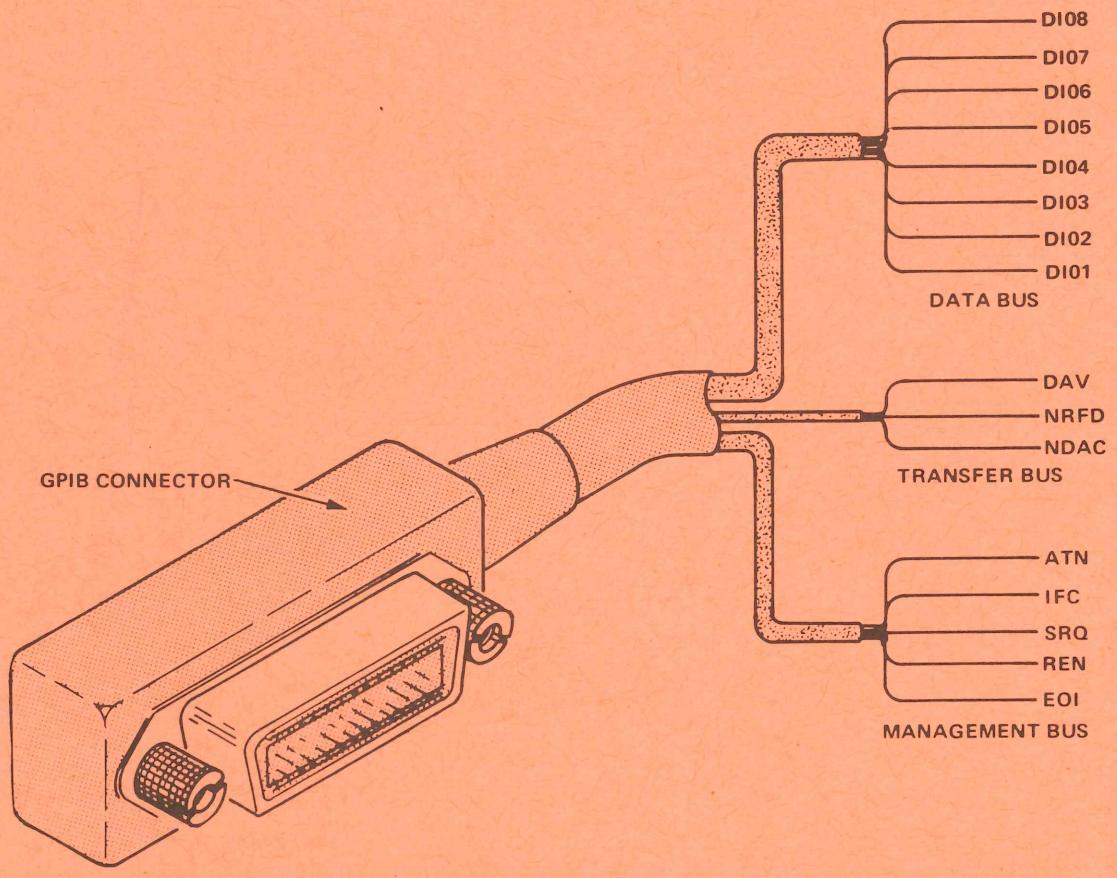
GPIB CONCEPTS



PREPARED BY
MARKETING TRAINING
MAINTENANCE GROUP

6/80

TABLE OF CONTENTS

GPIB SYSTEM CONCEPTS.....Pages 1 - 12

GPIB INTERFACE FUNCTIONS AND MESSAGES.....Pages 13 - 22

SOFTWARE MAKES THE SYSTEM WORK.....Pages 23 - 32

GPIB SYSTEM CONCEPTS

WHAT IS THE GPIB?

The GPIB is a digital interface that allows efficient communication between the components of an instrumentation system.

The primary purpose of the GPIB is to connect self-contained instruments to other instruments or devices. This means that the GPIB is an interface system independent of device functions.

There are four elements of the GPIB: mechanical, electrical, functional, and operational.

Of these four, only the last is device-dependent. Operational elements state the way in which an instrument reacts to a signal on the bus. These reactions are device-dependent characteristics and state the way in which the instruments use the GPIB via application software.

MECHANICAL ELEMENTS. The standard defines the mechanical elements: cables and connectors. Standardizing the connectors and cables ensures that GPIB-compatible instruments can be physically linked together with complete pin compatibility.

The connector has 24 pins, with 16 assigned to specific signals and eight to shields and grounds. Instruments on the bus may be arranged in either a linear or star configuration.

ELECTRICAL ELEMENTS. The voltage and current values required at the connector nodes for the GPIB are based on TTL technology (power source not to exceed +5.25V referenced to logic ground). The standard defines the logic levels as follows. Logical 1 is true state, low-voltage level ($\leq$+0.8V), signal line is asserted. Logical 0 is false state, high-voltage

level (≥+2.0V), signal line is not asserted. If open collector devices are used, the passive (undriven) state is a Logic 0 or high voltage level.

FUNCTIONAL ELEMENTS. The functional elements of the GPIB cover three areas:

Ten interface functions that define the use of specific signal lines so that an instrument can receive, process, and send messages (the ten interface functions - with their allowable subsets - provide an instrumentation system with complete communications and control capabilities). Interface functions are discussed in detail later.

The specific protocol by which the interface functions send and receive their limited set of messages.

The logical and timing relationships between allowable states for the interface singal lines.

INTERFACE FUNCTIONS

Not every instrument on the bus has all ten functions because only those functions important to a particular instrument's purpose need be implemented.

A TYPICAL SYSTEM ON THE GPIB

Figure 1 illustrates an example of the GPIB and the nomenclature for the 16 active signal lines. Only four instruments are shown, but the GPIB can support up to 15 instruments connected directly to the bus. However, more than 15 devices can be interfaced to a single bus if they do not connect directly to the bus but are interfaced through a primary device. Such a scheme can be used for programmable plug-ins housed in a mainframe where the mainframe is addressed with a primary address code and the plug-ins are addressed with a secondary address code.

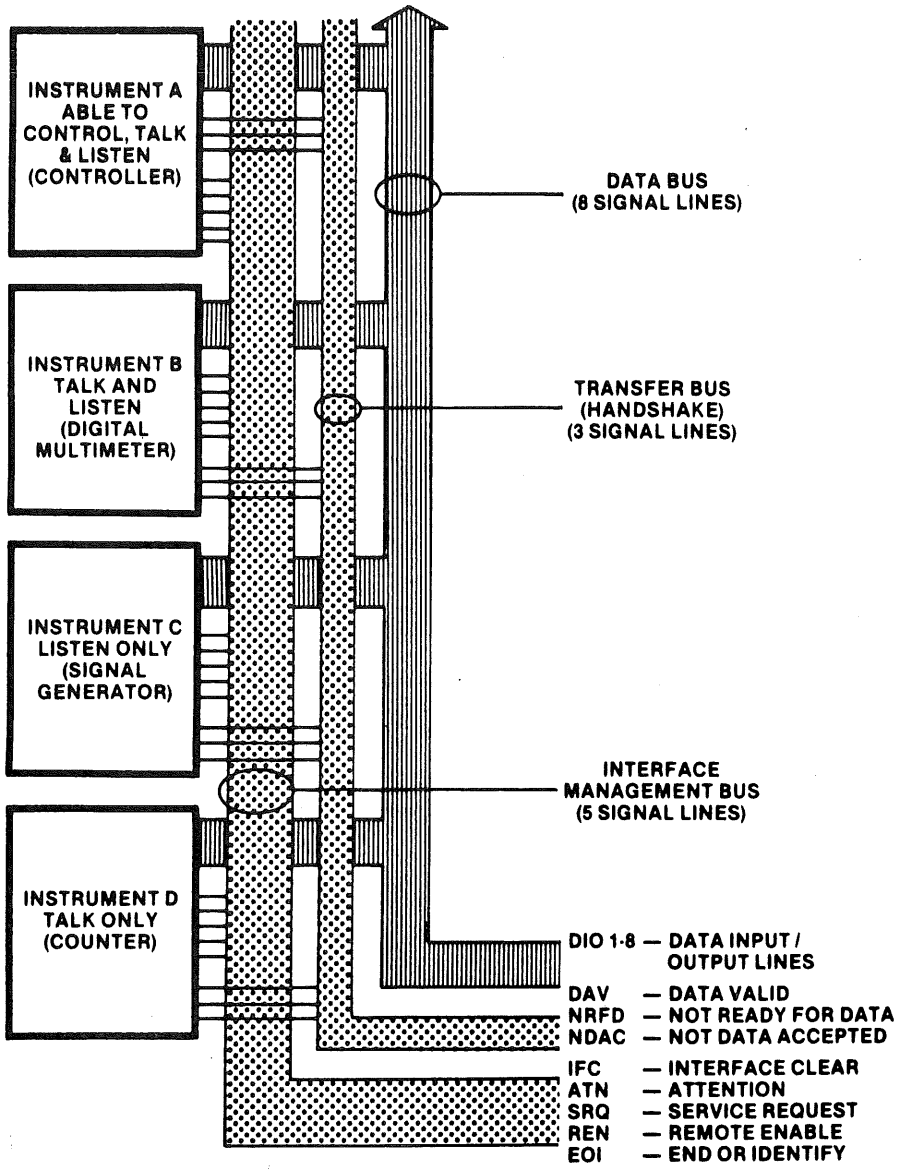


Figure 1. A typical system using the general purpose interface bus (GPIB).

The instruments connected to a single bus cannot be separated by more than 20 meters (total cable length) and at least one more than half the number of instruments must be in the power-on state. To maintain the electrical characteristics of the bus, a device load must be connected for each two meters of cable length. Although instruments are usually spaced no more than two meters apart, they can be separated further if the required number of device loads are lumped at any one point.

CONTROLLERS, TALKERS, AND LISTENERS

A talker is an instrument that can send data over the bus; a listener is an instrument that can accept data from the bus. No instrument can communicate until it is enabled to do so by the controller in charge of the bus.

A controller is an instrument that determines, by a software routine, which instrument will talk and which instruments will listen during any given time interval. The controller also has the ability to assign itself as a talker or listener whenever the program routine requires. In addition to designating the current talker and listeners for a particular communication sequence, the controller has the task of sending special codes and commands (called interface messages) to any or all of the instruments on the bus.

INTERFACE MESSAGES

The IEEE standard specifies that the interface messages be used to address and control instruments interfaced to the GPIB. Interface messages are sent and received only when the controller asserts the ATN bus line. The user can correlate interface message coding to the ISO 7-bit code by relating data bus lines DIO1 through DIO7 to bits 1 through 7, respectively.

Interface messages include the primary talk and listen addresses for instruments on the bus, addressed commands (only instruments previously addressed to listen respond to these commands), universal commands (all instruments, whether they have been addressed or not respond to these), secondary commands, Parallel Poll Enable (PPE) and Parallel Poll Disable (PPD). (Parallel Poll Enable means that after the controller configures the system for a parallel poll (PPC command), all instruments respond at the same time with status information on receipt of PPE.)

DEVICE-DEPENDENT MESSAGES

The IEEE 488-1975 does not specify coding of device-dependent messages,

messages that control the device's internal operating functions. After addressing (via interface messages) a talker and listener(s), the controller unasserts the ATN bus line. When ATN becomes false, any commonly-understood 8-bit binary code may be used to represent a device-dependent message.

The standard recommends that the alphanumeric codes associated with the numbers, symbols, and upper case characters (decimal 32 to decimal 94) in the ASCII Code Chart be used for device-dependent messages. One example of a device-dependent message is the ASCII character string

MODE V;2.5MV;FREQ 1E3

which may tell an instrument to set its front-panel controls to the voltage mode, with 2.5 millivolt output at a frequency of 1000Hz.

When 8-bit binary codes other than the ISO 7-bit code are used for device-dependent messages, the most significant bit must be on data line DIO8 (for bit 8).

To summarize the difference between interface and device-dependent messages, remember that any message sent or received when the ATN line is asserted (true) is an interface message. Any message (data bytes) sent or received when the ATN line is unasserted (false) is a device-dependent message.

GPIB SIGNAL LINE DEFINITIONS

Figure 1 shows the 16 signal lines of the GPIB functionally divided into three component busses: an eight-line data bus, a three-line transfer control (handshake) bus, and a five-line management bus.

THE DATA BUS. The data bus has eight bidirectional signal lines, DIO1 through DIO8. Information, in the form of data bytes, is transferred over this bus. A handshake sequence between an enabled talker and the enabled listeners transfers one data byte (eight bits) at a time. Data bytes in an interface or device-dependent message are sent and received in a byte-serial, bit-parallel fashion over the data bus.

Since the GPIB handshake sequence is an asynchronous operation, the data transfer rate is only as fast as the slowest instrument involved in a data byte transfer at any one time. A talker cannot place data bytes on the bus faster than any one listener can accept them.

Figure 2 illustrates the flow of data bytes when a typical controller sends ASCII data to an assigned listener on the bus. The first data byte, decimal 44, enables device 12 as a primary listener and the secondary address, decimal 108, enables a plug-in device as the final destination of the data to follow. The data is the two ASCII characters, A and B (decimal 65 and decimal 66).

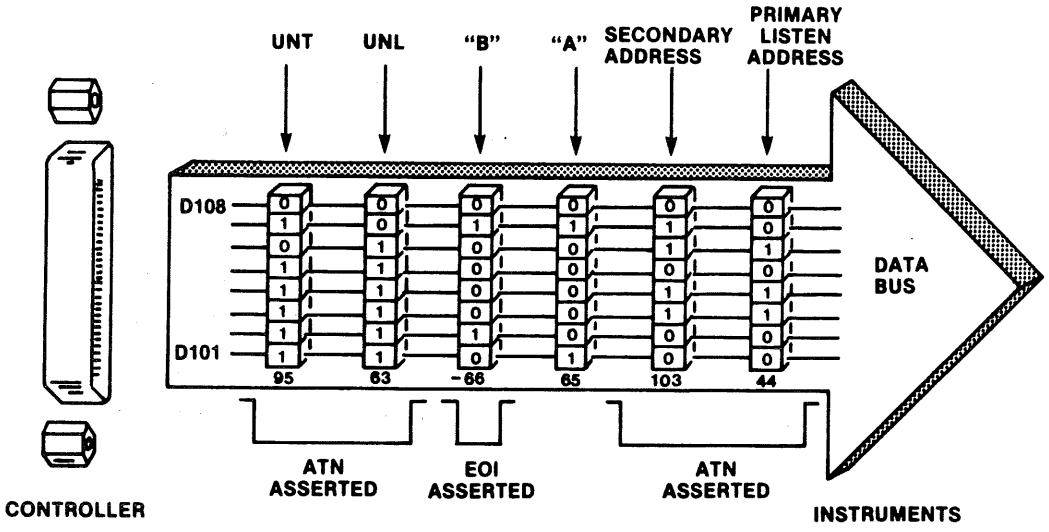


Figure 2. An example of data byte traffic on the GPIB.

The decimal value for B is specified as negative to activate the EOI line and signify the end of the device-dependent message. The controller activates the ATN line again and sends the universal unlisten (UNL) and untalk (UNT) commands to clear the bus. Six handshake cycles on the Transfer Bus are required to send the six data bytes.

THE TRANSFER BUS (HANDSHAKE). Each time a data byte is sent over the data bus, an enabled talker and all enabled listeners execute a handshake sequence via the transfer bus. Transfer-bus signal lines are defined below.

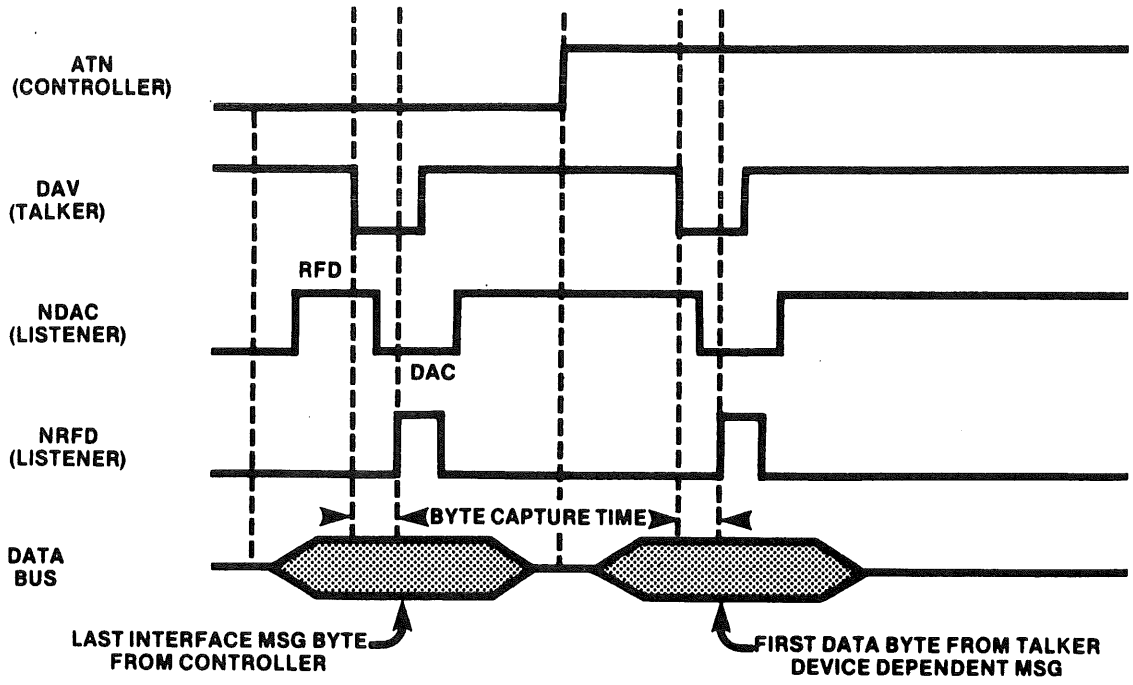


Figure 3. A typical handshake timing sequence (idealized). Byte capture time is dependent on the slowest instrument involved in the handshake.

Figure 3 illustrates the basic timing relationship between the three signals. The ATN line is shown to illustrate the controller's role in the process. A flow chart for the handshake sequence is shown in Figure 4.

NOT READY FOR DATA (NRFD). An asserted NRFD signal line indicates one or more assigned listeners are not ready to receive the next data byte from the talker. When all of the assigned listeners for a particular data byte transfer have released NRFD, the NRFD line becomes unasserted (high). The RFD message (Ready For Data) tells the talker it may place the next data byte on the data bus.

DATA VALID (DAV). The DAV signal line is asserted (low) by the talker after the talker places a data byte on the data bus. When asserted, DAV tells each assigned listener that a new data byte is on the data bus. The talker is inhibited from asserting DAV as long as any listener holds the NRFD signal line asserted.

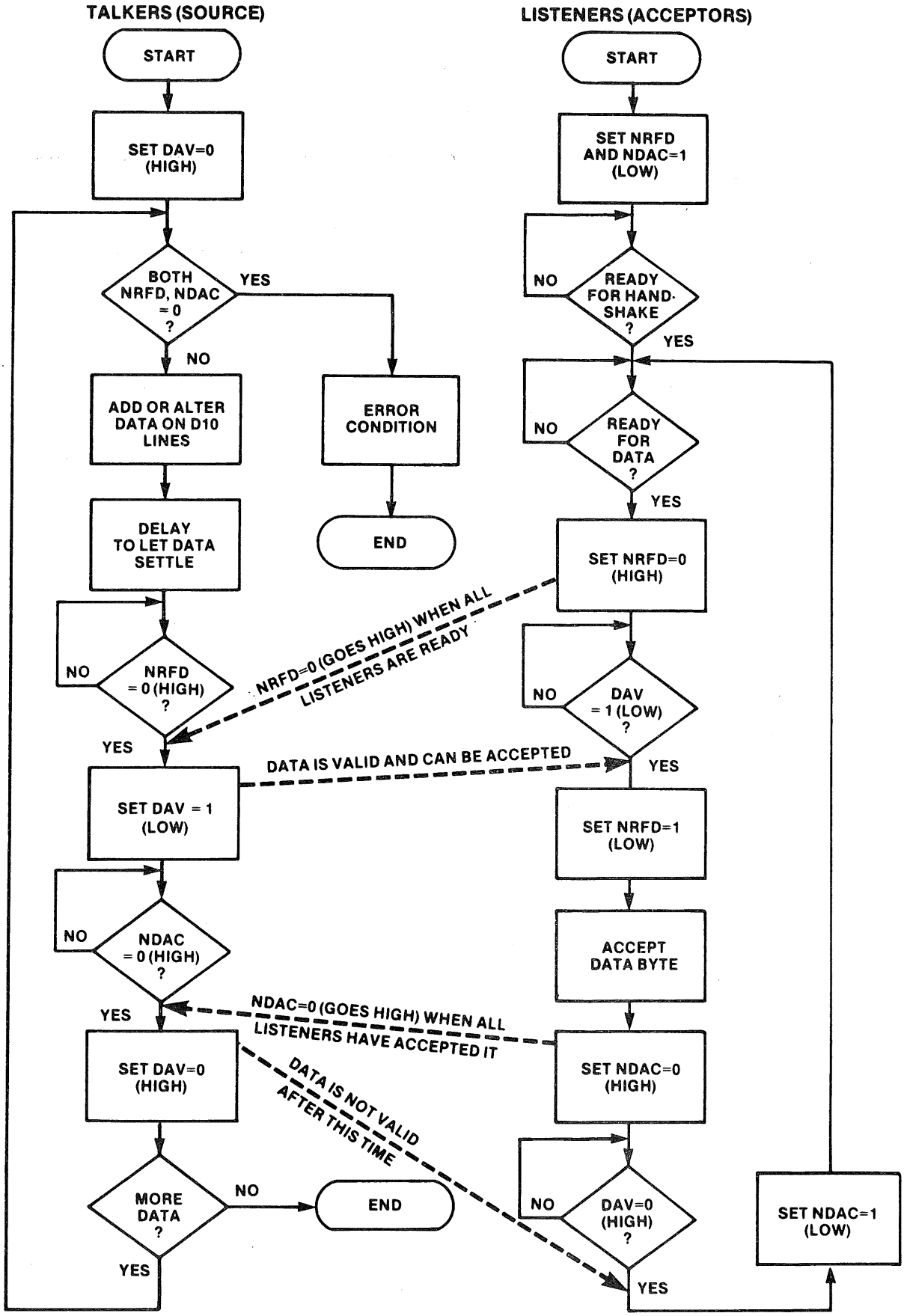


Figure 4. The handshake flow chart.

NOT DATA ACCEPTED (NDAC). Each assigned listener holds the NDAC signal line low-true (asserted) until the listener accepts the data byte currently on the data bus. When all assigned listeners accept the current data byte, the NDAC line becomes unasserted, telling the talker to remove the data byte from the bus. The DAC message (Data Accepted) tells the talker that all assigned listeners accepted the current data byte.

When one handshake cycle transfers one data byte, the listeners reset the NRFD line high and the NDAC line low before the talker asserts DAV for the next byte transfer. NDAC and NRFD both high at the same time is an invalid state on the bus.

THE MANAGEMENT BUS. The management bus is a group of five signal lines which are used to control the operation of the GPIB: IFC, ATN, SRQ, REN, and EOI.

INTERFACE CLEAR (IFC). The system controller asserts the IFC signal line to place all interface circuitry in a predetermined quiescent state which may or may not be the power-on state.

Only the system controller can generate this signal. IEEE 488-1975 specifies that only three interface messages (universal commands) be recognized while IFC is asserted: Device Clear (DCL), Local Lockout (LLO), and Parallel Poll Unconfigure (PPU).

ATTENTION (ATN). A controller asserts the ATN signal line when instruments connected to the bus are being enabled as talkers or listeners and for other interface control traffic. As long as the ATN signal line is asserted (ATN = 1), only instrument address codes and control messages are transferred over the data bus. With the ATN signal line unasserted, only those instruments enabled as a talker and listener(s) can transfer data. Only the controller can generate the ATN signal.

SERVICE REQUEST (SRQ). Any instrument connected to the bus can request the controller's attention by asserting the SRQ line. The controller responds by asserting ATN and executing a serial poll to determine which

instrument is requesting service. (An instrument requesting service identifies itself by asserting its DIO7 line after being addressed.) After the instrument requesting service is found, program control is transferred to a service routine for the instrument. When the service routine is completed, program control returns to the main program. When polled, the instrument requesting service unasserts the SRQ line.

REMOTE ENABLE (REN). The system controller asserts the REN signal line whenever the interface system operates under remote program control. Used with other control messages, the REN signal causes an instrument on the bus to select between two alternate sources of programming data. A remote-local interface function indicates to an instrument that the instrument will use either information input from the front-panel controls (Local) or corresponding information input from the interface (Remote).

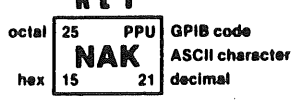
END OR IDENTIFY (EOI). A talker can use the EOI to indicate the end of a data-transfer sequence. The talker asserts the EOI signal line as the last byte of data is transmitted. In this case, EOI is essentially a ninth data line and must observe the same setup times as the DIO lines. When the controller is listening, it assumes that a data byte received is the last byte in the transmission (if the EOI signal line has been asserted). When the controller is talking, it may assert the EOI signal line as the last byte is transferred. The EOI signal is also asserted with the ATN signal if the controller conducts a parallel polling sequence. EOI is not used during serial polling.

ASCII & IEEE 488 (GPIB) CODE CHART

B7 B6 B5 BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0	20 DLE 16	40 SP 32	60 0 48	100 @ 64	120 P 80	140 ` 96	160 p 112
0 0 0 1	1 GTL SOH 1	21 LLO DC1 17	41 ! 33	61 1 49	101 A 65	121 Q 81	141 a 97	161 q 113
0 0 1 0	2 STX 2	22 DC2 18	42 " 34	62 2 50	102 B 66	122 R 82	142 b 98	162 r 114
0 0 1 1	3 ETX 3	23 DC3 19	43 # 35	63 3 51	103 C 67	123 S 83	143 c 99	163 s 115
0 1 0 0	4 SDC EOT 4	24 DCL DC4 20	44 $ 36	64 4 52	104 D 68	124 T 84	144 d 100	164 t 116
0 1 0 1	5 PPC ENQ 5	25 PPU NAK 21	45 % 37	65 5 53	105 E 69	125 U 85	145 e 101	165 u 117
0 1 1 0	6 ACK 6	26 SYN 22	46 & 38	66 6 54	106 F 70	126 V 86	146 f 102	166 v 118
0 1 1 1	7 BEL 7	27 ETB 23	47 ' 39	67 7 55	107 G 71	127 W 87	147 g 103	167 w 119
1 0 0 0	10 GET BS 8	30 SPE CAN 24	50 (40	70 8 56	110 H 72	130 X 88	150 h 104	170 x 120
1 0 0 1	11 TCT HT 9	31 SPD EM 25	51) 41	71 9 57	111 I 73	131 Y 89	151 i 105	171 y 121
1 0 1 0	12 LF 10	32 SUB 26	52 * 42	72 : 58	112 J 74	132 Z 90	152 j 106	172 z 122
1 0 1 1	13 VT 11	33 ESC 27	53 + 43	73 ; 59	113 K 75	133 [91	153 k 107	173 { 123
1 1 0 0	14 FF 12	34 FS 28	54 , 44	74 < 60	114 L 76	134 \ 92	154 l 108	174 ¦ 124
1 1 0 1	15 CR 13	35 GS 29	55 - 45	75 = 61	115 M 77	135] 93	155 m 109	175 } 125
1 1 1 0	16 SO 14	36 RS 30	56 . 46	76 > 62	116 N 78	136 ^ 94	156 n 110	176 ~ 126
1 1 1 1	17 S1 15	37 US 31	57 / 47	77 UNL ? 63	117 O 79	137 UNT _ 95	157 o 111	177 RUBOUT (DEL) 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

Interface messages are sent with ATN asserted.

KEY



GPIB INTERFACE FUNCTIONS AND MESSAGES

The ten interface functions of the GPIB (listed in figure 1) provide a variety of capabilities and options for an instrumentation system. These functions may be implemented in, or for, any particular instrument with instrument hardware or with a programming routine (software).

INTERFACE FUNCTION	SYMBOL
Source Handshake	SH
Acceptor Handshake	AH
Talker or Extended Talker	T or TE
Listener or Extended Listener	L or LE
Service Request	SR
Remote-Local	RL
Parallel Poll	PP
Device Clear	DC
Device Trigger	DT
Controller	C

Figure 1. The ten major interface functions for the GPIB.

Figure 2 illustrates the basic linkage between a GPIB controller and the interface functions implemented in another instrument on the bus. For a particular measurement or stimulus device, any or all of nine possible interface functions (SH through DT) may be selected to be linked to the tenth function (controller, C). Only those functions necessary for an instrument's purpose need be implemented by the instrument's designers; it is not likely that one instrument has all ten interface functions. For example, an instrument generally doesn't need to implement the Parallel Poll (PP) function if the instrument can respond to a serial polling sequence from the controller-in-charge (there may be more than one controller in a system).

The following is a discussion of the interface functions and their relationship to interface messages and commands.

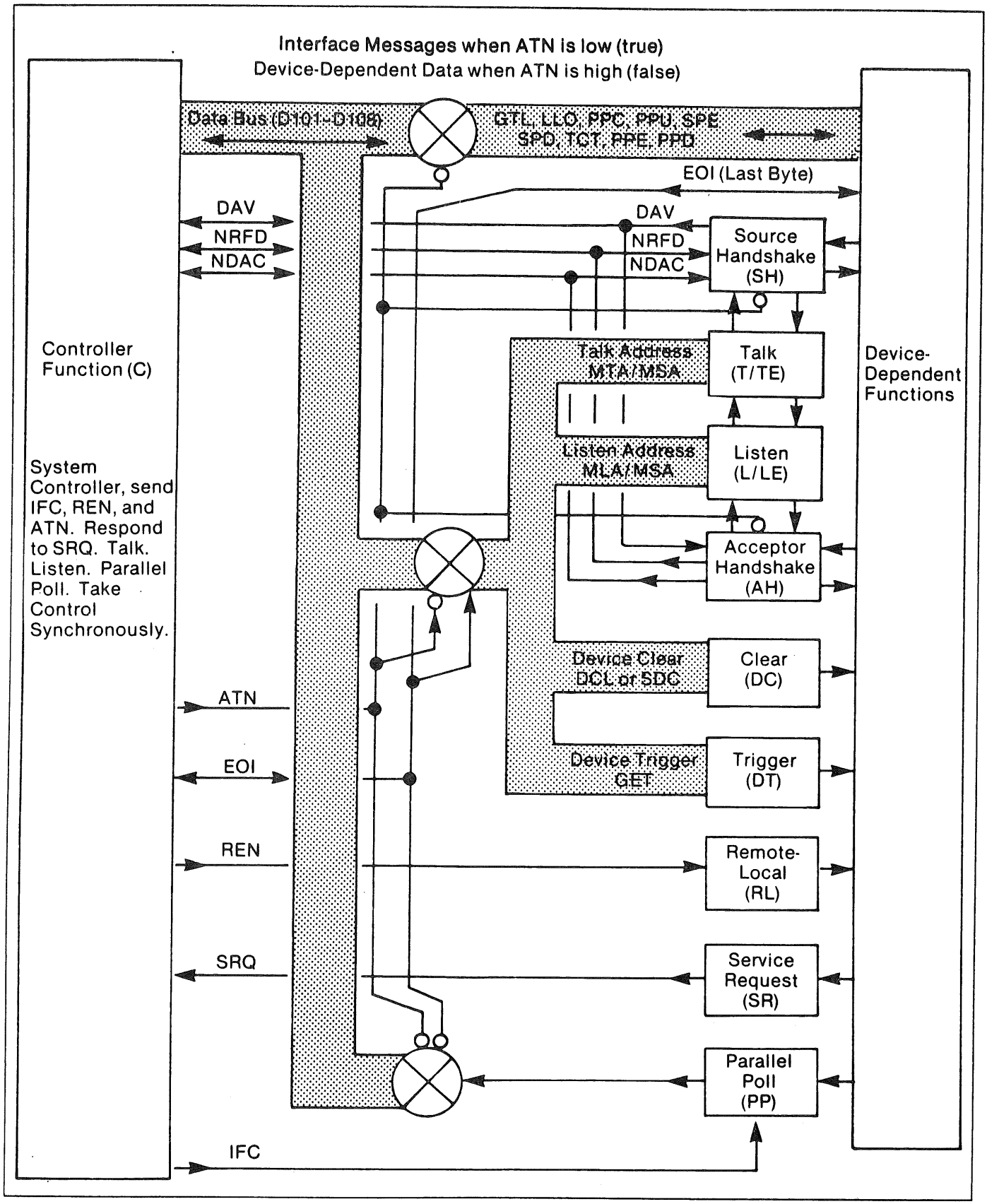


Figure 2. A maximum of ten interface functions can be linked between instruments on the GPIB.

The interface messages discussed here are shown in the ASCII and IEEE (GPIB) Code Chart. All interface messages and commands, except IFC, discussed here are sent and received over the GPIB with the ATN line asserted low true.

TALKER AND LISTENER FUNCTIONS (T/TE AND L/LE)

Although discussed under one heading, the T/TE and L/LE functions are independent of each other.

The T and TE functions provide an instrument and its secondary devices, if any, with the capability of sending device-dependent data (or, in the case of a controller, the capability to send interface messages or device-dependent program data) over the GPIB. The T (Talker) function is a normal function for a talker and uses only a one-byte address code called MTA (My Talk Address); the TE (Talker Extended) function uses a two-byte address code: an MTA code followed by an MSA code (My Secondary Address).

Only one instrument in the system can be in the talker active state at any given time. A non-controller commences talking when ATN is released and continues its talker status until an Interface Clear (IFC) message occurs, an Untalk (UNT) command is received from the controller-in-charge, the instrument is addressed as a listener (receives My Listen Address, MLA), or another instrument is addressed as a talker when a data byte called OTA, Other Talk Address, appears on the bus.

One or more instruments on the bus (up to a maximum of 14) can be programmed for the L (Listener) function by use of their specific primary listen address (MLA). All or none of these instruments may be programmed for the LE (Listener Extended) function (if implemented). The LE function requires a secondary listen address (MSA).

An instrument on the bus may be a talker only, or a listener only, or have both functions (T/TE and L/LE). In any case, its address code has the form X1OTTTTT for a talker and XO1LLLLL for a listemer. For instruments with both functions, the T-bit binary values are equal to the binary value of the L bits. The system operator sets these five bits by means

of address switches on each instrument before applying power to the system. The controller's address code may be implemented in software.

If an instrument has TE or LE functions, the five secondary address bits must not be set to the same value as the primary address bits. The system program, from the controller, designates the primary talker and primary listener status of the desired instruments by coding bits 6 and 7 (10 for talker and 01 for listener). Secondary listen addresses (or commands) are represented by the controller sending both bits (6 and 7) as a 1.

A talker may assert the EOI line with the last data byte or send a special "end of message terminator" code so that the assigned listeners will know that the talker has no more data to send.

SOURCE AND ACCEPTOR HANDSHAKE FUNCTIONS (SH AND AH)

Like the T/TE and L/LE functions, SH and AH are totally independent of each other. The SH function guarantees proper transmission of data, while the AH function guarantees proper reception of data. The interlocked handshake sequence between these functions guarantees asynchronous transfer of each data byte.

Both functions utilize the DAV, RFD, and DAC messages to transfer each byte. The SH and AH functions in some instruments, but especially in controllers, allow the source (talker) to listen to itself, with or without ATN asserted. Both functions must respond to the ATN message within 200 nanoseconds.

DEVICE CLEAR FUNCTION (DC)

The Device Clear function allows a controller-in-charge to clear (initialize) an instrument, either individually or as part of a group of instruments. The group can be either a part or all of the addressed instruments in one system.

The controller (under program direction) asserts ATN and sends either the

universal Device Clear command (DCL) or the Selected Device Clear command (SDC). When the DCL message is received, all instruments on the bus must clear or initialize their internal device functions. When the controller sends the SDC command, only those instruments which have previously been addressed to listen must clear or initialize their internal device functions. The IEEE 488 standard does not specify the state an instrument goes to as a result of the DCL or SDC command; it may be, but does not have to be, the power-up default setting.

DEVICE TRIGGER FUNCTION (DT)

The Device Trigger function allows the controller-in-charge to start the basic operation of an instrument, either by itself or as part of a group of instruments. The group may be either a part or all of the addressed instruments in one system. The IEEE 488 standard does not specify an instrument's basic operation when it receives the GET (Group Execute Trigger) command. To issue this command, the controller asserts ATN, sends the listen addresses of the instruments which are to respond to the trigger, and then sends the GET message.

Once an instrument starts its basic operation, the instrument must not respond to subsequent trigger-state transistions until the current operation is complete. Only after completing the operation can the instrument start the same operation in response to the next GET message; thus the basic operating time is the major factor that determines how fast the instrument(s) can be repeatedly "triggered."

REMOTE-LOCAL FUNCTION (RL)

The Remote-Local (RL) function provides an instrument with the capability to select between two sources of information input. The function indicates to the instrument that its internal device-dependent functions respond to information input from the front panel (Local) or corresponding information input from the GPIB (Remote). Only the system controller is permitted to assert the REN line, whether or not it is the controller-in-charge at the time.

When the system controller asserts the REN line, an instrument on the GPIB goes to the remote mode when it is addressed as a listener with its primary address; not before. In this case only, the primary listen address is sufficient to cause an instrument to go to the remote mode. For example, if several instruments have a different secondary, but a common primary address, they will all go to the remote mode when the primary address is received.

An instrument remains in the remote mode until the REN line is released, a front-panel switch on the instrument is activated to request the local mode, or a Go to Local (GTL) command is received while the instrument is enabled as a listener. However, the controller can disable the local mode function of an instrument by sending a Local Lockout (LLO) command, which applies to all instruments on the bus, addressed or not. The UNL (Unlisten) command does not return an instrument to a local mode.

All instruments must recognize when the REN line goes false (a high voltage level) and go to the local mode within 100 microseconds. If data bytes are still being placed on the data bus when REN goes false, the system program should assure that the data bytes are sent and received with the knowledge that the system is in a local mode, not remote.

CONTROLLER FUNCTION (C)

The Controller function provides the capability to send primary talk and listen addresses, secondary addresses, and universal commands to all instruments on the bus, and secondary commands to previously-addressed instruments. The controller function also provides the capability of responding to a service request (SRQ) message or conducting a parallel poll routine to determine the status of any or all instruments.

If an instrumentation system has more than one controller, only the system controller is allowed to assert the IFC and REN lines at any time during the system operation, whether or not it is the controller-in-charge at the time.

The controller function has specified time intervals for certain operations. For example, the execution time for parallel polling instruments on the bus cannot be less than 2 microseconds. The ATN message must have a controller delay of at least 500 nanoseconds to allow a current talker to see the ATN line asserted before placing a new data byte on the bus. The IFC message must be asserted for at least 100 microseconds.

If a controller requests system control from another controller and receives an internal message to send the remote enable message (REN), the controller must verify that the REN line remains unasserted (false) for at least 100 microseconds before asserting REN. The time interval that REN is asserted depends on the remote programming sequence and will vary with the program.

If a controller is in the controller active wait state and does not receive an internal message to conduct a parallel poll, it must wait at least 1.5 microseconds before going to the controller active state to give the NRFD, NDAC, and EOI lines sufficient time to assume their valid states.

TAKING CONTROL (ASYNCHRONOUS OR SYNCHRONOUS). All data bytes transmitted over the GPIB with ATN asserted are interpreted as system control information. Asserting ATN directly at any moment is an asynchronous operation with respect to the bus and may cause loss of data if a handshake cycles is in progress. To prevent loss of data, a controller can take control synchronously with the handshake cycle (if it is in progress) by first asserting the NRFD line to stop the next handshake cycle, and then automatically asserting ATN when the current talker releases DAV.

Taking control synchronously presents problems; the ATN line may not become asserted automatically if the data transfer has stopped for some reason. For example, (1) the talker may have finished talking (sent the last data byte), or (2) the talker may be very slow to send the next data byte, or (3) an instrument on the bus may not be functioning properly. Programmers can solve the first problem if they know that all talkers on the bus assert EOI with the last data byte. The second and third problems may require asserting IFC to clear the bus and then asserting ATN asynchronously.

PERFORMING A SERIAL POLL. The controller may conduct a serial poll at any time, whether or not an instrument has asserted the SRQ line. Most, but not all, instruments have the Service Request (SRQ) function.

To perform a serial poll, the controller first asserts ATN and issues the Untalk (UNT) and Unlisten (UNL) commands. The controller then sends Serial Poll Enable (SPE) command, followed by the talk address of the first instrument to be polled. The controller then releases ATN, and the talker responds by sending its status byte over the data bus. If the addressed talker has requested service, it must assert bit seven of the status byte and encode the remaining seven bits of the status byte to indicate the reason for asserting SRQ. Status bytes are device-dependent and are not specified in the IEEE 488 standard.

An addressed instrument will release its SRQ line when serial polled, but other instruments may still be holding it asserted. When the controller has read the status byte of an addressed instrument, it should send the UNT and Serial Poll Disable (SPD) commands before repeating the procedure to poll the remaining instruments. The routine should continue until the controller no longer detects SRQ asserted.

PERFORMING A PARALLEL POLL. The Parallel Poll (PP) function provides an instrument with the capability to present one, and only one, bit of status information to the controller without being previously addressed to talk. The parallel polling capability requires a commitment by the system program to periodically conduct a parallel poll sequence.

When an instrument responds to a parallel poll, the single data bit presented to the controller may or may not indicate a need for service. If the data bit is used as a service request function, the controller should perform a serial poll in order to obtain a complete status byte.

Before an instrument can respond to a parallel poll, the GPIB system must first be configured. The controller asserts ATN, sends the UNT command followed by the listen addresses of all instruments to be included in the parallel poll, and then sends the Parallel Poll Configure (PPC)

command. The PPC command is followed immediately with the Parallel Poll Enable (PPE) command to cause all listeners to go to the parallel poll standby state.

If the EOI line has been asserted along with ATN, all selected instruments have up to 200 nanoseconds to go to the parallel poll active state. EOI may be asserted along with the PPE command, or at any time after the PPE command. An instrument does not present its one bit of status information to the controller until it sees both ATN and EOI asserted.

The PPE message sent by the controller has the form X110SPPP. Bit 4 (S) is called the sense bit, and PPP is an octal number (000=0 through 111=7) designating a specific data line DI01 - DI08) that an instrument must assert if its internal status message has the same value as the sense bit (S may equal 1 or 0). If so designed, the controller can read the data lines while ATN is asserted to interpret the status of the instruments.

To conclude the parallel poll, the controller releases EOI and then sends the Parallel Poll Disable (PPD) command. If the system needs to be configured, the Parallel Poll Unconfigure (PPU) command is sent, followed by the Unlisten (UNL) command.

PASSING CONTROL. As a controller-in-charge, the system controller (program) may relinquish control to any other instrument in the system capable of acting as a controller. The controller-in-charge first addresses the other controller as a talker, and then sends the Take Control (TCT) command and other desired control messages. The other controller becomes controller-in-charge when ATN is released.

SOFTWARE MAKES THE SYSTEM WORK

The IEEE 488 standard does not specify how devices are to respond to signals on the GPIB data lines. It simply defines the GPIB as a common communication link for data and commands between instruments and controllers. System control software must be flexible to handle all "GPIB-compatible" instruments, including those that require entirely different commands to perform the same function.

The need for flexible programming goes further. As programmable instruments become more sophisticated, the user's interaction with the system is increasingly at the keyboard, not at the rear panel. "Human-readable" programs are necessary to help applications-oriented users work efficiently. For small test instrumentation applications where the system setups change frequently, higher-level languages are an even greater necessity.

As discussed earlier, the GPIB provides 10 interface functions between instruments and the controller. The IEEE 488 standard also defines the interface messages or commands that perform each function. However, the standard does not specify codes or formats for device-dependent messages, or data and instructions sent from device to device.

The following section describes the interface messages as well as conventions for encoding device-dependent messages. Figure 1 shows the functional relationship of interface messages and device-dependent messages.

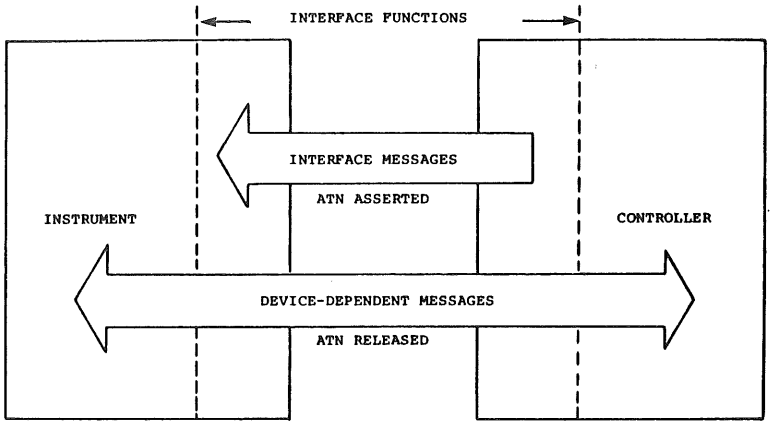


Figure 1. Interface messages and device-dependent messages.

INTERFACE MESSAGES

Interface messages are commands issued by the controller that define communication paths through the bus to instruments. Figure 2 shows the messages or instructions that execute each interface function. The standard sets a one-byte (7-bit) value for each interface message. The nomenclature that a controller uses to send these values to the GPIB can vary from controller to controller.

Interface Function	Interface Message
Talker (T or TE)	MTA (My Talk Address) MSA (My Secondary Address)
Listener (L or LE)	MLA (My Listen Address) MSA (My Secondary Address)
Source Handshake (SH) Acceptor Handshake (AH)	No commands
Remote-Local (RL)	GTL (Go to Local) LLO (Local Lockout)
Device Clear (DC)	DCL (Device Clear) SDC (Selected Device Clear)
Device Trigger (DT)	GET (Group Execute Trigger)
Service Request (SR)	SPE (Serial Poll Enable) SPD (Serial Poll Disable)
Parallel Poll (PP)	PPC (Parallel Poll Configure) PPU (Parallel Poll Unconfigure) PPE (Parallel Poll Enable) PPD (Parallel Poll Disable)
Controller (C)	UNT (Untalk) UNL (Unlisten) TCT (Take Control)

Figure 2. Interface functions and messages.

To issue any of these interface messages, the controller first asserts the ATN line. Then, for addressed messages, or those directed to selected devices, the controller issues UNT (Untalk) and UNL (Unlisten), and specifies new talk and listen addresses before sending the message. Addressed messages include GTL (Go to Local), SDC (Selected Device Clear), PPC (Parallel Poll Configure), GET (Group Execute Trigger), and TCT (Take Control).

When the controller issues universal commands, it does not specify addresses; all devices receive these messages. Universal commands include LLO (Local Lockout), DCL (Device Clear), PPU (Parallel Poll Unconfigure), SPE (Serial Poll Enable), and SPD (Serial Poll Disable).

DEVICE-DEPENDENT MESSAGES

Device-dependent messages are instructions and data that one device sends to one or more other devices on the bus. After asserting ATN to send bus-control messages, the controller releases ATN to allow the transmitting of device-dependent messages. A device continues talking until the controller performs one of the following operations:

asserts IFC (Interface Clear)
sends the universal UNT command
Addresses the talker as a listener
addresses another device as a talker

CODES AND FORMATS

The GPIB standard permits this flexibility in coding so that older instruments also can operate through the bus. Another advantage is that instrument users can choose coding which optimizes the use of each instrument on the GPIB. Most commonly, devices send messages in ASCII or binary code.

The format in which devices transmit codes defines the meaning of the information sent. Nearly all present-day measurement instruments use the ANSI X3.42 standard format, which recognizes three types of numbers: fixed with an implied decimal point, floating with an explicit decimal point, and exponential notation with the most significant character sent first. This format characteristic is not true for all instruments and should be examined carefully for each instrument.

So far, instrumentation manufacturers have not adopted a standard message termination character. Some controllers recognize CR (carriage return) as a termination character while some instruments send both CR and LF (line feed). In this example, the controller may stop the talking instrument until the next time the controller instructs it to talk. If the

controller is programmed not to accept LF as the first code in a message, the system may stop operating.

In addition to providing device compatibility, software also can increase human capability with the system. Higher-level languages that use English-word commands, descriptive headers for data values, and punctuation delimiters make sofrware easier for a system designer to write and modify; they also can improve controller interaction with programmable instruments.

For example, a power supply with minimum intelligence may require the command "2314" to output 15.7 volts. This character string bears no apparent relationship to the desired result. To interpret this number, the operator or programmer must know that the "2" tells the power supply to use its second range (0-50 volts), and "314" plugs into the formula "(XXX/1000) x 50" for determining the fraction of the voltage range to be produced ((314/1000) x 50 = 15.7).

A power supply with more intelligence simply may require "15.7" to produce the desired voltage, performing the conversion with its own microprocessor rather than requiring a programmer with a calculator. On the other hand, a device that makes two different measurements such as voltage and frequency may send numbers that the operator can understand but not identify. However, if the instrument is programmed to send a "header" first that describes the data value (such as VOLTS 3.74), the operator need not guess the measurements that the instrument is reporting.

Arguments within header descriptions go one step further in increasing programming capability. For example, if the header FREQ is used to indicate center frequency, the controller can attach the argument "?" to FREQ to prompt the instrument for the center frequency measurement. The instrument responds with FREQ NUM, where the NUM argument represents the actual frequency measured. A comma or a semicolon may be used to separate these reported measurement results.

PROGRAMMABILITY VERSUS COMPATIBILITY

As implementation of the GPIB has reduced the trial-and-error of

assembling measurement system, increased programmability of measurement instruments has made GPIB-based systems easier to operate. Instruments incorporating microprocessors are able to perform more complicated functions. For example, a digital multimeter may perform computations on its measured values; it also may perform self-calibration or error diagnosis.

Regardless of its level of programmability, a GPIB-compatible instrument may be exactly the wrong device for a system. A spectrum analyzer without the service request compatibility can cause problems, particularly when calibration errors occur in unattended testing situations. A further concern is whether an instrument can generate a service request after its setup program has finished preparing it for a trigger from the controller.

Potential buyers of GPIB equipment must read the documentation carefully to ensure that the devices can handle the necessary functions and are software-compatible. When the user's interaction with an instrument is primarily from a keyboard and not at the front panel, the ability to control instrument functions through the GPIB is crucial. Figure 3 shows a list of instrument capabilities to consider before buying GPIB-compatible equipment.

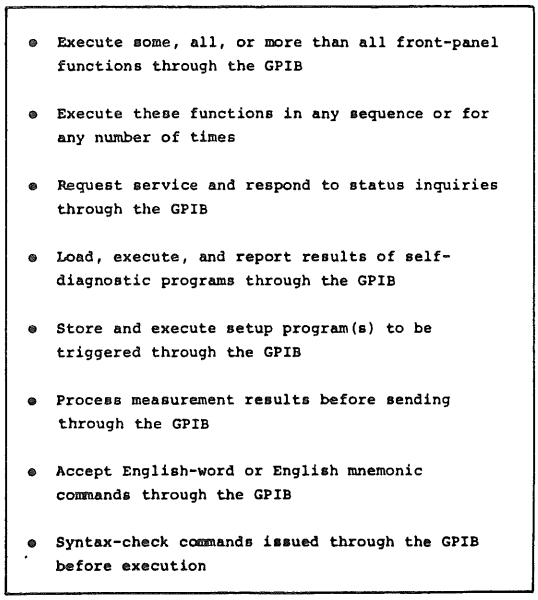


Figure 3. Instrument capabilities to consider for GPIB-compatible instruments.

A SAMPLE PROGRAM

The following short program illustrates typical operation of a small GPIB-based instrumentation system. The controller is a Tektronix 4051 Graphic Computing System programmed in standard BASIC. It is connected to the Tektronix 492P Programmable Spectrum Analyzer.

The program acquires ASCII messages entered at the 4051 keyboard and sends them to the 492P. The first message requests the measurement setup parameters. The second message prompts the user to enter a query. The controller then displays the 492P response(s) on the 4051 screen.

```
100 PRINT "ENTER SETUP PARAMETERS"
110 INPUT P$
120 PRINT @3: P$
130 PRINT "ENTER QUERY"
140 INPUT Q$
150 PRINT @3: Q$
160 INPUT @3: R$
170 PRINT R$
180 END
```

Please note:

1. The PRINT and INPUT statements perform the following functions: PRINT sends data in ASCII code format to the addressed device(s). INPUT receives data (measurement or device status) in ASCII code format into the control program from the addressed device(s).
2. A device address is determined by switch settings on the device. Five switches usually are present, enabling bus device addresses from 0_{10} (00000_2) through 31_{10} (11111_2) to be used (the address consists of the five lower-order bits of the 7-bit ASCII code). Tektronix software interprets these settings as decimal values. The ASCII and IEEE (GPIB) Code Chart shows codes for primary and secondary talk and listen addresses.
3. Specifying no address for the PRINT statement defaults to the 4051 display unit. The default address for the INPUT statement is the 4051 BASIC program.

PRINT and INPUT automatically execute the appropriate interface messages as they perform specified data handling functions. A data handshake occurs with each byte sent through the bus. The discussion below, accompanied by block diagrams (figures 4 and 5), explains the controller/ instrument interaction through the bus with each statement for the program listed above.

```
100 PRINT "ENTER SETUP PARAMETERS"
```

This statement tells the display unit to display or output the message within quotation marks. The display unit is the default address of the instruction when no other address is specified. The interface commands and device-dependent messages listed below correspond to the single BASIC statement above.

Message	Description
MLA [?]	Selects the 4051 display unit [TEK: what address?] as the listener.
E N T E R (space) S E T U P (space) P A R A M E T E R S	Sends the contents of the message to be displayed.
(CR-EOI)	Terminates the device-dependent message.
UNT	Stops the talker.
UNL	Stops all listeners.

For this example, the user enters the following parameters:

FRQRNG 1; FREQ 1G; SPAN 10M; REF -30; VRTDSP LOG:10; RES 1M

which specifies a frequency range of 1 MHz, a center frequency of 1 GHz, a frequency span of 10 MHz/div, a reference level of -30 dBm, a vertical scale factor of $\log_{10}$ dB/div, and a resolution bandwidth of 1 MHz.

110 INPUT P$

This statement reads the message entered at the keyboard into the control program. The user terminates the message with CR and LF.

120 PRINT @3: P$

This statement transfers the user message to the spectrum analyzer, which is at address 3. Figure 4 shows a functional diagram of this process. The 492P sets itself up according to the parameters issued by the user.

PRINT @3: P$

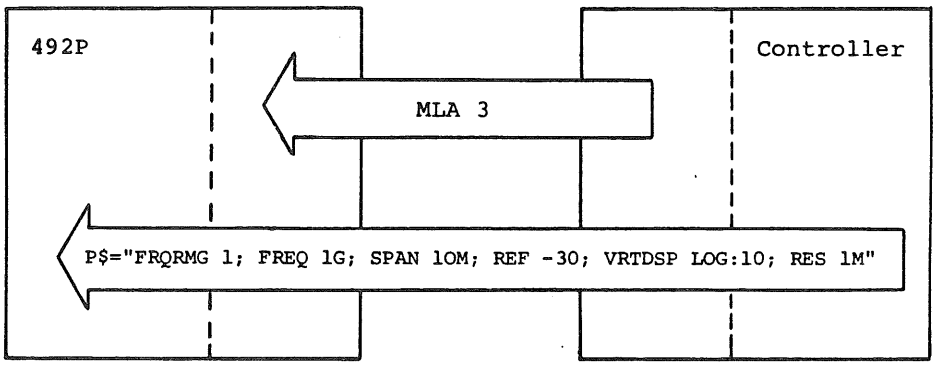


Figure 4. Functional diagram of PRINT statement.

130 PRINT "ENTER QUERY"

This statement instructs the display unit to output the words ENTER QUERY on the screen. This message prompts the user to type a query to the spectrum analyzer. The user enters the following statement:

FRQRNG INC; FRQRNG? FREQ?

which sets the frequency range to increment once and then requests the new frequency range and center frequency.

140 INPUT Q$

This statement loads the user message Q$ (FRQRNG INC; FRQRNG? FREQ?) into the 4051 control program.

150 PRINT @3: Q$

This statement sends the user message to the spectrum analyzer.

160 INPUT @3: R$

This statement reads the 492P responses, R$, into the 4051 control program. Figure 5 illustrates the data transfer. In this example the 492P responds with the following data: [TEK: Please provide 492P responses.]

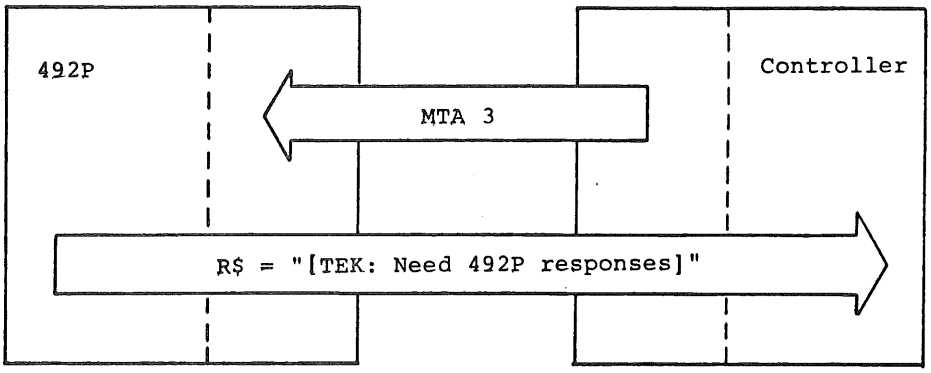


Figure 5. Functional diagram of INPUT statement.

The interface commands and device-dependent messages corresponding to this statement are:

UNL	Turns off all listeners.
MTA 3	Sets up the 492P as the talker.
MLA [?]	Establishes the 4051 control program [TEK: Please provide device address] as the listener.
R$	Transfers the 492P device message responses R$ [TEK: Need real responses] to the 4051 control program.
(CR-LF-EOI)	Terminates the 492P message.

```
170 PRINT R$
```

This statement outputs the 492P responses stored in program memory to the 4051 display screen.

```
180 END
```

This statement terminates program execution.