ES

....$ 69.88
..... 99.88
... 169.88
...... 99.88
... 289.88

...$ 54.88
..... 79.88
.... 69.88
...... 64.88
...... 34.88
...... 89.88
...... 34.88
...... 99.88
... 169.88
..... 44.88

...$ 54.88
..... 34.88
... 139.88
..... 34.88
..... 29.88
..... 34.88
..... 34.88
..... 44.88

..$ 29.88
.... 54.88
.... 54.88
.... 10.00
.... 34.88
.... 99.88
.... 59.88

rdware

..$ 34.88
... 24.88
... 79.88
.. 369.88
... 69.88
.. 119.88
... 44.88
.. 259.88
.... CALL
.. 499.88
.. 349.88
... 94.88

S DESK
855
R DESK
726
T, Mon-Fri
on orders
for shipping).
Express,
edit cards
ls
until your
days for
n orders.
an exact
unds issued

IE

) 343-0726

uiry 36

INTERFACING FOR DATA ACQUISITION

BY THOMAS R. CLUNE

A comparison of three interfaces

THE USE OF MICROCOMPUTERS for data acquisition in the sciences is surprisingly limited. It is widely recognized that the need for such applications exists. But I discovered in my experience at Brandeis University that most researchers have either had bad experiences with data acquisition on minicomputers or simply don't feel that they have the time to learn what they would need to know to retool their labs. Nonetheless, the advantages of computerizing are so substantial that microcomputer-based data acquisition is slowly moving into the lab. In this article, I'll share some of my experience with different approaches to computerizing data acquisition. Since I find the IEEE-488 to be the most versatile option for laboratory data acquisition, I will devote a fair amount of time to explaining that interface. My hope is that my experience may ease the problems that you might encounter in computerizing your setup.

THE PROBLEM

There are three basic reasons why microcomputers are so important in the context of data acquisition. First, for a minicomputer or mainframe to be affordable, its use must be shared by more than one person, but in data acquisition it is crucial to have the computer's attention when the data is ready. Microcomputers make single-user systems affordable. Second, mainframe computers are generally not located in the laboratory. Thus, in any but very low speed data-acquisition contexts, there is a communications bottleneck created by the data transmission. Third, there is no common standard for interfacing with laboratory instruments on mainframes, so each laboratory setup presents substantial and individual problems of design and implementation that exacerbate the financial and logistical difficulties.

At least one other concern is fueling the drive toward computerization in the lab: The cost of turnkey instruments has become so high that most institutions are unable to afford the state-of-the-art equipment needed to conduct research. This is particularly irritating because most instruments in the sciences have essentially the same components. You end up paying over and over again for a built-in chart recorder, a waveform digitizer, a monochrometer, a photomultiplier, etc. And when the new generation of an instrument comes out with a broader dynamic range or some other improvement in one component, the entire turnkey instrument must be replaced. We simply can't afford to pay for research done that way any more. With the availability of microcomputers, we don't have to. We can tie chart recorders, waveform digitizers, and whatever else we need together into a dedicated instrument and recycle the components as the field or our research evolves.

Thomas R. Clune is a BYTE *technical editor. Before coming to* BYTE, *he was the physical-chemistry lab coordinator at* Brandeis *University, where he taught data acquisition by microcomputer. He can be contacted at* POB 372, *Hancock*, NH 03449.

A/D CONVERTERS

The least expensive way to automate a lab is with an analog-to-digital (A/D)

(continued)

The speed of a transient tracked by D/D equipment is not limited by the computer's throughput.

converter. There are, however, a number of limitations to this approach. First, an A/D converter samples only one voltage source at a time. Typically, an experiment requires correlating one reading to others for the same instant of time (e.g., pressure versus temperature at time *t*). If the time requirements are sufficiently lax, that is, if readings taken 10 or 20 microseconds apart can be treated as simultaneous, an A/D converter may be acceptable. But often this time lag is sufficient to make the data hopelessly imprecise. The second problem with A/D converters is that they are slow. The maximum sampling rate on most "high-speed" A/D converter boards is 100 kilohertz (kHz). Practically speaking, this means that you can't track a transient of greater than approximately 20 kHz. Much of scientific data acquisition now requires at least the ability to track a transient of a few megahertz. A third problem with A/D converters is that, because the boards are made to be inexpensive, their linearity is not very good. A 12-bit board may have an effective resolution of only 7 or 8 bits. Finally, A/D converters are very susceptible to noise in a lab. Commonly, the cabling will be either twisted-pair or ribbon cable—very good antennae. In a well-designed board, the cabling is simple coax, which may still not give the level of noise immunity required in a laboratory environment.

Nonetheless, an A/D converter is a good buy if it will do your task. My feeling is that the best use of an A/D converter is to connect it to the chart output of a stand-alone instrument. Instead of junking a high-quality analog instrument in the interests of modernizing, use the capabilities available in your lab now. One big advantage of this kind of setup is that you can use a very slow A/D converter. This is desirable for two reasons: first, a slow A/D converter will be better made than a comparably priced high-speed board, and second, since you will only need a 30-Hz-or-so A/D converter, most noise in the lab will be too fast for the A/D converter to respond to it. Further, your low-pass filter will be able to cut out line voltages, which are an inevitable source of noise in any lab.

D/D AND RS-232C

If an A/D converter won't meet your needs, you need stand-alone instruments that can transfer digital information to the computer via a digital-to-digital (D/D) interface. The first advantage of D/D over A/D is that data may be analyzed at high speed and the digital "snapshot" of the analysis stored in a buffer of a few kilobytes on the stand-alone instrument. The buffer data can then be downloaded to the computer at whatever speed the interface will support. That is, A/D conversion necessarily requires real-time analysis, whereas the speed of a transient that can be tracked by D/D equipment is not limited by the microcomputer's throughput. Of course, speed of data transfer is still important because it determines how quickly the instrument can repeat an analysis.

D/D interfaces come in two flavors: serial, which transfers information a bit at a time; and parallel, which transfers data a word (commonly one byte) at a time. The most common serial port is an RS-232C interface.

There is a lot to dislike about the RS-232C. First, it is not standard. There are two ends to an RS-232C interface: the DTE (data-terminal equipment) end and the DCE (data-communications equipment) end. Often the two instruments you want to hook together will both be configured as DTEs, so you will probably have to create a cable that matches your particular setup once you find out what it is. Second, the only handshaking provided is on the level of whole messages. The interface does not verify that data has been received before proceeding. It is very easy to lose data on this interface. Third, RS-232C is a notoriously noisy interface—perhaps no worse than an A/D converter, but that isn't saying much. Fourth, RS-232C is slow. Since it sends only one bit at a time, it has a built-in speed disadvantage over parallel interfaces. And interference is an increasing problem with increasing transmission rates (as is true of any system). Finally, RS-232C is able to connect only two devices together. Thus, coordination and control of multiple data sources requires more than one RS-232C port on the computer and makes for devilishly difficult software integration.

The strong points of RS-232C are twofold. First, it is capable of transmitting information over long distances by telephone. Second, it is the only interface available on some older instruments. If you have to use it, you learn to live with it. But you'll never learn to love it.

IEEE-488

The IEEE-488 is a byte-serial, bit-parallel interface that overcomes the problems of the interfaces outlined above. First, the interface is incredibly resistant to interference. For example, at the Brandeis University chemistry department, we used the interface in a pulsed-nitrogen-laser experiment and found that the data transmission was unaffected by noise in any environment where the computer itself was able to function. Figure 1 shows the physical layout of the cable that provides such excellent noise immunity.

The second virtue of IEEE-488 is that the interface has a bus structure. That is, you can interface up to 15 devices at a time using the same board. This structure simplifies process control and allows true simultaneous data acquisition, as we shall see presently.

(continued)

Third, the interface is fast for a micro. Data can be transferred at up to 1 million bytes per second (using special tristate drivers on the lines) and without any special care will support transmission rates of about 250K to 300K bytes per second using DMA (direct memory access).

Fourth, the interface is standard and widely available. All IEEE-488 instruments are plug-compatible, and the interface is available on every major kind of laboratory device. Over 2000 devices are currently available with an IEEE-488 interface. Given that the standard was not set in its current form until 1978 and that there is a lag between specification and implementation, the rapid adoption of the standard gives an indication of how sorely needed it was.

The primary limitation of the standard is that the total cable length on an installation cannot exceed 20 meters without special (and expensive) repeaters. In practice, you will seldom need to exceed that length. And given that long cabling slows transmission rates and is more susceptible to noise, you generally do better to keep the cabling short anyway.

The Standard Explanation

The IEEE-488 standard is relatively involved because it accommodates a wide variety of uses. In the rest of this article, I'll examine the standard and then take a close look at a setup using the interface.

IEEE-488 began life as the General-Purpose Interface Bus (GPIB) of the Hewlett-Packard Corporation. In 1975, the IEEE adopted the GPIB as its standard. Some minor modifications were made to the standard in 1978, but IEEE-488 still goes by the name GPIB on HP products.

Devices on the interface may perform three kinds of functions. They may be talkers; that is, they may transmit data to other devices on the interface. Of course, there can be only one active talker at any given time. Alternatively, a device may be a listener—it may receive data or instructions from another device on the interface. There may be more than one active listener on the interface at any given time. And a device may act as a controller, a coordinator of which device may talk when and which devices may listen. Finally, a device may do nothing but stand by. A device may, at different times, assume any of the above functions.

The interface supports two modes of operation: command and data. As the name suggests, the command mode is for process control. For example, if one of the devices on the interface is a digital multimeter (DMM), the controller may program the DMM for reading DC voltages in the 3-volt full-scale deflection range. In the data mode, data is transferred from talker to listener(s) over the interface.

The interface has 24 lines, 8 of which are ground lines. The other 16 are divided into three groups: 8 bidirectional data lines, 3 data-byte control lines (handshake lines), and 5 general interface-management lines.

The three-line handshake protocol functions as follows: When information is going to be transferred over the bus, the listeners must be ready to receive the data. If they are not, they signal NRFD (not ready for data) by

(continued)

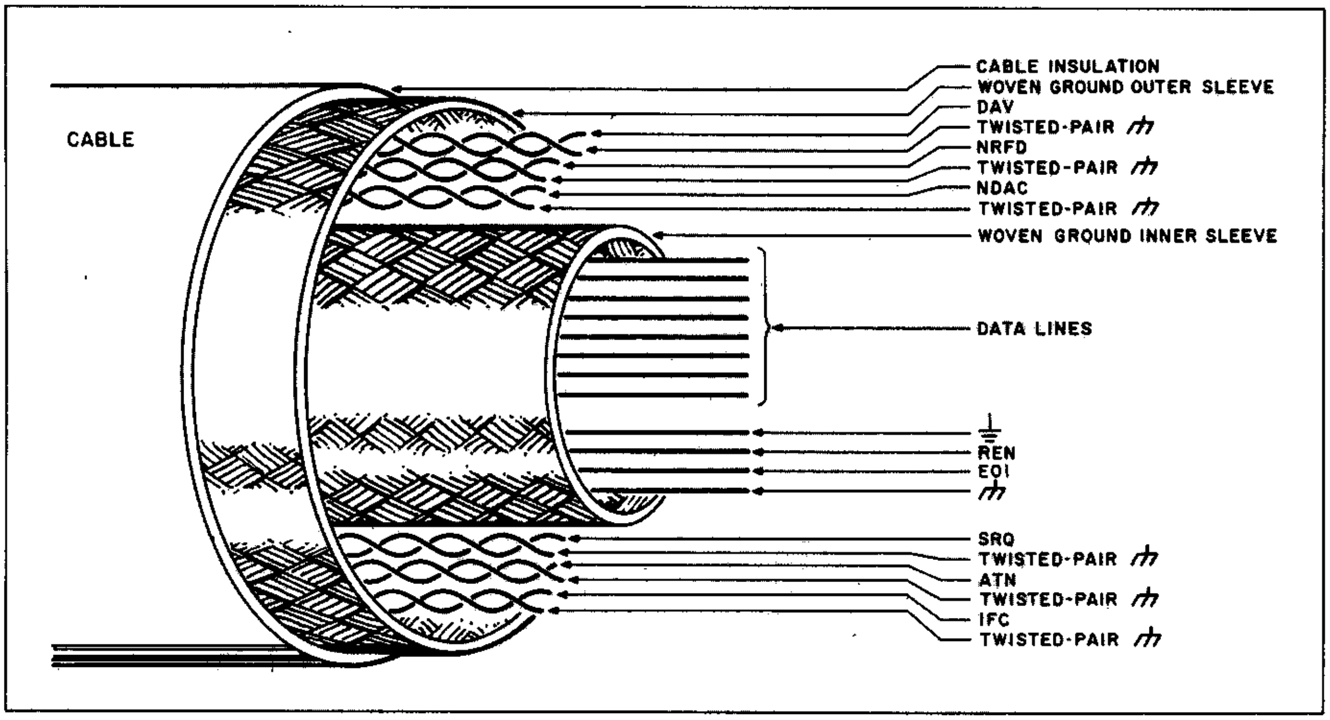


Figure 1: *Cutaway view of an* IEEE-488 *cable. Notice the large number of grounds for shielding.*

pulling the NRFD line low (low is defined as true by the IEEE-488 standard). The NRFD line has an open-collector design, so if any one listener is not ready, the line is kept low. When all the listeners are ready, the NRFD line goes high. If the talker is ready to transmit data, it sets the DAV (data valid) line low. The transition of the DAV line triggers the resetting of the NRFD line and the listeners pick up the latched byte of data. When each listener receives the data, it releases the NDAC (not data accepted) line, which is also open-collector. When all listeners have received the data, the NDAC line goes high, causing a reset of the DAV line, which in turn triggers the resetting of the NDAC line. This sequence, outlined in figure 2, is repeated for each byte in a transmission. It may not be immediately apparent why three lines are useful in this sequence. At first glance it appears that the DAV and NDAC line would accomplish everything necessary for the transmission of data. However, the NDAC line is released as soon as the IEEE-488 board of the listener has received the data. The information must still be downloaded from the IEEE-488 data register to, for example, the computer's main memory to be stored more permanently. By releasing the talker as soon as the data has been transferred, the talker becomes free to prepare the next byte for transmission at the same time that the listeners are "digesting" the last byte, so the rate of information transfer may be maximized. The NRFD line is thus necessary to prevent the possibility of a listener's data register being prematurely overwritten.

Since each byte of data transferred is a self-contained event on the interface, there must be some way of signaling the end of a data-transfer sequence. This may be done in two ways. The one I will mention here is to use one of the bus-management lines, the EOI (end or identify) line. When a talker sets this line, it signals that the data-transfer sequence is complete.

The "identify" in EOI applies to the controller's use of the line. If the interface is to be used for process control, there must be a way for the controller to monitor the "fitness for duty" of the various devices. One way it may do so is by conducting a parallel poll of the devices. If the controller asserts ATN (attention) and EOI, each device responds by using one data line to say whether or not it has any problems. If one does, the computer (the controller) can query that device further to determine the precise nature of the difficulty. The limitation of a parallel poll is that the controller must initiate the inquiry. IEEE-488 also provides for a serial poll, in which a device in trouble may alert the controller that all is not well by asserting SRQ (service request). The computer then can ask each device in turn what its status is to determine the source and nature of the problem.

ATN serves another, more general purpose as well. Any time the controller asserts ATN, it can change the function of a device from, say, talker to listener. When ATN is asserted, the board goes into the command mode. All subsequent information is control data. In general, control information will apply to only some of the available devices. How is the information restricted to only the appropriate devices' attention? Each instrument on the interface can be assigned a unique 5-bit address, generally by DIP (dual-inline package) switches on the backplane of the instrument. Valid addresses are numbers up to and including 30. When the computer wants to address its control data to a specific set of devices, it asserts ATN and outputs a list of the appropriate address numbers (notice that the same string of outputs would be treated as data were the board not in the command mode). Table 1 shows the protocols of the computer addressing for different functions. If a device is being told to listen to control information, an addressed command follows its address-to-listen call. Addressed control information defined by the IEEE-488 standard includes GTL (go to local), which releases a device from remote control; SDC (selected-device clear), which resets a device to its default setting; PPC (parallel-poll configure), which is used to assign a data line to a device for answering a parallel poll; GET (group-enable trigger), which initiates simultaneous data acquisition by each addressed device; and TCT (take control), which passes control of the bus management from the present controller to the specified device.

Two other kinds of multiline commands are shown in table 1. First is a secondary address. This is information after the primary address that configures a device for a particular kind of operation. This is one way that a DMM may be set for DC volts, for example. The primary address specifies the DMM device number, and the secondary address specifies the DC voltmeter function in the DMM. The significance of secondary addresses is not part of the standard. Each manufacturer decides whether

(continued)

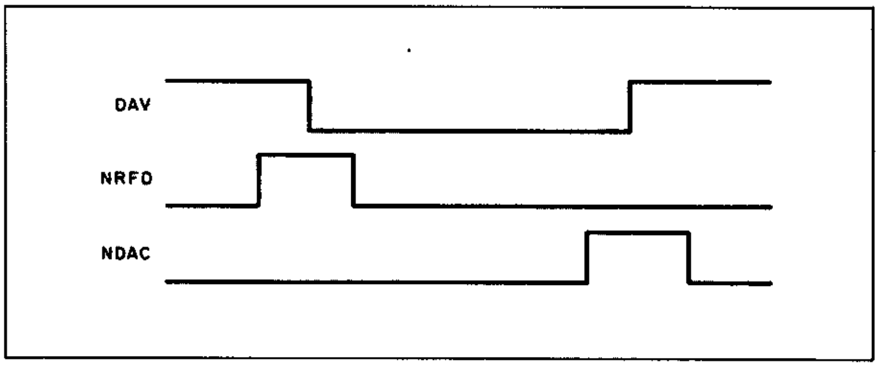


Figure 2: *The logic flow of the* IEEE-488 *handshake sequence. Low is true.*

Inquiry 296

Pascal and C
Programmers

Your programs can now compile the

FirsTime™

FirsTime is an intelligent editor that knows the rules of the language being programmed. It checks your statements as you enter them, and if it spots a mistake, it identifies it. *FirsTime* then positions the cursor over the error so you can correct it easily. *FirsTime will identify all syntax errors, undefined variables, and even statements with mismatched variable types.* In fact, any program developed with the *FirsTime* editor will compile on the first try.

More than a syntax checker!
FirsTime has many unique features found in no other editor. These powerful capabilities include a zoom command that allows you to examine the structure of your program, automatic program formatting, and block transforms.

If you wish, you can work even faster by automatically generating program structures with a single key-stroke. This feature is especially useful to those learning a new language, or to those who often switch between different languages.

Other Features: Full screen editing, horizontal scrolling, function key menus, help screens, inserts, deletes, appends, searches, and global replacing.

Programmers enjoy using *FirsTime*. It allows them to concentrate on program logic without having to worry about coding details. Debugging is reduced dramatically, and deadlines are more easily met.

FirsTime for PASCAL	$245
FirsTime for C	$295
Microsoft PASCAL Compiler	$245
Microsoft C Compiler	$395
Demonstration disk	$25

Get an extra **$100 off** the compiler when it is purchased with ***FirsTime.***
(N.J. residents please add 6% sales tax.)

Spruce
Technology Corporation
110 Whispering Pines Drive
Lincroft, N.J. 07738
(201) 741-8188 or (201) 663-0063

Dealer enquiries welcome. Custom versions for computer manufacturers and language developers are available.

FirstTime is a trademark of Spruce Technology Corporation.

to use secondary addresses and, if so, what they will mean. The last kind of multiline command is a universal command. Reasonably enough, universal commands apply to all devices on the bus and are therefore not preceded by an address list. The universal commands defined by the standard include LLO (local lockout), which disables instrument front-panel control; DCL (device clear), which resets all devices to their factory-selected default states (this is the universal version of SDC); PPU (parallel-poll unconfigure), which deactivates parallel polling; SPE (serial-poll enable), which initiates a serial poll; and SPD (serial-poll disable), which terminates a serial poll.

The logical difference between the uniline commands and the multiline commands is that uniline commands are unconditioned. That is, they operate immediately instead of requiring that the bus be in command mode. The last two uniline bus-management lines illustrate the need for such immediacy. REN (remote enable) places a device under computer control. When a device is first going to be addressed by the computer, this provides the "warm boot" needed to get its attention. IFC (interface clear) is the "panic button." When the controller asserts IFC, the active talker must immediately relinquish control of the data lines to the computer. As you can see, the standard is rather involved. But it is not complete.

HPIB

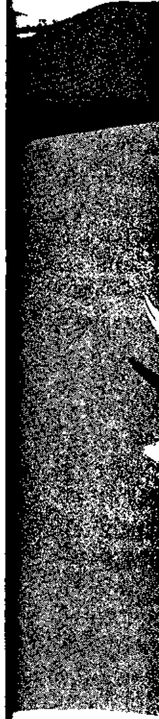
The IEEE-488 standard ensures electrical compatibility among instruments, but it does not insure that two instruments will understand each other. The analogy has been drawn between IEEE-488 and the telephone system: You can call Rome on your telephone, but you may not understand what the person who answers the phone is saying. Similarly, the IEEE-488 standard does not specify the code that is to be used by instruments in transmitting data. Some instruments speak binary, some speak ASCII, etc. The Hewlett-Packard Corporation has developed a software standard for IEEE-488 data *that is not universally employed*. However, it is the most common format for data transfers on the bus. The protocol is called the Hewlett-Packard Interface Bus (HPIB). GPIB and HPIB are often used interchangeably, but strictly speaking GPIB is the IEEE-488 standard and HPIB is the conformance to HP's software protocol. HPIB specifies the following:

1. All information is transferred in ASCII code.
2. Information is transmitted "left to right"; that is, "C A T" is transmitted "67 65 84," not "84 65 67."

(continued)

Table 1: IEEE-488 *interface management command bit protocols. These apply only when the controller asserts* ATN. A=*address bit*, C=*command bit*, S=*secondary address bit*, N=*not used.*

Data Lines Bit	Significance
7 6 5 4 3 2 1 0	
N 0 0 0 C C C C	addressed command
N 0 0 1 C C C C	universal command
N 0 1 A A A A A	address to listen
N 1 0 A A A A A	address to talk
N 1 1 S S S S S	secondary address



If you'r
graphs
got a pl

High sp
The ne
enough
plots in
ors at a
second,
and rep
units co

2780 Lomita

Inquiry 107

The significance of a device's program data is determined by the manufacturer, not IEEE-488.

3. All sequences of data transmission end with ASCII 13 (a carriage return) and, optionally, ASCII 10 (a linefeed) instead of using the EOI line.

The advantage of the standard is that data can be fed directly to a printer to produce properly formatted output in continuous-data-collection applications. Of course, the biggest advantage of the standard is simply that it is a standard.

USING THE INTERFACE

So much for the standard. Now let's take a look at how to use it. Manufacturers of IEEE-488 interface boards provide interface drivers for you, so using the interface is easier than learning about the standard in the first place. Usually the interface driver is a set of assembly-language routines that you can call. In high-speed applictions you will want an assembly-language driver. But in the program I provide here (listing 1), I use an interpreted BASIC driver. The program is taken from a course in interfacing I taught at Brandeis University. It is used to calculate the lattice energy of solid argon from temperature and pressure data pairs. This is a low-speed application, with readings being taken every 30 seconds. Thus, an interpreted BASIC interface driver will provide adequate speed. A further benefit to me is that students can study the driver routines to understand how the interface works. Tecmar also makes an assembly version of its interface driver.

The equipment used in this experiment includes an IBM PC with 128K bytes of memory, a Tecmar IEEE-488 interface for the PC, two HP 3478A DMMs with IEEE-488 installed, a copper-constantan thermocouple wire, and a Barytron 220 pressure transducer. The program listing includes only the data-acquisition part of the program, and Tecmar's interface driver routine is not reprinted here. Before the experiment can be run, the DMMs must be set to their respective addresses (17 and 19) by DIP switches on the DMM backplanes.

The program is largely self-explanatory. I will limit my remarks on it to points that the listing may not make sufficiently clear. Notice the statement BD.ADDR%=&H310 in line 40. This initializes the beginning memory location of the 16-byte buffer used for communication between the IEEE-488 interface and the computer. MY.ADDR%=1 in line 60 declares that the computer's device address number will be 1. Both these variable names are specified by the driver software. Line 110 shows the way that the Tecmar driver routine is invoked. The routine begins at line 10000 and is merged with your application program. PARAM$ is the variable name for any parameter to be passed to the driver routine. In this case, the operation performed is initializing the IEEE board for controller operation. In line 130, ADTR is the mnemonic for asserting REN, to let the DMMs know that they are connected to and will be controlled by the computer. Line 150 contains the information to be output to the DMM that will monitor the pressure transducer. The significance of this data is determined by the DMM manufac-

(continued)

Listing 1: A *sample data-acquisition routine using the* IEEE-488 *interface.*

```
 10        REM IEEE-488 PROGRAM FOR HEAT OF SUBLIMATION OF SOLID
                ARGON. PROGRAM SHOULD BE MERGED WITH TECMAR
                IEEE-488 SOFTWARE VER. 3.
 20        REM PROGRAM BY THOMAS CLUNE, BRANDEIS UNIVERSITY
                CHEMISTRY DEPARTMENT
 30        REM DMM #19 READS THE THERMOCOUPLE, DMM #17 READS THE
                PRESSURE TRANSDUCER.
                INITIALIZE IEEE-488 BUFFER LOCATION, DIMENSION ARRAYS
 40 BD.ADDR%=&H310:DIM PRES(250):DIM TEMP(250)
 50        REM SPECIFY COMPUTER DEVICE NUMBER, INITIALIZE DATA
                POINTER
 60 MY.ADDR%=1:DPT=1
 70        REM WAIT UNTIL READY TO BEGIN RUN.  PRESSURE READINGS
                MUST BE POSITIVE AND THERMAL EQUILIBRIUM MUST BE
                REACHED BEFORE THE RUN BEGINS.
 80 CLS:PRINT "PRESS ANY KEY WHEN YOU ARE READY TO BEGIN YOUR
    RUN"
 90 A$=INKEY$:IF A$="" THEN 90
100        REM INITIALIZE BOARD WITH COMPUTER AS CONTROLLER
110 PARAM$="INIT.C/":GOSUB 10000
120        REM SET BOTH DMM'S FOR REMOTE CONTROL BY COMPUTER
130 PARAM$="ADTR/":GOSUB 10000
140        REM SET INTERRUPT REGISTERS OF DMM'S FOR SYNTAX ERROR
                AND FRONT PANEL SRQ.
150 DATA.STRING$="KM24D2PRESSURE"
160 PARAM$="WR.STR/17/14///":GOSUB 10000
170 DATA.STRING$="KM24D2TEMPERATURE"
180 PARAM$="WR.STR/19/17///":GOSUB 10000
190        REM ENTER DATA.STRING$ AND WRITE PROGRAMMING
                INFORMATION TO DMM #17. ADD <CR> FOR EOS.
200 CLS:INPUT "ENTER COMMAND STRING FOR PRES. DMM
    (#17)";DATA.STRING$
210 DATA.STRING$=DATA.STRING$+CHR$(13)
220        REM OUTPUT DATA.STRING$ TO DMM
230 PARAM$="WR.STR/17//13/EOS/":GOSUB 10000
```

(continued)

Inquiry 193

magnum p.c.

800-544-4354

VISA GA Residents (404) 441-3112 MasterCard

5965 PEACHTREE CORNERS E, B-2, NORCROSS, GA 30071

COMPUTERS

IBM PC YOU CONFIGURE.....................CALL
SANYO MBC & SUPER..........NEW LOW PRICES

BOARDS

AST SIX PAC PLUS & MONOGRAPH..........CALL
EVEREX GRAPHICS EDGE.........................399
HERCULES GRAPHICS CARD....................335
IRMA 3278 EMULATOR................BEST PRICE
MICROTEK NEW PRODUCTS..................CALL
ORANGE MICRO ALL BOARDS................CALL
ORCHID PC BLOSSOM & TURBO.............CALL
PERSYST BOARDS & CARDS..................SAVE
QUADRAM ALL PRODUCTS......................CALL
STB GRAPHIX PLUS II.............................CALL
TECMAR ALL PRODUCTS........................CALL
TITAN ACCELERATORS FOR IBM OR APPLE..CALL
TSENG LABS ULTRA PAK..........................NEW

DISK DRIVES

FULL HEIGHTS (FOR IBM)...............FROM 110
HALF HEIGHTS (FOR IBM)..............FROM 125
ALPHA OMEGA TURBO 10 (FASTER THAN XT)800
EVEREX 10 MEG INTERNAL HARD DISK.......750
1/2 HT 1/4'' TAPE STREAMER...........999
IOMEGA BERNOULLI BOX (20 MEG)..........2795
MICRO SCI (FOR APPLE).......................CALL
PCjr 2ND DISK DRIVE OR HARD DISK.........CALL
PEACHTREE PERIPHERALS.....................CALL
QUADDISK 6 MEG REMOVABLE...............1725
QUENTIN (FOR APPLE & IBM).................SAVE
SUPER 5 (FOR APPLE)............................179
TALLGRASS HARD DISK WITH BACKUP.....CALL
TANDON TM100-2..................................189

PRINTERS

C. ITOH LETTER QUALITY......................CALL
EPSON FX, RX.......................................SAVE
FUJITSU ....CALL
NEC SPINWRITER, P2 & P3....................CALL
OKIDATA ALL MODELS..............BEST PRICES
PANASONIC ....CALL
QUME ....SAVE
STAR MICRONICS GEMINI 10X/15X.....269/379
TOSHIBA P1350 & P1340.......................CALL

MONITORS

AMDEK...CALL
PRINCETON GRAPHICS..........................SAVE
QUADRAM QUADCHROME.......................CALL
SUPER 5 ....CALL
TAXAN ALL MODELS.............................CALL

SOFTWARE

COPY II PC........SAVE	**D-BASE III**..........399
FRAMEWORK......399	**JANUS/ADA**......SAVE
LOTUS 1-2-3.....SAVE	**MULTIMATE**........279
SYMPHONY........475	**10 BASE**.........CALL

MODEMS

HAYES SMART MODEM 300/1200.......212/499
NOVATION SMART CAT PLUS W/MITE........399
VEN-TEL HALF CARD 1200 BAUD.............425
PENRIL...........SAVE **MAXWELL**........NEW

ACCESSORIES

BUFFERS/SPOOLERS ....CALL
CHIPS 64K - SET OF 9..............................39
COMPUSERVE STARTER KIT......................32
2 & 4 WAY SWITCH BOXES PAR. & SERIAL..SAVE

DISKETTES

VERBATIM/MAXELL .SAVE	CASE 100	BOX 10
BASF 5¼ SS/DD	145	17
5¼ DS/DD	170	20

AVAILABILITY AND PRICES SUBJECT TO CHANGE

APPROVED CORPORATE ACCOUNTS WELCOMED

```
240       REM CHECK FOR SYNTAX ERROR IN DATA.STRING$.  IF YES, LOOP
                    BACK TO REENTER DATA.STRING$
250 PARAM$="RBST/":GOSUB 10000
260 IF SRQ%=1 THEN PARAM$="SER.POLL/17/":GOSUB 10000:IF
    POLL.RESP% AND 4=4 THEN PRINT "SYNTAX ERROR IN
    COMMAND":SRQ%=0:GOTO 200
270       REM IF ERROR< >SYNTAX ERROR, LIST ERROR MESSAGE IN
                    OCTAL AND END.
280 IF SRQ%=1 THEN PRINT "ERROR.  STATUS REGISTER (IN
    OCTAL)= ";OCT$(POLL.RESP%):END
290       REM ENTER DATA.STRING$ AND WRITE PROGRAMMING
                    INFORMATION TO DMM #19. ADD <CR> FOR EOS.
300 INPUT "ENTER COMMAND STRING FOR TEMP. DMM
    (#19)";DATA.STRING$
310 DATA.STRING$=DATA.STRING$+CHR$(13)
320       REM OUTPUT DATA.STRING$ TO DMM
330 PARAM$="WR.STR/19//13/EOS/":GOSUB 10000
340       REM CHECK FOR SYNTAX ERROR IN DATA.STRING$.  IF YES, LOOP
                    BACK TO REENTER DATA.STRING$
350 PARAM$="RBST/":GOSUB 10000
360       REM IF ERROR< >SYNTAX ERROR, LIST ERROR MESSAGE IN
                    OCTAL AND END.
370 IF SRQ%=1 THEN PARAM$="SER.POLL/19/":GOSUB 10000:IF
    POLL.RESP% AND 4=4 THEN PRINT "SYNTAX ERROR IN
    COMMAND":SRQ%=0:GOTO 300
380 IF SRQ%=1 THEN PRINT "ERROR. STATUS REGISTER=(IN
    OCTAL) ";OCT$(POLL.RESP%):END
390       REM BEGINNING OF DATA-ACQUISION LOOP.  INITIATE A GROUP-
                    EXECUTE TRIGGER TO RECORD THERMOCOUPLE AND
                    PRESSURE TRANSDUCER READINGS SIMULTANEOUSLY.
400 PARAM$="GET/17,19/":GOSUB 10000
410       REM READ THE DMM VALUES INTO THE COMPUTER
420 PARAM$="RD.STR/17//10/EOS/":GOSUB 10000
430       REM STORE THE READING IN THE PRESSURE ARRAY.  1
                    TORR=10mV, SO *100 MAKES V=PRES IN TORR.
440 PRES(DPT)=VAL(DATA.STRING$)*100
450       REM NOTE THAT THE LINE FEED IS USED TO SIGNAL THE END OF
                    DATA INSTEAD OF THE LENGTH OF COUNT. LENGTH OF
                    COUNT CAUSES AN ERROR CONDITION HERE WITH V.3
460 PARAM$="RD.STR/19//10/EOS/":GOSUB 10000
470       REM STORE THE READING IN THE TEMPERATURE ARRAY.  IF
                    TEMP>77K GO TO CALCULATION ROUTINE
480 TEMP(DPT)=VAL(DATA.STRING$)*1000:IF TEMP(DPT)>-5.539 THEN 630
490       REM CHECK FOR FRONT PANEL SRQ.  IF YES, GO TO
                    CALCULATION ROUTINE
500 PARAM$="RBST/":GOSUB 10000
510 IF SRQ%< >1 THEN 550
520 PARAM$="SER.POLL/17/":GOSUB 10000:IF POLL.RESP%< >0 THEN 630
530 PARAM$="SER.POLL/19/":GOSUB 10000:IF POLL.RESP%< >0 THEN 630
540       REM READ NEW TIME, CHECK ELAPSED TIME
550       REM TIME$ IS RESET TO 0 WHEN RBST CHECKS FOR TIMEOUT
                    FAULTS
560 ENDCLK=VAL(RIGHT$(TIME$,2)):PRINT ENDCLK
570       REM IF AT LEAST 30 SEC HAVE ELAPSED, GET NEW READING
580 IF ENDCLK<30 THEN 560
590       REM CHECK FOR END OF ARRAY.  IF YES, JUMP TO CALCULATION
                    ROUTINE. OTHERWISE INCREMENT DPT AND COLLECT NEXT
                    POINT
600 IF DPT>249 THEN 630
610 DPT=DPT+1:GOTO 400
620       REM PRINT DATA AND DO SEMILOG REGRESSION GOES HERE.
                    ROUTINE DELETED FOR BYTE ARTICLE. FULL ROUTINE
                    AVAILABLE FROM AUTHOR.
```

(continued)

Inquiry 290 →

The cost of adding an IEEE-488 *interface is nominal and the added flexibility is not available from any other source.*

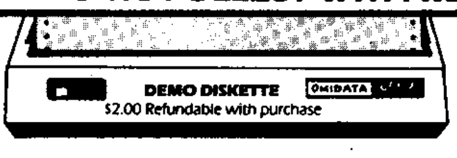
turer and is not a part of the IEEE-488 standard. This data is used to program the 3478A instead of secondary addresses, which the HP does not support. The significance of the string to the DMM is as follows: K = "clear the maskable interrupt register," M24 = "set a new mask to generate an SRQ if programming data sent to the DMM has a syntax error or if the front-panel SRQ button is pushed," and D2PRESSURE = "display the word 'PRESSURE' on the DMM's display panel." Since the pressure-monitoring DMM and the temperature-monitoring DMM look exactly the same, this prompt ensures that the instrument is connected to the right transducer. Line 160 has the programming string output to the correct DMM. Field 2 of WR.STR specifies the device number (17) of the appropriate DMM and field 3 says to transmit 14 characters. That is, the end of sequence (EOS) is identified by simply counting the number of characters transmitted. At the end of 14 characters, the computer will UNLISTEN the DMM to terminate transmission. Line 200 has the operator input the programming information that will specify the functions that the pressure-monitoring DMM will use. Since the number of characters in the command string will vary with what options the operator selects, we don't use a character count to signal EOS here. Rather, in line 210 we tack a <CR> code onto the end of the data and in line 230 specify that the transmission to the DMM should continue until "13" (the carriage-return code in ASCII) is encountered. Since each operator enters the programming information on each experimental run, we want to verify that the DMM string does not contain any typographical mistakes. Therefore, we read the IEEE-bus status (line 250) and see whether an SRQ flag has been set (line 260). Remember that the DMM was programmed in line 160 to generate an SRQ on a syntax error. If an SRQ has been sent, we examine the status register of the DMM (260 also) to make sure that the SRQ was caused by a syntax error and, if so, have the operator reenter valid programming information for the DMM. Note that the SRQ does not automatically interrupt the central processing unit. It only sets a flag on the IEEE bus. If we want to ignore it, all devices that are still able to function properly can carry on with their business as usual. If we want an SRQ to automaticaly interrupt the computer, we can tie the SRQ line to an IRQ line.

Now let's skip to line 400. This initiates a group-enable trigger for both DMMs (numbers 17 and 19). Thus, our pressure and temperature data readings are triggered at the same time and are truly simultaneous. In line 420, we read the pressure DMM data into the pressure array PRES. Character 10, a linefeed, is used as an EOS by the DMM and is so declared in line 420. Lines 500 through 530 check to see if an SRQ was sent by any device and, if so, conducts a serial poll. This is done because the program allows the experimenter to interrupt the experiment at any time by pressing a front-panel button on either DMM. The program will then treat the data collected up to that time as the complete data set and begin the data analysis routine. If there was no SRQ, the program waits 30 seconds, checks to make sure that the data arrays are not about to overflow, and then takes another data reading.

The program presented above is a very simple routine. However, even this basic level of process control is very difficult to achieve on interfaces other than the IEEE-488. If you have a choice, you should begin reshaping your lab to support IEEE-488 interfacing. As you replace outdated or broken equipment, the cost of adding an IEEE-488 interface is nominal and the added flexibility is not available from any other source. ■

ATTENTION OKIDATA OWNERS!

USE YOUR PRINTER TO ITS FULLEST WITH MARVEL PRINT™!



MARVEL PRINT FEATURES:
- **Proportional Spacing** of letter quality text–(justifies right margins).
- Enables you to create **graphics**–even in the middle of text.
- Lets you create your own **character sets.**
- Allows you to **backspace.**
- Includes a **powerful Label Printing Program.**
- Uses only **one character** for common codes:

SUBSCRIPT	EMPHASIZED
SUPERSCRIPT	DOUBLE WIDTH
UNDERLINE	PICA
ENHANCED	ELITE
DATA MODE	CONDENSED

MARVEL PRINT–the new user–friendly program that generates *ALL* the features of the Okidata 92 & 93 printers using *ANY* text-producing program (word processor, spread sheet, data base).

We sell Okidata 92 & 93 printers bundled with Marvel Print hardware at discount prices. Call or write for more information. Dealers welcome. Okidata is a trademark of the Okidata Corp.

Also available from Marvel Software by Popular Demand:
Character Sets: Italics • Script • Science & Math Symbols • Hebrew • Russian • Arabic • Greek • Foreign Language Marks
Character Clone Set: Allows you to take characters from different sets & combine them for simultaneous use.

Marvel Print only	$70
Marvel Print with any 1 character set	$85
Marvel Print with any 2 character sets	$90
Character Clone set	$10

Works with Apple, CP/M-80, PC-DOS, MS-DOS. **State System.** Visa, American Express, Mastercard welcome. Phone orders accepted or Send check or M.O. to:

MARVEL SOFTWARE
1922 Ave. N, B'klyn, N.Y. 11230, (718) 336-2323

Inquiry 198