How to Interface with the IEEE-488 Bus

Practical information on the timing and procedures

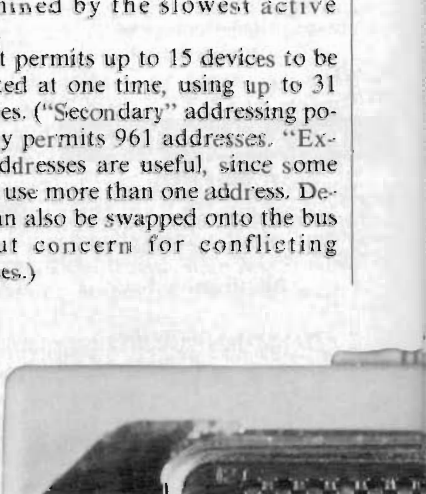
by Peter L. Andresen

Situations often arise where laboratory instruments may have to be portable and connected to more than one computer, or computers of different types. Since many instruments are sold complete with a built-in IEEE-488 interface, and the bus controller can handle many different types of instruments, it is no longer necessary to design a special computer interface for each instrument and for each computer. Only the bus controller needs to be specific for any given computer, though of course the software drivers may have to be different also.

The characteristics and use of the IEEE-488 bus, designed for controlling laboratory equipment, will be described in this article. Readers may wish to augment this with the official ANSI/IEEE Standard [1] or with more readily digestible information available from various manufacturers [2,3]. I will discuss the 10 basic functions or capabilities of the bus, the purpose of the 16 signal lines, the types and effects of bus commands issued by the Controller, device-dependent characteristics of the bus, and will give an example of a simple data acquisition program in Basic.

Some distinguishing features of the IEEE-488 bus are:

- It is highly standardized.
- It has universal commands to which all devices respond.
- It transmits 8 data bits in parallel with full handshaking at rates up to 1 MB (rarely achieved and automatically determined by the slowest active device).
- It permits up to 15 devices to be connected at one time, using up to 31 addresses. ("Secondary" addressing potentially permits 961 addresses. "Excess" addresses are useful, since some devices use more than one address. Devices can also be swapped onto the bus without concern for conflicting addresses.)



Weaknesses of the bus include:

• Restrictions on total cable length of 2 meters times the number of devices (maximum of 20 meters).

• No standardization of the operational characteristics of the bus (cf. interpretation of data such as ASCII, binary, BCD, etc., device program codes).

• Potential competition for access to the bus if there are a large number of slow devices.

• Generally, about two-thirds of the devices must be powered up to ensure proper bus operation. However, if power-down devices do not load the bus, the number of active devices is unimportant.

• Further restrictions on cabling (plus a lack of fast devices) make it difficult to attain transfer rates even remotely close to the maximum.

Additionally, the IEEE-488 specification itself is quite complex. This has resulted in subtle errors in its implementation and description (to which this article may not be immune).

The power and flexibility of the 488 bus reside predominantly in the Controller, which can dynamically reconfigure the bus participation and is itself generally equipped with Talker and Listener functions. For example, a Controller can send commands to address a device to talk, listen, be inactive, or cause either some selected devices or all devices to perform a common function (e.g., trigger). Controllers can be purchased as standalone dedicated devices or added onto general-purpose computers (as, for example, with Hewlett-Packard computers or as hardware additions to S-100 computers).

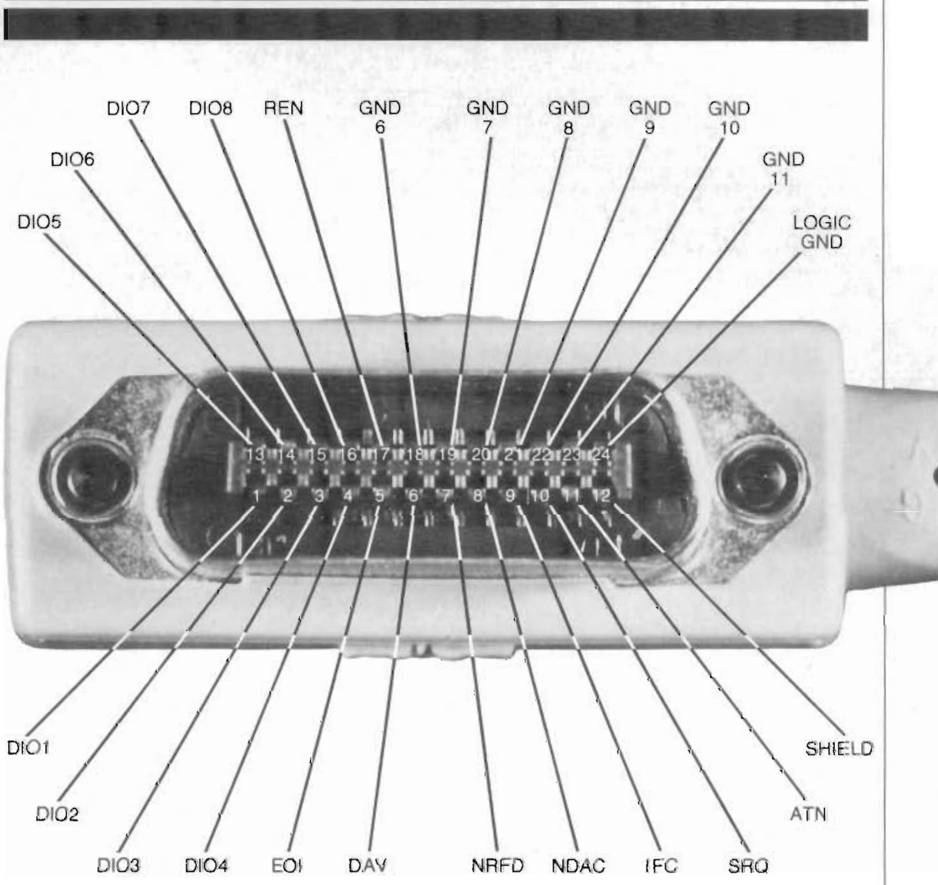
Several manufacturers offer LSI chips that perform the 488 interface functions (with varying degrees of capability and speed) and are used by most 488 devices (including the S-100 based controller) manufacturers. The 488 interface boards available for S-100 computers have grown markedly to include Cromemco, Dylon, D & W Engineering, I/O Technology, National Instruments, Pickles & Trout, and others. (Note to Osborne owners: errors in the 488 drivers in the (early) single-density ROM BIOS were corrected in the double-density version. In single density, either avoid using the Control Out routine with REN TRUE, or follow it by going to Standby, then Take Control.)

I have used some of these boards (such as Pickles & Trout) extensively. Most come with machine-language source code to drive the board, and many offer software that easily interacts with Microsoft Basic and various compiled languages. Some also offer S-100 interrupt capability. Since the S-100 processor must interact with the 488/S-100 interface board, transfer rates on the 488 bus that involve the 488/S-100 interface board are generally limited to a maximum of 5 to 50 K/sec.

IEEE-488 bus

There are only three basic participants (Table 1) on the bus: Controllers (only one may be in charge at a time); Talkers (only one at a time); and Listeners (limited by the 15-device maximum of the bus). Coupled to these functions are the Source and Acceptor Handshake functions necessary to ensure error-free transfers on the data lines. There are five remaining bus functions, which provide special optional capabilities. Devices usually specify their 488 capability in a code printed next to the connector.

The Controller orchestrates the activities on the bus and generally identifies its unique status on the bus by asserting TRUE on the Attention (ATN)



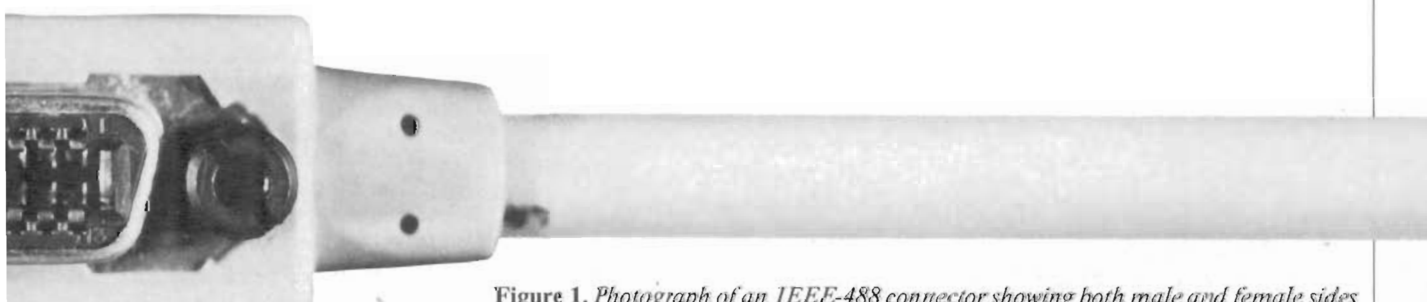


Figure 1. *Photograph of an IEEE-488 connector showing both male and female sides and the corresponding signal pinout.*

IEEE-488 BUS

Continued from page 89

line, at which point all other bus activity must cease and all devices must "listen" to it. This places the interface in the *command* mode, where all devices handshake data on the data lines and interpret them as (Multiline) commands. When the ATN line is FALSE, the interface is the *data* mode, where data (whose significance is device dependent) are sent by a Talker and received only by Listeners.

There are really two categories of commands: Uniline and Multiline. *Uniline* commands are so named because only one (Interface Management) line is used. *Multiline* commands use the data lines (and the ATN Interface Management line). They are further broken down into five basic groups: Addressed, Universal, Listen Address, Talk Address, and Secondary commands, which are distinguished by the three or four most significant bits (Table 2). These commands are subsets of Uniline and Multiline *messages* that describe all types of transmissions over the bus other than handshaking.

Most Controllers also have the capability to participate as normal Talkers or Listeners. This multiplicity of roles and abilities must be clearly understood, since the bus will not function at all well if the Controller's presence is felt when you want other bus transactions to occur.

Whenever a device is connected to the 488 bus and powered on, its interface module is monitoring the ATN (and other) lines to determine whether the Controller is issuing a command. Initially a device is inactive (except for Talk- or Listen-Only devices, which power up as fixed participants) in terms of participating in exchanges of data on the bus, although it can still be operating in a useful, independent manner.

The transition to bus participation comes when the Controller sends out a command that assigns a role (Talker or Listener) to the device. Each device has a 5-bit binary address that is switch settable and is sent along with a 2-bit code to indicate whether it is to be a Talker or Listener (Table 2). It is important to distinguish between a device that has been assigned and functions as a *Listener* (involved in data transfers), and a device that "listens" when the Controller speaks (which is something that all devices must do).

Finally, the remote versus local description of a device's operation can be confusing. Remote/Local capability refers to a device's ability to be programmed (for range, triggering, etc.) over the bus, and is not a description of its participation or lack thereof on the bus. A device can be in Local mode and yet be acting as a Talker or Listener.

IEEE-488 functions

The Standard defines 10 *interface functions* (Table 1). These are distinct from the device's internal functions, like range, "trigger" mode, etc., which are set on its front panel (and may be programmable over the bus). The use of the commands related to the 488 functions will be detailed in the section on Multiline Commands. Each of the 10 functions has different levels of implementation (Table 1) ranging from implemented (SH1) versus not implemented (SH0) for the Source Handshake, to many different levels of capability for Controllers. (Functions C1-C4 can be chosen in certain combinations and coupled with one of the C5-C28 functions for a total of approximately 193 possible variations.)

A device is not required to implement all functions. A signal generator may be equipped as a Listener with Remote/Local and Trigger (bus) capabilities, while a digital voltmeter might also include Talker, Service Request, Parallel Poll and Device Clear capabilities. A minimum capability would have to include a Listen function (with Acceptor Handshake). A simple bus configuration could consist of a digital voltmeter operating in a Talk-Only mode and a printer in a Listen-Only mode, where no Controller is needed.

The core of the Standard is a description of each function in terms of allowable *states* (Table 1) and the conditions under which transitions occur from state to state—for instance, from Talker Idle (not a Talker) to Talker Active. The number of allowable states depends on the level to which a function is implemented (a Talker with Serial Poll capability has five states; without, only three states). In general, on power-up, all bus functions are in an idle or equiva-

Table 1. IEEE-488 interface functions

Function	Implementation codes*	Allowable states†
1. Source Handshake	SH0-SH1	6
2. Acceptor Handshake	AH0-AH1	5
3. Talker	T0-T8 TE0-TE8	3-8
4. Listener	L0-L4 LE0-LE4	3-5
5. Service Request	SR0-SR1	3
6. Remote/Local	RL0-RL2	2-4
7. Parallel Poll	PP0-PP2	3-5
8. Device Clear	DC0-DC2	2
9. Device Trigger	DT0-DT1	2
10. Controller	C0-C28	up to 20

E1 = Open collector drivers
E2 = Tri-state drivers
* *"O" indicates no capability*
† *Depends on implementation*

Table 2. Categories of multiline commands

Mnemonic	Message name	Type	Class	DIO 8	DIO 7*	DIO 6*	DIO 5	DIO 4	DIO 3	DIO 2	DIO 1	ATN
ACG	address command group	M	AC	×	0	0	0	×	×	×	×	1
UCG	universal command group	M	UC	×	0	0	1	×	×	×	×	1
LAG	listen address group	M	AD	×	0	1	×	×	×	×	×	1
TAG	talk address group	M	AD	×	1	0	×	×	×	×	×	1
SCG	secondary command group	M	SE	×	1	1	×	×	×	×	×	1

M = Multiline
U = Uniline
1 = Logical one
0 = Logical zero
× = Don't care or not relevant.
L = 5-bit listen address
T = 5-bit talk address
AC = Addressed command
AD = Address (talk or listen)
DD = Device dependent
SE = Secondary command
ST = Status
UC = Universal command

IEEE-488 BUS

Continued from page 90

lent state. While not necessary for its use, a thorough understanding of the bus requires familiarity with these state diagrams [1]. The 10 functions are:

1,2. Handshaking (SH and AH). An interlocked handshake sequence is used to guarantee proper transfer of each byte on the data lines (DIO1-DIO8). The transfer cannot be initiated by the Source until all Acceptors are ready, and any Acceptor will delay the the termination of the transfer (or Multiline message) until it has fully accepted the byte. Thus, the slowest participating device determines the pace of the handshake. See "Hardware Details: Handshaking Lines" below.

3. Talker or Extended Talker (T, TE). All Talkers (only one permitted at a time) must be capable of Source and Acceptor Handshaking, since they must also be able to receive Multiline commands from the controller. Capabilities vary with a device's ability to *(1)* respond to Secondary (or extended, if more than 31) addresses (T vs. TE); *(2)* provide a Talk-Only mode (T & TE 1,3,5,7); *(3)* respond to a Serial Poll (T & TE 1,2,4,5); and *(4)* unaddress (unbecome a Talker) when its listen address is sent (T & TE 5,6,7,8). Talkers (except Talk Only) must also cease being Talkers if a Universal UNTalk (UNT) command or Other Talk Address (OTA) is sent by the Controller. The Standard requires the Talk-Only mode to be activated by a manual switch, which causes the device to power up and permanently occupy the Talker role on the bus (it won't UNTalk). Note that the Serial Poll capability is not a separate function, but a subset of the Talk function.

4. Listener or Extended Listener (L, LE). Listeners differ with respect to their ability to *(1)* respond to Secondary addresses (L vs. LE); *(2)* provide a Listen-only mode (L & LE 1,3) and *(3)* unaddress (unbecome a Listener) when their talk address is sent (L & LE 3,4). The Listen-Only capability is similar to Talk Only. Primitive Listeners (L & LE 1,2) require only Acceptor Handshake capability.

5. Service Request (SRQ). Devices with this capability (SR1) can request service from the Controller at any time by asserting TRUE on the SRQ Interface Management Line. They must also be able to respond to a Serial Poll (T & TE 1,2,4,5). See "Multiline Commands: (Universal) Serial Poll Enable."

6. Remote/Local (RL). Devices with this capability (RL1,2) can select between two sources of programming input (for range, function, triggering, etc.): *(1)* front panel control (local) or *(2)* 488 interface (remote). A device's programmability in Remote may be less extensive (or more so) than in Local. A device goes into Remote when the Controller sends its Listen address while asserting the REN line. In Remote, the front panel controls, except power on/off and bus-related controls (manual SRQ and Go To Local) are inoperative. RL1 differs from RL2 in its ability to permit Lockout (Local Lockout—LLO) of the Go To Local front panel button.

7. Parallel Poll (PP). Provides a device with the ability to convey its status over one of the data lines to the Controller. Devices are individually configured and can then be simultaneously polled by the Controller much more quickly than with Serial Polling. PP1 signifies that the device's response to a Parallel Poll is remotely configurable; PP2 signifies that it is locally configurable. See "Multiline Commands: (Addressed) Parallel Poll Configure."

The transition to bus participation comes when the Controller sends a command that assigns a role to a device.

8. Device Clear (DC). DC1 provides the ability to clear (initialize) devices individually or as a group using an Addressed command (Selective Device Clear) or a Universal command (Device Clear). DC2 omits Selective Device Clear. Generally, Device Clear places the device's functions (voltage scale, triggering, etc.) in the power-on state (this is not identical to power on in terms of placing the device's 488 interface in a quiescent state). However, a manufacturer is free to clear the device's functions to any condition.

9. Device Trigger (DT). DT1 is an Addressed command that triggers the operation of devices individually or in groups. Once an operation has been started, a device will not respond to subsequent Device-Trigger commands until its first operation is complete.

10. Controller (C). Only Controllers have the ability to assert the ATN, IFC or REN lines and are therefore uniquely able to create Talkers or Listeners, as well as send Addressed (e.g. Group Trigger) or Universal (e.g. Device Clear) commands. Controllers vary widely in their ability to perform a Parallel Poll, respond to a Service Request, perform Talk and Listen functions, pass or take control from a second Controller, etc. With the exception of the latter, many Controllers have all of the above capabilities.

Hardware details

For those experienced with the vagaries of setting up RS-232 interfaces, the 488 bus will come as a pleasant surprise. All 488 cable connectors incorporate both male and female (stackable) connections with locking screws (Figure 1), so that adapters are never necessary and both linear and star configurations are easily set up. (Watch the screws! Black threads are metric; silver are English.) Also, there are no ambiguities regarding baud rate, stop bits, which signals appear where, etc.

The 488 bus employs eight Data lines, three Handshaking lines, and five Interface Management lines. Lines are generally at 2.5 to 3.7 volts until an open collector line driver pulls the line to ground (tri-state drivers are optional on some of the lines; see Table 1). This results in a wired ORing scheme, whereby any device(s) can maintain a line low. The 488 Standard uses a negative TRUE logic convention, meaning that TRUE (logical 1) equals low (< 0.8 volts) and FALSE equals high (> 2.0 volts).

Data lines (DIO1-DIO8). The Data lines are used to transmit data (including "normal" data, device status and device program codes) between a Talker and Listeners, as well as Addressed commands (between a Controller and Listeners) and Universal commands (between a Controller and all devices).

Handshaking lines (NRFD, DAV, and NDAC). The three Handshaking lines operate under a fairly complex set of rules to provide a fully interlocked handshake for transmitting over the Data lines. The detailed sequence of events is described in Figure 2. It is again worth noting that the pace of the handshake (i.e., any transmission over the Data lines) is controlled by the slowest participating device (Controller, Talker or Listener).

Interface Management lines (ATN, EOI, IFC, REN and SRQ). Five lines are used to manage the 488 bus, and the information conveyed on

IEEE-488 BUS

Continued from page 92

these lines is usually known as *Uniline* messages, since only one line is involved (i.e., in general, the data lines are not used). These can take the form of commands from the Controller (e.g., IFC), but also include messages to the Controller (e.g., SRQ) or to Listeners (EOI). The five interface management lines are:

ATN. Attention is asserted low (TRUE) *only* by the Controller, indicating to all devices that the Data lines will contain a special command to tell devices to become Talkers or Listeners or perform a Secondary, Addressed (to Listeners) or Universal command.

EOI. End Or Identify is used to signal the last byte of a string sent on the Data lines asserted by a Talker (ATN FALSE) as the last byte is placed on the Data lines. It is also asserted (with ATN TRUE) by the Controller to initiate a Parallel Poll.

IFC. The Interface Clear line is asserted low (TRUE) only by the controller and places the bus (i.e., the 488 interface modules in all devices) in a known quiescent state. For example, all Talkers and Listeners go into the idle state (UNTalk and UNListen), although devices do not Go To Local (GTL), nor are they cleared (DCL or SDC).

REN. The Remote ENable line is asserted TRUE only by the Controller to permit remote programming of devices on the bus. Devices with Remote/Local (RL1,2) capability (and providing they are Listeners and REN is TRUE) are able to be remotely programmed by receiving Multiline messages (usually sent by the Controller acting as a Talker). Generally, remote programming mimics (some or all of) and disables the front panel controls (except those which relate to the bus, like Service Request or Go To Local buttons—see Universal Command: Local Lockout). On power-up or if REN becomes FALSE, all devices will shift to the Local state. See "IEEE-488 Functions: Remote/Local and Device-Dependent Messages."

SRQ. The Service ReQuest line is asserted TRUE by a device(s) capable of Serial Polling (a Talk function subset) to signal a need for attention. Some devices can be programmed (using a status mask) to define the conditions under which they will generate an SRQ. Some devices also permit SRQ to be generated from front panel (operator) input. The Controller must sense the SRQ line and perform a Serial Poll to determine which devices need service. See "Multiline Commands: (Universal) Serial Poll Enable."

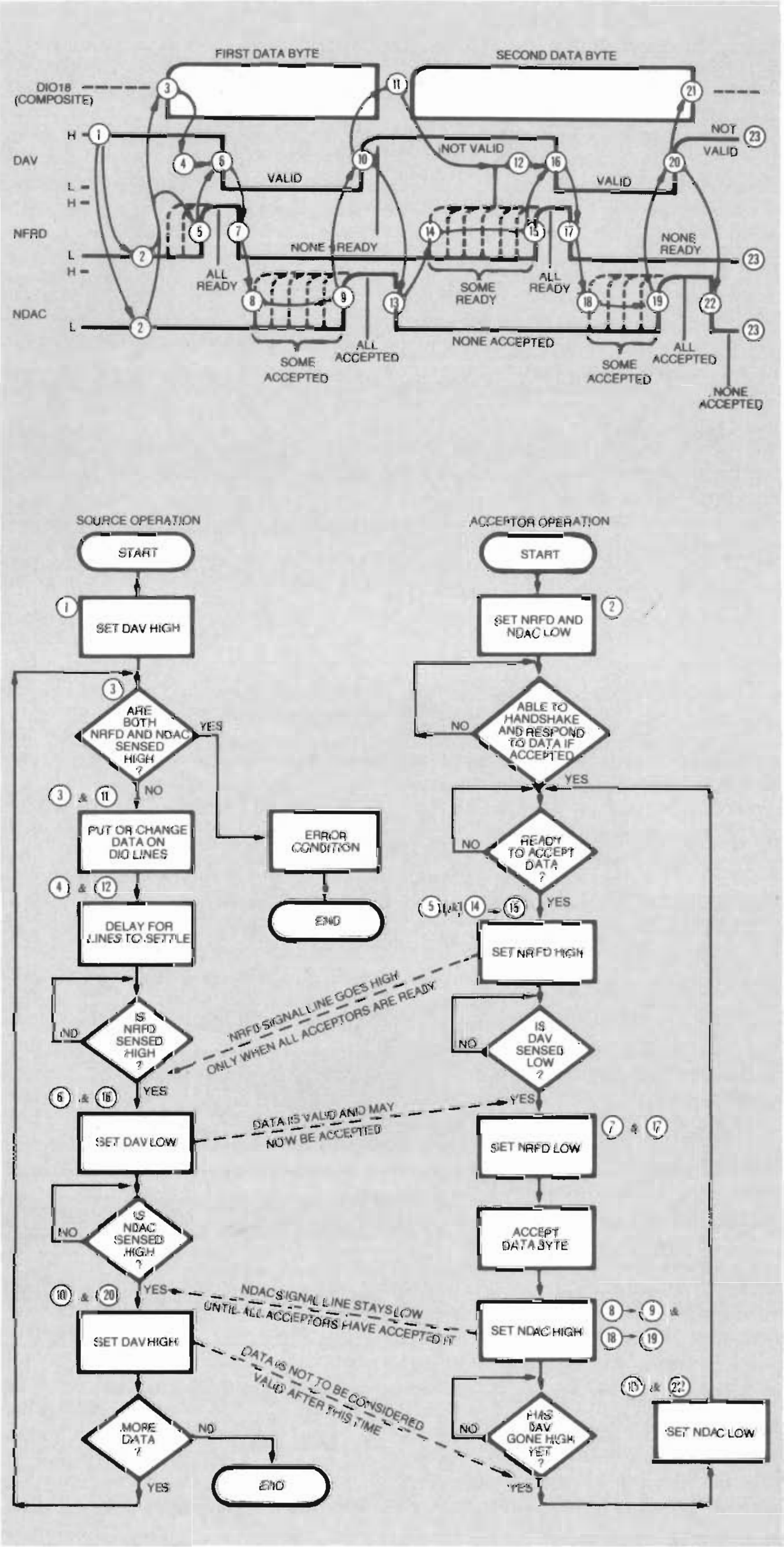


Figure 2. *The detailed sequence of events involved in handshaking process. Reprinted with permission from reference [1].*

IEEE-488 BUS

Continued from page 94

Multiline commands

Multiline commands (ATN TRUE—see Tables 2 and 3) are an artificial subset of Multiline messages, which are simply bus transactions that use the data lines and therefore involve handshaking. The Controller sends a Multiline command by asserting ATN TRUE and simultaneously placing a byte command (a 2-byte sequence is used for Secondary commands) on the data lines. The type of command is determined by the three most significant bits (Table 2), with the lower bits specifying the particular address or command. The Controller also uses Multiline messages by acting as a Talker (ATN FALSE—after first placing a device into the Listen state) to send a series of data bytes that program the device for voltage scale, triggering mode, etc. These noncommand messages are sent while ATN is FALSE and are called *device-dependent messages* (to be discussed in that section).

Talk and Listen (Addressed) commands. The Talk and Listen address commands are characterized by the three high bits (X10 for talk, X01 for listen—see Table 2). Devices can share a Listen address, although a Talk address cannot be shared by other Talkers. Devices which have both Talk and Listen capabilities often use a common address for both functions. The address of a device is set by five switches (usually at its rear), which select an address between 0 and 30 (binary 00000 to 11110) and correspond to the low 5 bits of the Talk (MTA) or Listen (MLA) address command (Table 3).

A device with a binary address of 01001 would recognize its Talk address (i.e., become a Talker) by seeing X10 01001 (ASCII 'I') on the data lines (with ATN TRUE), while its listen address would be X01 01001 (ASCII ')'—where 'X' means the bit isn't used). Upon hearing its Talk or Listen address, a device will assume its function once the ATN line is released. The one remaining address, 31 (binary 11111), is reserved to UNTalk (ASCII '__') or UNListen (ASCII '?') all devices. Note exceptions and limitations described in the IEEE-488 Functions section.

Secondary commands. This group (SCG—see Table 2) is distinct from the primary command group (PCG, which includes all other commands in this section). Table 2 indicates the characteristic bit pattern (X11) of Secondary commands. These are sent as a 2-byte

Table 3. Remote message coding*

Mnemonic	Message name	Type	Class	DIO 8	7	6	5	4	3	2	1	ATN	EOI	SRQ	IFC	REN
ATN	attention	U	UC	×	×	×	×	×	×	×	×	1	×	×	×	×
DAB	data byte	M	DD	D	D	D	D	D	D	D	D	0	×	×	×	×
DCL	device clear	M	UC	×	0	0	1	0	1	0	0	1	×	×	×	×
END	end	U	ST	×	×	×	×	×	×	×	×	0	1	×	×	×
GET	group execute trigger	M	AC	×	0	0	0	1	0	0	0	1	×	×	×	×
GTL	to to local	M	AC	×	0	0	0	0	0	0	1	1	×	×	×	×
IDY	identify	U	UC	×	×	×	×	×	×	×	×	×	1	×	×	×
IFC	interface clear	U	UC	×	×	×	×	×	×	×	×	×	×	×	1	×
LLO	local lock out	M	UC	×	0	0	1	0	0	0	1	1	×	×	×	×
MLA	my listen address	M	AD	×	0	1	L	L	L	L	L	1	×	×	×	×
MTA	my talk address	M	AD	×	1	0	T	T	T	T	T	1	×	×	×	×
MSA	my secondary address	M	SE	×	1	1	S	S	S	S	S	1	×	×	×	×
PPC	parallel poll configure	M	AC	×	0	0	0	0	1	0	1	1	×	×	×	×
PPE	parallel poll enable	M	SE	×	1	1	0	S	P	P	P	1	×	×	×	×
PPD	parallel poll disable	M	SE	×	1	1	1	D	D	D	D	1	×	×	×	×
PPR1	para. poll resp. 1	U	ST	×	×	×	×	×	×	×	1	1	1	×	×	×
PPR2	para. poll resp. 2	U	ST	×	×	×	×	×	×	1	×	1	1	×	×	×
PPR3	para. poll resp. 3	U	ST	×	×	×	×	×	1	×	×	1	1	×	×	×
PPR8	para. poll resp. 8	U	ST	1	×	×	×	×	×	×	×	1	1	×	×	×
REN	remote enable	U	UC	×	×	×	×	×	×	×	×	×	×	×	×	1
RQS	request service	U	ST	×	1	×	×	×	×	×	×	0	×	×	×	×
SDC	selected device clear	M	AC	×	0	0	0	0	1	0	0	1	×	×	×	×
SPD	serial poll disable	M	UC	×	0	0	1	1	0	0	1	1	×	×	×	×
SPE	serial poll enable	M	UC	×	0	0	1	1	0	0	0	1	×	×	×	×
SRQ	service request	U	ST	×	×	×	×	×	×	×	×	×	×	1	×	×
TCT	take control	M	AC	×	0	0	0	1	0	0	1	1	×	×	×	×
UNL	unlisten	M	AD	×	0	1	1	1	1	1	1	1	×	×	×	×
UNT	untalk	M	AD	×	1	0	1	1	1	1	1	1	×	×	×	×

** This listing is complete except for codes for Handshake and DAB variations. See Table 2 for code key.*

IEEE-488 BUS

Continued from page 96

sequence, the first byte of which is a Primary command such as Talk address (MTA), listen address (MLA), or Parallel Poll Configure (PPC).

SAD. The Secondary ADdress command is essentially a 2-byte address and is implemented only by extended Talkers (TE) and Listeners (LE). It permits 31 secondary addresses for every one of the 31 primary addresses, although its implementation is device dependent, and few devices have the capability.

PPE and PPD. The Parallel Poll Enable and Parallel Poll Disable Secondary commands are required for Parallel Poll capability (PP1,2—only occasionally implemented). The first byte of the sequence is the Addressed command: Parallel Poll Configure, followed by the PPE or PPD 'byte' (Table 3). See "Multiline Commands: (Addressed) Parallel Poll Configure."

Addressed commands (GET, GLT, PPC, SDC, and TCT). These Addressed commands (bit pattern X000—see Tables 2 and 3) are performed *only* by devices currently assigned as Listeners. Recall that the Standard does not require these devices to perform every bus function, although they must adhere to a specific, well-defined level of capability (Table 1).

GET. Group Execute Trigger (last 4 bits: 1000; ASCII 'cntrl-H') causes devices with the DT1 capability to be triggered simultaneously. See "IEEE-488 Functions: Device Trigger."

GTL. Go To Local (last 4 bits: 0001; ASCII 'cntrl-A'). A device with this capability (RL1,2) returns from Remote to Local (front panel programming control) upon receiving this command. A device will only go (back) to the Remote state when the Controller sends its Listen address (MLA) with the REN line asserted. If REN is released, all devices Go To Local. Remote and Local do not describe the device's participating on the bus; they indicate the source (front panel or 488) of its programming (for voltage range, triggering mode, etc.).

PPC. Parallel Poll Configure (last 4 bits: 0101; ASCII 'cntrl-E'). Parallel Polling is a complex capability (PP1,2), often not implemented, which is complimentary to the Serial Poll capability. In a Serial Poll the request for service is initiated by the device (using the SRQ line); a Parallel Poll is always initiated by the Controller and permits the status of many devices to be determined simultaneously and quickly. Conceptually and in contrast to Serial Polling, Parallel Polling is used for high throughput

Listing 1. A program to set up and acquire data using a DVM and scanner.

This section contains a canned routine which sets up the "hooks" from MBasic to the Pickles & Trout routines that drive the 488/S-100 board. First set aside (create) variable space in MBasic. The first CALL requires location (from CP/M) of P&T routines. It, in turn, passes the addresses of all of the variables to P&T for its access.

```
100 CNTL%=CNTLC%+TALK%+TALKC%+LSTN%+LSTNC%+SPOLL%+PPOLL%+DREN%
110 REN%=STATUS%+IFC%+BRSET%+IOSET%+PROTCL%+ECHO%
120 ERCODE%=TIME%+EOT%+EOS%+LENGTH%+POLL%+BUS%+ECHOIN%+ECHOOUT%
130 X=256*PEEK(7)+PEEK(6)+9
140 IF X>32767 THEN X=X-65536!
150 SETUP%=CINT(X)
160 CALL SETUP% (CNTL%,CNTLC%,TALK%,TALKC%,LSTN%,LSTNC%,SPOLL%,PPOLL%,DREN%,
  REN%,STATUS%,IFC%,BRSET%,IOSET%,PROTCL%,ECHO%,IOPORT%)
170 CALL IOSET%(ERCODE%,TIME%,POLL%,BUS%)
180 CALL PROTCL%(EOT%,EOS%,LENGTH%)
190 CALL ECHO%(ECHOIN%,ECHOOUT%)
200 CR$=CHR$(13) :  REM Assign contants:
210 SDC$=CHR$(4) :        REM Selective Device Clear = ^D
220 TRIGG$=CHR$(8) :      REM Group Execute Trigger = ^H
230 LLO$=CHR$(17) :       REM Local LockOut = ^Q
240 UNTALK$=CHR$(95) :    REM UNTalk address = ASCII "_"
250 UNLSTN$="?" :         REM UNListen address = ASCII "?"
260 DVM$="FL1Z0R3T3" :    REM String contains DVM program codes for:  Filter on;
270                       REM    Autozero off; 1 volt range; & single trigger.
```

We now can use the 488 bus to clear and set up the bus and devices.

```
300 CALL IFC% : REM Send Uniline message to cause InterFace Clear.
310 CALL REN% : REM Send (& maintain) Remote ENable Uniline message.
320 A$="6"+SDC$+LLO$ :  REM String has DVM Listen Address, Clear and Lockout
330 CALL CNTL%(A$) :    REM Act as Controller (ATN TRUE) & send string A$.
340 A$=DVM$ :           REM String contains DVM program codes.
350 CALL TALK%(A$) :    REM Act as Talker, send this set up message.
360 A$=UNLSTN$:CALL CNTL%(A$) : REM Act as Controller & UNListen all devices.
```

Obtain time/date from scanner clock and print and save it.

```
500 AC$="TD":GOSUB 800 :      REM "TD" prepares Scanner to send time/date.
510 A$="I":CALL CNTL%(A$) :   REM Scanner Talk address (tho remotely programmed
520    REM to send time, Scanner was not addressed to be a Talker in line 500).
530 CALL LSTN%(A$):TIME$=A$: REM Controller to act as Listener to hear time.
540    REM Data on the 488 bus will be stored in A$ by the P&T routines, & its
550    REM contents will change if the P&T LiSTeN routines are called again.
560 PRINT TIME$
```

Here we take DVM readings from Channel 0 and 1 of scanner. ERCODE% (line 700) is set by the P&T routines to various nonzero values which represent eight possible "error" conditions, such as the presence of an S-100 reset, another controller in the system, bus timeout error, Service Request asserted, no (handshake) Acceptors on the bus, etc.

```
600 AC$="AC0":GOSUB 800 :            REM Send "AC0" to Scanner: Close Channel 0
610 A$="6":CALL CNTL%(A$) :          REM "6" is the Listen address of the DVM.
620 A$=TRIGG$:CALL CNTL%(A$) :       REM TRIGG$ = addressed command: trigger.
630 A$="V":CALL CNTL%(A$) :          REM "V" is the DVM Talk address.
640 CALL LSTN%(A$) :                 REM Controller acts as Listener for DVM.
650 R(1)=VAL(MID$(A$,4,6)) :         REM Convert data from ASCII to numeric.
660 AC$="AC1":GOSUB 800 :            REM Send "AC1" to Scanner: Close Channel 1
670 A$="6"+TRIGG$:CALL CNTL%(A$) :  REM Condensed version of lines 610-630
680 CALL LSTN%(A$) :                 REM Controller to act as Listener for DVM.
690 R(2)=VAL(MID$(A$,4,6)) :         REM Convert data from ASCII to numeric.
700 IF ERCODE%>0 THEN GOSUB 1000 :  REM If 488 bus error goto error routine.
```

Note that we take the precaution of UNTalking all devices before we have our 488/S-100 interface act as a talker (TALK% - line 820, which requires only a software command and does not involve sending a new talk address over the bus). The two-Talker conflict can be created by controllers or talk-only devices. It is permissible for the 488/S-100 interface to act as a controller while the DVM is a talker, since all devices release the bus and pay attention when the controller asserts ATN.

```
800 A$=UNTALK$:CALL CNTL%(A$) :       REM UNTalks DVM.
810 A$=")":CALL CNTL%(A$) :           REM ")" is Listen address for Scanner.
820 A$=AC$+CR$:CALL TALK%(A$) :       REM Send prgm code string AC$ to Scanner.
830 A$=UNLSTN$:CALL CNTL%(A$):RETURN:REM Send UNListen command and return.
```

Continued from page 98

devices requiring a fast response (such as disk drives).

The Secondary command, Parallel Poll Enable (PPE), enables the device's response to a Parallel Poll by telling it how and under what conditions to respond. (The "how" is a 3-bit code sent on DIO1-DIO3 during PPE, which specifies which of the eight data lines the device should assert if it needs service when polled. The "what condition" tells a device to respond to a poll if its internal status "bit" matches the value of bit 4—DIO4—also sent during PPE.) Once enabled, a device will respond to Parallel Polling until it receives a Parallel Poll Unconfigure (Universal) command or the Parallel Poll Disable (secondary) command.

The actual poll is initiated by the Controller when it asserts TRUE on both the ATN and EOI lines, and all enabled devices will respond by sending a Parallel Poll Response (PPR) message by asserting TRUE on their assigned data line if they need service. Thus the status of up to eight devices can be uniquely determined simultaneously, and more than eight devices can be handled by data line sharing (a second Parallel Poll following reconfiguration would be necessary to identify which specific devices need service).

SDC. Selective Device Clear (last 4 bits: 0100; ASCII 'cntrl-D') is a capability (DC1 only) that will initialize a device's functions to a condition somewhat similar to power-on. See "IEEE Functions: Device Clear."

TCT. Take ConTrol (last 4 bits: 1001; ASCII 'cntrl-I') is a method for passing bus control among multiple Controllers. This command is unique in that the new Controller is first addressed to Talk (not Listen) prior to sending this command. Since few 488 systems employ more than one Controller, this is rarely used (or implemented).

Universal commands (DCL, LLO, PPU, SPE, SPD). Universal commands are received by all devices on the bus (not just those addressed to Listen), and are acted on if the device implements the specific function. Recall that the first several bits (X001) distinguish between the Addressed and Universal commands (Table 2).

DCL. Device CLear (last 4 bits: 0100; ASCII 'cntrl-T') is identical to the Addressed command Selective Device Clear, except that it affects all devices on the bus with this capability (DC1,2). See "IEEE-488 Functions: Device Clear."

LLO. Local LockOut (last 4 bits: 0001; ASCII 'cntrl-Q'). Requires RL1 capability and is primarily designed to augment the remote programming capability by inhibiting the front panel Go To Local button present on some devices. Note that during remote operation (RL1,2 capability), devices will not respond to front panel programming,

The Controller can dynamically reconfigure bus participation.

although the Go To Local button will remove them from Remote and restore front panel (Local) programming. Local Lockout must be issued with REN TRUE and will take effect whether the device is in Local (called 'Local With Lockout' state—all other front panel controls *will* be operative) or in Remote ('Remote With Lockout' state), permitting greater security against tampering. Note that if REN goes FALSE, all devices will exit both the Remote and Lockout states and return to the normal Local (without Lockout) state.

PPU. Parallel Poll Unconfigure (last 4 bits: 0101; ASCII 'cntrl-U') turns off the entire Parallel Poll capability of all devices and is the condition of the bus on power-up. See "Addressed Command: PPC."

SPE. Serial Poll Enable (last 4 bits: 1000; ASCII 'cntrl-X') is used by the Controller to determine the Source of an SRQ (Service Request). The proper sequence involves sending an UNListen command (to prevent other devices from listening to the Serial Poll response) and an SPE. The controller then sends a Talk address and listens (ATN FALSE) for a byte response, ending this sequence by sending Parallel Poll Disable when it finds the source (there may be more than one) of the SRQ. An UNTalk command should be sent to disable the last Talker. The Serial Poll capability is a Talk function subset (T & TE 1,2,5,6).

The response from a device asserting SRQ will be a data byte with bit 6 (DI06) asserted (known as a Request Service (RQS) response—see Table 3).

IEEE-488 BUS

Continued from page 101

The other bits can represent a variety of internal status conditions (such as data ready, error, or front panel SRQ button pressed), permitting the reason for the SRQ to be determined along with the requestor's identity. Some devices have a programmable mask to determine which of the status conditions will result in an SRQ.

SPD. Serial Poll Disable (last 4 bits: 1001; ASCII 'cntrl-Y') ends a serial poll initiated by the SPE.

Device-dependent messages

Device-dependent messages are simply bytes sent in parallel by a Talker (ATN FALSE) and received by one or more Listeners. This byte can represent:

1. Normal data (measurements by a voltmeter sent to a printer; floppy disk data sent to a microprocessor; etc.)
2. End-of-string indication
3. Device status information (e.g., a Serial Poll status byte)
4. Program codes (usually sent by the Controller acting as a Talker) which alter the internal operation of the device (range, triggering, etc.).

The device dependency comes in as a result of the different forms and uses to which data can be put. The data can be normal data or device programming codes. Data can be binary, ASCII, packed BCD, etc., and it is up to the Talker and Listeners to be in agreement. Devices also differ in their response to or handing of leading or trailing spaces, a negative zero ('-0'), inconsistent use of upper- and lower-case letters, rounding or truncation of digits, etc. Not all devices handle these situations thoughtfully or gracefully.

End-of-string indicators. The Standard does not specify what a device must do to indicate the end of a string. Generally, one or both of the following techniques are used by most devices: *(1)* The EOI line is asserted by the Talker when the last byte of the string is put on the data lines; *(2)* A <cr> (carriage return) or <cr> <lf> (line feed) terminates the string; and *(3)* the string is assumed to be a predetermined length. Some Controllers (or associated software) also have a timeout mechanism that is generally used to break the microprocessor out of its wait loop should something go awry.

This ambiguity can pose problems. For example, I use a HP 3456A system voltmeter which, fortunately, offers several options. Normally, it sends 12 ASCII data bytes followed by a <cr> <lf>, with the EOI line raised when the <lf> is sent. EOI can be disabled. If multiple readings (up to 350) are acquired and then sent on the bus, the ASCII readings are separated by commas, with the end-of-string indication at the end. A packed format is also available, which sends 4 bytes and uses only EOI (no <cr> or <cr> <lf>). In the multiple reading mode, no delimiters or EOI are used until the last byte is sent. Those with a passing familiarity of Microsoft Basic can readily conjure up interesting effects.

Fortunately, most devices provide some flexibility in adapting to this situation. For example, some devices have switch or software options. Also, the software that comes with the Pickles & Trout interface contains explicit commands that determine what it should listen for and send to represent the end of the string.

Device program codes. Program codes are generally sent by the Controller (acting as a Talker) to a device addressed as a Listener. There is no standard regarding how a device will respond to a program code sequence of 'FLOR4T3'. The recent IEEE Standard 728-1982 [4] contains "Recommended Practice for Code and Format Conventions," but adherence is not required, and many existing devices do not conform. However, this will be a problem only for those who wish to remember the program codes for a variety of devices.

Powering up a device like the Controller can disrupt the bus.

To provide an example: the HP 3456A DVM can be programmed over the 488 bus for function (F1-F5), range (R1-R9), auto zero off/on (Z0-Z1), filter off/on (FL0-FL1), various internal math functions (M0-M9), trigger mode (T1-T4), and a variety of other miscellaneous instrument functions. Sending the ASCII character sequence of 'F1R3Z1FL1T3' to the DVM (previously addressed as a Listener) would cause it to be set for DC volts, 1 volt scale, auto zero and filter on, and single Trigger. The DVM will implement each change as soon as it receives a single complete code (e.g., 'F1').

Continued from page 102

The same codes sent to an HP 3497A scanner are entirely meaningless, although no bus errors will result (the scanner, although not the DVM, will beep at you to notify you of its distress under these circumstances).

An example in Microsoft Basic

To convey a better feeling for the operation of the bus and the 488/S-100 interface, an example is in order. Let us imagine that we have a HP 3456A DVM and a HP 3497A scanner with clock. We wish to power up the devices, program them remotely, and gather data. First, however, some preliminary information.

Powering up a device (especially the Controller) can disrupt the bus, so it is wise to have (and leave) all devices turned on that will ever be needed during a sequence of bus operations. Note that the DVM can be remotely programmed to wait until its current reading is transferred over the bus before continuing with another reading (other devices may be made this way by default). The lack of a Listener to handshake the current reading will put the DVM in limbo. Thus the following general sequence is usually necessary:

1. Power up all devices that will be needed. Act as a Controller and send out an IFC and then REN (if the devices are to be remotely programmed).
2. Act as a Controller and send out the Listen address for each device. Then, acting as a Talker, send out program codes for that device.
3. Act as a Controller and configure the bus with a Talker and Listeners, then become inactive or participate in the bus (as a Talker or Listener). Note that as a Listener, the Controller can receive a data byte and (while holding up the final step of the handshaking process), go and process or store it to ensure that no data are missed. When inactive, the Controller will not assert any of the handshaking lines, and thus the bus proceedings will continue without its participation.
4. Be prepared to re-address a Talker and Listeners, respond to a Service Request (SRQ), etc., as needed.

A program to accomplish this is shown in Listing 1.

Final comments

My experience with the 488 bus has been excellent. My system employs a Pickles & Trout 488/S-100 interface board and has performed laboratory instrument control and data acquisition on an essentially continuous basis for four years. About a year ago I switched from CDOS (a CP/M derivative from Cromemco) to Cromix (a UNIX-like multiuser, multitasking operating system from Cromemco); the few problems encountered were very short lived.

The ease and viability of the transition attests to the versatility and excellent design of the 488 and S-100 buses, as well as the particular components I have used.

References

1. *IEEE Standard Digital Interface for Programmable Instrumentation, ANSI/IEEE Std. 488-1978.* The Institute of Electrical and Electronics Engineers, Inc. (345 E. 47th St., New York, NY, 10017; 212-705-7900), 1978.
2. *Tutorial Description of the Hewlett-Packard Interface Bus.* The Hewlett-Packard Company: San Diego, CA, 1980.
3. *Introduction to the GPIB.* Wavetek: San Diego, CA, 1979.
4. *IEEE Recommended Practice for Code and Format Conventions for Use with ANSI/IEEE Std., 488-1978:* The Institute of Electrical and Electronics Engineers, Inc. (345 E. 47th St., New York, NY, 10017; 212-705-7900), 1982.

Peter L. Andresen, General Electric Co., Corp. Research & Development Center, Schenectady, NY 12345