OSBORNE/McGraw-Hill

PET and the IEEE 488 Bus (GPIB)

Eugene Fisher - C.W. Jensen

PET and the IEEE 488 Bus (GPIB)

Eugene Fisher - C.W. Jensen

OSBORNE/McGraw-Hill
Berkeley, California

Published by
OSBORNE/McGraw-Hill
630 Bancroft Way
Berkeley, California 94710
U.S.A

For information on translations and book distributors outside of the U.S.A., please write OSBORNE/McGraw-Hill.

PET AND THE IEEE 488 BUS (GPIB)

1234567890 DODO 89876543210

ISBN 0-931988-31-4

Cartoons by Bruce Mishkin.

In recognition of their understanding and support in this project, we dedicate this work to our families — the Fishers and the Jensens — for in a large sense, it's a family affair.

Melissa, Christine and Randy Fisher
Mary Belle, Tom, Carl, Reg, and Edna Jensen

Thank you all
Gene
Bill

Contents

FIGURES

FIGURES (Continued)

TABLES

Preface

The purpose of this book is to describe the relationship between the PET* and the IEEE 488 Bus in sufficient technical depth so the PET user can find answers to timing and address problems that might occur while interfacing a variety of electronic instruments to the PET via the J1 interfacing port.

There are at least three levels of PET users to whom this book is addressed.

The first level of users includes hobbyists, scientists, or programmers who want to read digital voltmeter data (for example) into the PET. By referring to available "cookbook" instructions, these persons enter the necessary statements into the PET; the data is transferred onto the GPIB and is displayed on the PET CRT. Fine, as long as it works! But what if all procedures are seemingly correct, yet the data does not appear? What then? These people suddenly become second-level users.

Second-level users are the ones who, when the data fails to appear, ask why, and then proceed into the inner workings of the equipment, seeking answers by such methods as comparing oscilloscope-displayed pulse trains with documented timing diagrams. These persons might determine that a particular instrument, which has received a primary command to send its data to the PET, did not send data because the device was waiting for a secondary command. Such problems do occur frequently, yet PET users today find little — if any — documentation for reference and guidance toward the solution of these bus problems.

Finally, detailed timing information, programming flowcharts, and test pro-

*The PET, Model 2001.

grams are supplied so that those readers having digital logic background and circuit design experience could well design instruments that would be compatible with both the GPIB and the PET computer.

There are others to whom this book is addressed. They include community college electronics instructors and their students as well as teachers and students in training divisions throughout industry.

One great problem facing electronics industries today is the shortage of qualified engineers and technicians. With the burgeoning industrial market for electronic instruments that are IEEE 488 Bus compatible and the ever growing number of manufacturers who produce them, it is fast becoming apparent that industrial training programs and community college electronics departments should offer IEEE standard interconnecting bus technology as part of their curricula.

This book, when used in such training programs with the PET microcomputer and several IEEE Bus-compatible instruments, should provide students with a thorough understanding of the fundamentals of this new technology.

The text is organized into eight chapters, ranging from the evolution of digital interface systems and the history of the IEEE 488 Bus (Chapter 1) to a diagnostic troubleshooting program in Chapter 8.

You will find the PET data input and output requirements for sending statements to and from the IEEE Bus. You can study typical bus transactions and see, perhaps for the first time, the form of the actual data being transferred on the bidirectional bus. Transactions on the bus are illustrated briefly in terms of binary notations, with hexadecimal values (not part of the IEEE Standard) also explained.

For each of the eight input/output statements used by the PET, we show detailed timing sequences of the control and management lines as they relate to primary and secondary addresses and the transfer of data.

If you have non-IEEE standard devices and want to interconnect them on the GPIB with the PET, you can learn one method of doing so in Chapter 6, "Interfacing the GPIB With a Non-Standard Bus Device."

In Chapter 7, we show how the PET communicates with five IEEE standard instruments manufactured by prominent U.S. manufacturers. The PET outputs its data to an inexpensive on-line printer, displays readings from a digital voltmeter, and shows the transmitted frequencies from 13 channels of a CB transmitter read via a frequency counter. Two important instruments of general interest — a logic analyzer/display formatter for troubleshooting and a 488 Bus interface coupler for interfacing non-compatible instruments — are described in detail. From the information given, users should have little difficulty in connecting the PET to these and similar instruments. Better yet, if something in the GPIB system does not work, you probably can find out why.

The appendices have a list of mnemonics and their definitions contained in the IEEE Std 488-1978 publication. Also in the appendices will be found probably the most exhaustive reference list published to date of articles about the IEEE 488 Bus. Finally, there is a useful list of companies and IEEE 488 Bus-compatible instruments that they manufacture.

Before studying this book, the reader should know the PET's BASIC language.

Throughout the text, we make numerous references to the IEEE 488 Bus as it interconnects the PET with various electronic instruments. These references are in relation to the physical cable, connectors, and certain terms set fourth in publication IEEE Std 488-1978. Because of so many references to this interconnecting bus, we refer to it in these terms, which are all synonymous:

IEEE 488 Bus
Bus
General Purpose Interface Bus (GPIB)
The standard bus

ACKNOWLEDGMENTS

The name PET is a registered trademark with regard to any computer product, and is owned by Commodore Business Machines, a division of Commodore International. Permission to use the trademark name PET has been granted by Commodore Business Machines, and is gratefully acknowledged. Special thanks are due Mr. Lawrence Perry of Commodore Business Machines for the valuable assistance and support he provided.

A special "thank you" is due Mr. Donald C. Loughry of Hewlett-Packard. He offered many suggestions and provided much valuable assistance for the sections titled "Development of the IEEE Std 488 Bus" and "Mechanical Features."

We also wish to formally recognize and thank the following people for their splendid cooperation by loaning instruments for testing with the PET and the General Purpose Interface Bus. Many of these individuals have also provided technical material:

Sterling B. Hager, Public Relations Specialist, Centronics Data Computer Corporation.
H. W. Mette, Marketing Manager, Systron-Donner Corporation.
Peter Macalka, Vice President, Testing and Measurements, Marketing and International Group, Systron-Donner Corporation.
Gerald K. Mercola, ICS Electronics Corporation.
Richard Pawson, Commodore Systems Division, London, England.
James Hardin, Hewlett-Packard.
Earl Eason, Account Specialist, Test and Measurement Products, Tektronix Inc.
H. St. Onge, Manager, Corporate and Marketing Communications, Belden Corporation.
Ralph Gadsby and Thorson West, Representatives for Centronics Data Computer Corporation.
John A. Cooke, University of Edinburgh, England.

We wish to give special recognition to Janice Enger and Adam Osborne for their helpful suggestions and time devoted to organizing and editing the material; it is much appreciated.

Finally, we are grateful to Mary Belle Jensen for typing and preparing finished drafts sent to the publisher and for her watchful eye in picking up inconsistencies and errors. Those that remain are, naturally, our responsibility.

Constructive comments and suggestions for improvements in the book are solicited and will be most welcome.

Eugene R. Fisher
C. William Jensen

CHAPTER 1

Layman's Introduction to the GPIB

This book is about digital interfaces, the IEEE 488 Bus, which is one of the most popular digital interfaces, and the PET microcomputer, which uses the IEEE 488 Bus.

The PET has, like all microcomputers, a "brain" called a microprocessor. Often referred to as a "computer-on-a-chip," these tiny devices, no bigger than a thumbnail, have propelled desk-top microcomputers into prominence in industry and in the home.

Microcomputers are having a profound influence on almost every branch of industry, and in every country of the industrialized world. Digital interfaces, the subject of this book, are of greatest interest to scientific and technical users, for whom microcomputers have radically changed the methods of conducting research and collecting data.

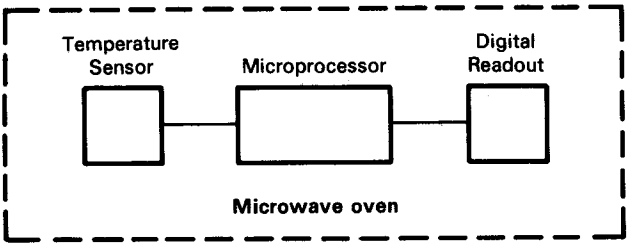
Most microcomputers have generally comparable features. Only the PET, in the under $1000 price range, has the unique capability to interconnect and control a variety of commercial digital instruments and input/output devices. These instruments and devices can communicate and interact with one another to perform a seemingly endless variety of operations; the type and variety of operations is limited only by the ingenuity of the operator/designer.

But what are some of these instruments and devices, and why would you want to connect them together?

There are literally hundreds of electronic instruments being marketed today that comprise a whole new world of instrumentation.* Under program control of the PET, these instruments record temperature and light intensity; read electrical current, voltage, and resistance; monitor humidity; measure liquid and gas flow rates; record acceleration, velocity, and various types of radiation; and count frequencies and other external events. Other apparatus, which you can also place under the program control of the PET, includes devices that switch electric, hydraulic, or pneumatic circuits off and on, regulate liquid or gas flow rates, raise and lower temperatures and humidity, and otherwise control environments and processes.

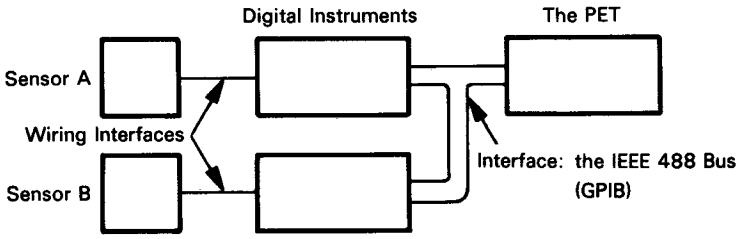
You use the PET digital interface to interconnect such digital instruments, and you control the interconnected instruments by programming the PET.

"Interfaces" exist in many forms. In a computer-controlled microwave oven, for example, we are not aware that interfaces even exist, because the electronics of the microwave oven are self-contained and concealed from view. However, one or more interfaces are present: one is the group of wires, the physical connections between the temperature or humidity sensor and the oven's microprocessor; another interface exists between the microprocessor and the digital display on the front panel.



In a microwave oven, the sensors, interface wires, and microprocessor are self-contained.

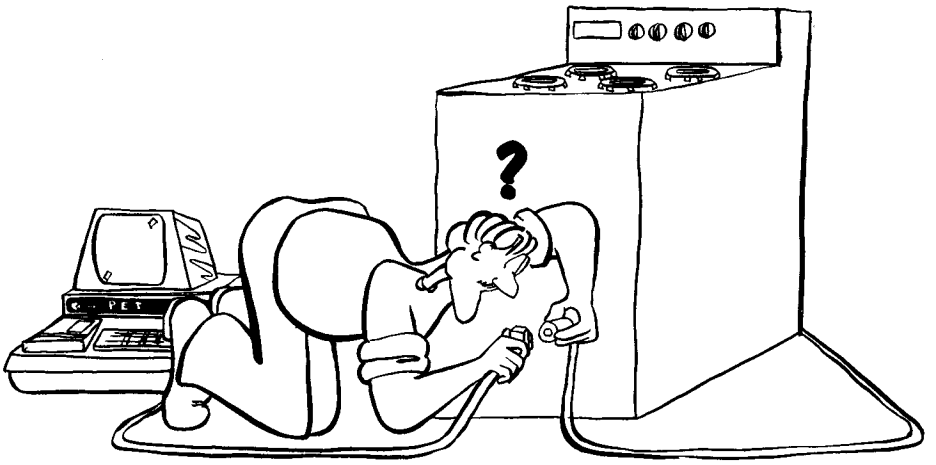
Whereas interfaces in a microwave oven are contained internally, PET computer/digital instrument system interfaces depend on external apparatus.



In the diagram above, the sensors are shown as separate components. Often, the sensors are an integral part of the digital instrument itself.

*See Appendix A, "Companies and their GPIB Products."

The need to standardize the wiring and digital signals used to interface digital instruments is akin to standardizing electrical connectors, outlets, and wiring in the home. Consider the simple example of trying to power-up certain electrical tools. How often have you tried to plug a three-prong (grounded) AC connector into a two-prong wall outlet, only to find that the two would not mate? The solutions? For the incompatible AC connectors a 3-wire to 2-wire AC adapter (an interface) will solve the problem. For incompatible digital instruments no such simple solution exists; extensive modifications to interconnecting wiring, hardware, signal levels, and timing may be needed. The ultimate solution, therefore, is to design digital instruments in accordance with a standard interface. This allows users of instruments built by different manufacturers to plug them together and control them using a single computer.



This problem of compatibility among digital instruments and computers has not been the exclusive concern of engineers and scientists. Novice computer users and hobbyists have sought to reduce system costs through improved, simplified design. Some computer owners try to cut home utility costs and improve energy efficiency in the home by more effectively controlling heat, humidity, air conditioning, hot water consumption, and the use of lights.

While such controls have existed for years, to build a control system, designers had to study the characteristics of all the involved components, then interface the various units in the system with circuits designed specifically for that system. There have been very few (if any) "standard" or "universal" ways to connect instruments, sensors, or drivers with each other and/or computers. Two problems (at least) have resulted. First, an undue amount of time was wasted designing excessively complex systems; second, systems have rarely been compatible. Instruments having input/output (I/O) connections to other instruments

or computers have generally required special interfacing before they could be interconnected. Due to the universal acceptance in 1975 of a national and an international standard for interfacing programmable digital instruments, these problems have, for the most part, been eliminated.

The standards referred to are:

- IEEE* Std 488-1978, IEEE Standard Digital Interface for Programming Instrumentation. Its international counterpart is:
- International Electrotechnical Commission (IEC) Publication 625-1.

These two documents form the worldwide standards for the General Purpose Interface Bus (GPIB) and provide the basis for this book.

*Institute of Electrical and Electronics Engineers.

DEVELOPMENT OF THE IEEE STD 488 BUS*

The basic concept of the GPIB is that it is an electrical/mechanical interfacing standard whereby an instrument manufactured by the XYZ Company or Brand X of another company can be interconnected with any other instrument conforming to the same electrical interface.

The IEEE Std 488-1978, generally referred to as the GPIB, has been painstakingly specified in its electrical, mechanical, and communication operation. This standard bus and its interconnecting cabling provide a means for the mini- or microcomputer user to interconnect a computer controller with a variety of instruments and equipment manufactured in the United States or in other countries and to program the entire system for optimum operation.

At the time of this publication, reliable sources estimate there are more than 200 manufacturers of instruments that are adhering, at least partially, to this GPIB interface standard. It is this interfacing standard, supported by the PET microcomputer, which provides the PET with its primary capability to communicate with a wide variety of instruments and devices in the outside world.

EVOLUTION OF DIGITAL INTERFACE SYSTEMS

Since digital instruments were first built, it has been necessary to link them together, thus allowing data to be passed between them and processed automatically.

At first, manufacturers designed instruments to their own standards. Such instruments had dedicated interface structures with specialized control, signal, and data lines. Logic levels, codes, formats, timing restrictions, and data rates were often unique. Electrical parameters differed not only from one instrument manufacturer to the next, but sometimes among instruments manufactured by the same company.

Engineers from various organizations sought solutions to the interfacing problems created by such a variety of digital systems designs. They attempted to standardize on a universal interface, from the perspective of system controllers, but this met with only limited success. The eventual standard adopted optimizes the interface from the perspective of the instruments.

As digital systems became more complex, engineers were confronted with monumental interfacing problems, in particular when dealing with programmable instruments that might require interconnecting cables with 50 to 100 wires. Considerable effort — and thus cost — was expended to complete such tasks.

In the early 1970s, interfacing problems became so severe that most U.S. electronics manufacturers who produced digital programmable equipment agreed that an all-out effort was needed to find a solution.

*Mr. Donald C. Loughry, Hewlett-Packard Company, inspired the words for this section in a talk delivered at the Engineering Technology Transfer Symposium, Lawrence Livermore Laboratory, Livermore, California, 1976. He has given his kind permission to extract material from it.

The United States was not alone. During the latter part of 1971 and early 1972 Germany, also recognizing the need for a standard among the manufacturers of digital electronics equipment, became the driving force for members of the International Electrotechnical Commission (IEC) to work on an international digital instrumentation standard.

At about the same time, in March 1972, an advisory committee of the IEEE met to explore various digital interface possibilities and to recommend a plan that could be adopted as a standard digital interface for programmable instruments and controllers.

The United States adopted the IEEE plan. This plan was then presented in October 1972 as the formal U.S. proposal to IEC Working Group 3 at its first meeting in Munich, Germany. The IEC adopted the United States' plan as the draft from which an IEC standard was to be developed.

Following the international meeting, the U.S. Delegation and the IEEE Subcommittee, realizing that an international standard would take considerable time to develop, prepared a similar document intended as an IEEE Standard. This document, the IEEE Std 488-1975 "Digital Interface for Programmable Instrumentation," was published in April 1975. This identical document was approved in October 1975 by the American National Standards Institute and bears the document number ANSI MC 1.1-1975.

Copies of IEEE Std 488-1978* are available from:

IEEE SERVICE CENTER
445 Hoes Lane
Piscataway, New Jersey 08854
U.S.A.
(201) 981-0060

The international standard is scheduled to be published in 1980 as IEC Publication 625-1, "An Interface System for Programming Measuring Apparatus (Byte Serial, Bit Parallel)."

PERSPECTIVES FOR INTERFACE DESIGN

There are two basic perspectives from which a digital instrumentation interface might be designed.

One optimizes the interface for system controllers, while ensuring that the interface can be used by a variety of digital programmable instruments.

The other optimizes the interface for digital programmable instruments, while ensuring that the interface is suited to system controllers. The IEEE standards committee chose the latter perspective; it selected an interface standard that was optimized for digital programmable instruments.

*The recent revision of IEEE Std 488.

Having decided that an interface should be optimized for a variety of digital programmable instruments, the IEEE committee considered two fundamental concepts:

1. A star interconnect with a dedicated interface to each device in the system.
2. A party-line bus structure.

While the second method adds some control function complexities when instruments communicate with each other, the party-line bus structure would, the committee believed, save hardware at the interfaces, providing advantages that outweighed the disadvantages.

Objectives

In preparing the standard, the IEEE committee focused on a manageable set of objectives. Since the party-line bus structure was to be basically a communications system, the committee examined four well-known communication parameters: data transfer rate, message length, transmission distance, and number of devices in the system. They also considered other factors: cost, flexibility, and compatibility.

Data Transfer Rate

The committee studied between 100 and 150 instruments or systems; they found that most data rates were less than a megabyte per second, typically in the kilobyte to tens of kilobytes per second range. Recognizing that there were memory disks and certain analog-to-digital converters operating in the megabyte per second range, the committee sought to optimize the system around the bulk of manufactured equipment having the greatest needs for communicating with each other. Hence the IEEE standard, when it was published, specified that the data rate was not to exceed one megabyte per second.

Message Length

In determining the message lengths to be handled, the committee again considered the majority of instruments being marketed at the time. They found that typical measurement device messages varied between 10 and 20 characters (or bytes) in length. This short message length (or even shorter) was deemed adequate, since short messages could be concatenated into longer data bursts. Accordingly, the selected bus structure was 8-bit parallel, byte serial.

Transmission Distance

The committee believed that a standard should be optimized for a cluster of closely grouped devices; therefore, they selected a maximum interconnecting cable length that was 20 meters, or twice the number of connected devices, whichever was less. Again they focused on typical systems, believing that interconnecting instruments operating at longer distances (e.g., 100 meters) should be handled separately. This led to a set of cabling restrictions.*

Number of Devices

Following the general guideline that an interface standard should serve the broadest need, the committee proposed that 15 be the maximum number of devices interconnected via a party-line bus structure. Although large systems were found to have 50 to 100 instruments interconnected, not all of these devices needed to communicate with each other. Typical systems had from five to ten communicating devices. Therefore, 15 was chosen as a maximum number.

Cost

Cost was a factor in the minds of committee members as they developed the bus concepts. They tried to establish a standard that minimized hardware costs — costs such as signal-line drivers, receivers and logic circuitry.

Flexibility

There are five categories of instrumentation products. These are measurement, stimulus, display, processor, and storage devices. The committee thought it imperative that the bus structure accommodate and interconnect all of these five product categories, as all are necessary in some, if not all, systems. The bus standard also needed to accommodate a wide variety of device capability — from card readers and paper-tape readers to desk-top calculators and minicomputers.

Compatibility

The term "compatibility" describes, in essence, the main objective of the standard. To put it in the words of Don Loughry, "you want to be able to have different engineers in different places at different times with different applications, define and design products to a common interface, and then be able to put these independently designed products together in a common system. . ."

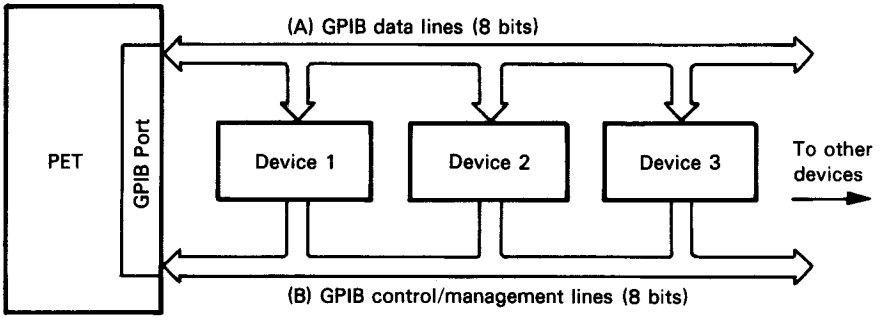
*IEEE Std 488-1978 par. 6.4.1, p. 64.

CHAPTER 2

GPIB Lines and Signals

This chapter describes each line of the GPIB, showing how it terminates in the PET and how the line is addressed. Sample programs demonstrate how to assert individual lines.

The GPIB connects the PET to remote devices in the general form shown below:



The GPIB consists of two groups of lines. One group (A) consists of eight data lines, each carrying one bit of data. The other group (B) has eight control/management lines. The eight bidirectional GPIB data lines carry parallel,

simultaneous bits transferred between the PET and a selected device on the GPIB, or between any talker and listener. The eight GPIB control/management lines supply information as necessary immediately before, during or after the transmission of data.

These 16 lines, with their associated signal ground return lines and logic ground, make up the total hard-wire connections between the PET and any one of 15 external devices (the maximum allowable) residing on the bus.

THE BUS STRUCTURE

The basic bus structure, IEEE Std 488-1978 (also referred to as the General Purpose Interface Bus — GPIB), is illustrated in Figure 2-1. A maximum of 15 devices, of four types, can be connected to this party-line bus structure. The four device types are:

1. The controller. In a PET system there can only be one, the PET itself.
2. Devices that talk only.
3. Devices that listen only.
4. Devices that talk or listen, depending on how they are programmed.

Along the bottom of Figure 2-1 is the bidirectional data bus, consisting of the eight parallel DIO lines that connect all instruments and the controller. These data lines, which are shown numbered according to the IEEE 488 connector pin designations, carry data, addresses, and other 8-bit parallel information.

To process the information (data flow) on the bidirectional data bus, up to eight control and status signals are required; they form two groups of lines. These two groups are labeled "Interface Management Lines" and "Transfer Control Lines." Not all of these lines take part in every data transaction, but all are important to the complete operation of the standard bus. One line, the REN (Remote Enable), is held permanently grounded by the PET controller to prevent remote devices from returning to local control.*

In the process of addressing the various instruments residing on the GPIB, the controller and the instrument it addresses make repeated use of four control lines. These are the ATN (Attention) line in the Interface Management Group and all three lines in the Transfer Control Group: DAV (Data Valid), NRFD (Not Ready For Data) and NDAC (Not Data Accepted). After ATN is asserted at the start of every addressing sequence, the other three lines take part in a procedure known as the "handshake." This is the protocol process where the listener and the talker (often the controller) take part in telling each other their status, such as "I'm not ready for data," "My data is valid," or "Your data is not accepted." Logic levels on these lines convey these conditions or, if the logic levels change, the converse of these conditions. These logic levels and the operation of the GPIB with PET as the controller are thoroughly explained later in this book.

*Normally, the coded LLO message in the IEEE Std 488 performs this function.

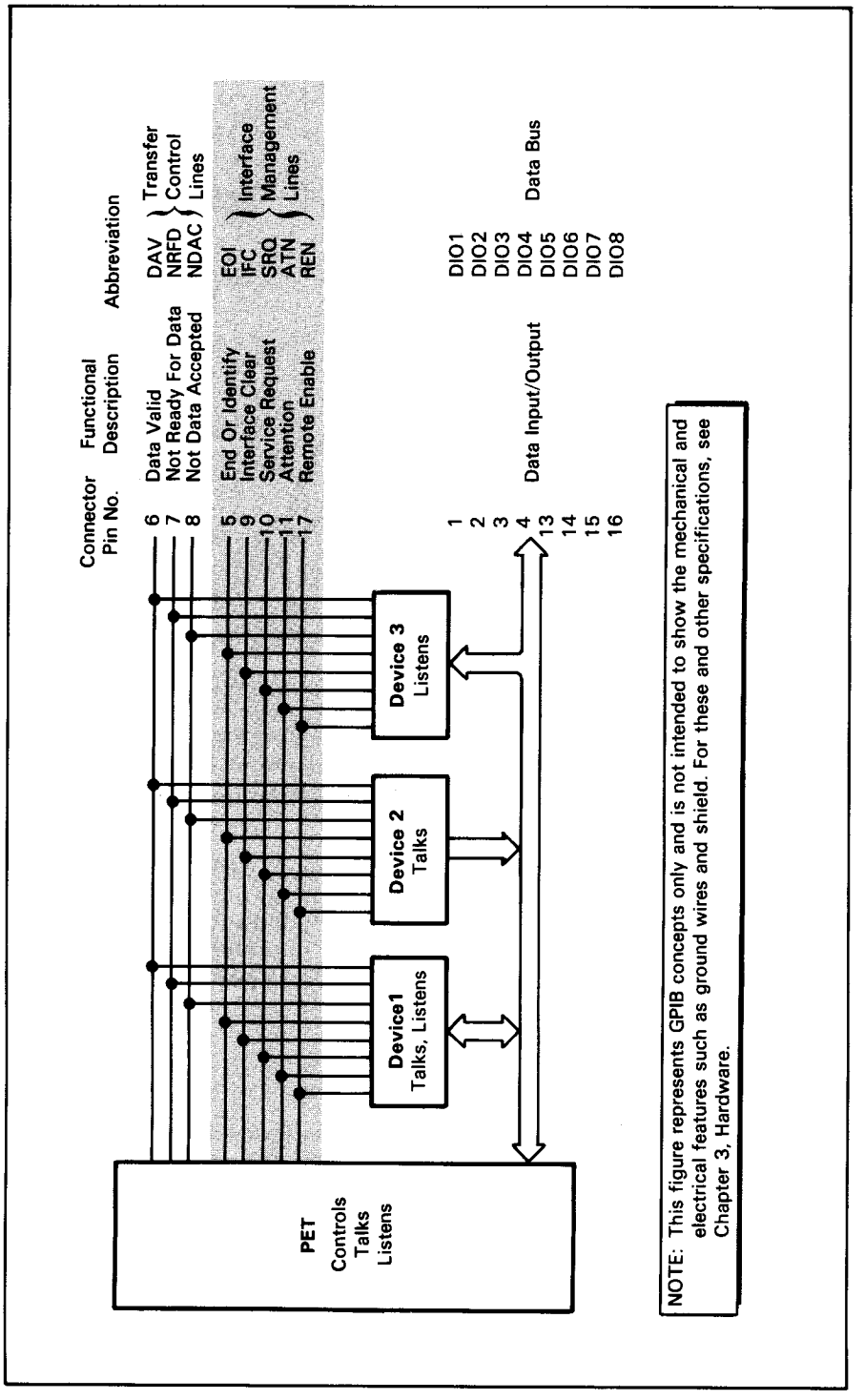
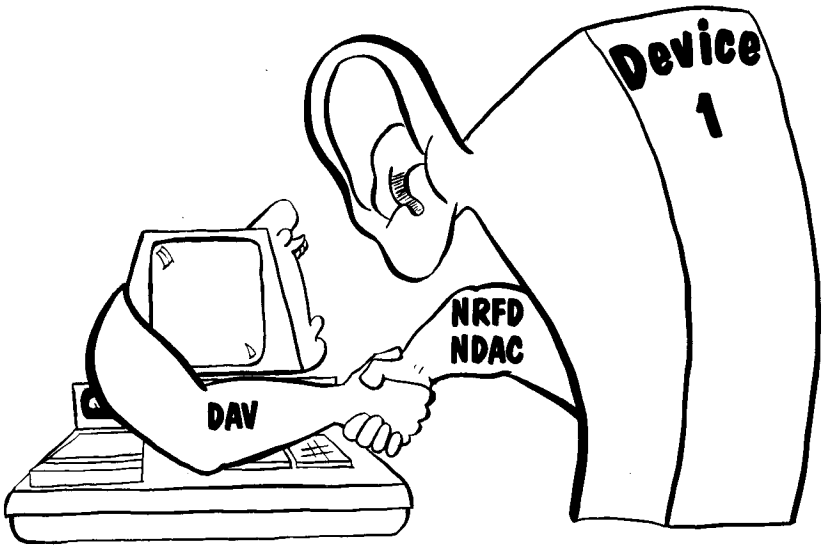


Figure 2-1. Block diagram of the PET and the IEEE 488 Bus (GPIB)

CONTROL LINES

Let us examine the Transfer Control Lines individually.



DAV (Data Valid)

The Data Valid line is asserted low (true) by a talker after it places its data on the DIO lines. This tells the listener that the information on the data lines is valid.

NRFD (Not Ready For Data)

The Not Ready For Data line, when asserted low (true), indicates that not all devices on the GPIB are ready to receive data. Each instrument, in its own time, releases this line. However, the line cannot return to its high (inactive) state until the slowest responding device lets go.

NDAC (Not Data Accepted)

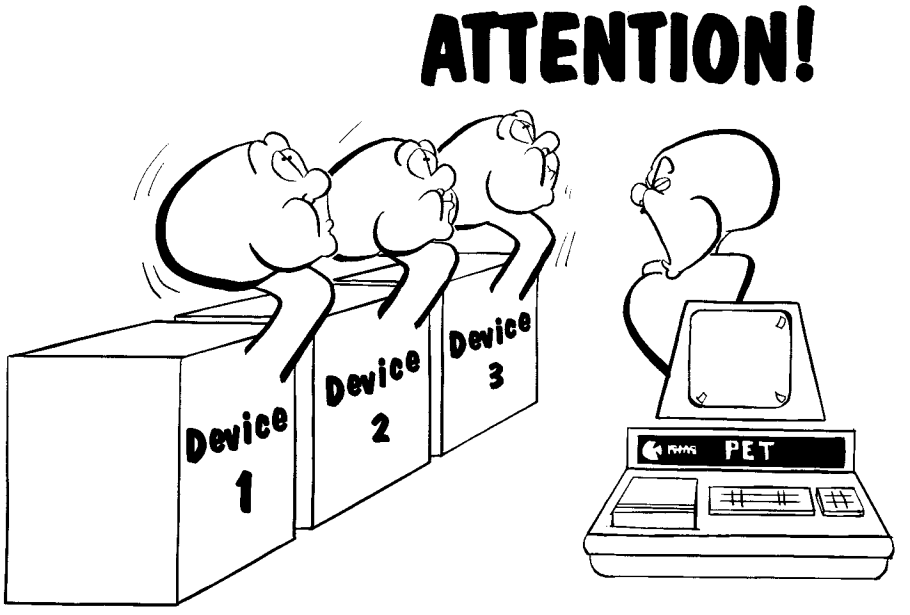
The Not Data Accepted line is controlled by the device or devices receiving the data. This line is held low (true) until all the receiving devices (listeners) capture the particular address or data byte; then the line is set high.

Now consider the Interface Management lines.

ATN (Attention)

The ATN line is asserted only by the controller during the addressing or command sequence. It can be activated low (true) only by the controller-in-charge to get (as the name implies) the attention of devices residing on the bus. Before sending commands on the eight data lines to peripheral devices, the PET (acting as the controller-in-charge) activates this ATN line low (true) to tell all devices that signal levels present (both true and false) on the data bus represent addresses and control messages for them.

When the ATN line is returned high, only the talker and listeners previously activated take part in the subsequent data exchange.



IFC (Interface Clear)

The Interface Clear line transports a reset signal that can be initiated only by the system controller.

When the PET (the controller) is powered ON or reset, this line is driven low (true) for approximately 100 milliseconds by the PET internal reset signal, and all bus devices are returned or set to their idle (inactive) states.

REN (Remote Enable)

The Remote Enable line can be activated only by the system controller. By having a grounded pin, which holds the REN line permanently low (true), the PET controller enables this line, and bus residing devices will then respond to the controller commands or those of another talker. If this REN line were allowed to become inactive high (false), all bus devices would return to local control.

SRQ (Service Request)

The Service Request line is a type of interrupt line, and it can be activated low (true) by any device residing on the bus and needing service from the controller. Such needs for service can, for example, be initiated by a digital meter having a voltage reading available, or a bus device having an internal error that has been detected. Even though signals on this line are not implemented in PET BASIC, this SRQ line is part of the GPIB bus structure and is available to the PET while operating as a controller.

EOI (End Or Identify)

The End Or Identify line can be asserted by the controller-in-charge or a bus-compatible talker. This line may be pulled low (true) by a talker during the transfer of its last data byte in a multiple-byte message to signal the END of the message. A talker has the option of using EOI. However, EOI is always pulled low (true) by the PET during the transfer of its last data byte onto the DIO lines. By using this same EOI signal line together with the ATN message, the controller-in-charge initiates a parallel poll sequence.

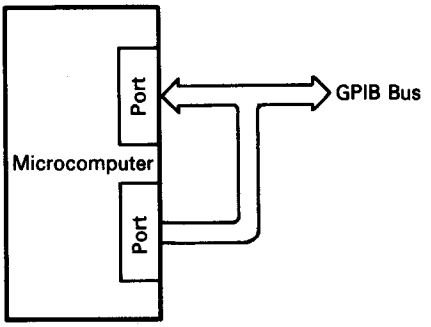
The GPIB line functions are summarized in Table 2-1.

Table 2-1. Functions of the GPIB lines

Name of Line	Description
ATN	**Attention.** Issued only by controller to gain attention of bus devices before beginning handshake sequence and to denote address/control information on the data bus.
DAV	**Data Valid.** Issued by talker to notify listener that data has been placed on the DIO lines.
DIO1-8	**Data Input/Output.** Eight data transfer lines. Also called data bus.
EOI	**End Or Identify.** Issued by talker to notify listener that the data byte currently on the DIO lines is the last one. Issued by controller together with ATN to initiate a parallel poll sequence.
IFC	**Interface Clear.** Issued only by controller to bring all active bus devices to a known state.
NDAC	**Not Data Accepted.** Issued by listener while fetching data from the DIO lines.
NRFD	**Not Ready For Data.** Issued by all listeners. Released by each listener as it becomes ready to accept data.
REN	**Remote Enable.** Permanently grounded by the PET to maintain control over the system.
SRQ	**Service Request.** Issued by any device needing service from the controller. Not implemented in PET BASIC.

DATA BUS LINES

The PET and GPIB data bus lines (DIO1-8) are similar to typical microprocessor input and output ports. These lines carry 8-bit parallel data to or from the PET.



When eight bits of data are sent to the output port, these same eight bits appear on the GPIB data lines. This operation is controlled completely by the PET (a controller) when the standard GPIB communication is used for the PRINT, INPUT, GET, and other statements. Data on the DIO bus lines is accompanied by transfer control (i.e., handshake line signals) and interface management line signals.

DATA TRANSFER

The data bus in Figure 2-1 is expanded in Figure 2-2 to show pictorially how addressing information and data are transferred in bit parallel, byte serial fashion on the bus.

Figure 2-2 (a) represents a section of the bidirectional data bus having eight wires numbered DIO1-8. The eight wires are divided into an upper-four group with DIO8 being the most significant bit (MSB) of the upper four. Likewise, the lower four wires form a group with DIO1 being the least signficant bit (LSB) in the lower four. Each of these sets of four wires has a decimal value of 1, 2, 4 or 8 assigned, as shown.

The division in this manner is not a part of the IEEE 488 standard. However, the division is a convenient way to clarify the conversion of line signals to hexadecimal numbers, and the illustration gives readers another perspective for binary representation of the bus data.

To understand how electrical signals are placed simultaneously on these wires, one has to visualize that, due to these signals traveling on these wires at the speed of light, each line, as it is energized, becomes energized from one end to the other. Therefore, a logic level (high or low) on each line can be shown to indicate the presence or absence of an electrical signal.

The high and low values on the data bus in Figure 2-2 (a) can be interpreted as an 8-bit binary number, with LO=1 and HI=0.*

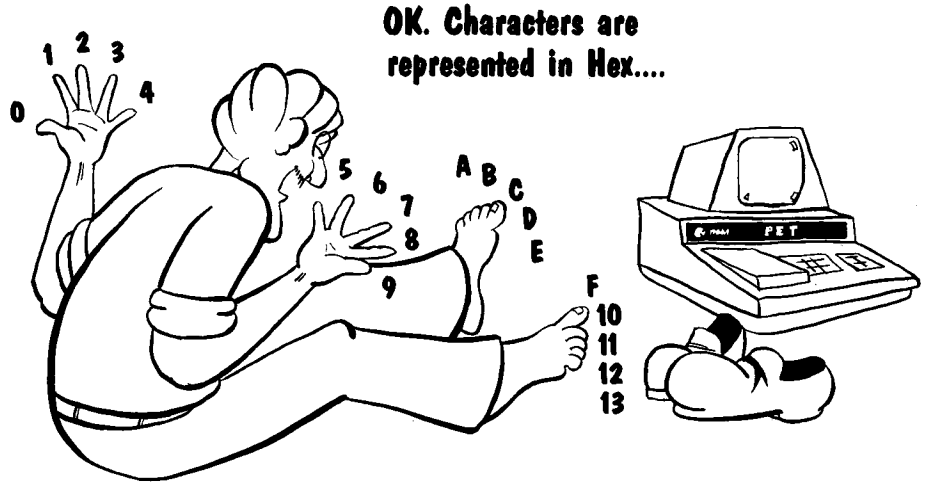
*The actual voltage levels are described in Chapter 4, Electrical Features.

Table 2-2. Number conversions

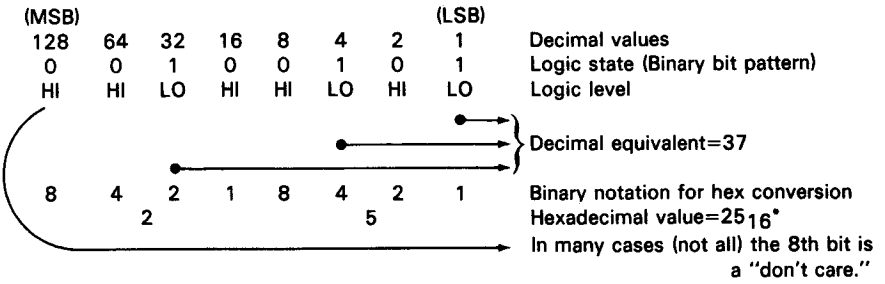
Binary	Decimal	Hexadecimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

The correspondence between binary, decimal and hexadecimal numbers for all of the hexadecimal digits is shown in Table 2-2. Note that binary-hexadecimal conversions are straightforward. Extended decimal-hexadecimal conversion tables are given in Appendix F.

In this book, numbers with no subscripts are always decimal numbers (for example, −123, 256). Binary numbers are denoted by a subscript of 2 (for example, 1101_2), and hexadecimal numbers are denoted by a subscript of 16 (for example, $1E_{16}$).



The decimal number 37, for example, can be converted to the following binary bit pattern:



These are the values shown in Figure 2-2 (a) and illustrate what is taking place electrically on the bus, although their hexadecimal values are generally referred to throughout the following text.

Figure 2-2 (b) illustrates the byte serial concept of alphanumeric characters (numbers and letters) being transferred on the Data Bus. In this example, you can visualize "flashes" of serial bytes, each composed of parallel 8-bit binary digits (bytes), streaming down the bus one after the other (from left to right) as the letters "GENE" are transmitted from the talker to a listener.

*Hexadecimal notation is not a part of the IEEE 488 Standard, but is used in this book for convenience.

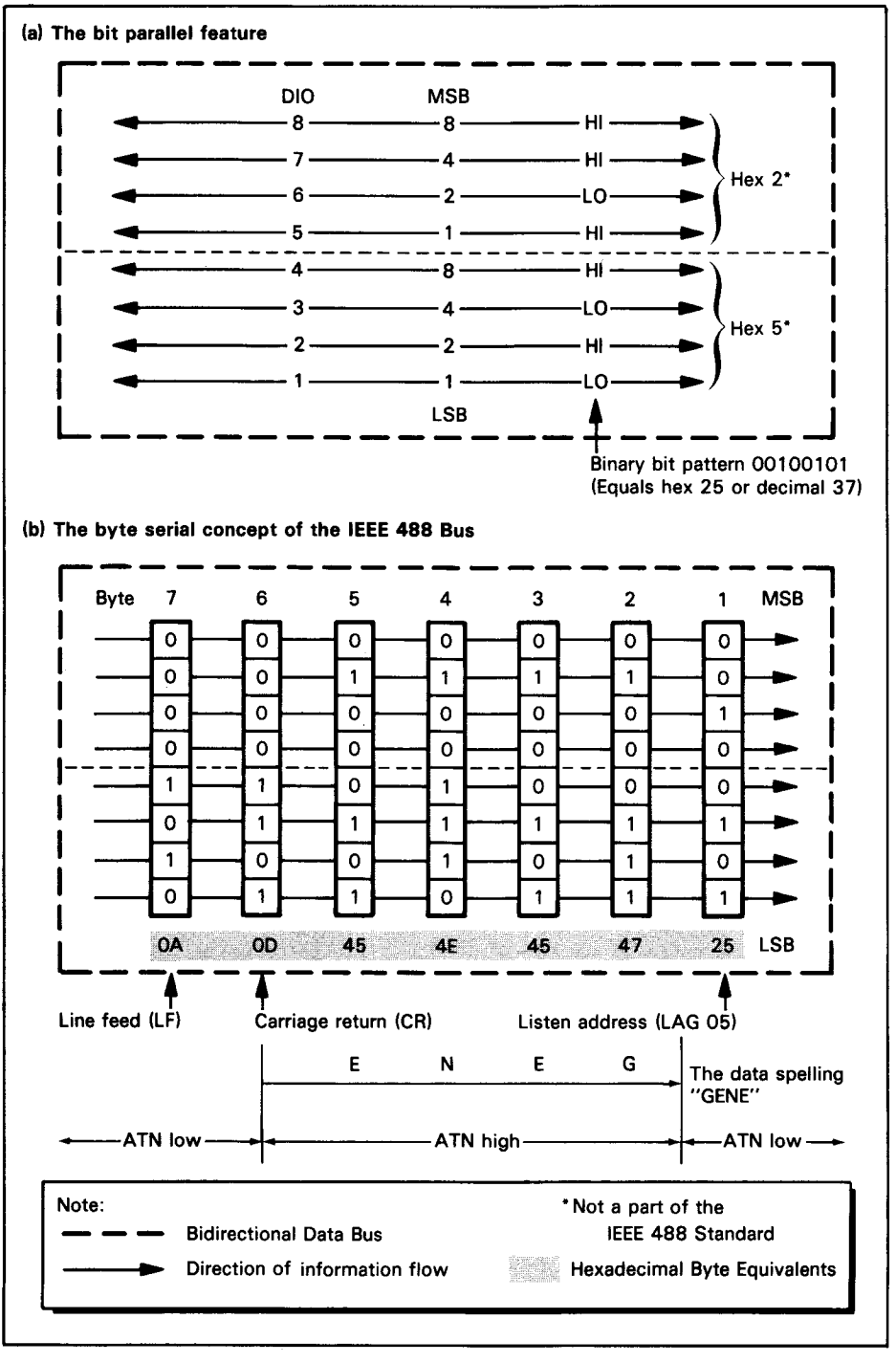
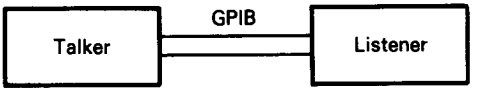


Figure 2-2. GPIB data bus

THE HANDSHAKE PROCEDURE*

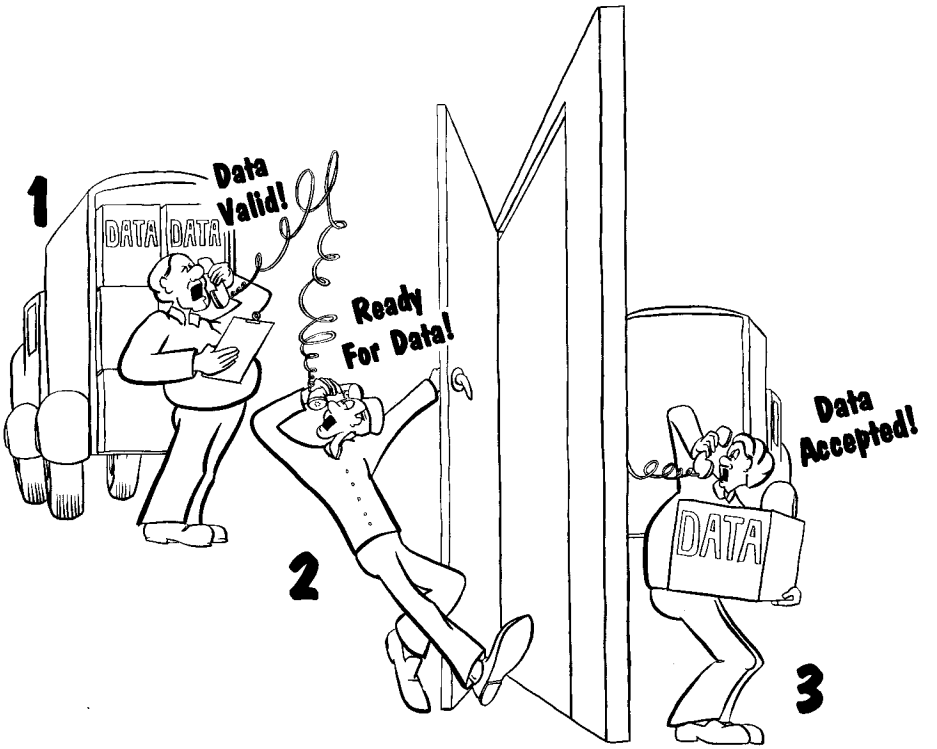
HOW THE HANDSHAKE PROCEDURE WORKS — A SIMPLIFIED VERSION

The entire bus can be visualized as a single communication link between one talker and (at least) one listener.



To communicate, the talker must let the listener know when data is available on the bus, and the listener must recognize that fact. Also, there must be an agreement as to which device — talker or listener — will activate the control lines.

The three lines that do the communication handshaking between the controller (talker) and the listener are DAV (Data Valid) NRFD (Not Ready For Data), and NDAC (Not Data Accepted). A simplified handshake timing diagram for these three lines is shown in Figure 2-3.



*Licensing details for the 3-Wire handshake may be obtained from the Hewlett-Packard Company, 1501 Page Mill Road, Palo Alto, CA 94304.

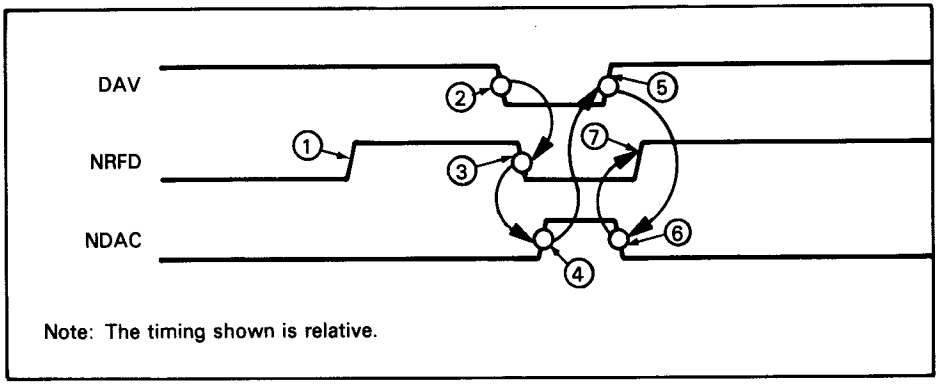


Figure 2-3. Simplified GPIB handshake timing diagram

The DAV line belongs to the controller (talker); it identifies valid data. The two remaining lines, NRFD and NDAC, belong to the listener. The listener uses NRFD to indicate when it is ready or not ready for data. The listener uses NDAC to indicate when it has received and accepted the data.

Assuming that the addressing information has already been transferred down the bus to a listener, the talker knows it's talking, and the listener knows it's going to listen, the listener raises its NRFD line ① (Figure 2-3) saying, in effect, that it is now ready to accept data that the talker will send down the bus. The talker then places its data on the 8-bit parallel GPIB data lines. After allowing enough time for the data to settle on the line, the talker drops its DAV line ② to a low (true) state, saying that the data is now available for the listener.

Upon sensing DAV low (true), the listener answers by lowering its NRFD line ③ when it is ready to accept data. After the listener has strobed the data into its internal buffers, it raises its NDAC line ④ saying that it has accepted the input data from the bus.

The talker, sensing that the NDAC line has been pulled high (false), raises its DAV line ⑤ to indicate to the listener that the data on the bus is no longer valid. When the listener detects the DAV line high (false) again, it drops NDAC low ⑥ , acknowledging the fact that data is being removed from the bus. The listener then raises NRFD high (false) ⑦ , saying it is ready for the next data byte to be sent down the bus.

The sequence is now complete. The listener is waiting for the next data byte to be put on the GPIB, and the talker is processing the next data byte before placing it on the bus. This sequence is the essence of all communication up and down the bus; it allows for signal propagation delays and processing time required by either talker or listener.

HANDSHAKE SEQUENCE FOR THE PET AS A CONTROLLER/TALKER

We will now describe how the PET meets the handshaking and data timing requirements for the GPIB. The accompanying flowchart (Figure 2-4) shows every bus transaction, including both address and data handshaking. This will help you check procedures for transferring data and locate instrument compatibility problems.

The flowchart in Figure 2-4 begins with the terminal box labeled "PET TALKER." As the flowchart proceeds from one step to the next, the decision and action points that the PET looks at or interrogates are numbered. These numbers are cross-referenced to the same numbers with arrows leading to the resulting action on the accompanying timing diagram.

The action of box "SET DAV=HIGH" is shown on the timing diagram at ① . This logic level may already exist; however, the PET sets DAV high (false). Next, the box "ARE NDAC AND NRFD=HIGH?" is shown at ② . The PET samples both the NDAC and NRFD lines to see if they are both high (false). If they are, this is an error condition, as shown by the "YES" branch on the flowchart. Status bit 7 is set (ST=-128) and the operations box "SET STATUS" indicates a device is not present. The normal handshaking sequence is terminated as indicated by the box "EXIT."

Again at point ② on the timing diagram, when NRFD and NDAC are sampled by the PET, if either line is low (true), then the "NO" branch from the decision box is taken. This branch leads to the operations box "DATA TO THE GPIB"; the result of this step is indicated on the timing diagram at ③ , where data, either an address or actual data, is placed on the GPIB.

The next decision box, "IS NRFD=HIGH?", indicates that the PET waits for the NRFD line to go high (false). If it is not, as shown by the "NO" branch, the PET is stuck in a very tight loop waiting for the NRFD line to go high (false); no error condition is sensed by the PET. Unless NRFD goes high, the PET is hung in this very tight loop until the GPIB cable is removed or the power to the PET is removed and reapplied. The sequence would then be run through again. At the point when NRFD is high, as shown on the timing diagram at ④ , the "YES" branch leading from the decision box is taken and the operation continues.

At point ⑤ in the timing diagram, the DAV line is set low (true) by the PET, as indicated by the flowchart operations box "SET DAV=LOW." The listener is now expected to accept the data, and the PET allows 65 milliseconds for the operation to take place, as indicated by the flowchart operations box "SET TIMER=65 MS." This time delay is set up in the PET software; the PET checks the NDAC line logic level continuously until either the NDAC line is pulled high (false) by the listener within the allotted 65 ms time or the time for the listener to answer runs out.

Whether the counter has counted down yet or not is shown by the "YES" and "NO" branches, respectively, leading from the decision box "IS TIME UP?".

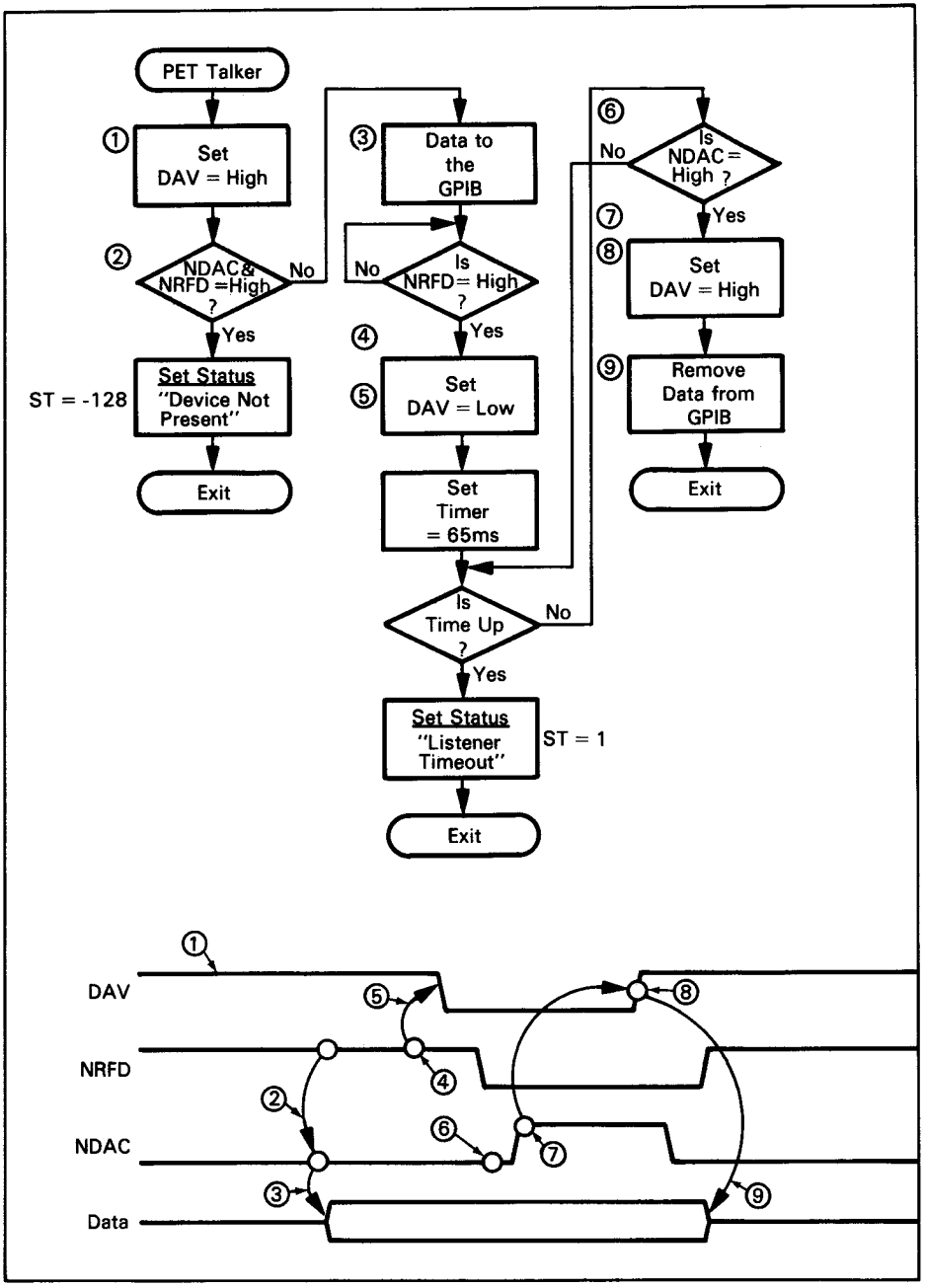


Figure 2-4. Flowchart and handshake timing diagram for the PET as a controller/talker

Examine the "NO" branch from the decision box "IS NDAC=HIGH?". If the listener has not reset NDAC high within the allotted time, then the PET runs in the program loop shown at ⑥ . This condition compares to NDAC remaining low (true) at point ⑥ on the timing diagram. If the listener has not reset its NDAC line high (false) within the allotted time, then status bit 0 is set (ST=1) and the operations box "SET STATUS" indicates the listener has timed out. The timing and handshake sequence terminates as indicated by the box "EXIT."

If, within the allotted 65 ms, the listener does respond by pulling its NDAC line high (false), the program continues to execute as shown by ⑦ on the timing diagram. This action corresponds to the "YES" branch leading from the decision box "IS NDAC=HIGH?".

After the listener raises its NDAC line high, the PET answers at time ⑧ by pulling the Data Valid line DAV high (false). This response by the PET corresponds to the action shown in the next operations box, "SET DAV=HIGH."

As indicated by the box "REMOVE DATA FROM GPIB" and the timing diagram at point ⑨ , the PET removes data from the DIO lines 1-8 and concludes the handshake sequence as indicated by the termination box "EXIT."

This flowchart and timing diagram have some interesting aspects for the experimenter and the interface designer. You can interrupt the normal handshake procedure, stop the PET, and step it one GPIB instruction at a time. For example, at the second decision box (the one labeled "IS NRFD=HIGH?"), the handshake procedure can be stopped by controlling the NRFD line. You can enter this timing sequence with a time-controlled interface that brings the NRFD line low following the placement of data (either an address or actual data) on the data bus. Then the PET continuously presents its data to the GPIB, and you can examine the logic levels of the eight data lines externally.

Contrary to the above, you cannot control the NDAC line and stop the listener device during the handshake sequence. This can be seen by referring to the flowchart, which shows that the PET will time the listener and set a timeout status if the allotted time expires without pulling NDAC high.

HANDSHAKE SEQUENCE FOR THE PET AS A CONTROLLER/LISTENER

The flowchart in Figure 2-5 describes how the PET, while it is in listen mode, inputs data placed on the GPIB by a talker. The chart begins after the PET has defined an external instrument as a talker and is waiting for input from the bus. Again, as the flowchart proceeds from step to step, the numbered areas (the decision and action points) that the PET looks at or interrogates are indicated on the timing diagram by an identical circled number with an arrow leading to the resulting action.

The flowchart is identified by the termination box "PET LISTENER." The chart begins by the PET setting NDAC low (true), which says the PET has not accepted data. (Note that "data" can mean address information or actual data.) This is indicated on the timing diagram at ① and on the flowchart as the operations box "SET NDAC=LOW."

Next, the PET sets the NRFD line high (false) indicating that it is now ready to accept data. This is shown at ② , and in the flowchart by the operations box "SET NRFD=HIGH." With NRFD high (false) and NDAC low (true), the talker knows it can assert the data, which it does at ③ .

The PET software now sets up and specifies a time frame of 65 ms, during which the talker device must transmit its data to the PET. This action is represented in the flowchart by the operation box "SET TIMER=65 MS."

The elapsed time of 65 ms is controlled as signified by the decision box "IS TIME UP?".

If the talking device does not answer within the 65 ms time frame, indicated by setting its Data Valid (DAV) line low (true), then the PET will time out as represented by the "YES" branch from the decision box. This branch leads to the operations box "SET STATUS 'TALKER TIMEOUT'," and status bit 1 is set (ST=2), causing "DEVICE NOT PRESENT" to be displayed on the CRT screen. When this action occurs, the program terminates at the "EXIT" box.

If the elapsed time has not reached 65 ms, this is indicated by the "NO" branch of the decision box. This branch enters another decision box in the flowchart titled "IS DAV=LOW?". Within the 65 ms time frame, if the DAV line has not been set low (true), the PET will hang up in a loop represented by the arrow coming from the "NO" branch. When this "NO" branch is entered, DAV is at a high (false) logic level; the program will continue in this loop while the PET waits for the talking device to lower its DAV line, indicating that data is valid and stable and that the PET is to receive the data.

When the talking device finally does answer by setting its DAV line low (true), this action is denoted by the "YES" branch of the box "IS DAV=LOW?" and at ④ in the timing diagram. The PET then acknowledges by setting the NRFD line low (true), time ⑤ . This is referenced in the flowchart by the operations box "SET NRFD=LOW."

At ⑤ on the timing diagram, the NRFD line is asserted low (true), which says the PET is no longer ready for data. Meanwhile, at time ⑥ the PET is in the process of reading the information from the data lines DIO1-8. In addition to sampling the data, the PET also looks at the End Or Identify (EOI) line, ⑦ on the timing diagram. This signal is the status bit initialized by the talker to signify the end of a data byte. Normally EOI is brought low (true) by a talker during the last data byte being transmitted on the GPIB; however, all bus-compatible instruments do not necessarily generate an EOI signal.

The decision box "IS EOI SET?" has "YES" and "NO" branching points. As the PET samples the EOI line, if it finds this line low (true), the PET sets status bit 6 (ST=64) in the status word. This action is indicated by the "YES" branch and the operations box "SET STATUS 'EOI IS ON'."

If the PET finds EOI high (false), it takes the "NO" branch, bypassing the Set Status box. Additional bytes of parallel data will continue to be transferred from the talker to the PET until the last data byte is transferred and EOI is asserted by the talker.

While EOI is being sampled, the PET reads GPIB data at ⑧ on the timing diagram. This operation is represented by the box "READ GPIB DATA." The PET then resets its NDAC line high (false) as indicated at ⑨ on the timing diagram and the flowchart box "SET NDAC=HIGH." This latter communication between the two devices satisfies the talker that the PET has accepted data off the bus.

After the PET controller has pulled its NDAC line high (false), the talker is free to remove its data from the GPIB. However, the controller waits for this operation as indicated by the decision box "IS DAV=HIGH?". Until the talker resets its DAV line high (false), as shown at ⑩ on the timing diagram, the PET controller runs in a very tight loop represented by the arrow leading from the "NO" branch of the decision box.

When DAV is pulled high (false) at ⑩ by the talker, the PET processes the data just received from the bus and sets its NDAC line low (true) as indicated by ⑪ on the timing diagram and the flowchart operations box "SET NDAC=LOW." Here the sequence concludes at the termination box "EXIT."

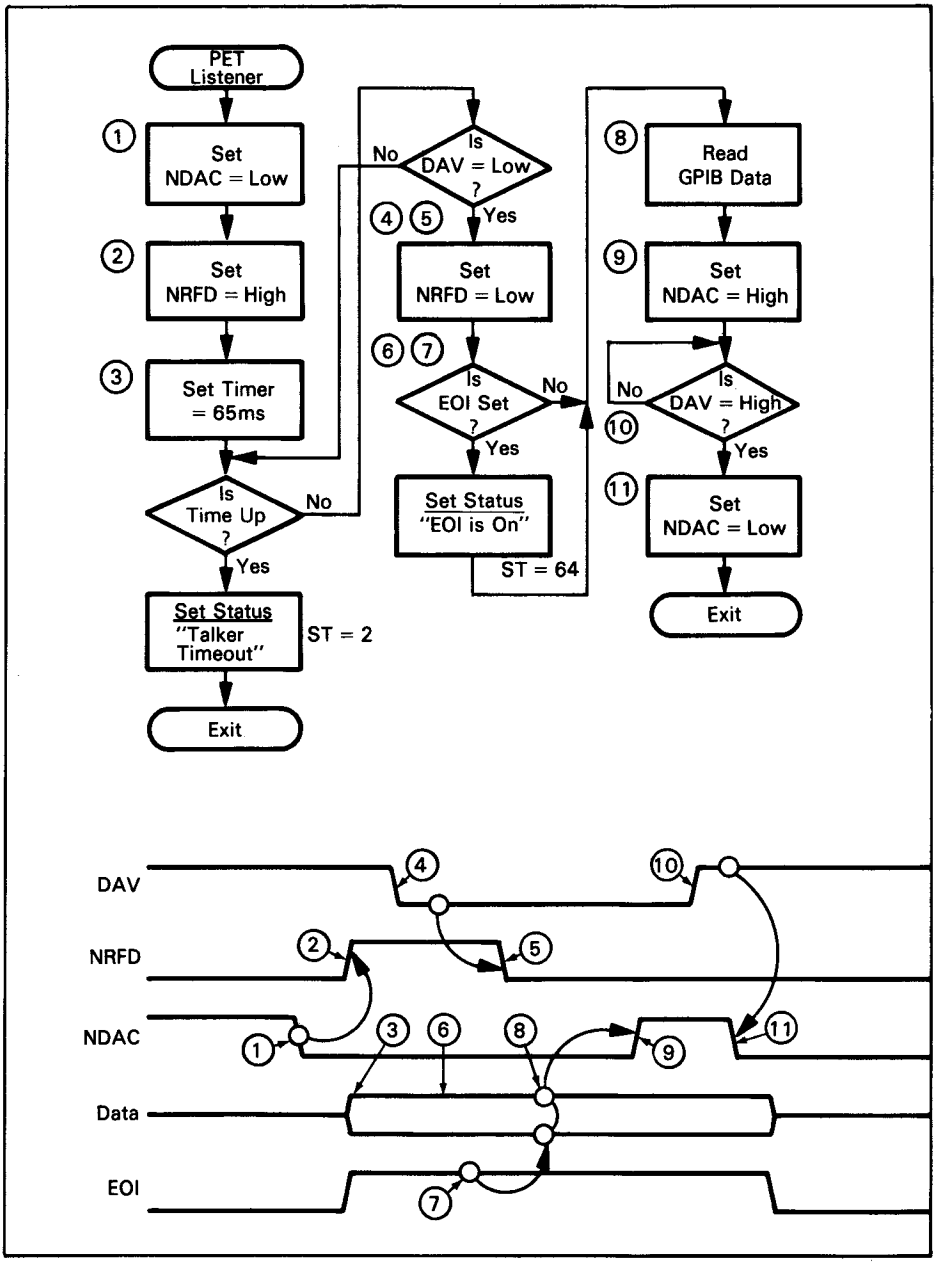


Figure 2-5. Flowchart and handshake timing diagram for the PET as a controller/listener

ACCESSING THE GPIB FROM PET I/O MEMORY

All data transfers, whether initiated by PET BASIC statements, assembly language instructions or other means, are propagated to the GPIB data lines by placing the data into a dedicated location in the PET memory space. Other dedicated I/O locations provide access to all of the control lines except REN, which is held permanently low by the PET, and IFC, which is asserted by the PET's reset hardware. The I/O memory locations assigned to the GPIB lines are listed in Table 2-3.

Table 2-3. PET hardware addresses for GPIB lines

GPIB Line		PET Memory Address		
Name	**Mode**	**Hexadecimal**	**Decimal**	**Bit(s)**
DIO1-8	Input	E820	59424	0-7
DIO1-8	Output	E822	59426	0-7
DAV	Output	E823	59427	3
DAV	Input	E840	59456	7
NRFD	Output	E840	59456	1
NRFD	Input	E840	59456	6
NDAC	Output	E821	59425	3
NDAC	Input	E840	59456	0
ATN	Output	E840	59456	2
ATN	Input	E821	59425	7
SRQ	Input	E823	59427	7
EOI	Output	E811	59409	3
EOI	Input	E810	59408	6

DATA LINES

The PET is normally programmed to send data to, or receive data from, the GPIB data lines DIO1-8 using INPUT, PRINT and other I/O statements as described in Chapter 4. However, you can use PEEK and POKE commands into I/O memory to access individual lines on the GPIB, as described here.

These data lines can be addressed and exercised by the simple programs outlined below. While exercising the data lines in these procedures, remember that to actually transfer data a remote, standard, bus-compatible device would have to be on the bus, and IEEE Std 488 handshake protocol would have to be observed.

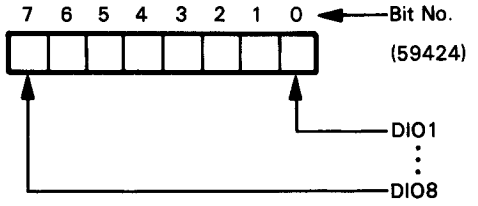
Input Data Location (59424)

The data bus is addressed for input with the PET memory address:

Hexadecimal	Decimal
E820	59424

This input represents data on the bus whether it comes from the PET or from an external device holding down the data lines.

The GPIB data in the PET is represented in the data input register (location 59424) below, having data bits 0 to 7 (right to left).



In the sample program below, the line 10 statement peeks a value from the data input register into the A variable. Line 20 then prints (displays) the A variable.

```
10  A=PEEK (59424)
20  PRINT A
```

Data on the GPIB comprises signals that are low true. If the above program yielded, for example, a value of 128 (10000000_2), the data on the GPIB, in terms of "true" data, would be 127 (01111111_2). Thus, the value displayed by the program will be the complement of the data on the bus.

Summarizing:

Decimal	PET Binary (high true) If program display is:	GPIB Binary (low true) The data on the bus is:
128	10000000	01111111
127	01111111	10000000

Output Data Location (59426)

The data bus is addressed for output with the PET memory address:

Hexadecimal	Decimal
E822	59426

This outputs data from the PET to the GPIB. As before, the data output register handles data to the GPIB in the form bit 7=MSB...bit 0=LSB.

The following sample program outputs data to the GPIB data bus:

```
5  A=96
10 POKE 59426, A
```

This program assigns a value of 96 to the A variable, then outputs the data to the GPIB with the POKE statement. The value 96 (01100000_2) will send 159 (10011111_2) as the data byte to the GPIB.

CONTROL LINES

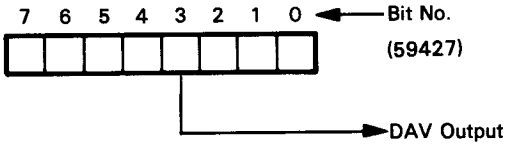
The programs below show how you can examine and change the logic state of each control line while keeping all others at their normal logic level. These operations are intended to illustrate how you can access individual lines from the PET and do not represent an IEEE 488 Bus implementation.

DAV Locations (59427, 59456)

To exercise the DAV line in its output mode, use memory address:

Hexadecimal	Decimal
E823	59427

The DAV output connects to bit 3 of location 59427.



This bit can be toggled from 1 to 0 to 1 logic levels by the following sample program:

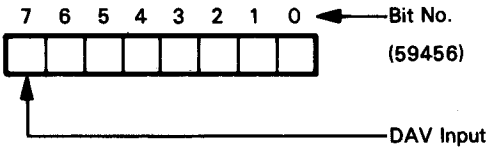
```
10 POKE 59427, 60:REM SETS DAV HIGH
20 POKE 59427, 52:REM SETS DAV LOW
30 POKE 59427, 60:REM SETS DAV HIGH
```

Similar programs that follow are toggled in a like manner.

To examine the DAV line in its input mode, use memory address:

Hexadecimal	Decimal
E840	59456

The DAV input connects to bit 7 of location 59456.



To examine the logic state of the DAV line in its input mode, use the following sample program:

```
PRINT PEEK (59456):REM LOOKS AT BIT 7
```

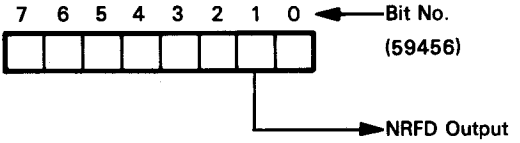
This statement causes a decimal number to appear on the PET's CRT. By converting this number to binary, you can determine the logic state of bit 7. Use the same procedure for similar programs that follow.

NRFD Location (59456)

To exercise the NRFD line in its output mode, use memory address:

Hexadecimal	Decimal
E840	59456

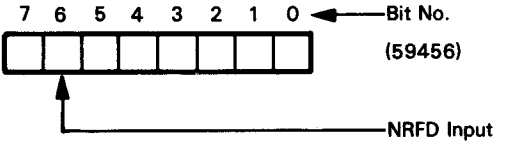
The NRFD output connects to bit 1 of location 59456.



This output bit can be toggled by the following sample program:

```
10 POKE 59456, 255:REM SETS NRFD HIGH
20 POKE 59456, 253:REM SETS NRFD LOW
30 POKE 59456, 255:REM SETS NRFD HIGH
```

The NRFD input connects to bit 6 of this same location (59456).



To examine the logic state of the NRFD line in its input mode, use the following sample program:

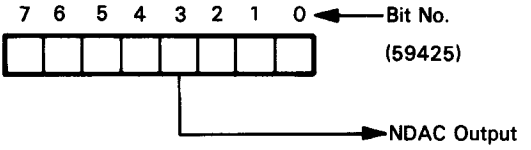
```
PRINT PEEK (59456):REM LOOKS AT BIT 6
```

NDAC Locations (59425, 59456)

To exercise the NDAC line in its output mode, use memory address:

Hexadecimal	Decimal
E821	59425

The NDAC output connects to bit 3 of location 59425.



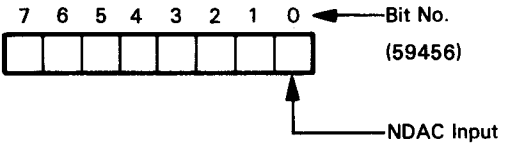
This output bit can be toggled by the following sample program:

```
10 POKE 59425, 60:REM SETS NDAC HIGH
20 POKE 59425, 52:REM SETS NDAC LOW
30 POKE 59425, 60:REM SETS NDAC HIGH
```

To examine the NDAC line in its input mode, use memory address:

Hexadecimal	Decimal
E840	59456

The NDAC input connects to bit 0 of location 59456.



To examine the logic state of the NDAC line in its input mode, use the following sample program:

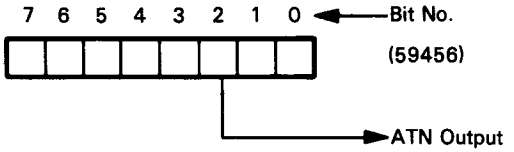
```
PRINT PEEK (59456):REM LOOKS AT BIT 0
```

ATN Locations (59456, 59425)

To exercise the ATN line in its output mode, use memory address:

Hexadecimal	Decimal
E840	59456

The ATN output connects to bit 2 of location 59456.



This output bit can be toggled by the following sample program

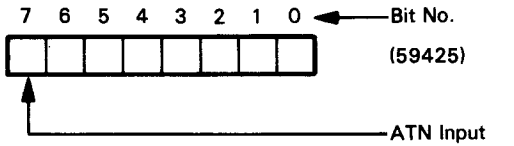
```
10 POKE 59456, 4:REM SETS ATN HIGH
20 POKE 59456, 0:REM SETS ATN LOW
30 POKE 59456, 4:REM SETS ATN HIGH
```

The ATN input is different from the other signals previously examined. Whereas the transfer control lines (DAV, NRFD and NDAC) are monitored in a low true or a high false state, the ATN line is monitored from a transition point of view. The Attention line must do a transition from a high to a low logic state before the signal will be received.

To examine the ATN line in its input mode, use memory address:

Hexadecimal	Decimal
E821	59425

The ATN input connects to bit 7 of location 59425.



To examine the logic state of the ATN line in its input mode, use the following program:

```
 5 A=PEEK (59424)
10 A=PEEK (59425)
15 B=A AND 128:IF B=0 GOTO 10
```

The sample program shows that it is necessary to clear location 59425 before testing bit 7 for a transition that will occur later. Statement 5 peeks location 59424 (the data input register) and clears any previous transitions that may have occurred prior to the time of interest. Statement 10 peeks the location, then tests it with an AND of 128, which tests bit 7. Statement 15 will then test the ATN bit 7 until a transition occurs. Then the program exits. Ground the ATN line externally to make the program stop.

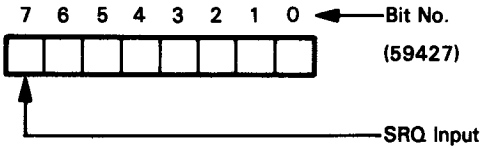
SRQ Location (59427)

The SRQ line has an input-only signal; there is no output from the PET. This line is similar to the ATN line in that bit 7 (of a different location) is tested for an input transition. The line must be cleared of the previous instruction so that a request can be acknowledged.

To exercise the SRQ line in its input mode, use memory address:

Hexadecimal	Decimal
E823	59427

The SRQ input connects to bit 7 of location 59427.



To examine the logic state of the SRQ line in its input mode, use the following sample program:

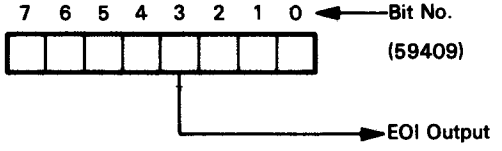
```
10 A=PEEK (59426):REM RESETS SRQ LINE
20 A=PEEK (59427):V=A AND 128
25 IF B=0 GOTO 20:REM LOOKS AT BIT 7
```

EOI Locations (59409, 59408)

To exercise the EOI line in its output mode, use memory address:

Hexadecimal	Decimal
E811	59409

The EOI output connects to bit 3 of location 59409.



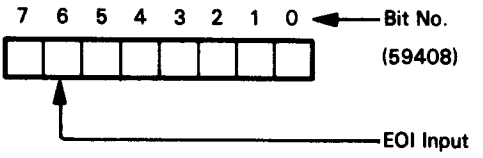
This output bit can be toggled by the following sample program:

```
10 POKE 59409, 52:REM SETS EOI LOW
20 POKE 59409, 60:REM SETS EOI HIGH
30 POKE 59409, 52:REM SETS EOI LOW
```

To exercise the EOI line in its input mode, use memory address:

Hexadecimal	Decimal
E810	59408

The EOI input connects to bit 6 of location 59408.



To examine the logic state of the EOI line in its input mode, use the following statement:

```
10 PRINT PEEK (59408):REM LOOKS AT BIT 6
```

CHAPTER 3

Hardware

THE PET-GPIB INTERFACE

Figure 3-1 is a simplified parts layout showing the portion of printed circuit board in the PET that has integrated circuits implementing the GPIB. This is for the PET with the 8K ROMS. The IC chips are the MC3446, 74LS04, and 7417 buffer chips, the 6520 peripheral interface adapters (PIAs) and the versatile interface adapter (VIA). The integrated circuits and their functions are described in more detail in the paragraphs that follow. Connector J1 is the IEEE 488 (GPIB) port, an extension of the PC board itself.

DATA BUS INTERFACE CIRCUITS

Figure 3-2 shows the interfacing circuits between the PET and the data input/output lines of the GPIB. These circuits comprise the MC3446 buffers A7 and A8, and the 6520 PIA, B8.

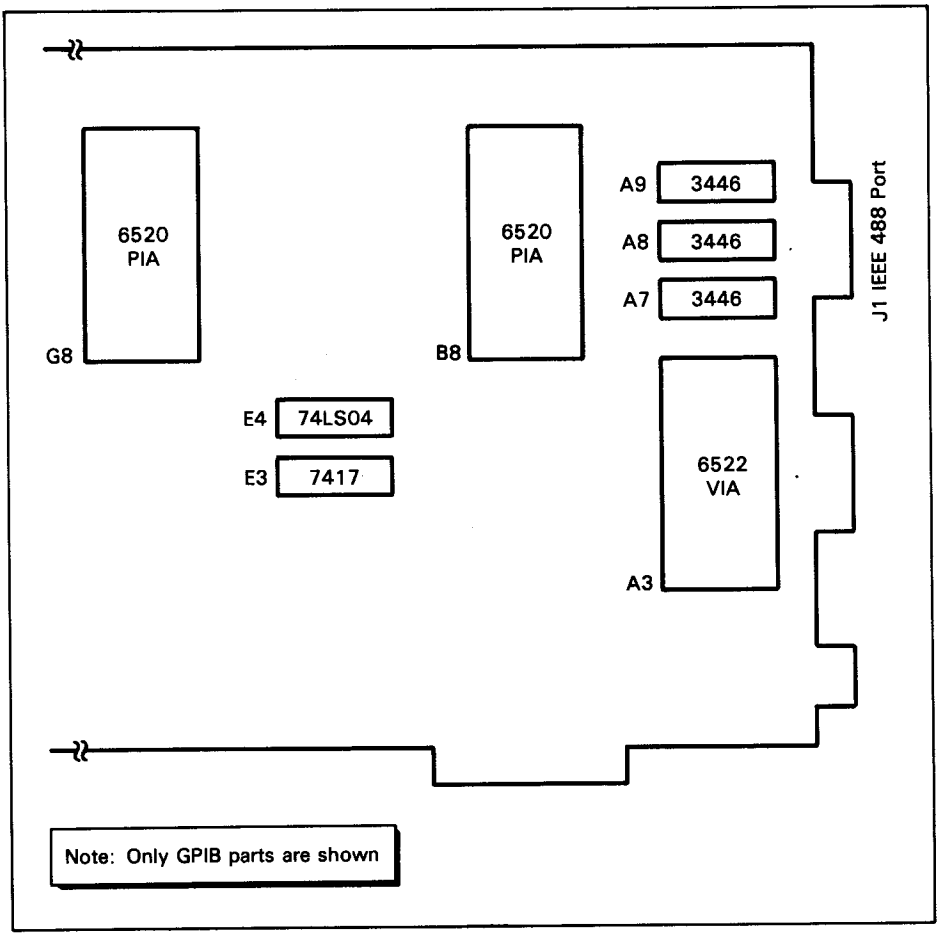


Figure 3-1. The GPIB component locations on the PET PC board having old ROMS (top view)

MC3446 Buffers

The two integrated circuits A7 and A8 provide the IEEE compatibility for the PET data bus. They furnish both circuit termination and buffering that the IEEE Std 488-1978 specification requires. This standard defines certain critical specifications, all of which are met by the MC3446 circuits.

Data lines must be terminated at an interface in each instrument or controller. (See "Electrical Features" for this and other IEEE Std 488-1978 specifications.) This particular specification is met by the 2.4K pull-up resistor and the 5K pull-down resistor shown in A7 of Figure 3-2. This is a typical termination for each data input/output line.

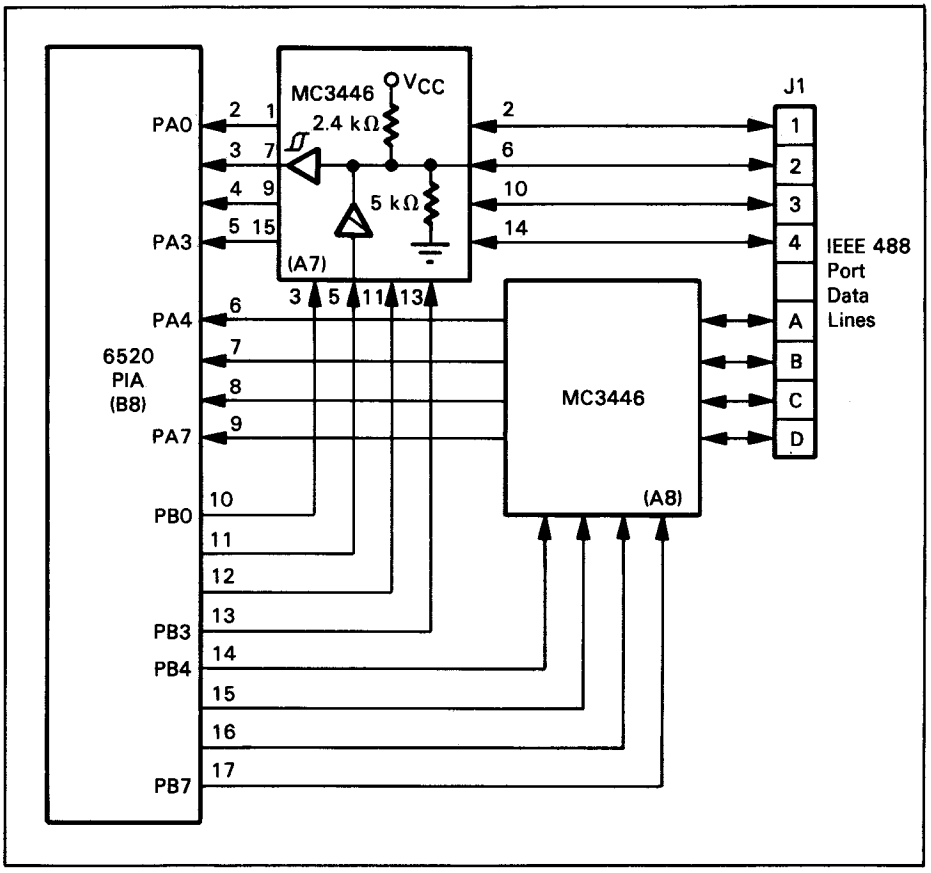


Figure 3-2. PET-GPIB interface diagram showing data lines

Data line drivers (output buffers) must have a drive capability with sufficient current capacity to feed the parallel combination of the terminating devices in up to 15 instruments, the maximum that can reside on the bus. This particular integrated circuit, MC3446, meets these requirements, as it has a drive capability of 48 mA at a guaranteed low output voltage of not less than 0.8 volts. The input circuitry of this chip is high impedance, thus offering very light loading on the 6520 PIA as well as to the remainder of the GPIB system. This "receiver" has a built-in hysteresis that tends to reduce any noise that might be injected into the bus lines.

Instrument interface buffers in a GPIB application must not load the bus when power is removed from the circuit. These circuits abide by this restriction.

The MC3446 circuits are non-inverting from input to output. This is an important feature, since the assertive (true) signal on the GPIB is a low (1) value. Because of this, a 0 must be written into the PIA chip to put a low (1) on the GPIB data bus.

Peripheral Interface Adapter (PIA)

Figure 3-2 also shows the peripheral interface adapter (PIA), B8, type 6520.* The function of this input/output circuit is to provide an input buffer and an output storage latch for the GPIB data. This adapter, along with the MC3446 buffers, constitutes the overall interface between the microcomputer inside the PET and the GPIB data lines.

The PIA is, in essence, a parallel port that is configured by PET software to send and receive parallel data to and from the data bus. Being a parallel device, the PIA can latch the parallel data from four input buffers (the four are contained in the two blocks labeled A7 and A8) to the 6502 microprocessor. The operational setup of the PIA is handled automatically by the PET operating system. You can access the PIA to perform other data transfer operations, but such special access of the PIA is not described in this book.

The PIA is located electrically within the address space of the microprocessor; it is accessed via memory locations 59392 and 61439 ($E800_{16}$ and $EFFF_{16}$).

CONTROL LINE INTERFACE CIRCUITS

Control signals for instruments on the GPIB are sent via the transfer control and interface management lines.

The PET controls the protocol in the handshake procedure, and implements it with the hardware in Figure 3-3.

In Figure 3-3, three PIA-type components in the PET handle the eight transfer control and interface management lines. DAV, NRFD, NDAC and ATN (upper right) are passed through the MC3446 (A9) buffer to maintain compatibility with the GPIB. These four lines can be thought of as complete or "full" handshake lines.

IFC and EOI (bottom right) have a part, however, in the handshake procedure, and they retain their GPIB compatibility because each line passes through its own termination circuitry.

IFC derives its signal from the PET power-up reset signal RES, which is buffered through a 74LS04 (E4) and used only at power-up time. No read interrogation of this signal is readily available.

Although driven with a signal from the PIA (G8), EOI can also be read directly into the same PIA chip from the GPIB. Because the EOI line, when considered as an input line, connects directly to the PIA (see Figure 3-3), it is particularly sensitive to high external voltages.

SRQ and REN, normally a part of the IEEE Std 488 handshake procedure, have a non-standard implementation for PET-GPIB operation.

The SRQ line is read into the PET via the PIA (B8) circuit identified as CB1. SRQ is monitored by the PET, never driven, as this is a service request line

*For a thorough description of this part, see An Introduction to Microcomputers: Volume 2 — Some Real Microprocessors. Osborne/McGraw-Hill, 1978.

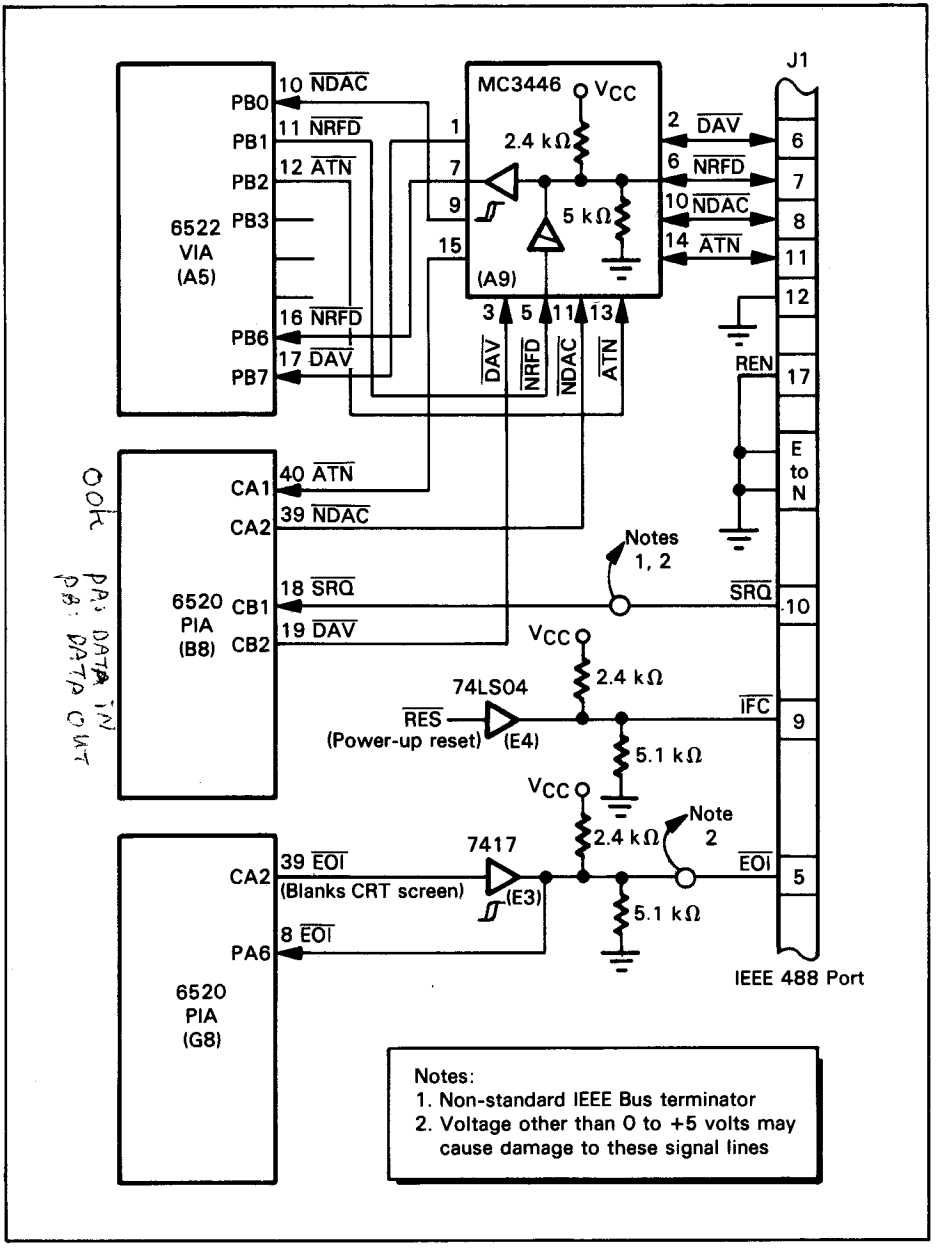


Figure 3-3. PET-GPIB interface diagram showing control lines

asserted only by external devices.

The REN line is grounded in the PET, thereby placing GPIB instruments in the remote-enable configuration.

ELECTRICAL FEATURES

The electrical specifications in the IEEE Standard 488-1978, which are based on TTL technology, define electrical parameters such as data rate, bus length, voltage, and current levels at the interface connector.

DATA RATE

The data rate sent on any signal line must be less than one megabit per second. Practical implementations of the bus operate from 5000 to 300,000 bits per second.

NUMBER OF DEVICES AND CABLE LENGTH

The bus can have 15 or fewer devices connected to it at any given time. The inter-device cable length cannot exceed 4 meters. The total transmission length of the bus cannot exceed 2 meters times the number of connected devices, or 20 meters (65.6 ft.), whichever is less.

BUS RECEIVERS AND DRIVERS

Although the IEEE Std 488-1978 does not specify types of bus signal receivers or drivers, if these are designed with TTL logic, the specified signal levels can easily be achieved while meeting other requirements of the GPIB system.

Signals are entered onto the bus as either active true or passive true values. By having open-collector drivers send signals into a terminated bus, active true values (the low state) can be made to override passive true values (the high state). The bus interface is so designed that conflicts between two devices trying to send opposite messages simultaneously over the bus can be resolved. The technique of active values overriding passive values is called the active transfer of a message. The specifications for such bus line receivers and drivers are shown in Table 3-1.

Table 3-1. Specifications for GPIB line receivers and drivers

GPIB Input (Receiver)	GPIB Output (Driver)	True/False	Logic State	Voltage or Current	
				Minimum	Maximum
V low		True	1	-0.6 V	+0.8 V
V high		False	0	+2.0 V	+5.5 V
	V low	True	1	0.0 V	+0.4 V
	V high	False	0	+2.4 V	+5.0 V
I low		True	1		-1.6 mA
I high		False	0		+50 μA
	I low	True	1	+48 mA	
	I high	False	0		-5.2 mA

While most bus signal driving circuits can be designed with tristate logic to attain higher data rates, the open collector configurations must be designed into drivers for signal lines SRQ, NRFD and NDAC. Also, during parallel polling, data lines DIO1-8 must be driven with open-collector drivers.

TYPICAL BUS-DEVICE INTERFACE

Figure 3-4 is typical of the interface between the GPIB and bus devices.

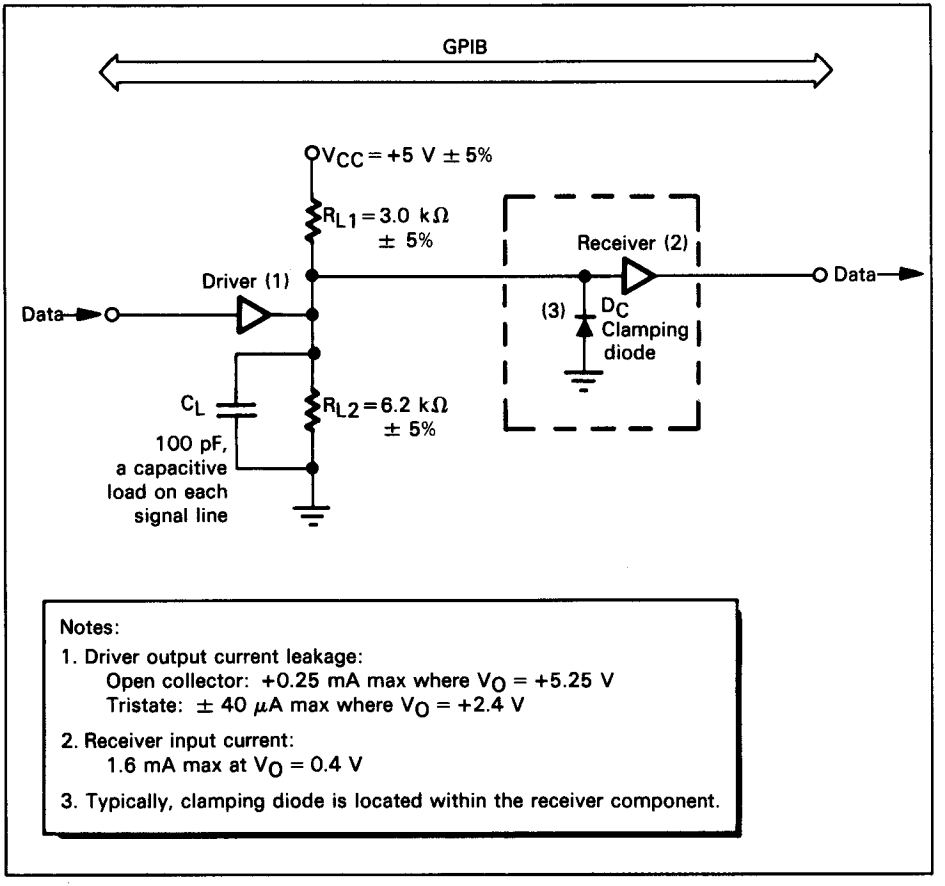


Figure 3-4. A typical GPIB-bus device interface

MECHANICAL FEATURES

COMPLETE CABLE ASSEMBLIES

Two types of connectors are necessary to interface GPIB peripheral devices with the PET. The first is a receptacle to fit the special PET edge-card connector (plug), a direct extension of the main logic card assembly in the PET computer. The second connector is the standard IEEE Std 488 connector to interface remote devices.

Complete cable assemblies are available from the manufacturers listed in Table 3-2. We recommend these cables with connectors since they are preassembled.

The IEEE 488 cable assembly is shown in Figure 3-5. A cable assembly for connecting the PET to the IEEE 488 Bus is shown later in this chapter (Figure 3-13).

Table 3-2. Cable assemblies

Manufacturer	Cable Part Number	Cable Description	Type
Belden	9642	1 m (3.3 ft.) cable with connectors	IEEE 488 Bus
	9643	2 m (6.6 ft.) cable with connectors	
	9644	4 m (13.1 ft.) cable with connectors	
	9645	8 m (26.2 ft.) cable with connectors	
Commodore	320101	PET IEEE Cable	PET to IEEE Bus
Pickles and Trout	PET-488	Cable Assembly	PET to IEEE Bus

CUSTOM CABLE ASSEMBLIES

You can make a cable assembly by putting together a plug/receptacle to fit the J1 edge connector on the PET and a GPIB bus connector on the other end of a cable. The cable must be the twisted pair type as specified in the IEEE Std 488-1978.

PET J1 Connectors

For the first connector, the interfacing connection to the PET, four known manufacturers' receptacles are available. These are shown in Table 3-3.

Any one of these receptacles can be inserted through a slot and attached to J1 to gain access to the IEEE Std 488 (GPIB) interface signals. The rear view of the PET and the location of the IEEE port is shown in Figure 3-6.

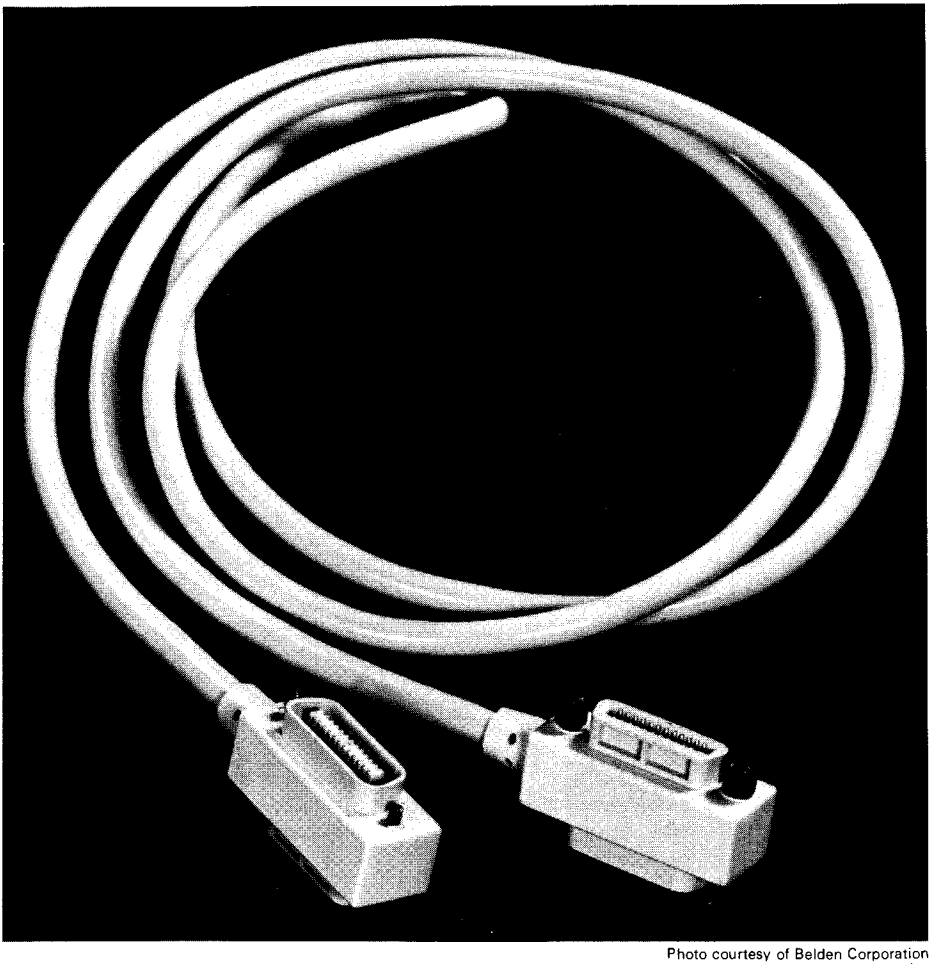
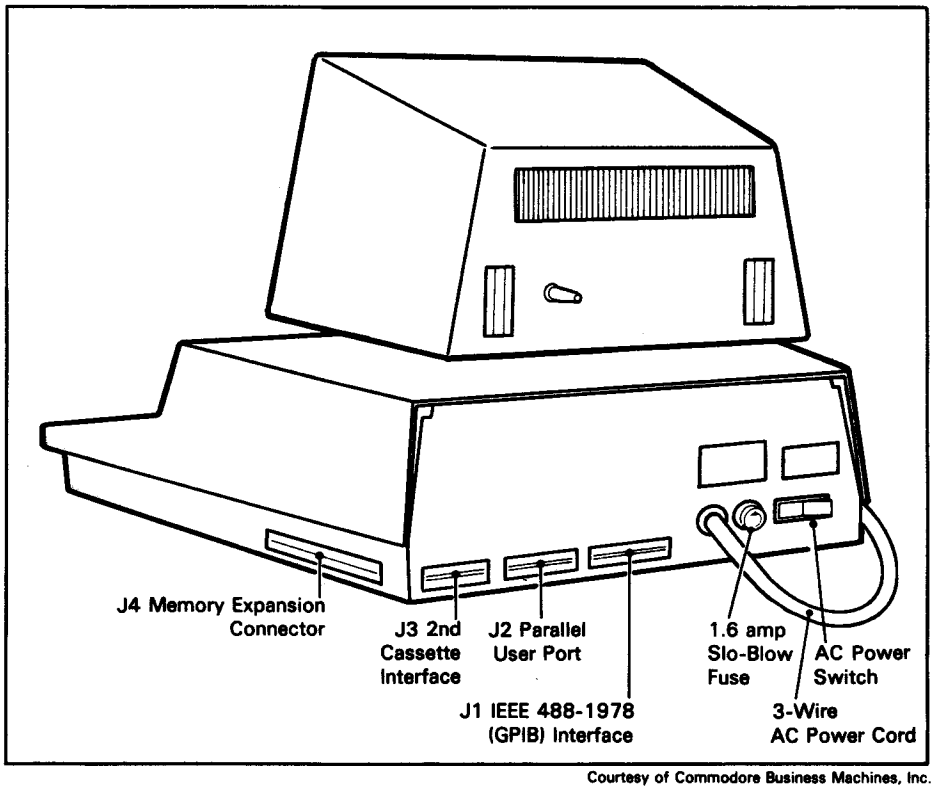


Photo courtesy of Belden Corporation

Figure 3-5. Belden IEEE 488 General Purpose Interface Bus cable

Table 3-3. A partial listing of manufacturers' receptacles that fit the J1 edge connector in the PET

Manufacturer	Plug/Receptacle Part Number
Amphenol	530654-3
Amphenol	530657-3
Amphenol	530658-3
Cinch	251-12-90-160
Sylvania	6AG01-12-1A1-01
TRW	50-24A-30 7-47

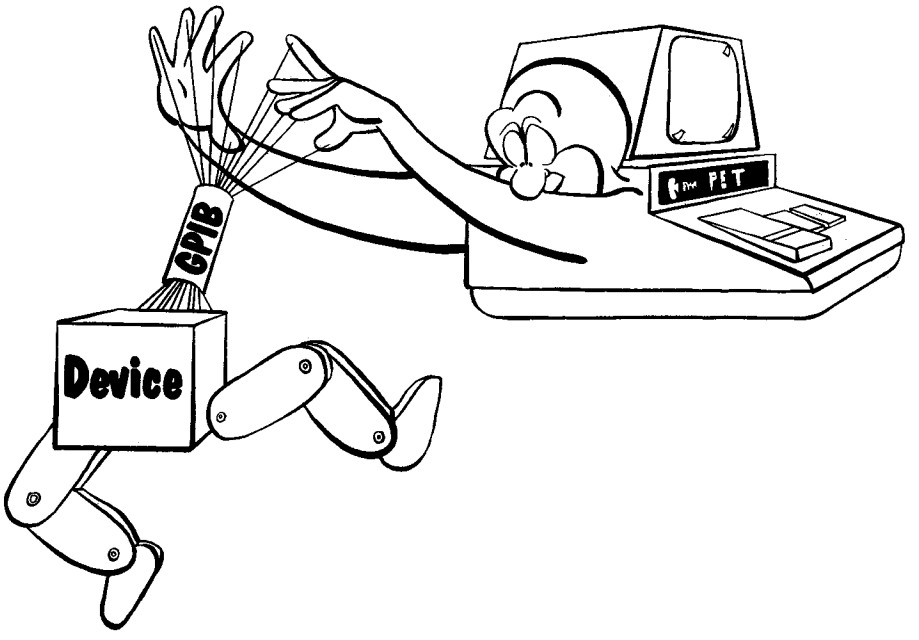


Courtesy of Commodore Business Machines, Inc.

Figure 3-6. Rear view of the 2001 Series computer showing switch, fuse, AC power cord, and interface connector locations

J1 is a standard 12-position, 24-contact edge connector, with "fingers" having 0.40 cm (0.156 in) between contact centers. (Keyway slots are located between fingers 2-3 and 9-10.) These edge connectors have two contacts at each finger location, which are identified with their number/letter designations in Figure 3-7. These simplified views show a) the connector fingers on the top side of the edge connector, an extension of the logic-card assembly; and b) an end-on view of the connector fingers as seen through the J1 slot from the rear of the PET.

The connector fingers shown in Figure 3-7 are connected to the IEEE 488 (GPIB) Bus wires in accordance with the designations given in Table 3-4.



GPIB Bus Connectors

The second connector is the GPIB interfacing connector, which interfaces a remote device. Three manufacturers are listed for reference in Table 3-5. These connectors cannot be connected directly to the PET, but must be interfaced according to the pin designations in Table 3-4 with connectors in Table 3-3.

Table 3-4. Connections to be made between the GPIB and the PET connector J1, when PET is used as a controller of peripheral devices

IEEE Bus (GPIB) Contact	PET J1 Contact	GPIB Functional Division	IEEE Function Assignment	IEEE Function Description
1	1	Data Bus	DIO1	Data Input/Output Wire 1
2	2		DIO2	Data Input/Output Wire 2
3	3		DIO3	Data Input/Output Wire 3
4	4		DIO4	Data Input/Output Wire 4
5	5	Interface Management Bus	EOI	End Or Identify
6	6	Transfer Control Bus	DAV	Data Valid
7	7		NRFD	Not Ready For Data
8	8		NDAC	Not Data Accepted (Data not Accepted)
9	9	Interface Management Bus	IFC	Interface Clear
10	10		SRQ	Service Request
11	11		ATN	Attention
12	12		SHIELD	GPIB Cable Shield and Chassis Ground
13	A	Data Bus	DIO5	Data Input/Output Wire 5
14	B		DIO6	Data Input/Output Wire 6
15	C		DIO7	Data Input/Output Wire 7
16	D		DIO8	Data Input/Output Wire 8
17	E	Interface Management Bus	REN	Remote Enable (Always at ground in the PET)
18	F	Grounds	Gnd 6	DAV)
19	H		Gnd 7	NRFD)
20	J		Gnd 8	NDAC) Matching grounds for
21	K		Gnd 9	IFC) these control lines
22	L		Gnd 10	SRQ)
23	M		Gnd 11	ATN)
24	N		Logic Gnd	EOI and REN Gnds

Table 3-5. IEEE Std 488 (GPIB) standard bus connectors. Matching receptacle/plug part numbers are shown for a connector assembly

Manufacturer	Connector Part Number	Connector Description
AMP	552305-1	Insulation displacement receptacle
	552301-1	Insulation displacement plug
Amphenol	57-2024-0	Receptacle
	57-1024-0	Plug
Cinch	5720240	Solder-type receptacle
	5710240	Solder-type plug

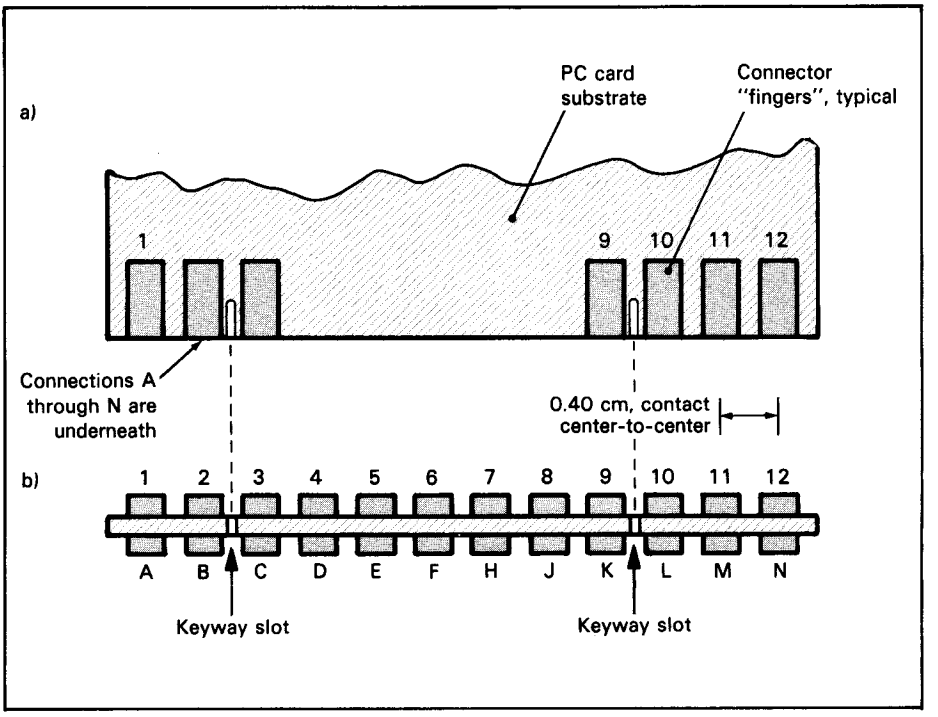


Figure 3-7. The PET J1 edge connector

METHODS OF INTERFACING — DAISY CHAIN AND STAR

By providing a receptacle and plug at the end of each interconnecting cable (see Figure 3-5), system designers can achieve rigid stacking of connectors at cable intersections or at remote device terminations. These receptacles, plugs, and cables, as just described, can be connected in either daisy chain or star configurations to interface the PET when it is a controller of remote devices. These two interfacing methods are illustrated in Figure 3-8.

Each GPIB plug and receptacle combination, as shown in Figure 3-9, has 24 pins arranged in two rows of 12, all of which is encased in a trapezoidal shell.

Pin designations for the GPIB connector are shown in Figure 3-10. Sixteen pins are assigned to three functional groups: eight pins for data, five pins for management of remote devices and three pins for data transfer control. In addition, there are seven ground pins, including one logic ground pin for logic signal commons. The shield pin is for terminating the outer shield, thus sending electrical noise signals to chassis ground.

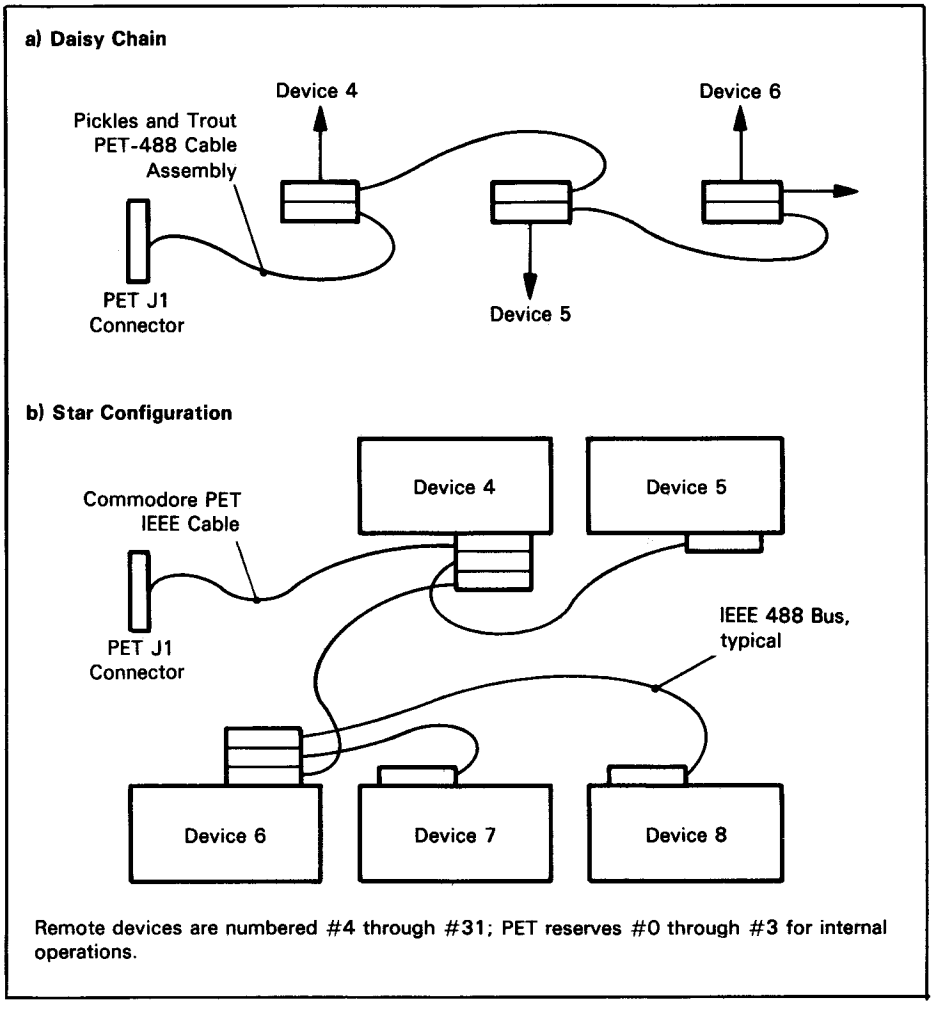


Figure 3-8. Two methods of interfacing devices on the GPIB: a) the daisy chain and b) the star configuration

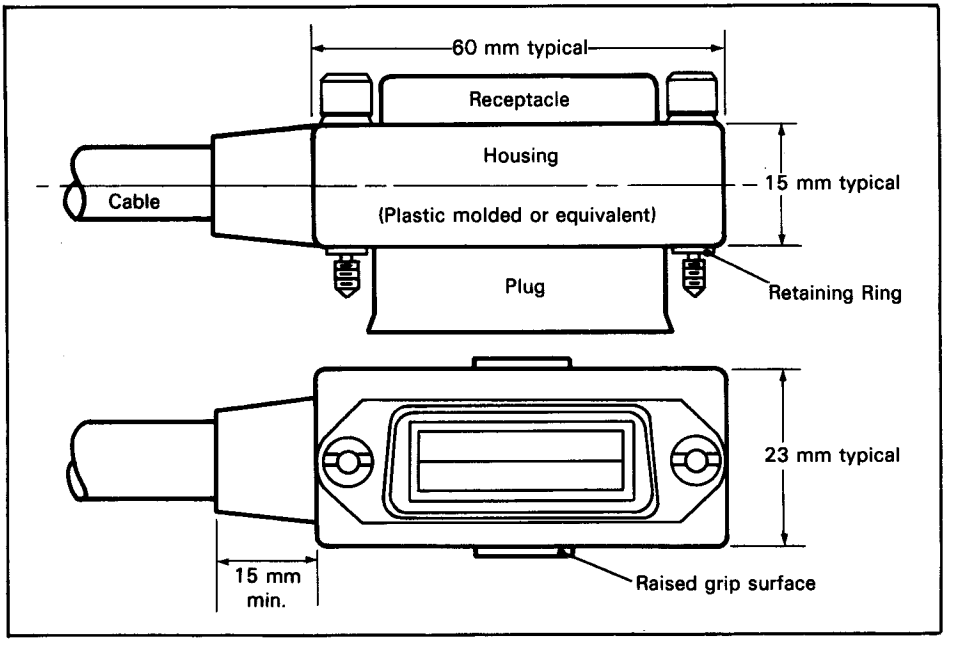


Figure 3-9. The GPIB cable connectors showing the receptacle/plug combination

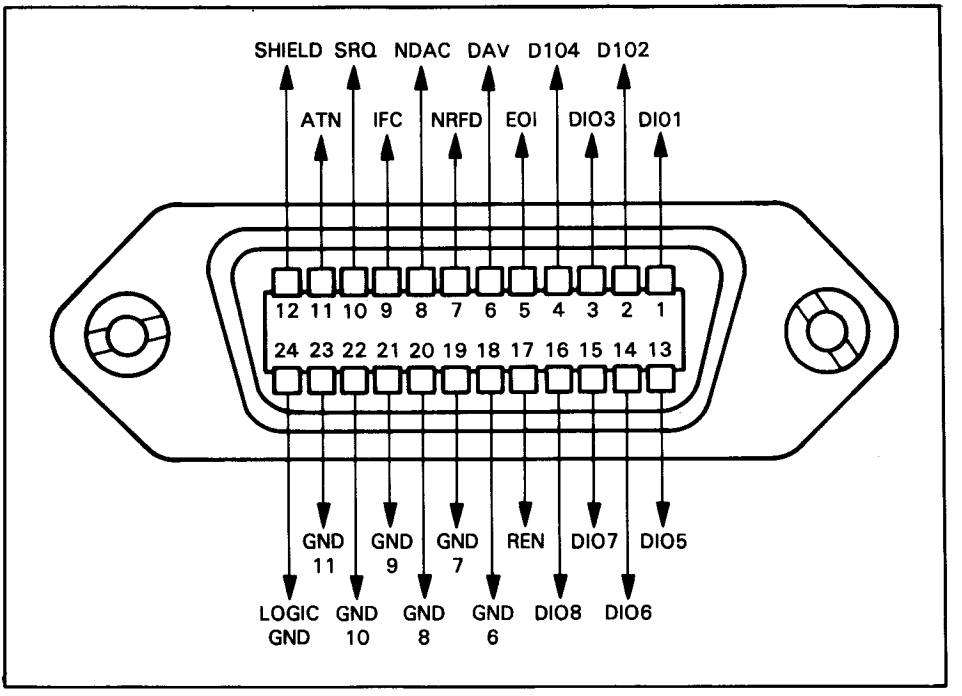


Figure 3-10. The GPIB connector showing pin designations

INTERCONNECTING THE PET WITH IEC STANDARD INSTRUMENTS

PET users (in the United States and abroad) should be aware of the difference between the International Electrotechnical Commission (IEC) Publication 625-1 and its IEEE Std 488-1978 counterpart (Revision of ANSI/IEEE Std 488-1975).

The IEEE Standard 488-1978 is described as being "in concert with its international counterpart, International Electrotechnical Commission (IEC) Publication 625-1. Major portions of the . . . (IEC) publication . . . are identical word for word."* However, there is a difference in the mechanical hardware that implements the two standards, as shown in Table 3-6. Other than the first four DIO lines (lines 1-4), all remaining connector pins carry different assignments. Also, the IEC Standard has 25 pins, as opposed to the IEEE Std 488-1978 which has 24. Users of the PET should not try to interface equipment on a bus with devices having the IEC Standard pin assignments without first checking and installing the necessary adapter(s).

Table 3-6. The bus connector pin designations — a comparison of IEEE Std 488-1978 and the IEC Standard 625-1

IEEE Standard		IEC Standard	
Pin	**Designation**	**Pin**	**Designation**
1	DIO1	1	DIO1
2	DIO2	2	DIO2
3	DIO3	3	DIO3
4	DIO4	4	DIO4
5	EOI	5	REN
6	DAV	6	EOI
7	NRFD	7	DAV
8	NDAC	8	NRFD
9	IFC	9	NDAC
10	SRQ	10	IFC
11	ATN	11	SRQ
12	SHIELD	12	ATN
13	DIO5	13	SHIELD
14	DIO6	14	DIO5
15	DIO7	15	DIO6
16	DIO8	16	DIO7
17	REN	17	DIO8
18	GND 6	18	GND 5
19	GND 7	19	GND 6
20	GND 8 (Ground pins for lines 6-11 inclusive)	20	GND 7
21	GND 9	21	GND 8 (Ground pins for lines 5-12 inclusive)
22	GND 10	22	GND 9
23	GND 11	23	GND 10
24	LOGIC GND	24	GND 11
		25	GND 12

*IEEE Standard Digital Interface for Programmable Instrumentation. The Institute of Electrical and Electronics Engineers, Inc., New York, 1978, frontispiece.

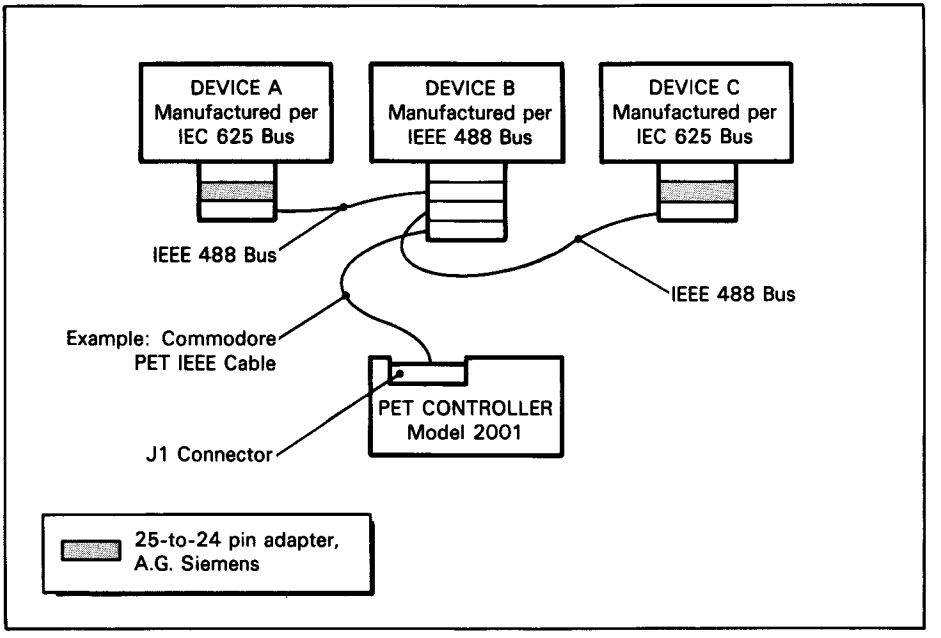


Figure 3-11. One method of cabling IEC Pub. 625-1 instruments to the PET and other programmable devices via the IEEE 488 Bus (GPIB). The necessary link is the cable adapter at each device manufactured per the IEC 625 Bus.

Figure 3-11 illustrates how the PET can be included in systems having devices manufactured in accordance with the IEC Standard. Normally, such programmable equipment will have chassis-mounted male 25-pin connectors of a type similar to DB-25P. These connectors mate with female connectors of the type DB-25S, which are part of the IEC Standard Interface bus cable assembly. However, if these devices (for example, A and C) are to be interconnected with the PET and other IEEE Std 488 equipment (Device B), then the IEEE 488 Bus could easily be used in the configuration shown. One important feature in this system is the connector interface adapter for devices A and C. These adapters convert the 25 connections of the IEC Standard to the 24 connections of the IEEE 488 Standard (Table 3-6). Manufacturers of these adapters include N.V. Philips, A.G. Siemens, and C.R.C. Schlumberger.

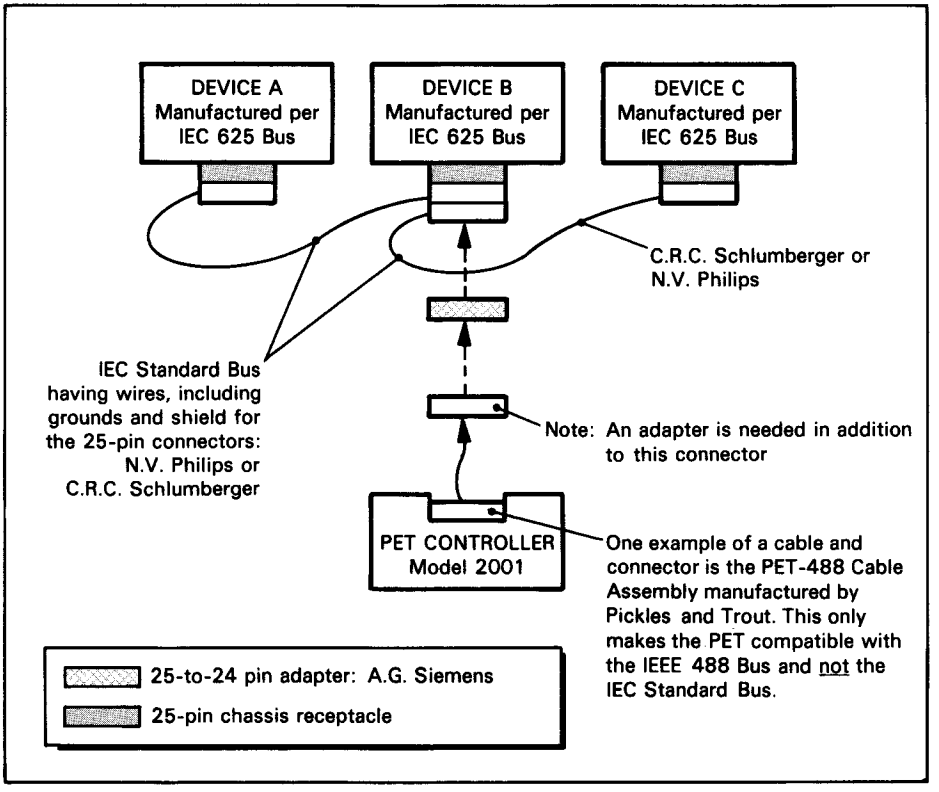


Figure 3-12. Connecting the PET in an IEC-compatible system

Figure 3-12 shows how to connect the PET with a system having only IEC-compatible equipment; here the devices would probably be interconnected using the IEC Standard bus. Programmable devices designed to be compatible with the IEC Standard bus have connectors with 25 pins. If the PET is connected to such systems, a 24-to-25 pin adapter is needed. You can install a cable adapter at the location shown in Figure 3-12, or you can convert directly from the 25-connection IEC Standard to acceptable PET connections at the J1 connector, using the connector-pin assignments in Tables 3-4 and 3-6.

SETTING UP AND TESTING BUS CONNECTIONS

When the time comes to interconnect the PET via the GPIB with an instrument, you should have good procedures for testing in order to take the guesswork out of what you do.

The first step is to plug into the PET's J1 port a compatible connector with a cable having a standard IEEE 488 connector on the opposite end. One such cable with connectors is the one manufactured by Pickles and Trout, shown in Figure 3-13. The connector specifications were described eariler. This cable connector system provides a cost-effective, shielded interconnect between the PET and a GPIB device.

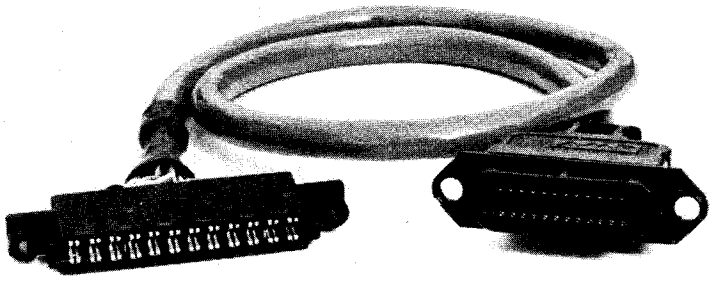


Photo courtesy of Pickles and Trout

Figure 3-13. The PET 488 cable assembly manufactured by Pickles and Trout

Having connected an instrument to the PET, you must set address switches in the instrument. These switch settings determine the instrument's primary (device) address. The proper address should be selected from among those assigned to GPIB talkers and listeners. (Many instruments marketed today have very confusing address descriptions; frequently the number system used to describe the address is not spelled out.)

After selecting the instrument address, try running the following sample program:

```
90 FOR I=4 TO 30
100 OPEN 5,1,2,"TEST"
110 IF ST=-128 GOTO 130
120 PRINT I
130 CLOSE 5
140 NEXT I
```

This program will go through all the available device numbers, looking for a response from your instrument. Any response causes the PET to print the responding device's address, in decimal, on the PET CRT. Occasionally two adjacent addresses will respond, one erroneously; this occurs when the instrument responds for a time long enough for the PET to think it received two responses. The first number will always be the correct address.

If the instrument is programmable, use PRINT statements to output appropriate control and data character strings to the instrument. If the characters required by the instrument are not part of the standard PET character set, it may be necessary to write characters using PEEKs and POKEs.

You will normally use the INPUT statement to receive data or characters from the instrument. Several numbers or strings may be read in with one INPUT statement. Occasionally there may be character incompatibility problems. In such cases, use the GET statement instead of INPUT. GET receives one character at a time from the GPIB that may be examined and displayed on the CRT.

In the next chapter we begin examining INPUT, GET and the other BASIC I/O statements in detail.

CHAPTER 4

Sample Bus Transactions

PROGRAMMING THE PET FOR THE IEEE 488 BUS

The following six BASIC statements and two BASIC commands are routinely used by the PET to communicate with IEEE 488 devices:

BASIC Statements	BASIC Commands
OPEN	SAVE
INPUT#	LOAD
GET#	
PRINT#	
CMD	
CLOSE	

These statements use the following four parameters:

Logical file number	(f)
Device number	(d)
Secondary command	(s)
Variable	(v)

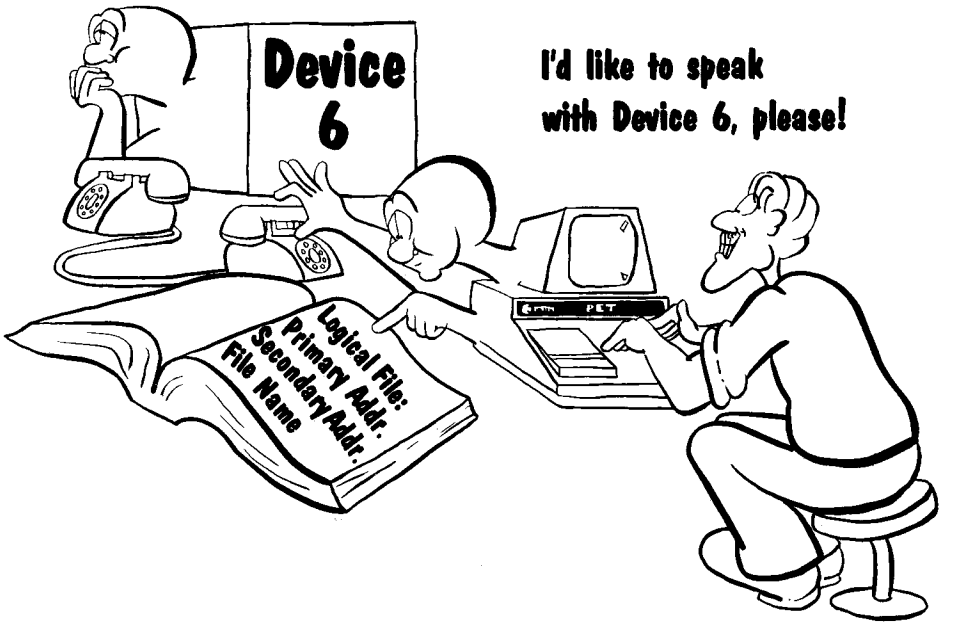
These statements, commands and their parameters as they apply to PET-GPIB communications are described below.*

*For general descriptions and detailed format presentations, refer to the PET Personal Computer Guide. Osborne/McGraw-Hill, 1980.

PARAMETERS

Logical File Number

Each device on the IEEE 488 Bus must be assigned a unique logical file number. This is done using an OPEN statement.



Device Number

Each device connected to the IEEE 488 Bus must be preassigned a device number. GPIB device numbers can range from 4 to 30 inclusive, the numbers 0 through 3 being reserved for the PET's captive peripherals. The captive peripherals and their assigned numbers are:

Keyboard	0
Cassette 1, front panel mount	1 (Default)
Cassette 2, interface J3	2
Video display	3

The device number, usually assigned by selecting a combination of switches on an IEEE Bus device, represents the instrument's address.

Secondary Command

The Secondary Command parameter, if present, specifies a preassigned secondary address. Secondary addresses, as used on the IEEE 488 Bus, require special discussion.

Device addresses can be either primary or secondary. There can be 31 primary talk addresses and 31 primary listen addresses; there is also one disable address for each (universal unlisten and universal untalk), giving a total of 64 addresses. These addresses give instrument designers flexibility; for example, a designer may use different addresses to switch from one instrument function to another (e.g., ohms to volts on a volt-ohmmeter) or to switch ranges (e.g., 15-volt range to 150-volt range).

During the formulation of the IEEE and IEC standards, many members of the committee believed that more than the 31 primary addresses were needed, and so a secondary address (the secondary command) group was agreed on. Each primary address may have up to 32 secondary addresses. Secondary addresses may range from 0 to 31.

Instructions accompanying any instrument you connect to the IEEE 488 Bus will specify the secondary addresses used.

The PET printer uses these six secondary addresses:*

Secondary Address	Operation
Default- 0	Normal printing
1	Printing under format statement control
2	Transfer data from PET to format statement
3	Set variable lines per page
4	Use expanded diagnostic messages
5	Byte data for programmable character

Variable

These statements or commands may have one or more variable parameters that specify a data item or file name.

*PET User Manual, Model 2001-8, first edition, page 66. Commodore Business Machines, Inc., 1978.

OPEN f, d, s, v

The OPEN statement is always the first statement in a program sequence that enables communication between the PET and any device on the bus:

OPEN f, d, s, "name"

The OPEN statement opens a logical file, identifies a device via its device number, and issues a secondary address, if one is used. The OPEN statement may further identify a portion of the device by giving a file name. The OPEN statement will transmit nothing to the bus — only communicate with the PET internal operating system — unless the "name" is specified.

By opening a logical file, you specify the primary and secondary addresses and the optional file name, which specifies a particular device thereafter. For example, PRINT# 5 will output to logical unit 5, which represents whatever device has been assigned this logical unit number by the OPEN statement. The PET, Model 2001, allows ten logical files to be opened at any one time. By opening more than 10 files, you might cause a permanent error in the PET BASIC operating system, making it necessary to power-down to restore operation.

INPUT# f, v

The INPUT statement inputs information to the PET from any device that is capable of transmitting data. The INPUT statement has the format:

INPUT# f, v

The logical file number (f) is the number assigned to the selected device by an OPEN statement.

Input data is subsequently identified by the variable name v.

GET# f, v

The GET statement is similar in form to INPUT:

GET# f, v

The logical file number and the variable name are supplied in the same sequence.

Whereas the INPUT statement accepts input characters while waiting for the delimiter characters CR (carriage return) to specify the end of the input, the GET statement receives a single character.

PRINT# f, v

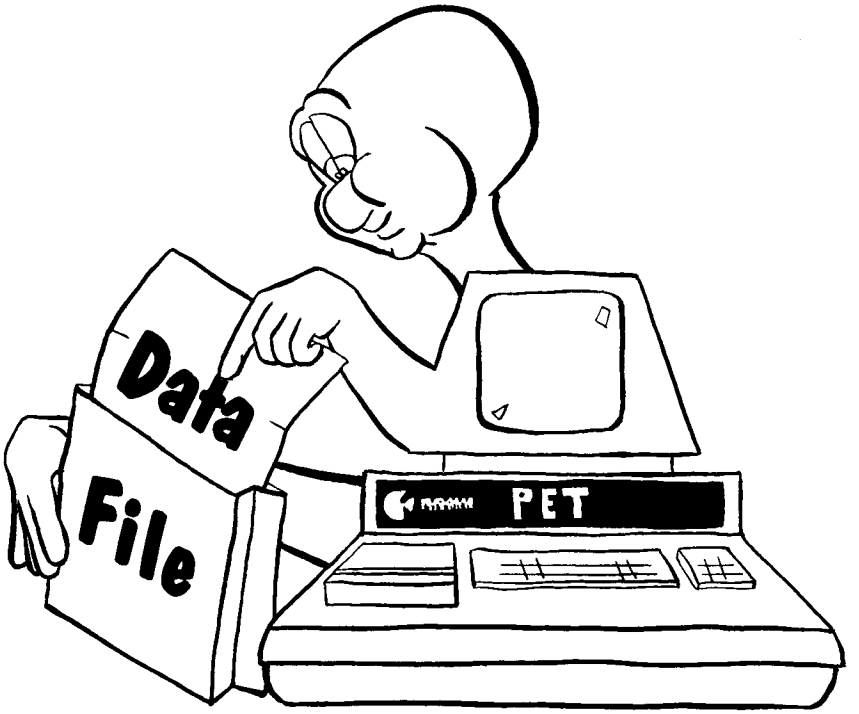
The PRINT statement is used to transfer data from the PET to a remote device on the IEEE 488 Bus that is capable of receiving data:

PRINT# f, v

The output from the PET will go to the device assigned to logical file f as specified by the OPEN statement. The "v" specifies the variable or variables (either numeric or string) that will be transmitted by this statement.

CMD f, v

The CMD (Command) statement is just like PRINT, except that the listening device on the IEEE 488 Bus is not told to unlisten after each transmission. Therefore, the device remains on the bus monitoring all IEEE transactions in progress. In addition, the CMD statement internally instructs the PET to redirect all display outputs to the IEEE 488 Bus.



CLOSE f

The CLOSE statement is used to close files that have been previously opened by the OPEN statement. You need only specify the logical file number. Each file must be closed separately.

SAVE v, d, s

The SAVE command is used to copy the current program in PET memory to an external device, thereby storing or "saving" a copy of it. The output operation is handled automatically by the PET.

SAVE "name", d, s

The name between double quotes is the file name by which the program is subsequently identified. The copy of the program is sent to the specified device d. The secondary command, issued automatically by the PET, makes the above secondary commands optional. In PET units having the old ROMs, the PET issues a primary address; the secondary address is transferred to the GPIB automatically, whether specified in the SAVE statement or not. This is followed by the file name and the unlisten command. During addressing, the ATN line assertions conform to the IEEE Std 488-1978 transactions. However, the ATN line is again asserted low (true) where it remains throughout the output message. Therefore, a portion of this command does not follow the IEEE Std 488-1978 transactions; the devices conforming to the standard and residing on the GPIB could malfunction due to their thinking that they have been addressed when they have not. The SAVE operation and timing for such PETs are described in detail in Chapter 5.

In PET units having the new ROMs, the ATN line is asserted in accordance with the IEE Std 488-1978 transactions, completely implementing the SAVE statement for the GPIB. Its operation is described in detail in Chapter 5.

To determine if your PET controller has old ROMs or new ROMs, observe the CRT when the PET is first turned on.

Units with old ROMs will display:

*** COMMODORE BASIC ***

Units with new ROMs will display:

COMMODORE BASIC

LOAD v, d, s

The LOAD statement loads into PET memory a program that has previously been saved by the SAVE command.

LOAD "name", d, s

If no name is specified, the next program is loaded from device d and the optional secondary command s.

In PET models having the old ROMs, the LOAD command is not completely implemented in the IEEE 488 Bus functions; its operation and timing are described in detail in Chapter 5. In PET models having the new ROMs, the LOAD command is completely implemented in the IEEE 488 Bus functions; its operation is also described in detail in Chapter 5.

TRANSACTION CODES (ASCII)

The PET uses ASCII* codes to transmit and receive characters. Table 4-1 illustrates ASCII codes, showing binary and hexadecimal numbers.

The body of Table 4-1 shows the letters, numbers, special characters, and control codes whose ASCII representations are used for GPIB instrument control. At the left the least significant bits (LSB) of the 7-bit ASCII code are shown for bits 1 through 4. To the right the most significant bits (MSB) are shown for bits 5 through 7.

To determine the ASCII code for the letter "A", for example, locate the letter "A" approximately midway through the table. At the far left in the same row is the binary bit pattern 0001. This is the LSB pattern b4 through b1, respectively. Directly above the column in which "A" appears is the MSB pattern 100. These are bits b7 through b5, respectively. Therefore, the total binary bit pattern for "A" in ASCII code is 1000001. This can be read in the "Hex" column as the hexadecimal value 41_{16}.

*Also USASCII: United States of America Standard Code for Information Interchange.

Table 4-1. ASCII to hexadecimal and binary conversions

Least Significant Bits 4 through 1				Most Significant Bits 7 through 5							
b7→				0	0	0	0	1	1	1	1
b6→				0	0	1	1	0	0	1	1
b5→				0	1	0	1	0	1	0	1
↓ Binary ↑				→ ↓Hex	$0	$1	$2	$3	$4	$5	$6	$7
b4	b3	b2	b1									
0	0	0	0	$0	NUL	DLE	SP	0	@	P	'	p
0	0	0	1	$1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	$2	STX	DC2	"	2	B	R	b	r
0	0	1	1	$3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	$4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	$5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	$6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	$7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	$8	BS	CAN	(	8	H	X	h	x
1	0	0	1	$9	HT	EM	)	9	I	Y	i	y
1	0	1	0	$A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	$B	VT	ESC	+	;	K	[	k	[
1	1	0	0	$C	FF	FS	,	<	L	\	l	\
1	1	0	1	$D	CR	GS	-	=	M	]	m	]
1	1	1	0	$E	SO	RS	.	>	N	^	n	^
1	1	1	1	$F	SI	US	/	?	O	—	o	DEL

Adapted from material furnished courtesy of Tektronix, Inc.

The letter "A" transferred between a talker and a listener on the GPIB would create high and low logic levels on the data bus as follows:

The eight data lines: 8 7 6 5 4 3 2 1

The binary code: MSB * 1 0 0 0 0 0 1 LSB

As another example, the letter "G" being transferred between two GPIB devices will cause the eight DIO lines on the bus to be asserted as follows:

	G							
	4				7				Hexadecimal
MSB	1	0	0	0	1	1	1	LSB	The binary bit pattern divided into MSB and LSB
	4	2	1	8	4	2	1		The column values
D8	D7	D6	D5	D4	D3	D2	D1		The eight DIO lines
	1	0	0	0	1	1	1		The logic levels appearing on the GPIB data bus

*With the exception of specific cases in the OPEN, SAVE and CLOSE statements, the PET uses a 7-bit code. Often, this eighth bit does not appear.

ADDRESS AND COMMAND CODES

An extended version of the same ASCII code table is used to transmit device codes, secondary addresses, and other control information on the GPIB, as shown in Table 4-2. This table shows address and command ASCII codes used by the PET.

Binary and hexadecimal values appear across the top and in the left columns, as in Table 4-1; however, there are two additional columns having hexadecimal values of $E and $F. These two columns add an eighth bit to every character code. $8, $9 and $D are omitted because they are unused.

The body of Table 4-2 differs from Table 4-1 in that it shows the numbers (in decimal values) that can be assigned to file variables and devices residing on the GPIB. These devices fall into the two broad categories of a Listen Address Group and a Talk Address Group. The Secondary Command Group applies to both listeners and talkers. These categories are identified across the bottom of the table.

Let us look at some examples.

If the PET, operating as a controller, sends a primary address to device number 5, a listener residing on the GPIB, the primary address would be 25_{16}. On the other hand, if device number 5 is a talker, then the primary address would be 45_{16}.

If the PET, operating as a controller, sends a primary address and a secondary address to device number 5, a talker, then it would first transmit the primary address 45_{16}. Now refer to the secondary command group, columns $6 and $7. If the secondary address has been specified as 6, device number 5 would be secondary-addressed by 66_{16}.

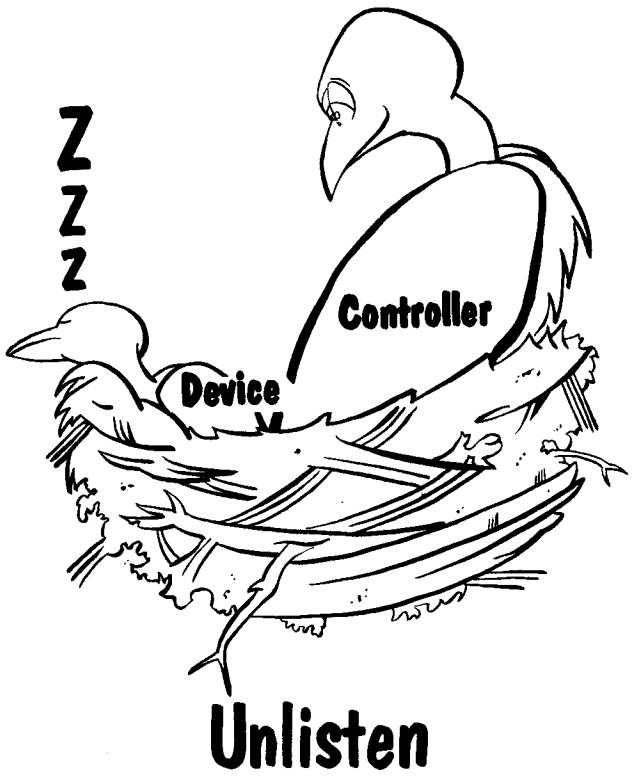
The column headed $E is a secondary command group which the PET uses for the CLOSE statement. For example, if the secondary command issued in the OPEN statement is 2, then the value $E2_{16}$ is sent to the GPIB data lines in response to the CLOSE statement.

The column headed $F is a secondary command group column which the PET uses for the OPEN and SAVE commands. For example, if the secondary command for a SAVE is 17, the value $F1_{16}$ is sent to the GPIB data lines.

Table 4-2. Address and command groups

Least Significant Bits 4 through 1				Most Significant Bits 7 through 5								8	
b8→										b8→		1	1
b7→				0	0	0	0	1	1	1	1	1	1
b6→				0	0	1	1	0	0	1	1	1	1
b5→				0	1	0	1	0	1	0	1	0	1
↓Binary↑				→ ↓Hex	$0	$1	$2	$3	$4	$5	$6	$7	$E	$F
b4	b3	b2	b1											
0	0	0	0	$0	—	—	00	16	00	16	00	16	00	16
0	0	0	1	$1	GTL	LLO	01	17	01	17	01	17	01	17
0	0	1	0	$2	—	—	02	18	02	18	02	18	02	18
0	0	1	1	$3	—	—	03	19	03	19	03	19	03	19
0	1	0	0	$4	SDC	DCL	04	20	04	20	04	20	04	20
0	1	0	1	$5	PPC	PPU	05	21	05	21	05	21	05	21
0	1	1	0	$6	—	—	06	22	06	22	06	22	06	22
0	1	1	1	$7	—	—	07	23	07	23	07	23	07	23
1	0	0	0	$8	GET	SPE	08	24	08	24	08	24	08	24
1	0	0	1	$9	TCT	SPD	09	25	09	25	09	25	09	25
1	0	1	0	$A	—	—	10	26	10	26	10	26	10	26
1	0	1	1	$B	—	—	11	27	11	27	11	27	11	27
1	1	0	0	$C	—	—	12	28	12	28	12	28	12	28
1	1	0	1	$D	—	—	13	29	13	29	13	29	13	29
1	1	1	0	$E	—	—	14	30	14	30	14	30	14	30
1	1	1	1	$F	—	—	15	UNL	15	UNT	15	31	15	31
					Addressed Command Group ACG	Universal Command Group UCG	Listen Address Group LAG		Talk Address Group TAG		Secondary Command Group SCG		Secondary Command Group PET SCG CLOSE Only	Secondary Command Group PET SCG OPEN "file" and SAVE Only

Adapted from material furnished courtesy of Tektronix, Inc.



Referring back to Table 4-1, notice the position of the question mark. The ? has the hexadecimal equivalent of 3F. Table 4-2 has UNL in the same position. This means that the ASCII equivalent of the character ?, or $3F_{16}$, appearing on the DIO lines during an attention sequence tells all listeners on the GPIB to unlisten, or disconnect themselves from the bus. (The same code transmitted as data represents the ? character.)

Likewise, the universal untalk command, UNT, has a value of $5F_{16}$; the same location in Table 4-1 shows a dash (–). This number ($5F_{16}$) transferred as a primary address will cause the untalk command to be issued.

Characters in the Addressed Command Group and the Universal Command Group are specified in the IEEE Std 488-1978; however, the PET does not implement these functions. (See Appendix C for definitions.)

EXAMPLE PROGRAMS

We will now look at some examples of addresses, commands, and data transmitted to and from the GPIB using various BASIC I/O statements. Refer to Tables 4-1 and 4-2 for address, command, and character codes.

GPIB PRINT TRANSACTIONS — IMMEDIATE MODE

The sample bus transactions in Table 4-3 occur in response to a simple PRINT statement:

```
OPEN 5,5
PRINT# 5, "GENE"
```

Table 4-3. PRINT transaction in immediate mode

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
OPEN 5,5 PRINT# 5, "GENE"	1	ATN	LAG 05	25	Primary address
	2		G	47	Data byte 1 } Program
	3		E	45	Data byte 2 } characters
	4		N	4E	Data byte 3 }
	5		E	45	Data byte 4 }
	6		CR	0D	Carriage return
	7	EOI	LF	0A	Line feed
	8	ATN	UNL	3F	Unlisten command

The PRINT statement, followed by the pound sign (#), transmits data to the bus.

The PRINT statement must be preceded by an OPEN statement, which specifies the files to be opened and the device to be addressed. In the example, the file is number 5 (first 5 in the OPEN statement) and the instrument on the bus being addressed is device number 5 (second 5 in the OPEN statement). The lack of a third parameter in the statement indicates there is no secondary address.

The next statement is PRINT# 5, "GENE". It is shown in direct or immediate mode, having no program line number. Hence, the statement is executed immediately; it is not stored in memory. Because there is no secondary address, the characters GENE, which constitute the actual data, are placed on the bus immediately following the primary address. (For timing details, see "Timing Signals for the PRINT Statement" in Chapter 5.)

Quotation marks cause GENE to be sent to the GPIB as characters, represented by the hexadecimal numbers 47, 45, 4E and 45 respectively. The closing quotes specify the second E to be the last character in the group. A carriage return (CR) initiates GPIB transfer, causing a $0D_{16}$ bit pattern to be placed on the bus. This is followed automatically by a line feed (LF) which places the binary bit pattern for $0A_{16}$ on the bus.

The End Or Identify (EOI) line is asserted by the PET, which says, in effect, that E is the last character.

Finally, $3F_{16}$ is placed on the DIO lines to send the universal unlisten command UNL to all listener devices, telling them to unlisten or "remove" themselves from the bus.

This entire operation has occurred in direct (immediate) mode; nothing has been stored in memory. The PET immediately does whatever it is directed to do.

GPIB PRINT TRANSACTIONS — RUN MODE

We will now examine the PRINT statement's execution in run mode as shown in Table 4-4.

The statements are:

```
10 OPEN 5,5,2
20 PRINT# 5, "GENE"
30 GO TO 20
```

Table 4-4. PRINT transaction in run mode

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
10 OPEN 5,5,2 20 PRINT# 5, "GENE" 30 GO TO 20	1	ATN	LAG 05	25	Primary address
	2	ATN	SCG 02	62	Secondary address
	3		G	47	Data byte 1 } Program characters
	4		E	45	Data byte 2 } Program characters
	5		N	4E	Data byte 3 } Program characters
	6		E	45	Data byte 4 } Program characters
	7		CR	0D	Carriage return
	8	EOI	LF	0A	Line feed
	9	ATN	UNL	3F	Unlisten command
10 OPEN 5,5,2 20 PRINT# 5, "GENE" 30 GO TO 20	10	ATN	LAG 05	25	Sequence repeated
	11		SCG 02	62	
	12		G	47	
	13		E	45	
	14		N	4E	
	15		E	45	
	16		CR	0D	
	17	EOI	LF	0A	
	18	ATN	UNL	3F	
	19	ATN	LAG 05	25	Continues until stopped

In this sample bus transaction, each program line begins with a statement number; this causes the program to be placed into the PET's memory and not executed until the RUN command is given.

Here a secondary address is issued; a Secondary Group code of 2 causes device number 5 to be addressed a second time, with the value 62_{16} being sent on the bus.

The third statement (line 30) tells the PET to return to line 20. This causes the PRINT statement to be executed over and over again. After the universal unlisten (UNL) command is sent in the first PRINT statement, the program runs in

a tight loop, sending the same sequence of characters to the bus until the STOP key is depressed.

GPIB OPEN/CLOSE TRANSACTIONS — IMMEDIATE MODE

The sample bus transactions in Table 4-5 show how a file in the PET can be identified with a number and given a file name, in this example, "TEST." The file is then closed. The statements are:

```
OPEN 5,5,2,"TEST"
CLOSE 5
```

Table 4-5. OPEN/CLOSE transactions in immediate mode

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
OPEN 5,5,2, "TEST"	1	ATN	LAG 05	25	Primary address
	2	ATN	SCG 18	F2	Secondary address
	3		T	54	File name
	4		E	45	
	5		S	53	
	6	EOI	T	54	
	7	ATN	UNL	3F	Unlisten command
CLOSE 5	8	ATN	LAG 05	25	Close file primary address
	9	ATN	SCG 02	E2	Close file secondary address

The OPEN statement is issued in immediate mode, as indicated by the statement's not having a line number. Thus, the word TEST will be immediately transferred to the bus.

The first 5 in the OPEN statement identifies the logical file number assigned for this particular operation. The second 5 identifies a device number 5 residing on the GPIB.

When the OPEN statement is issued with a file name, the secondary address, the 2 in the OPEN statement (code 62_{16}) becomes 18 (code $F2_{16}$). This is reflected in the table at entry 2; the Characters column shows the secondary command group SCG 18. This change in numbers comes from Table 4-2. (See SCG column and the rightmost column, where the PET, in this instance, changes the original 0 in bit 5 to a 1 and adds an eighth bit (b8) to the normal 7-bit hexadecimal code.) As this eighth bit is a "1" (true) value, the four most significant bits become 1111_2 (F_{16}) rather than 111_2 (7_{16}). This causes the binary bit pattern that is the equivalent of $F2_{16}$ to appear on the GPIB data bus.

The CLOSE statement (table entries 8 and 9) terminates this transaction on the bus. Here the fifth device is primary-addressed, and a secondary command group 02 is automatically issued. The "02" in SCG 02 has a direct correlation to the secondary command "2" in the OPEN statement. If the OPEN statement had been OPEN 5,5,7 then the secondary command group on table entry 9 would be SCG 07.

Note that table entry 9 is $E2_{16}$. This is another special case, similar to the one described above for the OPEN statement. Here again, $E2_{16}$ is determined from Table 4-2. The second column from the right has decimal values assigned by the PET for CLOSE statements only. This column shows that an eighth bit (b8) is appended to the normal 7-bit code. As this eighth bit is a "1" (true) value, the four most significant bits become 1110_2 (E_{16}) rather than 110_2 (6_{16}). Therefore, the binary bit pattern placed on the GPIB corresponds to $E2_{16}$, which terminates the transaction. There are 16 unique OPEN "file" and 16 unique CLOSE operators, although SCG can have 32.

GPIB "COMBINATION" TRANSACTIONS

Two bus transactions similar to the ones just described are now combined in one program and the combination is saved using the SAVE statement. The entire program has these three statements:

```
 5 OPEN 5,5,2, "GENE"
10 PRINT# 5, "TEST"
30 SAVE "GLOP", 5
```

The bus transactions (table entries 1 through 7) begin with an OPEN statement. This statement assigns logical file 5 and assigns the file name "GENE." The instrument to be addressed is a listener, device number 5. Also, a secondary address is issued. These bus transactions (through table entry 7) are similar to those previously discussed for the OPEN statement.

Table entries 8 through 16 show the bus transactions for the PRINT statement, this time with the word "TEST." The bus transactions are similar to those discussed before under "GPIB PRINT Transactions — Run Mode."

Next, for PET controllers having old ROMs, the SAVE statement is executed to save the program; this causes the entire program to be transferred to device number 5, creating a file named "GLOP." (Such a file could be referred to by similar non-descriptive names or a name such as "FILE-A," "RECORDS," or "DATA FILE.")

Table entry 2, the secondary command SCG 18, is a direct result of the 2 appearing in the secondary address position in the OPEN statement. Here, once again, the PET adds an eighth bit onto the 7-bit ASCII code, which changes the 72_{16} in the secondary command group to $F2_{16}$. However, most GPIB devices interpret the secondary command as an SCG 18 (see Table 4-2), ignoring the leading bit of this special secondary address.

Referring to Table 4-6, entry 25, a delimiter, 00_{16}, is necessary to separate the statement just transferred from the remaining part of the program. Note that this delimiter also appears at entries 43 and 56, signifying a necessary separation between OPEN/PRINT and PRINT/SAVE, respectively.

The address of the next statement in memory appears at entries 26 and 27, the least significant byte being transferred first. The address is 0413_{16}.

Table 4-6. Combination transactions

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
5 OPEN 5,5,2, "GENE"	1	ATN	LAG 05	25	
	2	ATN	SCG 18	F2	
	3		G	47	
	4		E	45	
	5		N	4E	
	6		E	45	
	7	ATN	UNL	3F	
10 PRINT# 5, "TEST"	8	ATN	LAG 05	25	
	9	ATN	SCG 02	62	
	10		T	54	
	11		E	45	
	12		S	53	
	13		T	54	
	14		CR	0D	
	15	EOI	LF	0A	
	16	ATN	UNL	3F	
30 SAVE "GLOP", 5	17	ATN	LAG 05	25	
	18		SCG 17	F1	
	19		G	47	
	20		L	4C	
	21		O	4F	
	22		P	50	
	23	ATN	UNL	3F	
	24	ATN*	UNL	3F	
	25			00	Delimiter
	26			13	Next statement address (0413_{16})
	27			04	
	28			05	Line number (0005_{16})
	29			00	
	30			9F	OPEN token
	31		5	35	
	32		,	2C	
	33		5	35	
	34		,	2C	
	35		2	32	
	36		,	2C	
	37		"	22	
	38		G	47	
	39		E	45	
	40		N	4E	
	41		E	45	
	42		"	22	

*The ATN line remains low (true). The bus transactions for the SAVE statement pertain to PET models having the old ROMs only. For some typical SAVE statement bus transactions pertaining to PET models with new ROMs, see Chapter 5.

Table 4-6. Combination transactions (Continued)

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
	43	ATN*		00	Delimiter
	44			21	Next statement address (0421_{16})
	45			04	
	46			0A	Line number ($000A_{16}$)
	47			00	
	48			98	PRINT token
	49		5	35	
	50		,	2C	
	51		"	22	
	52		T	54	
	53		E	45	
	54		S	53	
	55		T	54	
	56	ATN*	"	00	Delimiter
	57			0F	Next statement address ($040F_{16}$)
	58			04	
	59			1E	Line number ($001E_{16}$)
	60			00	
	61			14	SAVE token
	62		"	22	
	63		G	47	
	64		L	4C	
	65		O	4F	
	66		P	50	
	67		"	22	
	68		.	2C	
	69		5	35	
	70			00	Delimiter
	71			00	Delimiter
	72			00	Delimiter
	73	EOI		24	
	74		UNL	3F	
	75		UNL	3F	
	76			E1	

*The ATN line remains low (true). The bus transactions for the SAVE statement pertain to PET models having the old ROMs only. For some typical SAVE statement bus transactions pertaining to PET models with new ROMs, see Chapter 5.

At entries 28 and 29, the Notes column identifies line number 5 as having the statement OPEN 5,5,2, "GENE."

Beginning at entry 30, the SAVE command transmits to the GPIB a copy of what is stored in the PET's user area of memory. However, before this memory can be examined in detail by the SAVE statement, the primary and secondary addresses are issued, the word "GLOP," identifying device number 5, is sent, and the unlisten commands are issued. These addresses, data, and commands can be seen at entries 17 through 24.

Entry 30 introduces a new term, "token." Because BASIC has many terms such as OPEN, CLOSE, PRINT, GET, etc., which it uses over and over, the PET substitutes a different number (token) for each of them. Issued automatically by the PET, these token numbers conserve space in the PET's memory. For example, the four letters in "OPEN" might have been sent to the bus, each letter having been transferred by its equivalent ASCII code. However, by using a token, BASIC accomplishes the same operation by issuing the equivalent of OPEN — the hexadecimal number 9F. Other tokens are shown on lines 48 and 61, for PRINT and SAVE, respectively. A list of tokens for the I/O statements is given in Table 4-7.*

Table 4-7. PET tokens for BASIC I/O statements

BASIC Statement	Token Code	
	Decimal	Hexadecimal
INPUT#	132	84
LOAD	147	93
SAVE	148	94
PRINT#	152	98
CMD	157	9D
OPEN	159	9F
CLOSE	160	A0
GET#	161	A1

Entries 31 through 42 illustrate how numbers, letters, and special characters are transferred on the GPIB by sending the corresponding hexadecimal equivalents. Similarly, characters in the PRINT transaction are listed at entries 49 through 55, and those for SAVE at entries 62 through 69.

Entries 70 through 72 signify that the 00_{16} delimiters are required at the end of a program. The PET automatically issues three. At entry 73 the PET asserts the EOI signal on the next free byte in the PET memory, which, in this case, happens to yield 24_{16}. Two universal unlisten commands are next issued by the PET (entries 74 and 75). The transmission is terminated by $E1_{16}$ being placed on the bus, which tells the bus (in a non-standard way) that the SAVE command is complete.

*For a complete list of tokens, refer to the PET Personal Computer Guide. Osborne/McGraw-Hill, 1980.

Finally, note that the ATN line is asserted beginning at entry 23; it is held low through the remainder of the bus transactions. This is a feature of the SAVE command in PET controllers having old ROMs, and is not a normal operation of the IEEE 488 standard. The PET controllers with new ROMs implement the ATN line in accordance with the IEEE Std 488-1978.

GPIB MULTIPLE OPEN/PRINT TRANSACTIONS

The transactions shown in Table 4-8 illustrate GPIB communications for OPEN statements specifying different numbered files and different secondary addresses. The statements, issued in immediate mode, are:

```
OPEN 6,6,3                    First statements
PRINT# 6, "TEST"

OPEN 5,5,2                    Second statements
PRINT# 5, "GLOP"
```

Table 4-8. Multiple OPEN/PRINT transactions

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
OPEN 6,6,3 PRINT# 6, "TEST"	1		LAG 06	26	
	2		SCG 03	63	
	3		T	54	
	4		E	45	
	5		S	53	
	6		T	54	
	7		CR	0D	
	8	EOI	LF	0A	
	9	ATN	UNL	3F	
OPEN 5,5,2 PRINT# 5, "GLOP"	10		LAG 05	25	
	11		SCG 02	62	
	12		G	47	
	13		L	4C	
	14		O	4F	
	15		P	50	
	16		CR	0D	
	17	EOI	LF	0A	
	18	ATN	UNL	3F	

The first statement (OPEN 6,6,3) specifies via the first 6 that logical file 6 is to be opened. The second 6 signifies that device 6 on the GPIB is to be addressed. The 3 specifies that a secondary address (SCG 03) will be issued.

Entry 1 shows LAG 06 (Listen Address Group 6) being output via the GPIB. As shown in Table 4-2, the LAG 06 will cause 26_{16} to be placed on the GPIB data lines.

Entry 2 shows the correlation between the 3 in the OPEN 6,6,3 statement and the secondary address SCG 03. This secondary command group number will always correspond to the secondary address number specified in the OPEN statement when the associated file number (in this example, 6) is used, such as in the

PRINT statement. If the OPEN statement is issued with a file name (i.e., "TEST"), the PET causes this secondary command number to be SCG 19, code $F3_{16}$, rather than SCG 03, code 63_{16}, as illustrated in Table 4-2.

At entries 3 through 6 the PRINT command is executed, transferring the characters TEST to device number 6.

Entries 10 through 18 show the GPIB transactions for the second pair of statements:

```
OPEN 5,5,2
PRINT# 5, "GLOP"
```

Note that up to ten files may be open at a time.

At entry 11, SCG 02 represents the 2 specified as the secondary address of the second OPEN statement. Bus signals, characters, and hexadecimal numbers can all be identified as previously described.

CHAPTER 5

Execution and Timing Sequences

In this chapter we will discuss eight statements. These are OPEN, PRINT, CMD, SAVE, INPUT, LOAD, GET, and CLOSE. Each of these statements forms a section in the chapter. First, the statement is used in an example and shown in a tabular form that identifies the resulting bus transactions, like the examples that were given in Chapter 4. Then, for the particular statement being discussed, we show you one or more diagrams so you can observe the timing sequences of the transfer control and interface management lines involved.

Timing sequences are given for address timing with a primary and a secondary address, address timing for a primary address only, data burst timing, and unaddress timing. Some timing sequences are the same for more than one statement. For example, primary and secondary address timing is identical or very similar for all of the statements in this chapter. We reproduce the timing diagrams for each statement and give an abbreviated description when the timing events have already been covered in detail.

TIMING SIGNALS FOR THE OPEN STATEMENT

We will now examine timing signals associated with the OPEN statement. Table 5-1 illustrates the statements that will be used to provide examples. For a discussion of terminology used in this table, refer to Chapter 4.

Table 5-1. Sample bus transactions for the OPEN statement

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
OPEN 5,5					Sends nothing on the GPIB
OPEN 5,5,2					Sends nothing on the GPIB
OPEN 5,5,2 "TEST"	1	ATN	LAG 05	25	Primary address
	2	ATN	SCG 18	F2	Secondary address
	3		T	54	Data
	4		E	45	Data
	5		S	53	Data
	6	EOI	T	54	Data
	7	ATN	UNL	3F	Bus unlisten

Transmits on the GPIB

In Table 5-1, the statement OPEN 5,5 uses logical file number 5, which will be associated with external device number 5. This simple OPEN statement contains no secondary address and causes no information to be placed on the bus.

The statement OPEN 5,5,2 contains the elements of the first example, but adds a secondary address. No transmission is yet made via the GPIB.

The third example in the table, OPEN 5,5,2 "TEST," uses all four possible parameters of the OPEN statement. In contrast to the previous examples, this statement causes a series of bus transactions in which the characters "TEST" are transmitted via the GPIB.

In the step-by-step description that follows, the events accompanying GPIB activity are identified in time sequence. Each event has a circled number that is used in the associated timing diagram as well as in the accompanying text.

Only the third OPEN statement is traced, since the other two examples do not generate any bus activity.

OPEN Address Timing — Primary and Secondary Addresses

Before the OPEN statement can transmit data on the data lines, the listener device must be addressed using a primary address or a primary and secondary address. This event sequence uses the ATN line and all three transfer control lines — DAV, NRFD, and NDAC. They take part in the three-wire handshake. At the start of the transaction, the GPIB is typically in its idle state. The ATN and DAV lines are high; the NRFD and NDAC lines can be either high or low. These are the conditions illustrated in Figure 5-1 at time ① .

When ready, the PET at time ② sets the ATN line low (true). It remains low for 190 μs minimum; longer times can occur, depending on the response times of instruments on the GPIB. By lowering the ATN line the PET, as the system controller, tells all equipment residing on the bus that it is ready to issue addresses or commands.

Although the IEE Std 488-1978 specifies that each bus device respond to ATN within 200 ns, the PET allows these peripheral devices up to 14 μs to respond. Within this time, each device on the GPIB is to answer by setting its NDAC line low (true); the NRFD line can be low (true) or high (false). For illustration, we are showing the NRFD line in its high logic state. See time ③ .

Before time ② the logic states of the NDAC and NRFD lines are undefined; they are not monitored by the PET for the purpose of starting the handshake procedure until time ③ , 14 μs after the ATN line is asserted low (true). At this time, if both lines are high (false), the PET assumes there is a "device not present" error and terminates the OPEN statement. Table 5-2 describes this and three additional errors that can arise during the addressing sequence and subsequent data transfer.

When the conditions at time ③ are met (NDAC low and NRFD high or low), the PET places the primary talk address on the DIO lines at time ④ . This occurs a minimum of 25 μs after the PET drives the ATN line low (true).

After time ③ , and before asserting its DAV line low (true), the PET samples the status of the NRFD line. If it remains low, the PET assumes that the GPIB device has not had time to respond, and the PET waits for the NRFD line to go high (false) before continuing the communication.

After the PET places the primary address on the DIO lines at time ④ , the data is allowed to settle for 11 μs. The PET asserts the DAV line low (true) at time ⑤ , thus telling all devices on the bus that the primary talk address data is valid and can be read. This action initiates the handshake sequence.

Each device residing on the GPIB responds in its own time; each sets its NRFD line low (true) at time ⑥ , strobes the primary address byte off the DIO lines, then signifies it has done so by resetting its NDAC line high (false) at ⑦ . This part of the handshake sequence timing will vary, with the NDAC line returning to its high state in response to the slowest instrument.

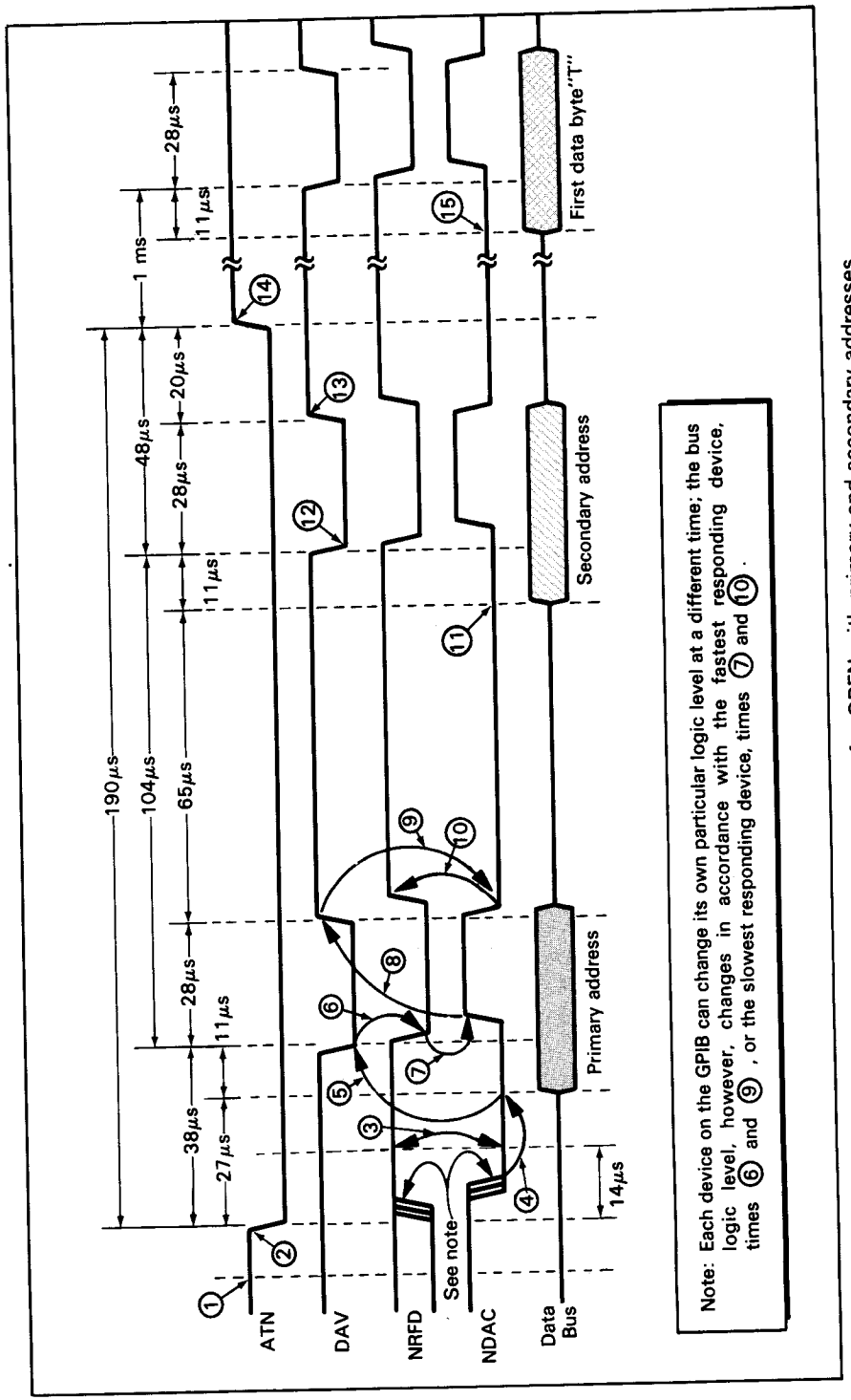


Figure 5-1. Address timing sequence for OPEN with primary and secondary addresses (OPEN 5,5,2 "TEST")

Table 5-2. The four types of GPIB device errors

Error	Problem	Time of Occurrence	Problem Indications			Meaning	Result: PET action and/or subsequent actions
			DAV	NRFD	NDAC		
1	No response from any bus device	During any signal transfer	No response	High	High	No physical device present	PET terminates program with a ? (device not present). An IEEE 488 subroutine sets the Status word (ST), bit 7 (-128).
2	Wrong level on NRFD or NDAC	PET writes first character to the bus	NA	Low	NA	Device not ready for data	Indicates device not present by hanging up the PET. The PET will not respond to any keyboard or program inputs.
3	No device response on the NDAC line for 65 ms	Data transfer from the PET	NA	NA	Low	Listener did not take data	Status word (ST), bit 1, is set.
4	No device response on the DAV line for 65 ms	Data transfer into the PET	High	NA	NA	Talker did not make data available	Status word (ST), bit 2, is set.

After the PET asserts the DAV line low (true) at time ⑤ , it allows 65 ms for the NDAC line to be reset high (false). If there is no device response, the PET sets the status word (ST) bit 1 to low (true), terminating the OPEN statement. This error is described in Table 5-2.

When the NDAC line finally goes high (false), the PET deduces that each device has accepted the primary address data byte. The PET then pulls DAV high at time ⑧ , telling all bus devices that address signals on the data lines are no longer valid. This occurs a minimum of 28 μs after DAV initially went low (true); the actual timing depends on when the slowest listener releases the NDAC line.

Sensing that the DAV signal is high, each peripheral device tries to set the NDAC line low (true) at time ⑨ .

When each device is ready to accept the next data byte, it sets its connection to the NRFD line high. However, the NRFD line will go high (false) only after the slowest device has responded at time ⑩ . When the NRFD line finally resets to a high logic level, the PET interprets this to mean that each GPIB device is ready to receive the secondary address byte or other data. This action ends the handshake procedure.

When the transfer control line DAV goes high (false) at time ⑧ during the primary listen address, the PET prepares to issue a secondary address 65 μs later. This is shown at ⑪ . (Note that the minimum time measured for a secondary address preparation was 56 μs.) At time ⑪ the PET places the secondary address on the eight DIO lines, where the signals are allowed 11 μs to stabilize. Following time ⑩ and before time ⑫ , the PET again checks the NRFD line for a high logic level. If NRFD is low when the check is made, the PET interprets

this as one or more of the peripheral devices not having had time to respond to the final sequence in the primary address handshake. The PET waits. If NRFD is high, the PET drives the DAV line low , initiating the handshake procedure for the secondary address at time (12) .

The sequence of timing for transfer control lines DAV, NRFD, and NDAC is exactly the same during the transfer of the secondary address as it was for the primary address. There may be slight differences in the times when NRFD is reset high (false), due to the various response times of the different devices on the GPIB.

The series of addressing events is brought to a close during the secondary address transfer when the PET pulls the DAV line high (false) at time (13) ; it waits 20 μs, then resets ATN high (idle state) at time (14) . The PET places the first data byte on the DIO lines at time (15) .

The PET transmits these listen addresses in a minimum time of 190 μs.

OPEN Address Timing — Primary Address Only

If the secondary address is not present in an OPEN statement, then the PET will issue the primary address only. For example, the statement:

OPEN 5,5 "TEST"

has no secondary address; addressing activity on the GPIB is reduced to a nominal 86 μs, as illustrated in Figure 5-2. In the absence of a secondary address, the GPIB primary address timing is still essentially the same as that described for Figure 5-1.

In Figure 5-2, the PET places the primary address on the eight DIO lines at time ① . Again it settles for 11 μs before DAV is asserted low (true) at time ② .

The first data byte is placed on the DIO lines just prior to the DAV line being set low (true) at time ③ , approximately 1000 μs after the ATN line is reset high (inactive).

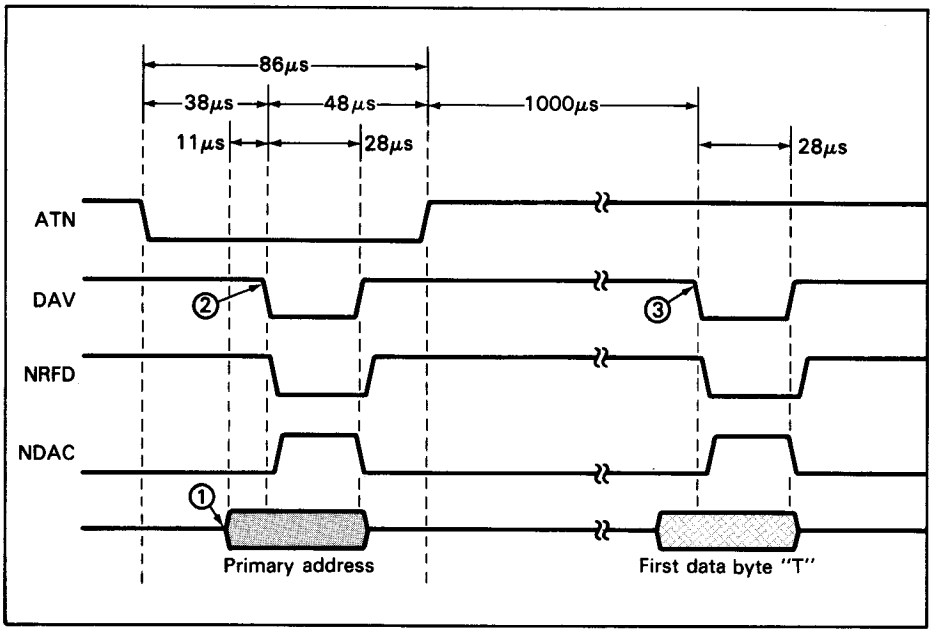


Figure 5-2. Address timing sequence for OPEN with primary address only (OPEN 5,5 "TEST")

TIMING SIGNALS FOR THE PRINT STATEMENT

Using the PRINT statement, we will now describe how the PET addresses a listener device, establishes the handshake sequence during a PRINT data burst, and finally how it deactivates the device from the bus by the unaddressing sequence.

An analysis of how messages reach a listening device would not be complete without noting exactly which characters are sent down the data bus in typical PRINT statements. Two such examples are given. Table 5-3 illustrates a program where the listener is addressed by both primary and secondary addresses. Table 5-4 illustrates a program where the listener is addressed by a primary address only. Table 5-4 indicates the same bus transactions as Table 5-3, with the exception that there is no secondary address issued. These sample transactions represent data that might actually be sent during the timing diagram sequences.

Table 5-3. Sample bus transactions for the PRINT statement with secondary address

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
OPEN 5,5,2 PRINT# 5, "TEST"	1	ATN	LAG 05	25	Primary address
	2	ATN	SCG 02	62	Secondary address set by OPEN
	3		T	54	
	4		E	45	
	5		S	53	
	6		T	54	
	7		CR	0D	
	8	EOI	LF	0A	
	9	ATN	UNL	3F	Bus unlisten

Table 5-4. Sample bus transactions for the PRINT statement without secondary address

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
OPEN 5,5 PRINT# 5, "TEST"	1	ATN	LAG 05	25	Primary address
	2		T	54	(Note no SCG 02)
	3		E	45	
	4		S	53	
	5		T	54	
	6		CR	0D	
	7	EOI	LF	0A	
	8	ATN	UNL	3F	Bus unlisten

PRINT Address Timing — Primary and Secondary Addresses

The statements given in Table 5-3:

```
OPEN 5,5,2
PRINT# 5, "TEST"
```

specify a print sequence to device number 5 (second 5 in OPEN statement) via assigned logical file number 5 (first 5 in OPEN statement and 5 of PRINT statement) with a secondary address of 2 (2 in OPEN statement).

The timing for this address sequence is shown in Figure 5-3. It is exactly the same as that shown in Figure 5-1 for the OPEN statement:

```
OPEN 5,5,2 "TEST"
```

A summary of timing events is given below for reference with Figure 5-3. For a detailed description, refer back to the section "OPEN Address Timing — Primary and Secondary Addresses."

Time	Event
1	Typical idle state.
2	PET sets ATN line low (true), indicating it is ready. ATN is low for 190 μ minimum.
3	External device responds by pulling NDAC low. Device has up to 14 μs to respond. Error if no response (see Table 5-2).
4	PET places primary address on DIO lines. Data is allowed 11 μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device responds by pulling NRFD low at the beginning of the read.
7	Each device pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the address data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
11	PET places secondary address on the DIO lines.
12	PET issues the data valid signal by pulling DAV low.
13	PET pulls DAV high to remove data valid indication.
14	PET resets ATN high, terminating the address timing.
15	PET places first data byte on the DIO lines.

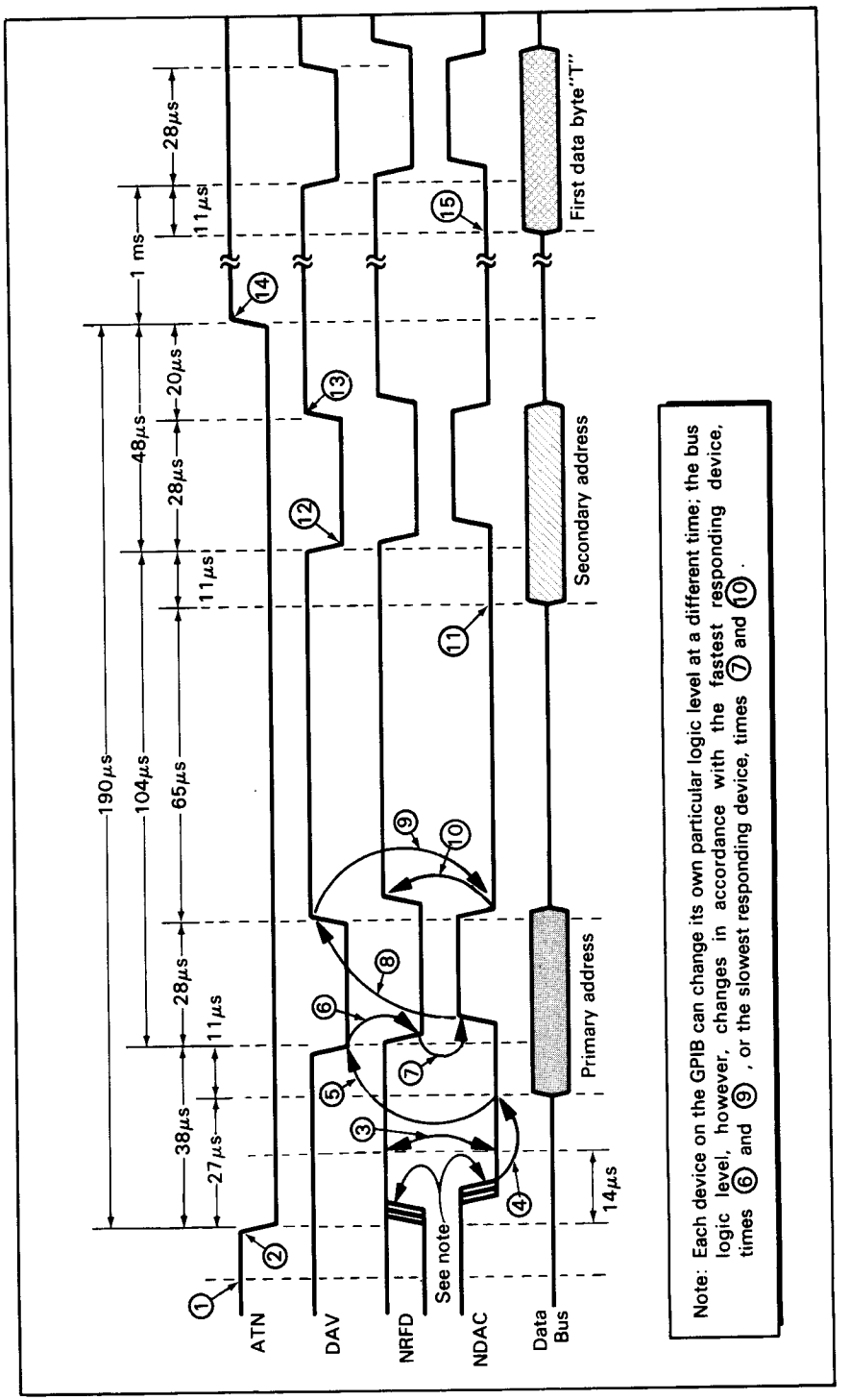


Figure 5-3. Address timing sequence for PRINT with primary and secondary addresses (OPEN 5,5,2 PRINT# 5 "TEST")

PRINT Address Timing — Primary Address Only

If the secondary address is omitted from an OPEN statement's parameters, as shown in Table 5-4, then the overall minimum addressing time in the PRINT sequence is cut to 84 μs. Figure 5-4 shows timing for a PRINT statement following an OPEN without a secondary address. The GPIB timing events for leading and trailing pulse edges of DAV, NRFD, and NDAC are nearly identical to the timing sequences for the primary and secondary addresses shown in Figure 5-3.

In Figure 5-4, the PET places data (in this instance addressing data) on the eight DIO lines at time ① , where it settles for 11 μs before DAV is driven low (true). The first data byte is placed on the eight DIO lines at time ② , just prior to the DAV line being set low (true) at time ③ . This is 960 μs after ATN is pulled high (inactive).

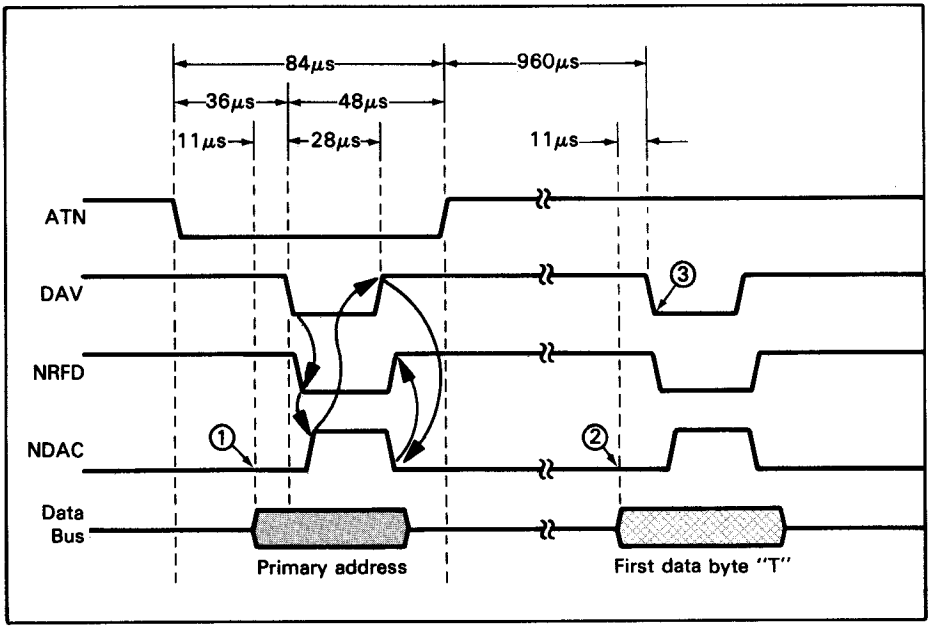


Figure 5-4. Address timing sequence for PRINT with primary address only (OPEN 5,5:PRINT# 5 "TEST")

The PRINT Data Burst — Transferring the Data

The actual data to be transferred between the PET and a device responding to a PRINT statement is specified in the PRINT statement. Timing for this data transfer is given in Figure 5-5 for optimum conditions, with a minimum time of 210 μs to complete the transfer of one byte of information. This yields a maximum data transfer rate of 4762 bytes per second.

Just prior to the data transfer, the PET can prepare a maximum of 192 characters for transfer; therefore, the data burst can last only while these 192 characters are transferred. Once this number of characters has been transferred, the PET prepares another group of 192 characters if more are to be transmitted. Thus, a long character string is transmitted in 192-character bursts, each character occurring every 210 μs. The GPIB will experience periods of activity interspersed with periods of inactivity. During this sequence of data transfer, the PET's clock interrupt and resulting keyboard scanning continue to run, using 1.7 ms every 16.6 ms.

When the ATN line has been pulled high (inactive) after the addressing sequence (Figures 5-3 and 5-4) the PET assumes that the addressed device (for example, a printer) has taken part in the handshake sequence, responded properly to the address, and is ready to capture data specified in the PRINT statement parameter list.

The data transfer sequence begins at time ① in Figure 5-5, 949 μs after ATN is reset high (inactive). The PET places data on the eight DIO lines, where it settles for 11 μs. Between times ① and ② the PET looks at the NRFD line to see if it is high. If it is not, the PET assumes that the addressed device is slow in responding and waits. If NRFD is high, the PET drives the DAV line low (true) and begins the handshake sequence. In the handshake now beginning, only one device on the GPIB — the one addressed in the previous sequence — is taking part. When this device senses that the DAV line is low (true), it reacts by setting NRFD low (true) at time ③ ; it accepts the data byte and resets the NDAC line high (false) at time ④ . The PET, in turn, acknowledges that NDAC is high (false) by pulling DAV high (false) at time ⑤ . Minimum time from DAV low to DAV high is 26 μs, as shown.

After the DAV line has been reset high (false), the PET readies the next data byte for transfer at time ⑧ . Meanwhile, the listener device returns NDAC to low (true) at time ⑥ , resets the NRFD line high (false) at time ⑦ , and prepares for the next handshake sequence; this follows placement of data on the eight DIO lines at time ⑧ . The PET places this next data byte on the DIO lines 173 μs after DAV is brought high (false) and waits 11 μs for the data to once again stabilize on the data bus.

In preparation for the next handshake, the PET senses the logic level of the NRFD line. If NRFD is low (true), the PET waits. If NRFD is high (false), having been reset by the listener device, the PET assumes the listener is ready for the next data byte and pulls the DAV line low (true) to initiate the next handshake sequence. This series of events is identical to those shown previously at times ③ through ⑦ in Figure 5-5.

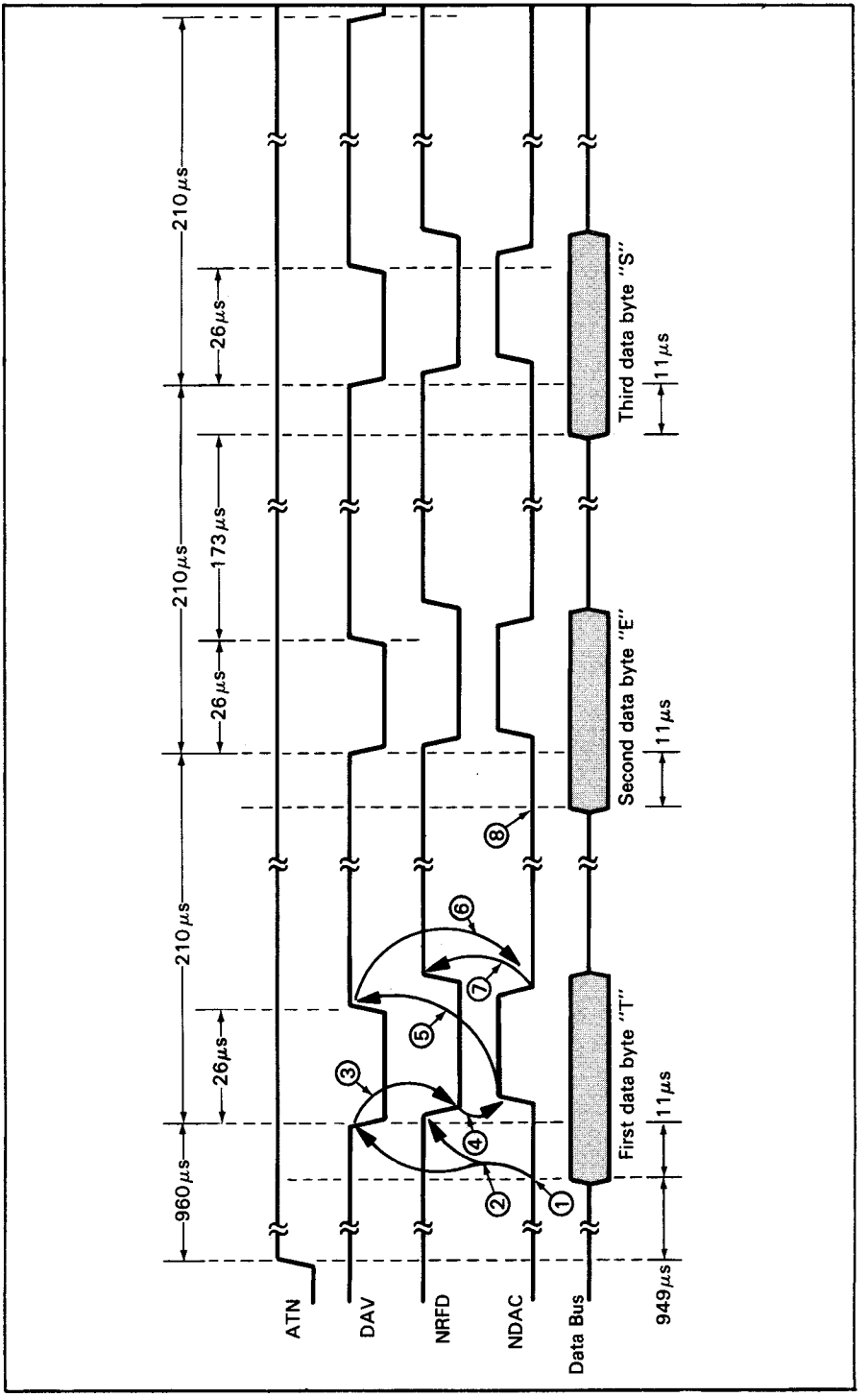


Figure 5-5. Data burst timing for the PRINT statement (PRINT# 5 "TEST")

PRINT Unaddress Timing

After the PET transfers the data in the PRINT statement to the listener, it concludes the entire sequence by issuing the untalk command on the DIO lines. This command releases the GPIB device. The released device is returned to its idle state.

The timing steps are shown in Figure 5-6. The PET asserts the ATN line low (true) to initiate the process. As shown, the untalk command is placed on the data bus, where it is allowed to settle for 11 μs. Then the PET again samples the NRFD line to see if it is high (false). If so, the PET sets DAV low (true) to begin the now familiar handshake sequence. The entire untalk sequence takes a minimum of 88 μs.

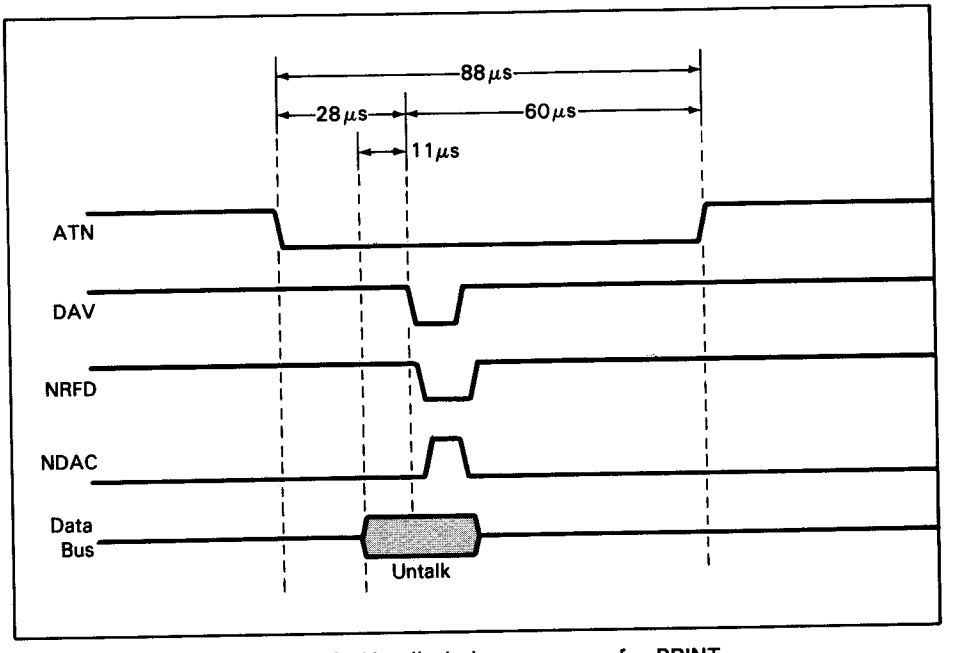
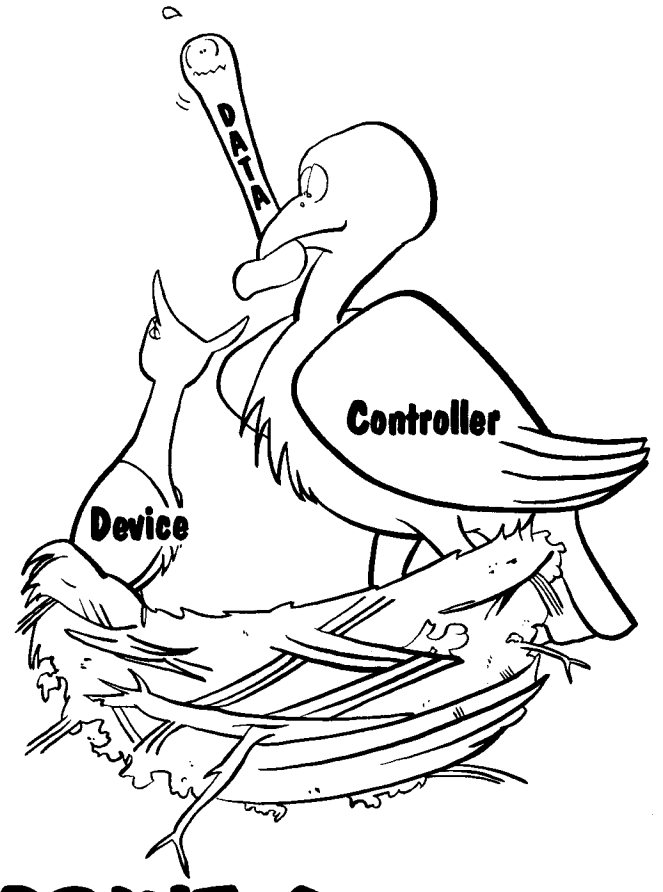


Figure 5-6. Untalk timing sequence for PRINT



PRINT Statement

TIMING SIGNALS FOR THE CMD (COMMAND) STATEMENT

Usually, a controller of peripheral devices attached to the GPIB will address and send PRINT commands to only one device on the bus at a time. PRINT is used irrespective of the addressed device, be it a hard-copy printer, a signal generator or a voltmeter, etc. For example, the PET programs a digital voltmeter to read resistance on the times-ten scale (a function and a range) by executing an appropriate PRINT# command.

At the end of the PRINT statement, the PET issues the universal unlisten command, which tells all instruments to disengage from the bus, even though only one device is normally engaged.

On occasion, you may want to have one or more instruments remain on the bus while data is being sent on the GPIB to another device. For example, the PET might send function and range data to the digital voltmeter and simultaneously record the data as hard copy on a printer. To accomplish this, a special form of the print command is used. This is the CMD (command) statement.

The following statements cause a program listing to be sent to a printer. The statements are typed into the PET in the immediate mode.

OPEN 5,5	This opens a file #5 for device #5, in our example a printer.
CMD 5	This issues a print command to device #5 without any data. It opens device #5 to the GPIB data bus, but does not unlisten the device.
LIST	The LIST command directs the PET to output a listing of a program stored in the PET's memory.

The OPEN 5,5 statement does two things. It tells the PET that data will be sent to a GPIB device identified as device number 5, and it names the logical file number 5. The CMD 5 statement tells the PET that all output, which normally would go to the CRT, must go to the GPIB and device number 5. Also, the CMD statement tells the PET not to unlisten the bus device, thus leaving it on line.

Normally LIST tells the PET to list a program on the CRT display; but since the PET has been given the CMD statement, the listing will go to the bus and to the printer (device 5) instead.

To release the PET and the external device from the CMD mode of operation, you must execute a PRINT# to an open file, or in the immediate mode type something like:

AAA [RETURN]

thus causing a syntax error.

Table 5-5 lists two sample bus transactions and shows the data that is actually transferred on the GPIB for these CMD statements.

In part a) of Table 5-5, under "Statements," the example shows an OPEN statement specifying secondary address 2. The OPEN statement is followed by CMD 5, "TEST." These two statements send a primary listen address of 5 (LAG 05 = Listen Address Group 5), with a value of 25_{16}, to the GPIB.

Table entry 2 shows the characters for the secondary address, 2 (SCG 02 = Secondary Command Group 2). Here the value 62_{16} is transferred to the eight DIO lines.

Table entries 3 through 6 show output of the characters "TEST" which were specified as the output data in the CMD statement. Each letter has a corresponding eight-bit binary number transmitted on the eight DIO lines of the GPIB.

Table entries 7 and 8 indicate a carriage return (CR) and a line feed (LF), respectively. Compare this CMD statement with the PRINT statement in Table 5-3 and notice that there is no bus unlisten command. Neither is there an EOI indication signifying an end of string.

Table entries 9 and 10 illustrate how, after the first carriage return and line feed are sent, another carriage return and line feed instruction are sent, because these instructions, which normally would have gone to the CRT screen, are sent on the GPIB.

Table 5-5 part b) is similar to part a), but the secondary command address is not issued.

Table 5-5. Sample bus transactions for the CMD statement:
a) With primary and secondary address, b) Primary address only

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
a) OPEN 5,5,2 CMD 5, "TEST"	1	ATN	LAG 05	25	Primary address
	2	ATN	SCG 02	62	Secondary address
	3		T	54	
	4		E	45	
	5		S	53	
	6		T	54	
	7		CR	0D	
	8		LF	0A	Notice there is no bus unlisten
	9		CR	0D	Normal CRT screen output
	10		LF	0A	will now be displayed (i.e., READY)
b) OPEN 5,5 CMD 5, "TEST"	1	ATN	LAG 05	25	Primary address
	2		T	54	
	3		E	45	
	4		S	53	
	5		T	54	
	6		CR	0D	
	7		LF	0A	Notice there is no bus unlisten

CMD Address Timing — Primary and Secondary Addresses

CMD address timing is exactly the same as for a PRINT sequence. Figure 5-7 shows the timing events that occur for the CMD sequence in part a) of Table 5-5, in which the OPEN statement has both a primary and a secondary address.

A summary of timing events is given below for reference with Figure 5-7. For a detailed description, refer back to the section "OPEN Address Timing — Primary and Secondary Addresses."

Time	Event
1	Typical idle state.
2	PET sets ATN line low (true), indicating it is ready. ATN is low for 190μs minimum.
3	External device responds by pulling NDAC low. Device has up to 14μs to respond. Error if no response (see Table 5-2).
4	PET places primary address on DIO lines. Data is allowed 11μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device repsonds by pulling NRFD low at the beginning of the read.
7	Each device pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the address data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
11	PET places secondary address on the DIO lines.
12	PET issues the data valid signal by pulling DAV low.
13	PET pulls DAV high to remove data valid indication.
14	PET resets ATN high, terminating the address timing.
15	PET places first data byte on the DIO lines.

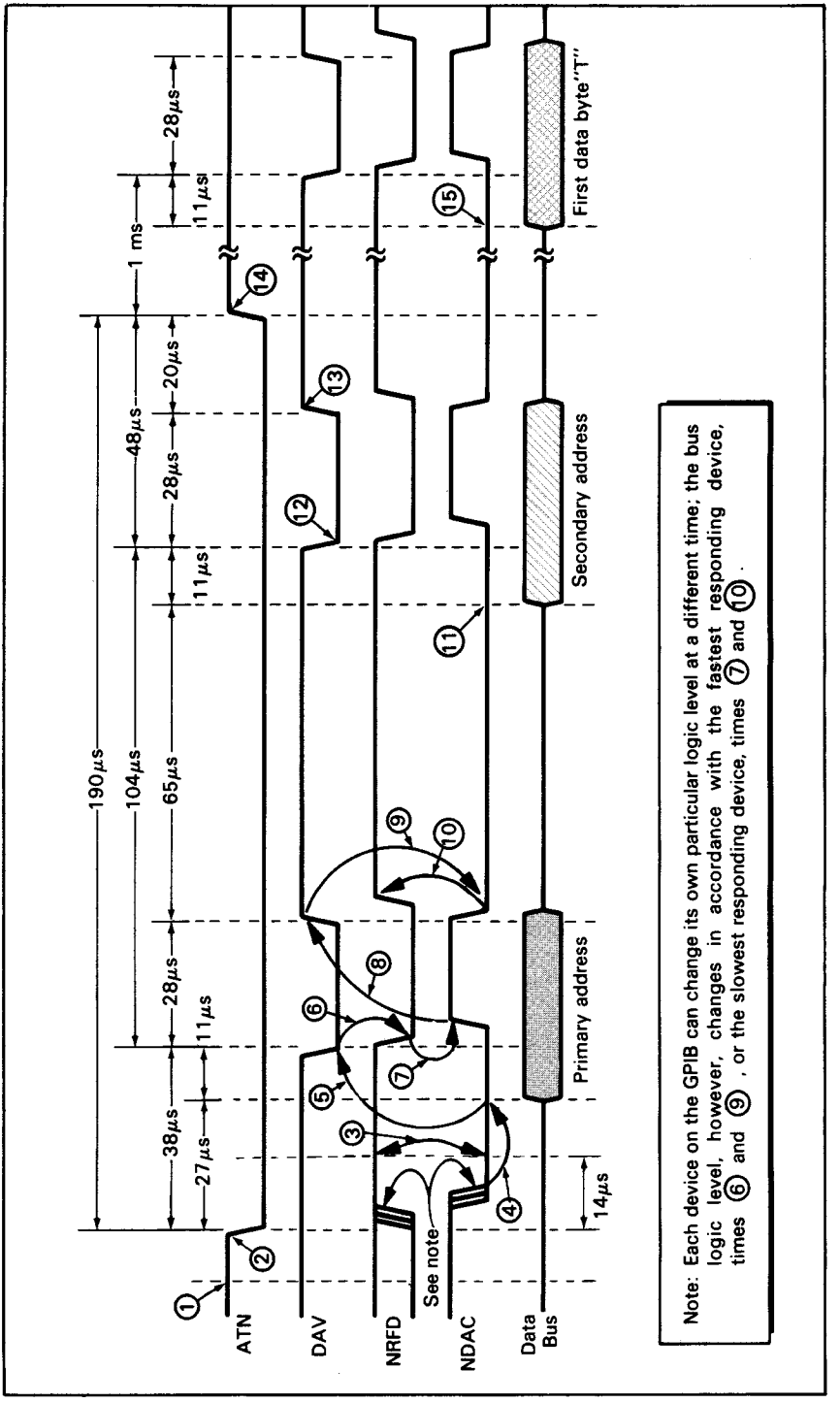


Figure 5-7. Address timing sequence for CMD with primary and secondary addresses (OPEN 5,5,2:CMD 5, "TEST")

CMD Address Timing — Primary Address Only

When the secondary address is omitted from a CMD sequence, as shown in part b) of Table 5-5, the overall minimum addressing time is cut to 84 μs. Figure 5-8 shows timing for a CMD statement following an OPEN without a secondary address. The timing is identical to that of the PRINT statement already discussed.

In Figure 5-8, the PET places data (in this instance addressing data) on the eight DIO lines at time ① , where it settles for 11 μs before DAV is driven low (true). The first data byte is placed on the eight DIO lines at time ② just prior to the DAV line being set low (true) at time ③ . This is 960 μs after ATN is pulled high (inactive).

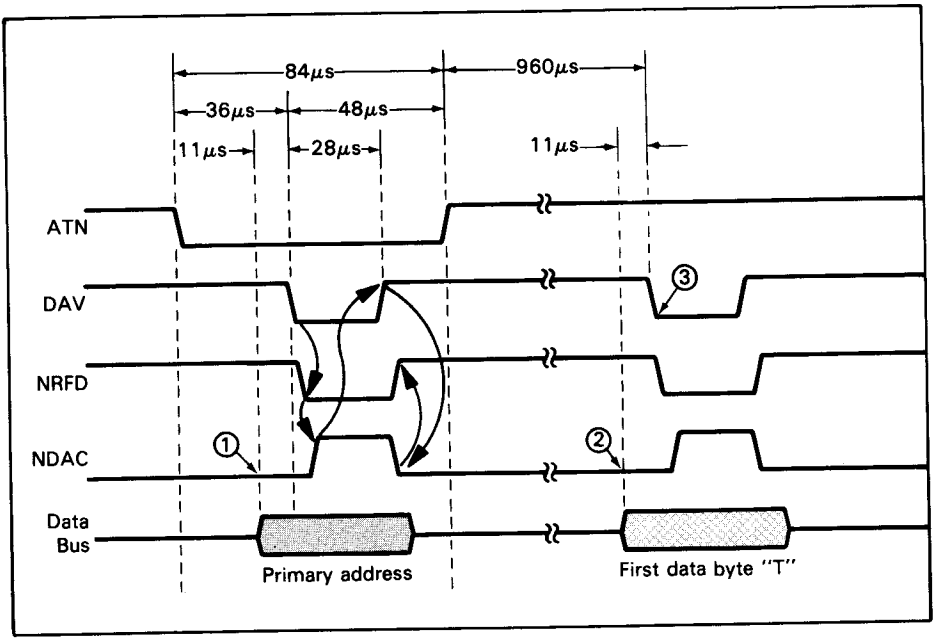


Figure 5-8. Address timing sequence for CMD with primary address only (OPEN 5,5:CMD 5, "TEST")

The CMD Data Burst — Transferring the Data

Timing for the CMD data burst is shown in Figure 5-9. The CMD data burst occurs in exactly the same way as the PRINT data burst, with characters transmitted at most every 210 μs in 192-character bursts, interrupted by 1.7 ms of PET interrupt service out of every 16.6 ms, giving a maximum data transfer rate of 4762 bytes per second. With CMD, however, more than one external device may listen; the addressed device takes part, plus any devices previously addressed by CMD sequences that are still active.

A summary of timing events for the data burst is given below for reference with Figure 5-9. For a detailed description, refer back to the section "The PRINT Data Burst — Transferring the Data."

Time	Event
1	PET places data on the DIO lines 949 μs after ATN is reset high (inactive) following the addressing sequence (Figure 5-7 or 5-8). Data is allowed 11 μs to settle.
2	PET asserts DAV low as data valid indicator.
3	Each device responds by polling NRFD low at the beginning of read.
4	Each active device pulls NDAC high (false) when read is complete.
5	PET pulls DAV high to indicate tht the data is no longer valid.
6	Each active external device sets NDAC low.
7	Each active external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
8	PET places next data byte on the DIO lines.

CMD Unaddress Timing

There is no unaddressing sequence for the CMD instruction. The listening device remains open, thus making it the primary output device instead of the CRT.

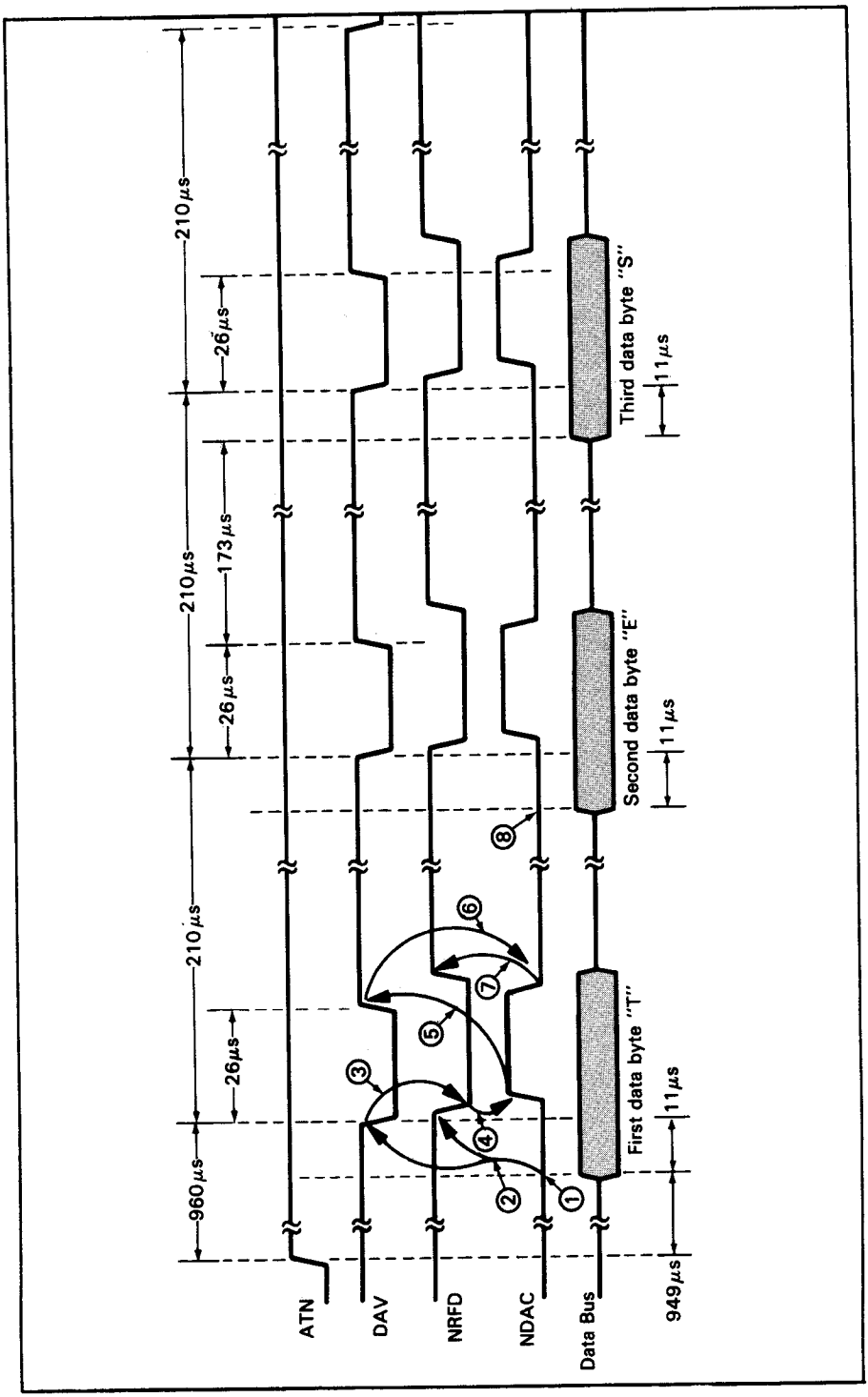


Figure 5-9. Data burst timing for the CMD statement (CMD 5, "TEST")

TIMING SIGNALS FOR THE SAVE COMMAND

The SAVE command provides a means of "saving" a BASIC program by transferring it directly from the PET's memory via the GPIB to a remote listener. The SAVE command has the format:

SAVE "name", device number

A name within quotes causes the PET to automatically issue a special secondary command in the first two bytes of bus transaction. In this way, the SAVE command initiates the transfer of a BASIC program from a named file to the GPIB. These and other SAVE features are shown by the sample bus transactions in Table 5-6.

Sample Bus Transactions — Old ROMs

The SAVE statement discussion which follows pertains to PET units that have original, or "old" ROMs. These ROMs, with their original programming, were first installed in PET computers having 4K or 8K of RAM and manufactured through 1979. These PETs are distinguished by asterisks (***) in the title line on power-up.

Table 5-6 identifies the data that is actually transferred on the GPIB for the statement:

SAVE "TEST", 5

Table entry 1 identifies the start of the bus transactions. At this time, 25_{16} is sent to the GPIB.

The numbers and letters in the "Characters" column do not appear on the PET CRT display nor on a display that listener 5 might have. Rather, these characters, as specified in the IEEE Std 488-1978, are part of the internal control functions for the PET and all devices residing on the bus. LAG is the standard character group that identifies listeners, in this case listener 5.

The first character group and the SCG 17, table entry 2, are generated automatically as a result of the SAVE command. SCG 17 is the secondary address, which is sent on the bus by the code $F1_{16}$. This is a unique secondary address which indicates that a SAVE is to be sent from the PET. The SAVE command cannot be issued by the PET using a primary address only.

Table entries 3 through 6 identify the name of the program as "TEST," each letter being sent from the PET to listener 5 as a separate byte of information. At table entry 6, the EOI (End Or Identify) interface management line is asserted low (true) by the PET, now in the role of talker, to signify the last data byte in this group.

Table 5-6. Sample bus transactions for the SAVE command (old ROMs)

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
	1	ATN	LAG 05	25	
	2*	ATN	SCG 17	F1	PET's SAVE secondary address
	3*		T	54	
	4*		E	45	
	5*		S	53	
	6*	EOI	T	54	
	7*	ATN	UNL	3F	
	8*	ATN**	UNL	3F	
	9			00	Delimiter
	10			09	Next BASIC instruction at 0409_{16}
	11			04	
SAVE "TEST", 5	12			05	Line # 05
	13			00	
	14		REM	8F	REM token
	15		5	35	
	16		5	35	
	17			00	Delimeter
	18			00	Delimiter
	19			00	Delimiter
	20	EOI		24	Free (unused) memory
	21		UNL	3F	Bus unlisten
	22		UNL	3F	Bus unlisten
	23			E1	Close file

*These characters (entries 2-8) are not sent if there is no file name (e.g., "TEST") used in the SAVE statement. ** The ATN line remains low (true).

At table entries 7 and 8 ATN is again asserted; $3F_{16}$ is placed on the data bus, causing UNL (the universal unlisten command) to be transferred to the listener.

The delimiter byte, table entry 9, identifies the beginning of the PET BASIC program memory image area. At entries 10 and 11, 09_{16} and 04_{16} are a memory pointer, stored with the least significant byte first. These numbers identify (point to) the next BASIC statement in memory. In this example, the next BASIC statement begins at memory location 0409_{16}. Note carefully that the most significant byte is transferred last.

The next two bytes, table entries 12 and 13, identify where the BASIC instruction is to be saved. Table entries 14 through 16 identify, in this case, a very short program. Entry 14 shows the PET BASIC token for REMARK (abbreviated REM), $8F_{16}$. The remaining parts of this short program are the numerals 5 and 5 at table entries 15 and 16. These table entries show that the entire program being saved consists of a single Remark statement, REM 55.

At table entries 17 through 19 there are three delimiters, hexadecimal 00 bytes, that are necessary to end the BASIC program.

Table entry 20 shows the EOI interface management line asserted. This is a function of the PET's internal operating system. In this particular example 24_{16} happens to be the next available memory location in the PET's memory, and this one extra byte is sent to the GPIB.

At table entries 21 and 22, the PET sends two UNL commands on the data bus, one of which is necessary to cause GPIB devices to disengage from the bus.

Table entry 23 shows the indicator that closes the PET's BASIC file for the SAVE command; this is $E1_{16}$.

Notice that the ATN line is asserted by the PET throughout the output message, table entries 9 through 23. Therefore, this portion of the SAVE command does not follow the IEEE Std 488-1978 transactions; the devices conforming to the standard and residing on the GPIB could malfunction due to their interpreting that they have been addressed when they have not.

Sample Bus Transactions — New ROMs

The discussion which follows pertains to PET units that have the "new" ROMs. These PETs are distinguished by pound signs (###) in the title line on power-up. In these machines, the SAVE statement is completely implemented in the IEEE 488 Bus function. The following information and table of bus transactions use a SAVE statement similar to the one discussed earlier, so you can compare the two.

Table 5-7 contains characters and numbers identifying the data that is actually transferred on the GPIB for the statement:

SAVE "TEST", 8

At table entries 1 and 2, the first and second character groups (LAG 08 and SCG 17) are issued automatically as a result of the SAVE command. The latter is a unique secondary address, $F1_{16}$, which indicates that a SAVE is to be sent from the PET. The SAVE statement cannot be issued by the PET using a primary address only.

Table entries 3 through 6 identify the name of the program as "TEST," each letter being sent from the PET to listener 8 as a separate byte of information. At table entry 6, the EOI (End Or Identify) interface management line is asserted low (true) by the PET, now in the role of talker, to signify the last data byte in this group.

At table entry 7, ATN is again asserted. The number $3F_{16}$ is placed on the data bus, causing UNL (the universal unlisten command) to be transferred to the listener.

At table entries 8 and 9, bus device number 8 is again addressed. The ATN line is lowered and raised for each transaction in the normal manner and in accordance with the IEEE 488 Bus standard; the program data then follows.

Beginning at table entry 10 and continuing through table entry 21, the entire program is transferred on the GPIB from the PET to listener number 8. The actual program saved is shown at table entries 16 through 18.

At table entries 22 through 25, the ATN line is lowered four times by the PET to bring the bus transactions to a close. Refer to the "Notes" column of the table for the meaning of these last four table entries.

Table 5-7. Sample bus transactions for the SAVE command (new ROMs)

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
SAVE "TEST", 8	1	ATN	LAG 08	28	Device listen address
	2	ATN	SCG 17	F1	PET's SAVE secondary address
	3*		T	54	
	4*		E	45	
	5*		S	53	
	6*	EOI	T	54	
	7*	ATN	UNL	3F	Bus unlisten
	8*	ATN	LAG 08	28	
	9*	ATN	SCG 01	61	
	10			01	Statement location in memory
	11			04	
	12			09	Pointer to next statement
	13			04	
	14			05	Line # 05
	15			00	
	16		REM	8F	REM token — Program being saved
	17		5	35	REM token
	18		5	35	REM token
	19			0	Delimiter
	20			0	
	21	EOI		0	
	22	ATN	UNL	3F	Bus unlisten
	23	ATN	LAG 08	28	Device listen address
	24	ATN	SCG 01	E1	PET's CLOSE file
	25	ATN	UNL	3F	Bus unlisten

*These characters (entries 3-9) are not sent if there is no file name

SAVE Address Timing — Primary and Secondary Addresses

In Figure 5-10, the primary and secondary addresses and the first data byte for the foregoing bus transactions are examined in detail.

Bear in mind that a predetermined secondary address code is issued whenever a file name is given as a parameter in the SAVE command, as shown for our examples in Tables 5-6 and 5-7. A secondary address may also be specified as the third parameter in a SAVE command of the form:

SAVE "file", device, secondary command

The address timing events shown for SAVE in Figure 5-10 are exactly the same as described for previous statements. A summary of timing events is given below for reference with Figure 5-10. For a detailed description, refer back to the section "OPEN Address Timing — Primary and Secondary Addresses."

Time	Event
1	Typical idle state.
2	PET sets ATN line low (true), indicating it is ready. ATN is low for 190μs minimum.
3	External device responds by pulling NDAC low. Device has up to 14μs to respond. Error if no response (see Table 5-2).
4	PET places primary address on DIO lines. Data is allowed 11μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device responds by pulling NRFD low at the beginning of the read.
7	Each device pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the address data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handhsake sequence.
11	PET places secondary address on the DIO lines.
12	PET issues the data valid signal by pulling DAV low.
13	PET pulls DAV high to remove data valid indication.
14	PET resets ATN high, terminating the address timing.
15	PET places first data byte on the DIO lines.

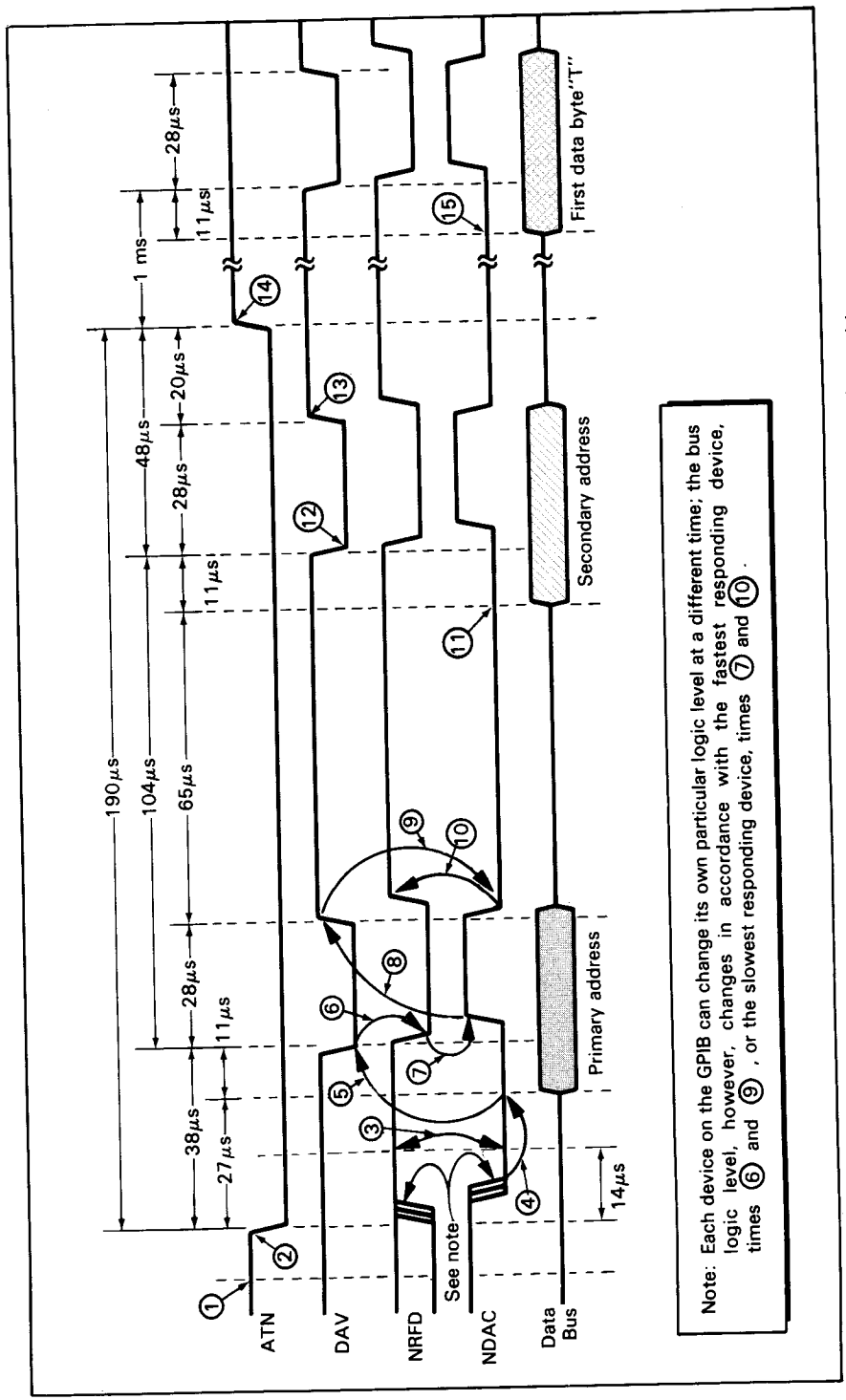


Figure 5-10. Address timing sequence for SAVE with primary and secondary addresses (SAVE "TEST", 5)

The SAVE Data Burst — Transferring the Data

The SAVE data burst is very similar to other PET data bursts. The difference is that, for PET controllers having old ROMs, the ATN line remains low throughout the SAVE data burst sequence.

As with the other data bursts, the PET transmits SAVE data characters a maximum of one every 210 μs in 192-character bursts, interrupted by 1.7 ms of PET interrupt service out of every 16.6 ms. This gives a maximum data transfer rate of 4762 bytes per second.

Timing for the SAVE data burst is shown in Figure 5-11. The ATN line remains low (true) following the transfer of the UNL command and continues low throughout the transfer of data initiated by the SAVE command. The PET assumes that the addressed device (for example, a floppy disk) has taken part in the handshake sequence, responded properly to the address, and is ready to capture data specified as a result of the SAVE command.

A summary of timing events for the data burst is given below for reference with Figure 5-11.

Time	Event
1	PET places data on the DIO lines following the addressing sequence (Figure 5-11). Data is allowed 11μs to settle.
2	PET asserts DAV low as data valid indicator.
3	Each active device responds by pulling NRFD low at the beginning of the read.
4	Each active device pulls NDAC high (false) when read is complete.
5	PET pulls DAV high to indicate that the data is no longer valid.
6	Each active external device sets NDAC low.
7	Each active external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
8	PET places next data byte on the DIO lines.

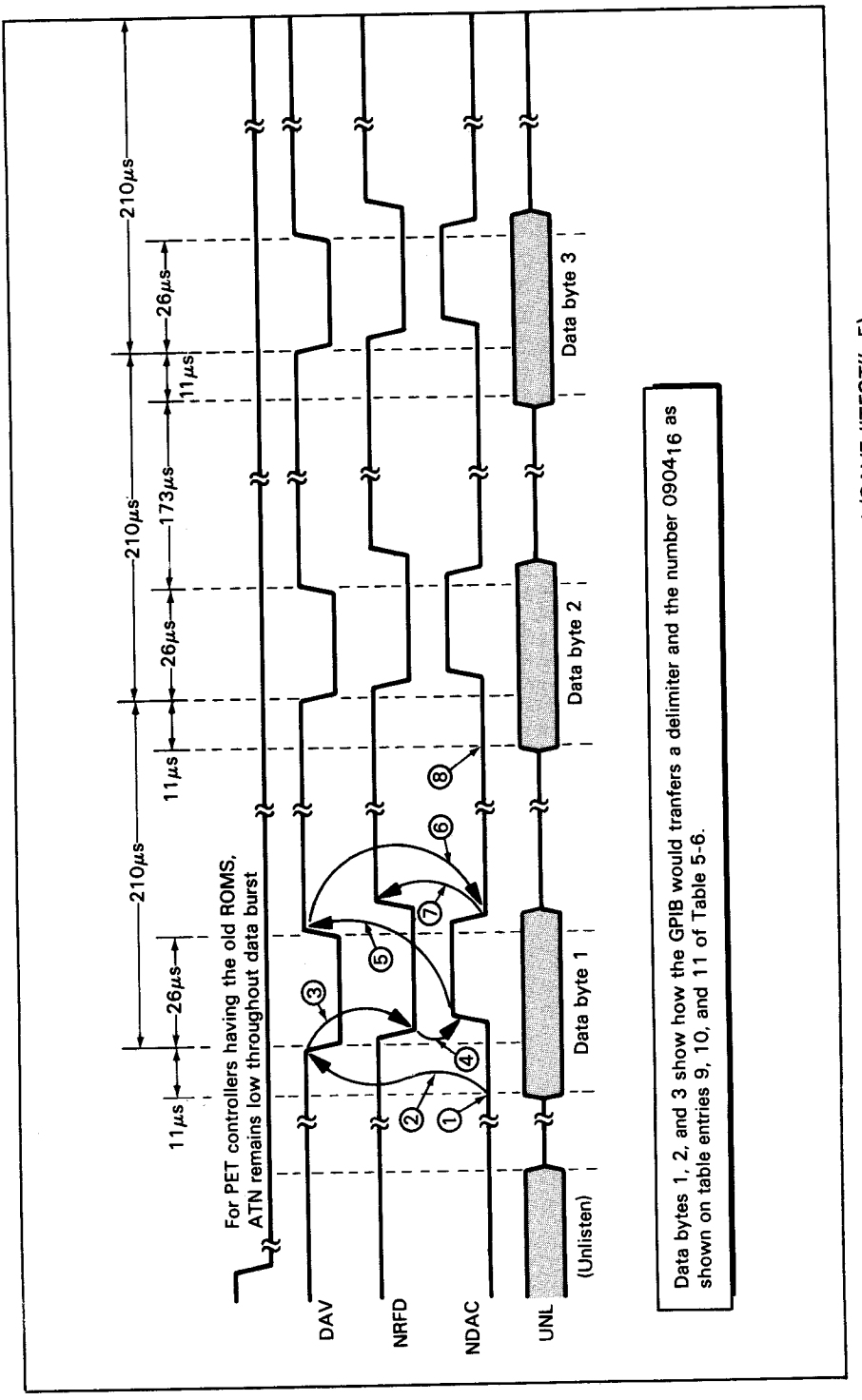


Figure 5-11. Data burst timing for the SAVE command (SAVE "TEST", 5)

TIMING SIGNALS FOR THE INPUT STATEMENT

We will now examine timing for signals on the GPIB that occur following execution of an INPUT statement. As in previous sections, the text will refer to timing diagrams, with changes in logic levels identified by circled numbers.

For the INPUT statement, several transactions occur on the GPIB: the addressing sequence (with and without secondary address), the data burst, and the unaddressing sequence.

Data is transferred on the bus as shown by the examples in Table 5-8. The handshake timing sequence is identical for received data and control characters. The PET differentiates between data and control characters once they have been received.

Table 5-8. Sample bus transactions for the INPUT statement:
a) With primary and secondary address, b) Primary address only

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
a) OPEN 5,5,2 INPUT# 5,A$	1	ATN	TAG 05	45	Primary talk address
	2	ATN	SCG 02	62	Secondary address
	3			XX	Remote, talker-supplied information
	4			XX	
	5			XX	
	6			XX	
	7	EOI	CR	0D	
	8	ATN	UNT	5F	Bus untalk
b) OPEN 5,5 INPUT# 5,A$	1	ATN	TAG 05	45	Primary talk address
	2			XX	Remote, talker-supplied information
	3			XX	
	4			XX	
	5			XX	
	6	EOI	CR	0D	
	7	ATN	UNT	5F	Bus untalk

INPUT Address Timing — Primary and Secondary Addresses

Before the INPUT statement can transmit data on the data lines, the talker device must be addressed using a primary address or a primary and secondary address. Timing events for addressing an INPUT device with both primary and secondary addressing are shown in Figure 5-12.

The timing sequence shown in Figure 5-12 for INPUT is the same as shown for previous BASIC statements with a secondary address. A summary of timing events is given below for reference with Figure 5-12. For a detailed description, refer back to the section "OPEN Address Timing — Primary and Secondary Addresses."

Time	Event
1	Typical idle state.
2	PET sets ATN line low (true), indicating it is ready. ATN is low for 190μs minimum.
3	External device responds by pulling NDAC low. Device has up to 14μs to respond. Error if no response (see Table 5-2).
4	PET places primary address on DIO lines. Data is allowed 11μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device responds by pulling NRFD low at the beginning of the read.
7	Each device pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the address data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
11	PET places secondary address on the DIO lines.
12	PET issues the data valid signal by pulling DAV low.
13	PET pulls DAV high to remove data valid indication.
14	PET resets ATN high, terminating the address timing.
15	PET places first data byte on the DIO lines.

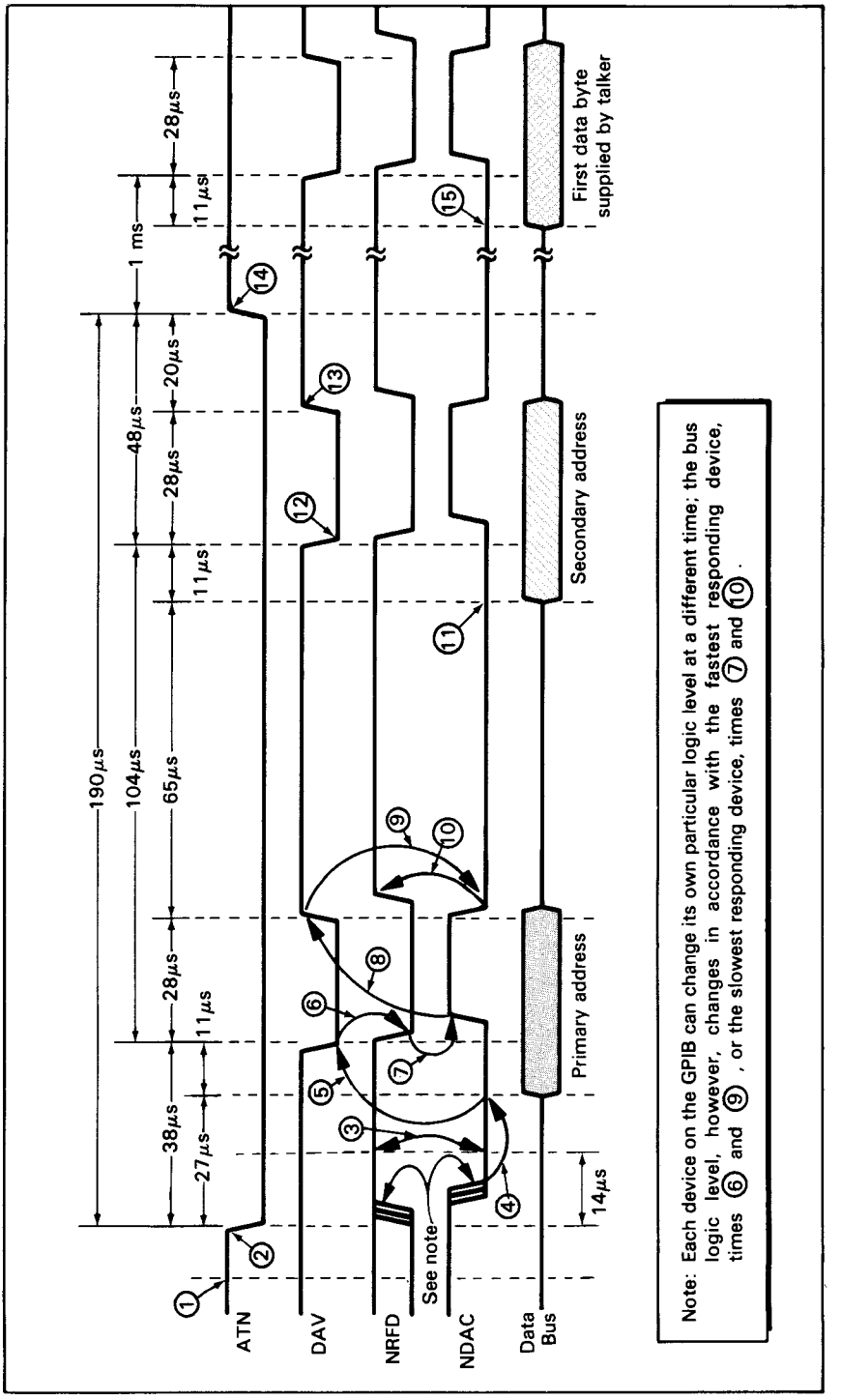


Figure 5-12. Address timing sequence for INPUT with primary and secondary addresses (OPEN 5,5,2: INPUT# 5,A$)

INPUT Address Timing — Primary Address Only

Figure 5-13 illustrates the handshake timing sequence for the PET and GPIB talkers when only a primary address is issued. Note that the secondary address handshake sequence is missing and the primary address is followed immediately by the data that the talker supplies. At the close of the addressing sequence for the INPUT statement, the PET asserts both its NDAC and NRFD lines low (true) and then releases the ATN line. The PET then prepares itself to capture data bytes transferred on the GPIB from the addressed instrument. The sequence of events for issuing this primary talk address is identical to that just described when both primary and secondary addresses are present.

The INPUT Data Burst — Transferring the Data

The PET INPUT statement timing cannot be represented accurately during buffer overhead periods. There are periods of inactivity during a received data burst when the PET I/O buffer is dumped following receipt of a delimiter or terminator character. While these periods cannot be represented accurately in the timing diagrams, designers of PET- and GPIB-compatible devices should be aware that these periods of inactivity exist, and take them into account when determining input data rates.

Figure 5-14 illustrates the INPUT data timing events as they occur after the PET pulls the ATN line high (inactive).

When the PET is to assign itself as a listener, it first releases the ATN line at time ① ; this indicates to the talker residing on the GPIB that it can prepare to place its first byte of data on the DIO lines. Sometime between time ② and time ④ , when the PET pulls its NRFD line high (false), the talker places its data on the GPIB data lines. There is a fairly long waiting period (80 μs) after ATN goes high before the PET releases its NRFD line to high at time ④ ; this ensures that a talker has ample time to place its data on the bus and have it stabilize. Data from a remote device should be on the data bus at time ③ , between 2 μs and 11 μs before the remote device sets its DAV line low (true) at time ⑤ .

As mentioned above, the PET first releases its ATN line and then 80 μs (minimum) later raises its NRFD line high (false). When NRFD is high, then the remote talker can lower its DAV line to the true state. This is illustrated at time ⑤ , and the action tells the PET that data on the DIO lines is valid. The PET waits only 65 ms for the talker to place the data on the bus, then sets bit 2 of the status word (see Table 5-2). When the EOI line is sensed, before a carriage return (CR) is available on the DIO lines, status bit 6 is set and CR is forced to the input.

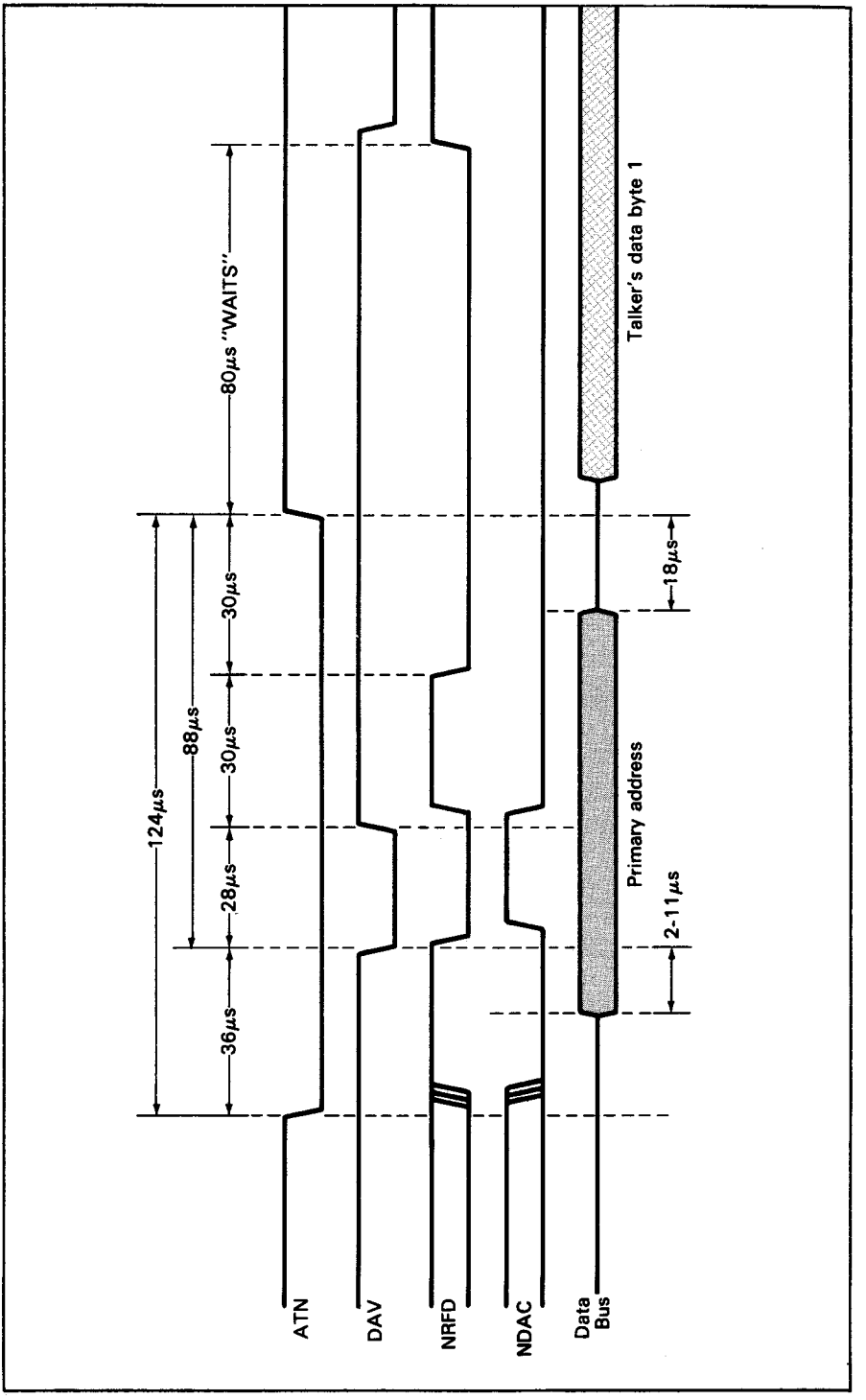


Figure 5-13. Address timing sequence for INPUT with primary address only (OPEN 5,5:INPUT# 5,A$)

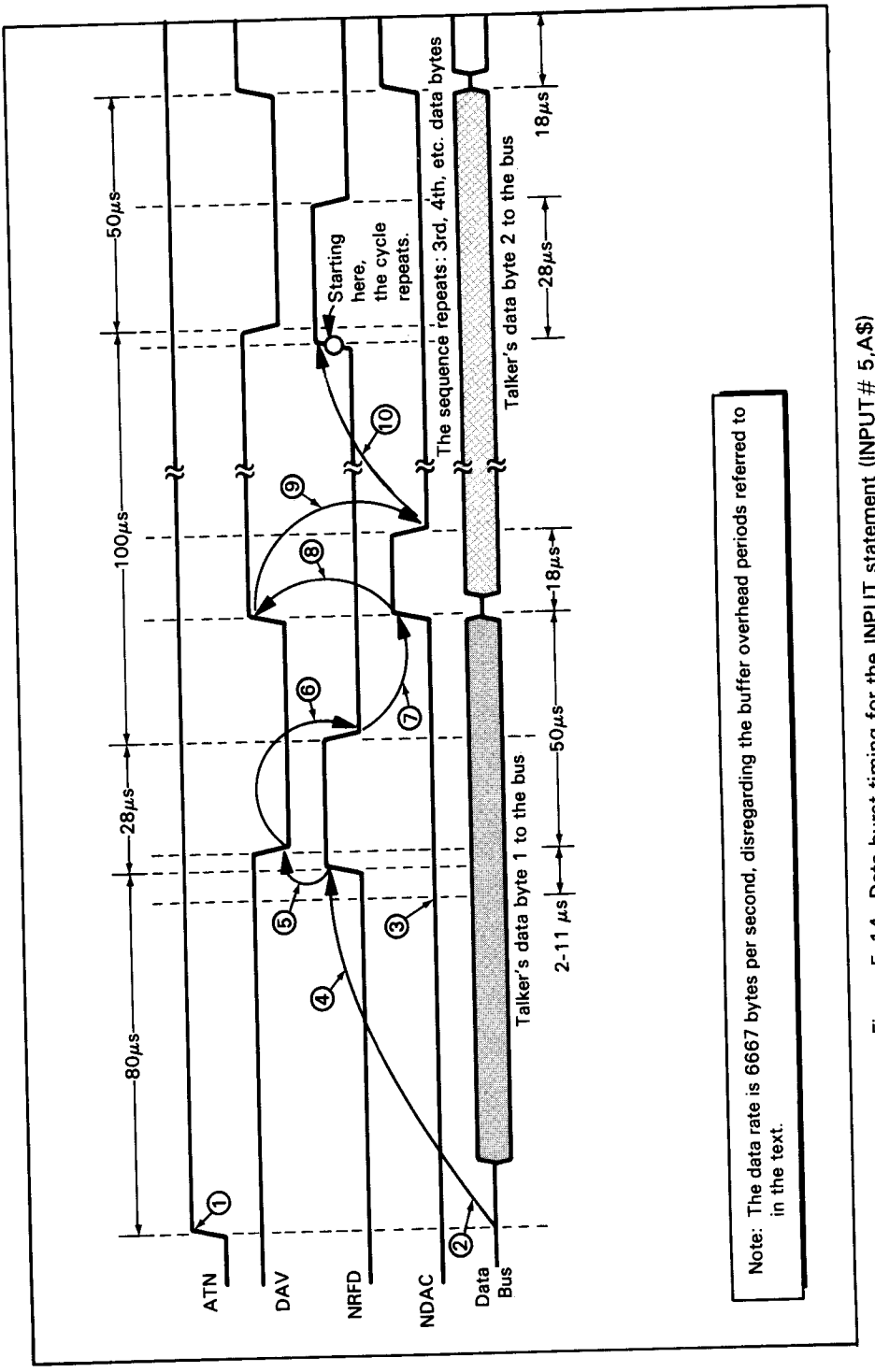


Figure 5-14. Data burst timing for the INPUT statement (INPUT# 5,A$)

CAUTION: On some PET ROMs currently manufactured, this will cause an unrecoverable error and the PET power must be cycled.

Next the PET responds by setting its NRFD line low (true), which takes a minimum of 28 μs. See time ⑥ . The PET reads the data byte and strobes it into its I/O buffers. To tell the remote talker that the data byte has been received, the PET raises its NDAC line high (false) at time ⑦ . The response time for the PET to raise its NDAC line is typically 50 μs after the talker asserts its DAV line low (true).

Next the remote talker responds at time ⑧ by resetting its DAV line high (false). This operation indicates to the PET that the talker's byte of data on the DIO lines is no longer valid. Following this, the remote talker can transmit the next data byte.

While the talker transmits its next data byte, the PET at time ⑨ asserts its NDAC line low (true); then, over the next 82 μs, the PET examines this byte of data to determine if it is a CR (carriage return) delimiter. If it does not find a CR delimiter, the PET raises its NRFD line high (false) at time ⑩ , thus preparing itself to capture the remote talker's next data byte. The PET continues to receive consecutive data bytes in this fashion until 80 characters are received or the carriage return (CR) delimiter is sensed. Either one of these situations occurring will cause the PET to stop the data transfer.

CAUTION: With the original ROMs installed in the PET, more than 80 characters input before a CR will cause a fatal error. To recover, you must power down the PET.

When sensing either of the above conditions, the PET asserts its NRFD line low (true) to stop the data flow, then it processes the information in its I/O buffer.

The PET converts input data to the data type specified by the INPUT statement. There are three data types: floating point, integer, and string.

If the INPUT statement specifies either a floating point or an integer variable, the PET will convert received data from ASCII to internal floating point format, then assign the data to the specified variable or array element. Integers are further converted to integer storage format.

If the INPUT statement specifies a string variable, then the PET will convert received data, up to the first CR or alternate delimiter, to its internal ASCII format, then assign the characters to the specified string variable or array element.

The PET receives data from the DIO lines until it detects a delimiter. At that time the PET assigns received data to the specified variable name.

The time required by the PET to convert and assign received data varies, and is dependent on the type of data conversion.

As illustrated in Figure 5-14, data from the GPIB is processed sequentially by the PET. A minimum of 150 μs is required to process each character. However, PET interrupt processing and memory management uses 1.7 ms every 16.6 ms, effectively slowing the rate of data transfer.

The PET continues to alternately receive data from the talker and process the received data until a delimiter is detected; then the PET again stops and processes received data. This sequence repeats until all variables specified by the INPUT statement have been assigned values. Remember, with certain of the PET ROMs, talking devices that transmit fewer than 80 characters must end with a carriage return. Failure to do so will result in a fatal error.

INPUT Unaddress Timing

Figure 5-15 illustrates the transfer control line timing on the GPIB during the unaddressing sequence for the INPUT statement.

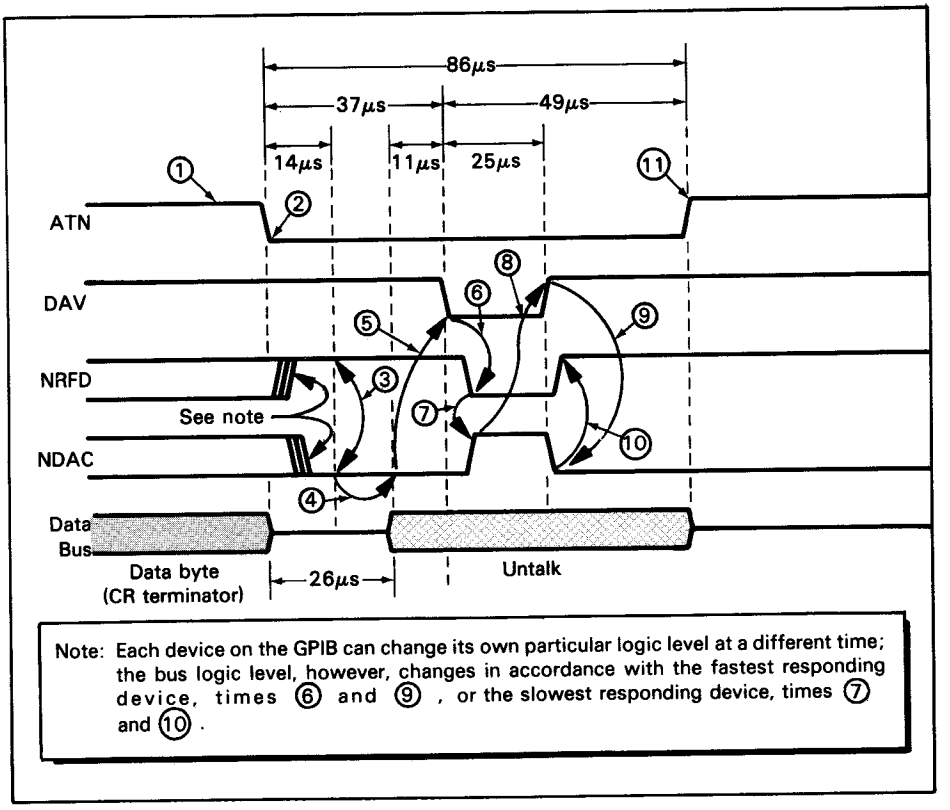


Figure 5-15. Unaddress timing sequence for the INPUT statement (INPUT# 5,A$)

Unaddressing occurs at the end of each INPUT statement; it commences at time ① with ATN and DAV inactive high (false). The NRFD and NDAC lines can be either high or low. If NRFD is low, it is released high after ATN is asserted. Similarly, the NDAC line can be brought low (true) at different times according to the response time of a particular device residing on the bus. (The logic level of NDAC is not a determining factor in ATN being set to its low (true) state.)

The PET asserts the ATN line low (true) at time ② , immediately following the reception of the CR terminator. By setting ATN low, the PET initiates the untalk sequence for the GPIB, telling all devices to release the bus.

After the PET drives the ATN line low at time ② , the remote talker is allowed 14 μs to respond by setting its NDAC line low and its NRFD line high at time ③ . If NRFD is low, the PET assumes the talker on the GPIB has not had time to respond, and waits. When the remote talker resets the NRFD line high (false), the condition shown at time ③ , the PET interprets this as the talker being ready to proceed with the untalk handshake.

At time ④ , 26 μs after the PET asserts ATN low, the PET assumes that the talker is ready and transmits the untalk data byte ($5F_{16}$), which stabilizes for 11 μs.

At time ⑤ the PET asserts its DAV line low (true) telling the talker that the untalk command is valid. Every device receives the untalk command, as it would any other data byte, using the timing described earlier.

Although only one talker was active during the INPUT sequence, each device residing on the GPIB is expected to answer the ATN signal. Each device individually sets its NRFD line low, receives the untalk command, then signifies it has done so by resetting its NDAC line high. This action is shown at times ⑥ and ⑦ .

At time ⑧ the PET releases its DAV line to a high state, thus indicating that the untalk data on the DIO lines is no longer valid. The active talker responds by setting its NDAC line low at time ⑨ ; then it resets its NRFD line high at time ⑩ . As each device is acting individually, the times at ⑥ , ⑦ , ⑨ , and ⑩ can vary.

49 μs after the PET asserts its DAV line low (true) to indicate valid untalk data, it releases the ATN line to its high (inactive) state, time ⑪ . The INPUT unaddressing sequence takes 86 μs minimum.

THE LOAD COMMAND

Sample Bus Transactions — Old ROMs

The discussion which follows pertains to PET units that have the original, or "old" ROMs. The ROMs, with their original programming, were first installed in PET computers having 4K or 8K of RAM and manufactured through 1979. These PETs are distinguished by asterisks in the title line on power-up.

The LOAD command is intended to load programs into the PET's user area of memory from an external device, with execution by the PET to be carried out later. This statement, which uses the IEEE 488 Bus functions, is not completely implemented in PET units having the old ROMs; however, the following information is included to document what happens.

Table 5-9 lists the GPIB transactions for the statement:

```
10 LOAD "TEST", 5
```

Table 5-9. Sample bus transactions for the LOAD command (old ROMs)

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
10 LOAD "TEST", 5	1	ATN	TAG 05	45	
	2	ATN	SCG 17	F1	
	3		T	54	
	4		E	45	
	5		S	53	
	6		T	54	
	7	ATN	UNT	5F	Bus untalk
	8	ATN		7F	
	9	ATN			First input character at location 400_{16} (1024_{10})

When the LOAD command is executed, the word "TEST," being within quotes, is sent to the GPIB.

As table entries 1 and 2 indicate, the PET asserts ATN low (true) while the Talk Address Group (TAG 05) and the Secondary Command Group (SCG 17) transfer their equivalent 1-byte codes to the bus.

The Talk Address Group is generated from the device number 5 parameter in the LOAD statement. The Secondary Command Group, although not specified in the LOAD command, is $F1_{16}$; it is reserved to indicate that a LOAD (or SAVE) is in progress.

Table entries 3 through 6 show the program file name "TEST," which the PET transmits to the external device. This is the title of the program to be loaded into the PET's memory.

Next, at entry 7, the PET again asserts ATN while the untalk command UNT is issued to the GPIB as $5F_{16}$. This is followed at table entry 8 by a unique code, $7F_{16}$, the last transmission before input data is transferred to the PET.

ATN remains asserted beginning at table entry 7 and continuing throughout the entire LOAD transmission. The actual loading, as indicated on line 9, starts at location 400_{16}, 1024 decimal. The command always loads characters beginning at this predetermined address. The LOAD operation, starting at address 400_{16}, loads characters up through the end of user memory from low to high addresses, until the PET's STOP key is depressed.

The LOAD command can be implemented in the immediate or run mode. In the immediate mode of operation, when the statement is:

```
LOAD "", 5
```

without sending a file name, the codes representing the Talk Address Group (TAG 05) and the Secondary Command Group (SCG 17) are sent to the bus immediately. Then the loading begins. Again, all loading commences at location 400_{16} and continues up towards high memory until the PET's STOP key is depressed.

When the command from the PET keyboard is given in the immediate mode, such as:

LOAD "TEST", 5

the CRT displays "SEARCHING FOR TEST." After the first two characters are input from the GPIB, the PET displays "LOADING" on the screen; characters continue loading into memory.

In another example, in an immediate mode statement with no file name, such as:

LOAD "", 5

the CRT will display "SEARCHING." After two characters are input, the screen displays "LOADING"; then characters continue loading into PET memory.

For the timing sequence when data is being loaded into the PET user area of memory, see Figure 5-14.

During execution of the LOAD command, the PET receives data from the GPIB and sends it directly to memory without interpretation. Therefore, unless the tokens and data are presented to the PET in the proper format, examples of which were shown for the SAVE statement in Tables 4-6, 5-6, and 5-7, the LOAD command can cause an error from which the BASIC interpreter cannot recover; the PET may go into a "hung-up" condition.

The LOAD command appears to provide no protection from extraneous inputs from the GPIB.

Sample Bus Transactions — New ROMs

The discussion which follows pertains to PET units that have the "new" ROMs. These PETS are distinguished by pound signs (###) in the title line on power-up. In these machines, the LOAD statement is completely implemented in the IEEE 488 Bus function. The following information and table of bus transactions use a LOAD statement similar to the one discussed earlier, so you can compare the two.

Table 5-10 lists the GPIB transactions for the statement:

10 LOAD "TEST", 8

As the entries in the table illustrate, the LOAD statement can be run in the immediate or run mode.

Table entries 1 and 2 indicate that the PET asserts ATN low (true) while the Listen Address Group (LAG 08) and the Secondary Command Group (SCG 16) transfer their equivalent 1-byte codes to the bus.

The Listen Address Group is specified by the LOAD statement:

10 LOAD "TEST", 8

and the Secondary Command Group, although not specified in the LOAD statement, is $F0_{16}$. It is reserved to indicate that a LOAD statement is in progress.

Table entries 3 through 6 show the program file name "TEST," which the PET transmits to the external device. This is the title of the program to be loaded into the PET's memory.

Table 5-10. Sample bus transactions for the LOAD command (new ROMs)

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
10 LOAD "TEST", 8	1	ATN	LAG 08	28	External listener address
	2	ATN	SCG 16	F0	LOAD secondary address
	3**		T	54	
	4**		E	45	
	5**		S	53	
	6**	EOI	T	54	
	7**	ATN	UNL	3F	Bus unlisten
	8	ATN	TAG 08	48	Talker address
	9	ATN	SCG 00	60	Secondary address
	10			01*	Starting address memory
	11			04*	
	12			09*	Pointer to next statement (in BASIC)
	13			04*	
	14			05*	Line number (05)
	15			00*	
	16			8F*	REM Token — Actual BASIC program being loaded
	17			35*	5 — Actual BASIC program being loaded
	18			35*	5 — Actual BASIC program being loaded
	19			00*	Delimeter
	20			00*	
	21		EOI*	00*	
	22	ATN	UNT	5F	Bus untalk
	23	ATN	LAG 08	28	Device listen address
	24	ATN	SCG 00	60	Secondary address (close file)
	25	ATN	UNL	3F	Bus unlisten

*These items sent by the external talking device

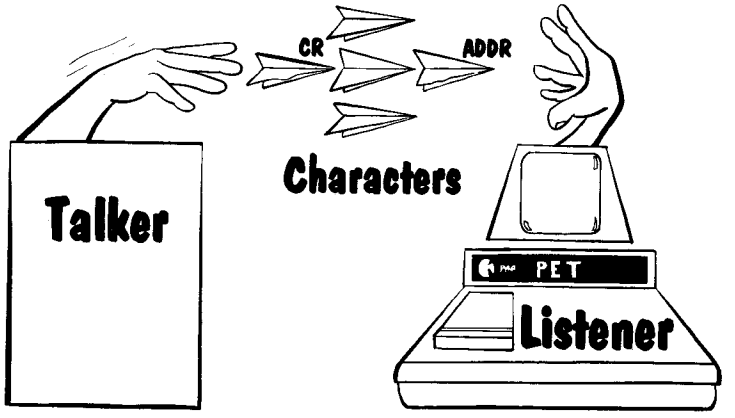
**These characters (entries 3-7) are not sent if there is no file name (i.e., "TEST") used in the LOAD statement.

Next, at table entry 7, the PET again asserts ATN while the universal unlisten command UNL is issued to the GPIB as $5F_{16}$. Having told bus device number 8 to unlisten, the PET next issues the talk address (table entry 8). This is followed by a secondary address (table entry 9). The GPIB transactions initiated by the talker are given in table entries 10 through 21. Table entries 12 through 18 show the program to be loaded into PET memory.

Following the last bus transaction initiated by the talker (table entry 21), the EOI signal is asserted by the talker; the PET lowers the ATN line four times (table entries 22 through 25) to conclude these LOAD statement bus transactions. Refer to the table for the meaning of these last four table entries.

TIMING SIGNALS FOR THE GET STATEMENT

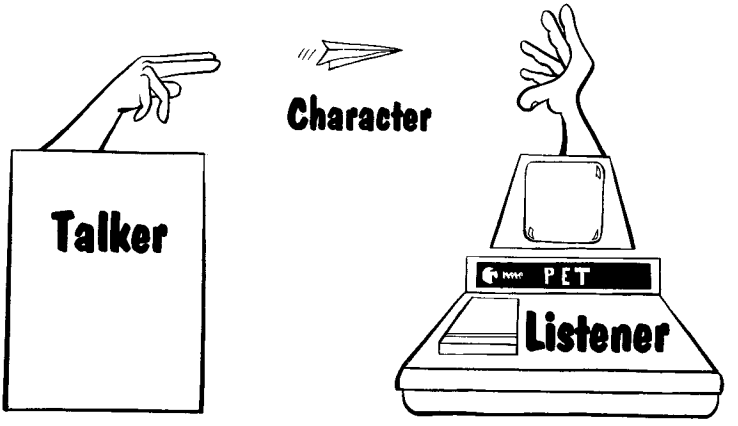
The INPUT statement expects a device to transfer data via the GPIB beginning with the most significant digit. The actual data transfer begins after the primary and secondary (if used) addresses — and continues until the INPUT statement receives a delimiter code.



INPUT Statement

The GET statement, in contrast, reads one character at a time. GET does not wait for a delimiter or termination character.

To illustrate the use of the GET statement, suppose a signal generator sends the character "A" to specify waveform amplitude, followed by numerals to describe peak voltage. If a non-printing control character appeared instead of an "A," you could use the GET statement to receive and assemble the number that follows.



GET Statement

Another useful application for the GET statement would be to examine data character by character. For example, an instrument might send its data least significant digit first. Although in conformance with the IEEE Std 488-1978, such order is not compatible with the PET. You can write a program that reorganizes numeric digits and creates the intended number.

Table 5-11 illustrates two methods of initiating a GET, causing data to be transferred one byte at a time to the PET from a remote talker residing on the GPIB.

In Table 5-11, part a), the statements initialize a file for the GET statement by opening a logical file for the device and specifying the secondary address. This is accomplished by the statement:

OPEN 5,5,2

As shown on line 10 of the program, the GET statement is directed to logical file 5, device 5, and the talker is told to send to the GPIB its alphanumeric data (to be assigned to the string variable A$). Thus, the second statement is:

10 GET# 5,A$

Table 5-11 a), entries 1 through 4 illustrate associated GPIB transactions.

Table entry 1 shows the ATN line asserted. The code 45_{16} on the DIO lines causes characters TAG 05 (Talk Address Group 5) to select the primary address of device number 5.

Table entry 2 illustrates an SCG 02 (Secondary Command Group 2) being transferred to device number 5.

Table entry 3 shows the PET listening for a single byte of data received from the remote talker; the line ⟷ symbolizes this single byte transfer.

Table entry 4 shows ATN again asserted; the value $5F_{16}$ is the universal untalk command UNT. Note that no CR/LF has been issued.

Table 5-11 b) illustrates a similar but shorter bus transaction, achieved by omitting the secondary address. This is accomplished by omitting the 2 from the OPEN statement OPEN 5,5,2.

Table 5-11. Sample bus transactions for the GET statement

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
a) OPEN 5,5,2 *10 GET# 5,A$	1	ATN	TAG 05	45	Primary talker address
	2	ATN	SCG 02	62	
	3		⟷	XX	Remote talker data
	4	ATN	UNT	5F	Bus untalk
b) OPEN 5,5 *10 GET# 5,A$	1	ATN	TAG 05	45	Primary address
	2		⟷	XX	Remote talker area
	3	ATN	UNT	5F	Bus untalk

*Cannot be a direct command, i.e., issued without a line number such as the 10 shown here.

Timing diagrams for the PET GET statement show primary/secondary addressing, the primary address/data burst, and the unaddressing sequence in detail. Changes in logic levels and other pertinent details are identified by a circled number on the diagram.

GET Address Timing — Primary and Secondary Addresses

Before the GET statement can receive data on the data lines, the talker device must be addressed using a primary address or a primary and secondary address. Figure 5-16 illustrates address timing for the GET sequence shown in part a) of Table 5-11, with a secondary address. The timin is similar to that of previous BASIC statements, the differences being in the overall time (extended to 214 μs for the GET statement) and in the address termination sequence preparatory to transmitting the data byte.

A summary of timing events is given below for reference with Figure 5-16. For a detailed description, refer back to the section "OPEN Address Timing — Primary and Secondary Addresses."

Time	Event
1	Typical idle state.
2	PET sets ATN line low (true), indicating it is ready. ATN is low for a nominal 214 μs.
3	External device responds by pulling NDAC low. Device has up to 14μs to respond. Error if no response (see Table 5-2).
4	PET places primary address on DIO lines. Data is allowed 11μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device responds by pulling NRFD low at the beginning of the read.
7	Each devices pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the address data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
11	PET places secondary address on the DIO lines.
12	PET issues the data valid signal by pulling DAV low.
13	PET pulls DAV high to remove data valid indication.
14	PET resets ATN high, terminating the address timing.
15	PET pulls NRFD low (see description of (15) below).
16	Talker places GET data byte on DIO lines (see description of (16) below).

As the GET statement calls for the PET to assume a listener role, the PET assigns itself to this status between the time it pulls DAV high at (13) and the time it brings the ATN line to its inactive high logic level at (14). The PET sets the NRFD line low (true) at time (15), approximately 25 μs after DAV has been reset high. At the time the PET resets the ATN line inactive high, it prepares to accept the GET data byte placed on the bus by the talker at time (16), just after ATN is brought high.

This GET statement with primary and secondary address executes in a nominal 214 μs, as illustrated in Figure 5-16.

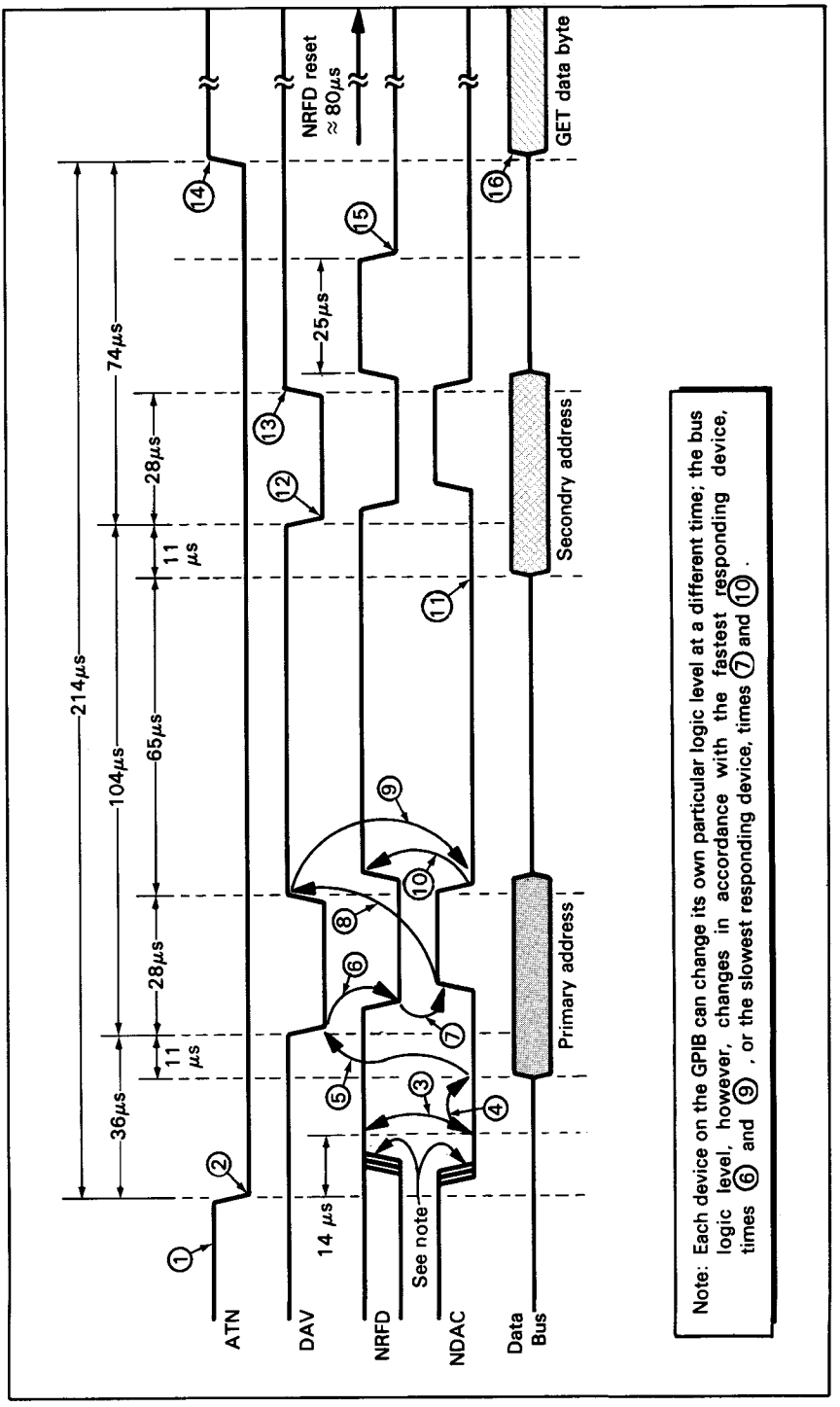


Figure 5-16. Address timing sequence for GET with primary and secondary addresses:

```
OPEN 5,5,2
10 GET# 5,A$
```

GET Address Timing — Primary Address Only

If no secondary address is specified in the OPEN statement (OPEN 5,5 for example), timing signals appear as in Figure 5-17 with only the primary address appearing before the GET data byte is issued by the talker. The circled numbers identify the handshake procedure and the changes in control line logic level, which are identical to those described previously in Figure 5-16 for times ① through ⑩ . As illustrated in Figure 5-17, the total addressing time is cut to 130 μs minimum.

GET Data Burst

In Figure 5-17, prior to the time PET resets the ATN line to inactive high, it prepares itself for the role of a listener at time ⑪ and sets NRFD low (true). This occurs approximately 25 μs after DAV is pulled high at time ⑧ . After the ATN line is reset to inactive high at time ⑫ , the talker places the GET data byte on the eight DIO lines at time ⑬ (the actual time depends on the talker's response time). 80 μs after ATN is released high (inactive), the PET sets the NRFD line high at time ⑭ . The talker then drops its DAV line at time ⑮ , which advises the PET that the information on the data bus is valid and can be received.

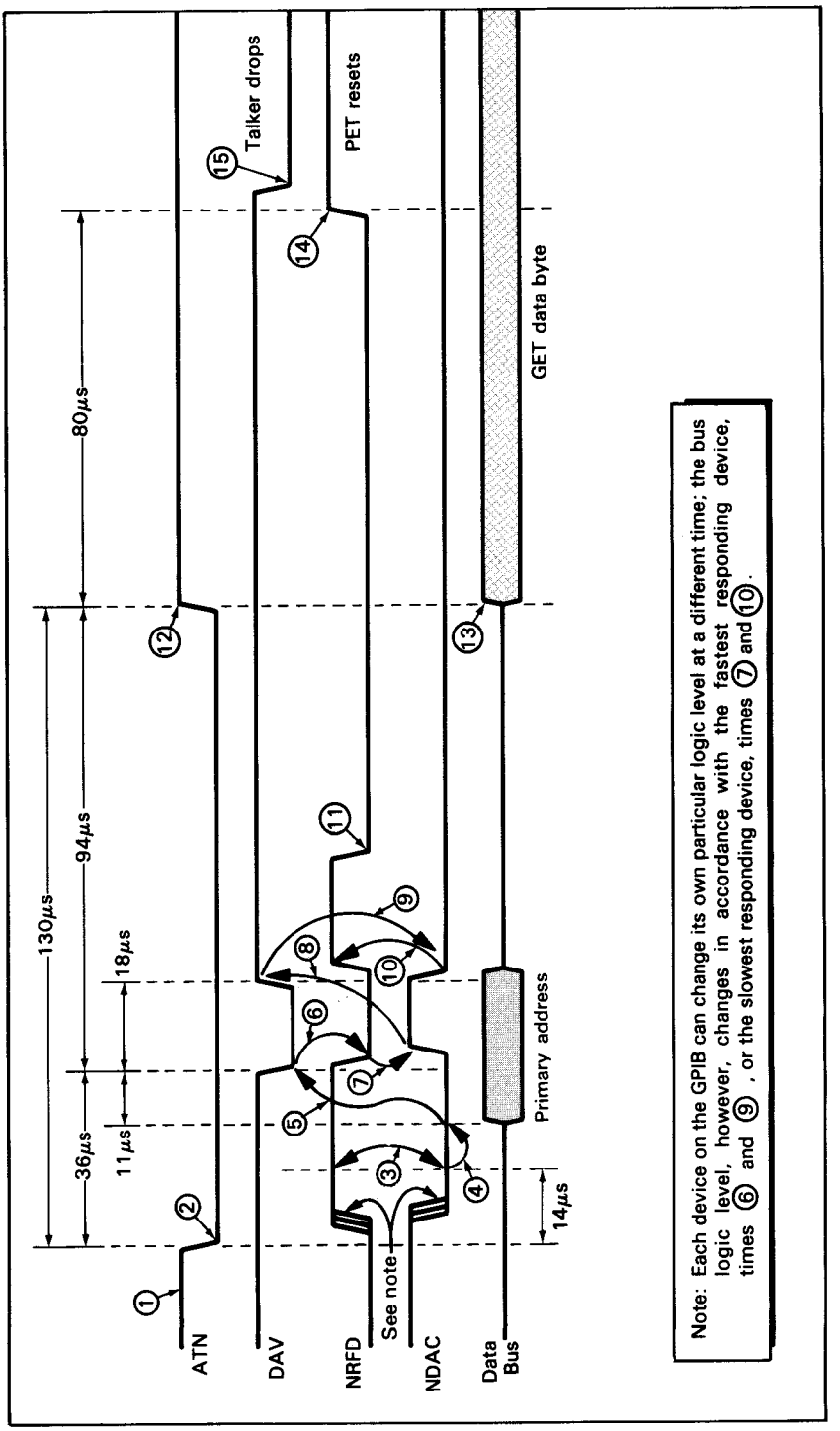


Figure 5-17. Address timing sequence for GET with primary address only:

```
OPEN 5,5
10 GET# 5,A$
```

GET Unaddress Timing

Figure 5-18 illustrates the transfer control line timing on the GPIB during the unaddressing sequence for the GET statement. This sequence is identical to that already described for the INPUT statement.

A summary of timing events is given below for reference with Figure 5-18. For a detailed description, refer back to the section "INPUT Unaddress Timing."

Time	Event
1	Unaddressing occurs at the end of each GET statement with ATN and DAV inactive high (false).
2	PET sets ATN line low (true), initiating the untalk sequence.
3	Remote talker responds by pulling NDAC low and NRFD high. Talker has up to 14μs to respond. Error if no response (see Table 5-2).
4	PET places untalk data byte on DIO lines. Data is allowed 11μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device responds by pulling NRFD low at the beginning of the read.
7	Each device pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
11	PET resets ATN high, terminating the unaddress sequence.

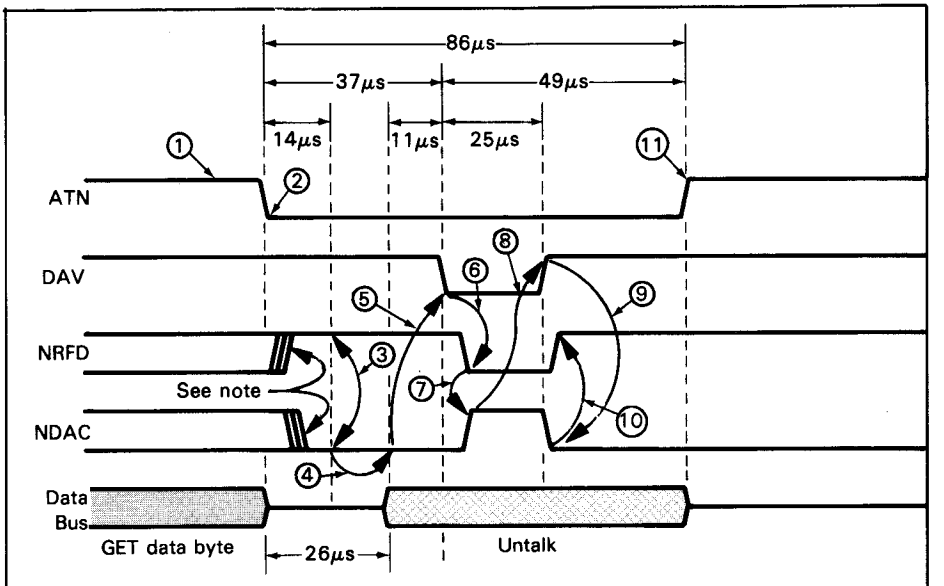


Figure 5-18. Unaddress timing sequence for GET

TIMING SIGNALS FOR THE CLOSE STATEMENT

A CLOSE statement is required to close a logical file opened during execution of an OPEN statement. If a file name is used in the OPEN statement, the CLOSE statement asserts the ATN line, initiates the address handshake procedure with the usual primary and secondary addresses, closes the file, and then resets the ATN line high. This returns the GPIB to its idle state and clears the way for the next communication. Without a file name, there are no GPIB transactions for OPEN or CLOSE.

Table 5-12 a) illustrates the GPIB transactions when a file with a secondary address is closed. First, an OPEN statement with a secondary address must be issued, for example:

```
OPEN 5,5,2, "TEST"
CLOSE 5
```

The OPEN statement opens logical file number 5, addresses device number 5, and calls for a secondary address to be issued. File number 5 is identified by the title "TEST."

At table entry 1, ATN is asserted and Listen Address Group 05 places 25_{16} on the data bus. Next, the secondary address $E2_{16}$, shown at table entry 2, will be issued. As a result of the CLOSE command, the secondary address (the 02 in the SCG column) is processed internally in the PET to produce the command reserved for the CLOSE statement. This is the 02 appearing in the column "PET SCG CLOSE Only" in Table 4-2. From this number, $E2_{16}$ can be verified.

This action closes the file automatically, and no further bus transactions occur after the addresses are issued.

If a CLOSE statement is sent when a file has not previously been opened with a secondary address and a file name, no addresses are sent on the GPIB and there is no bus activity for the CLOSE. See Table 5-12 b).

Table 5-12. Sample bus transactions for the CLOSE statement

Statements	Table Entry	Bus Signal	Characters	Hex	Notes
a) OPEN 5,5,2, "TEST" CLOSE 5	1 2	ATN ATN	LAG 05 SCG 02	25 E2	Primary address Secondary address from OPEN statement With the secondary address and file name in the OPEN statement, there are GPIB transactions.
b) OPEN 5,5 CLOSE 5					Without the secondary address and file name in the OPEN statement, there are no GPIB transactions.

CLOSE Address Timing

The CLOSE statement returns the GPIB to its idle state by issuing a primary and dedicated secondary address.

Figure 5-19 illustrates transfer control line timing for the CLOSE statement. This is the same address timing as shown for previous BASIC statements except that there are no data bytes following the two address bytes.

A summary of timing events is given below for reference with Figure 5-19. For a detailed description, refer back to the section "OPEN Address Timing — Primary and Secondary Addresses."

There is no address byte following the secondary address; neither is there an untalk or unlisten command. The ATN line merely returns to its idle state. This is because of the special way the PET causes a logical file to close.

Time	Event
1	Typical idle state.
2	PET sets ATN line low (true), indicating it is ready. ATN is low for 190μs minimum.
3	External device responds by pulling NDAC low. Device has up to 14μs to respond. Error if no response (see Table 5-2).
4	PET places primary address on DIO lines. Data is allowed 11μs to settle.
5	PET asserts DAV low as data valid indicator.
6	Each device responds by pulling NRFD low at the the beginning of the read.
7	Each device pulls NDAC high (false) when read is complete.
8	PET pulls DAV high to indicate that the address data is no longer valid.
9	Each external device sets NDAC low.
10	Each external device sets NRFD high when it is ready for the next data byte. This ends the handshake sequence.
11	PET places secondary address on the DIO lines.
12	PET issues the data valid signal by pulling DAV low.
13	PET pulls DAV high to remove data valid indication.
14	PET resets ATN high, terminating the address timing.

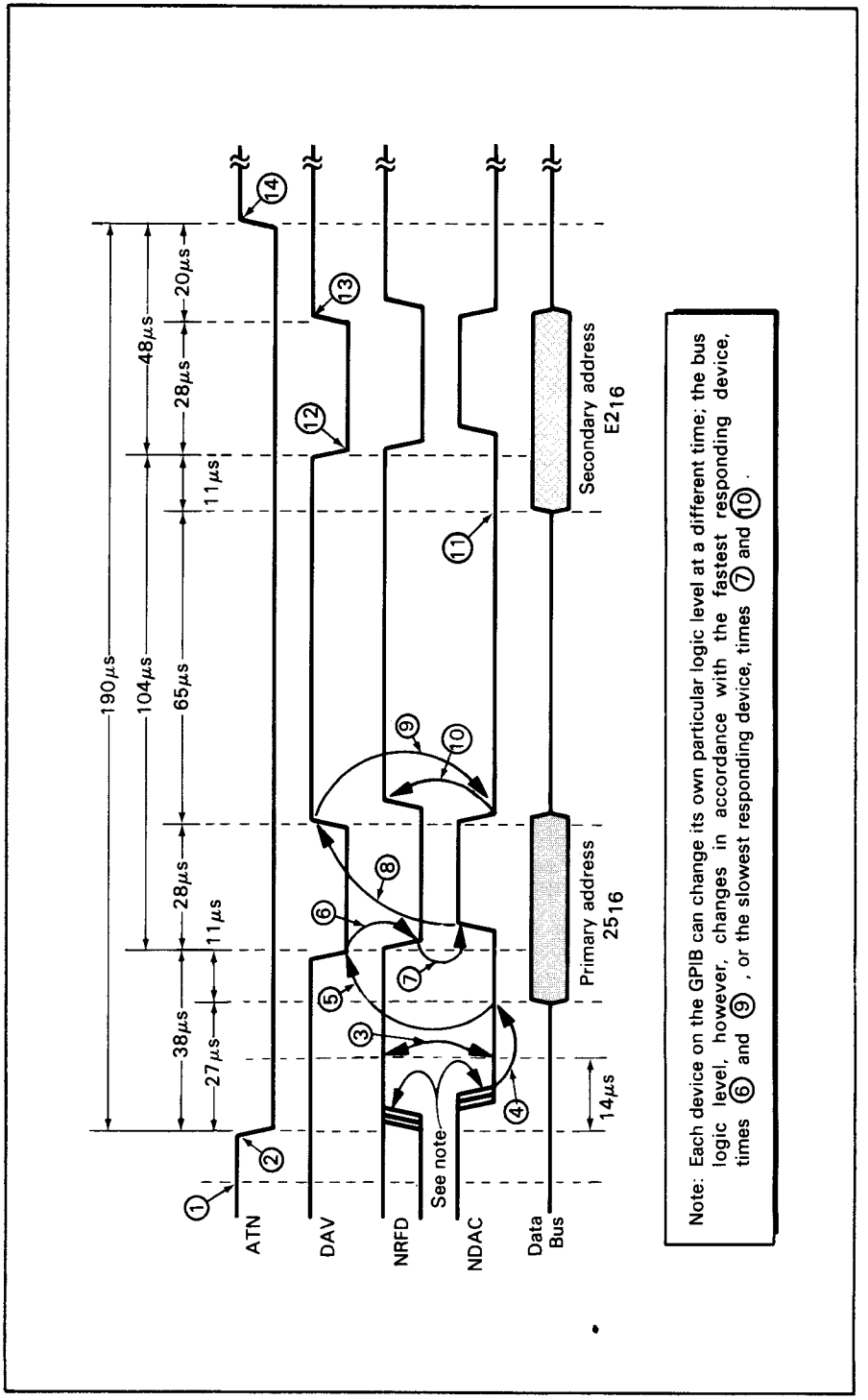


Figure 5-19. Address timing sequence for CLOSE with primary and secondary addresses (OPEN 5,5,2, "TEST":CLOSE 5)

CHAPTER 6

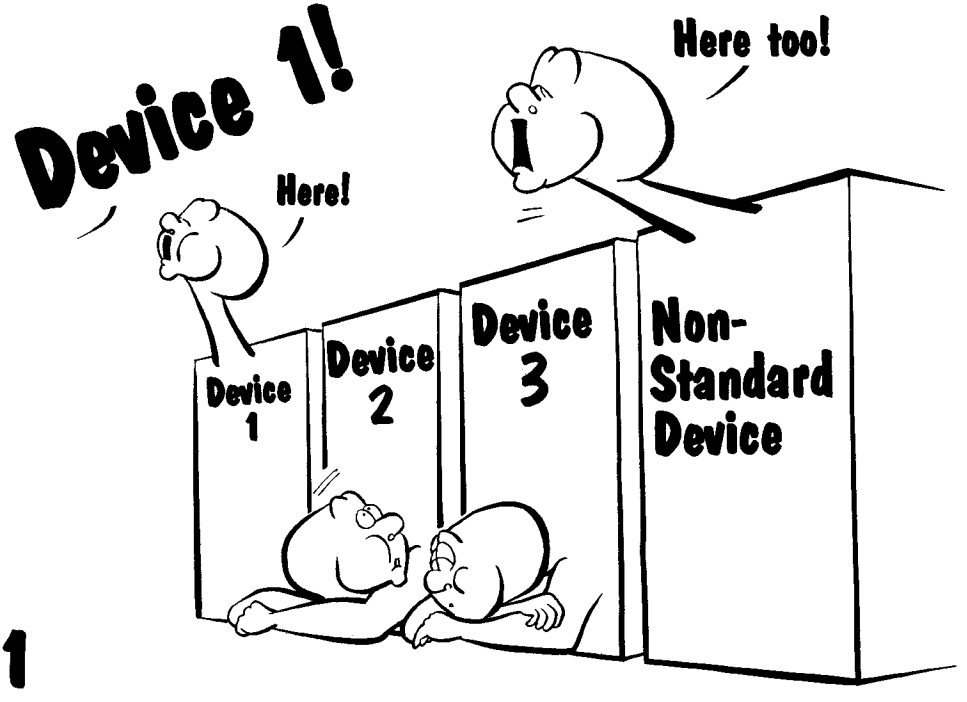
Interfacing the GPIB with a Non-Standard Bus Device

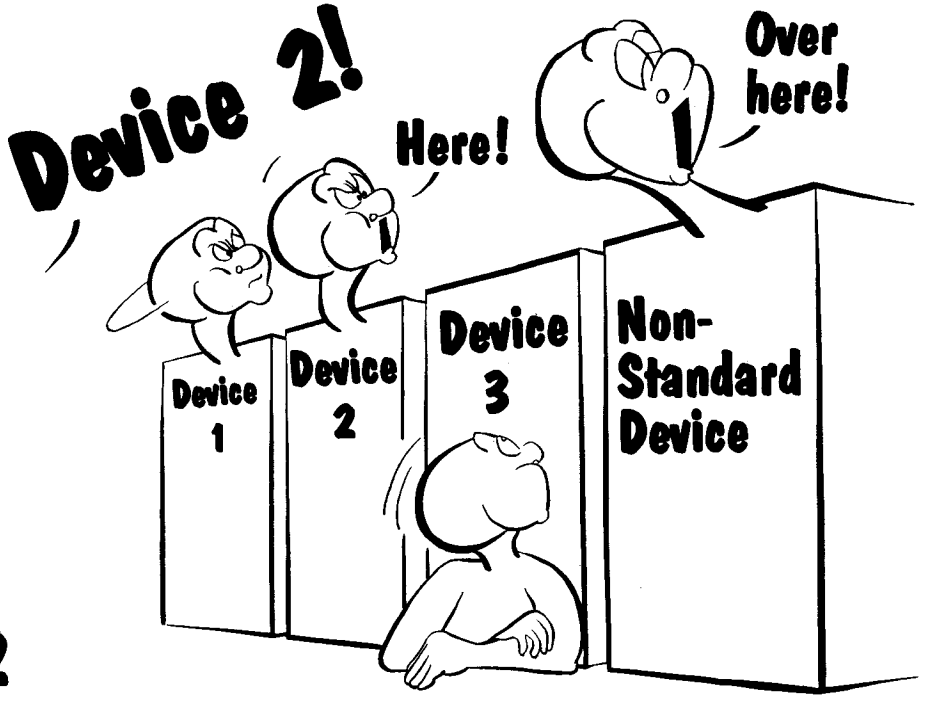
This chapter explains how to interface a non-standard bus device to the PET via the GPIB. Using the techniques described, you can connect one such device to the GPIB, providing all bus-compatible devices are disconnected.

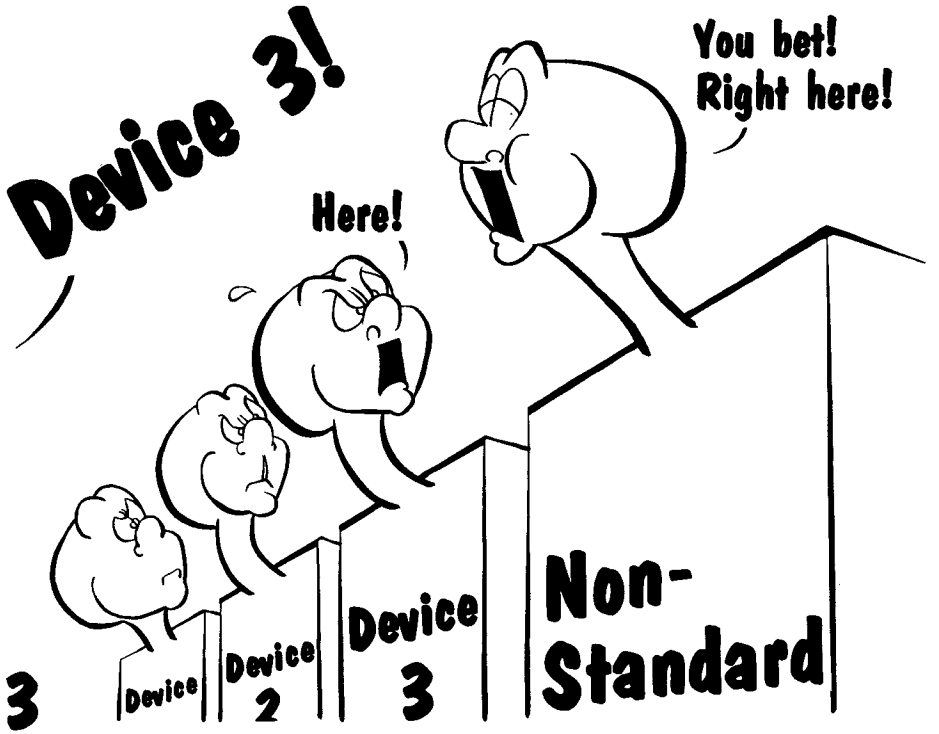
This chapter concludes with a discussion of the IEEE 488 Bus functions, including bus transaction speeds, that are not fully implemented on the PET but that can be simulated by BASIC or assembly language programs. If you must implement some function of the IEEE 488 Bus that is not normally handled by the PET, consider writing programs using the PEEK/POKE techniques described here. If response time or overall speed is a factor, consider writing programs in assembly language, following the one given as an example.

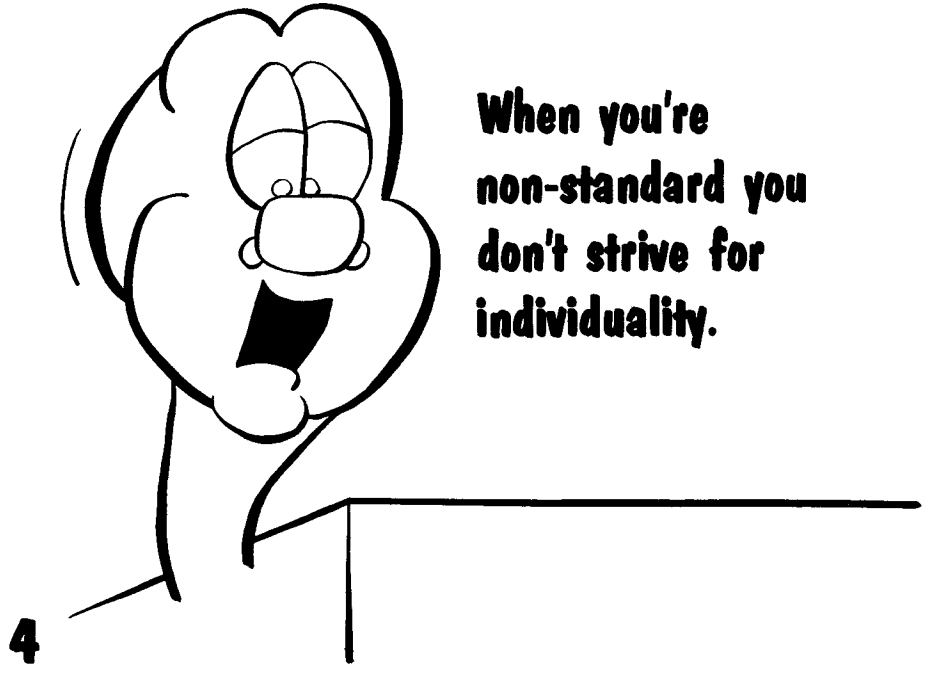
NON-STANDARD DEVICE INTERFACING

Recall that the PET's normal output handshake sequence begins with address information that selects a specific listener to receive the data being output on the GPIB. Such address information is ignored by a non-standard device; if the PET were to address and send data to bus-compatible instruments simultaneously connected to the bus, then the same addresses and data would also be processed by the non-standard device.


Device 1!
Here too!
Here!
Device 1
Device 2
Device 3
Non-Standard Device
1


Device 2!
Here!
Over here!
Device 1
Device 2
Device 3
Non-Standard Device
2


Device 3!
You bet!
Right here!
Here!
Device
Device
2
Device
3
Non-
Standard
3


When you're
non-standard you
don't strive for
individuality.
4

Figures 6-1(a) and 6-1(b) are logic diagrams that progressively develop the circuit shown in Figure 6-1(c). This circuit can be used for interfacing a non-standard device to the GPIB.

In Figure 6-1(a), simple inverters satisfy the handshake protocol and strobe the data from the eight DIO lines. The transfer control lines DAV, NRFD, and NDAC, which control the transmission of data, allow this operation to be performed.

After placing data on the bus, the PET looks at the NRFD line to see if it is high. If NRFD is low, the PET waits. If NRFD is high, the PET outputs DAV low (true); the interface shown in Figure 6-1(a) acts by inverting this signal twice, once in INV-1 and once again in INV-2, to drive NRFD low (true). Simultaneously, the strobe line is driven high; this positive-going pulse indicates that data is valid for the duration of the pulse. The strobe pulse tells the non-standard device that it is ready to receive data.

At the time DAV is driven low (true), INV-3 causes NDAC to go high (false). NDAC high indicates that data has been received. The PET acknowledges NDAC high by pulling DAV high (false). Timing of these occurrences is shown opposite each transfer control line.

Note that two of the inverters, INV-2 and INV-3, are open-collector types, thereby adhering to the IEEE 488 Standard Bus configuration. The circuit in Figure 6-1(a) will not delay any bus communication, and data will proceed at the full bus rate. Every signal propagated down the bus will be transferred to the non-standard device when the device is triggered by the strobe pulse.

Figure 6-1(a) assumes that data is already on the DIO lines; this logic does not take into consideration the addressing sequence. Since addressing information is not to be used in the data mode of the non-standard listening device, the strobe pulse must be suppressed during an addressing sequence. The ATN signal is used for this purpose.

During the addressing sequence ATN is low (true); during the transfer of data, ATN is high (false). If the ATN line and the strobe output from the interface in Figure 6-1(a) are fed to the two inputs of the NAND-1 gate as shown in Figure 6-1(b), the strobe pulse output (now a negative-going pulse) will occur only during the actual data portion of the bus communication. the ATN line, being low (true) during the addressing sequence, can prevent a low (true) strobe pulse from reaching the non-standard device during the addressing sequence. Whereas a positive-going strobe pulse during data transfer time was derived from the circuit in Figure 6-1(a) the circuit in Figure 6-1(b) produces, during data-transfer time, a negative-going strobe pulse, the logic level required for Figure 6-1(c).

Figure 6-1(c) depicts a modified and expanded version of the interface shown in Figure 6-1(b). The circuit in Figure 6-1(c) sets up two conditions: one for fast non-standard devices, the other for slow non-standard devices. The interface first assumes that the device will accept data at the maximum bus signal rate; then, if the device cannot, the circuit sets up conditions so the device can receive data at the slower rate as determined by the device itself.

If the non-standard device (listener) cannot process the data at the full rate

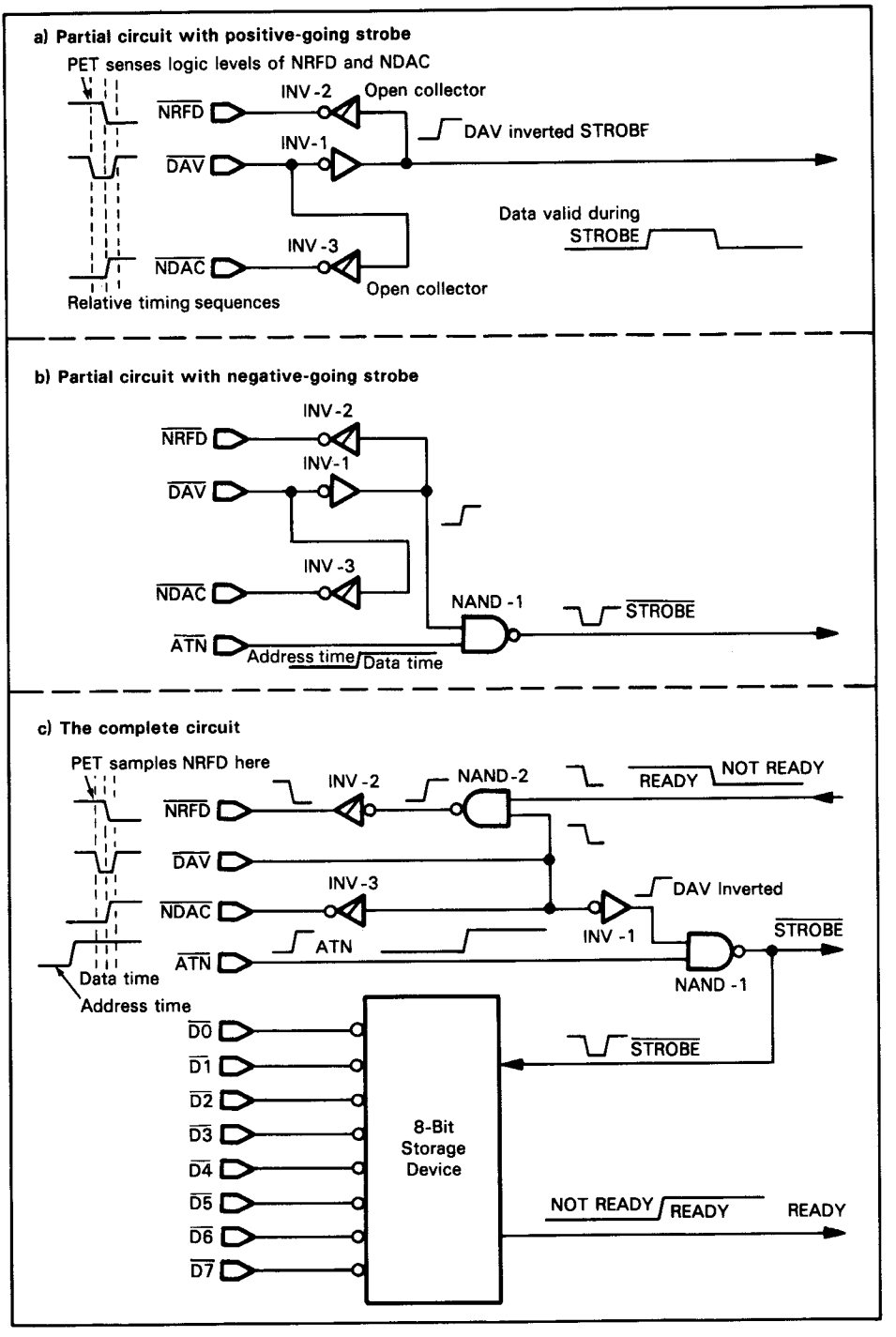


Figure 6-1. Circuits for interfacing a non-standard device to the GPIB

being sent by the PET, then the listener must stop the GPIB handshake sequence with some sort of a READY signal. A simple feature makes it possible to do this, as shown in Figure 6-1(c). At the bottom of the listener device (an 8-bit storage device, for example), a READY signal, generated from the device, is used via NAND-2 gate and INV-2 to assert the NRFD line. This READY signal indicates that the external device is ready for data. By asserting the READY line low, the transfer of data from the bus is inhibited, and the PET waits until the listener is ready to accept the next byte of information.

FAST NON-STANDARD DEVICE

When a non-standard device (for example, the 8-bit storage device) residing on the bus can accept the byte serial data at high rates from the PET, the device should let the READY line be in its high (false) state (the not busy logic level) at the time PET samples the NRFD line to see that it is high (false). With the READY line high and DAV high, the NAND-2 gate has a low output that is inverted by INV-2 to let NRFD go high (false). The PET senses that NRFD is high, then drives DAV low (true) to begin the handshake protocol. Next, three changes take place.

First, the DAV low (true) signal is sent, coincident with READY high (false), to the NAND-2 gate. The high logic level at the output of NAND-2 is inverted by INV-2, which drives the NRFD line low (true), thus allowing the handshake procedure to continue.

Second, the DAV low (true) signal is inverted by INV-1 and is fed to the NAND-1 gate simultaneously with the ATN line, which has previously been pulled to its high logic level. This causes the strobe pulse to be sent low (true) to the non-standard bus device. This strobe pulse indicates that the data byte on the DIO lines is valid during the entire width of the pulse, and the information is strobed into the listener (the 8-bit storage device).

Third, the DAV low (true) signal is inverted by INV-3 to release the NDAC line and allow it to go high. The PET acknowledges by pulling the DAV line high (false). As long as the non-standard device can accept the fast data rate, this sequence will be repeated by the interface and the listener for each data byte being talked down the DIO lines from the PET.

SLOW NON-STANDARD DEVICE

If the non-standard device cannot accept data at the fast rate being sent by the PET (i.e., the next data byte), the device READY signal line should be low (true) at the time the PET examines the NRFD line to see if it is ready for data. With the READY line low (true) and DAV high (false), the NAND-2 gate will have a high output and the NRFD line will be low (true), not ready for data. Thus, the NRFD line will be low (true), and the handshake procedure will be stopped until the READY line is driven high (the not busy state) by the listener.

The above conditions could occur, for example, when a parallel-output printing device takes data at a high rate until its internal buffer is filled. At that

Table 6-1. Suggested parts list for components in Figure 6-1

Figure	Part Identification	Suggested Part
6-1(a)	INV-1	7403
	INV-2	7403
	INV-3	7403
6-1(b)	INV-1	7403
	INV-2	7403
	INV-3	7403
	NAND-1	7400
6-1(c)	INV-1	7400
	INV-2	7403
	INV-3	7403
	NAND-1	7400
	NAND-2	7400

time the READY line would go low (the busy indication), stopping the NRFD signal from being asserted to the GPIB.

Remember that this is not a standard IEEE 488 interface. This particular interface has no provision for looking at addressing information being sent down the bus and, in fact, it will store the data being transferred to or from any device. Also, the circuit has no provision for any of the polling or other operations that normally would be performed.

Before using this type of interface on signals or commands other than the print mode described above, review the particular statement and the data that is to be sent down the DIO lines. The GPIB port of the PET can sink 40 mA of current to ground — plenty of drive for most devices. If a serial output is required, then a UART (Universal Asynchronous Receiver/Transmitter) could be installed as the parallel device that would convert to a serial output.

IEEE 488 FUNCTIONS NOT IMPLEMENTED BY THE PET

There are a number of functions specified in the IEEE Std 488-1978 that are not implemented by the PET computer. Some of these are Service Request (SRQ), Group Execute Trigger (GET), Parallel Poll (PP), Serial Poll (SP), Device Clear (DC), and Device Trigger (DT). While these functions are very useful and a part of complete instrumentation systems, they are not all required for every implementation. However, if one or more functions are needed for a particular application, you can program the PET to generate a simulation. For example, consider the SRQ and GET functions.

In Chapter 7, the section titled "488 Bus Interface Coupler" shows a program that causes a non-IEEE Bus device to send a service request to the PET through a coupler manufactured by the ICS Electronics Corporation. The program is previewed below. It causes the PET to answer a service request, thus imple-

menting a capability not provided by the standard PET BASIC implementation.

The demonstration program is:

```
5 PRINT "□":REM CLEAR SCREEN
10 OPEN 5,6
20 INPUT #5,B$
30 A = PEEK(59426):REM CLEAR PENDING SRQ'S
40 IF PEEK(59427) AND 128 THEN GOSUB 100:REM SRQ TEST
50 PRINT "TESTING"
60 GOTO 40
100 PRINT"□":REM CLEAR SCREEN
105 INPUT #5,I$
110 PRINT #5,I$
120 A = PEEK(59426):REM CLEAR SRQ
130 RETURN
```

Another program in the "488 Bus Interface Coupler" section causes the PET to implement the Group Execute Trigger (GET) command, another function that the PET normally does not have. This is accomplished by using standard PET BASIC statements, with PEEKs and POKEs, causing the PET GPIB hardware to provide the timing required for the GET function. In the example, the 488 coupler is instructed by the PET to give a pulse output which, in turn, causes the non-IEEE 488 device to execute its function. While such methods do implement a non-standard function, there is a response time penalty of 10 to 100 times normal.

CONCERNING SPEED

The overall speed with which the IEEE 488 Bus operates is largely dependent on two factors. One is controller speed when performing the handshake protocol and the data manipulation. The other is the speed with which each instrument on the GPIB is required to respond to controller functions (such as ATN).

Being a slow operating controller, the PET is tolerant of slow responses from remote devices connected to the GPIB; that is, slow as compared to response times specified in the IEEE 488 Standard. Probably the only parts in the IEEE Std 488-1978 specifications that are difficult to implement with the PET microcomputer are some of these "critical" response times.

Two fast response time requirements in the IEEE 488 Standard occur in the talk and listen operating modes during a source-accepter handshake. After the ATN line is asserted, each device residing on the bus is allowed 200 ns to respond. However, this requirement is very tolerantly implemented by most controllers, including the PET; it allows up to 14 μs for a remote device to answer. Thus, in a sense, the PET nullifies the requirement that bus devices respond in 200 ns.

IEEE BUS HANDSHAKE ROUTINE IN MACHINE LANGUAGE

Neither of these fast time response requirements can be implemented in PET BASIC. However, by using assembly language, faster overall data transfer response times can be achieved.

We refer specifically to the article "IEEE Bus Handshake Routine in Machine

Language" by John A. Cooke, reproduced in Appendix E. In this example, the author describes how he overcame a critical character transmission problem between an external voltmeter and the PET. When a speed of only 75 readings per second could be achieved by the standard PET statements written in BASIC, the IEEE 488 handshake procedure was implemented in assembly language, which produced character transfers on the GPIB of up to 5000 bytes per second.

CHAPTER 7

Applications

DIGITAL VOLTMETER, HEWLETT-PACKARD MODEL 3455A

The Hewlett-Packard Model 3455A is a six-and-one-half digit digital voltmeter with auto range and auto calibration. It is illustrated in Figure 7-1. This meter can measure DC volts, AC volts, and resistance using two methods of measurement. The instrument has an IEEE Std 488-1978 bus so that, as purchased, it will interface with the PET.*

To interconnect this device with the PET, simply attach the GPIB cable between the appropriate connectors of the two units. The digital voltmeter can now be programmed by the PET.**

*We will not describe all of the functions of the Model 3455A digital voltmeter in this book; rather, enough of its GPIB interface will be described to provide a basic understanding of its operation on the bus. For further information see the Hewlett-Packard product catalogs, specification guides, and the instrument instruction manual.

**See "Mechanical Features" in Chapter 3 for suggested GPIB cable sources and several manufacturers of connectors that will mate with the J1 connector in the PET.

Photo courtesy of Hewlett-Packard

Figure 7-1. Hewlett-Packard Model 3455A Digital Voltmeter designed for both bench and systems applications

DATA OUTPUT

The format of data output by the HP Model 3455A is compatible with the PET. Transferred down the GPIB to the PET, the data will have the format:

$$\pm 0.000000E \pm 00 \text{ CR/LF}$$

This consists of a + or − sign for the mantissa, followed by the significant digits, then an E signifying the power of ten exponent. This exponent consists of a + or − sign followed by two digits that show the multiplier value to the base ten. The exponent is followed by a CR/LF (carriage return/line feed), which terminates the character being sent down the bus.

ADDRESSING

After the HP Model 3455A digital voltmeter has been connected to the PET via the GPIB, you can immediately begin to address and program the meter so that any of its various functions can be selected under program control.

Figure 7-2 illustrates the seven switches on the rear panel of the digital voltmeter. Each address switch represents a binary number (least significant bit on the right). When the meter is shipped from the factory, the normal positions of these switches are as shown in the figure, forming the number 22 (decimal), or 0010110 (binary). This setting selects the address of the digital voltmeter; the PET will use this address to access the instrument.

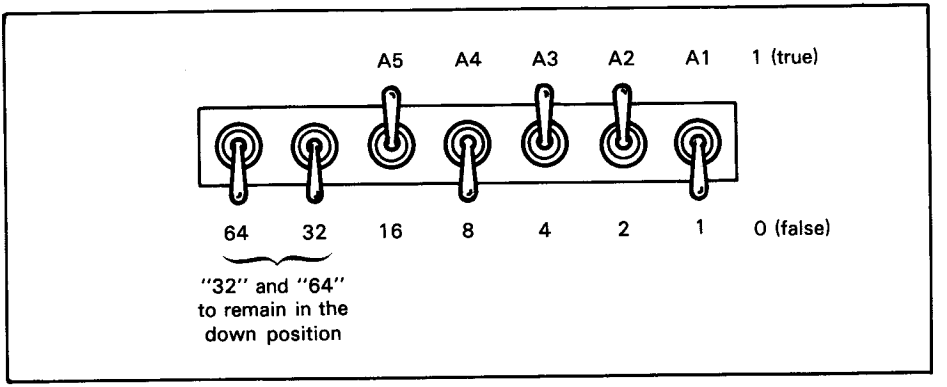
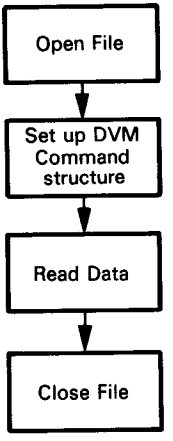


Figure 7-2. A representation of the GPIB switches on the rear panel of the HP 3455A

PROGRAMMING THE HP MODEL 3455A

The block diagram below shows the steps necessary to transfer data down the bus from the PET to any instrument such as the Model 3455A digital voltmeter.

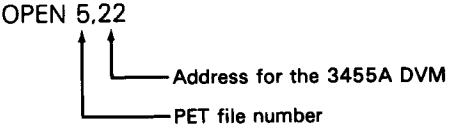


First a file must be opened for PET bookkeeping purposes. Then a command word must be sent to the digital voltmeter (DVM); this selects the desired function. Next the data is read into the PET and, finally, the PET file must be closed. In this programming process, the voltmeter will have been told which function to choose and the range in which to operate. The data in the DVM will have been read into the PET for future reference as a BASIC variable. This data can be processed immediately after reading, after closing the file, or at a later time if the data is saved.

We will now examine the above flow diagram in more detail.

OPEN FILE

We described opening PET files in Chapter 4. Here is an example of the OPEN statement:

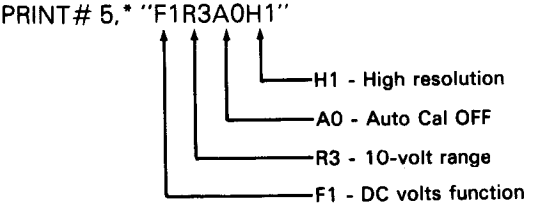


The 5 is a logical file number that is assigned to device number 22 by the OPEN statement and is referenced in subsequent input/output statements.

The 22 designates the address that this particular instrument expects to receive when it is addressed on the GPIB.

COMMAND STRUCTURE

Shown below is an example of the command structure that programs the instrument to the desired function.



PRINT# 5 specifies an output operation and identifies the instrument via the associated file number. The groups of numbers enclosed in double quotation marks describe the functions to be programmed into the instrument. A complete list of these codes is given in Table 7-1.

*When the print statement is used, the pound sign # must be typed after PRINT without entering a space; this avoids a syntax error.

In the example, F1 is the program code that places the digital voltmeter in the DC volts function. The next item is R3, which indicates that the meter is to be switched to the 10-volt range. Next, A0 means that the Auto Cal is to be OFF. Finally, H1 indicates that the meter is to be operated in the high-resolution mode. The closing quotes (") terminate the characters.

Each function of the DVM can be programmed simply by outputting the program code that represents the chosen function. Program codes are listed in the right-hand column of Table 7-1. All are two characters except the single-character code B, for Binary Program. The codes within each group begin with the same letter of the alphabet.

Table 7-1. Program codes for the Model 3455A digital voltmeter

Function Group	Control	Program Code
Function	DC Volts	F1
	AC Volts	F2
	Fast AC Volts	F3
	2 Wire kΩ	F4
	4 Wire kΩ	F5
	Test	F6
Range	0.1	R1
	1	R2
	10	R3
	100	R4
	1 K	R5
	10 K	R6
	AUTO	R7
Trigger	Internal	T1
	External	T2
	Hold/Manual	T3
Math	Scale	M1
	Error	M2
	Off	M3
Enter	Y	EY
	Z	EZ
Store	Y	SY
	Z	SZ
Auto Cal	Off	A0
	On	A1
High Resolution	Off	H0
	On	H1
Data Ready RQS	Off	D0
	On	D1
Binary Program		B

Courtesy of Hewlett-Packard Company

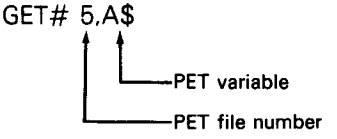
The letter is followed by a number representing range or other control. (Exceptions are Enter and Store, which require a second letter — Y or Z — to be used.) Note that several entries in the column headed "Control" are simply On-Off, such as Auto Cal, High Resolution, Data Ready RQS (request) and Binary Program. Such functions can be programmed to require merely a 0 (Off) or a 1 (On) to be typed on the PET keyboard.

Look carefully at the example command statement PRINT# 5, "F1R3A0H1" and notice that there are no delimiters required between the program codes. Each of these codes (in our example: F1, R3, A0, and H1) will be sent down the bus, one character at a time immediately after the other. The digital voltmeter will pick out the ones of interest, using them to complete its internal programming.

READ DATA

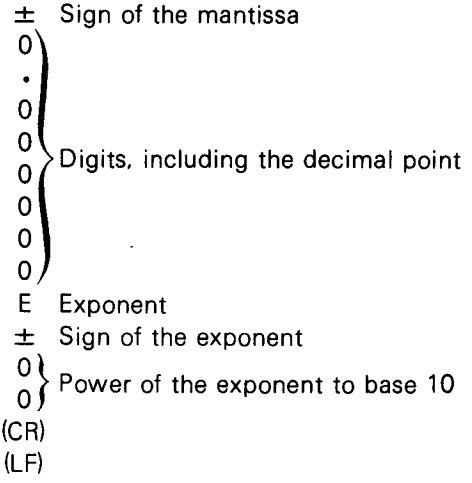
The third block in our simple flow diagram represents data on the bus being read into a PET variable. This can be done using a GET or INPUT statement.

An example of the GET statement is:



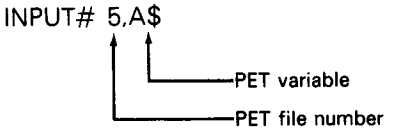
This statement causes data to be input from the device assigned to logical file 5 and subsequently referenced via the string variable name A$.

The GET statement reads one character at a time from the GPIB, without formatting. Therefore, if the GET statement is used to read data from the digital voltmeter, consecutive readouts would yield the data in a sequence as follows:



The GET statement is very useful if you want to examine each data byte being transferred down the bus. However, if you read data in this format, each byte must be assigned a separate variable; your program must put this data into a more usable format if, for instance, you wish to manipulate the data arithmetically. Therefore, from a data-manipulation point of view, you will probably prefer the format yielded by the INPUT statement.

An example of the INPUT statement is:



This statement includes the file number 5, as before, and a variable name A$ via which the data will subsequently be referenced. However, with this statement the PET automatically puts the data into its floating point format. Internally, real numbers are represented in five bytes as normalized floating point numbers with the exponent in excess 128 format.* When printing to the display, the PET adjusts the exponent as necessary to represent the number with one significant digit to the left of the decimal point. For example:

Number input	Number displayed
625E9	6.25E+11

Relatively small numbers (between 0.01 and 1 million) are displayed in non-exponent form. As an example, if the number generated by the digital voltmeter is:

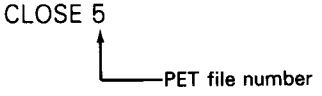
5.000E+02 (Note three zero digits do not appear)

for display purposes the PET will translate the number into 500.

Thus, the INPUT statement is the more practical way to read the data from this instrument.

CLOSE FILE

The CLOSE statement tells the PET to close out file 5 and unaddress the bus device.



*For complete data format specifictions see the PET Personal Computer Guide, Osborne/McGraw-Hill, 1980.

SAMPLE PROGRAM

The flowchart in Figure 7-3 shows a simple verification of the Model 3455A voltmeter operations. This flowchart is implemented with the program listing given in Figure 7-4. This program is not intended to be a comprehensive diagnostic test of the instrument, but rather illustrates the communication method between the PET and the instrument via the GPIB.

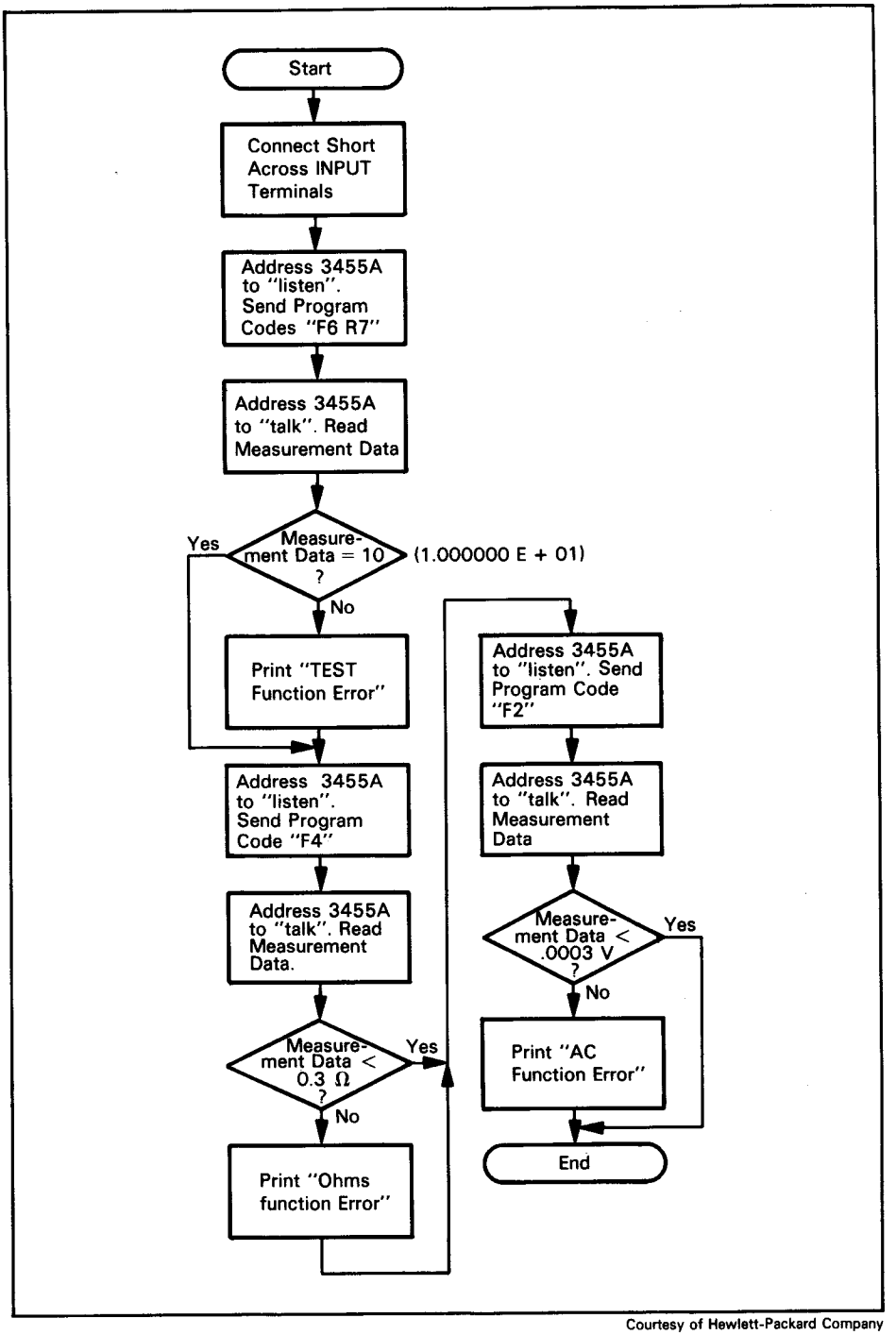
The program first clears the PET display screen; the results of the tests are returned via messages printed on the screen. Lines 5-40, 50-70, and 80-91 perform the three tests delineated in the accompanying flowchart. If the only message is END OF TEST, printed at line 95, then no errors were encountered. The program loops back at line 96 to begin the same test again in a continuous loop (to reenter BASIC, depress the STOP key).

The data collection program starting at line 100 opens a file in the PET and sends the data that programs the instrument. Notice the relatively long delay loop inserted inside the data collection subroutine (line 125). This allows the 3455A DVM ample time to take its readings and perform the diagnostic tests it was programmed to do.

The PRINT statement at line 120 sends the program codes for the tests (refer to Table 7-1):

D0 — DATA READY RQS Off	
T3 — TRIGGER	Hold/Manual
Fx — FUNCTION	Test 1: x=6 Test
	Test 2: x=42 Wire kΩ
	Test 3: x=2 AC Volts
R7 — RANGE	Auto

Note how the same PRINT statement sends different values of the Fx group by presenting the variable second character in continuous format (Rx is also formatted this way, but in this program the value of R is not changed). Lines 130-140 input and display the value of the measurement data. Because of a minor compatibility problem between the PET and the 3455A, you will find that you must put a GET statement (line 145) after the INPUT and PRINT statements. This will read out the last character which is a line feed sent by the 3455A DVM. The reason for this extra programming step is that the PET terminates the input on a carriage return and appears to "hang up" on the last line feed being sent by this instrument and therefore does not complete the entire handshake procedure. The GET statement can be used to handshake out the line feed character.



Courtesy of Hewlett-Packard Company

Figure 7-3. Model 3455A operational verification flowchart

```
2 PRINT "□":REM CLEAR SCREEN
5 F=6
10 R=7
20 GOSUB 100
30 IF A=10 GOTO 50
40 PRINT "TEST FUNCTION ERROR"
50 F=4
55 GOSUB 100
60 IF A<.3 THEN GOTO 80
70 PRINT "OHMS FUNCTION ERROR"
80 F=2
85 GOSUB 100
90 IF A<.0003 GOTO 95
91 PRINT "AC FUNCTION ERROR"
95 PRINT "END OF TEST"
96 GOTO 5
100 OPEN 5,22
120 PRINT# 5,"D0T3F"F;"R"R
125 FOR I=1 TO 2000:NEXT I
130 INPUT# 5,A
140 PRINT "                "A"        "
145 GET# 5,A$
150 CLOSE 5
170 RETURN
```

Figure 7-4. Model 3455A operational verification progam to show sample PET-DVM communication

FREQUENCY COUNTER, SYSTRON-DONNER MODEL 6043A

We chose a Systron-Donner Model 6043A communication system counter (Figure 7-5) to show how the PET can be connected to a frequency counter on the GPIB. This example indicates how simple it is to combine instruments with the PET in a variety of practical applications on the GPIB.

The 6043A is one of the two counters in the 6040A series. The frequency range for each is:

Model 6042A: 20 Hz to 512 MHz
Model 6043A: 20 Hz to 1250 MHz

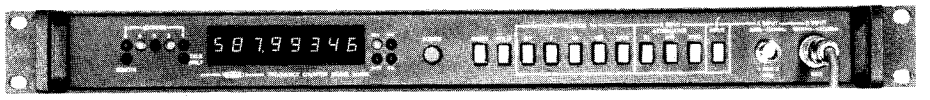
The only interfacing mode for this counter is the IEEE 488 Bus (GPIB).

A citizen's band transmitter was "keyed" on each of 13 channels, and the frequency was recorded by the counter and transmitted to the PET CRT screen. During the experiment, the counter was placed under program control so the PET could select the frequency counter resolution, range, input attenuation, type of sampling, and other parameters.

FRONT PANEL AND ADDRESS SWITCHES

The Model 6043A has several features that make it attractive for use with the GPIB.

Front panel controls (see Figure 7-5) can be set to manually select frequency range and input resolution; however, all except the power switch are also programmable.



Courtesy of Systron-Donner Corporation

Figure 7-5. Systron-Donner Model 6043A Communication Systems Counter

Near the left edge of the front panel are five light-emitting diodes (LEDs). They show the device assignment number on the GPIB that addresses the counter. For the applications described, the address switches were set so that the bit 2 and bit 4 LEDs were lighted; thus, the address selected was 10 ($01010_2 = 10_{10}$).

The address switches, located inside the chassis, are set to the counter's assigned device number by removing the top panel and setting the small switches in the front left corner. Figure 7-6 shows the address switches toggled to identify the counter as a talker residing on the GPIB at address number 10.

Switches 1 through 5 are the address switches that control both the input and output addresses for the counter. To identify the counter as number 10, switches 2 and 4 are ON and 1, 3, and 5 are OFF.

Switch 6, in the ON position, places the counter in the "Local," or TALK-only mode and energizes LED number 6 on the front panel. The front panel switches can be operated to choose frequency range, attenuation and resolution, etc. In the OFF position, this TALK-only switch places the counter in command mode for response to input programming. With the TALK-only switch OFF, the front panel controls of the counter will be locked out as long as the counter is plugged into the GPIB with the PET as the controller.

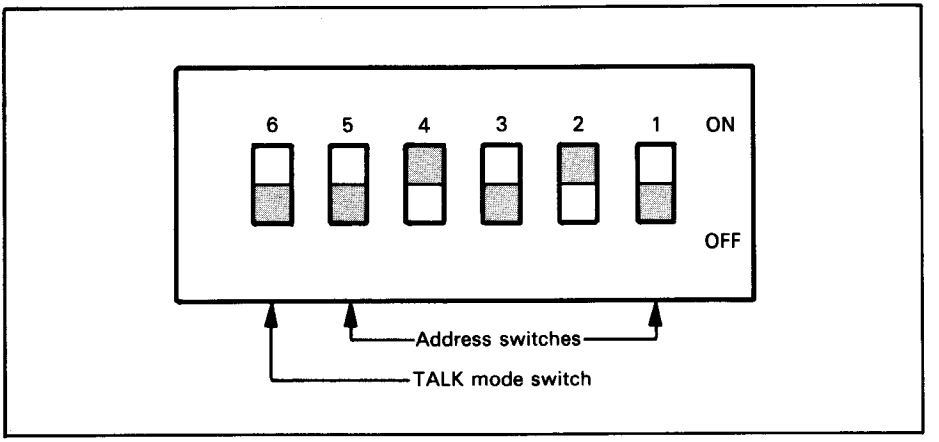


Figure 7-6. A drawing simulating address switches on the Model 6043A counter

PROGRAM CONTROL

The program codes to set the counter, range and input controls are listed in Table 7-2. (Such tables are typical for GPIB-compatible instruments.) The ASCII characters @ through G are assigned to select the resolution of 1 kHz to 0.1 Hz respectively. Characters H through K select the frequency range and the amount of attenuation desired. Of these ASCII characters, H, I, and J select the frequency range 20 Hz to 100 MHz with choices of attenuation ranging from x1 to x100. The character K selects frequency range 100 MHz to 1250 MHz.

ASCII character L is selected to tell the counter to hold its last reading until commanded to recycle or sample. For the counter to continue to take new readings using an internal sampling rate, the ASCII character M is used. The character N tells the digital meter to take a new sample, while O is the ASCII character that resets the LED display counter and causes a new conversion to take place.

These program controls are duplicates of the functions that can be chosen by switches on the right side of the front panel.

Table 7-2. Program codes for the Model 6043A counter

Program Control			
ASCII Character	**Function**		
@	1 kHz	Resolution	Latched (Retains Last Setting in Group)
A	1 kHz	Resolution	
B	100 Hz	Resolution	
C	10 Hz	Resolution	
D	1 Hz	Resolution	
E	0.1 Hz	Resolution	
F	0.1 Hz	Resolution	
*G	0.1 Hz	Resolution	
H	A x 1	A Input Atten	Latched
*I	A x 10	A Input Atten	
J	A x 100	A Input Atten	
K	B	B Input	
*L	Hold		Latched
M	Recycle		
N	Sample		
O	Reset		
* Wake Up State Program Subsets: SH1, AH1, T7, TE0, L4, LE0, SR0, RL2, PP0, DC0, DT1, C0.			

Courtesy of Systron-Donner Corporation

PROGRAMMING THE COUNTER

Program control characters are sent from the PET via the GPIB by using OPEN and PRINT statements such as:

```
10 OPEN 2, 10
20 PRINT# 2, "D,H,L,O"
```

The OPEN statement opens logical file number 2 and designates a primary address of 10 for the frequency counter.

The PRINT statement tells the PET to send to file number 2, which is for device number 10, the characters D, H, L, and O (including commas). As shown in Table 7-2, the D programs the frequency counter for a resolution of 1 Hz. The H tells the counter to use the A input (20 Hz to 100 MHz) with an attenuation of x1. The L directs the counter to hold the last reading until commanded to do differently. The O specifies that the counter reset and start the next conversion.

The two statements OPEN and PRINT take over the controls from the front panel and program the instrument. The counter becomes locked into remote programming due to the PET's continuous assertion of the remote enable (REN) line. With the TALK-only switch ON, and with the GPIB attached, the front panel controls of the counter are not used.

Table 7-3. The output data sequence of the Model 6043A counter

Output Sequence		
Byte	**Function**	**ASCII Character**
1	"OFF"	SP or O
2	"SCALE"	SP or S
3	MSD	0 → 9
4		0 → 9
5		0 → 9
6		0 → 9
7		0 → 9
8		0 → 9
9		0 → 9
10	LSD	0 → 9
11	Exponent Pointer	E
12	Exponent Sign	+ or −
13	Exponent	0 → 3
14	Carriage Return	CR
15	Line Feed	LF

Note: Reading Expressed in Hz.
Example: OS 1 2 3 4 5 6 7 8 E−1 C_R L_F
= 1 2 3 4 5 6 7 8 10^{-1} Hz Off Scale
Input Freq. = 21.2345678 MHz

Courtesy of Systron-Donner Corporation

COUNTER OUTPUT SEQUENCE

Table 7-3 shows the format for the data being returned from the counter to the PET. Bytes 1 and 2 are blanks, or show an off-scale (OS) indication. Bytes 3 through 10 reproduce the frequency readings displayed on the front panel of the counter. Byte 11 is an E, standing for the power-of-ten exponent. Bytes 12 and 13 are the sign and the exponent. Bytes 14 and 15 are the termination characters for carriage return (CR) and line feed (LF).

The PET reads frequencies from the counter when you open the file for device number 10 and read back the data. Sample statements are:

```
10 OPEN 2,10
20 INPUT# 2, A$
```

The data now exists in a string variable, A$.

IMPLEMENTATION

We connected the PET and the Systron-Donner Model 6043A frequency counter using GPIB standard cables and connectors for the interconnection. A citizen's band (CB) transmitter operated nearby, and the broadcast frequency of each of the 13 channels was measured. Figure 7-7 illustrates the equipment setup for taking these measurements.

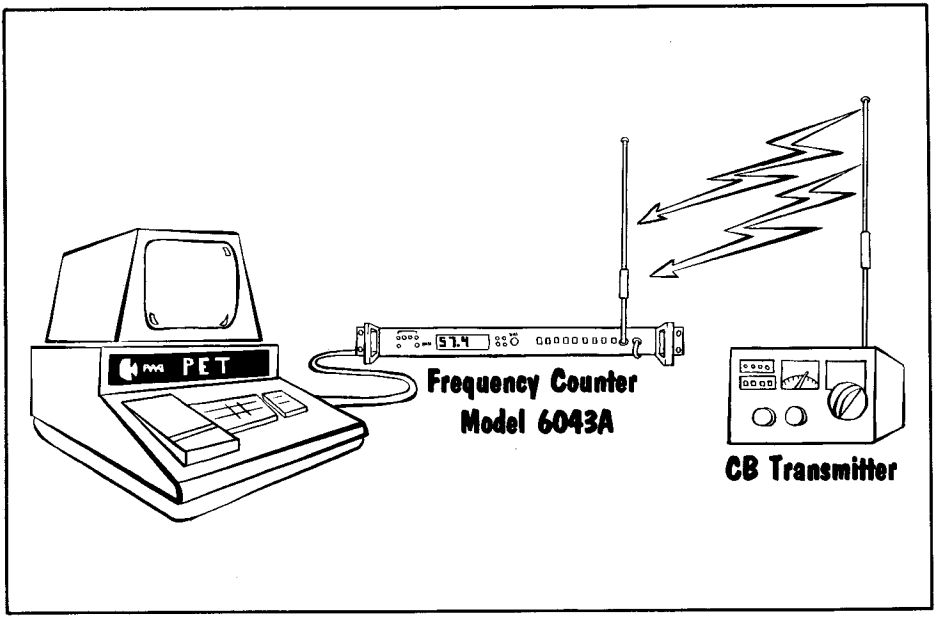


Figure 7-7. Equipment setup to measure frequencies under program control of the PET

The program to obtain the measurement readings is shown in Figure 7-8.

To run this program, enter the number of channels in your CB transmitter in response to the message as shown on line 102; then depress the RETURN key. The number of channels is input to variable A in line 105. Line 110 dimensions a string array A$ to hold the frequency data (string input accepts any characters and will not halt the program for a single spurious character as numeric input would; the strings are converted to floating point format for display on line 180). In response to the messages shown in lines 120-121, turn the CB selector to the channel to be counted and turn the transmitter ON. Depress any character key on the PET to start a conversion.

Line 130 waits until a key is pressed before calling the data collection subroutine beginning at line 1000. At line 1000, an OPEN is issued for the counter to respond to the PRINT statement, line 1010. By means of a small wait loop as shown on line 1015, the PET waits for the frequency conversion, then reads the resulting data into an array variable as indicated on line 1020. The PET then returns to the main program to ask for the next channel to be counted.

```
100 REM THIS IS A TEST TO CERTIFY CB RADIOS
102 PRINT "ENTER THE NUMBER OF CHANNELS IN THIS CB RADIO"
105 INPUT A
110 DIM A$(41)
112 FOR I=1 TO A
119 PRINT "□":REM CLEAR SCREEN
120 PRINT "SET THE CHANNEL SELECTOR TO CHANNEL "I" AND TURN TRANSMITTER ON"
121 PRINT "AND THEN HIT ANY CHARACTER"
130 GET B$:IF B$="" THEN 130
140 GOSUB 1000
150 NEXT I
155 PRINT "□":REM CLEAR SCREEN
170 FOR I=1TO A
180 PRINT "CHANNEL "I" FREQUENCY IS "VAL(A$(I))" HERTZ"
190 NEXT I
200 END
1000 OPEN 2,10
1010 PRINT# 2,"D,H,L,O
1015 FOR J=0 TO 1000:NEXT J
1020 INPUT# 2,A$(I)
1030 CLOSE 2
1040 RETURN
```

Figure 7-8. Frequency Measurement Program

On lines 155 through 190, the program calls for the output frequencies to be printed after all channels have been counted and recorded. A sample display of the output frequency data is shown in Figure 7-9.

```
CHANNEL  1  FREQUENCY IS    26965081 HERTZ

CHANNEL  2  FREQUENCY IS    26975141 HERTZ

CHANNEL  3  FREQUENCY IS    26985226 HERTZ

CHANNEL  4  FREQUENCY IS    27005206 HERTZ

CHANNEL  5  FREQUENCY IS    27015265 HERTZ

CHANNEL  6  FREQUENCY IS    27025319 HERTZ

CHANNEL  7  FREQUENCY IS    27035379 HERTZ

CHANNEL  8  FREQUENCY IS    27055393 HERTZ

CHANNEL  9  FREQUENCY IS    27065266 HERTZ

CHANNEL 10  FREQUENCY IS    27075307 HERTZ

CHANNEL 11  FREQUENCY IS    27085373 HERTZ

CHANNEL 12  FREQUENCY IS    27105385 HERTZ

CHANNEL 13  FREQUENCY IS    27114707 HERTZ
```

Figure 7-9. Sample PET display showing frequency data when running the Frequency Measurement Program

LINE PRINTER, CENTRONICS MODEL P1 MICROPRINTER

Although you find the 25-line PET CRT display very useful for editing and interactive checking of programs, you might encounter difficulty writing a lengthy program without having immediate reference to the beginning and end of the program. Hence the need for a hard-copy printout.

We will describe how the PET can be interfaced with an inexpensive line printer and how such a device can operate correctly although not in strict conformance with the IEEE 488 Bus standard. We chose the Microprinter-P1 (Figure 7-10) manufactured by the Centronics* Data Computer Corporation. This device requires only a very simple interfacing network between it and the PET to make the printer compatible with the PET and the GPIB. Also, we describe a modification that saves printer paper; the modification causes a single — rather than double — line feed (LF) during each carriage return.

Although this interface is straightforward and simple, it is applicable to other Centronics printers in the 700 series. These are impact printers capable of printing multiple-copy forms and other information for mailing.

FEATURES OF THE MODEL P1

Hard-copy printers are often more expensive than the computers themselves. One exception is the electrostatic Microprinter-P1, which uses a 4-inch roll of aluminized paper. This paper requires no toners or ribbons; instead, the conductive aluminized coating is vaporized by a low-voltage discharge from the print head. The copy will not fade or otherwise degrade with age, and it reproduces very well in office copying machines.

The printer measures 13 inches wide by 10.5 inches deep by 4.25 inches high. Under software control, the Microprinter-P1 will print 5, 10, or 20 characters per inch with a standard 96-character character set. The unit has an end-of-paper indicator and a feature that turns off the motor when the printing is complete.

*Centronics is a registered trademark of the Centronics Data Computer Corporation.

Photo courtesy of Centronics Data Computer Corporation

Figure 7-10. The Centronics Microprinter-P1

INTERFACING

The Microprinter-P1 interfaces cost-effectively with the PET. The interface described below can be powered directly from the printer, as the +5 volts is brought out on pin 18 of the printer cable. The interface requires only three low-cost integrated circuits and provides the advantage that when power is removed from the printer, the interface disables and removes the printer electrically from the GPIB. Therefore, although it is a non-standard device in terms of the IEEE 488 standard, the Microprinter-P1 can easily be made to operate while other devices are simultaneously attached to the bus.

This interface follows the criteria for a non-standard interface to the PET/GPIB described in Chapter 6, "Interfacing the GPIB with a Non-Standard Bus Device." The non-standard connections can be seen in Figure 7-11. The data coming from the GPIB must be inverted to supply the positive-true data required by the printer. This is accomplished by the ICs labeled "04," inverters* that change GPIB data to high true. As shown at pins B and D, bits 5 and 7 must be ORed together before being presented to the printer. This is because the PET graphics characters are not in the Microprinter-P1's character set. This ORing changes the PET graphics characters into other characters that will, at least, print.

*One IC, "03" is also used.

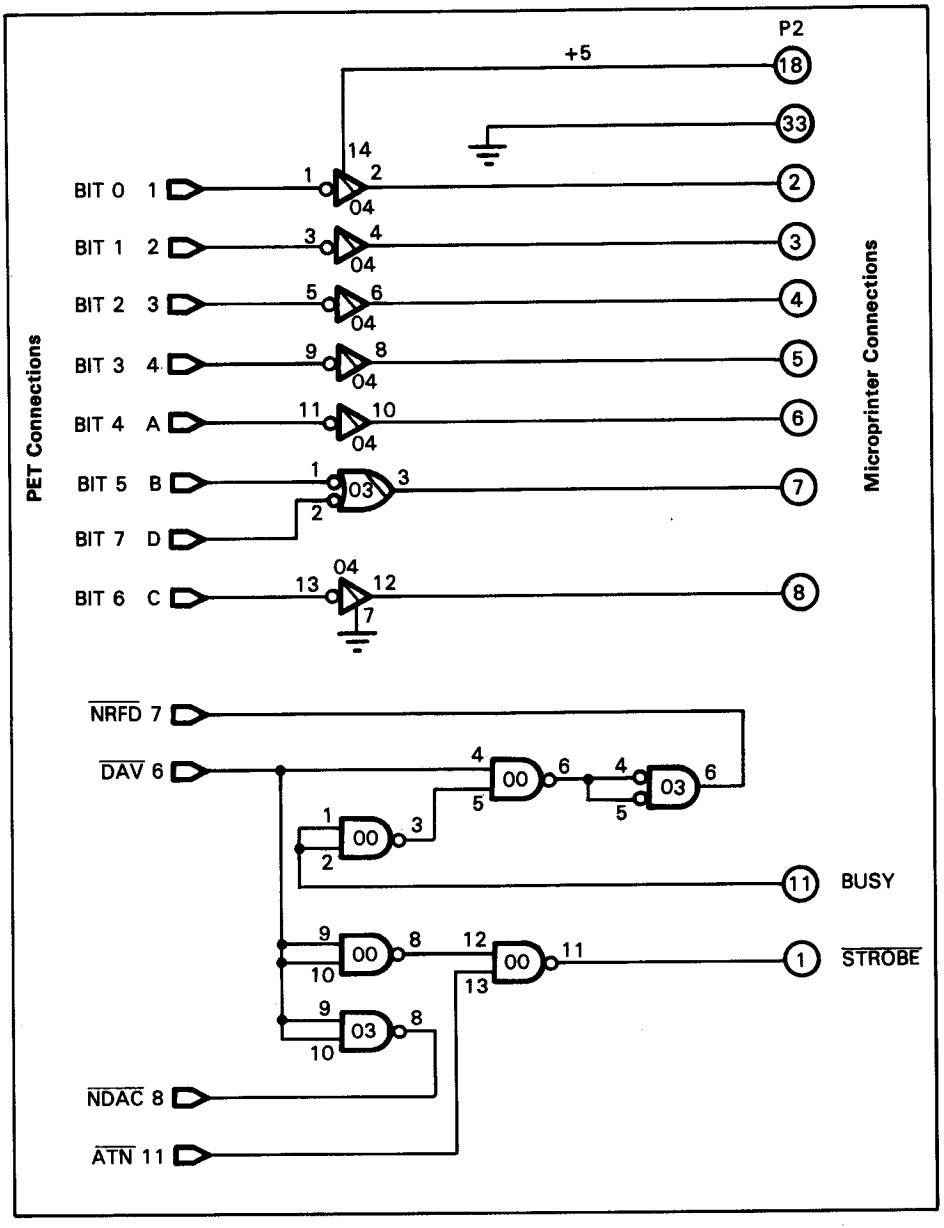


Figure 7-11. Interface necessary between the PET and a Centronics Microprinter-P1

The NRFD, DAV, NDAC, and ATN lines are the same as shown in Chapter 6.

Notice that the BUSY line from the printer goes through an SN7400 inverter. This line supplies the gating signal that stops the PET when the printer's internal line buffer is full. The PET must be prevented from outputting data when the printer is unable to accept more input; this is accomplished by the BUSY signal, which disables the NRFD signal supplied to the PET. Such a Busy time occurs after the printer line buffer has received its 80-character limit (the line of text that is to be printed across the page) or immediately after the PET has sent a carriage return/line feed.

During this Busy time, the mechanism in the printer begins moving the print head across the paper. Each of the seven vertical styli electrodes is energized at the proper time; the charge vaporizes the one-micron aluminum coating from the black background of the paper. The styli, which employ this non-impact discharge technology, produce a variable-pitch 5x8 dot matrix to form the characters. When the print head reaches the right margin of the paper, the paper is moved up one line while the head reacts to a CR/LF and returns to the left margin. Here the head remains at rest until the next line of characters has been fed to the printer line buffer.

At the bottom of Figure 7-11 the STROBE signal, which is generated by the DAV line and gated by the ATN line, tells the printer that data is valid. This signal prevents addresses from reaching the printer while the interface continues to handshake with the PET.

IMPLEMENTATION

To list a program on the printer, simply use the statements:

```
OPEN 5,5
CMD 5
LIST
```

The OPEN statement tells the PET that logical file 5 (first 5) is going to be sent to GPIB device number 5 (second 5). With this non-standard interface, the addressing information is ignored by the printer, and the device number "5" merely tells the PET that data is not going to one of its internal devices.*

The CMD statement redirects all of the PET display output to the GPIB rather than the CRT.

After issuing the CMD statement, you can use any command that normally causes data to be sent to the CRT display; the data will be output to the printer. By issuing a LIST, you send an entire program to the printer and receive a hard-copy output.

*As described in the section "Programming the PET for the IEEE 488 Bus" in Chapter 4, the PET reserves device numbers 0 through 3 for its internal operation; therefore, all devices residing on the GPIB must be assigned numbers of 4 or greater.

INTERFACE FABRICATION

To implement this interface, fabricate a flat ribbon cable assembly having on one end a mating connector for P2 located in the printer; terminate the other end in a small printed circuit board that will mate with J1 located in the PET. Such an assembly, illustrated in Figure 7-12, does not provide for other instruments to be interconnected with the system. The parts for this assembly (and those in Figure 7-11) are listed below.

Parts List

Designation	Part Number
00	SN7400
03	SN7403
04	SN7404
J1	251-12-90-160, Cinch (See also "Mechanical Features" in Chapter 3.)
P2	57-40360, Amphenol or equivalent

If you want to attach other instruments to the GPIB simultaneously with the Centronics printer, you can do so by fabricating the interface shown in Figure 7-13. This is an identical interface, with one exception: the Cinch 251-12-90-160 (or equivalent) connector on the PC card is replaced with an IEEE 488 standard connector such as AMP 552305-1. This latter connector mates with the Commodore "PET 488 Cable Assembly" and provides for other devices to be connected to the bus.

PRINTER MODIFICATION

An interesting feature of the Microprinter-P1 is the carriage return (CR) character. It will cause a CR/LF pair to be executed, and the paper will be advanced one line. Since the GPIB transactions always end with a CR/LF pair, however, an extra blank line will always occur between each line on the listing.

To conserve paper and correct this condition, you can either:

a) Remove the top cover from the Microprinter-P1 as though preparing to replace a fuse.* Cut pin #1 of IC2B as shown in Figure 7-14, or

b) Solder a short jumper wire from pin 9 to pin 7 of IC2D, as shown in Figure 7-14.

Note: Only one of the above modifications needs to be made.

*See the Microprinter-P1 Instruction Manual.

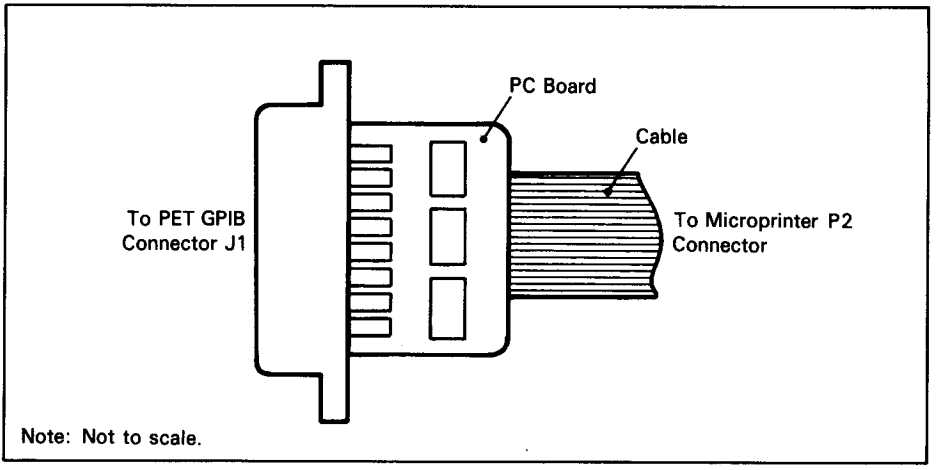


Figure 7-12. Cable assembly to interface the PET and the Microprinter-P1. This PC card/cable connector arrangement does not allow IEEE 488 standard devices to be interconnected with the system.

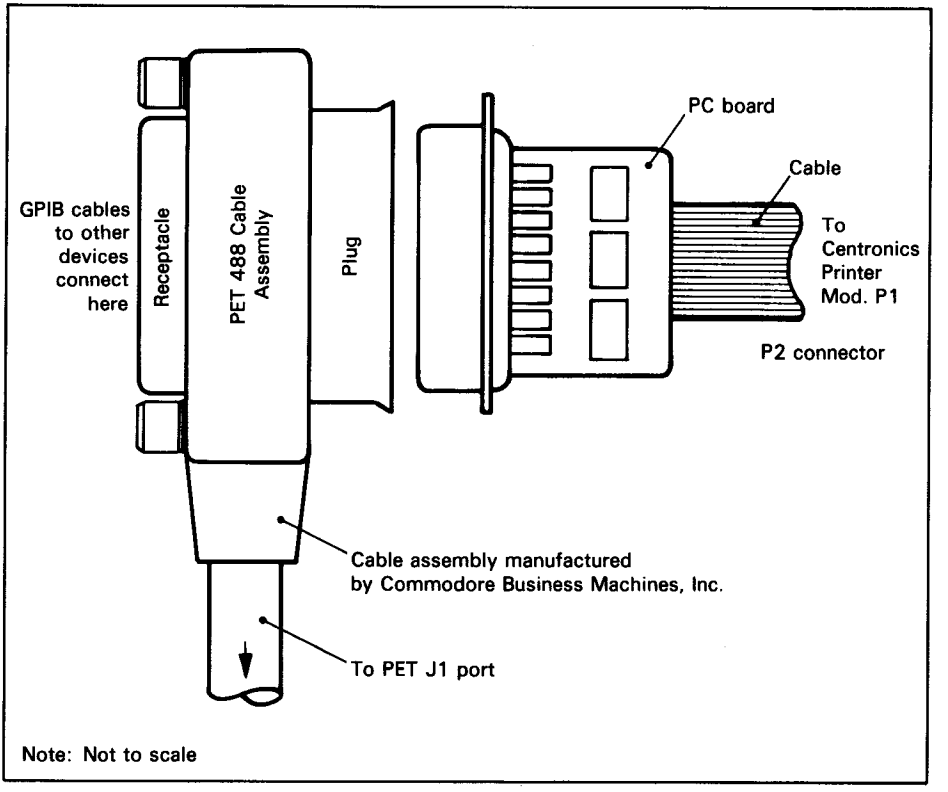


Figure 7-13. Interfacing method that will connect the Centronics printer with the PET and also allow standard IEEE 488 devices on the bus

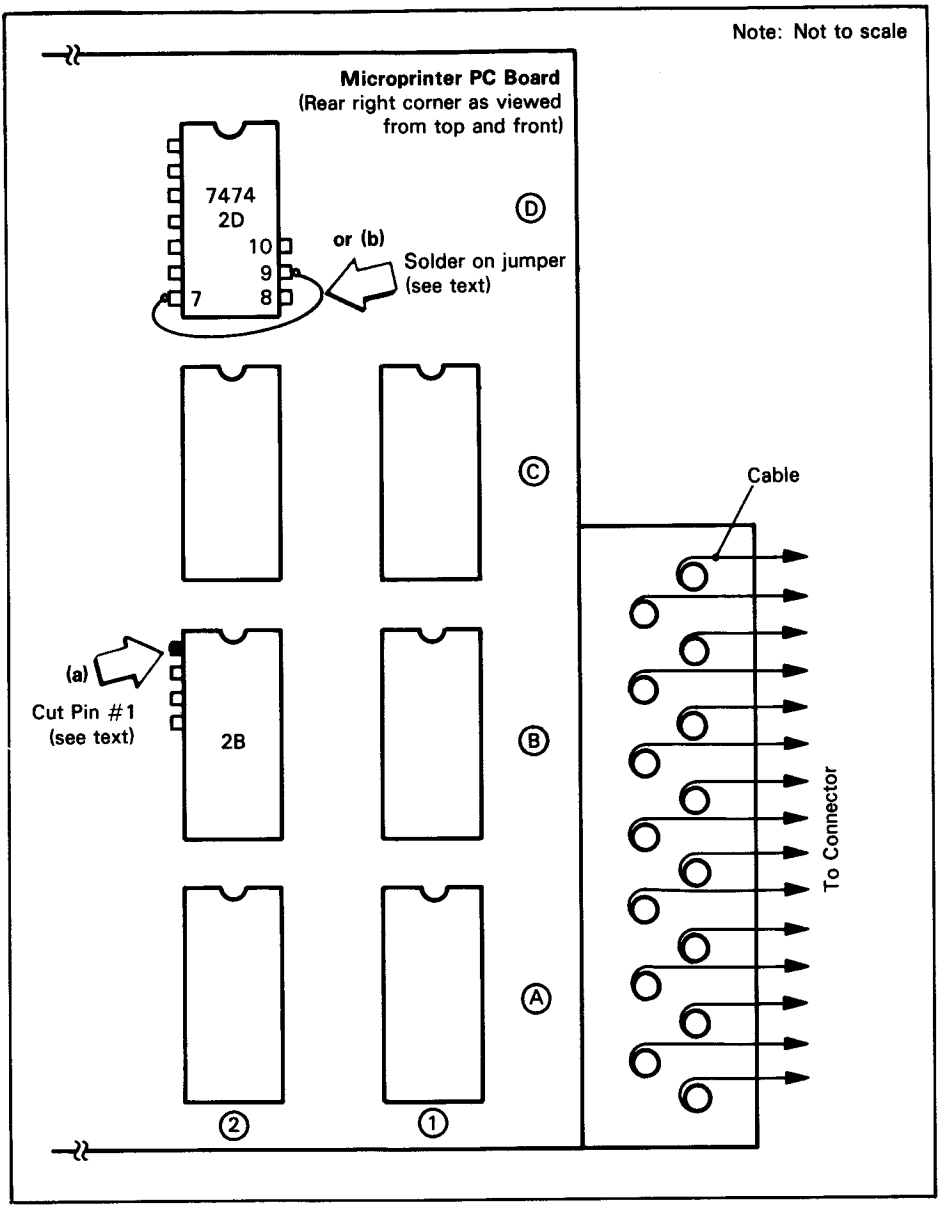


Figure 7-14. Partial top inside view of the right rear corner of the Microprinter-P1, showing the IC locations. (See text for modification directions)

LOGIC ANALYZER, TEKTRONIX MODEL 7D01 AND DISPLAY FORMATTER, TEKTRONIX MODEL DF2

When a controller and several programmable digital instruments are interconnected with the GPIB and certain functions do not work, the user may have an extremely difficult time determining the cause of the problems. These problems may exist because of incompatibilities between instruments due to signal timing constraints or, perhaps, to a signal line being held at a permanent logic level, thereby causing errors in data transfer. Changes in individual logic levels can be displayed and studied with an analog or DC oscilloscope, but the problem of locating an error is nearly insurmountable because every transmission of a control signal down the GPIB is time-critical, often sequenced and dependent on a combination of other signal occurrences.

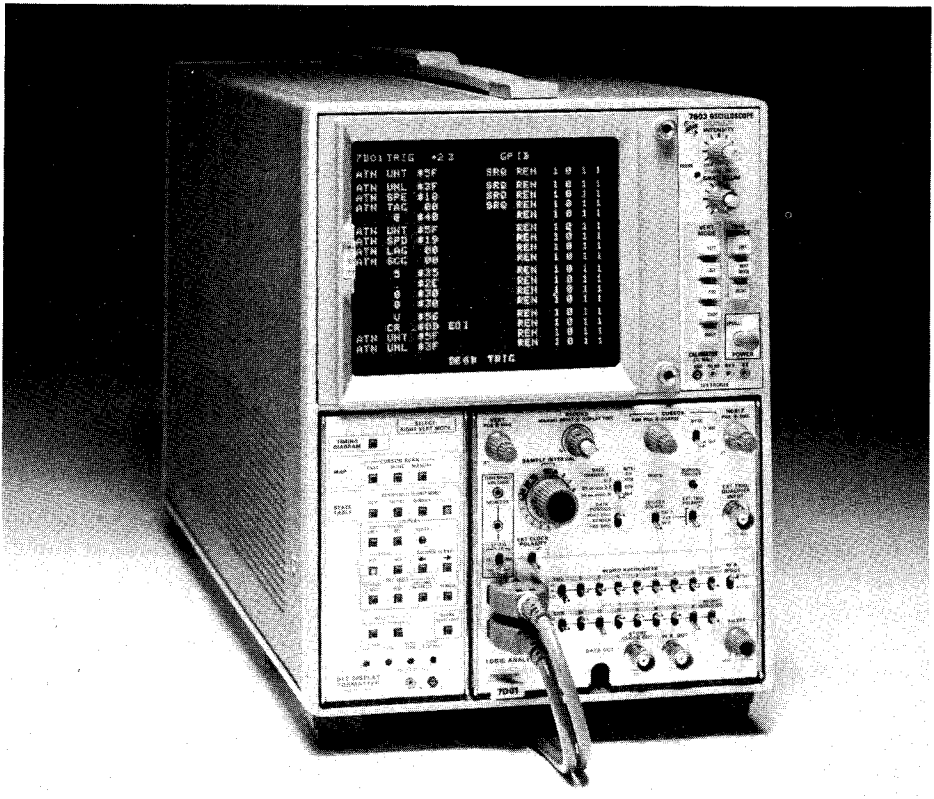


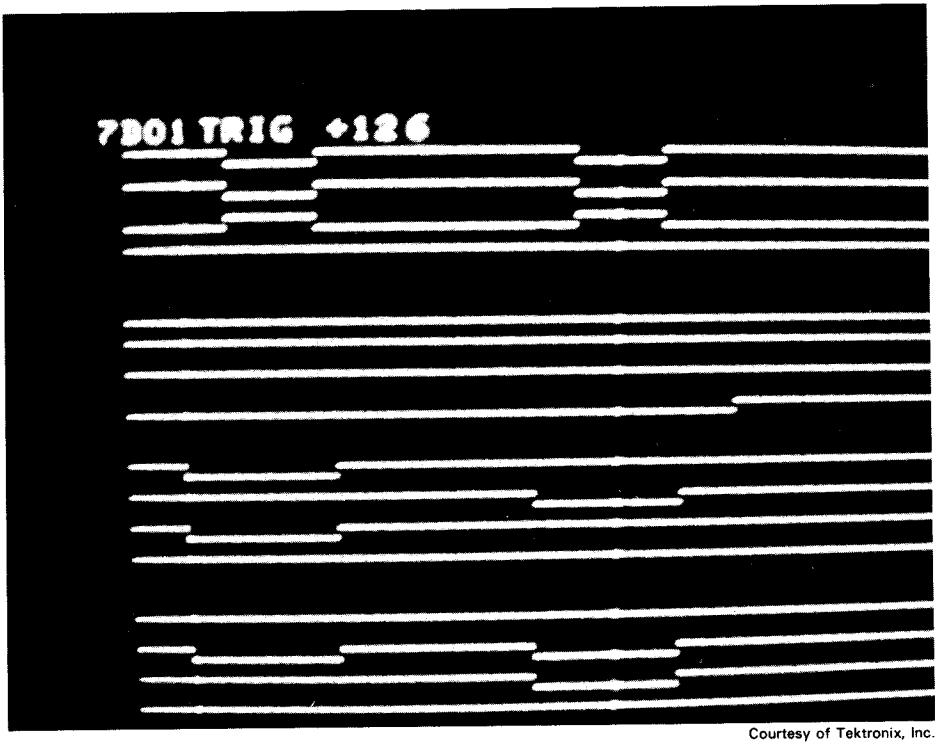
Photo courtesy of Tektronix, Inc.

Figure 7-15. Tektronix Model 7603 Oscilloscope with the 7D01 Logic Analyzer and DF2 Display Formatter. The display shown is that of the DF2 Display Formatter.

In this section we describe an instrument that displays each signal on the GPIB simultaneously, thus making signal analysis and error detection easier.

The Model 7D01 logic analyzer (lower right corner of the instrument in Figure 7-15) manufactured by Tektronix, Inc., is a test instrument that can display the logic states of the 16 lines in the IEEE 488 Bus. This analyzer is a plug-in module that uses a Tektronix 7000 series oscilloscope chassis to monitor digital logic signals.

One possible use of such an analyzer would be to examine the binary values of a set of lines coming from a microprocessor. This would be done by noting the high or low logic level on each line within a set of lines and, by monitoring successive bytes of identical information, determining if the logic level on each line was correct. Another use for such an analyzer would be to examine logic levels of ATN and the handshake lines DAV, NRFD, and NDAC, or the levels of the data transmission lines in the IEEE 488 Bus. A sample logic analyzer display is shown in Figure 7-16. The display is similar to the timing diagrams presented in this book. Most logic analyzers on the market today could be attached to the 16 lines of the GPIB to show these handshake timing and data-burst sequences.



Courtesy of Tektronix, Inc.

Figure 7-16. Model 7D01 Logic Analyzer display. A logic analyzer displays signal levels of many lines simultaneously; however, such signals are difficult to interpret. Compare this display with the one in Figure 7-15.

While such analyzers have decided advantages over DC oscilloscopes for taking timing and logic-level measurements, the 7D01 analyzer has a companion plug-in module that makes such an analysis of GPIB signals even easier. This is the Model DF2 display formatter, shown in the lower left corner of the instrument in Figure 7-15. Rather than displaying the data as an array of timed pulses indicating logic levels, the DF2 module causes the data to be displayed in a tabular format. An example is shown on the CRT screen in Figure 7-15.

The combined capability of the logic analyzer and display formatter is extremely helpful for GPIB signal analysis because of the variety of complex operations transacted by the bus. While the eight DIO lines transmit data reasonably straightforwardly in that they carry information in bit parallel, byte serial format, the control lines have particularly rigid constraints imposed on them by the IEEE Standard and can therefore be extemely difficult to analyze.

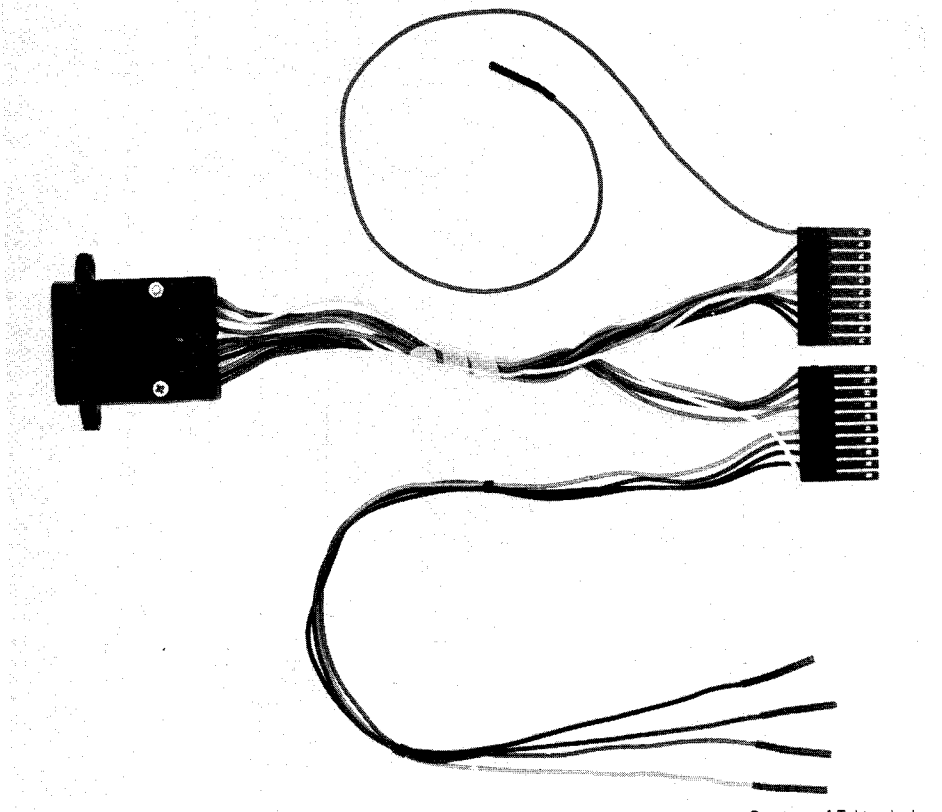
Portions of the technical material in this section have been paraphrased or taken directly from the Tektronix instruction manual "Operating Instructions for GPIB-DF2."* This is done with the permission of Tektronix, Inc.

CONNECTING THE DISPLAY FORMATTER TO THE GPIB

The DF2 display formatter is easily connected to the GPIB by means of an adapter, shown in Figure 7-17, manufactured by Tektronix, Inc. This adapter connects a standard GPIB connector to two P6451 data acquisition probes. The two probes provide 16 channels of data input, the DAV clock input, and one qualifier input to the Model 7D01 logic analyzer.

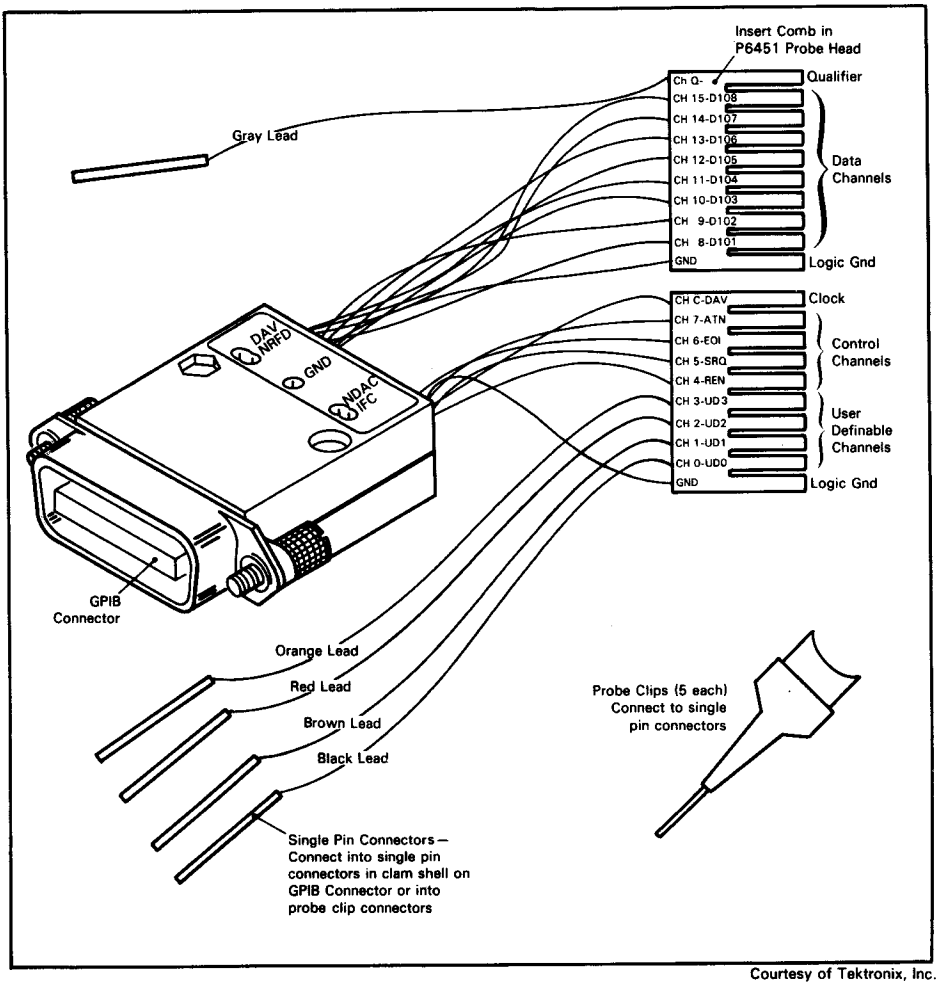
The adapter consists of a GPIB connector and interconnecting wires terminating in two "combs." Each comb is inserted into the head of a P6451 probe. Figure 7-18 identifies the individual teeth on the adapter and cross-references them to the GPIB lines and the channels leading to the "clamshell" housing that surrounds the GPIB connector. In this connector there are four holes, with recessed single-pin connectors that attach to the three handshake lines (DAV, NRFD, and NDAC) and the Interface Clear line (IFC). Plugging into the recessed connectors are four user-definable, single-pin connectors (UD0 through UD3) that connect to channels 0 through 3 in the channel 0-7 comb. These leads may be connected in any desired order; however, it is recommended, when observing the handshake cycle, that you connect DAV to UD0, NRFD to UD1, and NDAC to UD2. If you want to look at other GPIB signals, you can connect probe clips to the single-pin connectors of UD0 through UD3. The probe clips have a retractable hook for attaching to a terminal, wire, or test point. Figure 7-19 shows the IEEE-488 (GPIB) connector, the GPIB adapter, and the P6451 probe heads connected together.

*The manual and documentation supplied with the Model DF2 is an excellent tutorial for the novice user of the IEEE Std 488-1978 Bus.



Courtesy of Tektronix, Inc.

Figure 7-17. Tektronix P.N. 103-0209-00 Adapter showing IEEE 488 standard connector (left) and the input combs (right) that input to Logic Analyzer P6451 probe heads



Courtesy of Tektronix, Inc.

Figure 7-18. The GPIB connector and input combs with interconnecting wires identified.

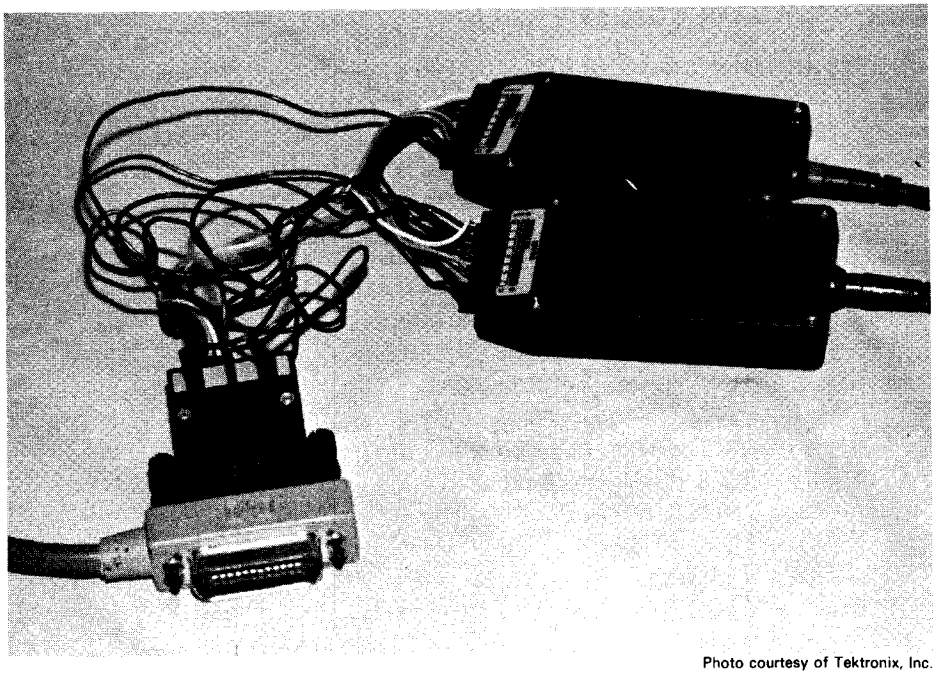


Photo courtesy of Tektronix, Inc.

Figure 7-19. A close-up view of the GPIB connector interconnected with the Tektronix Logic Analyzer P6451 probe heads

INSTALLATION OF THE GPIB ADAPTER

Install the GPIB adapter as follows:

1. Insert the two combs of the GPIB adapter into the two P6451 probe heads. (The combs are marked for channels 0-7 and channels 8-15; follow the labeling to make sure each is inserted into the correct probe head.)
2. Insert the two P6451 connector probes into the input connectors on the Model 7D01 logic analyzer. (Top connector — channels 0-7; bottom connector — channels 8-15.)
3. Connect the four user-definable lines as desired.
4. Connect the GPIB adapter to your GPIB connector.

The fully installed system, connected to the PET, is shown in Figure 7-20.

SET-UP FOR DATA ANALYSIS DISPLAY

Adjust the logic analyzer for the desired mode of triggered operation by setting the word recognizer switches on the front panel switch array (shown in Figures 7-15 and 7-20) to any of a number of different combinations.* Besides setting these switches to represent various ASCII characters, you can set the following:**

- TRIGGER ON ANY ATN MESSAGE
- TRIGGER ON ANY ATN MESSAGE GROUP, for example:
  - ACG (Address Command Group)
  - UCG (Universal Command Group)
  - LAG (Listen Address Group)
  - TAG (Talk Address Group)
  - SCG (Secondary Command Group)
  - SPE (Serial Poll Enable)
  - UNL (Unlisten)
  - UNT (Untalk)
  - LAG 25
  - TAG 14
  - SCG 22
  - CR (Carriage Return)

Trigger on any ATN Message

For a demonstration of how the word recognizer (WR) switches can be adjusted to analyze signals on the GPIB when using the PET as a controller, set the switch array to trigger on any ATN message. Figure 7-21 shows the switch array set to do this. After the reset button is depressed, the 7D01 logic analyzer will trigger on the first ATN message received.

Whenever the ATN line is asserted at the beginning of an addressing sequence, the logic analyzer is triggered. It begins to sample data from the GPIB and continues until its buffers are full; then it displays the data on the analyzer CRT.

*Many of these combinations are listed in the Tektronix "Operating Instructions for GPIB-DF2."

**According to the Tektronix manual.

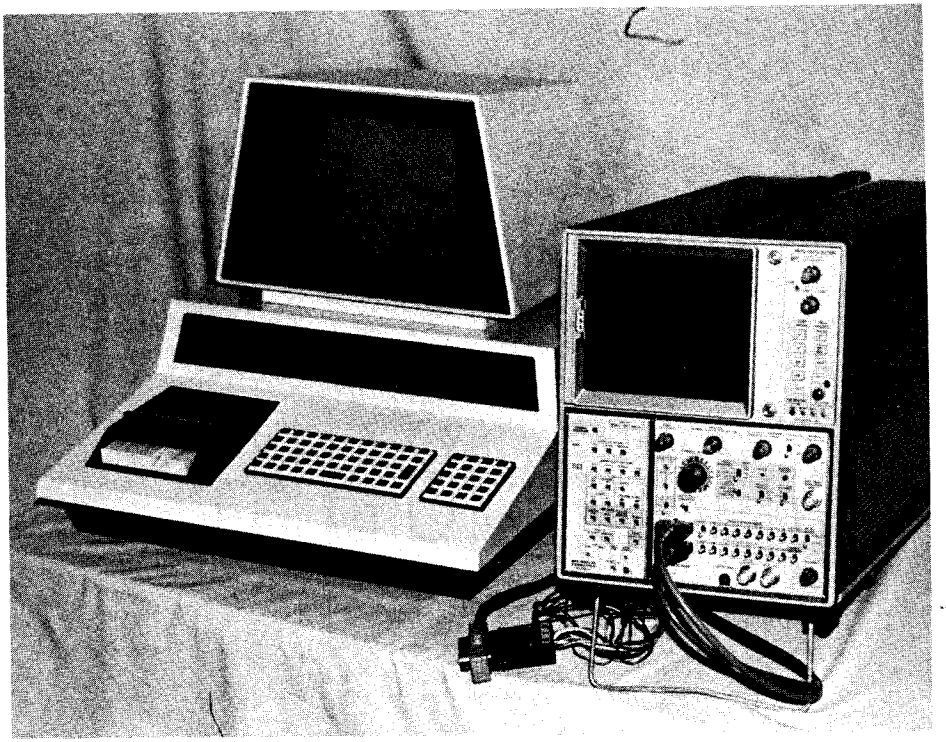
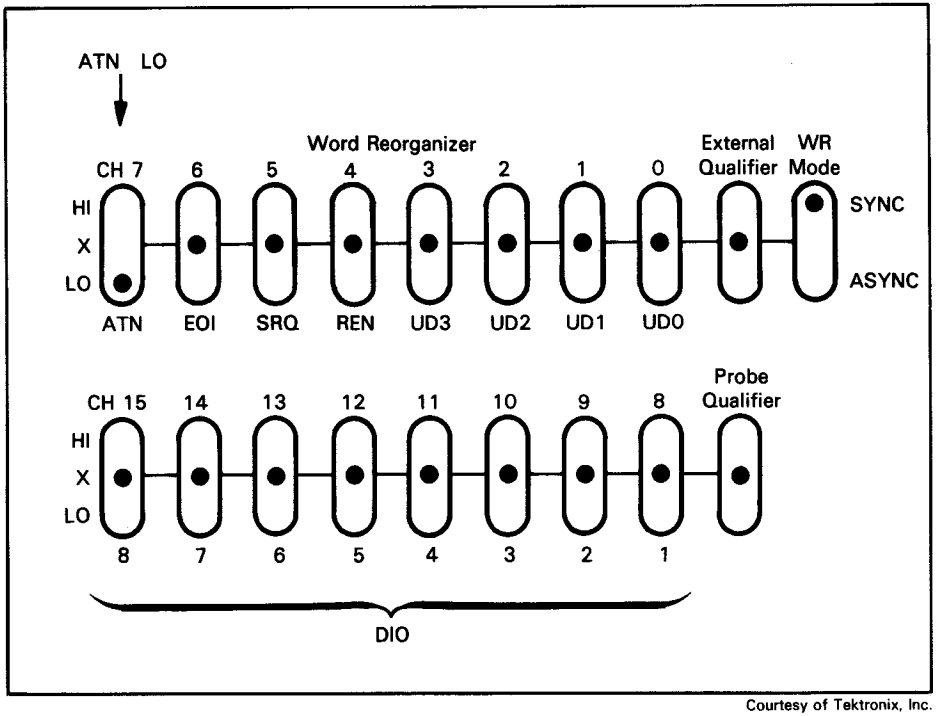


Figure 7-20. The Tektronix 7D01 Logic Analyzer and DF2 Display Formatter connected to the PET via the Tektronix GPIB Adapter and the GPIB shown in the foreground



Courtesy of Tektronix, Inc.

Figure 7-21. Triggering on any ATN message. To trigger the logic analyzer on any ATN message, set the word recognizer switch CH7 in the LO position and all other switches 0-6 and 8-15 to the X (don't care) position.

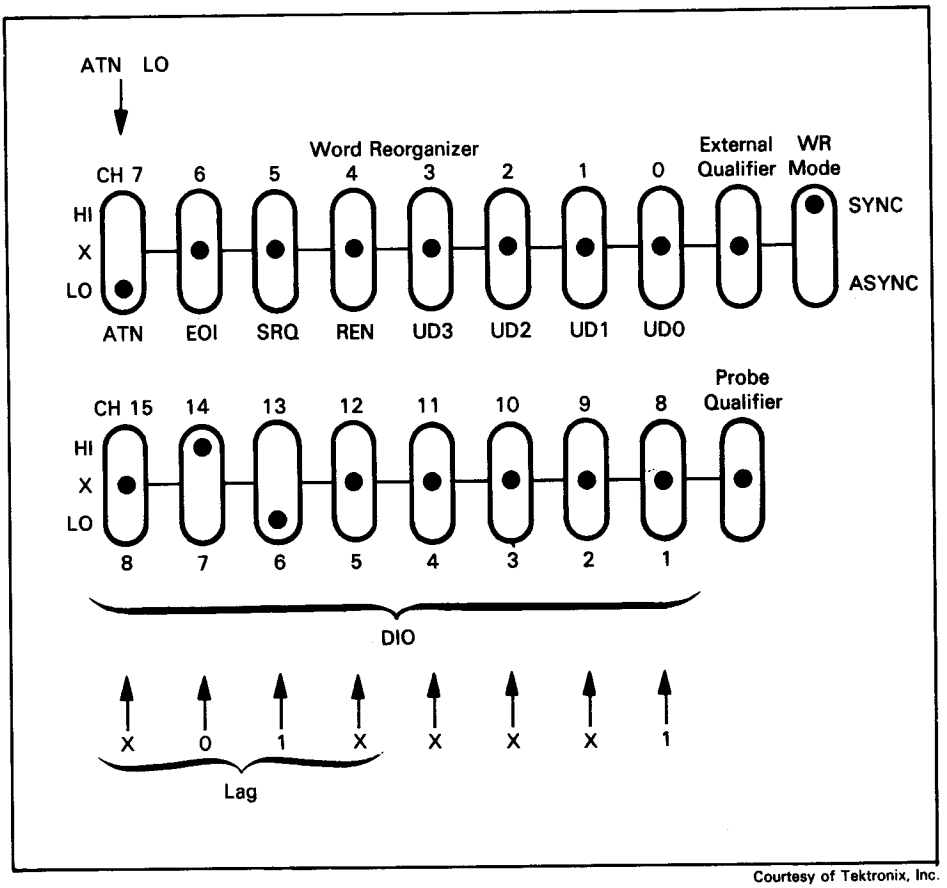
Trigger on Any ATN Message Group

After the CH7 switch is set, any of the choices for triggering on an ATN message group can be made by setting switch channels 8-15.

When the Model 7D01 logic analyzer is triggered on an ATN message group, the WR switches for channels 8-15 (corresponding to DIO lines 1 through 8) must be set to recognize the desired group. The higher-order bits define the group, and the lower-order bits define the specific address or message within the group. For example, to trigger the logic analyzer on the LAG (Listen Address Group), set WR switches 8-15 as follows:

$20 - $3F	Hexadecimal value range
X01X XXXX	Binary value
LO	ATN-CH7
X HI LO X XXXX	CH15-8

As illustrated in Figure 7-22, the switches are set so that when ATN is driven low (true), the logic analyzer will trigger and collect data. This data is operated on by the DF2 display formatter and displayed in a tabular format.



Courtesy of Tektronix, Inc.

Figure 7-22. Triggering on a message group. These positions of the WR switches select the LAG (Listen Address Group).

Data Display Example

A sample display obtained by triggering the logic analyzer on the LAG, as described above, is shown in Figure 7-23. For this example, the PET BASIC statements are:

```
OPEN 5,5,2
PRINT# 5, "TEST"
```

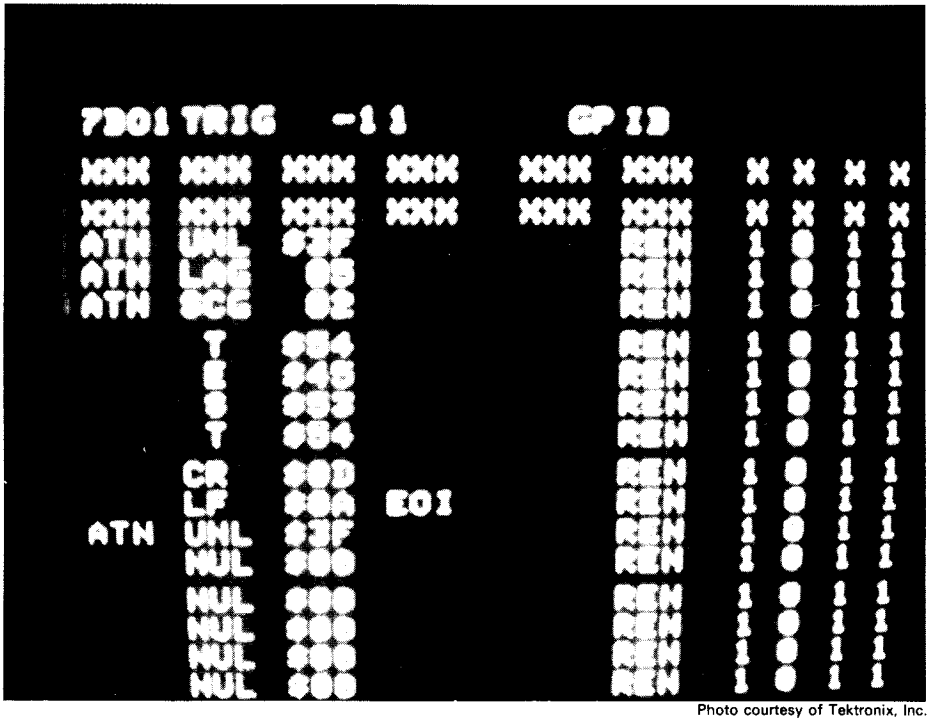


Photo courtesy of Tektronix, Inc.

Figure 7-23. The image of the DF2 Display Formatter when the logic analyzer is triggered on the LAG (Listen Address Group). See text for the OPEN and PRINT statements that provide data for the display.

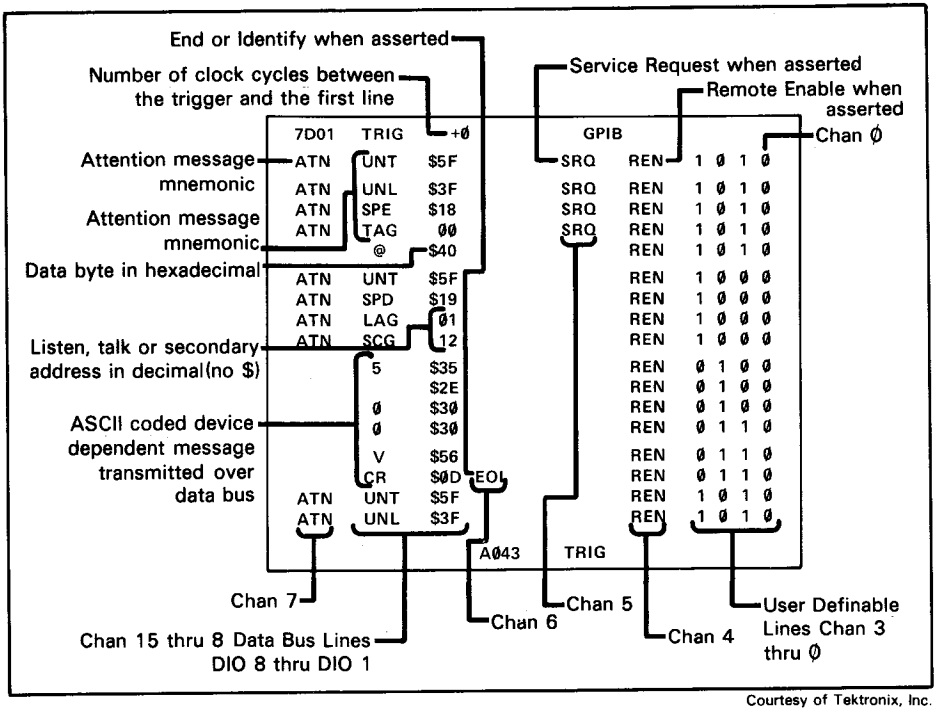
At the top of the screen in Figure 7-23, the ATN line is shown asserted; the UNL (Unlisten) signal sends $3F_{16}$ to the GPIB. Next, the OPEN 5,5,2 statement to the PET results in device number 5, a listener on the bus, being sent. This is illustrated by the fourth entry from the top, "ATN LAG 05." The next entry shows the results of the secondary address in the OPEN statement. These are the characters SCG (Secondary Command Group) 02.

The bus transactions that are actual data show up on the next four entries as TEST. Each letter is accompanied by its corresponding hexadecimal value transferred on the GPIB. (The $ sign preceding each group of two numbers signifies the numbers to be hexadecimal.)

The termination of the data transmission is signified by the CR/LF (Carriage Return/Line Feed), which is sent on termination of the PRINT statement. The CR/LF causes values of $0D_{16}$ and $0A_{16}$, respectively, to be transmitted on the GPIB. Note that the EOI line is also asserted, which says, in effect, that T is the last character in the data burst.

Detailed Display Format

Figure 7-24 is another example of the tabular format produced by the DF2 display formatter when the logic analyzer is attached to the GPIB. This display gives an even more complete representation of how the various communication signals between the PET, the GPIB, and remote devices can be analyzed. For example, in this display on the top line the number +0 indicates that there were zero clock cycles between the trigger pulse and the first entry of the bus transactions; in Figure 7-23 there were 11 such clock cycles. Whereas Figure 7-23 shows that a Listen Address Group 05 was chosen (a listener), Figure 7-24 illustrates on the ninth entry from the top that listener number 01 is chosen on the bus. (Note that with the PET controller on the bus, this number would not be likely because PET reserves locations 0 through 3 for its own internal functions.) In Figure 7-24 the data sample sent on the GPIB is 5.00 volts (V); again each numeral or letter is represented by its corresponding hexadecimal value.



Courtesy of Tektronix, Inc.

Figure 7-24. Typical DF2 tabular format of GPIB bus transactions showing IEEE Std 488-1978 message mnemonics.

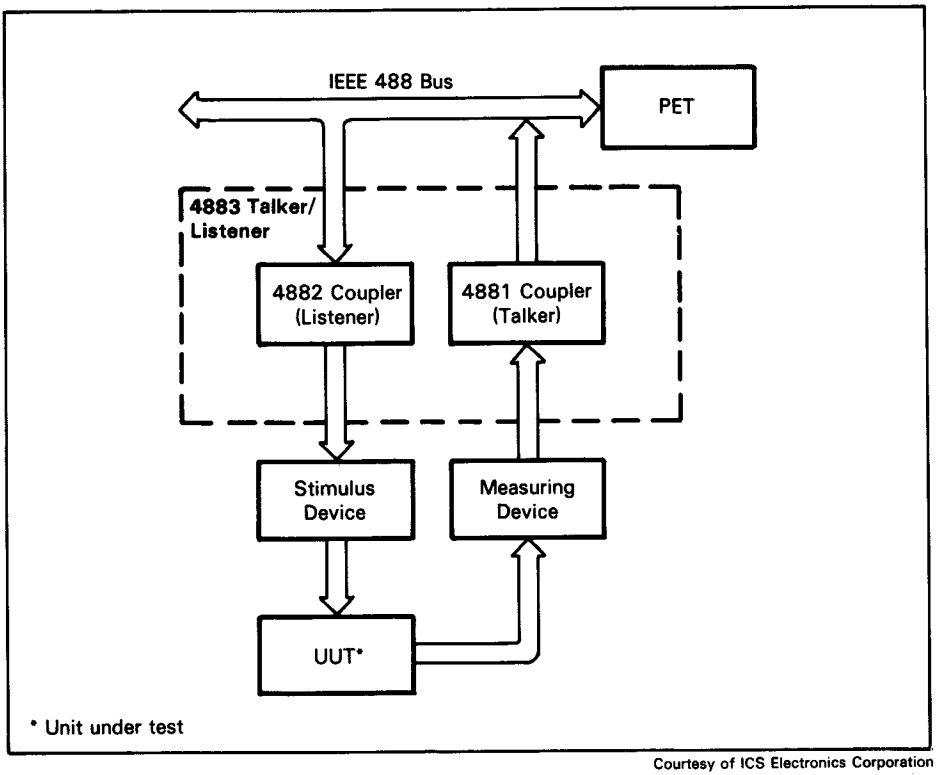
Many other indications are on the display, such as SRQ (Service Request), REN (Remote Enable), and EOI (End Or Identify). All of these mnemonic "tags" represent standard conditions in the IEEE Std 488-1978 specifications. This display relieves you from having to interpret the logic level signals on the GPIB to locate an error in a bus transaction.

488 BUS INTERFACE COUPLER, ICS MODEL 4880

Quite often digital electronic instruments, although TTL compatible, are designed for a dedicated purpose and do not communicate with other devices on an IEEE 488 Bus. Also, many of the older digital instruments are not IEEE 488 Bus compatible; these instruments typically have multiple digits of input/output where each digit requires four bits (lines). For example, a minimum of 32 lines or wires is required for an 8-bit device. For some time a means has been needed that would interconnect these devices through a buffer and provide IEEE 488 Bus compatibility. Such an instrument is now marketed. It is the 488 Bus Interface Coupler, Model 4880, manufactured by ICS Electronics Corporation. This device, illustrated in Figure 7-25, has 40 lines for either input or output and comes in three basic configurations: talker (Model 4881), listener (Model 4882), and talker/listener (Model 4883).

Photo courtesy of ICS Electronics Corporation

Figure 7-25. The Model 4880 Instrument Coupler



Courtesy of ICS Electronics Corporation

Figure 7-26. The PET and the 4880 Coupler in a typical application

The Model 4880 couplers provide all the necessary interface functions to connect any digital programmable device or an instrument's data output to the GPIB. The devices, which will be connected by the coupler to the GPIB, can be talkers, listeners, or talker/listeners. Figure 7-26 shows two, one as a stimulus device (a listener) and the other as a measuring device (the talker).

When the system is assembled, addresses are set into each coupler device (Models 4881, 4882, and 4883) to provide a unique address or code for the instrument to which it is coupled. Thus, the 4880 modules respond (for the coupled devices) to a poll and they accept input or output data depending on the coupled devices' function within the system. Each 4880 module, according to its role in the system, performs the appropriate IEEE functions such as handshake, single or extended address talk or listen, service request, parallel poll, device clear, and device trigger.

With the permission of the ICS Electronics Corporation, the information below is taken directly from the data sheet describing the 488 Bus Interface Coupler Model 4880.

TALKER VERSION

As a talker, the 4880 Coupler can be used to transfer data from any instrument to the IEEE 488 Bus. In this configuration the 4880 can recognize the BUS Device Trigger command and initiate an instrument read cycle or wait for a free running instrument.

At the end of the instrument read cycle, the coupler will generate a service request (SRQ) or output the data if the 4880 has been addressed to talk. The 4880 will respond to a serial poll with a hex 41 character if it has data ready to be sent to the controller, with a hex 0 if no data is available. The 4880's data transmission format can be any mix of data and ASCII symbol characters, up to 16 characters long. The standard 4880 data transmission format consists of 12 characters and is:

D D D D D D D D D D CR LF

Where:

D is either a numeric digit 0-9 or coded as a . , + – E or space

CR is carriage return

LF is line feed

The user can order a non-standard transmission format (option-06) or reprogram the UV Prom format table following the directions in the manual. The 4880 talker converts BCD and HEX instrument data into ASCII numbers and characters according to the Talk Character Table. Again, the user can order a non-standard table or alter the character table by following the instructions in the manual.

The 4880 talker versions respond to the universal device clear and selected device clear commands by resetting their internal logic and all device command lines. The response to the universal command, DT (device trigger), is to output a pulse which initiates a reading. The 4880 will asynchronously generate a service request for the controller when the data is ready. The user can inhibit generation of service requests by depressing the front panel SRQ INHIBIT switch. The user can put the 4880 into the 'talk only' mode by setting the ONLY and TALK switches on the rear panel (part of the address switch).

LISTENER VERSION

As a listener, the 4880 can be used to control programmable devices from the bus or to output parallel data. In this configuration, the 4880 can be told to listen only when addressed by the bus controller or at all times. The user can obtain the commanded device status from the 4880 by conducting a parallel poll. The BUS Parallel Poll Enable (PPE) command causes the 4880 to put the current READY Status bit in the poll response word. The 4880 outputs the poll response word during the parallel poll IDY 'on' time.

When addressed as a listener, the 4880 accepts the complete ASCII number set and transfers the least four bits on to the output register. The ASCII numbers 0-9 result in the BCD digits 0-9 on the output connector while the ASCII symbols : ; < = > and ? result in the HEX characters A through F on the output connector. The command string may be up to 10 characters long followed by a delineator (CR, LF or comma) or ended by the EOI signal. The 4880 holds all command characters in latches which are only changed when new data is transferred from memory after receipt of the complete command message.

The 4880 Listener version responds to the universal device clear and selected device clear commands by resetting its internal logic and all command outputs to their initial value. The user can place the 4880 listener into the 'listen-only' mode by setting the ONLY and LISTEN switches on the rear panel.

TALKER/LISTENER VERSION

A 4880 Coupler configured as a talker/listener can be used to control or take data from a programmable measuring device like a programmable DVM. The coupler would accept program commands from the bus and pass them onto the DVM's control inputs, initiate readings and output data when addressed. The talker/listener version combines all the capability and operational features of the talker and listener in one unit. The talker/listener can also be used to control one instrument and input data from a second instrument by properly cabling it to the two instruments. The talker/listener can be reduced to either a talker only or a listener only by properly setting the mode switches on the rear panel.

PET SERVICE REQUEST (SRQ) PROGRAM

Those who are familiar with the PET's internal connections at the IEEE 488 user port J1 know that the PET's Service Request (SRQ) line recognizes an input-only signal. There is no output control of this line by the PET, nor is SRQ implemented in PET BASIC. Thus, a problem exists when a remote device wants to tell the PET, for example, to take the device's readings.

However, connected through the 4880 coupler, a non-bus device (for example a talker) can be made to tell the PET when to take its readings. This is accomplished via the GPIB by a program that tells the PET to monitor the SRQ line. (Location 59427, bit 7, which senses the input of a service request, is tested.) This program allows the remote device beyond the coupler to "probe" the coupler, which in turn makes a service request to the PET. With this set-up, the PET is able

to recognize that the coupler is making a service request, a capability that the PET normally does not have.

To illustrate how such a service request can be made, we connected the 4880 coupler and the PET to the GPIB with the standard bus cabling. The program, which is listed in Figure 7-27, was written to test the coupler's input and output functions.

```
5 PRINT "□":REM CLEAR SCREEN
10 OPEN 5,6
20 INPUT# 5,B$
30 A = PEEK(59426)
40 IF PEEK(59427) AND 128 THEN GOSUB 100
50 PRINT "TESTING"
60 GOTO 40
100 PRINT "□":REM CLEAR SCREEN
105 INPUT# 5,I$
110 PRINT# 5,I$
120 A = PEEK(59426)
130 RETURN
```

Figure 7-27. Service Request (SRQ) program

In this program, the SRQ line in its non-asserted state allows the PET controller to continue performing some off-line operation such as a calculation or data manipulation. When asserted by the 4880 coupler, the SRQ signal is detected by the PET. It then accepts a reading from an external device.

Upon detecting an SRQ, the program clears the PET CRT screen, reads output data from the device coupler, and then immediately prints the data back out to the bus and to the input of the device coupler. The program remains in a wait loop, printing a "TESTING" message continuously to the screen, until an SRQ is received.

In this program, line 5 clears the screen. Line 10 opens a file number 5 for device number 6 residing on the bus. The INPUT statement on line 20 clears any pending service request from the device coupler. Also, any previous transitions that may have occurred on the bus and been stored in the PET are cleared (line 30). Both of these would have interfered with coming operations. The loop at lines 40-60 monitors the SRQ signal line to see if there has been a high-to-low transition. At line 40, location 59427 is tested for bit 7, which senses the input of an SRQ. If no transition has occurred, the word "TESTING" is printed repeatedly to the CRT; the screen scrolls continuously.

If there has been a transition of the SRQ signal, the subroutine at line 100 is called. After clearing the screen (line 100), the subroutine reads input from the device coupler (line 105). Then the program echoes the same data (line 110) to be picked up by the device coupler. Line 120 clears the acknowledged SRQ, and line 130 returns to the wait loop.

OTHER IMPLEMENTATION NOTES

The following paragraphs are adapted from Application Bulletin 48-7, put out by the ICS Electronics Corporation and used with its permission. The material was contributed by Robert Huenemann of United Air Line Maintenance Base, San Francisco, California. The operation described in detail below is based on having several devices take data simultaneously as a result of a Group Execute Trigger (GET) command. The program suggests other possibilities that could provide IEEE functions not directly implemented by PET BASIC statements. Examples are DEVICE CLEAR, PARALLEL POLL and SERIAL POLL.

Group Execute Trigger Program

Not to be confused with the GET statement calling for PET to "get" and examine one character at a time, this GET command is an acronym for Group Execute Trigger. Such a command is often necessary to command all devices on the bus simultaneously and cause them to trigger and start taking readings (for example) at the same time. However, this command is not implemented directly by the PET.

The GET can be accomplished by using standard BASIC statements shown in the program in Figure 7-28.

```
1000 Z=63:GOSUB 7000:REM UNLISTEN
1010 Z=36:GOSUB 7000:REM LISTEN ADDRESS
1020 Z=8:GOSUB 7000:REM GET
1030 INPUT# 4,Z$,Y$
1040 IF Z$ = " " THEN Z$ = ", "+Y$
1050 REM 1030 AND 1040 CIRCUMVENT THE FACT THAT THE PET INTERPRETS ", " AS A
1060 REM DELIMITER, EVEN IN A CHARACTERSTRING.  THE FIRST CHARRACTER FROM THE
1070 REM 4880 CAN BE ANY OF 16 CHARRACTERS IN THIS APPLICATION, INCLUDING ", ".
7000 GOSUB 8000:REM PET TALK HANDSHAKE
7010 IF (PEEK(59456) AND 64) <> 64 GOTO 7010
7015 REM WAIT FOR NRFD FALSE
7020 POKE 59426, 255-Z
7025 REM INVERT DATA AND OUTPUT
7030 POKE 59456, 251:REM SET ATN TRUE
7040 POKE 59427, 52:REM SET DAV TRUE
7050 IF (PEEK(59456) AND 1) <> 1 GOTO 7050
7055 REM WAIT FOR NDAC TRUE
7060 GOSUB 8000:RETURN
8000 REM HOUSEKEEPING
8005 POKE 59427, 60:REM SET DAV FALSE
8010 POKE 59456, 255:REM SET ATN FALSE
8020 POKE 59426, 255:REM SET DATA HIGH
8030 POKE 59425,60:REM SET NDAC HIGH
8040 RETURN
```

Figure 7-28. PET Group Execute Trigger program

In this program, line 1000 executes the universal unlisten command on the GPIB, which tells all listeners to "get off the bus."

Line 1010 first issues the address to the listener that is going to be triggered. This is followed by the GET command on line 1020, which causes the instrument to send a trigger pulse. This GET command causes the 4880 coupler to generate an output strobe, which an external device can use to start its own conversion or triggering function.

The remark at line 1050 states that the small input routine shown on lines 1030 and 1040 will allow characters (such as commas) from the 4880 coupler to be input without the PET interpreting these inputs as delimiters.

At line 7000, a subroutine performs the IEEE handshake procedure at the PET BASIC rate, which is much slower than normal GPIB operation. However, because of the handshake nature of the IEEE Bus, i.e., the sequential, one-signal-dependent-on-another timing, this subroutine yields satisfactory results.

The subroutine at line 8000 returns several of the control lines to their inactive states.

Output Limitations of the PRINT Statement

If you have remote instruments coupled to the GPIB through the Model 4880, you must become aware of the format limitations in the PRINT statement concerning the characters that are sent to the bus. In particular, note that when the PET sends numbers through the coupler to a remote device, a different series of characters is generated at the receiving end from those initiated by the PET. This series of characters depends on the size of the number and whether the number is an integer, floating point, or string.

You should also be thoroughly familiar with the unit before proceeding to write programs or otherwise try to operate it in the GPIB system.* This is because there are unique features in the 4880 instrument. For example, the coupler looks at the most significant digit first and proceeds to the least significant digit, using only the last four bits of each individual character that is sent down the bus. This takes place before the coupler can read in a character and load its output device. Therefore, if the PET places the binary bit pattern 0100 0001 on the bus, the coupler would provide only the last four bits (the 0001) at its output.

*Refer to the 4880 Coupler Instruction Manual.

CHAPTER 8

A GPIB Diagnostic Test

THE SYSTEM DOESN'T WORK — NOW WHAT?

When communication difficulties occur between instruments on the GPIB, you probably will have nagging thoughts that the problem could be with the bus hardware. Therefore, we have written this diagnostic test to help you locate the problems, or at least alleviate your fears.

The test validates the correct operation of all the PET's GPIB lines except interface management lines IFC, SRQ, and REN. For the 13 lines that this test does validate, the PET indicates bad bits or signals by flashing their names on the CRT. Remember that the data bits start at 0, the least significant bit (LSB), and go to 7, the most significant bit (MSB).

Avoid inconclusive results; remove all devices from the GPIB before running this test.

Proceed as follows:

On the PET, type the GPIB Diagnostic Test Program as it is listed at the end of this chapter, being careful to avoid or correct typing errors; as the words and symbols appear on the CRT, compare them with the actual listing. You can add spaces between keywords, but this is not necessary. For example, GOTO is a keyword to be typed preferably without space. This and other keywords require no spacing.

After entering the last line in the program, type RUN and then depress the RETURN key. The PET will run the program and continue executing it in a loop. If

there are no faulty control lines or bad data bits, the words "END OF TEST" will continue to flash in the upper left corner of the CRT. This is due to the continuous looping of the program as it is being executed. However, should there be one or more faulty lines, each will be displayed one under the other, beginning in the upper left corner of the screen. For example, if data bits 7 and 2 are found to be held at either the high or low logic level, the PET will display:

```
THE BAD GPIB DATA BITS ARE
BIT 7
BIT 2
END OF TEST
```

Only those lines that are not operating will be displayed and continue to flash off and on. If a bad bit or line is cleared, the flashing indication disappears.

DIAGNOSTIC PROGRAM DESCRIPTION

The program, executing in a loop, tests the transfer control and management lines and data bits.

In the program listings, lines 100 to 230 test the GPIB data bus. Each bit is written at a 1 level (low true) as a first step, and then at a 0 level. Following each step, a comparison is performed on each bit to determine if it is held at a high or low logic level.

Next in the program listing, lines 300 to 411 test the NDAC line. This test is similar to those performed on the data bus except that only a single bit is tested. The remaining transfer control lines are tested similarly.

Further on in the program listing, lines 420 to 511 test the DAV signal bus line, and lines 520 to 611 test the NRFD bus line.

Finally, lines 620 to 700 test the interface management line ATN. The Attention line is tested differently; only a transition is looked for in the PET, and the transition causes an interrupt.

The GPIB Diagnostic Test Program may be summarized as follows:

Program Listing Numbers	GPIB Line Tested
100 to 230	Data bus (DIO bits 0 to 7)
300 to 411	Not Data Accepted (NDAC)
420 to 511	Data Available (DAV)
520 to 611	Not Ready For Data (NRFD)
620 to 700	Attention (ATN)
702 to 800	End Or Identify (EOI)
(No test)	PET non-standard interface management lines IFC, SRQ, and REN

To write this program on a magnetic tape cassette installed in the PET, type:

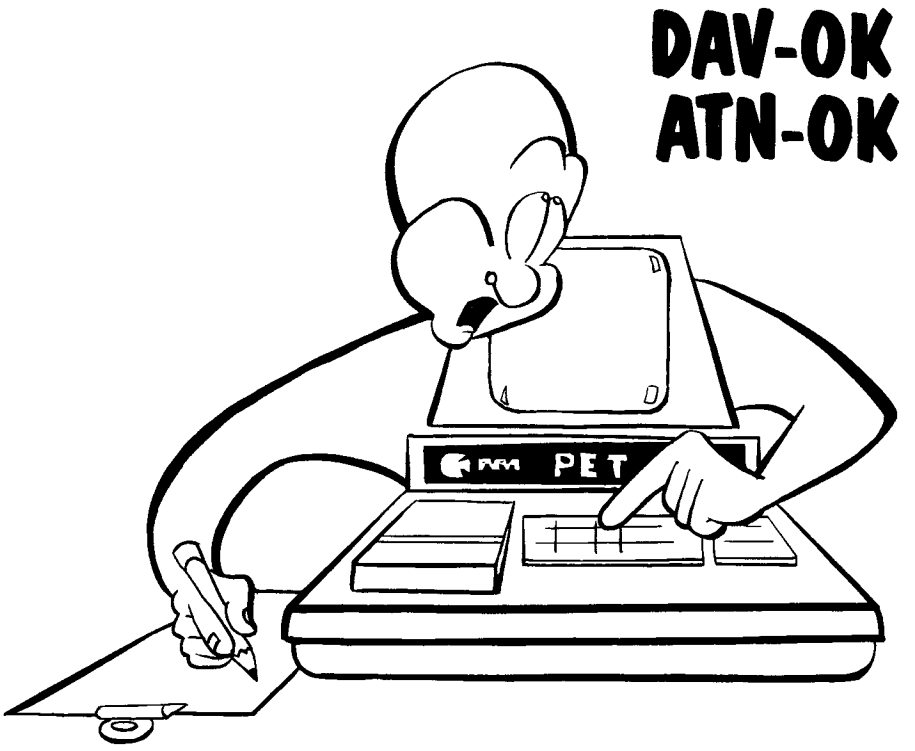
SAVE "GPIB TEST" (Depress RETURN)

Verify that the program was written correctly by rewinding the tape and type:

VERIFY "GPIB TEST" (Depress RETURN)

The program can subsequently be reloaded by rewinding the tape and issuing the command:

LOAD "GPIB TEST" (Depress RETURN)



```
70 PRINT "J":REM CLEAR SCREEN
100 POKE 59426,0
110 C=PEEK(59424)
120 C1=255 AND C
130 POKE 59426,255
140 C2=NOT PEEK(59424) AND 255
150 BAL=(ABS(NOT C1 AND NOT C2))-1
154 IF BAL=0 GOTO 300
155 PRINT "THE BAD GPIB DATA BITS ARE:"
160 IF BAL>127 THEN PRINT "BIT 7":BAL=BAL-128
170 IF BAL>63 THEN PRINT "BIT 6":BAL=BAL-64
180 IF BAL>31 THEN PRINT "BIT 5":BAL=BAL-32
190 IF BAL>15 THEN PRINT "BIT 4":BAL=BAL-16
200 IF BAL>7 THEN PRINT "BIT 3":BAL=BAL-8
210 IF BAL>3 THEN PRINT "BIT 2":BAL=BAL-4
220 IF BAL>1 THEN PRINT "BIT 1":BAL=BAL-2
230 IF BAL=1 THEN PRINT "BIT 0":BAL=BAL-1
240 :
300 POKE 59425,52:C=PEEK(59456)
310 POKE 59425,60:C1=PEEK(59456)
330 MSK=1
350 A=C AND MSK
355 IF A=1 THEN GOTO 400
360 A=C1 AND MSK
370 IF A=1 THEN GOTO 410
400 PRINT "NDAC IS BAD"
410 POKE 59425,60
411 :
420 REM TEST DAV
430 POKE 59427,52:C=PEEK(59456)
440 POKE 59427,60:C1=PEEK(59456)
450 MSK=128
460 A=C AND MSK
470 IF A=128 THEN GOTO 500
480 A=C1 AND MSK
485 IF A=128 THEN GOTO 510
500 PRINT "DAV IS BAD"
510 POKE 59425,60
511 :
520 REM TEST NRFD
530 POKE 59456,255:C=PEEK(59456)
540 POKE 59456,253:C1=PEEK(59456)
550 MSK=64
560 A=C AND MSK
570 IF A=0 THEN GOTO 600
580 A=C1 AND MSK
590 IF A=0 THEN GOTO 610
600 PRINT "NRFD IS BAD"
610 POKE 59456,255
611 :
620 REM TEST ATN
630 POKE 59456,251:C=PEEK(59425)
631 C1=PEEK(59424)
640 POKE 59456,255:C1=PEEK(59425)
650 MSK=128
660 A=C AND MSK
670 IF A=0 THEN GOTO 700
680 A=C1 AND MSK
685 IF A=0 THEN GOTO 705
700 PRINT "ATN IS BAD"
702 :
703 REM TEST EOI
705 POKE 59409,52:C=PEEK(59408)
710 POKE 59409,60:C1=PEEK(59408)
730 MSK=64
750 A=C AND MSK
755 IF A=64 THEN GOTO 800
760 A=C1 AND MSK
770 IF A=64 THEN GOTO 810
800 PRINT "EOI IS BAD"
810 PRINT "   END OF TEST"
820 FOR I=1 TO 600:NEXT I
830 GOTO 70
```

Figure 8-1. GPIB Diagnostic Test Program

APPENDIX A

Companies and their GPIB Products

Following is a partial listing of companies and their products manufactured for use with the IEEE 488 Standard Bus, the GPIB. This list is offered as a reader service to help locate industries and products. As is true with any such list, it is dated. Therefore, neither the authors nor the publisher make claims as to the accuracy of the information. Some devices may have been dropped from product lines, others superseded, while new instruments surely will have been marketed.

CALCULATORS/COMPUTERS/CONTROLLERS/PROCESSORS

Product	Model	Manufacturer
Calculator	9815A	Hewlett-Packard
Calculator	9825A	Hewlett-Packard
Calculator	9830A	Hewlett-Packard
Calculator	9830B	Hewlett-Packard
Computer	9800, System 45	Hewlett-Packard
Computer/Controller	2001, PET	Commodore
Computer measurement and control system	2240A	Hewlett-Packard
Computer, micro	LSI-11	Digital Equipment Corporation
Computer series	LS14	Computer Automation
Computer, system	1000	Hewlett-Packard
Controller	2649A	Hewlett-Packard
Controller, IEEE Instrumentation Bus		MDB Systems Inc. (for Perkin-Elmer (Interdata) Computers)
Controller	2200	Wang
Controller	BIO-2	Wyle
Controller, bus	583	Wavetek Indiana Inc.
Controller, instrument	PIC-1	Instrumentation Tech Corp.
Controller, instrument	3530	Systron-Donner
File Manager (Mass storage device)	4907	Tektronix
Processor, for SBC-80	80	Ziatech
Processor, instrument	6771	Electro Design Inc.
Processor, interface	1100B	Fluke
Instrumentation Controller	3530	Jaycor
Matrix Switching System	3570A	Jaycor
RS-232 to IEEE 488 Bus Controller	4885A	ICS Electronics Corporation

INTERCONNECTING DEVICES, SWITCHES

Product	Model	Manufacturer
Adapter, Centronix/ GPIB	C-101	F.I. Electronics/Com-plications
Adapter, GPIB/RS232	111	System Consultants
Cable Assembly	PET 488	Pickles and Trout (Connects IEEE 488 system to PET board edge connector, J1)
IEEE standard cables w/connectors		
1 meter	9642	Belden Corp.
2 meter	9643	Belden Corp.
4 meter	9644	Belden Corp.
8 meter	9645	Belden Corp.
Communications Card	53A-127	Computer Data System
Converter, GPIB	3910/3911	Wavetek Indiana, Inc.
Coupler, instrument	4880	Fairchild
Coupler, GPIB/RS 232	4884	Fairchild
Coupler, modem	IEEE	Livermore Data
488 Bus Interface Coupler	4880	ICS Electronics Corporation
HP-IB Common Carrier Interface	59403A	Hewlett-Packard

INTERCONNECTING DEVICES, SWITCHES (Continued)

Product	Model	Manufacturer
HP-IB Interconnection Cable	10631A,B,C,D	Hewlett-Packard
IEEE-488 Bus Cont/ Analyzer Card	53A-488	Hewlett-Packard
IEEE Interface Option	2240B-15	Fluke
IEEE-488 Bus Interface Unit	10A	Boonton
IEEE Standard connector P.N. with connections to P6451 probe head	103-0209-00	Tektronix
Insulation displacement receptacle	552305-1	AMP
Insulation displacement plug	552301-1	AMP
Interface	780,790	Keithley Instruments, Inc.
Interface	IEC11A	Digital Equipment Corporation
Interface, digital multimeter	1723	Keithley Instruments, Inc.
Interface, GPIB	703	AMS, Inc.
Interface, GPIB	704	AMS, Inc.
Interface, GPIB	55	Dana Laboratories, Inc.
Interface, GPIB	IBV11A	Digital Equipment Corp.
Interface, GPIB	70755A	Guideline Instruments
Interface, GPIB	55	Keithley Instruments, Inc.
Interface, GPIB	GPIB11	National
Interface, GPIB	2254	Wang
Interface, GPIB/CAMAC	3388	Kinetic Systems Corp.
Interface, Scanner	7033	Keithley Instruments, Inc.
Multiplexer, reed relay	12	A.D. Data Systems
Party-Line System	53A	Computer Data Systems
Plug	57-1024-0	Amphenol
Plug, solder type	5710240	Cinch
Programmable Ratio Transformer	PRT-10	Ailtech
Receptacle	57-2024-0	Amphenol
Receptacle, solder type	5720240	Cinch
Scanner	2460	Philips Test and Measuring, Inc.
Selector	1554	Bauel and Kjaer
Smart Hardware	53A-1BX	Computer Data Systems
Switch, access	3754	Hewlett-Packard
Switch, controller	9411A	Hewlett-Packard
Switch, matrix	56A	A.D. Data Systems
Switch, matrix	9414A	Hewlett-Packard
Switch, modular	9412A	Hewlett-Packard
Switch, relay	59306A	Hewlett-Packard
Switch, system matrix	3570	Systron-Donner
Switch, VHF	9413A	Hewlett-Packard
Switch, VHF	59307A	Hewlett-Packard

MEASURING INSTRUMENTS

Product	Model	Manufacturer
Analyzer	660	Nicolet Scientific Corp.
Analyzer, AF	3347	Bruel and Kjaer
Analyzer, bus system	59401A	Hewlett-Packard
Analyzer, data line	3771A and B	Hewlett-Packard
Analyzer, digital signal	5420A	Hewlett-Packard
Analyzer, frequency	2131	Bruel and Kjaer
Analyzer, frequency	1174	Solartron Elec/Schlumberger
Analyzer, logic	1130	EH Research Laboratories, Inc.
Analyzer, logic	LA1850	EH Research Laboratories, Inc.
Analyzer, logic	1602A	Hewlett-Packard
Analyzer, logic	532	Paratronics, Inc.
Analyzer, logic	1625	Vector Assoc., Inc.
Analyzer, modulation	8901A	Hewlett-Packard
Analyzer, noise level	4426	Bruel and Kjaer
Analyzer, pri. multiplex	3779A and B	Hewlett-Packard
Analyzer, pulse	780	Systron-Donner
Analyzer, RF network	8505A	Hewlett-Packard
Analyzer, serial data	1640A	Hewlett-Packard
Analyzer, spectrum	3582A	Hewlett-Packard
Analyzer, spectrum	8568A	Hewlett-Packard
Analyzer, spectrum	600	Polarad Electronics Inst.
Analyzer, storage	8501A	Hewlett-Packard
Analyzer, swept spectrum	3585A	Hewlett-Packard
Analyzer, vector	2PV	Rohde and Schwarz
Analyzer, waveform	770	Systron-Donner
Analyzer, waveform	774	Systron-Donner
Angle Indicator	8800	North Atlantic
Angle Indicator	8810	North Atlantic
Attenuator	DPVP	Rohde and Schwarz
Bridge, automatic	76A	Boonton
Capacitance Bridge	4270A	Hewlett-Packard
Capacitance Meter, digital	4282A	Hewlett-Packard
CINCH Intelligent Data Acquisition and Control Systems		Control Logic
Communication Receiver	3000F	Decca Communications Ltd.
Converter, A to D	47310A	Hewlett-Packard
Converter, A to D	59318A	Hewlett-Packard
Counter	351D	EIP Dana
Counter, frequency	11X	Data Technology
Counter, general purpose	5345A	Hewlett-Packard
Counter, microwave	331	EIP Dana
Counter, microwave	371	EIP Dana
Counter, microwave	5340A	Hewlett-Packard
Counter, microwave	5341A	Hewlett-Packard
Counter, microwave	5342A	Hewlett-Packard
Counter, pulse	451	EIP Dana
Counter series	2700	Enertec/Schlumberger
Counter, time interval	797	Eldorado
Counter, time interval	5770A	Hewlett-Packard
Counter/Timer	5500B	Ballantine
Counter/Timer	9000	Dana Laboratories, Inc.

MEASURING INSTRUMENTS (Continued)

Product	Model	Manufacturer
Counter/Timer	9015	Dana Laboratories, Inc.
Counter/Timer	9035	Dana Laboratories, Inc.
Counter/Timer	10X	Data Technology
Counter/Timer	1953A	Fluke
Counter/Timer	6600 series	Philips Test and Measuring Inc.
Counter/Timer	9900 series	Racal-Dana Inst. Ltd.
Counters	6054B,C,D	Systron-Donner
Counter, universal	5328A	Hewlett-Packard
Digital Multimeter	DSM44	California Instrument Co.
Digital Multimeter	DSM52	California Instrument Co.
Digital Multimeter	3400	Data Precision
Digital Multimeter	7500	Data Precision
Digital Multimeter	1051	Datron
Digital Multimeter	8520A	Fluke
Digital Multimeter	8810A	Fluke
Digital Multimeter	172A	Keithley
Digital Multimeter	173A	Keithley
Digital Multimeter	177,179, 179-20A, 480	Keithley
Digital Multimeter	2441	Philips Test and Measuring, Inc.
Digital Multimeter	2526	Philips Test and Measuring, Inc.
Digital Multimeter	2527	Philips Test and Measuring, Inc.
Digital Multimeter	7075	Solartron Elec./Schlumberger
Digital Multimeter	7115	Systron-Donner
Digital Multimeter	7241	Systron-Donner
Digital Multimeter	7244A	Systron-Donner
Digital Multimeter	7344A	Systron-Donner
Digital Noise Figure and Gain Monitor	7380	Ailtech
Digital Voltmeter	5000	Dana Laboratories, Inc.
Digital Voltmeter	5900	Dana Laboratories, Inc.
Digital Voltmeter	6900	Dana Laboratories, Inc.
Digital Voltmeter	8500A	Fluke
Digital Voltmeter	8920A	Fluke
Digital Voltmeter	9575	Guideline Instruments
Digital Voltmeter	3437A	Hewlett-Packard
Digital Voltmeter	3438A	Hewlett-Packard
Digital Voltmeter	3455A	Hewlett-Packard
Digital Voltmeter	3490A	Hewlett-Packard
Digital Voltmeter, precision	9577	Guideline Instruments
EMI/RFI Test Instrumentation	FC-67 and NM-67	Ailtech
Filter, low pass	4001	Precision Filter, Inc.
Filter, multi-channel	616	Precision Filter, Inc.
HP-IB Interface, talker	5312A	Hewlett-Packard
Laser Transducer	5501A	Hewlett-Packard
LC Bridge	1687	General Radio
LCR Meter	296	Electro Scientific Industries
LCR Meter	4274A	Hewlett-Packard
LCR Meter	4275A	Hewlett-Packard
LCR Meter, automatic	4262A	Hewlett-Packard
LCR Meter, digital	4271B	Hewlett-Packard
Level Measuring Set	3735A and B	Hewlett-Packard
Level Measuring Set	3747A and B	Hewlett-Packard
Meter, digital power	1009	Pacific Measurements, Inc.

MEASURING INSTRUMENTS (Continued)

Product	Model	Manufacturer
Meter, frequency, series	9910	Racal-Dana Inst., Ltd.
Meter, modulation	82AD	Booton
Meter, power	436A	Hewlett-Packard
Meter, power	10188	Pacific Measurements, Inc.
Meter, preset C	4272A	Hewlett-Packard
Meter, RD power	1045	Pacific Measurements, Inc.
Monitor, noise gain	7380	Ailtech
Monitor, system noise	7370	Ailtech
Multiplexer, analog	620	Neff Instrument Corp.
Oscilloscope, storage	OS4000, 4001, 4002	Gould Inc.
Oscilloscope, digital	111	Nicolet Scientific Corp.
Programmable Transient Digitizer	7912AD with 7A16P, 7B90P	Tektronix
Quartz Thermometer	2804A	Hewlett-Packard
RLD Bridge	76A	Boonton
RLC Bridge	1658	General Radio (GenRad)
RF Millivoltmeters	92B Series	Boonton
Selective Level Measuring Set	TF2356 TF2357	Marconi Instruments
Signal Recorder	4102	EG and G
S-Parameter Test Set	8503A and B	Hewlett-Packard
Spectrophotometer	SP8-100	Philips Test and Measuring, Inc.
Storage Normalizer	8501A	Hewlett-Packard
Thermometer	2804A	Hewlett-Packard
512 MHz "Thin Line" Counter	6041A, 6042A	EG and G
1250 MHz "Thin Line" Counter	6043A	EG and G
Time Interval Probes	5363A	Hewlett-Packard
Transmission Impairment Measuring System	4943A	Hewlett-Packard
Transmission Impairment Measuring System	4944A	Hewlett-Packard
Universal Counter/Timer	7250A	Fluke
Universal Counter/Timers	7260/7261A	Fluke
Universal Printer	2020A	Fluke
Voltmeter, phase angle	220/225	North Atlantic

PERIPHERAL DEVICES (Continued)

Product	Model	Manufacturer
Amplifier, Lock-In	9503	EG and G
Card reader	PCL	Rohde and Schwarz
Data logger	4000	Philips Test and Measuring, Inc.
Digital Cartridge Tape Drive	4924	Tektronix
Display, numeric	59304A	Hewlett-Packard
Floppy disk	21	Tri-Data Corp.
Floppy disk, dual drive	2040	Commodore (PET)
Formatter, tape	1015A	Dylon
Graphic System	4051	Tektronix
GPIB 488 Bus Cassette System	System I, II	Analog and Digital Peripherals, Inc. (ADPI)
Hard Copy Unit	4661	Tektronix
Interactive Digital Plotter	4662	Tektronix
Interactive Digital Plotter	4663	Tektronix
Printer	DP-8000	Anadex
Printer	2312	Bruel and Kjaer
Printer	P1 and S1	Centronics Data Computer Corp.
Printer	Microprinter-51	Centronics Data Computer Corp.
Printers	700 series	Centronics Data Computer Corp.
Printer	20-L	Gulton Industries, Inc.
Printer	2631	Hewlett-Packard
Printer	2467	Philips Test and Measuring, Inc.
Printer, 132 column	9871A	Hewlett-Packard
Printer drive unit	2465	Philips Test and Measuring Inc.
Printer, Spinwriter	5530-2	NEC
Printer, thermal, 20 col.	5150A	Hewlett-Packard
Plotter, multicolor	9872A	Hewlett-Packard
Plotter/Printer, Thermal	7245A	Hewlett-Packard
Recorder, 4 channel	3964A	Hewlett-Packard
Recorder, 8 channel	3968A	Hewlett-Packard
Recorder, data	4200/50	Philips Test and Measuring, Inc.
Recorder, mag. tape	1015AS	Dylon
Tape unit	4600	Information Development and Applications, Inc.
Tape unit, cartridge	9875	Hewlett-Packard
Terminal, data entry	3070A	Hewlett-Packard
Terminal, printing	2635	Hewlett-Packard
Translator, graphics	1350A	Hewlett-Packard
Video Hard Copy Unit	4632	Tektronix

STIMULUS INSTRUMENTS

Product	Model	Manufacturer
AC Sources	DDP	Elgar Corp.
AM/FM/0M Signal Gen.	460	Ailtech
Amplifier	RA3	Electronic Development Corp.
ASCII-Parallel Converter	59301A	Hewlett-Packard
D to A Converter	59303A	Hewlett-Packard
DC Standard	PCS1	Electronic Development Corp.
DC Source Synthesizer	800	Interstate Electronics Corp.
Digital Clock	59309A	Hewlett-Packard
Digital Voltage Source	6129C	Hewlett-Packard
Digital Voltage Source	6130C	Hewlett-Packard
Digital Voltage Source	6131C	Hewlett-Packard
Digital Voltage Source	6140A	Hewlett-Packard
Generator, data and time	R5432	Interface Technology
Generator, function	340	Dana/Exact Electronics
Generator, programmable pulse	8160A	Hewlett-Packard
Generator, function (BCD output to ICS 4880)	5500	Krohn-Hite Corp.
Generator, function	5900	Krohn-Hite Corp.
Generator, function	152	Wavetek Indiana, Inc.
Generator, pulse	1503	EH Research Laboratories
Generator, pulse	1504	EH Research Laboratories
Generator, pulse	154	Systron-Donner
Generators, programmable function	600 Series	Exact
Generator, logic pattern	8170A	Hewlett-Packard
Generators, low frequency	340/350 Series	Exact
Generator, signal	601A	Fluke
Generator, signal	6011A	Fluke
Generator, signal	TF2020	Marconi Inst.
Generator, signal	1702	Systron-Donner
Generator, signal	300D Series	Wavetek Indiana Corp.
Generator, signal	430A	Weinschel Engr. Inc.
Generator, signal	4310	Weinschel Engr. Inc.
Generator, serial data	8018A	Hewlett-Packard
Generator, sweep function	121	Exact
Generator, synthesized signal	360	Ailtech
Generator, synthesized	2230A	Adret Corp.
Generator, synthesized signal	8660A and C	Hewlett-Packard
Generator, synthesized signal	8671A	Hewlett-Packard
Generator, synthesized signal	8672A	Hewlett-Packard
Generator, synthesized signal	1618	Systron-Donner
Generator, sweep	610D	Weinschel Engr. Inc.
Generator, timing	59308A	Hewlett-Packard
Generator, timing	R5648	Interface Technology
Generator, waveform	605	Dana/Exact Electronics

STIMULUS INSTRUMENTS (Continued)

Product	Model	Manufacturer
Generator, waveform	158	Wavetek Indiana, Inc.
Generator, waveform	159	Wavetek Indiana, Inc.
Generator, waveform	175	Wavetek Indiana, Inc.
Generator, word	8016A	Hewlett-Packard
IEEE-488 Translator	1120A	Fluke
Multiprogrammer	6940B	Hewlett-Packard
Oscillators, sweep	6600B/9514/9515	Ailtech
Oscillators, sweep	6600B/6608EDA	Ailtech
Oscillator, sweep	8620	Hewlett-Packard
Oscillators (BCD Outputs to ICS 4880)	4141R, 4030R, 4031R	Krohn-Hite Corp.
Power Source	DPSD50	Systron-Donner
Power Supply	LF904	Lambda Electronics
Power Supply	6002A	Hewlett-Packard
Power Supply, controller	SN488	Kepko, Inc.
Processor, RF	4307	Weinschel Engr. Inc.
Source, prog. signal	8165A	Hewlett-Packard
Source, RF	4312	Weinschel Engr. Inc.
Source, signal	172A	Wavetek Indiana, Inc.
Synthesizer	4311B	Weinschel Engr. Inc.
Synthesizer	3310	Adret Corp.
Synthesizer	360	Ailtech
Synthesizer, frequency	801	Dana/Exact Electronics
Synthesizer, frequency	802	Dana/Exact Electronics
Synthesizer, frequency	3320B	Hewlett-Packard
Synthesizer, frequency	8671A	Hewlett-Packard
Synthesizer, frequency	811	Micro-Tel Corp.
Synthesizer, frequency	35	Real Time Systems
Synthesizer, frequency	312	Real Time Systems
Synthesizer, frequency	352	Real Time Systems
Synthesizer, function	3325A	Hewlett-Packard
Synthesizer, level generator	3335A	Hewlett-Packard
Synthesizer, sweeper, auto.	3330B	Hewlett-Packard
Synthesizer, time	5359A	Hewlett-Packard
Standard, volt/current	501J	Electronic Development Corp.

TESTING DEVICES

Product	Model	Manufacturer
Analyzer, bus system	59401A	Hewlett-Packard
Analyzer, DC	401	Autek Systems Corp.
Analyzer, logic	7D01	Tektronix
Analyzer, spectrum	4520	Unigon Industries, Inc.
Analyzer, waveform	502/503 505	Autek Systems Corp.
Automatic Logic Board Tester	TL-8	Trace Instruments
Calibrator, AC	5200A/5215A	Fluke
Calibrator, digital VM	5100A	Fluke
Calibrator, digital VM	5101A	Fluke
DC Voltage Calibrator	2701	Valhalla Scientific, Inc.
Digital Shaker Control System	SD 1009	Valhalla Scientific, Inc.
Display Formatter	DF2	Tektronix
Distortion Analyzer	6880	Krohn-Hite Corp.
Dual Coaxial Scanner	FX 30, FX 36	Autek Systems Corp.
FFT Signal Analyzer	SD345	Spectral Dynamics
For Computerized Signal Analysis	SD2001D	Spectral Dynamics
GPIB Signal Director with Modules 8601, 8603, 8604, 8605, 8606	8600	Jaycor
Intelligent Logic State Analyzer	P1532	Paratronics
Spectra Tester	SD210	Paratronics
Transmission Impairment Measuring Set (TIMS)	4942A	Hewlett-Packard
TIMS	4943A	Hewlett-Packard
TIMS	4944A	Hewlett-Packard

APPENDIX B

Bibliography

The following list of journal articles, conference papers, and books is a reference for those who seek additional information about the GPIB and related interface subjects.

Abenaim, D. (Electronic Instrumentation Div., GenRad Inc., Concord, MA), "Impact of Microprocessors on Design of Analog Instrumentation," Electro 76 Professional Program, Boston, May 11-14, 1976.

Ableidinger, A. (Tektronix, Inc., Beaverton, OR), "Unraveling the Mystery on the GPIB, " Tekscope 10 (2), 1978, pp. 3-6.

Allen, D. "A Minifloppy Interface," Byte, February 1978, pp. 114-125.

Baunach, S.C. (Tektronix Inc., Beaverton, OR), "Design Advantages and Limitations in Connecting Computational and Readout Equipment to the GPIB," 1976 WESCON Professional Program, Los Angeles, September 14-17, 1976.

Baunach, S.C. "An Example of an M6800-Based GPIB Interface," EDN, September 20, 1977.

Bond, J. "Computer-Controlled Instruments Challenge Designer's Ingenuity," EDN, March 5, 1974.

Burhans, R.W. "A Simpler Digital Cassette Tape Interface," Byte, October 1978, pp. 142-143.

Ciarcia, S. "No Power for Your Interfaces? Build a 5W DC to DC Converter," Byte, October 1978, pp. 26-31.

Ciarcia, S. "A Penny Pinching Address State Analyzer," Byte, February 1978, pp. 6-12.

Coates, T. "Interfacing to the Interface: Practical Considerations beyond the Scope of IEEE Standard 488," Session 3, WESCON 1975, September, 1975.

Dietrich, H.E. "Visualizing Interface Bus Activity," Hewlett-Packard Journal, January 1975, pp. 19-23.

Digital Design, "RS-449 Ousts RS-232," in Technology Trends, Digital Design, December 1977, p. 18.

Evans, J. and Merrill, H. "A Standard Interface Applied to Measurement Systems," in ISA Proc., 22nd Int. Instrument Symp., Vol. 3, 1976, pp. 17-28.

Everett, C.M. (Dana Laboratories Inc., Irvine, CA), "Design and Application Restraints in the Implementation of Smart Instrumentation," Electro 76 Professional Program, May 11-14, 1976.

Fisher, E.R. (Lawrence Livermore Laboratory, Livermore, CA), "The System Designer Looks at the GPIB," 1976 WESCON Professional Program, Los Angeles, September 14-17, 1976.

Fisher, P.D. and Welch, S.M. "Part 2: How to Build an Automated System Around a Programmable Calculator," Electronics, May 16, 1974.

Fluke, J.M., Jr. (John Fluke Manufacturing Co., Inc., Mountlake Terrace, WA), "System Considerations in Using the IEEE Digital Instrument Bus," WESCON 1975, Session 3.

Fluke, J.M., Jr. IEEE Standard 488-1975 Digital Interface for Programmable Instrumentation, Applications Bulletin, Fluke AB-36.

Fluke, J.M., Jr. "What Makes an Instrument Truly System Compatible?," EE/Systems Engineering Today, May 1973.

Freytag, H.H. and ten-Thiede, A.K. "Konzeption Programmierbarer Messgerate mit IEC-Standard-Interface" Frequenz, July 30, 1976, pp. 190-195.

Hewlett-Packard Co. Calculator Control of the 8660A/C Synthesized Signal Generator," Hewlett-Packard Publication AN 164-2.

Hewlett-Packard Co. "Recent Advances in Pulsed RF and Microwave Frequency Measurements," Hewlett-Packard Publication AN 173.

Hewlett-Packard Co. "Measuring Linearity of VCOs from 10 Hz to 23 GHz," Hewlett-Packard Publication AN 181-1.

Hewlett-Packard Co. "Data Acquisition with the 5300B Measuring System," Hewlett-Packard Publication AN 181-2.

Hewlett-Packard Co. "Configuration of a 2-18.6 GHz Synthesized Frequency Source Using the 8620C Sweep Oscillator," Hewlett-Packard Publication AN 187-2.

Hewlett-Packard Co. "Three HP-IB Configurations for Making Microwave Scalar Measurements," Hewlett-Packard Publication AN 187-3.

Hewlett-Packard Co. "Calculator Control of the 8620C Sweep Oscillator Using the HP-IB," Hewlett-Packard Publication AN 187-5.

Hewlett-Packard Co. "Automatic Measurements Using the HP 436A Microwave Power Meter," Hewlett-Packard Publication AN 196.

Hewlett-Packard Co. "Routine OA Measurements of Precision Resistors," Hewlett-Packard Publication AN 201-1.

Hewlett-Packard Co. "Measuring Differential Non-linearity of a Voltage-Controlled Oscillator," Hewlett-Packard Publication AN 201-2.

Hewlett-Packard Co. "A Multiple Station Electronic Test System," Hewlett-Packard Publication AN 201-3.

Hewlett-Packard Co. "Performance Evaluation of HP-IB Using RTE Operating Systems," Hewlett-Packard Publication AN 201-4.

Hewlett-Packard Co. "Measuring Wide-Band Noise with the HP 3045A Automatic Spectrum Analyzer," Hewlett-Packard Publication AN 201-6.

Hewlett-Packard Co. "HP 1000/HP-IB: High Performance Software for the HP 3455/3495A Subsystem," Hewlett-Packard Publication AN 201-7.

Hewlett-Packard Co. "Applications and Performance of the 8671A/8672A Microwave Synthesizers," Hewlett-Packard Publication AN 218-1.

Hewlett-Packard Co. "Obtaining Millihertz Resolution from the 8671A and 8672A," Hewlett-Packard Publication AN 218-2.

Hewlett-Packard Co. "A 1 MHz-18 GHz Signal Generator with 1, 2 or 3 Hz Resolution," Hewlett-Packard Publication AN 218-3.

Hewlett-Packard Co. "Semi-Automatic Measurements Using the 8410B Microwave Network Analyzer and the 9825A Desktop Computer," Hewlett-Packard Publication AN 221.

Hewlett-Packard Co. "Signal Conditioning: HP 22914A Breadboard Card," Hewlett-Packard Publication AN 224-2.

Hewlett-Packard Co. "HP-IB Power Supply Interface Guide," Hewlett-Packard Publication AN 250-1.

Hewlett-Packard Co. "An Example of Automatic Measurement of Conducted EMI with the HP 8568A Spectrum Analyzer," Hewlett-Packard Publication AN 270-1.

Hewlett-Packard Co. "Adding Soft Copy Graphics to 9825A Based HP-IB Systems Using the Model 1350A Graphic Translator," Hewlett-Packard Publication AN 271-1.

Hewlett-Packard Co. "Monitoring the IEEE-488 Bus with the 1602A Logic State Analyzer," Hewlett-Packard Publication AN 280-2.

Hewlett-Packard Co. "The 1602 Logic State Analyzer as an Automatic Test Instrument," Hewlett-Packard Publication AN 280-3.

Hewlett-Packard Co. "Making Measurements on Wide Buses with HP's Model 1602A Logic State Analyzers," Hewlett-Packard Publication AN 280-4.

Hewlett-Packard Co. "8662A RF Synthesized Signal Generator Performance, Familiarization, and Applications," Hewlett-Packard Publication AN 283-1.

Hewlett-Packard Co. "Applications and Operation of the 8901A Modulation Analyzer," Hewlett-Packard Publication AN 286-1.

Hewlett-Packard Co. "Waveform Analysis Using the 5328A Universal Frequency Counter," Hewlett-Packard Publication AN 287-1.

Hewlett-Packard Co. "Hewlett-Packard Interface Bus User's Guide," Publication 9830A, 1974.

Hewlett-Packard Co. "HP-IB, Improving Measurements in Engineering and Manufacturing — A Collection of Useful Technical Information," Hewlett-Packard Publication #5952-0058.

Hordeski, M. "Balancing μP-Interface Tradeoffs," Digital Design, April 1977, pp. 66-72.

Huang, I.Y. "An Interconnection Standard," Electronic Engineering Times, September 27, 1976, p. 14.

IEEE. IEEE Standard Digital Interface for Programmable Instrumentation (The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y., April 4, 1975), IEEE Std 488-1975-ANSI MC 1.1-1975.

IEEE. IEEE Standard Digital Interface for Programmable Instrumentation (The Institute of Electrical and Electronics Engineers, Inc., New York, N.Y. November 30, 1978), IEEE Std 488-1978 (Revision of ANSI/IEEE Std 488-1975).

Kimball, J. (Tektronix, Inc., Beaverton, OR), "The IEEE 488 Bus — Going Your Way?," Tekscope 10 (2), 1978, pp. 7-10.

Klaus, J. "Wie Fonktioniert der IEC-Bus?" Electronik, Heft 4, 1975, pp. 72-78, and Heft 5, pp. 73-78.

Klein, P.E. Wilhelmy. "Der IEC-Bus," Elektronik, Vol. 10, October 1977, pp. 63-74.

Knoblock, D.E., Loughry, D.C., and Vissers, C.A. "Insight into Interfacing," IEEE Spectrum, May 1975, pp. 50-57.

Knoblock, D.E. "Identifying, Understanding, and Selecting Among the Capabilities Provided by IEEE Standard 488," WESCON 1975, Session 3.

Laengrich, N.L. (Dana Laboratories Inc., Irvine, CA), "The IEEE Std 488/1975 — General-purpose Means of Providing Measurement and Stimulus Instrument Communication," 1976 WESCON Professional Program, Los Angeles, September 14-17, 1976.

Laroff, G.P. (Tektronix, Inc., Boston, MA), "Tektronix® GPIB Applications Support," Publication #006-2519-00, 1976.

Lee, R.C. "Microprocessor Implementation of a Measurement Instrument and Its Interface," WESCON 1975, Session 3, September 1975.

Levine, S.T. "Instrument Interfacing Can Be Done — But It Takes a Bit of Doing," Electronic Design 24, November 22, 1973.

Lipovac, V. "Design an IEEE-488 Bus into an FPLA and Speed System Operation. An Extra Benefit: The Technique Allows You to Interface Any Number of Programmable Instruments," Electronic Design 24, November 22, 1977, pp. 104-111.

Loughry, D.C. "The Hewlett-Packard Interface Bus: Current Perspectives," Hewlett-Packard Journal, January 1975, pp. 2-4.

Loughry, D.C. and Allen, M.S. "IEEE Standard 488 and Microprocessor Synergism," Proceedings of the IEEE, Vol. 66, No. 2, February 1978, pp. 162-172.

Loughry, D.C. "Digital Bus Simplifies Instrument System Communication," EDN, September 1972.

Loughry, D.C. "A Common Digital Interface for Programmable Instruments: The Evolution of a System," Hewlett-Packard Journal, October 1972.

Loughry, D.C. "Instrument Communications: A New Interface System," INTERCON 1973.

Loughry, D.C. "What Makes a Good Interface?", IEEE Spectrum, November 1974, pp. 52-57.

Loughry, D.C. "Interface Standard Simplifies Instrumentation Systems," Canadian Electronics Engineering, November 1975.

Loughry, D.C. "A New Instrument Interface: Needs and Progress toward a Standard," ISA Transactions 14 (3) 1975, pp. 225-230.

Marshall, M. "Intelligent Instruments — They Expand Measurement Capabilities, yet Simplify the Task," EDN, April 20, 1977, pp. 54-65.

Motorola, (Motorola Semiconductor Products Inc., Phoenix, AZ), "Getting Aboard the 488-1975 Bus," Publication 11338, October 1977.

Muche, C.E. and Lin, C.S. "IEEE 488 Applied to Radar and Scientific Systems," presented at ELECTRO 76 Conference, Session 29, May 1976.

Myles, R. "IEC Interface: A Microprocessor Based Implementation," IMEKO VII Congress, London, England, Vol. 3, May 1976.

Nadig, H.J. "A Quiet, HP-IB Compatible Printer that Listens to Both ASCII and BCD," Hewlett-Packard Journal, January 1975, pp. 14-16.

Nelson, G.E. and Ricci, D.W. "A Practical Interface System for Electronic Instruments," Hewlett-Packard Journal, October 1972.

Nelson, J.E. (Hewlett-Packard Co., Loveland, CO), "Trends in Instrumentation," Electro 76 Professional Program, Boston, May 11- 14, 1976.

Nelson, J.E. "Exploding Technology is Shaping the Course of Tomorrow's Instrumentation," Instruments and Control Systems, June 1976, pp. 43-46.

Nelson, J.E. "Trends in Instrumentation," presented at WESCON 76 Conference, Session 24, 1976.

NIM Committee, National Bureau of Standards. CAMAC, A Modular Instrumentation System for Data Handling (United States Atomic Energy Commission, Washington, D.C. 20545), Document Number TID-25875, July 1972.

Pannach, A. "Interface Entwicklung für den IEC-Bus," Elektronik, 1975, pp. 61-64.

Pine, K. "Troubleshooting with Logic Analyzers," Digital Design, October 1977, pp. 64-75.

Rhine, G. "IEEE 488 Applied to a Graphic Terminal," presented at ELECTRO 76 Conference, Session 29, 1976.

Ricci, D.W. and Nelson, G.E. "Standard Instrument Interface Simplifies System Design," Electronics, November 14, 1974, pp. 95-106.

Ricci, D.W. and Stone, P.S. "Putting Together Instrumentation Systems at Minimum Cost," Hewlett-Packard Journal, January 1975, pp. 5-11.

Richert, U. "IEC-Bus Interface für Prozebrechner," Elektronik, Heft 12, 1976, pp. 58-62.

Richter, M. "Das Zustandsdiagramm und seine Anwendung beim IEC-Bus," Elektronik, Heft 2, February 1977, pp. 55-58.

Santoni, A. "What's Wrong with 488? Not Much, but..." Electronic Design 24, November 22, 1977, pp. 48-51.

Schultz, S.E. and Trimble, C.R. "Filling in the Gaps — Modular Accessories for Instrument Systems," Hewlett-Packard Journal, January 1975, pp. 12-13.

Silvernale, L.P. (Wavetek, San Diego, CA)," GPIB Command Structures," 1976 WESCON Professional Program, Los Angeles, September 14-17, 1976.

Smolin, Dr. M., Graves, D., Winter, K., and Schwartz, M. "Unified Buses Make the Peripheral LC/μC Connection," Part 1, Digital Design, May 1978, pp. 34-44.

Smolin, Dr. M., Graves, D., Winter, K., and Schwartz, M. "Unified Buses Make the Peripheral IC/μC Connection," Part 1, Digital Design, July 1978, pp. 86-96.

Socolovsky, A. "The Happy State of a Standard," EE/Systems Engineering Today, May 1973.

Stein, P. "Small World — The 488 Interface Bust," Computer Decisions, p. 16.

Sugarman, R. "IEEE Interface Beats Microprocessor as Most-Wanted Signal Generator Feature," Electronic Engineering Times, February 1976.

Trifari, J. "Bus Standard Brings New Power to Bench-top Instrumentation," Electronic Products, July 1976, pp. 31-34.

Wiewald, J. and West, B. (E-H Research Laboratories Inc., Oakland, CA) "Recent Advances in Microprocessor-Based Test and Measuring Equipment," Electro 76 Professional Program, May 11-14, 1976.

Wiseman, D. "Getting on the Bus," Electronics Test, Benwill Publishing Corp., June 1979.

Wolpert, D.L. "Multifunction Scanner for Calculator-Based Data Acquisition Systems," Hewlett-Packard Journal, January 1975, p. 17.

Yencharis, L. "Expansion of IEEE Interface to Automatic Test Equipment Cited as Main Improvements: Microprocessor still not a Major Addition," Electronic Engineering Times, June 27, 1977, pp. 16-19.

APPENDIX C

IEEE Std Definitions

The following mnemonics and their definitions have been compiled from the pages of the publication "IEEE Std 488-1978."

Most mnemonics are capitalized, while a few appear as lower-case letters. Capitalized mnemonics represent interface states and remote messages sent and received; lower-case mnemonics indicate local messages received (by interface functions). Local messages sent (to interface functions) are not defined.

The entire document is available from:

IEEE SERVICE CENTER
445 Hoes Lane
Piscataway, New Jersey 08854
U.S.A.
(201) 981-0060

AC	Addressed command
ACDS	Accept data state
ACG	Addressed command group
ACRS	Acceptor ready state
AD	Addressed

AH	Acceptor handshake
AH1	Complete capability
AH0	No capability
AIDS	Acceptor idle state
ANRS	Acceptor not ready state
ANSI	American National Standard's Institute
APRS	Affirmative poll response state
ATN	Attention
AWNS	Acceptor wait for new cycle state
C	Controller
CACS	Controller addressed state
CADS	Controller idle state
CAWS	Controller active wait state
CIDS	Controller idle state
CPPS	Controller parallel poll state
CPWS	Controller parallel poll wait state
CSBS	Controller standby state
CSNS	Controller service not requested state
CSRS	Controller service requested state
CSWS	Controller synchronous wait state
CTRS	Controller transfer state
DAB	Data byte
DAC	Data accepted
DAV	Data valid
DC	Device clear
DCAS	Device clear active state
DCIS	Device clear idle state
DCL	Device clear
DD	Device dependent
DIO	Data input
DT	Device trigger
DTAS	Device trigger active state
DTIS	Device trigger state
END	End
EOI	End or identify
EOS	End of string
F	Active false
(F)	Passive false
GET	Group execute trigger
GTL	Go to local
gts	Go to standby
IDY	Identify

IFC	Interface clear
ist	Individual status
L or LE	Listener or extended listener
LACS	Listener active state
LADS	Listener addressed state
LAG	Listen address group
LIDS	Listener idle state
LLO	Local lockout
LOCS	Local state
lon	Listen only
LPAS	Listener primary addressed state
[lpe]	Local poll enable
LPIS	Listener primary idle state
ltn	Listen
lun	Local unlisten
LWLS	Local with lockout state
M	Multiline
MLA or [MLA]	My listen address
MSA or [MSA]	My secondary address
MTA or [MTA]	My talk address
nba	New byte available
NDAC	Not data accepted
NPRS	Negative poll response state
NRFD	Not ready for data
NUL	Null byte
OSA	Other secondary address
OTA	Other talk address
PACS	Parallel poll addressed to configure state
PCG	Primary command group
POFS	Power off
pon	Power on
PP	Parallel poll
PPAS	Parallel poll active state
PPC	Parallel poll configure
PPD or [PPD]	Parallel poll disable
PPE or [PPE]	Parallel poll enable
PPIS	Parallel poll idle state
PPR	Parallel poll response
PPSS	Parallel poll standby state
PPU	Parallel poll unconfigure
PUCS	Parallel poll unaddressed to configure state
rdy	Ready (for next message)
REMS	Remote state

REN	Remote enable
RFD	Ready for data
RL	Remote local
rpp	Request parallel poll
RQS	Request service
rsc	Request system control
rsv	Request service
rtl	Return to local
RWLS	Remote with lockout state
SACS	System control active state
SCG	Secondary command group
SDC or [SDC]	Selected device clear
SDYS	Source delay state
SE	Secondary
SGNS	Source generate state
SH	Source handshake
SIAS	System central interface clear active state
sic	Send interface clear
SIDS	Source idle state
SIIS	System control interface clear idle state
SINS	System control interface clear not active state
SIWS	Source idle wait state
SNAS	System control not active state
SPAS	Serial poll active state
SPD	Serial poll disable
SPE	Serial poll enable
SPIS	Serial poll idle state
SPMS	Serial poll mode state
SR	Service request
SRAS	System Control remote enable active state
sre	Send remote enable
SRIS	System control remote enable idle state
SRNS	System control remote enable not active state
SRQ	Service request
SRQS	Service request state
ST	Status
STB	Status byte
STRS	Source transfer state
SWNS	Source wait for new cycle state
T or (TE)	Talker or extended talker
T	Active true
(T)	Passive true
TACS	Talker active state
TADS	Talker addressed state

TAG	Talk address group
tca	Take control asynchronously
tcs	Take control synchronously
TCT or [TCT]	Take control
TIDS	Talker idle state
ton	Talk only
TPAS	Talker primary addressed state
TPIS	Talker primary idle state
U	Uniline message
UC	Universal command
UCG	Universal command group
UNL	Unlisten
UNT	Untalk

APPENDIX D

ASCII Definitions

This appendix lists American Standard Code for Information Interchange (ASCII) abbreviations and their definitions.

ACK	Acknowledge
BS	Backspace
BEL	Bell (audible or attention signal)
CAN	Cancel
CR	Carriage Return
DLE	Data Link Escape
≡	Delete
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
EM	End of Medium
ETX	End of Text
EOT	End of Transmission
ETB	End of Transmission Block
ENQ	Enquiry
ESC	Escape
FS	File Separator
FF	Form Feed
GS	Group Separator
HT	Horizontal Tabulation (punched card skip)
LF	Line Feed
NAK	Negative Acknowledge
NUL	Null
RS	Record Separator
SI	Shift In
SO	Shift Out
SOH	Start of Heading
STX	Start of Test
SUB	Substitute
SYN	Synchronous Idle
US	Unit Separator
VT	Vertical Tabulation

APPENDIX E

IEEE Bus Handshake Routine

The following article is by John A. Cooke, University of Edinburgh, and is reprinted from the Commodore PET User's Club Newsletter, England. It is used with the permission of John A. Cooke and Commodore Systems Division, London, England.

This article originally appeared in Issue No. 4 of Commodore PET User's Club Newsletter, published by Commodore Systems Division in the UK.

While the published routine has proved satisfactory for the particular purpose for which it was developed at the Department of Astronomy, Royal Observatory, Edinburgh, neither Commodore Systems Division nor Mr. John A. Cooke can be held responsible for its validity in any particular user application.

IEEE BUS HANDSHAKE ROUTINE IN MACHINE LANGUAGE

To use the IEEE-488 bus on the PET at maximum speed it is necessary to use machine language rather than BASIC 'INPUT' and 'PRINT'. The routine given here has been used with an HP3437A systems voltmeter to reach data transfer speeds of over 5000 bytes per second, corresponding to 2500 voltage readings in 2-byte packed binary format or 625 readings in 8-byte ASCII format. The best speed attained in BASIC is 75 readings per second transferred as character strings.

The IEEE Bus

Details of the IEEE-488 bus are given in PET documentation, but some clarification of the register addresses is helpful. These are:

Hex	Decimal	Bits	IEEE	Direction
E820	59424	0-7	DIO 1-8	from bus
E822	59426	0-7	DIO 1-8	to bus; 'PET' controlled
E821	59425	3	NDAC	'PET' controlled
E823	59427	3	DAV	'PET' controlled
E840	59456	0	NDAC	from bus
		1	NRFD	'PET' controlled
		2	ATN	'PET' controlled
		6	NRFD	from bus
		7	DAV	from bus

Note that on the IEEE bus, 'high' is logic false and 'low' is logic true; and that the data bus must be left with all bits 'high' when PET has finished to avoid confusion of data put to the bus by other devices.

The Program

The program controls a given number of data transfers, each of 8 bytes, from the HP3437A to the PET. Each one is preceded by a trigger (GET — group execute trigger) on the IEEE bus, and the HP3437A must be correctly addressed as a 'talker' or a 'listener' at all times by sending MTA (my talk address) or MLA (my listen address) before transfers as appropriate. The sending of messages (GET, MTA, MLA, etc.) or data is controlled by the ATN line; ATN is true when messages are being sent.

The program and returned data are held in the top 2K of memory; this is hidden from BASIC using POKE 134,255: POKE 135,23 as the first line of the BASIC control program. The number of readings required is POKE'd into 6400 (decimal), then control passed to the machine language program by SYS(6144). The data bytes coming in on the IEEE bus are stored in locations 6401 (decimal) onwards; these are PEEKed out on return to BASIC, and converted into numbers using the function VAL. As the index register is used for counting, only 256 bytes can be transferred using this program, but it would be easy to modify the program to perform more transfers.

Disassembled listings with comments and a separate listing (for ease of copying into BASIC DATA statements!) are given.

This program was prepared using a machine language handler written by the author, and the listings produced by this handler and by a modified version of the 'disassemble' part of the PETSOFT ASSEMBLER 'EXEC' program.

IEEE bus handshake routine — main program

```
1800  A200     LDX  #00     prepare index register
1802  A9FB     LDA  #FB     set ATN low
1804  2D40E8   AND  E840
1807  8D40E8   STA  E840
180A  A928     LDA  #28     MLA (28 for this device)
180C  8501     STA  01
180E  208018   JSR  1880    handshake into bus
1811  A908     LDA  #08     GET
1813  8501     STA  01
1815  208018   JSR  1880    handshake
1818  A948     LDA  #48     MTA
181A  8501     STA  01
181C  208018   JSR  1880    handshake
181F  A9FD     LDA  #FD     set NRFD low (ready to receive data)
1821  2D40E8   AND  E840
1824  8D40E8   STA  E840
1827  A9F7     LDA  #F7     and NDAC low also
1829  2D21E8   AND  E821
182C  8D21E8   STA  E821
182F  A904     LDA  #04     set ATN high
1831  0D40E8   ORA  E840
1834  8D40E8   STA  E840
1837  A008     LDY  #08     ready to count 8 bytes
1839  20B018   JSR  18B0    handshake data from bus
183C  A502     LDA  02      result to A
183E  9D0119   STA  1901,X  store in 1901+X
1841  E8       INX
1842  88       DEY
1843  D0F4     BNE  1839    jump if Y not zero
1845  A9FB     LDA  #FB     set ATN low
1847  2D40E8   AND  E840
184A  8D40E8   STA  E840
184D  A902     LDA  #02     set NRFD high
184F  0D40E8   ORA  E840
1852  8D40E8   STA  E840
1855  A908     LDA  #08     set NDAC high
1857  0D21E8   ORA  E821
185A  8D21E8   STA  E821
185D  A95F     LDA  #5F     UNT
185F  8501     STA  01
1861  208018   JSR  1880    handshake to bus
1864  A904     LDA  #04     set ATN high
1866  0D40E8   ORA  E840
```

IEEE bus handshake routine — main program (continued)

```
1869  8D40E8   STA E840
186C  CE0019   DEC 1900   decrease counter
186F  D091     BNE 1802   jump if not zero
1871  60       RTS        return to BASIC program
```

subroutine to handle handshake into bus

```
1880  AD40E8   LDA E840   NRFD ?
1883  2940     AND #40
1885  F0F9     BEQ 1880   jump back if not ready
1887  A501     LDA 01     ready: get data byte
1889  49FF     EOR #FF    complement it
188B  8D22E8   STA E822   send to bus
188E  A9F7     LDA #F7    set DAV low
1890  2D23E8   AND E823
1893  8D23E8   STA E823
1896  AD40E8   LDA E840   NDAC ?
1899  2901     AND #01
189B  F0F9     BEQ 1896   jump back if not accepted
189D  A908     LDA #08    accepted; set DAV high
189F  0D23E8   ORA E823
18A2  8D23E8   STA E823
18A5  A9FF     LDA #FF    25510 into bus
18A7  8D22E8   STA E822
18AA  60       RTS        return to main
```

subroutine to handle handshake from bus

```
18B0  A902     LDA #02    set NRFD high
18B2  0D40E8   ORA E840
1885  8D40E8   STA E840
18B8  AD40E8   LDA E840   DAV ?
18BB  2980     AND #80
18BD  D0F9     BNE 18B8   jump back if not valid
18BF  AD20E8   LDA E820   get data byte from bus
18C2  49FF     EOR #FF    complement
18C4  8502     STA 02     store in $ 0002
18C6  A9FD     LDA #FD    set NRFD low
18C8  2D40E8   AND E840
18CB  8D40E8   STA E840
18CE  A908     LDA #08    set NDAC high
18D0  0D21E8   ORA E821
18D3  8D21E8   STA E821
18D6  AD40E8   LDA E840   DAV high ?
18D9  2980     AND #80
```

subroutine to handle handshake from bus (continued)

```
18DB F0F9      BEQ 18D6   jump back if not
18DD A9F7      LDA #F7    set NDAC low
18DF 2D21E8    AND E821
18E2 8D21E8    STA E821
18E5 A9FF      LDA #FF    25510 into bus
18E7 8D22E8    STA E822
18EA 60        RTS        return to main
```

IEEE bus handshake routine listing

```
1800  A2 00 A9 FB 2D 40 E8 8D
1808  40 E8 A9 28 85 01 20 80
1810  18 A9 08 85 01 20 80 18
1818  A9 48 85 01 20 80 18 A9
1820  FD 2D 40 E8 8D 40 E8 A9
1828  F7 2D 21 E8 8D 21 E8 A9
1830  04 0D 40 E8 8D 40 E8 A0
1838  08 20 B0 18 A5 02 9D 01
1840  19 E8 88 D0 F4 A9 FB 2D
1848  40 E8 8D 40 E8 A9 02 0D
1850  40 E8 8D 40 E8 A9 08 0D
1858  21 E8 8D 21 E8 A9 5F 85
1860  01 20 80 18 A9 04 0D 40
1868  E8 8D 40 E8 CE 00 19 D0
1870  91 60 EA EA EA EA EA EA
1878  EA EA EA EA EA EA EA EA
1880  AD 40 E8 29 40 F0 F9 A5
1888  01 49 FF 8D 22 E8 A9 F7
1890  2D 23 E8 8D 23 E8 AD 40
1898  E8 29 01 F0 F9 A9 08 0D
18A0  23 E8 8D 23 E8 A9 FF 8D
18A8  22 E8 60 EA EA EA EA EA
18B0  A9 02 0D 40 E8 8D 40 E8
18B8  AD 40 E8 29 80 D0 F9 AD
18C0  20 E8 49 FF 85 02 A9 FD
18C8  2D 40 E8 8D 40 E8 A9 08
18D0  0D 21 E8 8D 21 E8 AD 40
18D8  E8 29 80 F0 F9 A9 F7 2D
18E0  21 E8 8D 21 E8 A9 FF 8D
18E8  22 E8 60
```

IEEE bus handshake routine listing (continued)

```
0001 data to go into bus
0002 data from bus

1900 counter for number of data transfers

1901 start of results area
```

APPENDIX F

Conversion Tables

This appendix contains the following reference tables:

- Hexadecimal-Decimal Integer Conversion
- Powers of Two
- Mathematical Constants
- Powers of Sixteen
- Powers of Ten

HEXADECIMAL-DECIMAL INTEGER CONVERSION

The table below provides for direct conversions between hexadecimal integers in the range 0–FFF and decimal integers in the range 0–4095. For conversion of larger integers, the table values may be added to the following figures:

Hexadecimal	Decimal	Hexadecimal	Decimal
01 000	4 096	20 000	131 072
02 000	8 192	30 000	196 608
03 000	12 288	40 000	262 144
04 000	16 384	50 000	327 680
05 000	20 480	60 000	393 216
06 000	24 576	70 000	458 752
07 000	28 672	80 000	524 288
08 000	32 768	90 000	589 824
09 000	36 864	A0 000	655 360
0A 000	40 960	B0 000	720 896
0B 000	45 056	C0 000	786 432
0C 000	49 152	D0 000	851 968
0D 000	53 248	E0 000	917 504
0E 000	57 344	F0 000	983 040
0F 000	61 440	100 000	1 048 576
10 000	65 536	200 000	2 097 152
11 000	69 632	300 000	3 145 728
12 000	73 728	400 000	4 194 304
13 000	77 824	500 000	5 242 880
14 000	81 920	600 000	6 291 456
15 000	86 016	700 000	7 340 032
16 000	90 112	800 000	8 388 608
17 000	94 208	900 000	9 437 184
18 000	98 304	A00 000	10 485 760
19 000	102 400	B00 000	11 534 336
1A 000	106 496	C00 000	12 582 912
1B 000	110 592	D00 000	13 631 488
1C 000	114 688	E00 000	14 680 064
1D 000	118 784	F00 000	15 728 640
1E 000	122 880	1 000 000	16 777 216
1F 000	126 976	2 000 000	33 554 432

Hexadecimal fractions may be converted to decimal fractions as follows:

1. Express the hexadecimal fraction as an integer times 16^{-n}, where n is the number of significant hexadecimal places to the right of the hexadecimal point.

$$0.\,CA9BF3_{16} = CA9\,BF3_{16} \times 16^{-6}$$

2. Find the decimal equivalent of the hexadecimal integer

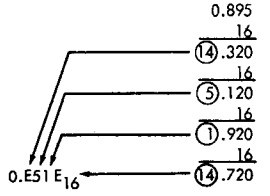
$$CA9\,BF3_{16} = 13\,278\,195_{10}$$

3. Multiply the decimal equivalent by 16^{-n}

$$\begin{array}{l} 13\,278\,195 \\ \times\ 596\,046\,448 \times 10^{-16} \\ \hline 0.791\,442\,096_{10} \end{array}$$

Decimal fractions may be converted to hexadecimal fractions by successively multiplying the decimal fraction by 16_{10}. After each multiplication, the integer portion is removed to form a hexadecimal fraction by building to the right of the hexadecimal point. However, since decimal arithmetic is used in this conversion, the integer portion of each product must be converted to hexadecimal numbers.

Example: Convert 0.895_{10} to its hexadecimal equivalent



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
01	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
02	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
03	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
04	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
05	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
06	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
07	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
08	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
09	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
11	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
12	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
13	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
14	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0332	0333	0334	0335
15	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
16	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
17	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
18	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
19	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
20	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
21	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
22	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
23	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
24	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
25	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
26	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
27	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
28	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
29	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
30	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
31	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
32	0800	0801	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
33	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
34	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
35	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
36	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
37	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
38	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
39	0912	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
40	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
41	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
42	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
43	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
44	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
45	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
46	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
47	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
48	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
49	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
50	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
51	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
52	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
53	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
54	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
55	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
56	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
57	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
58	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
59	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
60	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
61	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
62	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
63	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
64	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
65	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
66	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
67	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
68	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
69	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
70	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
71	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
72	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
73	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
74	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
75	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
76	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
77	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
78	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
79	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
7A	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
7B	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
7C	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
7D	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
7E	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
7F	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
80	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
81	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
82	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
83	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
84	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
85	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
86	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
87	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
88	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
89	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
90	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
91	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
92	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
93	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
94	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
95	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
96	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
97	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
98	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
99	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A0	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A1	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A2	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A3	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A4	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A5	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A6	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A7	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A8	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A9	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC	2752	2753	2754	2755	2756	2757	2758	2759	2760	2761	2762	2763	2764	2765	2766	2767
AD	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B0	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B1	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B2	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B3	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B4	2880	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895
B5	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B6	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B7	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B8	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B9	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
BB	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C0	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C1	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C2	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C3	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C4	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C5	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C6	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C7	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C8	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C9	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Continued)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D0	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D1	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D2	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D3	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D4	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D5	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D6	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D7	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D8	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D9	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
DD	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E0	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E1	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E2	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E3	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E4	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E5	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E6	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E7	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E8	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E9	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F0	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F1	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F2	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F3	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F4	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F5	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F6	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F7	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F8	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F9	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125

MATHEMATICAL CONSTANTS

Constant	Decimal Value	Hexadecimal Value
π	3.14159 26535 89793	3.243F 6A89
π^{-1}	0.31830 98861 83790	0.517C C1B7
$\sqrt{\pi}$	1.77245 38509 05516	1.C5BF 891C
$\ln \pi$	1.14472 98858 49400	1.250D 048F
e	2.71828 18284 59045	2.B7E1 5163
e^{-1}	0.36787 94411 71442	0.5E2D 58D9
$\sqrt{e}$	1.64872 12707 00128	1.A612 98E2
$\log_{10} e$	0.43429 44819 03252	0.6F2D EC55
$\log_2 e$	1.44269 50408 88963	1.7154 7653
γ	0.57721 56649 01533	0.93C4 67E4
$\ln \gamma$	-0.54953 93129 81645	-0.8CAE 9BC1
$\sqrt{2}$	1.41421 35623 73095	1.6A09 E668
$\ln 2$	0.69314 71805 59945	0.B172 17F8
$\log_{10} 2$	0.30102 99956 63981	0.4D10 4D42
$\sqrt{10}$	3.16227 76601 68379	3.29B8 075C
$\ln 10$	2.30258 40929 94046	2.4D75 3777

POWERS OF SIXTEEN$_{10}$

16^n	n	16^{-n}
1	0	$0.10000\ 00000\ 00000\ 00000 \times 10$
16	1	$0.62500\ 00000\ 00000\ 00000 \times 10^{-1}$
256	2	$0.39062\ 50000\ 00000\ 00000 \times 10^{-2}$
4 096	3	$0.24414\ 06250\ 00000\ 00000 \times 10^{-3}$
65 536	4	$0.15258\ 78906\ 25000\ 00000 \times 10^{-4}$
1 048 576	5	$0.95367\ 43164\ 06250\ 00000 \times 10^{-6}$
16 777 216	6	$0.59604\ 64477\ 53906\ 25000 \times 10^{-7}$
268 435 456	7	$0.37252\ 90298\ 46191\ 40625 \times 10^{-8}$
4 294 967 296	8	$0.23283\ 06436\ 53869\ 62891 \times 10^{-9}$
68 719 476 736	9	$0.14551\ 91522\ 83668\ 51807 \times 10^{-10}$
1 099 511 627 776	10	$0.90949\ 47017\ 72928\ 23792 \times 10^{-12}$
17 592 186 044 416	11	$0.56843\ 41886\ 08080\ 14870 \times 10^{-13}$
281 474 976 710 656	12	$0.35527\ 13678\ 80050\ 09294 \times 10^{-14}$
4 503 599 627 370 496	13	$0.22204\ 46049\ 25031\ 30808 \times 10^{-15}$
72 057 594 037 927 936	14	$0.13877\ 78780\ 78144\ 56755 \times 10^{-16}$
1 152 921 504 606 846 976	15	$0.86736\ 17379\ 88403\ 54721 \times 10^{-18}$

POWERS OF TEN$_{16}$

10^n	n	10^{-n}
1	0	1.0000 0000 0000 0000
A	1	0.1999 9999 9999 999A
64	2	$0.28F5\ C28F\ .5C28\ F5C3 \times 16^{-1}$
3E8	3	$0.4189\ 374B\ C6A7\ EF9E \times 16^{-2}$
2710	4	$0.68DB\ 8BAC\ 710C\ B296 \times 16^{-3}$
1 86A0	5	$0.A7C5\ AC47\ 1B47\ 8423 \times 16^{-4}$
F 4240	6	$0.10C6\ F7A0\ B5ED\ 8D37 \times 16^{-4}$
98 9680	7	$0.1AD7\ F29A\ BCAF\ 4858 \times 16^{-5}$
5F5 E100	8	$0.2AF3\ 1DC4\ 6118\ 73BF \times 16^{-6}$
3B9A CA00	9	$0.44B8\ 2FA0\ 9B5A\ 52CC \times 16^{-7}$
2 540B E400	10	$0.6DF3\ 7F67\ 5EF6\ EADF \times 16^{-8}$
17 4876 E800	11	$0.AFEB\ FF0B\ CB24\ AAFF \times 16^{-9}$
E8 D4A5 1000	12	$0.1197\ 9981\ 2DEA\ 1119 \times 16^{-9}$
918 4E72 A000	13	$0.1C25\ C268\ 4976\ 81C2 \times 16^{-10}$
5AF3 107A 4000	14	$0.2D09\ 370D\ 4257\ 3604 \times 16^{-11}$
3 8D7E A4C6 8000	15	$0.480E\ BE7B\ 9D58\ 566D \times 16^{-12}$
23 8652 6FC1 0000	16	$0.734A\ CA5F\ 6226\ F0AE \times 16^{-13}$
163 4578 5D8A 0000	17	$0.B877\ AA32\ 36A4\ B449 \times 16^{-14}$
DE0 B6B3 A764 0000	18	$0.1272\ 5DD1\ D243\ ABA1 \times 16^{-14}$
8AC7 2304 89E8 0000	19	$0.1D83\ C94F\ B6D2\ AC35 \times 16^{-15}$

APPENDIX G

ASCII-IEEE 488 HEX CODES

ASCII — IEEE 488 BUS MESSAGES (COMMANDS AND ADDRESSES) HEX CODES

MSD / LSD	0		1		2		3		4		5		6		7	
	ASCII	MSG	ASCII	MSG	ASCII	MSG[1]	ASCII	MSG[1]	ASCII	MSG[1]	ASCII	MSG[1]	ASCII	MSG	ASCII	MSG
0	NUL		DLE		SP	00	0	16	@	00	P	16	`		p	
1	SOH	GTL	DC1	LLO	!	01	1	17	A	01	Q	17	a		q	
2	STX		DC2		"	02	2	18	B	02	R	18	b		r	
3	ETX		DC3		#	03	3	19	C	03	S	19	c		s	
4	EOT	SDC	DC4	DCL	$	04	4	20	D	04	T	20	d		t	
5	ENQ	PPC	NAK	PPU	%	05	5	21	E	05	U	21	e		u	
6	ACK		SYN		&	06	6	22	F	06	V	22	f		v	
7	BEL		ETB		'	07	7	23	G	07	W	23	g		w	
8	BS	GET	CAN	SPE	(	08	8	24	H	08	X	24	h		x	
9	HT	TCT	EM	SPD	)	09	9	25	I	09	Y	25	i		y	
A	LF		SUB		*	10	:	26	J	10	Z	26	j		z	
B	VT		ESC		+	11	;	27	K	11	[	27	k		{	
C	FF		FS		,	12	<	28	L	12	\	28	l		|	
D	CR		GS		-	13	=	29	M	13	]	29	m		}	
E	SO		RS		.	14	>	30	N	14	^	30	n		~	
F	SI		US		/	15	?	UNL	O	15	_	UNT	o		DEL	

MEANING DEFINED BY PCG CODE

MEANING DEFINED BY PCG CODE

ADDRESSED COMMAND GROUP

UNIVERSAL COMMAND GROUP

LISTEN ADDRESS GROUP

TALK ADDRESS GROUP

SECONDARY COMMAND GROUP

PRIMARY COMMAND GROUP (PCG)

Notes:

1. Device Address messages shown in decimal
2. Message codes are:
   DCL — Devices Clear LLO — Local Lockout SDC — Selected Device Clear
   GET — Device Trigger PPC — Parallel Poll Configure SPD — Serial Poll Disable
   GTL — Go to Local PPU — Parallel Poll Unconfigure SPE — Serial Poll Enable
3. ATN off, Bus data is ASCII; ATN on, Bus data is an IEEE MSG.

Index

Eugene R. Fisher is a senior design engineer at the University of California Lawrence Livermore Laboratory. He has 15 years design/application experience in digital logic circuits, mini- and microcomputer systems and, recently, the GPIB.

A leading proponent of microcomputer modular design techniques for laboratory automation and data acquisition systems, Mr. Fisher has designed and implemented a microcomputer floppy-disk system which has become a standard peripheral device on many microcomputers. His designs in other areas of microcomputer technology include process controls, automation of laboratory apparatus, vacuum systems, and other scientific instrumentation.

Mr. Fisher has made significant design contributions to commercial and industrial products, including the Commodore PET 2001. He recently became president of Com-plications, a California consulting firm specializing in applications of the PET computer.

Eugene Fisher is a faculty associate of Colorado State University, Department of Electrical Engineering. As a teacher, author, and lecturer, he has written numerous microcomputer and GPIB articles for journals and the IEEE proceedings, and has presented many of his tutorial papers in the United States and Europe.