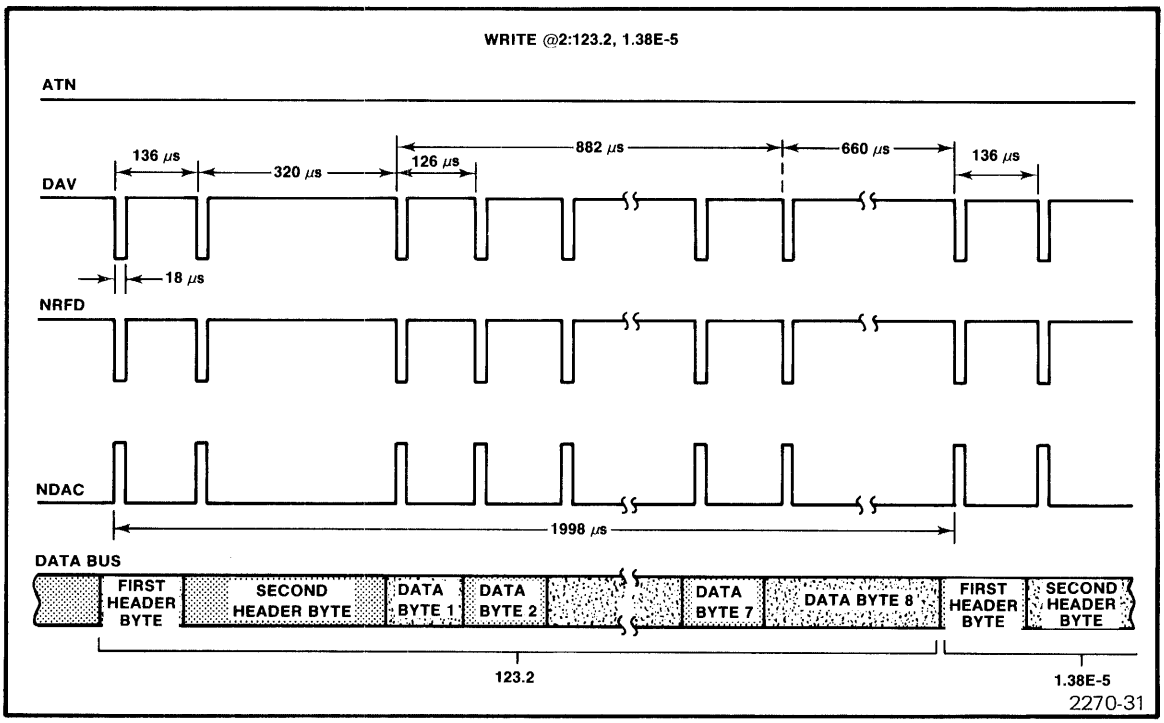


Fig. 4-12. Handshake Sequence for a WRITE Numeric Data Burst.

The following is a step-by-step description of the events as they occur on the GPIB as the numeric data item is transferred.

1. The transfer begins when the 4051 places the first byte of the header (decimal 32) on the Data Bus. The 4051 waits 20 μs for the lines to settle, then checks NRFD. If NRFD is high, the 4051 sets DAV low and the handshake with the peripheral device begins.

2. The handshake to transferring data byte during a WRITE operation is the same as the handshake for any other statement. After DAV goes low, the peripheral device sets NRFD low, captures the data byte, then sets NDAC high. The 4051 responds to NDAC by setting DAV high. The minimum time to accomplish this action is 18 μs for each data byte. The rest of this discussion concentrates on the time lapse between each handshake.

3. After the first byte of the header is transferred to the peripheral device, the 4051 prepares the second byte of the header for transfer. This takes 136 μs as shown in Fig. 4-12.

4. After the second byte of the header is transferred, the 4051 prepares to transfer the eight bytes of the floating-point number. This preparation takes 320 μs.

5. The data burst lasts for eight handshake cycles and the minimum time to complete one handshake cycle is 126 μs. The first seven cycles are completed in 882 μs. The last cycle is intermixed with the set up time for the next numeric data item. The total (last handhake cycle plus set up time for the next data item) is 660 μs.

6. Adding all the time increments up, the total time to transfer one numeric value in 4051 internal floating-point notation is 1998 μs. This means that in a continuous burst, the effect rate for transferring data samples (or data points) is 500.5 data samples/sec.

Transferring Character String Data

Each character string specified in a WRITE statement is transferred as a two-byte header followed by a stream of data bytes, one for each character in the string. The header identifies the data as a character string and also specifies the length (in bytes).

Fig. 4-13 illustrates the timing events that occur on the GPIB when a character string is transferred using the WRITE statement. The timing is similar to the time for transferring numeric data, except that the delay after the second header byte is 300 μs instead of 320 μs. The handshake cycle during the data burst is variable, depending on the length of the character string. The time lag at the end of the data burst is also different; 800 μs as opposed to 660 μs for a numeric data item.

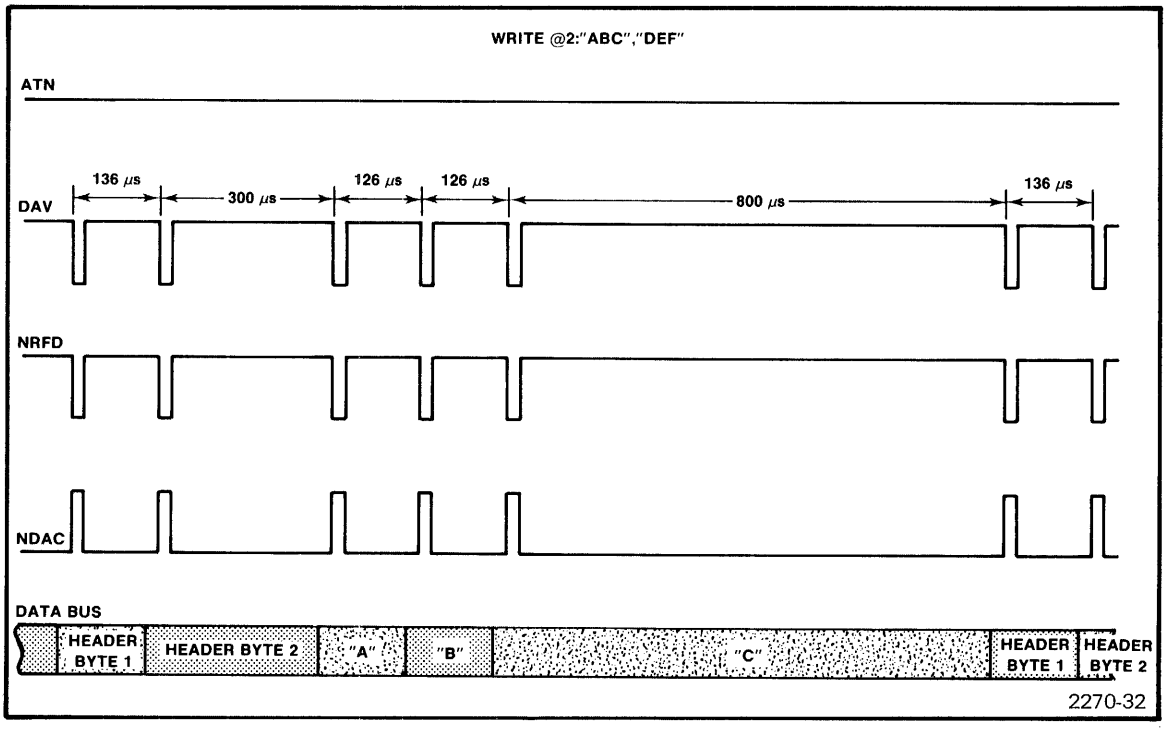


Fig. 4-13. Handshake Sequence for a WRITE String Data Burst.

Computing the Data Sample Rate for Character Strings

Since it takes 436 μs to transfer the header and 126 μs to transfer each byte in the character string, followed by an 800 μs delay, the data sample rate is computed as follows:

Sample Rate=1/(436 + ((characters in string-1)x126) + 800)E-6

So, for example, a series of character strings with 8 characters each can be transferred at a rate of:

1/(436 + ((8-1)x126) + 800)E-6 samples/sec

which is equal to:

1/2118E-6 samples/sec

which is equal to:

472.2 samples/sec

THE UNADDRESSING SEQUENCE FOR THE WRITE STATEMENT

After the last data byte is transferred in a WRITE operation, the 4051 clears the GPIB by activating ATN, issuing UNTALK and UNLISTEN, then releases ATN. This returns the GPIB to an idle state and frees the peripheral device to process the information just received. Fig. 4-14 illustrates the timing events which occur on the GPIB when the unaddressing sequence is executed at the end of a WRITE statement.

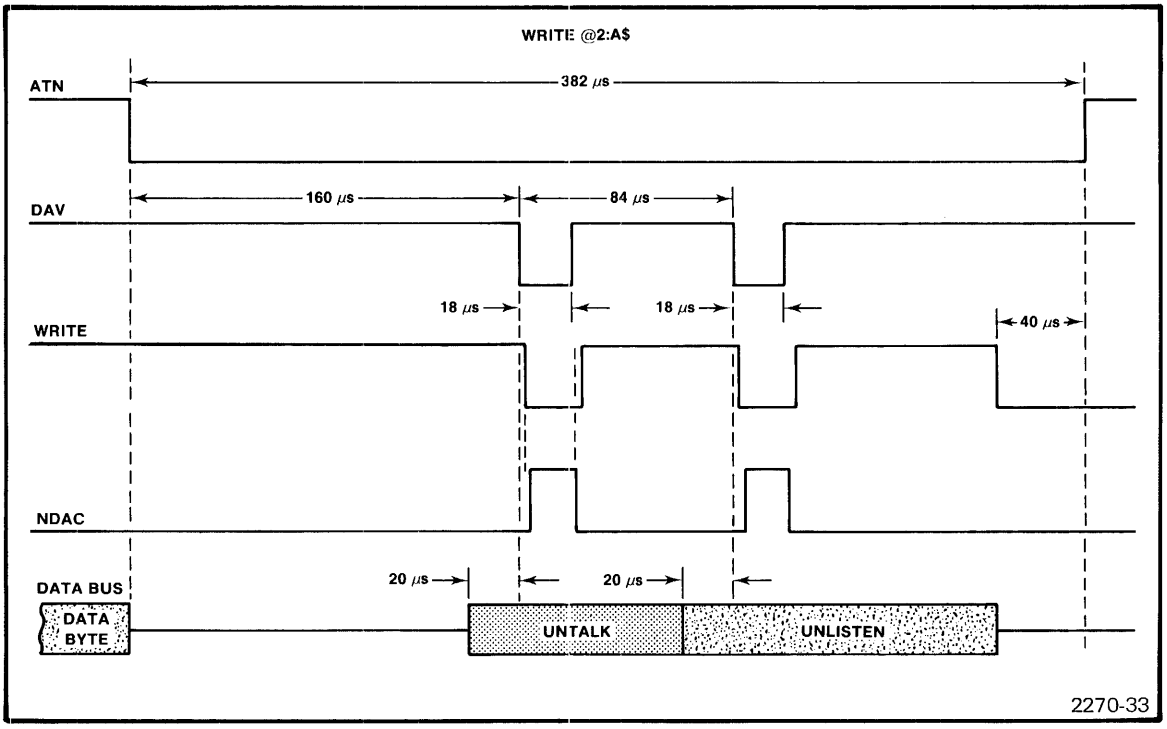


Fig. 4-14. Unaddressing Sequence for the WRITE Statement.

TIMING DETAILS FOR THE READ STATEMENT

INTRODUCTION

The following text and illustrations describe the detailed timing events that occur on the GPIB during the execution of a READ statement. Three events which occur on the bus; the peripheral addressing sequence, the data burst, and the peripheral unaddressing sequence.

Since the data transferred in a READ statement must be formatted in 4051 internal binary code, the number of data bytes transferred during the data burst is predictable. Each numeric value in the data stream consists of two-byte header plus eight bytes of internal floating-point notation. Each character string contains a two-byte header plus one byte for each character in the string. The maximum length of any one string is 8191 bytes plus the header.

In addition to the predictable length of each data item, the gaps in the data burst are considerably reduced when compared with the INPUT statement. Since the conversion from ASCII code format to internal binary format is not necessary, the data is taken directly from the I/O buffer and placed in the internal RAM memory. The buffer overhead is virtually eliminated, so the total time to complete a READ data transfer is considerably less than the time it takes to complete an INPUT data transfer.

ADDRESS TIMING FOR READ WITH THE SECONDARY ADDRESS NOT SUPPRESSED.

If the secondary address 32 is not specified in a READ statement, a default secondary address (14) is issued after the primary talk address during the addressing sequence. Fig. 4-15 illustrates the timing events that occur on the GPIB during this address sequence. The statement READ @2:A$ is used as an example. A step-by-step description of the events of GPIB follow the figure.

Responding to Attention

1. The GPIB is normally in an idle state prior to the execution of the READ statement. All signal lines are high (inactive), except NRFD and NDAC which are held low by the 4051.

2. At the beginning of the READ statement, the 4051 analyzes the parameter data list and prepares to assign data to the variables. The GPIB stays in an idle state during this time. When the 4051 is ready, the 4051 sets ATN (Attention) active low. This tells the peripheral devices on the bus that addresses and controller commands are about to be issued by the 4051.

3. According to the IEEE standard, each peripheral device on the bus must respond to ATN within 200 ns by setting NDAC low and NRFD either high or low. (IEEE Standard, page 93). The 4051, however, allows a peripheral device up to 45 μs to respond.

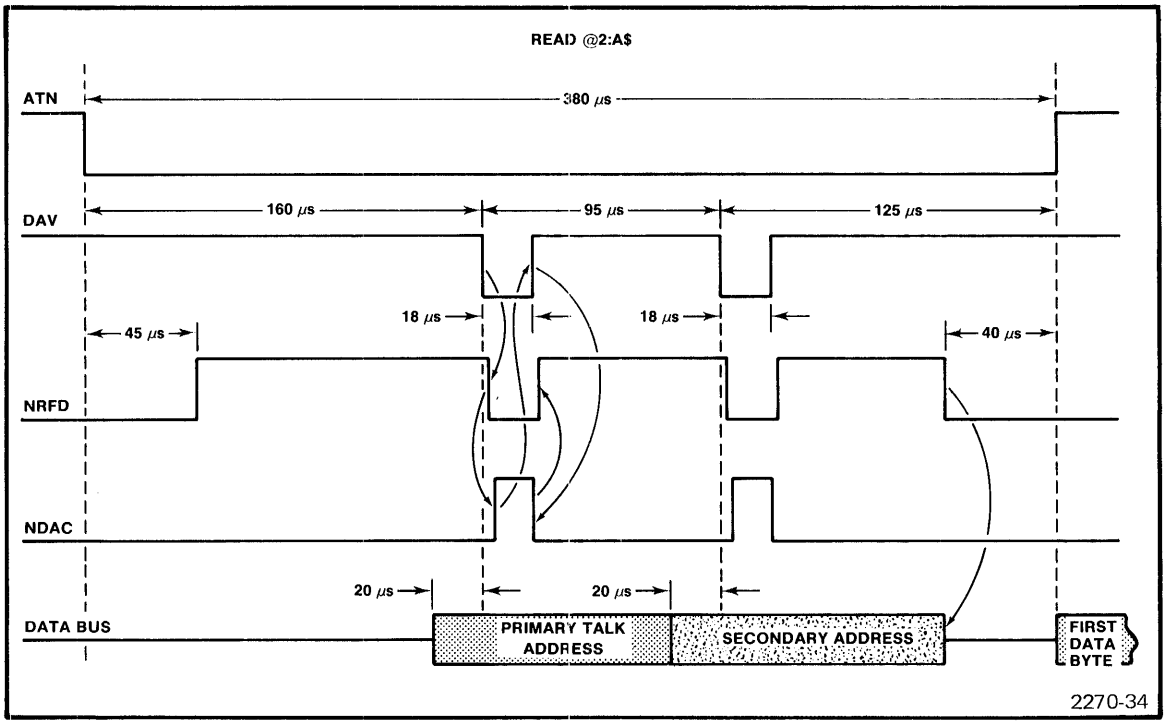


Fig. 4-15. Addressing Sequence for the READ Statement with the Secondary Address Not Suppressed.

4. After 45 μs, the 4051 releases NRFD and checks to see if NRFD and NDAC are not both high. If they are both high, the 4051 assumes that there is an error and the READ operation is aborted. The 4051 prints a GP Interface Error Message on the screen (message no 69). If NDAC is low and NRFD is low, the 4051 waits for NRFD to go high before continuing with the address operation. If NDAC is low and NRFD is high, the 4051 assumes that all devices are ready to receive the first peripheral address and the 4051 prepares to place the primary talk address on the Data Bus.

Transferring the Primary Talk Address

1. When all the devices are ready, the 4051 places the primary talk address for the specified device on the Data Bus. Normally this occurs 140 μs after ATN goes low. After waiting 20 μs for the bus lines to settle, the 4051 sets DAV (Data Valid) to an active (low) state. This tells each peripheral device on the GPIB that the address on the Data Bus is valid and can be captured.

2. The peripheral devices respond individually by setting NRFD low; they capture the address byte, then set NDAC high. The total time to complete this portion of the handshake is determined by the slowest peripheral device on the bus.

3. When NDAC goes high, the 4051 assumes that all devices on the bus have received the data byte; 18 μs later, the 4051 sets DAV high to tell the devices that the address is no longer valid.

4. When DAV goes high, the peripheral devices reset NDAC to a low (active) state. Each device then sets NRFD high, when the device is ready to receive the next address byte. (The NRFD signal goes high, only after the last peripheral device sets it high.)

Transferring the Secondary Address

1. After DAV goes high (inactive), the 4051 prepares to transfer the secondary address. This preparation takes 57 μs. The 4051 then places the secondary address on the Data Bus, waits 20 μs for the bus lines to settle, and checks to see if NRFD is high. If NRFD is high, the 4051 sets DAV low. If NRFD is low the 4051 assumes that a peripheral device is still busy digesting the primary talk address, so the 4051 waits. When NRFD goes high, the 4051 sets DAV active low.

2. The handshake sequence to transfer the secondary address occurs exactly the same as the handshake sequence to transfer the primary talk address. Again, the total time to complete the handshake depends on the slowest peripheral device on the bus.

3. Since the 4051 must listen to the GPIB during a READ operation, the 4051 must at this time assign itself as a listener while ATN is still down. It does this 85 μs after the secondary address is issued. The 4051 takes the secondary address off the Data Bus, assigns itself as a listener, then sets both the NRFD and NDAC signal lines low; 40 μs later, the 4051 releases ATN and prepares to receive data bytes from the talker over the GPIB.

4. The entire address sequence takes a mimimum of 380 μs to execute. Immediately after ATN goes high, the addressed peripheral device is free to place the first data byte on the Data Bus and wait for the 4051 to set NRFD high.

INTERFACE DESIGN NOTE

It is important for the peripheral device to wait for ATN to go high before it starts transmitting data over the GPIB. If the peripheral device starts talking as soon as it receives it s talk address, it will interfere with the transmission of the secondary address.

SUPPRESSING THE READ SECONDARY ADDRESS

If 32 is specified as the secondary address in a READ statement (READ @2,32:"A" for example), the 4051 suppresses the secondary address and issues only the primary talk address. Although this suppression adds time to the statement overhead (approxiamately 1.7 ms), the addressing activity on the GPIB is cut by 86 μs to 296 μs.

Fig. 4-16 illustrates the address timing events on the GPIB when the secondary address in a READ statement is suppressed. The events are identical to the events just described, except that the secondary address isn't issued. The minimum time to execute this sequence is 296 μs.

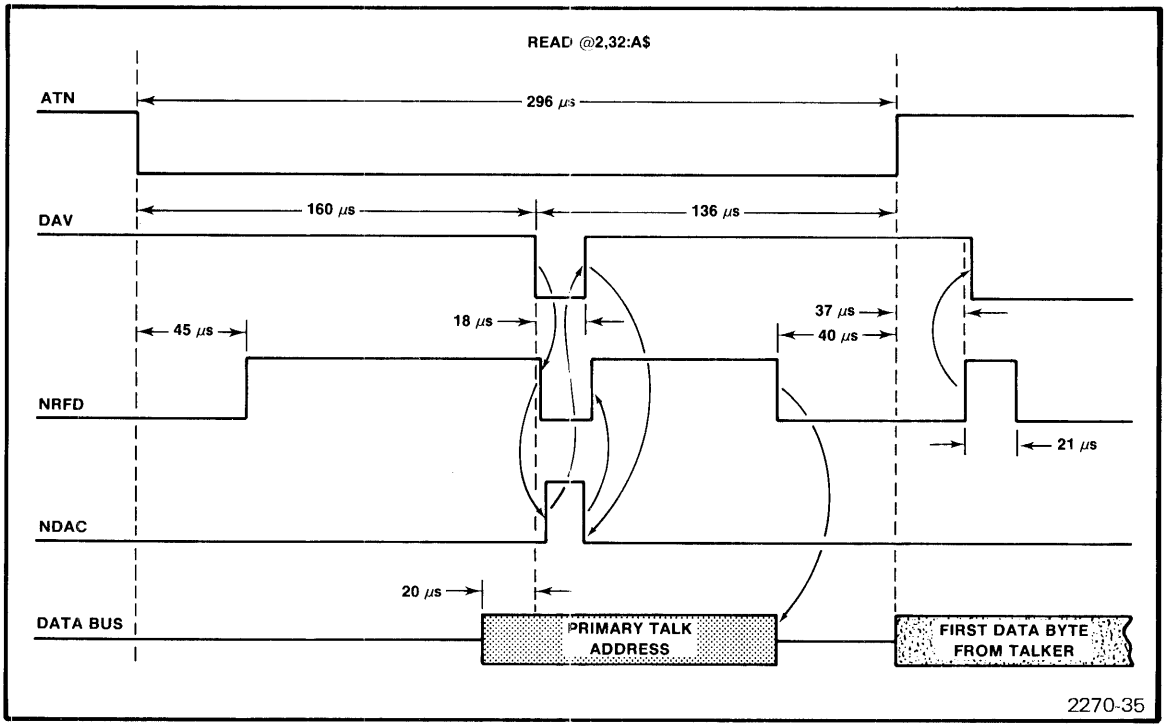


Fig. 4-16. Addressing Sequence for the READ Statement with the Secondary Address Suppressed.

THE READ DATA BURST

Starting the Data Transfer

If the handshake sequence occurred normally while the primary talk address was on the bus, the 4051 **assumes** that the correct peripheral device received the talk address and is prepared to transmit 4051 internal binary data. The 4051, therefore, prepares to receive the first data item. This preparation takes 37 μs after ATN goes high, as shown in Fig. 4-16.

Transferring Numeric Data

If a numeric variable is specified in the READ parameter list, then the talker must transfer a numeric data item formatted in 4051 internal floating-point notation. Each numeric data item is transferred as an eight-byte floating-point number preceded by a two-byte header. (This format is described in Appendix A.) If a numeric array is specified, each array element is transferred as an eight-byte floating-point number with a two-byte header, one after another in row major order. Fig. 4-17 illustrates the timing events which occur on the GPIB when a number is transferred in floating-point using the READ statement. A step-by-step description of the events follows the figure.

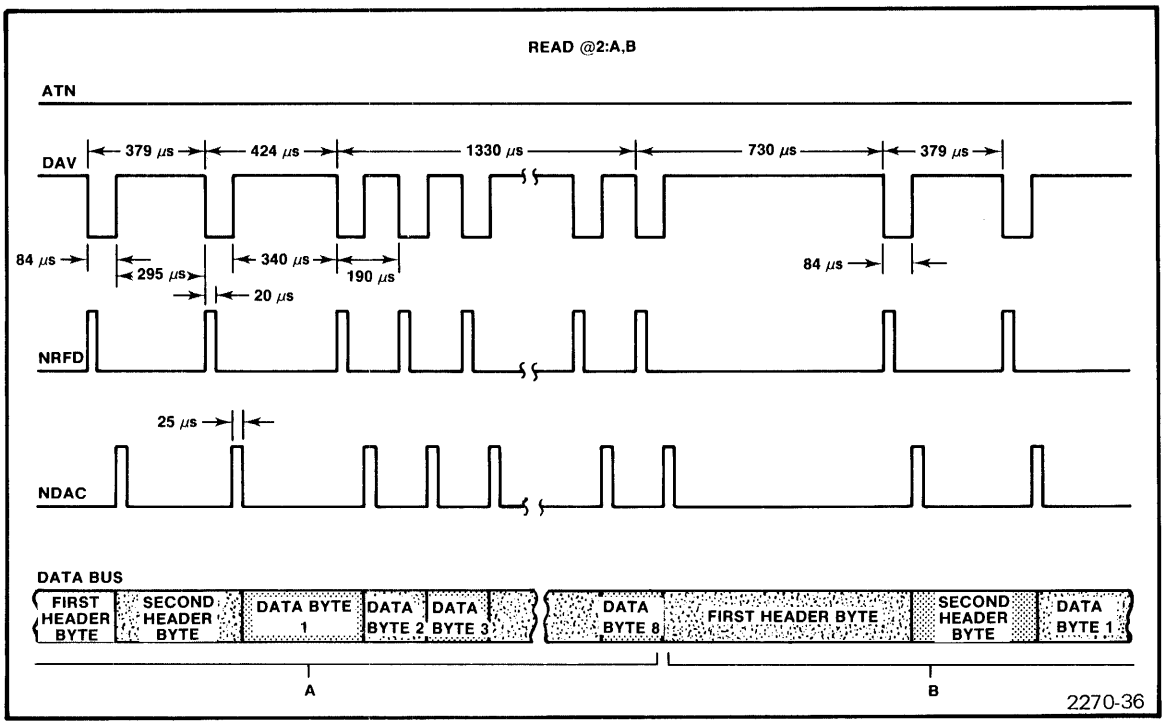


Fig. 4-17. Handshake Sequence for a READ Numeric Data Burst.

1. After ATN goes high at the end of the addressing sequence, the talker is allowed to place the first data byte on the Data Bus and wait for the lines to settle. The talker then waits for the 4051 to set NRFD high.

2. When the 4051 is ready to receive the first data item, the 4051 sets NRFD high. This happens 37 μs after ATN goes high.

3. Since a numeric variable is specified first in the READ parameter list, the talker must send a two byte header which tells the 4051 to prepare for an eight-byte floating-point number. The first header byte must be decimal 32; the second header byte must be decimal 8. If the first two bytes are not 32 and 8 respectively, then a data item mis-match occurs and the 4051 aborts the READ operation.

4. After the 4051 sets NRFD high, the talker sets DAV low. The 4051 responds by setting NRFD low; the 4051 then captures the data byte on the Data Bus and sets NDAC high. This action takes a minimum of 84 μs.

5. When NDAC goes high, the talker sets DAV high, then places the second header byte on the Data Bus. The 4051, in the meantime, analyzes the header byte to make sure that is is decimal 32. 295 μs after DAV goes high, the 4051 sets NRFD high, indicating that it is ready to receive the second byte of the header.

6. The handshake for the second byte of the header occurs the same as the handshake for the first byte of the header. It takes 84 μs (minimum) to execute.

7. The 4051 analyzes the second header byte to make sure that it is a decimal 8. In the mean time, the talker places the first byte of the floating-point number on the Data Bus and waits for the 4051 to set NRFD high. After 340 μs of analyses, the 4051 sets NRFD high and the 8-byte floating-point number is transferred as fast as the 4051 can gulp it in.

8. Since the header tells the 4051 all it needs to know about the numeric data item, the 4051 doesn't analyze the next eight data bytes as they are brought into memory. The 4051 assumes that the data is in the correct format, so all the 4051 does is receive the data bytes and place them in memory. In this case, the first eight data bytes after the header are assigned to the variable A. The data burst during this part of the transfer is much faster. Each handshake cycle is 190 μs; this gives a burst rate of 5263.2 bytes/sec.

9. After the eight data bytes are in memory, the 4051 has to stop and figure out where to store the next eight bytes. This time, coupled with the last handshake cycle takes 730 μs.

10. While the 4051 is placing the first numeric data item in memory, the talker prepares to transfer the second data item. During the 730 μs time lapse, the talker places the first byte of the next header on the Data Bus and waits for the 4051 to set NRFD high.

Computing the Data Sampling Rate for Character String Data

Fig. 4-17 shows that the total time to transfer a complete numeric value over the GPIB is 2863 μs. This gives an effective data sample rate of 349.3 samples/second.

Transferring Character String Data

If a string variable is specified in a READ statement parameter list, then the peripheral device (acting as a talker) must send a character string preceded by a two-byte header. The header must identify the data as a character string and tell the 4051 how many bytes are in the string. The header byte can be any byte from decimal 64 to decimal 127. The second header byte can be any byte from decimal 0 to decimal 255. (Refer to page 7-168 of the 4051 Graphic System Reference Manual for a complete explanation of the header format).

After the header is analyzed by the 4051, the 4051 transfers the number of bytes specified by the header. Each byte is treated as an ASCII character. For example, decimal 65 is converted to an "A," decimal 90 a "Z," and so on.

Fig. 4-18 illustrates the events on the GPIB when character strings are transferred during a READ operation. Two character strings are transferred in this sequence and are assigned to A$ and B$, respectively. Although A$ and B$ are dimensioned to eight characters each by a previous DIM statement, the controlling factor which determines the size of the character strings is the header. If the incoming character string is larger than the dimensioned size of the target variable, all of the characters in the string are still transferred, but those characters which fall out side the dimensioned size of the variable are discarded.

The events in Fig. 4-18 are discussed in a step-by-step fashion following the diagram.

1. When the 4051 finds a string variable in a READ parameter list, the 4051 prepares to receive a character string from the assigned talker. When the 4051 is ready, the 4051 sets NRFD high. The talker, having already placed the first byte of the header on the Data Bus, responds to NRFD by setting DAV low.

2. When DAV goes low, the 4051 goes through the handshake sequence and captures the data byte. The 4051 then checks the data byte to make sure that the byte meets the requirements for a character string header byte. If it doesn't, the 4051 aborts the READ operation at this point.

3. While the 4051 is checking the first byte of the header, the talker places the second byte of the header on the Data Bus and waits for the 4051 to set NRFD high.

4. It takes the 4051 system 211 μs to analyze the first header byte. When the 4051 is finished, the 4051 sets NRFD high.

5. The talker responds to NRFD by setting DAV low, and the handshake on the second header byte takes place.

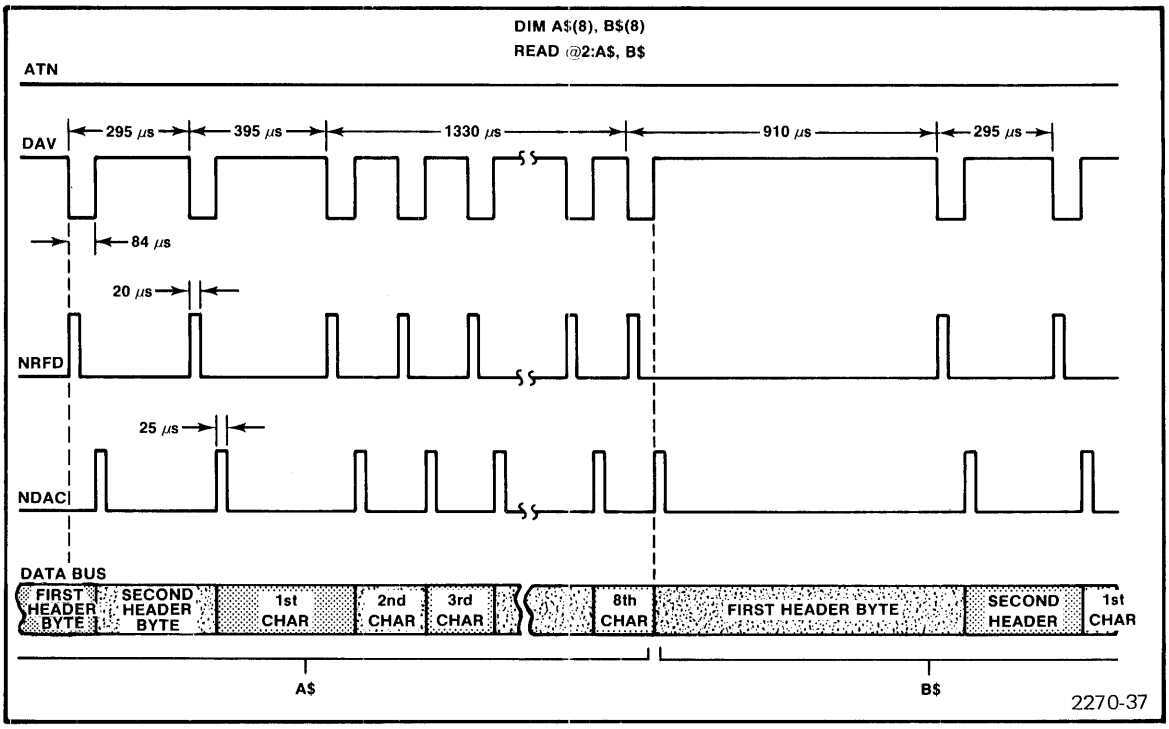


Fig. 4-18. Handshake Sequence for a READ String Data Burst.

6. The 4051 checks the second header byte and determines how many bytes are in the character string; in this example, eight bytes are in each string.

7. The analysis on the second header byte takes 311 μs. The 4051 then sets NRFD high, and the data burst for the character string starts.

8. Since the 4051 knows at this point how many bytes are in the character string, checking the data stream for delimiters is not necessary. The 4051, therefore, stuffs the specified number of data bytes into memory as fast as it can without looking at the data. Since the minimum handshake cycle during this burst is 190 μs, the burst rate during this phase of the transfer is 5263.2 bytes/sec.

9. After the last byte in the character string is transferred, the 4051 prepares to receive the next data item. This time period plus the last handshake cycle is 910 μs in duration.

Computing the Data Sampling Rate for Character String Data

Fig. 4-18 shows that the total time to transfer a character string into a 4051 memory with a READ statement depends on the length of the string. For an eight-byte string, the time to swallow the header is 690 μs; the data burst for the first 7 bytes lasts for 1330 μs, and the clean-up period lasts for 910 μs. The total time is 2930 μs, so the data sample rate is 341.3 samples/second in this case.

In general:

Data Sample Rate=1/(690+((No. of Characters − 1) x 190)+910) E-6

THE UNADDRESSING SEQUENCE FOR THE READ STATEMENT

After the last data byte is transferred in a READ operation, the 4051 clears the GPIB by activating ATN, issuing UNTALK and UNLISTEN, then releases ATN. This returns the GPIB to an idle state and frees the peripheral device to go on about its business. Fig. 4-19 illustrates the timing events that occur on the GPIB when the unaddressing sequence is executed at the end of a READ statement.

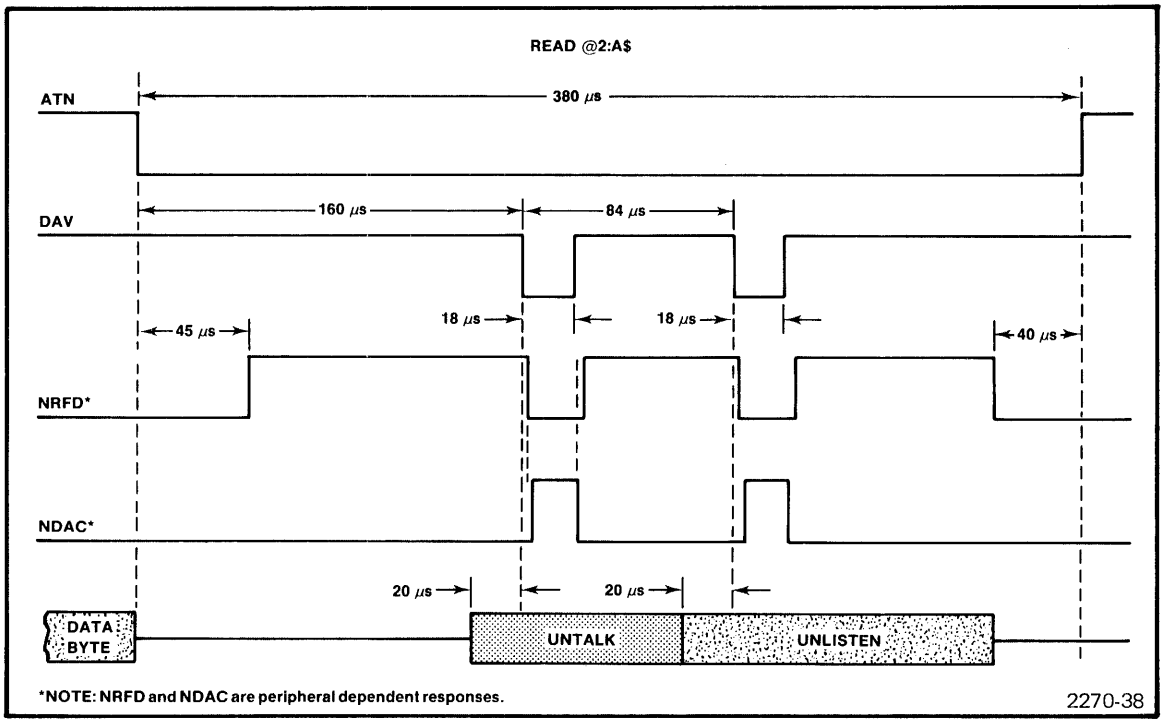


Fig. 4-19. Unaddressing Sequence for the READ Statement.

TIMING DETAILS FOR WBYTE

INTRODUCTION

The WBYTE statement is used to individually transfer peripheral addresses and data bytes over the GPIB, one at a time. Normally, this statement is used to communicate with peripheral devices which cannot talk in ASCII code or 4051 internal binary code. Unlike PRINT and WRITE, the WBYTE statement gives complete control over all eight lines on the Data Bus. Any binary bit pattern from 00000000 to 11111111 (decimal 0-255) can be sent to a peripheral device over the Data Bus with ATN active or inactive, EOI active or inactive, or both ATN and EOI active or inactive.

WBYTE gives complete control over the GPIB, however, the tradeoff for this versatility is slower speed. The reason that WBYTE is slower than PRINT or WRITE is that the statement overhead which normally occurs at the beginning of statements like PRINT and WRITE is sprinkled through the data transfer during WBYTE operations. Normally, the necessary conversions and preparations are made before activity occurs on the GPIB in statements like PRINT. In the case of WBYTE, however, each parameter in the parameter list is converted to the appropriate binary bit pattern just before it is transferred over the bus. The handshake cycle for each data byte is also different, because the conversion time for each decimal value is different.

In addition to slower speed, the WBYTE statement calls for more detail in the parameter list. The reason is that WBYTE is a more primative command (lower level) and much of the GPIB protocol is not taken care of automatically. For example, specifying a peripheral device number in a WBYTE statement for an I/O address is not enough. The decimal equivalent of the primary listen address or the primary talk address must be specified. Secondary addresses are not issued automatically either, nor is an UNTALK/UNLISTEN sequence issued automatically to clear the bus at the end of the statement. All of these events must be specified in detail in the WBYTE parameter list in order to make them happen.

TIMING DETAILS

Although the bus protocol (handshake sequence) for WBYTE is the same as in other BASIC statements, the time delays between signal transitions is different; furthermore, the delays vary as the parameters change. Because of this, two timing diagrams are discussed in detail to give you an idea about the relative time required to execute a WBYTE statement. The timing events for the statement WBYTE @66: are discussed first to give you an idea on how long it takes to issue a primary talk address. Remember, however, that changing 66 to another value, say 67, changes the total time it takes to execute the statement.

The second discussion centers around the statement WBYTE @35:A, where A is a three element array with the value of PI (3.14159265359) for each element. When this statement is executed, the primary listen address 35 is issued with ATN active low; ATN is released, then each element in array A is rounded to the integer 3 and the decimal 3 (00000011) is issued over the bus three times. This is not only illustrates how a variable in the WBYTE parameter list is automatically rounded to an integer, but it also illustrates how an array can be used to specify data bytes in a WBYTE statement.

TRANSFERRING MY TALK ADDRESS FOR DEVICE 2 WITH WBYTE

Fig. 4-20 illustrate the timing events that occur on the GPIB when the statement WBYTE @66: is executed by the 4051. A detailed discussion of these events follows the figure.

1. Normally, the GPIB is in an idle state (all line high, except NRFD and NDAC) before a WBYTE statement is executed, (however, the GPIB doesn't have to be idle). Since the first item specified in the WBYTE parameter list in Fig. 4-20 is an @ sign, the 4051 starts the bus activity by activating ATN (Attention).

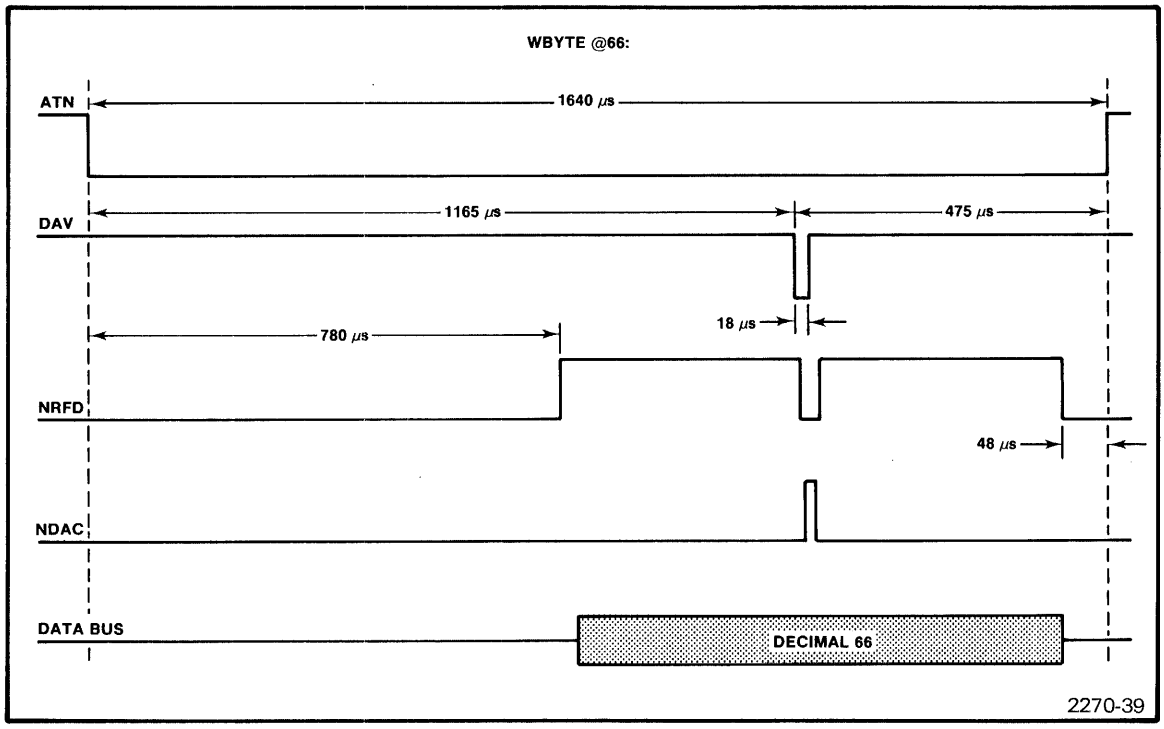


Fig. 4-20. Transferring One Peripheral Address with WBYTE.

2. 780 μs after ATN goes down, the 4051 releases NRFD and NDAC checks to make sure that both signals are not both high (inactive). If NRFD and NDAC are both high, the 4051 assumes that a peripheral isn't out there, so the WBYTE opertion is aborted and the message GP INTERFACE BUS I/O ERROR is printed on the display. If NDAC and NRFD are both low, the 4051 waits until NRFD goes high, before proceeding. And if NDAC is low and NRFD is high, the 4051 assumes that all devices on the GPIB are ready to receive the first byte.

3. If the peripheral devices are ready, the 4051 places the binary equivalent of decimal 66 on the Data Bus 780 μs after ATN goes low. 385 μs later, the 4051 sets DAV low which tells the peripheral devices that the data byte on the Data Bus is valid and can be captured.

4. The peripheral devices individually respond at their own pace by first setting NRFD low. They capture the data byte, then set NDAC high. The NDAC signal line goes high only after the last device sets NDAC high.

5. The 4051 responds the NDAC by setting DAV high. This indicates that the data byte is no longer valid. The total time to complete this handshake sequence is at a minimum 18 μs (plus a few nanoseconds).

6. Since the colon (:) is the next item in the parameter list, the 4051 knows that it should end the addressing sequence. So, 409 μs after DAV goes high, the 4051 takes the data off the Data Bus and sets NRFD and NDAC low; 48 μs after that, the 4051 releases ATN which ends the activity on the bus.

At this point, device 2 is authorized to place a data byte on the Data Bus, but is not authorized to activate DAV until NRFD goes high. Since the WBYTE statement is over, the 4051 thinks the operation is finished and proceeds to the next BASIC statement in memory. If the next BASIC statement is not an RBYTE statement which allows the 4051 to receive data bytes from device 2, or if the next BASIC statement is not another WBYTE statement which assigns another peripheral device on the line to listen to device 2, then the bus will hang in this unfinished state with device 2 trying to talk and no one to listen.

TRANSFERRING MY LISTEN ADDRESS AND THREE DATA BYTES WITH WBYTE

Fig. 4-21 illustrates the activity on the GPIB when the statement WBYTE @34: A is executed by the 4051. In this case, A is a three element array with each element equal to PI (3.14159265359). A detailed step-by-step description of these events follows the diagram.

Transferring the Address

1. When the WBYTE statement in Fig. 4-21 is executed, the GPIB is in an idle state with all lines high, except NRFD and NDAC. Normally the bus is in an idle state, but doesn't have to be.

2. The WBYTE statement in Fig. 4-21 starts with @34:, so the 4051 starts the bus activity by setting ATN low; 910 μs later, the 4051 places the binary equivalent of decimal 34 on the Data Bus and releases NRFD and NDAC.

3. At this point, the 4051 checks to make sure that NRFD and NDAC are not both high. If they are both high, the 4051 assumes that nobody is out there and aborts the WBYTE operation. If NRFD and NDAC are both low, then the 4051 assumes that a peripheral device is busy and waits for NRFD to go high. If NDAC is low and NRFD is high, then the 4051 assumes that all the peripheral devices on the bus are ready to receive the data byte.

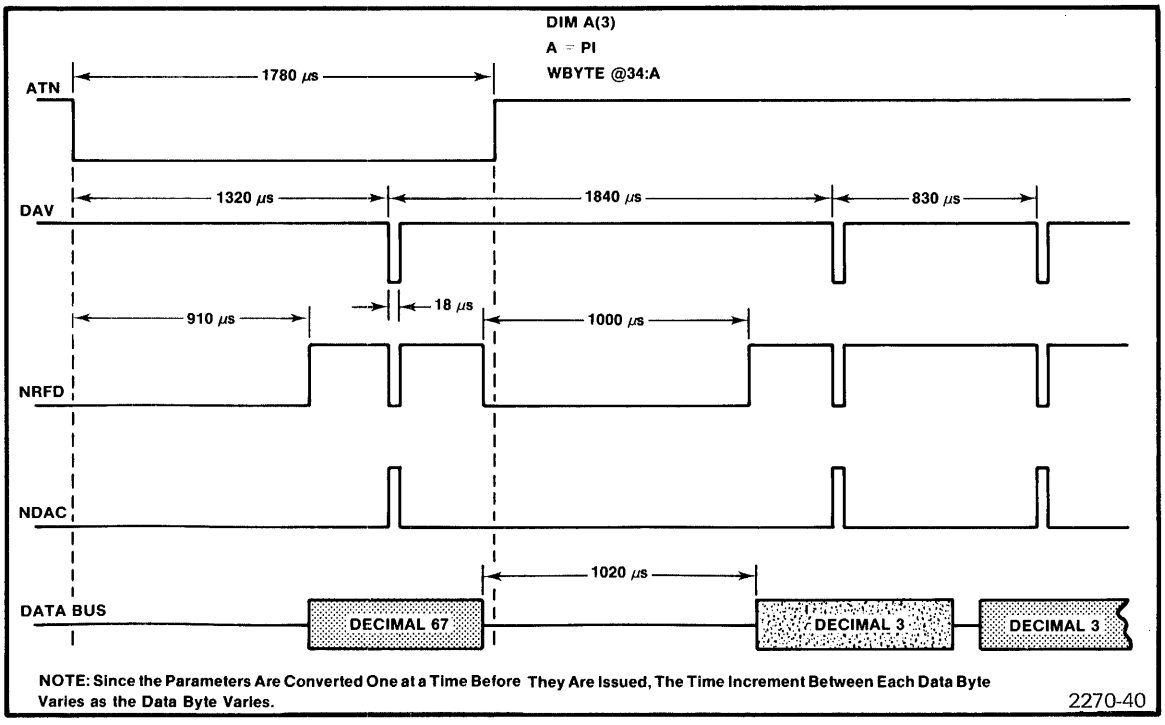


Fig. 4-21. Transferring an Array of Data Bytes with WBYTE.

4. If all are ready, the 4051 sets DAV low.

5. Each peripheral device on the bus must respond to DAV, because ATN is low. The devices respond individually at their own pace by setting NRFD low and capturing the data byte; when the byte is captured, the peripheral devices individually set NDAC high.

6. When all the peripheral devices have set NDAC high, the signal line goes high. This tells the 4051 that all have captured the data byte, so the 4051 responds by setting DAV high. This tells the peripheral device that the data byte is no longer valid.

7. At this point, the 4051 knows that it has issued decimal 34 with ATN down, but doesn't know whether 34 is a primary listen address, a primary talk address, or a secondary address. So, the 4051 returns the GPIB to its previous idle state before ATN was set low.

8. Approximately 400 μs after the DAV is set high, the 4051 sets NRFD and NDAC low, then releases ATN 48 μs after that.

Transferring the Data Bytes

1. After the address 34 is transferred, the 4051 assumes that the correct peripheral device received the address and is prepared to respond accordingly. In this case, 34 is the My Listen Address for device 2 and the 4051 assumes that device 2 is prepared to receive the data bytes about to be placed on the bus.

2. When ATN is set high, the 4051 leaves the bus momentarily to prepare the first data byte for transfer. In this case, array A is specified as the data source, so the 4051 retrieves the value assigned to the first element A(1), rounds the value to an integer, converts the integer to its binary equivalent, and places the data byte on the Data Bus. In this case, the value of PI is rounded to decimal 3, and the binary number 00000011 is placed on the Data Bus. The conversion time in this case is 952 μs after ATN is released, but it varies in each case.

3. When the first data byte is placed on the Data Bus, the 4051 releases NRFD and NDAC and checks the condition of the peripheral device. If NRFD is high and NDAC is low, the 4051 waits for the data lines to settle, then starts the handshake by setting DAV low.

4. After the first data byte is transferred, the 4051 retrieves the value assigned to the second element in array A, converts the value to its binary equivalent and places the byte on the Data Bus. This cycle takes 830 μs in this case.

5. The 4051 repeats the above operation until all the data assigned to array A is transferred over the GPIB.

At this point, the statement is over and the 4051 moves on to the next BASIC statement in memory. Normally, the next statement is a WBYTE @95,63: statement which clears the bus by issuing UNTALK/UNLISTEN, or another WBYTE statement which contains more data bytes for device 2. If device 2 is not released at this point with an UNLISTEN command, then the GPIB hangs in this state until power is removed from the 4051 or until an INIT statement is executed to reset the peripheral interface.

TIMING DETAILS FOR RBYTE

INTRODUCTION

The RBYTE statement is a primitive command (lower order) and can only be executed if an active talker is waiting on the GPIB to talk to the 4051 or to another listener device. The only way to set up this arrangement is to execute a WBYTE statement prior to the RBYTE statement and assign a peripheral device as a talker.

If only one target variable is specified in an RBYTE statement, then the 4051 comes on the bus, handshakes with the talker, captures a data byte, then leaves the bus. The decimal equivalent of the data byte is assigned to the target variable; the 4051 then moves on to the next BASIC statement in memory. The above activity on the GPIB lasts for 110 μs.

If more than one target variable is specified in the RBYTE parameter list, then the 4051 keeps handshaking and receiving data bytes until each variable has an assigned value. The handshake cycle for simple numeric variables is 1.78 ms. The handshake cycle for array variables is a little faster at 1.48 ms.

RECEIVING ONE DATA BYTE

Fig 4-22 illustrates the activity on the GPIB when the statement RBYTE A is executed (where A is not an array). A detailed explanation of these events follows the figure.

1. After a peripheral device is assigned as a talker with a WBYTE statement (WBYTE @66: for example), all the GPIB signal lines are high, except for NRFD and NDAC which the 4051 keeps low. At this point, the talker should have the first Data Byte ready on the Data Bus and should be monitoring NRFD, waiting for the line to go high so the transfer can begin.

2. The 4051 starts the RBYTE sequence by setting NRFD high. This tells the talker that the 4051 is ready to receive the first data byte.

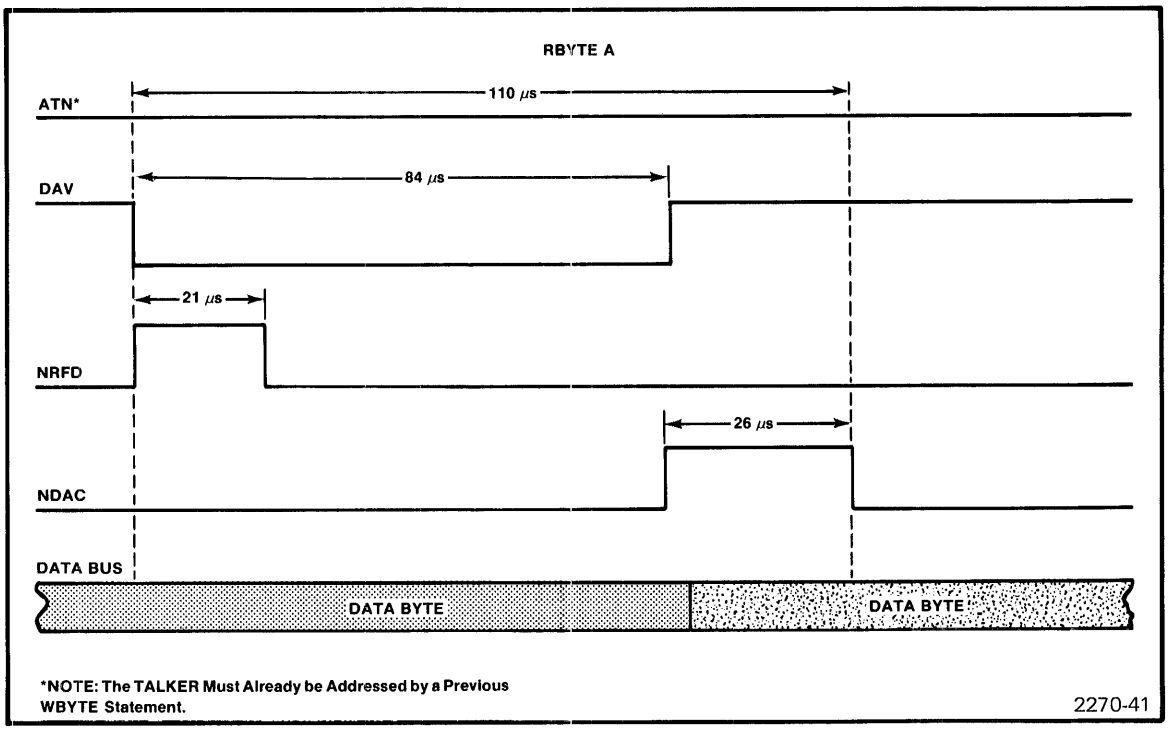


Fig. 4-22. Handshake Sequence for RBYTE.

3. The talker responds to NRFD by setting DAV low. This tells the 4051 that the data byte on the Data Bus is valid and can be captured.

4. The 4051 responds 21 μs later by setting NRFD low; 80 μs after that, the 4051 captures the data byte and sets NDAC high.

5. The talker responds to NDAC by setting DAV high. Since this is the only data byte that the 4051 wants, the 4051 terminates the RBYTE statement at this point; 26 μs after DAV goes high, the 4051 sets NDAC low, keeps NRFD low, then moves on to the next BASIC statement in memory.

6. At this point, the operation is over as far as the 4051 is concerned. However, the talker doesn't know this because an UNTALK command has not been issued. The talker thinks the 4051 is busy digesting the first data byte, so it waits for NRFD to go high. If the next BASIC statement is not a WBTYE @95: statement, then the talker might wait forever in this state and not be allowed to continue its operations.

RECEIVING MORE THAN ONE DATA BYTE

Fig. 4-23 illustrates the GPIB activity when more than one target variable is specified in the RBYTE parameter list. A detailed explanation of these events follows the figure.

1. The handshake timing for receiving multiple data bytes with an RBYTE statement is the same as shown in Fig. 4-22. The only difference is that the 4051 keeps handshaking with a time delay between each handshake. The time delay is of primary importance when determining the data rate over the bus.

2. The time delay between handshakes is illustrated in Fig. 4-23. If simple numeric variables are specified in the parameter list, then the handshake cycle is repeated every 1.78 ms until every target variable has an assigned value.

3. If an array variable is specified in the parameter list, the handshake cycle is reduced to 1.48 ms until each element in the array has an assigned value.

4. After the last vaiable in the parameter list has an assigned data byte, the statement is over and the BASIC interpreter starts executing the next BASIC statement in memory (or monitors the 4051 keyboard for further input).

Fig. 4-23 shows that the input data rate for numeric variables is 1/1.78 ms which computes to 561.8 bytes/second.

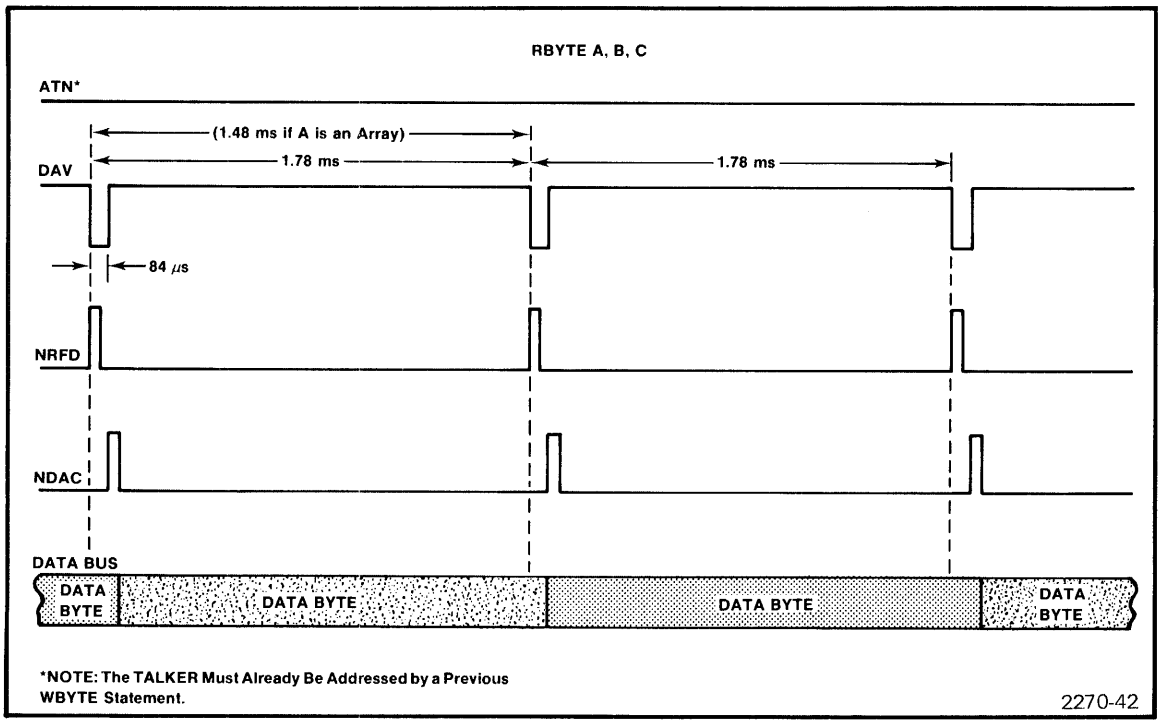


Fig. 4-23. Receiving Three Data Bytes with RBYTE.

TIMING DETAILS FOR SERIAL POLL

INTRODUCTION

When a peripheral device on the 4051 GPIB needs service, the device requests service by setting the GPIB signal SRQ (Service Request) active low. Because SRQ is a common line on the GPIB, and shared by all devices, 4051 must execute a serial poll to determine which device is requesting service. Serial poll operations are implementated within the 4051 BASIC language. Normally, an ON SRQ THEN statement is executed first in a BASIC program before the main program in run. This statement tells the 4051 microprocessor where to branch in the BASIC program when a GPIB device sets SRQ low. The branch normally directs the microprocessor to a POLL statement in the BASIC program. This statement causes the 4051 controller to execute a serial poll on the GPIB. Since the parameters of the polling operation are specified in a POLL statement, this statement gives the 4051 keyboard operator complete control over the priority interrupt structure on the GPIB. The POLL statement returns with two pieces of information: (1) the device requesting service, (2) the status byte for the device requesting service. From this information, the BASIC program can select alternate courses of action. Normally, a GOTO. . .OF statement is placed after the POLL statement to direct the 4051 to the beginning of the device service routine which is located elsewhere in the BASIC program. On the bases of the status byte information, the BASIC program can be determined what kind of service the device needs and then take the appropriate action. Complete details on how to use the POLL statement are given in the 4051 Graphic System Reference manual in Section 7.

The rest of this text concentrates on the actual serial poll sequence itself and explains what takes place on the 4051 GPIB during the execution of a POLL statement.

SERIAL POLL FLOW DIAGRAM

The flow diagram (Fig. 4-24) illustrates the 4051 controller routine during the execution of a POLL statement. Refer to this flow diagram as you need to in order to get a clear picture of the serial poll sequence.

THE 4051 POLL STATEMENT TIMING DETAILS

Fig. 4-25 is a timing diagram that illustrates the events on the 4051 GPIB during the execution of the statement 500 POLL M,W;2, 13;3;5. Three peripheral devices are connected to the GPIB. Device number 2 is a frequency scanner and is given the highest priority position in the address list. In this case, device 2 needs a device dependent secondary address 13 to indicate that it should send its status byte. Device 3 is a digital voltmeter connected to the GPIB. In this example, device 3 requests service by holding the SRQ line low on the GPIB. Because device 3 is requesting service, bit 7 in its status byte is set to a binary 1. Device 5 is a Tektronix 4924 digital tape unit. Because the 4924 tape unit requires the least amount of service in this arrangement, the tape unit is listed last in the POLL statement address list.

Timing Details for Serial Poll

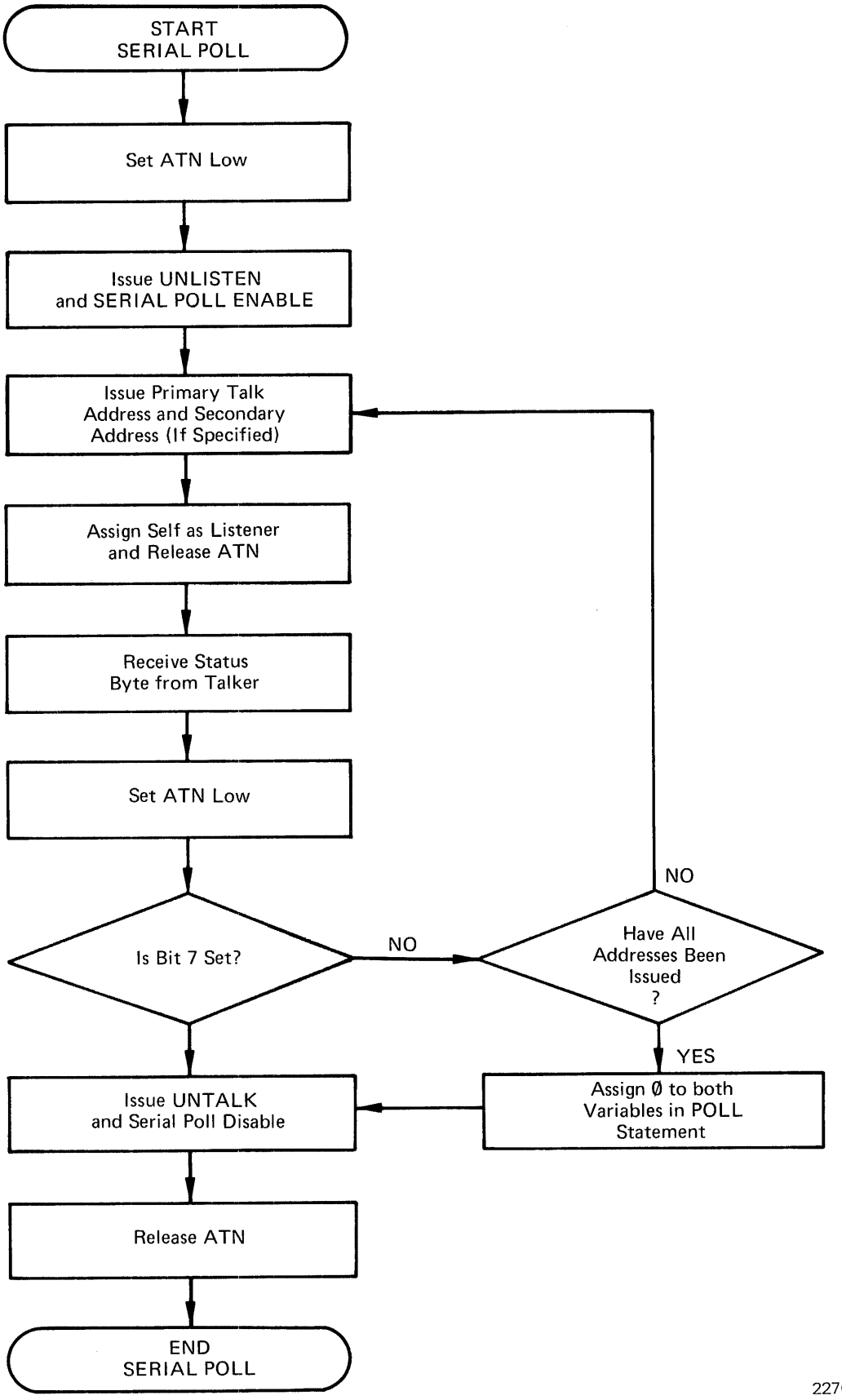


Fig. 4-24. 4051 Serial Poll Flow Diagram.

Timing Details for Serial Poll

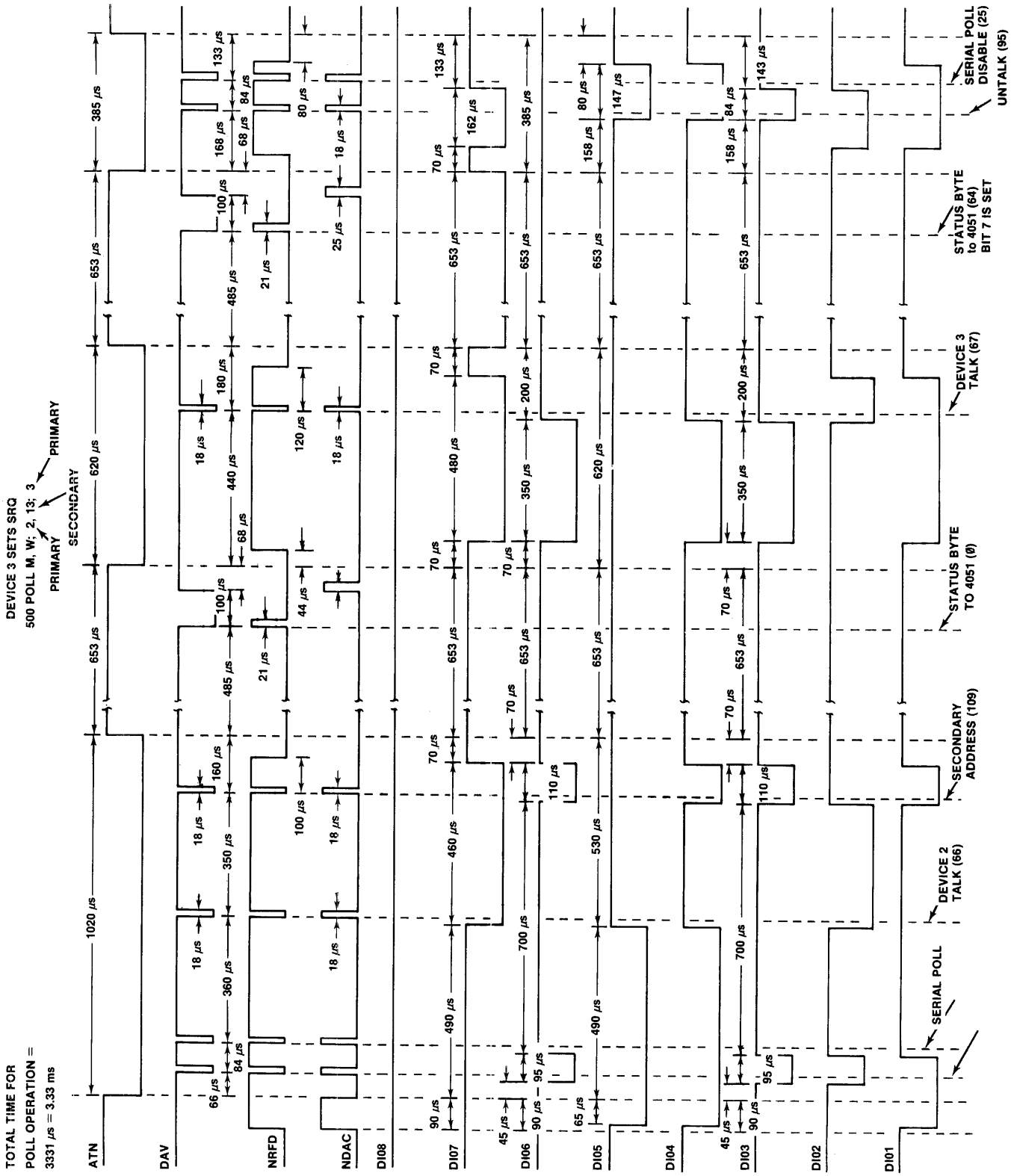


Fig. 4-25. 4051 serial poll timing diagram.

Timing Details for Serial Poll

Assume that the 4051 is executing a BASIC program and device 3 (the digital voltmeter) sets SRQ on the GPIB to a low state. This causes the 4051 to branch to a previously executed ON SRQ THEN 500 statement to see how to handle the SRQ interrupt. The 4051 then branches to the statement 500 POLL M,W;2,13;3;5 and executes at serial poll on the GPIB as follows:

1. The 4051 controller starts the serial poll operation by releasing NRFD and NDAC, then set ATN low. This causes all peripheral devices on the GPIB to stop what they are doing and prepare to receive and decode addresses and controller commends.

2. After 45 μs, the 4051 places the command UNLISTEN (decimal 63) on the bus, waits 21 μs for the lines to settle, then sets DAV low.

3. The three devices on the bus handshake with the 4051 in unison and receive the UNLISTEN command byte. This clears the listen state of every device on the GPIB.

4. Next, the 4051 issues the command SPE (serial poll enable—decimal 24). The devices on the GPIB handshake in unison and receive this command. SPE tells each device that the 4051 is executing a serial poll. Each device at this time updates its status byte and prepares to transmit its status byte to the 4051 when device is the assigned to be a talker. The SPE command is issued 84 μs after the handshake is finished on the UNLISTEN command byte.

5. After SPE is transferred over the GPIB, the 4051 prepares to transfer the primary talk address and the secondary address (if one is specified) to the first device listed in the POLL statement address list. In this case, device 2 is listed first with secondary address 13. The 4051, therefore, issues decimal 66, followed by decimal 109. This tells device 2 to be ready to transfer its status byte.

6. After secondary address 13 is transferred, in this case, the 4051 assigns itself as a listener, sets both NRFD and NDAC low, then releases ATN.

7. Device 2 responds to the high (inactive) ATN signal by placing its status byte on the GPIB Data Bus. In this case, assume the status byte for device 2 is decimal 0 (all lines high).

8. After 485 μs, the 4051 handshakes with device 2 and receives the status byte. Since bit 7 (on the DI07 line) is not set to a binary 1, the 4051 assumes that device 2 is not requesting service.

9. Sixty-eight microseconds after the status byte for device 2 is received, the 4051 sets ATN low. This causes all devices on the GPIB to listen again to the GPIB.

10. The 4051 places the talk address for device 3 on the GPIB (decimal 67), then sets DAV low. All devices handshake with the 4051 and receive the talk address.

11. Device 3 prepares to send its status byte. Device 2 at this time must interpret the talk address for device 3 as an "implied" untalk command and gets off the bus. Device 5 does not recognize the device 3 talk address and remains idle.

12. The 4051 then assigns itself as a listener again, holds both NRFD and NDAC low, then releases ATN.

13. This time device 3 places its status byte on the GPIB Data Bus and waits for the 4051 to set NRFD high. Since device 3 is holding SRQ low on the GPIB, bit 7 in its status byte is set low (true).

14. 485 μs after the 4051 releases ATN, the 4051 handshakes with device 3 and receives the status byte. This action tells device 3 to release SRQ. SRQ goes high at this time (unless device 5 is also requesting service, or unless device 2 has set SRQ low after it was polled).

15. Since bit 7 is set to binary 1 in the status byte for device 3, the 4051 knows that device 3 is requesting service. The 4051, therefore, assigns the number 2 to the variable M, because device 3 is the second device in the POLL statement address list. The 4051 also assigns the decimal equivalent of the status byte (decimal 64 in this case) to the variable W. This allows the BASIC program to analyze the meaning of the status byte in BASIC the statements to follow.

16. The 4051 terminates the serial poll operation at this point. The 4051 sets ATN low, then issues the command UNTALK (to unaddress device 3) and SPD (serial poll disable—decimal 25). SPD tells each device on the GPIB that the serial poll is over. The 4051 then releases ATN and leaves the GPIB.

At this point, the 4051 executes the next BASIC program statement in memory (the line following the POLL statement). If the next statement is a RETURN statement, the 4051 branches back to the point in the BASIC program where it was interrupted in the first place and continues with the main program. Normally, however, the next statement after the POLL statement is a GOTO M OF statement or a GOSUB M OF statement that directs the BASIC interpreter to the service routine for device M (device 3 in this case). A RETURN statement following the service routine directs the 4051 back to the point in the main program where the SRQ interruption first occuered. (See Section 7 in the 4051 Graphic System Reference manual for full details.)

USING EOI TO TERMINATE A DATA TRANSFER

The EOI signal line on the 4051 GPIB is a general purpose control line normally used to signal the end of a data transfer. Although EOI can take on additional meaning at the option of the peripheral designer, the 4051 always interprets EOI as meaning "End of Transfer".

If the transmitting device activates EOI to mark the end of a data transfer, EOI must be activated just prior to DAV going low on the last data byte transferred. EOI must then stay low until the handshake is complete and the receiving device sets NDAC inactive high. The timing for this event is shown in Fig. 4-26.

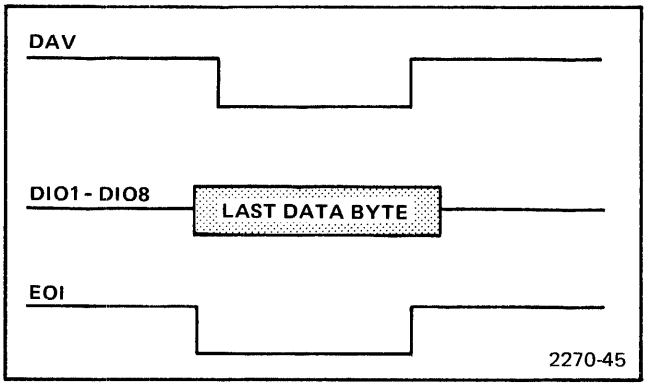


Fig. 4-26. Activating EOI with the last data byte.

Normally, the receiving device captures the active state of the EOI signal at the same time the last data byte is captured and treats the EOI signal as a ninth data bit in the byte. The receiving device should interpret EOI data byte combination the same as it would if a CR delimiter is received. For example, if the 4051 is executing the statement INPUT @2:A$,B$ and the transmitting device issued EOI with a data byte, the 4051 assigns all the characters received up to that point to A$ (including the data byte that was being passed when EOI went active). The transmitting device then continues to send characters to the 4051 until EOI goes active again, or until CR is issued, or until both events occur simultaneously. At that point, the 4051 terminates the INPUT because B$ is the last variable specified in the INPUT statement. In all cases, the 4051 treats the data byte/EOI combination as data byte/CR.

If the 4051 is transmitting data over the GPIB, the 4051 always activates EOI with the last byte transferred, except during a WRITE operation. Since WRITE operations transfer internal binary data and each data item has a header describing the length of the item, the 4051 assumes that the receiving device is able to read the header and determine when the last byte is being transferred. (See Appendix A for the details on interpreting the binary header format.)

Using EOI to Terminate a Peripheral to Peripheral Transfer

Sometimes it is desireable to have one peripheral device talk directly to another peripheral device on the GPIB without involving the 4051. For example, you might want to transfer data from a digital voltmeter to a line printer or a storage device. The following example illustrates how one peripheral device is assigned a talker role and several other peripheral devices are assigned listener roles. This sequence starts a direct transfer from one peripheral device to another peripheral device. The transfer is terminated when the transmitting device activates the EOI signal line on the GPIB for at least 350 microseconds.

```
140 REM Set Up Transfer
150 ON EOI THEN 180
160 WBYTE %95,63,70,109,52,35:
170 WAIT
180 WBYTE @95,63:
190 REM Continue with BASIC program
.
.
.
.
```

When line 150 is executed under program control, the EOI (End or Identify) interrupt facility is activated. This tells the BASIC interpreter to be on the lookout for an active EOI signal line on the GPIB. When EOI goes active at a later time, the BASIC interpreter will transfer program control to line 180.

Line 160 is executed next. The UNTALK and UNLISTEN commands make sure the GPIB is clear. Primary talk address 70 then tells peripheral device number 6 that it is going to be the talker for the upcoming data transfer. Secondary address 109 tells device 6 to start sending ASCII data over the GPIB as soon as the ATN signal line is released. The primary listen address for device 20 is issued next (decimal 52), followed by primary listen address for device 3 (decimal 35). This tells peripheral device 20 and peripheral device 3 to start listening to the talker when the ATN signal line is released. The percent sign (%) is specified in this case instead of the "at" sign (@); this tells the 4051 BASIC interpreter to get off the bus and let the assigned talker take over when the ATN line is released. The colon in the WBYTE statement (:) causes the BASIC interpreter to release ATN.

At this point, the talker (device number 6) takes over the bus and starts sending data to device number 20 and device number 3. The BASIC program in the meantime goes to line number 170 and waits patiently for the talker to activate EOI as the last byte of data is transferred over the bus. When EOI is activated (for at least 350 μs), program control is tranferred to line 180 where the BASIC interpreter activates the ATN signal line and issues the universal commands UNTALK/UNLISTEN over the GPIB. This places all active peripheral devices on the GPIB in a known quiescent state and terminates the I/O operation. BASIC program execution continues on normally from that point.

Section 5

4051 GPIB DATA RATES

FACTORS THAT CONTROL GPIB DATA RATES

This section discusses the methods used to calculate GPIB data rates for the 4051 Graphic System. The variables that control the data rates are also examined. Fixed values are then established for these variables to illustrate how accurate data transfer rates can be calculated for a given application. This section also contains the technical details about the data rates and their origin.

MAXIMUM IEEE STANDARD 488-1975 DATA RATES

The IEEE Standard 488-1975 defines the maximum rate of transfer on the GPIB to be 250K bytes/second using 48 mA open-collector bus drivers and 1 mega bytes/second using 48 mA tri-state drivers (clause 31 on page 99). These figures are based only on the electrical limitations of the bus. More important are the design limitations of the devices connected to the bus. These design limitations are the dominant and controlling factors that determine the maximum transfer rates for any one system.

THE GPIB IS AN ASYNCHRONOUS BUS

Asynchronous means that the GPIB operates at the speed of the slowest device taking part in a transfer at any one time. If two devices are relatively fast, then the data transfer rate will be relatively fast. But, if a fast device is transferring data to a slower device, the bus operates at the speed of the slower device. If many devices are connected to the bus at one time, then the data rate will vary according to the devices that are taking part in a particular transfer.

WHY GPIB DATA RATES ARE DIFFICULT TO COMPUTE

Data rates for asynchronous GPIB operations are difficult to compute because the data rate varies as different devices are connected to the system. Furthermore, a device may be fast during some phases of its operation and slower during others. And, in some cases this speed variance depends on the type of data being transferred at the time (character string data or numeric data) and the data format (ASCII code, binary code, BCD, etc.). In addition, the data rate may also depend on the programed state of a transmitting or receiving device, if the device is programmable.

Taking all these factors into account, it can be seen that general statements about GPIB data rates need to be well qualified and the system parameters well defined. The details of each application must be investigated before an attempt can be made to calculate the data rate. In some cases, a system must be put together and placed in operation, then timing measurements made with an oscilloscope or logic analyzer to determine the actual data rates for each type of data transfer.

FACTORS THAT GOVERN THE 4051's PERFORMANCE ON THE GPIB

Essentially three factors govern the 4051's speed on the GPIB.

1. **The 4051 is a Microprocessor Controlled Device.** The microprocessor within the 4051 gives the 4051 the advantage of low cost and versatility — while maintaining a high degree of reliability. The microprocessor itself is fast, running on a 1.2 microsecond cycle. On the average it takes 4 clock cycles to execute a microprocessor instruction, and it takes several microprocessor instructions to respond to an event on the bus. Because of this, the 4051's response to an event on the GPIB is governed by how many microprocessor instructions are programmed into the routine which responds to the event. The number of microprocessor instructions usually depends on how the 4051 is programmed at the time of the transfer; that is, it depends on which BASIC statement is being executed and how that statement is constructed.

2. **The 4051 is an Intelligent Device.** Any device which can be programmed, can perform a multitude of tasks not normally possible with a strictly hardware design interface. For example, the 4051 not only has the ability to receive information over the GPIB, but at the same time it has the ability to analyze and process the information as it is being received. The more the 4051 has to "think" during a process, the more time it takes to perform the task. As an example, if the "%" mode is specified in an INPUT operation, the 4051 must check the incoming data stream for alternate delimiters along with the many other tasks it must perform. This additional task adds more time to the transfer and the effective data rate is decreased.

3. **The General Purpose Input/Output Data Buffer is 72 Characters in Length.** ASCII data traveling into and out of the 4051 memory passes through a general purpose input/output data buffer. The length of this buffer corresponds to the maximum length of a BASIC program line (72) and also to the default length of a character string (72).

During ASCII data transfers to and from peripheral devices over the GPIB, the data appears to travel in 72 character bursts. The time between the bursts represents the time it takes the 4051 microprocessor to load, process, and remove the data from the I/O buffer. This time is called buffer overhead.

The 4051 buffer overhead depends on what the microprocessor has to do with the data in the buffer. On an INPUT operation involving numeric data, the buffer sorting characteristics are dependent on the data format. If numeric variables are specified in the INPUT statement, for example, the microprocessor stops the data transmission after a delimiter is encountered or after 72 characters are received, whichever occurs first. The data stream is then examined for valid numeric data. In this process, the microprocessor ignores all non-numeric characters; the processor converts the numeric data items from ASCII code to internal floating-point format, then stores each data item in memory.

Floating-point conversions take time, and it's hard to predict how much time; it depends on the data format. If, for example, 72 characters are input and each data item contains 10 characters each, then only seven floating-point conversions have to be made before the microprocessor can get back to the bus and input more data. But if each data item in the buffer is one character in length with a one character delimiter (i.e., 1,2,3,4,...) then 36 floating point conversions have to be made before the processor can get back to the bus to input more data. The buffer overhead time in the latter case is approximately five times longer. It is apparent then that the data format plays a major role in determining the buffer overhead and the over-all effective data rate.

ASSUMPTIONS THAT MUST BE MADE WHEN USING 4051 DATA RATES

Because there are so many variables which affect the data rates on the GPIB, a few assumptions must be made concerning the 4051's performance on the GPIB. The data rates quoted in this manual assume that all peripheral devices respond instantly to the 4051 at all times. If the peripheral device slows the 4051 up at any point, then the stated data rates may not be achieved. Second, each data rate depends on a specific set of conditions. If these conditions change, then the data rates will change accordingly.

4051 GPIB DATA RATE SUMMARY

Table 5-1 lists the maximum effective data rates that can be achieved on the 4051 GPIB with the 4051 taking part in the transfer. This table summarizes the conclusions that can be drawn from material in the rest of this manual. Note that these figures are approximate values. Also be aware that the figures are valid only if the 4051 is not slowed down by a peripheral device taking part in the transfer.

In most cases, the number of data samples (or readings) that can be transferred to and from the 4051 in one second is different than the number of data bytes that can be transferred on one second. In most cases the data byte transfer rate is misleading. On the other hand, the data byte transfer rate is sometimes easier to specify exactly. For example, the chart shows that during a READ operation, data bytes are transferred at a rate of 5236 bytes/second. However, the maximum data sample rate is 350 samples/second. This data sample rate is fixed for READ operations; it varies with INPUT operations, depending on the number of data bytes in each sample.

Exercise caution when you use these figures. Make sure you understand where the figures come from by studying the rest of this manual. Then you can be reasonably sure that your GPIB data rate calculations are accurate.

Table 5-1

MAXIMUM EFFECTIVE 4051 GPIB DATA RATES

BASIC STATEMENT	DATA BYTE TRANSFER RATE DURING A BURST	DATA SAMPLE TRANSFER RATE (10 Bytes/Sample) (<72 byte total)	DATA SAMPLE TRANSFER RATE (10 Bytes/Sample) (=>72 byte total)
PRINT @2:A;	7936.5 Bytes/Sec	793.7 Samples/Sec	UNKNOWN (Depends on the Format)
PRINT @2:A$;	7936.5 Bytes/Sec	793.7 Samples/Sec	UNKNOWN (Depends on the Format)
DIM A (100) INPUT @2:A	4098.4 Bytes/Sec	409.8 Samples/Sec	293.5 Samples/Sec
INPUT %2:A$	3984.1 Bytes/Sec	398.4 Samples/Sec	368.5 Samples/Sec
WRITE @2:A	7936.5 Bytes/Sec	500.5 Samples/Sec	500.5 Samples/Sec
WRITE @2:A$	7936.5 Bytes/Sec	421.9 Samples/Sec	421.9 Samples/Sec
DIM A (100) READ @2:A	5263 Bytes/Sec	350 Samples/Sec	350 Samples/Sec
DIM A$ (1000) READ @2:A$	5236 Bytes/Sec	731 Samples/Sec	731 Samples/Sec
WBYTE A (A is an Array)	≈1190 Bytes/Sec	≈1190 Samples/Sec*	≈1190 Samples/Sec*
RBYTE A,B,C	561.8 Bytes/Sec	561.8 Samples/Sec*	561.8 Samples/Sec*
DIM A (1000) RBYTE A	675.7 Bytes/Sec	675.7 Samples/Sec*	675.7 Samples/Sec*

*Assumes a 1 Byte Sample

COMPUTING 4051 GPIB DATA RATES

During the course of an input or output operation over the 4051 GPIB, six events occur in sequence. These events are graphically illustrated in Fig. 5-1.

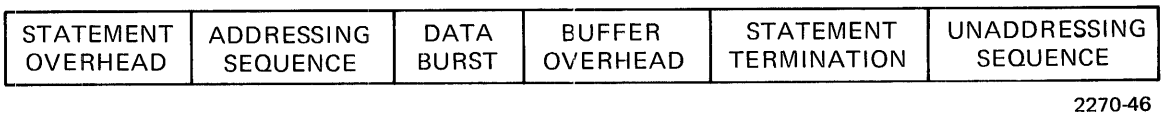


Fig. 5-1. Six Timing Events During a GPIB Data Transfer

THE STATEMENT OVERHEAD TIME PERIOD

When the 4051 BASIC interpreter executes an input or output statement, the interpreter first examines the statement to see what's there. The interpreter must determine whether the statement is an INPUT statement, a READ statement, or a PRINT statement, etc. The interpreter must look at the I/O address and analyze the address to see if default values are needed. The interpreter must look at the parameter list to see if string variables, numeric variables, or array variables are specified. If numeric expressions are specified for output, the interpreter must reduce the expressions to numeric constants before anything else is done.

This analysis time is called "statement overhead". This time period starts when the program line counter advances to the BASIC statement and ends when the first activity occurs on the bus. The statement overhead is variable and depends entirely on how the statement is constructed. Furthermore, the overhead may involve the execution of other BASIC statements and functions. Details on how to compute the statement overhead for PRINT, INPUT, WRITE, READ, WBYTE, and RBYTE are given later on in this section.

THE ADDRESSING SEQUENCE

The first activity on the bus marks the beginning of the second timing event; the BASIC interpreter activates ATN, issues the I/O address for the peripheral device specified in the I/O statement, then releases ATN. This is called the Addressing Sequence and is shown in Fig. 5-1.

THE DATA BURST TIME PERIOD

The data burst is the third event in the I/O operation and starts when ATN goes inactive high after the addressing sequence. At this time, the BASIC interpreter starts transferring data bytes over the GPIB at the maximum rate for the given operation. On input operations, data bytes are received over the bus from the specified peripheral device and are placed in the general purpose input/output buffer (henceforth called "I/O buffer"). On output operations, the BASIC interpreter takes data bytes out of the I/O buffer and places them on the GPIB Data Bus. Since the ASCII data I/O buffer can hold a maximum of 72 characters, the data burst may last as long as it takes to transfer 72 characters.

THE BUFFER OVERHEAD TIME PERIOD

The buffer overhead time period (henceforth called "buffer overhead") begins at the end of every data burst and lasts for a variable amount of time, depending on the operation being performed. On output operations like PRINT, for example, the buffer overhead is the time it takes the BASIC interpreter to convert the specified data from internal binary format to external ASCII format and to load the data into the I/O buffer. On input operations like INPUT, the buffer overhead is the time it takes the BASIC interpreter to analyze the data from the I/O buffer and place the data in the random access memory (RAM). In all cases, the buffer overhead depends on the quantity of data in the buffer and the format of that data; this time varies in each case. If the data stream contains more than 72 characters, then the data bursts and buffer overhead time periods will be intermixed. This must be taken into account when calculating the data rate for a particular transfer.

STATEMENT TERMINATION TIME PERIOD

This time period occurs at the end of every data transfer on the GPIB. During this time, the BASIC interpreter checks to ensure that all the specified data has been output on output operations and verifies that enough data and the right kind of data has been received on input operations.

UNADDRESSING SEQUENCE TIME PERIOD

After the BASIC interpreter is sure that the conditions of the I/O operation have been satisfied, the interpreter activates ATN, issues the universal commands UNTALK and UNLISTEN, then releases ATN. This clears the bus and returns every peripheral device to an idle state. After this sixth and last event, the 4051 program line counter advances to the next line number in memory and program execution continues from that point.

The rest of this section examines each of these six events in detail for PRINT, INPUT, READ, WRITE, WBYTE, and RBYTE operations. Details are given on methods that can be used to compute the time for each event for a particular data format. Examples at the end of each statement show how data rates are calculated for typical data formats.

COMPUTING EFFECTIVE DATA RATES FOR PRINT

Introduction

The PRINT statement is used to transfer data in ASCII code format to a peripheral device over the GPIB. Only one peripheral device can receive the ASCII data string at a time (unless another device has been previously addressed as a listener with a WBYTE statement and is still on the bus.)

Any sequence of ASCII characters (decimal 0 through 127) can be transferred over the bus; and, for all practical purposes, the length of the output data string is unlimited if a PRINT USING format string is specified. (Refer to the PRINT statement and the IMAGE statement in the 4051 Graphic System Reference Manual for details.)

Because of this versatility and infinite variation, the PRINT statement overhead and setup time is practically impossible to calculate. The set-up time depends on how the parameter list is constructed; a slight variation in the statement (like replacing a semicolon with a comma, for example) may cause a significant variation in the setup time.

When a PRINT statement is executed, the BASIC interpreter prepares 72 characters at a time for transfer. The data in the parameter list is formatted into an ASCII character string according to the guidelines set forth in the IMAGE format string or the default print format, which ever is specified. As soon as the I/O buffer receives 72 characters, the 4051 stops the formatting operation and transmits the data as fast as it can (7936.5 bytes/sec. maximum). After the 72nd byte is transferred, the 4051 resumes the formatting operation until the I/O buffer is full again. This procedure continues until all of the data in the PRINT parameter list is transferred.

Since the IMAGE format string determines the number of characters that are added and deleted from the data string at any one time, the setup time (time between data bursts) is different in each case. Generally, this time is close to 15 ms, but may be as long as one second.

Computing Data Sample Rates For 72 Characters or Less

Since the first 72 character data burst in a PRINT statement has an upper limit of 7936.5 bytes/second for a data rate, the maximum data sample rate is computed by dividing the number of characters in the data sample into 7936.5. The following examples illustrate how to compute the data sampling rate for PRINT.

Example 1

```
PRINT @2: 1;2;3;4;5;6;7;8;9
```

Knowing that the default PRINT format inserts a space before each number in the ASCII string, each number (or sample) contains two characters. The data sample transfer rate is therefore 7936.5/2 = 3968.5 samples/second.

Example 2

```
PRINT @2: 1,2,3,4,5,6,7,8,9
```

Knowing that the comma delimiter in the parameter list calls for a TAB, the default PRINT format places each data sample in a 18 character field. Space characters are used to fill in the fulfilled slots in the field. With 18 characters per sample, the data sample transfer rate is 7936.5/18 = 440.9 samples/second.

It can be seen from these two examples that a slight difference in the PRINT parameter list can make a big difference in the data sample rate; a detailed knowledge of the PRINT format is necessary when it comes to computing the PRINT data sample rate.

COMPUTING EFFECTIVE DATA RATES FOR INPUT

Introduction

The INPUT statement is used to transfer data in ASCII format from a peripheral device on the GPIB to the 4051 random access memory. Only one peripheral device can take part in the transfer at any one time. Since most peripheral devices on the market today transfer data in ASCII format, the INPUT statement is the most commonly used statement to input data. The data does not have to be formatted in ASCII code, but the 4051 treats the data as ASCII code by stripping off the eighth bit in each byte, then converts each byte to its ASCII character equivalent. The data is treated as a character string or numeric data, depending on the target variable specified in the INPUT statement. If a string variable is specified, for example, then the incoming data bytes up to the first CR (or alternate delimiter, if one is specified) are treated as a character string. If a numeric variable, an array variable, or a subscripted array variable is specified, then the BASIC interpreter searches through the incoming data stream for valid numeric data items; when numeric data is found, the interpreter assigns the data to the specified numeric variable. Numeric data must be separated by one or more non-numeric characters such as spaces, commas, etc.

Advantages of Using the INPUT Statement

1. The data format (ASCII code) conforms to the format of many peripheral devices on the market.

2. The numeric format can be a "free" format; that is, any sequence of digits 0 through 9. An embedded decimal point and scientific format (E notation) are allowed. (Refer to page C-17 in the 4051 Reference Manual for details.)

3. When numeric variables are specified in an INPUT statement, all non-valid numeric characters, such as NULL characters or extraneous alphanumerics, are automatically thrown out.

Disadvantages of Using the INPUT Statement

1. It's a slower form of input. Since a great deal of freedom is given to the data format, the BASIC interpreter must constantly scan the data stream for valid data and valid delimiters; this slows down the transfer rate.

2. The peripheral device must activate EOI, send a valid delimiter, such as CR, or an alternate delimiter as specified in a PRINT @37,0: statement. (See page 2-25 in the 4051 Reference Manual.) If a valid delimiter is not sent to the 4051, then the 4051 continues to address the peripheral device asking for more data, even if all the target variables in the INPUT statement have assigned values. This continues until a valid delimiter is sent, or until the BREAK key on the 4051 keyboard is pressed twice and program exeuction is aborted.

Procedure for Computing INPUT Date Rates

INPUT data rates are the most difficult to compute. The data rate depends on many factors and is different in each application. Basically, the time it takes to complete an INPUT operation is computed by determining the INPUT statement overhead time period, the addressing period, the data burst periods, the buffer overhead period for each data burst, the statement termination period, and the unaddressing period. These time increments are then added together to arrive at a total time for the I/O operation. After this time has been determined, the data rate is computed by dividing the number of data bytes transferred by the total time taken to complete the operation. The data sampling rate can be determined by dividing the number of data samples transferred into the total time.

Computing the Statement Overhead for INPUT

The statement overhead is a function of the statement format. First determine what the format of the I/O address is, then look up the time for that format in Table 5-2. Next, examine the paramter list and determine how many numeric variables, array variables, subscripted array variables, and string variables are listed. Add .8 milliseconds for each numeric variable, array variable, and string variable. Add 5.4 millisconds for each subscripted array variable. Add all these time increments together to determine the statement overhead period for that particular statement; this specifies the delay from when the BASIC interpreter first looks at the BASIC statement to the time when the first activity occurs on the GPIB.

Table 5-2

COMPUTING THE STATEMENT OVERHEAD FOR INPUT

STATEMENT	T_{format}	COMMENTS	TIME INCREMENTS FOR TARGET VARIABLES		
INPUT @2:	5.8 ms		$T_{numeric}$	=	.8 ms
INPUT @P:	5.2 ms	P = 2			
INPUT @2,13:	7.8 ms		T_{array}	=	.8 ms
INPUT @2,S:	7.2 ms	S ≠ 32			
INPUT @P,13:	7.0 ms	P = 2	T_{string}	=	.8 ms
INPUT @P,S:	6.6 ms	S ≠ 32			
INPUT @2,32:	7.5 ms		$T_{sub\ array}$	=	5.4 ms
INPUT @P,32:	7.0 ms	P = 2			
INPUT @P,S:	6.6 ms	S = 32			
INPUT @2,S:	7.2 ms	S = 32			

The Formula

Statement Overhead = $T_{format} + n \times T_{numeric} + n \times T_{array} + n \times T_{string} + n \times T_{sub\ array}$

where n = number of variables in each category

Example 1

```
DIM A (100)
INPUT @2:A,B$,C,D,E(4,3)
```

Statement Overhead = 5.8 ms + 2 x .8 ms + .8 ms + .8 ms + 5.4 ms = 14.4 ms.

Example 2

```
INPUT @P,S: A$,B$,C,D,E
```

Statement Overhead = 6.6 ms + 5 x .8 ms = 10.6 ms

The Addressing Period for INPUT

Calculating the addressing time period is simple. If a primary address and/or a secondary address other than 32 is specified in the INPUT statement, then the BASIC interpreter issues one primary address and one secondary address. This takes 308 μs (minimum). If the secondary address 32 is specified, then the secondary address is suppressed, and the BASIC interpreter issues only the primary address. This takes 224 μs (minimum).

Computing the Data Burst Periods for INPUT

After the peripheral device is addressed, the 4051 inputs data bytes until a delimiter is received, or until 72 data bytes are received, whichever occurs first. Knowledge of the peripheral devices data format is important at this point. If, for example, the peripheral device sends a five digit data sample with a CR delimiter, then the data burst will last for six bytes. If on the other hand, the peripheral device sends a sign byte followed by 10 digits with an embedded decimal point, followed by four bytes of status information, followed by a CR/LF delimiter, then the data burst will last for 18 bytes. (The 4051 delimits on the CR as a default, but inputs the LF as the first character in the next data burst.) Once the data format is known, the total time for the data burst is calculated as follows:

In @ Mode: T_{burst} = Number of Bytes X 244 μs

In % Mode: T_{burst} = Number of Bytes X 251 μs

If the I/O address in the INPUT statement begins with an @ sign, then CR is used as the data item delimiter. In this case, it takes a minimum of 244 μs to transfer a data byte into the 4051 I/O buffer.

If the I/O address starts with a % sign, then an alternate delimiter is used, as specified in a previous PRINT @37,0: statement. It takes the BASIC interpreter longer to check the data stream for the alternate delimiter, so each data byte is transferred in 251 μs (minimum).

Remember, the above figures assume that the peripheral device is driving the 4051 at a maximum speed. If the peripheral device slows down, then the data burst will take longer.

Computing the Buffer Overhead for INPUT

After the 4051 has detected a valid delimiter in the data stream or has input 72 data bytes, the 4051 stops, converts the data to internal format, then stores the data in the read/write random access memory. The time it takes to do this depends on the number of samples in the buffer and the format of each sample. Generally, the buffer overhead takes from 3 ms to 12 ms. During this period, all activity on the GPIB ceases. If large amounts of data are transferred, many data bursts will occur, followed by periods of inactivity as the I/O buffer is dumped. To calculate the total transfer time, these time increments should be added together.

After the last data burst, the BASIC interpreter empties the I/O buffer, then checks to make sure that all the target variables in the INPUT statement have assigned values. This is called the statement termination time period. Depending on the statement it may take the BASIC interpreter a longer time or a shorter time to execute this operation than the buffer overhead time period.

Buffer Overhead for Numeric Data. If a numeric variable, array variable, or subscripted array variable is specified in an INPUT statement, the buffer overhead is a function of the number of data samples in the I/O buffer at the time of analysis, and the number of digits in each sample. Roughly speaking, the time to convert a 1 digit sample from ASCII format to internal floating-point format is 3.6 milliseconds. For each addition digit in the sample, it takes an extra .3 ms. The total time to dump the whole I/O buffer can be calculated by converting each sample, then adding the subtotals for an overall total.

EXAMPLE 1: INPUT @2: A,B,C,D,E(3)

When the above statement is executed, device 2 sends the following data:

25000,50,125CR82

In this example, the digits up to the first CR are input into the 4051 I/O buffer. The 4051 then assigns the first data item 25000 to A, the second data item 50 to B, and the third data item 125 to C. Next, the 4051 returns to the GPIB, inputs the fourth data item 82, then terminates the I/O operation.

Since 13 characters are included in the first three samples (CR included) the first data burst lasts for 13 X 244 μs = 3172 μs. The 4051 then converts and stores the contents of the I/O buffer. It takes (3.6 + 5 x .3) ms = 5.1 ms to convert the first sample, (3.6 + 2 x .3) ms = 4.2 ms to convert the second sample, and (3.6 + 3 x .3) ms = 4.5 ms to convert the third sample. The total time to empty the I/O buffer and return to the bus is 5.1 ms + 4.2 ms + 4.5 ms = 13.8 ms. Since the data value 82 is input as the last sample, the conversion time is mixed in with the statement termination time and cannot be measure accurately.

EXAMPLE 2:
```
DIM A(1000)
INPUT @2:A
```

When the above program is executed, device 2 sends five digit data samples with a CR delimiter after each sample. The data burst is intermixed with periods of maximum activity followed by periods of total inactivity. How long are the periods of maximum activity and how long are the periods of inactivity?

ANSWER: Since each data sample is followed by a CR, the 4051 stops after each CR is received and processes the data sample in the I/O buffer. Each sample is five digits plus a CR, so six characters are transferred in each burst. The handshake of each character takes 244 μs, so the total time for each burst of data is 1464 μs.

The gaps in the bus activity occur when the 4051 stops to convert each sample. Since each sample contains five digits, the conversion time is 3.6 + 5 x .3 ms = 5.1 ms. Therefore, each buffer overhead gap in the data burst is approximately 5.1 ms.

EXAMPLE 3:
```
DIM A(1000)
INPUT @2:A
```

When the above program is executed, device 2 sends eight digit-data samples with a comma delimiter after each sample. During the data burst, how long is each period of maximum activity, and how long is each period of total inactivity (buffer overhead)?

ANSWER: Since comma delimiters are used between each data sample, the 4051 inputs 72 characters before it stops to process the contents of the I/O buffer. Each sustained data burst lasts 244 μs x 72 = 17.568 ms.

Each sample contains nine characters. This means that eight samples are in the buffer when the 4051 starts processing. It takes 3.6 ms + (8 x .3) ms = 6 ms to process each sample. Therefore, eight samples x 6 ms = 48 ms to dump the buffer and get back to the bus.

CONCLUSION: The data burst consists of a 17.568 ms sustained burst, followed by a 48 ms period of bus inactivity, followed by a 17.568 ms burst, etc., until 1000 data samples are input.

Buffer Overhead for Character Strings. The buffer overhead for processing a character string in the I/O buffer is a function of the number of characters in the buffer. If a CR is the only character in the buffer, it takes 1.5 ms. Add .1 milliseconds for each additional character in the buffer.

EXAMPLE 1:
```
INPUT @2: A$,B$,C$,D$
```

When the above statement is executed, device 2 sends four data samples to the 4051 via the GPIB. The samples are assigned to A$,B$,C$, and D$ respectively. Each sample contains 35 characters and is terminated with a CR. How long does it take to transfer each character string and what is the delay between the transmission of each string?

ANSWER: Since each character string contains 35 characters, it takes 36 x 244 μs = 8.78 ms to transfer each string. After a character string is in the 4051 I/O buffer, it takes 1.5 ms + 35 x .1 ms = 5 ms to take the string out of the buffer and place the string in the RAM memory.

EXAMPLE 2:
```
PRINT @37,0: 4,255,13
DIM A$(20000)
INPUT %2: A$
```

When the above program is executed, device 2 sends a 20,000 byte character string over the GPIB. What does the data burst look like on the bus during the transfer?

ANSWER: In this operation, the 4051 inputs 72 characters, dumps the I/O buffer, inputs 72 more characters, dumps the I/O buffer, and so on, until the EOT (End of Transmission character—decimal 4) is received from device 2. Since CR (decimal 13) is specified in the PRINT @37,0: statement as the character to be deleted from the data stream, all CR delimiters are ignored when the % mode INPUT operation is executed.

The data bursts last for 72 x 251 μs = 18.072 ms each. And, it takes 1.5 ms + (72 x .1 ms) = 8.7 ms to dump the I/O buffer each time it fills.

Statement Termination Period

The statement termination period starts when the last character is placed into the I/O buffer (the last time DAV goes high) and ends when the 4051 sets ATN low to start the unaddressing period. The statement termination period includes (1) the time it takes to dump the remaining contents of the I/O buffer and (2) the time it takes the 4051 to prepare to issue UNTALK and UNLISTEN. Table 5-3 lists the statement termination periods for typical INPUT statements.

Table 5-3

TERMINATION PERIODS FOR TYPICAL INPUT STATEMENTS

STATEMENT	TERMINATION PERIOD
INPUT @2: A	3050 μs
DIM A(1000) INPUT @2: A	3150 μs
INPUT @2: A(1)	3000 μs
INPUT @2: A$	760 μs

Statement Unaddressing Period

It takes the 4051 approximately 380 μs to terminate an INPUT operation each time. This is the time it takes the 4051 to activate ATN, issue UNTALK and UNLISTEN, then release ATN.

Putting It All Together

The effective data rates for INPUT are computed by dividing the total elapsed time for the INPUT operation by the number of data bytes transferred, or the number of data samples transferred, whichever is appropriate. In some cases, it may not be appropriate to include the statement overhead as part of the overall transfer period. In these cases, add the addressing period, the data bursts with their associated buffer overhead periods, the statement termination period, and the unaddressing period, then divide by the number of data bytes or data samples transferred. The following examples illustrate this procedure:

EXAMPLE 1 INPUT @2: A,B,C,D$,E,F(9)

Device 2 sends the following data string:

55,890,763977264,HI MOM CR44CR

Statement Overhead = 5.8 ms + 5 x .8 ms + 5.4 ms = 15.2 ms

Addressing Period = .380 ms

First Data Burst = 25 x 244 μs = 6.1 ms

First Buffer Overhead Period

First Data Sample = 3.6 ms + 2 x .3 ms = 4.2 ms
Second Data Sample = 3.6 ms + 3 x .3 ms = 4.5 ms
Third Data Sample = 3.6 ms + 9 x .3 ms = 6.3 ms
Fourth Data Sample = 1.5 ms + 7 x .1 ms = 2.2 ms

Total 17.2 ms

Second Data Burst = 3 x 244 μs = .732 ms

Statement Termination = 3.05 ms

Unaddressing Period = .380 ms

Total Time Including Statement Overhead = 43.042 ms
Total Time Excluding Statement Overhead = 27.842 ms
Effective Data Byte Rate for the Entire Operation = 1/(43.042E-3/28) = 650.53 bytes/sec
Effective Data Byte Rate (excluding statement overhead) = 1/(27.842E-3/28) = 1005.67 Bytes/sec

EXAMPLE 2:
```
DIM X(1000)
INPUT @4,32: X
```

Device 4 sends 1000 data samples with five digits in each sample. CR delimiters separate each sample.

Statement Overhead = 7.5 ms + .8 ms = 8.3 ms

Addressing Period = .224 ms

Each Data Burst = 6x244E-6 = 1.46 ms

Buffer Overhead for each sample = 3.6 ms + 5 x .3 ms = 5.1 ms

Statement Termination = 3.05 ms

Unaddressing Period = .380 ms

Total Time including statement overhead = 8.3 ms + .224 ms + 1000 x 1.46 ms + 999 x 5.1 ms + 3.05 ms + .380 ms = 6.57 seconds

Effective Data Byte Transfer Rate = 1/(6.57/6000) = 913.24 bytes/sec

Effective Data Sample Transfer Rate = 1/(6.57/1000) = 152.2 samples/sec

COMPUTING EFFECTIVE DATA RATES FOR WRITE

Introduction

The WRITE statement is used to transfer data to a peripheral device in 4051 internal binary code. Each data item is preceded by a two byte header which identifies the data item type (numeric or string) and the length of the item (in bytes). (Refer to the Appendix for details.) The length of a numeric data item is always eight bytes plus the header. The length of a character string is one byte per character plus the header.

Since each numeric data item is formatted in 4051 internal binary floating point notation, it is of little practical use to send this information to a peripheral device other than a storage device. Most peripheral devices cannot interpret this format. Character strings, however, are different. After the two byte header, the bytes representing the data are the same as ASCII code. For example, A is represented by decimal 65, B is represented by decimal 66, etc. A peripheral device that can interpret ASCII code could, for example, throw away the first two bytes of the READ string (the header) and correctly interpret the rest of the string. This would be advantageous when transmitting a character string with more than 72 characters, because the gap (or setup time) found in the PRINT statement is virtually eliminated. Therefore, the 7936.5 bytes/sec burst rate can be sustained over the entire I/O operation.

Advantages of Using the WRITE Statement

1. For storing numbers, the data is often more compact and the space required is predictable.

2. ASCII character strings greater than 72 characters can be transferred faster, because the 7936.5 bytes/sec burst rate is sustained throughout the transfer.

Disadvantages of Using the WRITE Statement

1. Most peripheral devices cannot interpret the 4051 internal floating-point notation for numeric data. Therefore, the only practical use for transferring numeric data with WRITE is to store the data on tape or a suitable storage media, then bring it back into the 4051 at a later time, when it is needed.

2. Character string data can be interpreted as ASCII data, but the peripheral device must be programmed to ingore the first two bytes of the string (the header).

Procedure for Computing WRITE Data Rates

The Addressing Period. The addressing period for WRITE is the same as any other I/O statement. It takes 380 μs to issue a primary address and a secondary address. If 32 is specified as the secondary address, then only the primary address is issued. This takes 224 μs.

The Data Burst for Numeric Data. The handshake cycle during different phases of the burst is different. Therefore, to state a burst rate is misleading. The shortest transfer period for a 10-byte numeric data item is 1998 μs (5005 bytes/sec). This places the maximum output data sample rate at 500.5 samples/second.

The Data Burst for Character String Data. The burst rate for the header part of a character string is slower than the main part of the string; a termination time period must also be considered.

The burst rate during the main part of the string is 7936.5 bytes/second. On long strings, the burst rate will be slightly less than 7936.5 bytes/second.

If each character string is treated as a data sample, then the data sample rate is computed as follows:

$$\text{Data Sample Rate} = 1/(1236 + (\text{(No. of Characters - 1)} \times 126)) \text{ E-6}$$

EXAMPLE 1:

A series of character strings are output with a WRITE statement. Each string is 10 characters in length. What is the data sample rate?

ANSWER: Data Sample Rate = 1/(1236 + ((10-1) x 126)) E-6
Data Sample Rate = 1/(1236 + 1134) E-6
Data Sample Rate = 1/2370 E-6
Data Sample Rate = 421.9 samples/second

EXAMPLE 2:

A series of character strings are output with a WRITE statement. Each string is 72 characters in length. What is the data sample rate?

ANSWER: Data Sample Rate = 1/(1236 + ((72-1) x 126)) E-6
Data Sample Rate = 1/(1236 + 8946) E-6
Data Sample Rate = 1/10182E-6
Data Sample Rate = 98.2 samples/second

Note that if a series of data values are stored in the 4051 memory as one long character string, then the data burst rate is approximately 7936.5 bytes/second throughout the entire transfer. Therefore, the data sample rate is equal to 7936.5 divided by the number of characters in each sample (include the delimiter).

EXAMPLE 3:

A series of data samples are stored in the 4051 memory as one long character string. Each sample is five characters in length with a linefeed delimiter (i.e., A$="12345J12345J12345J12345J12356J"). The data is transferred to a peripheral device which throws out the first two bytes of the string (the header), then reads each sample as though it is a five digit ASCII number with an LF delimiter. What is the data sample rate over the GPIB when the character string is being transferred?

ANSWER: Data Sample Rate = 7936.5/6 = 1322.8 samples/second.

Computing the Unaddressing Time Period. The unaddressing sequence for the WRITE statement is the same in all cases—380 μs.

COMPUTING EFFECTIVE DATA RATES FOR READ

Introduction

The READ statement is generally used to bring back 4051 internal binary format data from a storage device. Getting non-storage peripheral devices to format numeric data into 4051 internal floating-point notation is a complicated task and generally not practical in most applications.

Transferring character strings to the 4051 with a READ statement can be done, however, since the 4051 stores character strings internally in ASCII code. The 4051 will accept an incoming ASCII string if it is preceded by the proper 2-byte header. Therefore, if a peripheral device can generate a two-byte header, an ASCII character string can be transferred to the 4051 with the READ statement at a high effective data rate.

Advantages of Using READ Over INPUT

The advantages in using READ instead of INPUT to transfer an ASCII character string are as follows:

1. READ is faster. If the header is set up to say "character string—8000 bytes long," the 4051 will input 8000 ASCII characters from a peripheral device in a sustained burst at 5263 bytes/sec. The INPUT statement accomplishes the same task at an effective burst rate of about 2000 bytes/sec. The reason is that the INPUT burst is intermixed with buffer overhead gaps of up to 12 ms, where the READ burst doesn't.

2. Delimiters in the READ data stream do not stop the input burst during a READ operation. When an INPUT statement is executed, the 4051 stops after every delimiter (CR) is transmitted and processes the contents of the I/O buffer. In a READ operation, the 4051 doesn't look at the data stream. The header tells the 4051 how many data bytes to input, so the specified amount are input without stopping.

Disadvantages of Using READ Over INPUT

When using READ to input ASCII numeric data, the data all goes into a large character string in memory and can't be used in math operations until the data is converted with the VAL function. This conversion can take up to 60 seconds or more after the data is in memory. So, using READ to input ASCII data is only practical in applications where the data is perishable and must be transferred to the 4051 as fast as possible. But, once the data is in memory (or stored on the internal tape), the time it takes to process the data must not be critical.

Computing Sample Data Rates For READ

Discussing burst rates for the READ statement is not relevant here, because the handshake timing differs throughout the data sample. The data sample transfer rate, however, is computed as the following topics illustrate.

Transferring Numeric Data (Two byte header + eight bytes floating point)

The maximum rate is fixed at 350 samples/second.

Transferring Character Strings

The maximum rate is determined by the number of characters in each string.

Data Sample Rate = 1/(1600 + ((No. of Characters-1) x 126)) E-6

EXAMPLE 1:

A digital voltmeter is connected to the 4051 GPIB. The voltmeter interface is designed to output a two-byte header before each sample reading. Each reading contains five ASCII digits, followed by a decimal point, followed by two more ASCII digits, plus a CR. What is the maximum data sample rate that can be achieved over the GPIB?

ANSWER: From the above information, each data sample (voltmeter reading) contains a two-byte header followed by eight ASCII characters. Assuming that the voltmeter is faster than the 4051 during all phases of the transfer, the maximum data sampling rate is computed as follows:

Data Sample Rate = 1/(1600 + ((8-1) x 126)) E-6
Data Sample Rate = 1/(1600 + 882) E-6
Data Sample Rate = 1/2482 E-6
Data Sample Rate = 402.9 samples/second

EXAMPLE 2:

A system designer connects a peripheral device to the 4051 GPIB which scans a field of data, then transmits 1000 readings to the 4051. The readings are taken in parallel, then transmitted in digit serial format, one after another, over the GPIB. Each reading contains 3 ASCII digits with no delimiter. Since the data is perishable, the system designer chooses to transmit two header bytes to the 4051 which says "ASCII character string—3000 bytes long." All the data samples are then input into the 4051 memory at maximum speed and assigned to one string variable with a READ @2:A$ statement.

After the data is in the memory, the designer uses the BASIC string functions to separate and convert each sample to a numeric value and assign the value to an array element.

Assuming that the peripheral device is faster than the 4051 during all phases of the data transfer, how long does it take to input the 1000 samples into the 4051 memory?

ANSWER: Since all 1000 samples are transferred as one ASCII string, the answer is found by computing the total transfer time for a 3000 byte character string. The time to transfer each sample is then found by dividing the total time by 3000; the reciprocal of the result is the answer.

Total Time to Transfer the ASCII String = 1600 μs + ((No. of Characters-1) x 126) μs
= 1600 μs + ((3000-1) x 126) μs
= 1600 μs + 377874 μs
= 379474 μs
= .379 seconds

Data Sample Transfer Rate = 1/(379474E-6/1000)

Data Sample Transfer Rate = 1/3.79474E-4

Data Sample Transfer Rate = 2635.23 samples/second.

COMPUTING EFFECTIVE DATA RATES FOR WBYTE

Introduction

The WBYTE statement is used to issue a series of peripheral addresses, controller commands, and special eight-bit data bytes to peripheral devices on the GPIB. Normally the WBYTE statement is used only if the facilities in the PRINT statement and the WRITE statement cannot be used to setup a transfer. For example, if two or more peripheral devices must listen to the same data transfer from the 4051, then the WBYTE statement must be used to set it up; if one peripheral device on the bus needs to talk to another peripheral device on the bus, then the WBYTE statement must be used to set it up; and, if a peripheral device needs a special series of binary bit patterns in order to work, the WBYTE statement can be used to issue those bit patterns to the device.

As a rule of thumb, if the PRINT and WRITE statements cannot be used to get the job done, then use WBYTE. Although WBYTE provides maximum flexibility through very primitive operations, the trade off for this flexibility is a slower data rate.

Advantages in Using WBYTE

Any combination of peripheral addresses and controller commands can be issued over the GPIB, along with any series of eight-bit binary numbers (bytes). ATN, or EOI, can be active as a byte, or series of bytes, are transferred.

Disadvantages in Using WBYTE

1. There isn't automatic addressing and unaddressing as there is in the PRINT and WRITE.

2. Each data byte (or character) must be specified as the decimal equivalent of the binary bit pattern.

3. The byte transfer rate is considerably less than the rates for PRINT and WRITE.

Computing the Statement Overhead for WBYTE

The statement overhead for WBYTE is the time from when the 4051 starts working on the statement until the time when the first activity occurs on the GPIB. Generally, the statement overhead for WBYTE is less than the overhead for PRINT and WRITE, because part of the overhead is carried out after the transfer begins. The conversion of each decimal value to a binary bit pattern is not carried out, however, until just before the value is placed on the Data Bus. This means that the data byte conversions occur during the handshake cycles all through the transfer. More on this in a minute.

The statement overhead for WBYTE generally falls in the range from 4 milliseconds to 6 milliseconds. Table 5-4 lists the statement overhead for several WBYTE statement formats. The following conclusions can be drawn from the data in Table 5-4.

1. The statement overhead is reduced if a decimal value is assigned to a numeric variable, then specified as a variable in the WBYTE statement.

2. Specifying one peripheral address in a WBYTE statement results in a statement overhead of 4 milliseconds. Each additional address adds approximately 1.25 milliseconds to the statement overhead.

3. If addresses are specified as variables, each additional address adds approximately .5 millisecond to the statement overhead.

4. Each numeric data byte which is specified after the colon (:) in a WBYTE statement adds approximately 1.3 millisconds to the statement overhead.

5. Each data byte which is specified as a variable after the colon (:) adds approximately .4 millisecond to the statement overhead.

Table 5-4

EXAMPLES OF WBYTE STATEMENT OVERHEAD PERIODS

Statement	Set-up Period
WBYTE @66:	4.0 ms
WBYTE @66,67:	5.3ms
WBYTE @66,67,68:	6.5 ms
WBYTE @66,67,68,69:	7.8 ms
WBYTE @P:	3.7 ms
WBYTE @P,Q:	4.2 ms
WBYTE @P,Q,R:	4.6 ms
WBYTE @67:100	5.3 ms
WBYTE @67:100,100	6.5 ms
WBYTE @66:100,100,100	7.8 ms
WBYTE @67:A	4.5 ms
WBYTE @67:A,B	4.9 ms
WBYTE @67:A,B,C	5.3 ms
WBYTE @66:A,B,C,D	5.8 ms

Computing the Data Burst Rate for WBYTE

Since the internal to external format conversions occur before every handshake during a WBYTE operation, every handshake cycle is different. Therefore, general statements cannot be made about the maximum data rate during a WBYTE data burst. Data sample rates fall in the same category because WBYTE transfers are primarily used to get the right information to the right device. WBYTE is not generally used to get data samples transferred at maximum speed.

To determine accurate rates for WBYTE, it is best to hook up the equipment in the system, use an oscilloscope, and observe a particular WBYTE statement in action. To give you a general idea about how fast WBYTE is, the statement WBYTE @34:A, (where A is an array of bytes) transfers bytes at a rate of 1190 bytes/second.

COMPUTING EFFECTIVE DATA RATES FOR RBYTE

Introduction

The RBYTE statement is used by the 4051 to receive individual data bytes one at a time over the GPIB. The advantage of using RBYTE to receive data bytes is that all eight data lines on the Data Bus can be used to transfer data. That is, a full eight-bit byte can be transferred; any byte from decimal 0 to decimal 255. This cannot be done with INPUT and READ.

Normally, the RBYTE statement is used in special applications such as receiving data from a 16-bit Analog to Digital (A/D) Converter. The first eight bits are transmitted as one byte and the second eight bits are transmitted as another byte. Specifying an array variable with 200 elements in an RBYTE statement, for example, allows 100 readings (200 bytes) to be input into the 4051 memory at 337.9 readings/second. Once the data bytes are in memory, the two data bytes for each reading can be mathematically combined to reflect the true nature of the reading, then processed according to the directions in the BASIC program.

The RBYTE statement can be used to transfer ASCII character strings a byte at a time, but this method is extremely slow (counting the conversion time) and it is much more practical to use the INPUT statement to transfer ASCII data.

Computing the Statement Overhead for RBYTE

Using the RBYTE statement requires that a previous WBYTE be used to address a peripheral device as a talker. This means that when an RBYTE statement is execute, a talker is already on the GPIB waiting to talk.

The only time periods in the RBYTE statement are the statement overhead and the Data Burst as the 4051 receives the data bytes. After a data byte is received for each target variable, the statement ends. There is no statement termination period or unaddressing sequence. The BASIC interpreter at this point continues to the next BASIC statement in memory which is generally a WBYTE @63,95: statement. This statement clears the GPIB of listeners and talkers.

The statement overhead period starts when the BASIC interpreter starts working on the RBYTE statement. During this time the interpreter examines each target variable in the RBYTE parameter lists and prepares to assign data bytes to these variables. The statement overhead period ends when the first activity on the GPIB begins.

The RBYTE statement overhead is computed as follows:

Statement Overhead = 2.9 ms + No. of Variables x .6 ms

Example 1:

```
RBYTE A,B,C
```

Statement Overhead = 2.9 ms + 3 x .6 ms = 4.7 ms

Example 2:

```
DIM A(100)
RBYTE A
```

Statement Overhead = 2.9 ms + 1 x .6 ms = 3.5 ms

Computing the Data Burst Rate for RBYTE

If only one target variable is specified in an RBYTE statement, the 4051 comes on the bus, handshakes with the talker, captures a data byte, then leaves the bus. This activity takes 100 μs (minimum).

If several numeric variables are specified in the RBYTE parameter list (RBYTE A,B,C for example) the 4051 stays on the bus handshaking until each variable has an assigned value. The data burst is uniform (no buffer overhead gaps) with a handshake cycle of 1.78 ms. This gives an effective input data rate of 561.8 bytes/second.

If an array variable is specified in the RBYTE parameter list, the 4051 keeps receiving data bytes until each element has an assigned value. The handshake cycle is a little faster (1.48 ms) so the effective input data rate is 675.7 bytes/second.

Since an unaddressing sequence doesn't occur at the end of an RBYTE statement, the statement is over as soon as the last data byte is received.

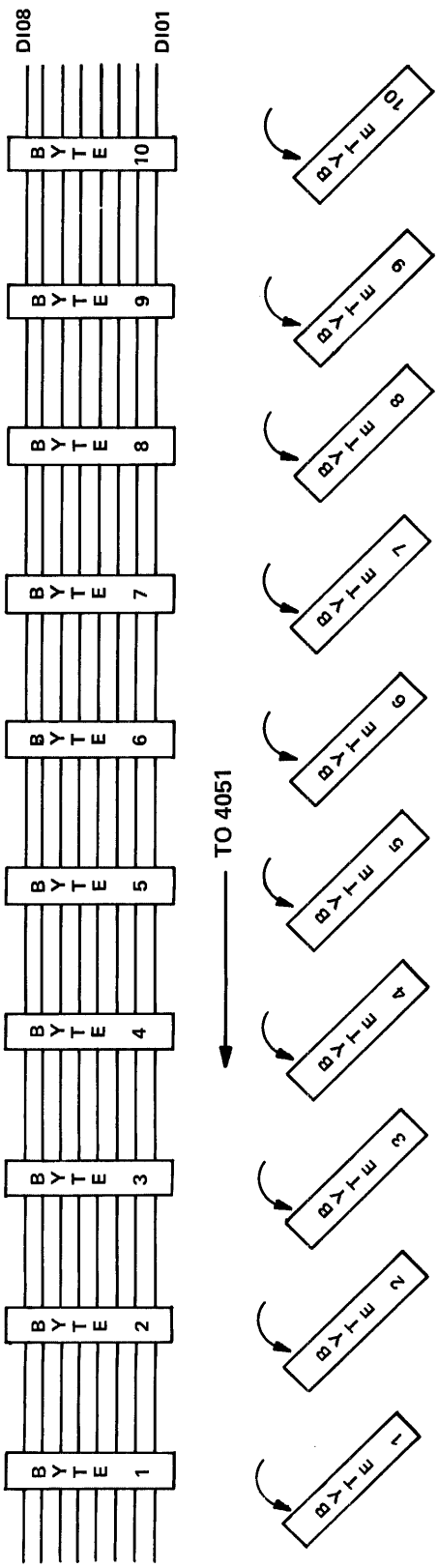
APPENDIX A

THE 4051 INTERNAL FLOATING-POINT FORMAT

INTRODUCTION

All numeric values that enter the 4051 Random Access Memory are stored in a special eight-byte floating-point format. INPUT operations from the keyboard, the internal magnetic tape, and the GPIB bring numbers into the memory formatted in ASCII code. These numbers must be converted to the special floating-point format before they are stored in memory. Likewise, all numbers leaving the memory during a PRINT operation are converted back to ASCII code before they are output to the specified peripheral device. Because the conversions between ASCII code and the internal floating-point format take time, the READ and WRITE facility was implemented in the 4051 BASIC language to transfer data directly in internal binary code. These two statements by-pass the ASCII conversion routines and thus increase the data transfer rate. The device communicating with the 4051 must be able to correctly interpret the data in the 4051 internal format or store the data received via the WRITE statement for later retrieval with the READ statement.

Fig. A-1 illustrates the format that must be used to send numeric values back and forth over the GPIB during a READ operation and WRITE operation. Each number contains ten bytes (two header bytes + eight bytes floating point). In the illustration, the ten bytes are being transferred from a peripheral device on the right to the 4051 on the left. Byte number 1 is leading the pack on the left. If these bytes are rotated counter-clockwise as shown in the illustration and stacked end-to-end, the floating-point format becomes clearly visible.



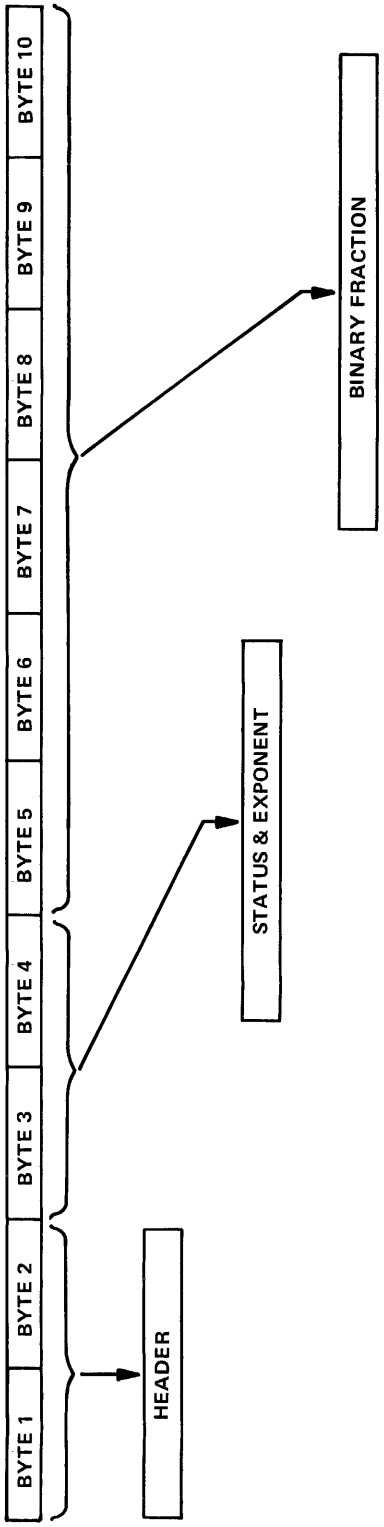
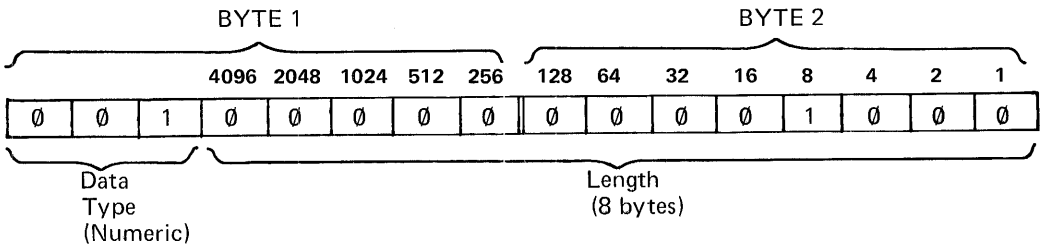


Fig. A-1. How the 4051 Internal Floating-Point Format is Transferred over the GPIB.

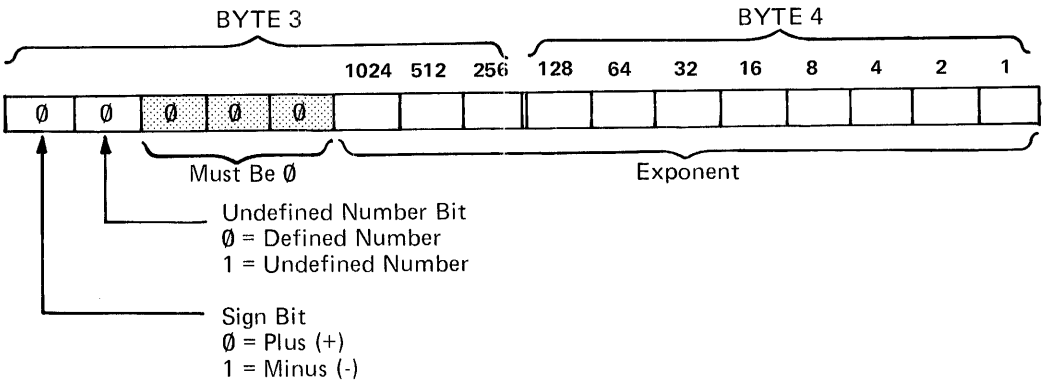
THE HEADER

The first two bytes contain the header information. The header tells the 4051 that an eight-byte floating point number is coming next. Since every floating point number is the same length (eight bytes), the header is always the same. The header format is shown below:



THE STATUS AND EXPONENT INFORMATION

The third byte and the fourth byte contain status and exponent information. The format for these two bytes are shown below:



The Sign Bit

The bit on the far left (bit 8 in byte 3) is the sign bit. This bit tells the 4051 whether the numeric value is positive or negative. If the bit is Ø, the number is positive. If the bit is 1, the number is negative.

The Undefined Number Bit

The second bit from the left (bit 7 in byte 3) is the undefined number bit. If this bit is set to 1, the 4051 assumes the number is undefined. The 4051 uses this bit in the following way. If a number is assigned to a numeric variable in memory (such as A=1) and a DELETE statement is

executed (such as DELETE A), the 4051 sets the undefined number bit to 1. When the 4051 needs more memory (at a later time), the eight bytes occupied by this undefined number are reclaimed automatically.

Three Bits must be set to Zero

Three bits in byte 3 must be set to zero. These bits are bit 6, bit 5, and bit 4.

The Exponent

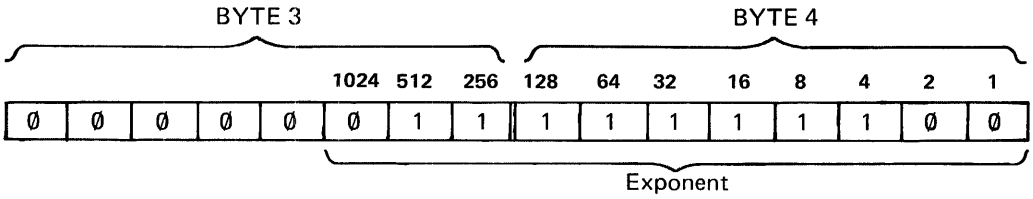
The three least significant bits in byte 3 and all the bits in byte 4 form a binary number that serves as the exponent. The exponent range for 4051 numbers is 2^{-1024} to 2^{1023}. To keep the exponent representation positive, 1024 is added to each exponent to make the range 0 - 2047. This means that if bit 3 in byte 3 (the most significant in the exponent) is set to 1, the exponent is Ø or positive. If bit 3 in byte three is set to Ø, the exponent is negative.

Example 1. Bit 3 in byte 3 and bit 2 in byte 4 are set to 1. The rest of the exponent bits are set to Ø. What is the true exponent of the floating-point number?

ANSWER: This binary bit pattern represents 1026. The true exponent is found by subtracting 1024 from this number. The exponent is 2 (i.e., 2^2).

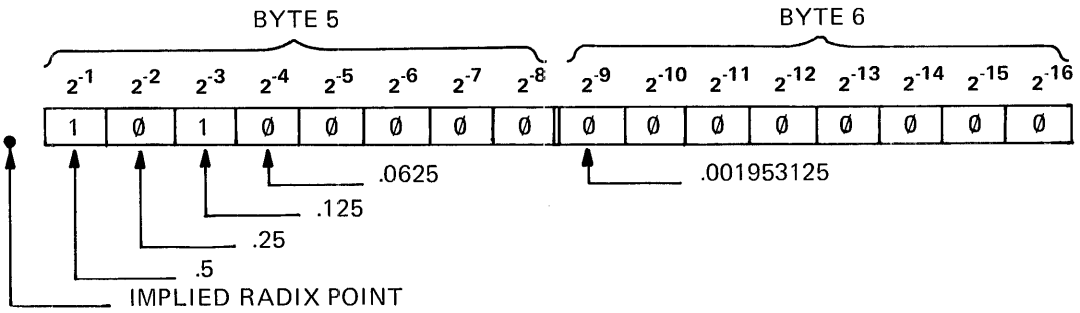
Example 2. The statement A=.0625 is entered into the 4051 from the keyboard. How is the exponent for the number .0625 stored in the 4051 memory?

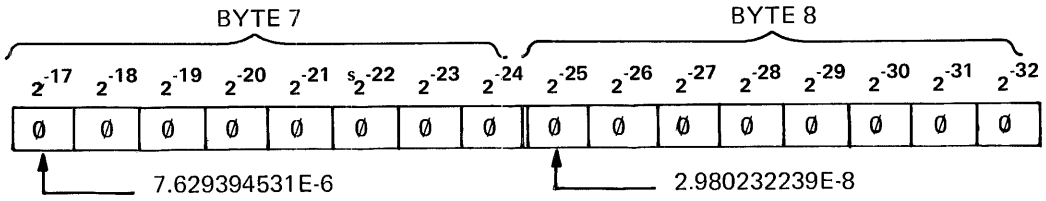
ANSWER: The number .0625 is equal to 2^{-4}. The exponent is −4, therefore the binary number representing the exponent must be 1024 − 4 = 1020. This bit pattern is shown below:

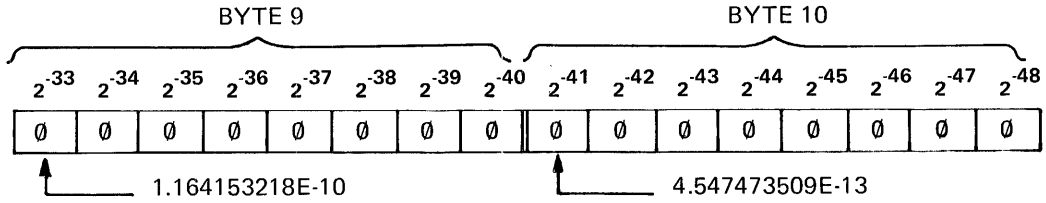


THE BINARY FRACTION

The remaining bytes in the floating point number form a binary matissa which allows numbers to be represented with 64 bits of precision. This representation is shown below:







If bit 6 and bit 8 in byte 5 are set to 1 as shown above, and the rest of the bits are Ø, the binary mantissa is equal to $2^{-1} + 2^{-3}$ which is equal to .625 in base 10. In another case, if bit 8 in byte 6 and bit 8 in byte 9 are set to 1 and the rest of the bits are zero, the binary mantissa is equal to $2^{-9} + 2^{-33}$. This number is equivalent to .00195312511642 (base 10).

It is apparent that this representation method allows numbers to be entered into the 4051 memory with a great deal of precision.

PUTTING IT ALL TOGETHER

The complete floating point representation is expressed in the following formula:

$$N = (-1)^s \times M_{10} \times 2^{(E-1024)}$$

where N = the decimal number entered into the 4051.
M_{10} = the decimal equivalent of the binary mantissa (last six bytes).
s = the sign bit (1 or Ø).
E = the decimal equivalent of the binary exponent.

A COMPLETE EXAMPLE

Fig. A-2 is a work sheet which is filled in to express the floating point equivalent of the decimal number 2184.00881958 (additional work sheets are provided for your convenience and may be removed from the manual). Here is how the floating point representation in Fig. A-2 is converted to a decimal number.

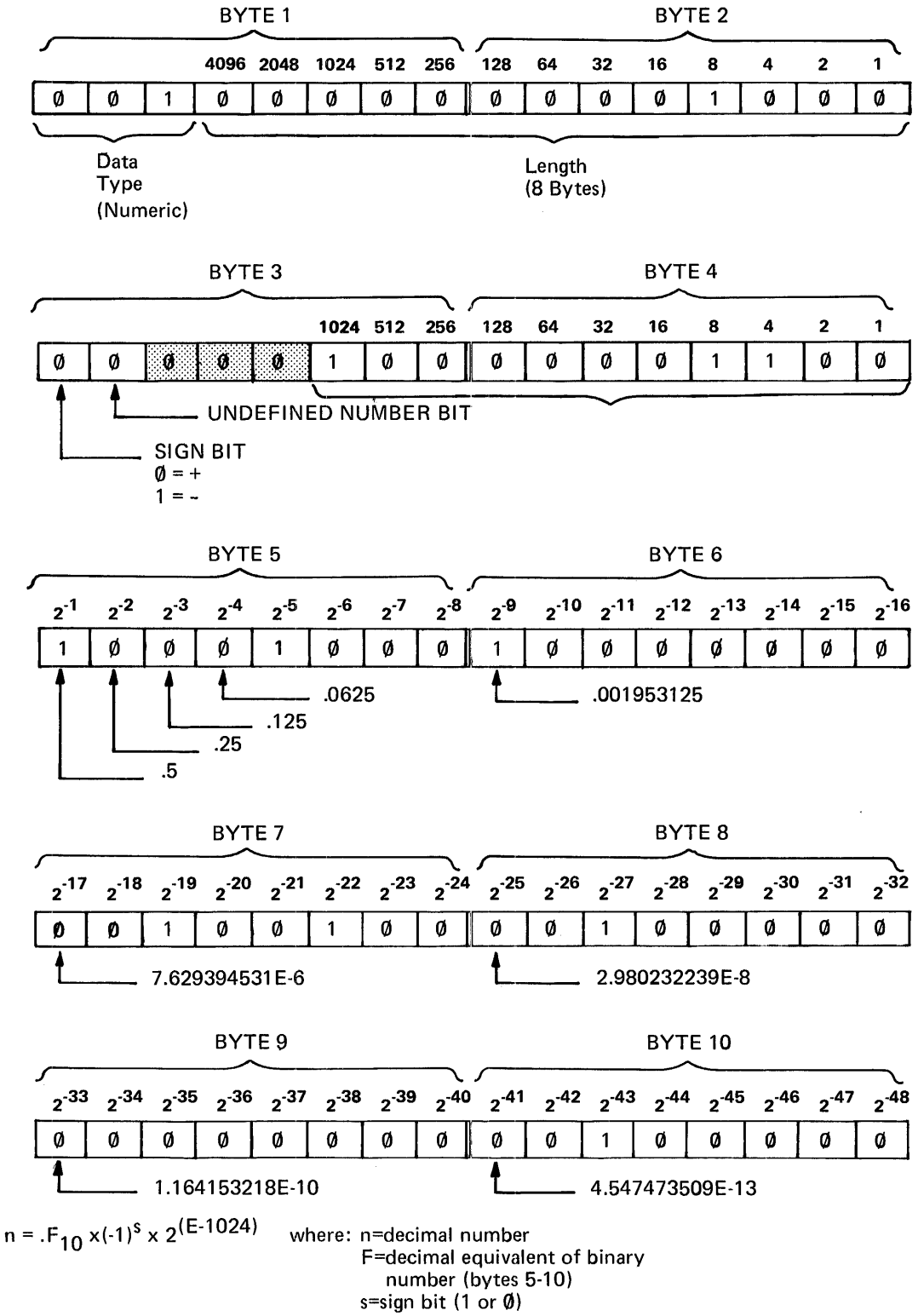
The sign bit S = Ø

The exponent $E = 10000001100_2 = 1036_{10}$

The mantissa $M_{10} = 2^{-1} + 2^{-5} + 2^{-9} + 2^{-19} + 2^{-22} + 2^{-27} + 2^{-43} = Ø.533205278218$

The decimal number $N = (-1)^0 \times Ø.533205278218 \times 2^{(1036-1024)}$
$N = Ø.533205278218 \times 2^{12}$
$N = Ø.533205278218 \times 4096$
$N = 2184.00881958$

4051 INTERNAL FLOATING-POINT WORKSHEET



$n = .F_{10} \times (-1)^{s} \times 2^{(E-1024)}$

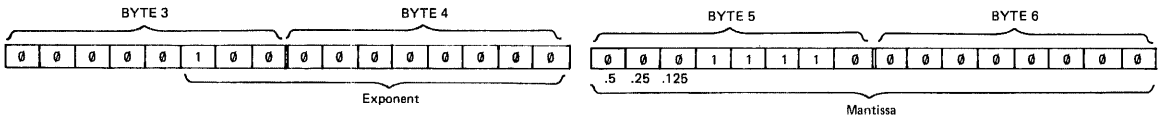
where: n=decimal number
F=decimal equivalent of binary number (bytes 5-10)
s=sign bit (1 or Ø)

Fig. A-2. An Example of a Filled-Out 4051 Floating-Point Worksheet.

FLOATING POINT NUMBERS COMING IN MUST BE NORMALIZED

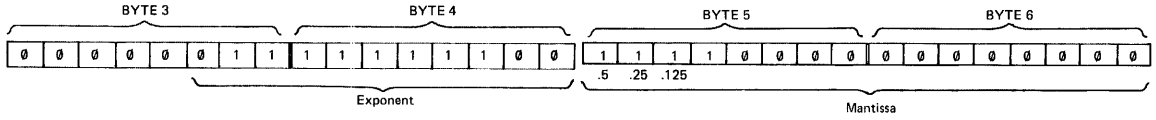
It is important that floating-point numbers coming into the 4051 from the GPIB be normalized; that is, the binary bits in the mantissa must be shifted left as far as possible and the difference made up in the exponent. The following examples illustrate the difference between a floating point number that is not normalized and a number that is normalized. Both numbers represent the same decimal value.

Example 1. A Floating Point Number that is not Normalized



Decimal Equivalent = .1171875

Example 2. The Same Floating Point Number After Normalization



Decimal Equivalent = .1171875

In the first example, the floating point number is not normalized because there are leading zeros in the mantissa. It is important to normalize this number before the number is placed in memory because the 4051 firmware math routines **assume** that all floating point numbers in memory are normalized. The decimal equivalent of this number is computed as follows:

The sign bit S = 0

The exponent E = 1024

The mantissa $M_{10} = 2^{-4} + 2^{-5} + 2^{-6} + 2^{-7} = .1171875$

The decimal number $N = (-1)^0 \times .1171875 \times 2^{(1024-1024)}$

$N = .117875$

This floating point number is normalized in Example 2 by shifting the mantissa bits three places to the left. To make up for the increase in the value of the mantissa, the exponent is decreased by 2^{-3}. To illustrate that the decimal value of this normalized number is the same, the decimal equivalent of the number is computed as follows:

The sign bit $S = 0$

The exponent $E = 1021$

The mantissa $M_{10} = .9375$

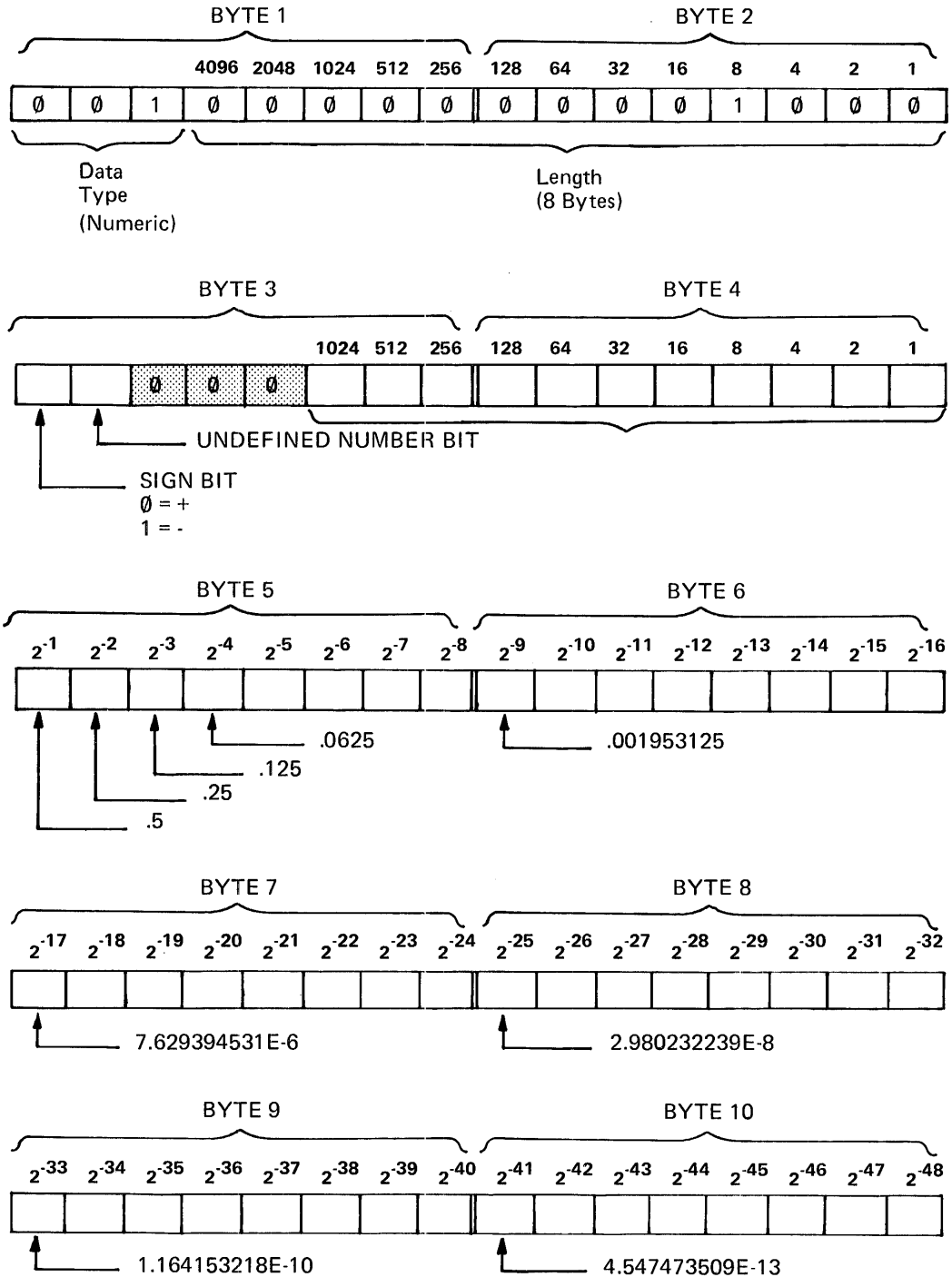
The decimal number $N = (-1)^0 \times .9375 \times 2^{(1021-1024)}$

$N = .9375 \times 2^{-3}$

$N = .117875$

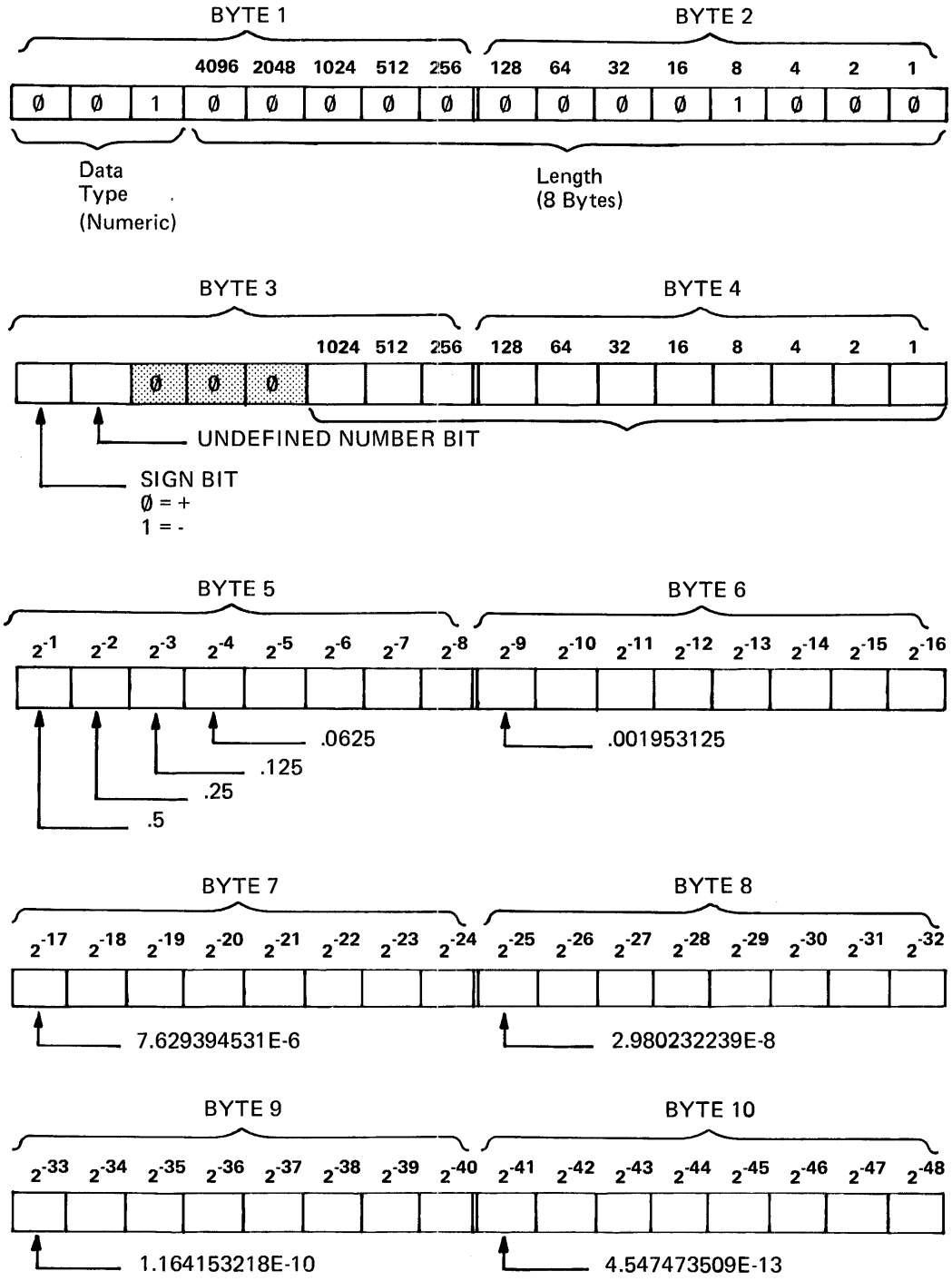
NOTES

4051 INTERNAL FLOATING-POINT WORKSHEET



$n = .F_{10} \times (-1)^{s} \times 2^{(E-1024)}$ where: n=decimal number
F=decimal equivalent of binary number (bytes 5-10)
s=sign bit (1 or Ø)

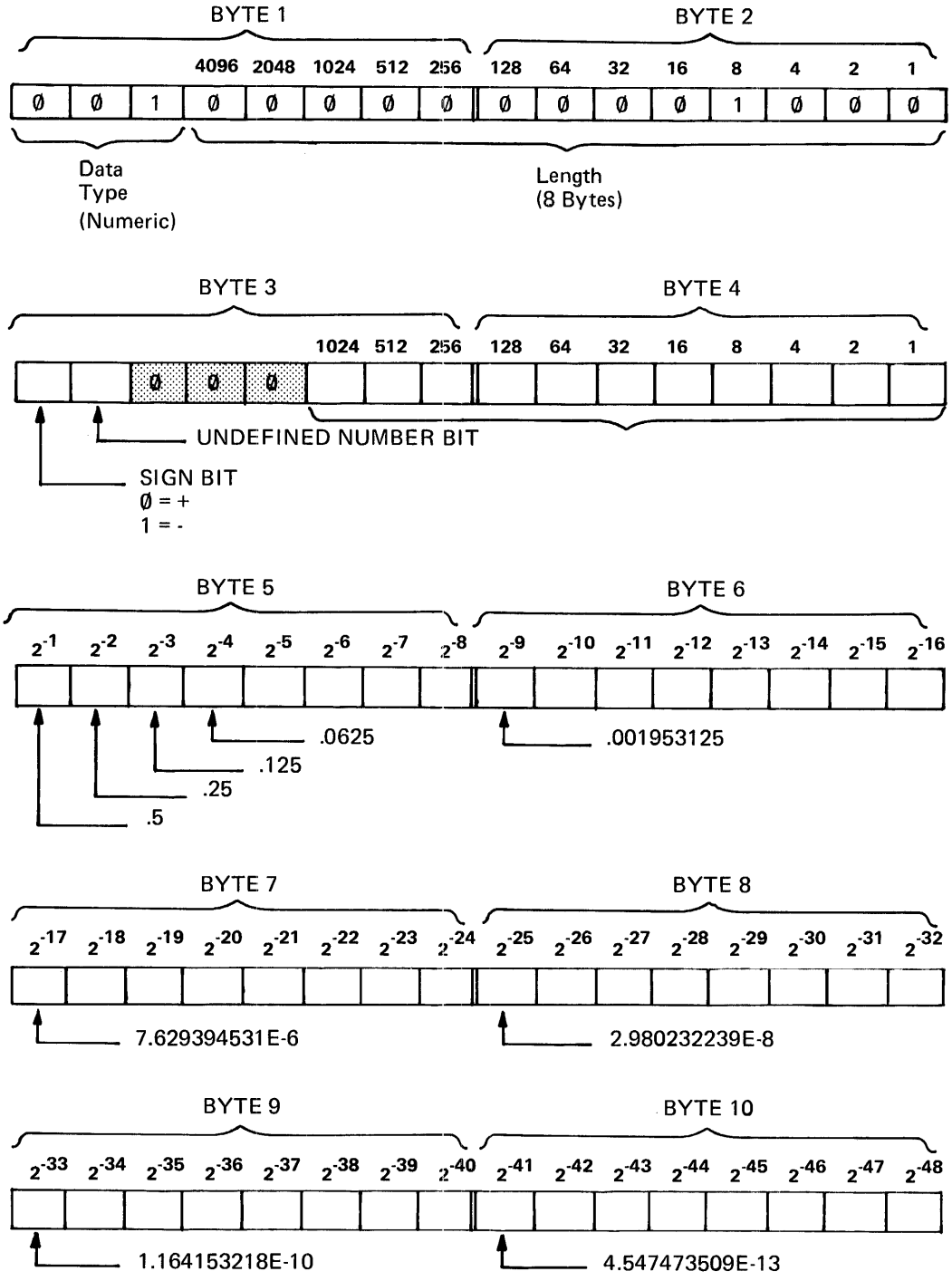
4051 INTERNAL FLOATING-POINT WORKSHEET



$n = .F_{10} \times (-1)^{s} \times 2^{(E-1024)}$

where: n=decimal number
F=decimal equivalent of binary number (bytes 5-10)
s=sign bit (1 or Ø)

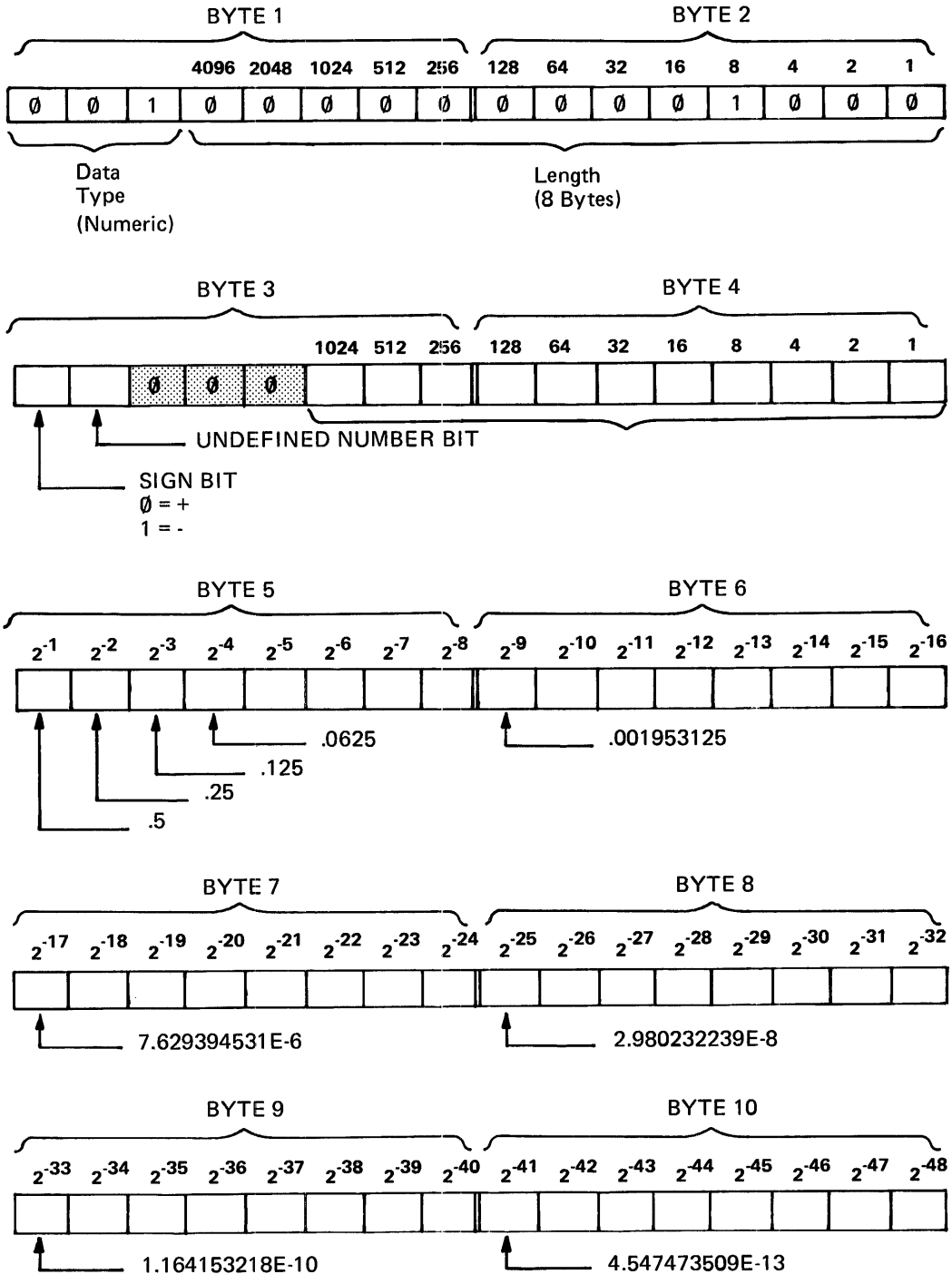
4051 INTERNAL FLOATING-POINT WORKSHEET



$n = .F_{10} \times (-1)^{s} \times 2^{(E-1024)}$

where: n=decimal number
F=decimal equivalent of binary number (bytes 5-10)
s=sign bit (1 or Ø)

4051 INTERNAL FLOATING-POINT WORKSHEET



$n = .F_{10} \times (-1)^s \times 2^{(E-1024)}$ where: n=decimal number
F=decimal equivalent of binary number (bytes 5-10)
s=sign bit (1 or Ø)

DIAGRAMS

Symbols used on the diagrams are based on ANSI Standard Y32.2—1975. Logic symbols are based on ANSI Y32.16—1975. Logic symbols depict the logic function performed and may differ from the manufacturer's data. The following explanations and accompanying schematic example (Fig. A) show typical usage of symbols and their meaning.

1. TRUE HIGH and TRUE LOW Signals

All signal names on the schematics will be followed by −1 or −0. A TRUE HIGH signal will be indicated by −1 and a TRUE LOW signal will be indicated by −0.

SIGNAL −1 = TRUE HIGH
SIGNAL −0 = TRUE LOW

2. Cross-References

Schematic cross-references (from/to information) will be included on the schematics (Fig. B). The "from" reference will only indicate the signal "source," and the "to" reference will list all loads where the signal is used. All from/to information will be enclosed in parenthesis.

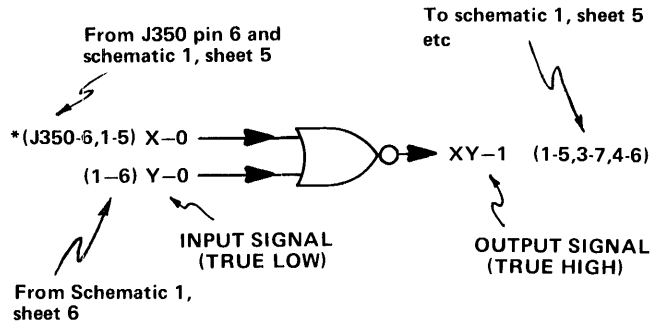


Fig. B

*In special cases where a signal has more than one source (wire OR for example) or comes from "the outside world," the multiple sources or connectors will be indicated by an asterisk preceding the reference.

3. Board Edge Information

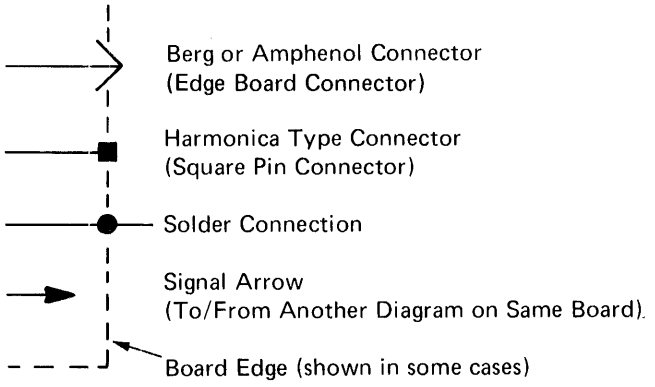


Fig. C

4. Jacks and Plugs

The fixed portion of a connector shall be labeled J for Jack and the movable portion shall be labeled P for Plug. If neither is more moveable than the other, the female connector shall be labeled P and the male connector, J.

5. Schematic Example

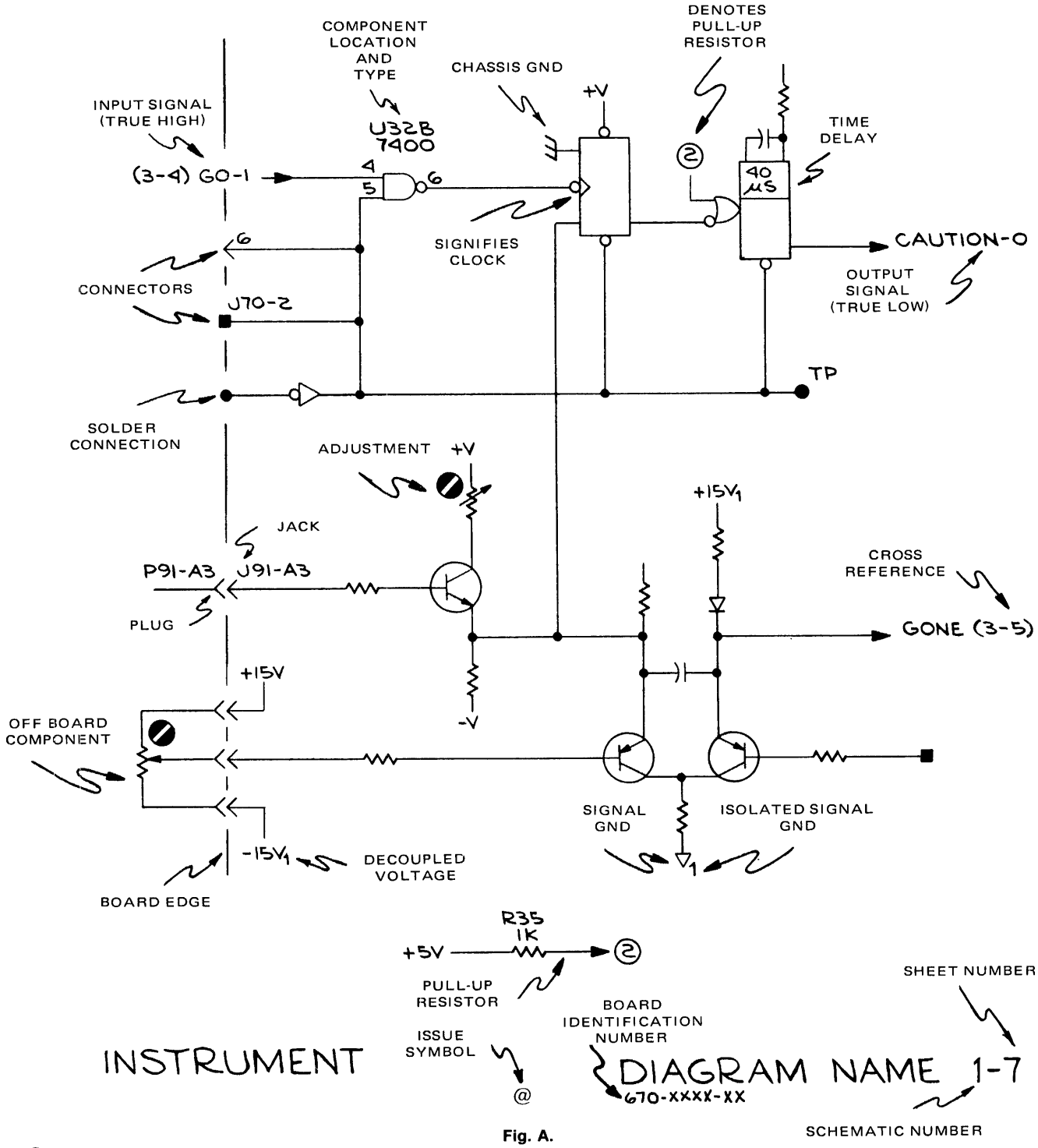
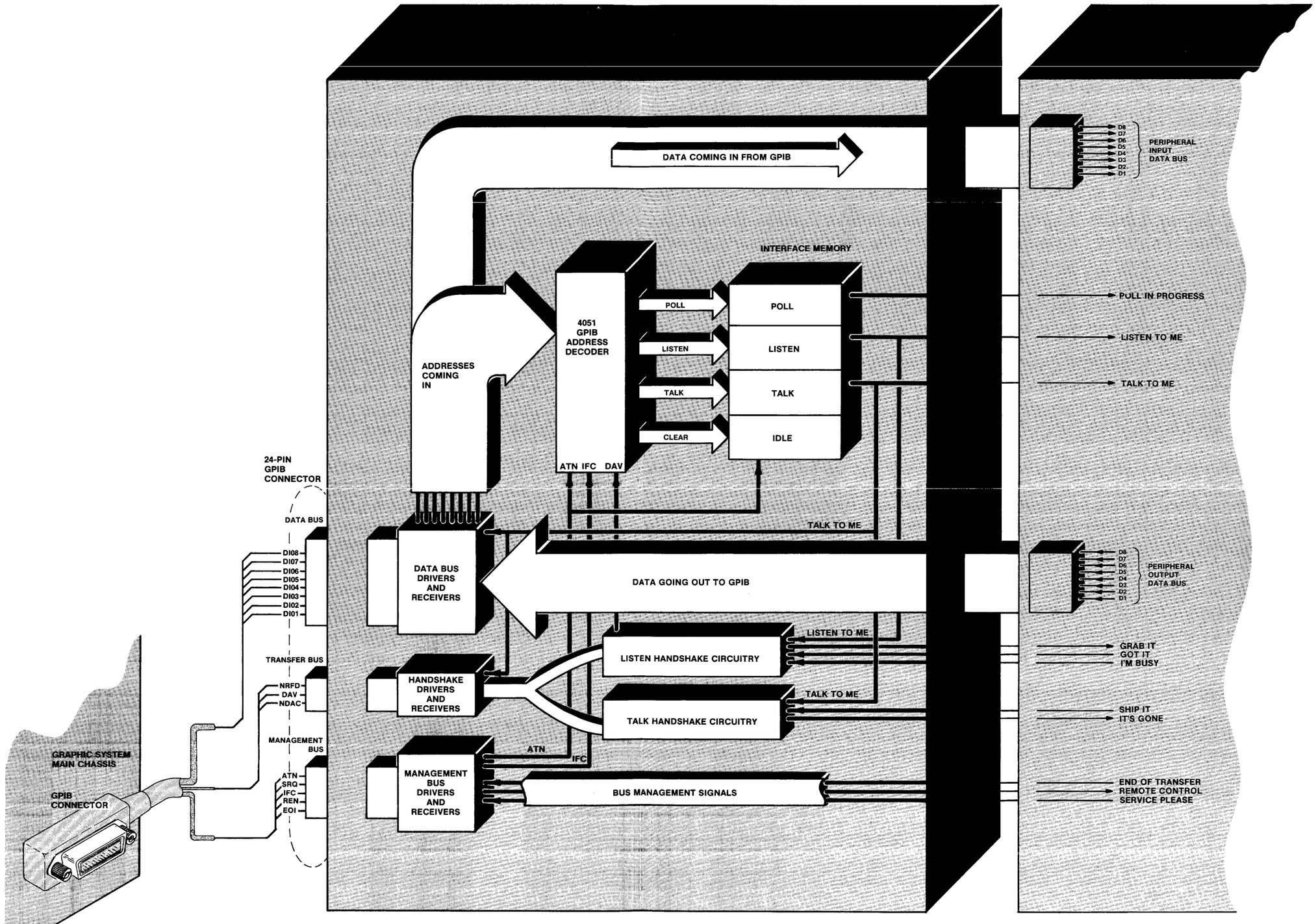
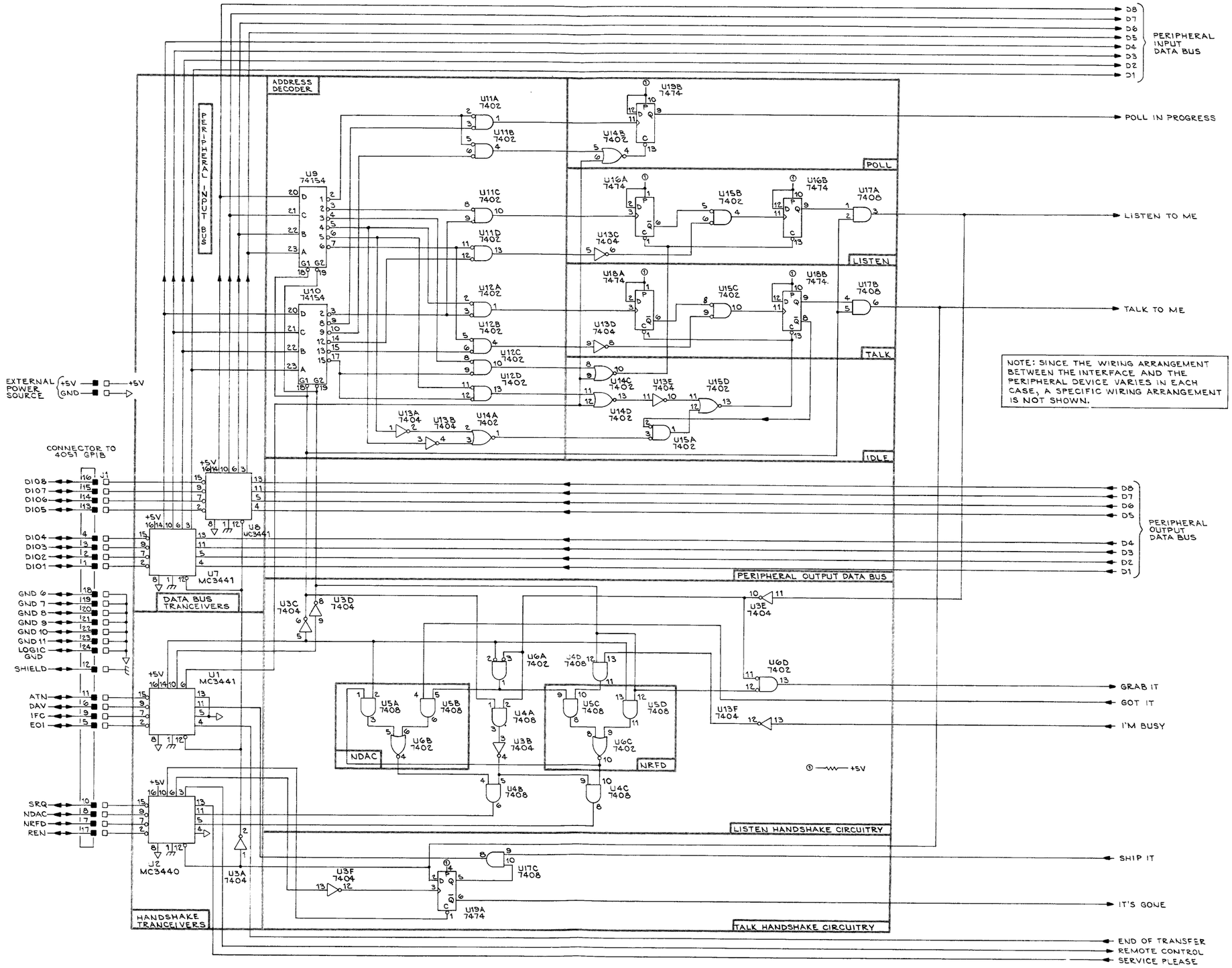


Fig. A.

@



4051 GRAPHIC SYSTEM **GENERAL PURPOSE INTERFACE FOR 4051 GPIB** **PERIPHERAL DEVICE**



4051 GPIB HARDWARE SUPPORT

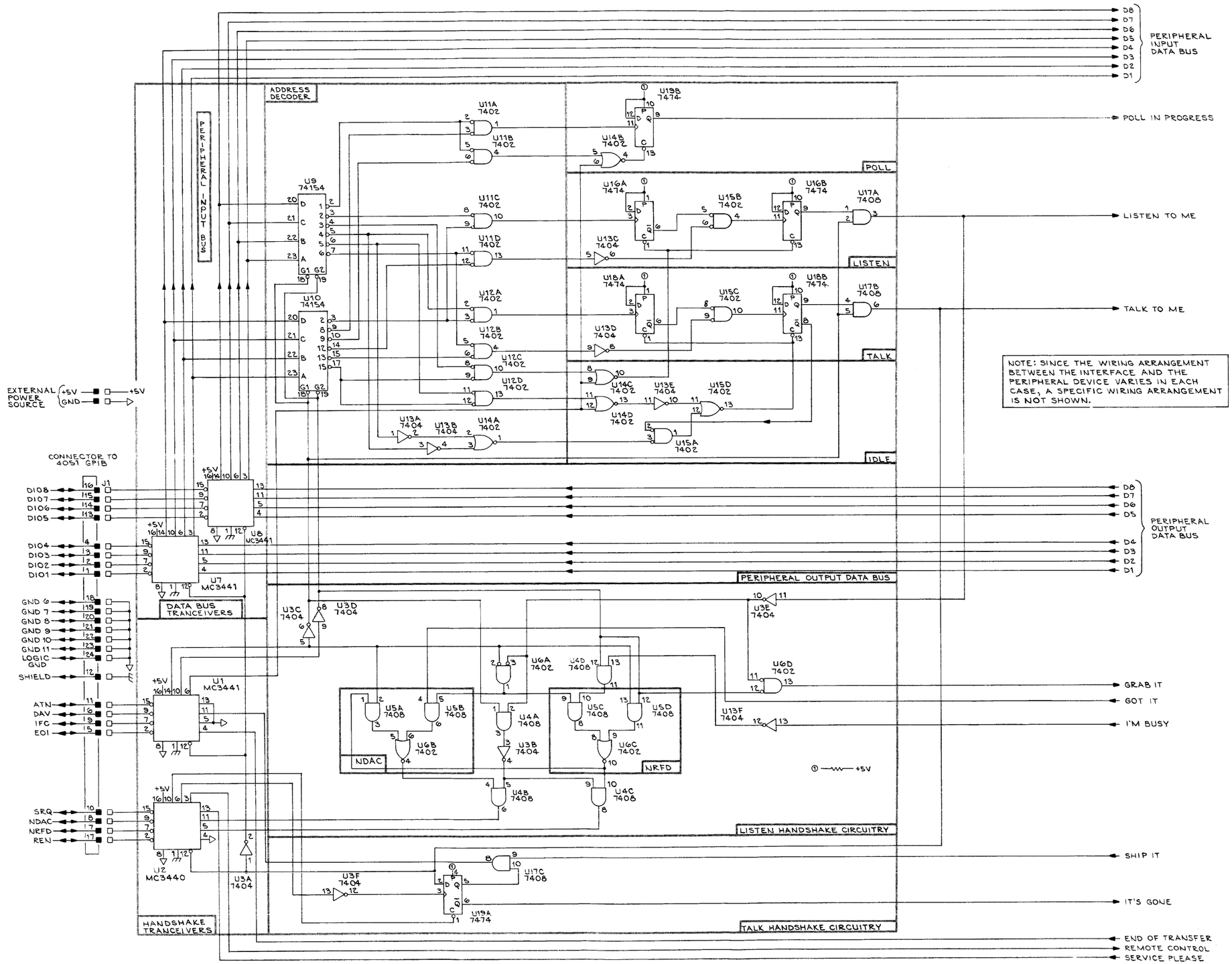
2270-48
REV. A, MAR 1979

4051 GPIB GENERAL PURPOSE INTERFACE

DIAGRAM B

IC LIST

COMPONENT	TYPE
U1	MC 3441
U2	MC 3440
U3	SN 7404
U4	SN 7408
U5	SN 7408
U6	SN 7402
U7	MC 3441
U8	MC 3441
U9	SN 74154
U10	SN 74154
U11	SN 7402
U12	SN 7402
U13	SN 7404
U14	SN 7402
U15	SN 7402
U16	SN 7474
U17	SN 7408
U18	SN 7474
U19	SN 7474

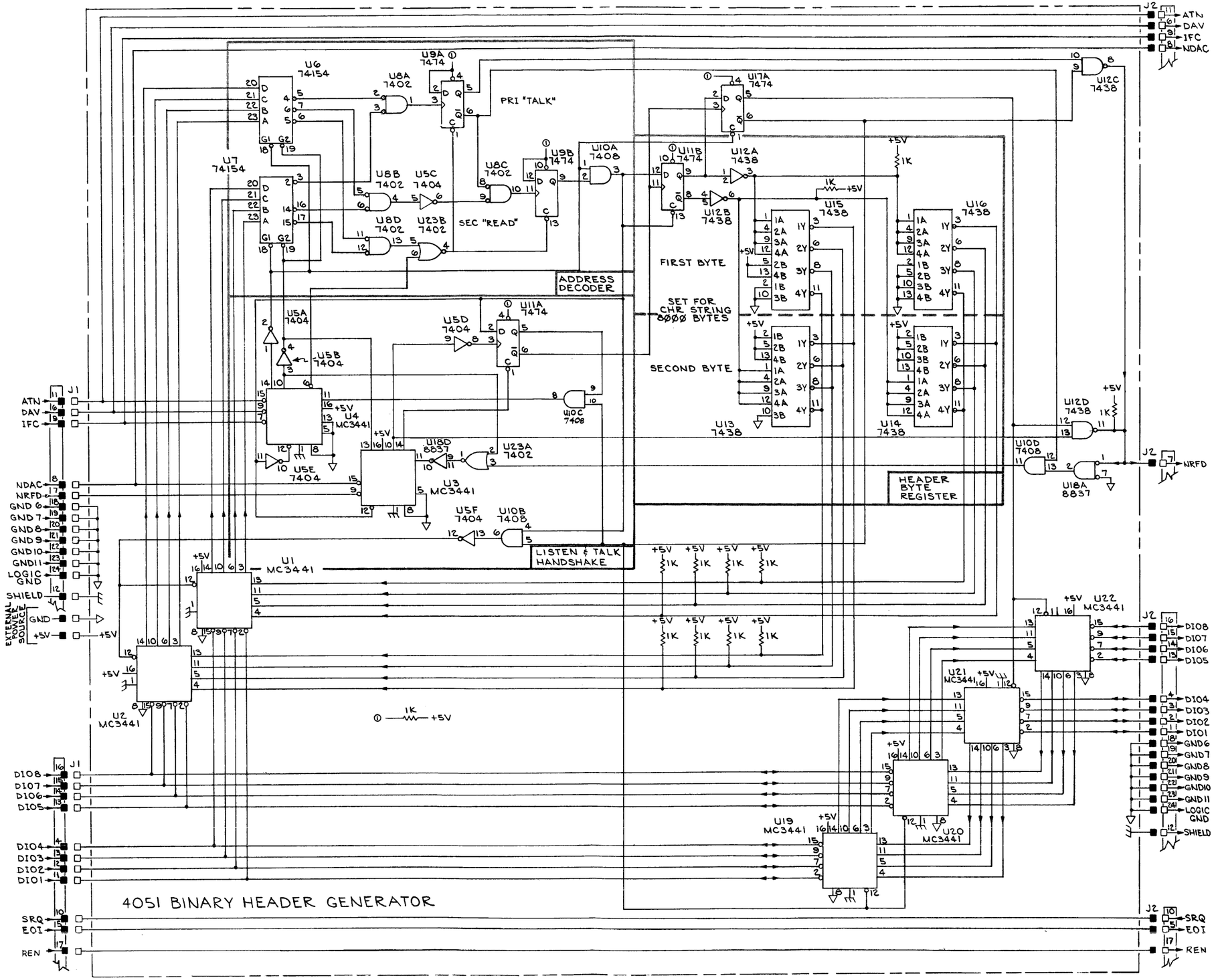


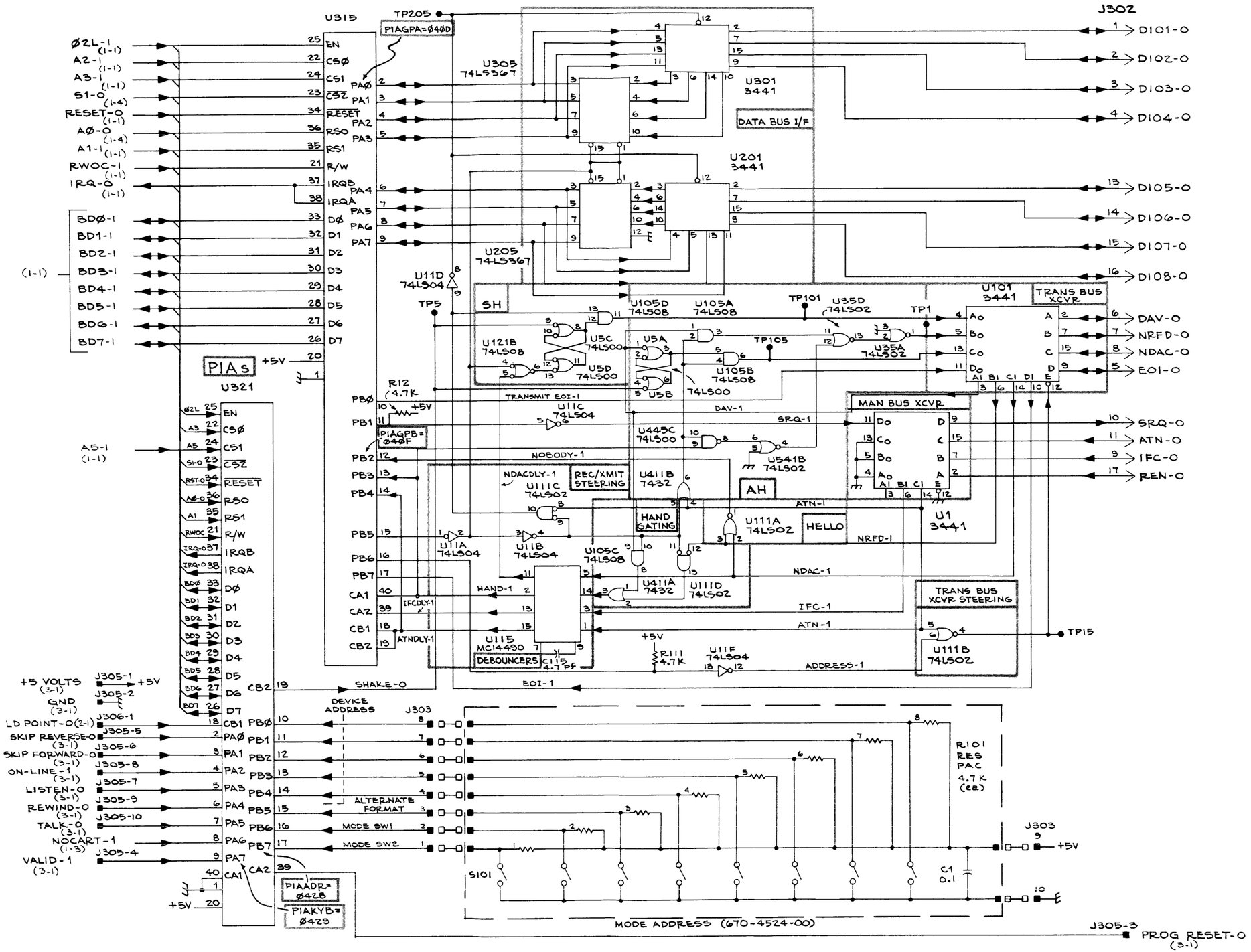
2270-48
REV. A, MAR 1979

4051 GPIB GENERAL PURPOSE INTERFACE DIAGRAM B

IC LIST

COMPONENT	TYPE
U1	MC 3441
U2	MC 3441
U3	MC 3441
U4	MC 3441
U5	SN 7404
U6	SN 74154
U7	SN 74154
U8	SN 7402
U9	SN 7474
U10	SN 7408
U11	SN 7474
U12	SN 7438
U13	SN 7438
U14	SN 7438
U15	SN 7438
U16	SN 7438
U17	SN 7474
U18	8837
U19	MC 3441
U20	MC 3441
U21	MC 3441
U22	MC 3441


PRI "TALK"
SEC "READ"
FIRST BYTE
SET FOR CHR STRING 0000 BYTES
SECOND BYTE
ADDRESS DECODER
HEADER BYTE REGISTER
LISTEN & TALK HANDSHAKE
ATN
DAV
IFC
NDAC
NRFD
GND 6
GND 7
GND 8
GND 9
GND 10
GND 11
LOGIC GND
SHIELD
EXTERNAL POWER SOURCE
GND
+5V
DIO8
DIO7
DIO6
DIO5
DIO4
DIO3
DIO2
DIO1
SRQ
EOI
REN
1K
+5V
4051 BINARY HEADER GENERATOR
4051 GPIB HARDWARE SUPPORT
2270-49
REV A, MAR 1977
DIAGRAM C
4051 HEADER GENERATOR



4051 GPIB Hardware Support

@

4924 GPIB INTERFACE

670-4525-00

DIAGRAM D