Behavior Research Methods, Instruments, & Computers
1989, 21 (4), 435-440

An inexpensive interface for the IBM PC/XT and compatibles

DAVID G. LAVOND
University of Southern California, Los Angeles, California

and

JOSEPH E. STEINMETZ
Indiana University, Bloomington, Indiana

An inexpensive interface designed for IBM PC/XTs or compatibles, potentially useful for controlling behavioral/neural experiments, is presented. Descriptions of basic machine language routines for the control of input/output functions, timing, and data acquisition by the interface are provided. Also, a brief description of available Forth routines for controlling the delivery of stimuli and the acquisition of behavioral/neural data during classical conditioning experiments is presented.

In this article, we describe an interface for the IBM PC/XT that has been used for several years to control and collect data in classical conditioning experiments (e.g., Lavond, Steinmetz, Yokaitis, & Thompson, 1987). The system can be used on-line, providing immediate display of responses, or off-line. Although IBM no longer makes the PC/XT computer, there is still a large market for their clones at reasonable prices. Given the availability and power of these models, the interface and software described in this article may serve as an attractive alternative to the KIM-1 system developed by Solomon and colleagues (Solomon & Babcock, 1979; Solomon, Weisz, Clark, Hall, & Babcock, 1983) and to the system developed by Gormezano and colleagues for classical conditioning with the Apple II series of computers (Scandrett & Gormezano, 1980). The retail cost of parts for this interface is less than $100 (including chips, board, wire, sockets, and cables), and it takes about 8-10 h to wire wrap by hand. In designing and programming this interface, we found books by Eggebrecht (1983), National Semiconductor (1981), Rector and Alexy (1980), Royer (1987), and Willen and Krantz (1984) to be very useful.

INTERFACE HARDWARE

Figure 1 is a schematic of the component side of the interface. The interface to the IBM or compatible computers requires three sets of functional connections. First, the 74LS245 (an octal bus transceiver) provides buffering for the data lines between the host computer and interface. Second, the interface resides in an address space decoded by the 74LS688 (an 8-bit magnitude comparator; a 25LS252 may be substituted), and the 74LS138 (a three-to-eight line decoder/multiplexer). The base address is currently decoded between $0390 and $039F (the "$" indicates hexadecimal numbering system) in the computer's input/output (I/O) space. Third, control lines from the computer are generally connected directly (although these could have been buffered with another chip), with the exception that the 74LS02 (a quad two-input NOR gate), in conjunction with the address decoders, is used to read from and write to the analog-to-digital (A/D) converter (ADC0808). These decoding and buffering functions are relatively straightforward and have been incorporated into some commercial prototype boards (e.g., by Jameco Electronics, JDR Microdevices). See Table 1 and the discussion below for important addresses used by the interface.

With a relatively small number of parts, the circuit diagram for the interface depicted in Figure 1 provides 24 programmable I/O bits (Intel 8255), four multiplexed A/D converters (National Semiconductor ADC0808), and three timers/counters (Intel 8253-5). The interface for classical conditioning experiments has a subset of these features including 6 TTL-level outputs (i.e., 0 V $\leq$ logic 0 $\leq$ +0.8 V, and +2.4 V $\leq$ logic 1 $\leq$ +5.0 V) used for controlling a tone conditioned stimulus (CS), a light CS, an air puff unconditioned stimulus (US), a shock US, a tape recorder, and synchronization pulses for CS and US events. (Additional equipment must be added to effect these events. Our system is coupled to Coulbourne or BSR equipment, with optoisolator circuits.) The interface also provides four A/D channels of input, two counters for collecting neural unit activity, one timer for intratrial events, and one external trigger for analyzing data previously recorded on tape.

The interfaces are constructed with sockets for the chips and wire wrapping to connect pins. Since some signals are distributed to a number of different places, wiring

Requests for reprints should be sent to Joseph E. Steinmetz, Department of Psychology, Program in Neural Science, Indiana University, Bloomington, IN 47405.

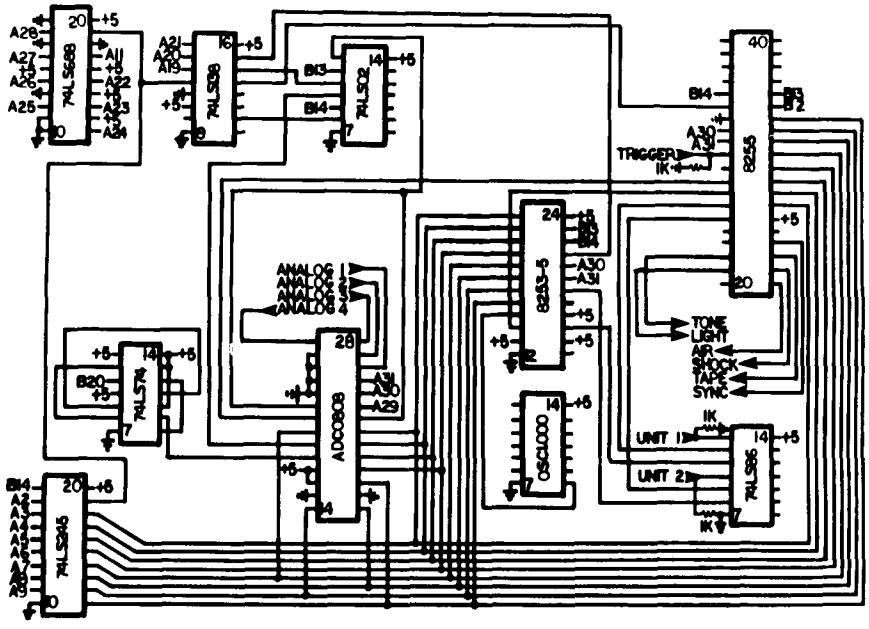

Figure 1. Component side of an interface for IBM PC/XT and compatible computers. Pin numbers for the lower left and upper right of each chip are indicated (these are the traditional placements for ground and +5 V for TTL chips). Connections numbered with letters (e.g., A28 and B14) refer to numbered connections from the IBM expansion slot. Arrowheads indicate outputs to external equipment (e.g., to a tone generator), and arrow tails indicate inputs from external equipment (e.g., from a potentiometer measuring the position of the nictitating membrane). All voltage levels must be within the range of 0 to +5 V.

should be planned so that the smallest amount of wire is used, in order to reduce inductions of noise. Similarly, care should be taken in routing the clock signals for the 8253-5 and ADC0808 chips, so that they are not near signal lines from outside equipment. Finally, unused A/D inputs should be grounded and 1-kΩ pull-down resistors used on input bits (i.e., on the external trigger and the unit counters).

Table 1
Summary of Addresses Used by Interface Board

Device	Address	Function
IBM PC/XT Computer or Compatibles		
Computer I/O	$0390-$039F	16-bit I/O lines
Programmable I/O bits (Intel 8255)		
Control Register	$0B93	Configure I/O Ports
Port A	$0B90	Input (8 bits)
Port B	$0B91	Output (8 bits)
Port C	$0B92	Input (Upper 4 bits)
		Output (Lower 4 bits)
Timer/Counters (Intel 8253-5)		
Control Register	$0393	Configure Timer/Counter
Timer/Counter 0	$0390	Timer
Timer/Counter 1	$0391	Counter
Timer/Counter 2	$0392	Counter
A/D Converter (ADC0808)—Four Decoded Channels		
Channel 0	$0790	Start Conversion Address
	$1790	Read Data Address
Channel 1	$0791	Start Conversion Address
	$1791	Read Data Address
Channel 2	$0792	Start Conversion Address
	$1792	Read Data Address
Channel 3	$0793	Start Conversion Address
	$1793	Read Data Address

The chips on this prototyping board are powered through the computer. To help create steady voltages on the board, a 100 μF polarized capacitor should be added to the board near the connector at the points where the power supply enters the board. To prevent electrical noise, each TTL chip that has high-frequency switching should have a 0.01 or 0.001 μF disk or mica capacitor soldered between its power supplies (the shorter the distance for the connections, the better). These despiking capacitors should be added to the 74LS74, 74LS138, 74LS245, 74LS688, and OCS1.000.

The 74LS138 has four unused decoded addresses (pins 7, 9, 11, and 12), which could be used to add more chips to the board (in fact, a full-length prototype board has much more room for additional chips). Another 8255 I/O chip and another 8253-5 timer/counter chip have been added in another application with no problem. It is also possible to add a D/A converter to the interface.

INTERFACE SOFTWARE

Control of the interface hardware is accomplished through two levels of software: (1) a collection of machine language subroutines that control I/O, A/D conversion of behavioral responses, counting of discrete events (such as discriminated neural spikes), and the timing of the variety of events associated with presentation of stimuli; and

(2) a RUNTIME program written in Forth that integrates the fast machine language subroutines with slower routines used in data reduction, graphic displays, parameter changes, and other features of running classical conditioning experiments. The Forth programs will be described briefly at the end of this paper; a number of simple, generic, machine language routines for controlling portions of the interface are described below.

Input/Output with the 8255

Intel (1986, 1987) publishes hardware and software documentation for the 8255. The general design of the 8255 is to provide three 8-bit I/O ports, labeled Port A, Port B, and Port C. Two of the ports (A and B) can be programmed to be all input or all output, while one of the ports (C) can be split into four inputs and four outputs. In our design, the I/O lines generally fall into two categories: lines that make external connections to equipment, and lines that make internal connections to control the functions of other interface chips. Of the external connections, we currently use six lines for output and one line for input (although more lines could be used). The six outputs are connected to external equipment to provide for tone and light CSs, air puff and shock USs, control over a tape recorder, and synchronization pulses to mark trial events. The one input is used as an external trigger for collecting data off-line. The internal lines include two outputs and two inputs. The outputs are used, through the 74LS86 (a quad two-input exclusive OR gate), to set the neural unit counters. One input is used to read the end-of-conversion (EOC) signal from the A/D converter. The EOC indicates that a valid conversion has been made. The second input reads the intratrial timer and is used to signal the completion of a timing period.

To use the 8255, the addresses of its internal registers must be known. The control register at $0B93 is used to determine how the various ports are configured (i.e., as inputs or as outputs). To set up the 8255 for classical conditioning, the commands are:

```
MOV   AX,$98
MOV   DX,$0B93
OUT   DX,AL
```

These commands set Port A's 8 bits as input, Port B's 8 bits as output, Port C's lower 4 bits as output, and Port C's upper 4 bits as input. (Although Port A is not used in our application, we purposely configure it to function as an input port, thus causing it to act as a high-impedance source. This prevents the activation of equipment that might accidentally be attached at this point.) To use these connections, Port A is at $0B90, Port B is at $0B91, and Port C is at $0B92. To turn on bit 0 of Port B, for instance, use:

```
MOV   AX,$01
MOV   DX,$0B91
OUT   DX,AL
```

To turn on bit 7 while leaving bit 0 on the following command is issued:

```
MOV   AX,$81
MOV   DX,$0B91
OUT   DX,AL
```

—instead of issuing the command

```
MOV   AX,$80
MOV   DX,$0B91
OUT   DX,AL
```

which would turn on bit 7 and turn off all other bits. Since the state of the output bits might be unknown, the following routine can be used to read the current output and set a specified bit without affecting the other bits:

```
MOV   BX,bit-number-to-set
MOV   DX,$0B91
IN    AL,DX        ; read the current port
                     setting
OR    AL,BL
SUB   AH,AH
OUT   DX,AL
```

On the other hand, the bits that are currently set may not be known, yet a specific bit may need to be reset (i.e., set to zero). The following routine resets a specified bit without affecting the other bits:

```
MOV   BX,bit-number-to-reset
MOV   DX,$0B91
IN    AL,DX        ; read the current port
                     setting
PUSH  AX
AND   AX,BX
NOT   AL
POP   BX
AND   AL,BL
SUB   AH,AH
OUT   DX,AL
```

Note from examining the last two examples that reading a port is relatively simple. For example, the following command reads Port C:

```
MOV   DX,$0B92
IN    AL,DX
```

Timing and Counting with the 8253

Intel (1986, 1987) publishes hardware and software documentation for the 8253-5. The -5 version of the chip was chosen because it was designed for use with the IBM PC or compatibles. The 8253-5 has three programmable timers/counters. The control register is at address $0393. The first timer/counter (at $0390) is configured as a timer. Its counter frequency of 1 MHz is determined by its input clock (i.e., the OSC1.000 crystal that is included as a part of the interface). This frequency was chosen because of its convenience in timing for interstimulus events

that underlie our classical conditioning experiments. The other two timer/counters are configured as counters for collecting discriminated neural unit activity and are located at $0391 and $0392. The following commands can be issued to initialize the 8253-5 for these timer/counter functions:

```
        MOV   DX,$0393    ; set timer/counter 0 to
                            mode 3
        MOV   AX,$36
        OUT   DX,AL
        MOV   AX,$03      ; divide base frequency
                            by decimal 1000
        OUT   DX,AL       ; (i.e., $03E8) to give a
                            1 kHz output
        MOV   AX,$E8      ; square wave
        OUT   DX,AL
;
        MOV   AX,$70      ; set timer/counter 1 to
                            mode 0
        OUT   DX,AL
        MOV   AX,$00
        OUT   DX,AL
        OUT   DX,AL
        MOV   AX,$01      ; set count for counter 1
        MOV   DX,$0B92
        OUT   DX,AL
;
        MOV   AX,$BO      ; set timer/counter 2 to
                            mode 0
        OUT   DX,AL
        MOV   AX,$00
        OUT   DX,AL
        OUT   DX,AL
        MOV   AX,$02      ; set count for counter 2
        MOV   DX,$0B92
        OUT   DX,AL
```

With this initialization, Timer 0 ($0390) produces a square wave output every 1 msec. The output of Timer 0 is wired to bit $10 of Port C of the 8255. By polling this bit, the program can detect transitions of the square wave pulse. The following code acts as a software debouncer to ensure that an exit is made only on detection of a positive transition:

```
        MOV   DX,$0B92
LP1     IN    AL,DX
        AND   AL,$10
        JE    LP1         ; loop until input equals
                            logic 0
LP2     IN    AL,DX
        AND   AL,$10
        JNE   LP2         ; loop until input equals
                            logic 1
```

The second and third timer/counters on the 8253-5 are used to count neural unit activity. The following routine sets the first unit counter to zero, waits for a timing cycle, latches the count, and then reads the count:

```
        MOV   DX,$0391
        MOV   AL,$00
        OUT   DX,AL
        OUT   DX,AL       ; set count to 0 (zero)
;
        MOV   DX,$0B92
        IN    AL,DX
        OR    AL,$01
        OUT   DX,AL
        AND   AL,$FE
        OUT   DX,AL       ; set count by pulsing
                            input once
;
;       MOV   DX,$0B92    ; DX is already set
                            above
LP1     IN    AL,DX
        AND   AL,$10
        JE    LP1         ; loop until input equals
                            logic 0
LP2     IN    AL,DX
        AND   AL,$10
        JNE   LP2         ; loop until input equals
                            logic 1
;
        MOV   DX,$0393
        MOV   AX,$40
        OUT   DX,AL       ; latch count
;
        MOV   DX,$0391
        IN    AL,DX
        MOV   AH,AL
        IN    AL,DX       ; read count
        XCHG  AH,AL
;
        NEG   AX          ; convert count to
                            positive number
```

The counters of the 8253-5 count down (rather than up), giving a negative number. To obtain the correct positive number, the routine listed above simply negates a negative. Also, the starting count of zero is not set into the counter until the counter receives one input. For this reason, the interface is wired so that an output of the 8255 indirectly gives an external pulse to the 8253-5. The 8255 signal goes through one input of the 74LS86 (quad two-input exclusive OR gates) and the output goes to the 8253-5. The other input to the exclusive OR gate comes from the unit discriminator output (e.g., a Haer slope/height window discriminator with a 200-μsec output). Either of the inputs will set the initial zero count into the counter. Since the pulse from the 8255 is much shorter in duration than the pulse from the discriminator, and an exclusive OR design is used, this design allows for the accurate count of all incoming signals except at the rare

times when two signals turn on or off at exactly the same times.

Analog-to-Digital Conversion with the ADC0808

National Semiconductor (1981) publishes hardware and software documentation for the ADC0808 A/D converter. This chip has eight single-ended, multiplexed A/D channels. Due to the design of our interface, four of the A/D inputs are not used and are therefore grounded to avoid parasitic noise on the active channels. (The interface is limited to four A/D channels by the Forth software that was written to collect and analyze data on-line. To use all eight channels would require a different address decoding scheme, as well as alterations in existing Forth programming.) The remaining four "active" channels can be used to collect analog data. (Unused "active" channels—i.e., channels that are wired to other components on the board but are not currently being used—are also grounded to prevent the induction of parasitic noise.) Each of these four "active" channels in the A/D converter has two addresses, one for starting the conversion (a write operation) and the other a read operation. Channel 1 has a start address of $0790 and a read address of $1790. Channel 2 has a start address of $0791 and a read address of $1791. Subsequent channels continue consecutively. The following code uses Channel 1:

```
      MOV   AX,$00
      MOV   DX,$0790
      OUT   DX,AL       ; begin conversion for
                          Channel 1
;
      MOV   DX,$0B92
LP1   IN    AL,DX
      AND   AL,$20
      JE    LP1
LP2   IN    AL,DX
      AND   AL,$20
      JNE   LP2         ; wait for EOC signal
                          from A/D converter
;
      MOV   DX,$1790
      IN    AL,DX       ; read A/D value
```

Although the ADC0808 has more than one input (this design decodes for four of the eight channels), these channels are multiplexed to share a single A/D converter. Starting a second channel before the A/D converter is finished with the first channel abandons the first conversion.

The conversion rate for the ADC0808 is a function of the input clock: the faster the clock, the quicker the conversion. The ADC0808 accepts clock speeds ranging from 10-1280 kHz. In the design of this interface, the clock is derived by dividing the IBM computer's master clock (4.77 MHz) by 4 with the 74LS74 (a dual D positive-edge-triggered flip-flops with preset and clear), which results in a clock speed of 1.15 MHz, or close to the maximum speed. Computers with higher clock speeds can cause problems for the A/D converter (like those higher clock speeds typically found in IBM ATs and compatibles). To eliminate this problem, the master clock resident in the computer and the 74LS74 can be substituted with (or shared between) a crystal similar to the OSC1.000 used by the 8253-5. This substitution would make the interface's timing independent of the computer. According to National Semiconductor's documentation (1981), the typical conversion frequency (with a 640-kHz clock) results in a sample rate of 10 kHz. Using the Nyquist rule that one needs to sample at twice the frequency of the signal of interest, this interface can practically sample signals up to 5 kHz. This rate is more than adequate for sampling muscle movements, EEG signals, and other applications. For example, the fastest rate that we typically use to sample nictitating membrane movement during classical conditioning is 500 Hz (i.e., once every 2 msec). This rate has proven sufficient for describing movement of the eyelid. Although timing problems have been encountered when computers with relatively faster clock speeds have been used (e.g., IBM ATs), no address decoding problems have been observed as long as the interface board resides in a 16-bit expansion slot in the AT or compatibles.

CLASSICAL CONDITIONING WITH THE INTERFACE

One implementation of classical conditioning procedures with the present interface requires two separate programs. One program controls the experiment, collects data, and initially analyzes data on-line for immediate trial-to-trial feedback of results. This is a compiled program called RUNTIME. The source code for RUNTIME can be altered or rewritten and compiled as desired. The second program, SUMMARY, is really a collection of several programs that allow further data analysis, or reanalysis of the same data with different analysis parameters. SUMMARY is compiled as it is being used. In this fashion, the SUMMARY program allows the user to conveniently alter the existing analysis programs or to write other programs as desired. The RUNTIME program contains code for all timing routines used to present the classical conditioning stimuli (e.g., tones, lights, air puff, shocks, etc.) as well as other intra- and intertrial timing events (e.g., randomizing intertrial intervals, turning on tape decks or other recording devices before trial onset, etc.). A variety of different experimental procedures can be stored simultaneously within the RUNTIME program, so that a number of different experimental protocols can be recalled for use (e.g., trace conditioning, delay conditioning, discrimination training, etc.). Data acquisition routines are also part of the RUNTIME program, including A/D conversion and storage of analog (i.e., behavioral) signals and also the counting and storage of discriminated neural spikes. Some on-line data reduction and analysis is performed by the RUNTIME program, and graphic displays of the behavioral and neural data are generated for each trial by the program. The SUMMARY program con-

tains a number of routines that are helpful for further reducing and storing data collected with the RUNTIME program. Routines for data reductions such as obtaining session and block summaries for a number of behavioral dependent measures used commonly during classical conditioning are included within the SUMMARY program (e.g., response amplitudes, onset latencies, peak latencies, etc.). The summary of neural data is also performed with these SUMMARY programs, and routines for storing reduced, blocked data on library disks, routines for summarizing the data graphically, and routines for altering parameters of analysis are also included.

Much of the RUNTIME program that controls the experiment (i.e., the presentation of stimuli) and collects data (both behavioral and neural) has been written in 8086/8088 machine language for the ease of controlling the hardware as well as providing the necessary speed of execution (see above for examples of the subroutines). However, we use the intermediate language, Forth, to pull the bits of machine code together. Like the First language used by Scandrett and Gormezano (1980), Forth is a threaded interpretive language. Forth was chosen because it is easy to integrate machine code directly in the Forth stream, and because Forth is easy to debug, readily available for a wide range of computers, and documented in a number of readily available books. In fact, the programs were originally written and the hardware interface designed for the Apple II+ computer, and the part written in Forth was easily ported over to the IBM PC (i.e., the only major changes involved in creating the IBM version occurred in modifying the interface slightly and rewriting the machine language code). The enhanced Forth-83 version available from Micromotion was used. Micromotion (12077 Wilshire Blvd., No. 506, Los Angeles, CA 90025) also sells a floating-point software package that accompanies the Forth language, which is useful when floating-point arithmetic is required during data acquisition and reduction. Forth is also available from a number of other vendors, including Laboratory Microsystems, Inc. (4147 Beethoven St., Los Angeles, CA 90066), Mountain View Press, Inc. (P.O. Box 4656, Mountain View, CA 94040), and Miller Microcomputer Services (61 Lakeshore Road, Natick, MA 01760). Information on these, other sources, and public-domain versions of Forth is available from the Forth Interest Group (P.O. Box 1105, San Carlos, CA 94070). There are a number of good books available on Forth (e.g., Brodie, 1984, 1987, and Anderson & Tracy, 1984, which Micromotion includes with purchase of their software package).

Assuming that the researcher already owns MS-DOS or PC-DOS and has purchased Micromotion Forth for the IBM, the complete RUNTIME and SUMMARY programs can be purchased from the authors for $900. We believe that this software and hardware together provide a satisfactory solution for conducting classical conditioning experiments and could be modified for other uses. Furthermore, the hardware design presented in this article seems flexible enough to provide interfacing solutions for researchers with access to programming skills.

REFERENCES

ANDERSON, A., & TRACY, M. (1984). *Mastering Forth*. Bowie, MD: Brady Communications.

BRODIE, L. (1984). *Thinking Forth: A language and philosophy for solving problems*. Englewood Cliffs, NJ: Prentice-Hall.

BRODIE, L. (1987). *Starting Forth* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

EGGEBRECHT, L. C. (1983). *Interfacing to the IBM personal computer*. Indianapolis, IN: Howard W. Sams.

INTEL. (1986). *Intel yellow pages software directory*. Santa Clara, CA: Author.

INTEL. (1987). *Intel microsystem component book*. Santa Clara, CA: Author.

LAVOND, D. G., STEINMETZ, J. E., YOKAITIS, M. H., & THOMPSON, R. F. (1987). Reacquisition of classical conditioning after removal of cerebellar cortex. *Experimental Brain Research*, **67**, 569-593.

NATIONAL SEMICONDUCTOR. (1981). *Logic databook*. Santa Clara, CA: Author.

RECTOR, R., & ALEXY, G. (1980). *The 8086 book: Includes the 8088*. Berkeley, CA: Osbourne/McGraw-Hill.

ROYER, J. P. (1987). *Handbook of software and hardware interfacing for IBM-PCs*. Englewood Cliffs, NJ: Prentice-Hall.

SCANDRETT, J., & GORMEZANO, I. (1980). Microprocessor control and A/D data acquisition in classical conditioning. *Behavior Research Methods & Instrumentation*, **12**, 120-125.

SOLOMON, P. R., & BABCOCK, B. A. (1979). KIM and the rabbit: The use of the KIM-1 microprocessor to control classical conditioning of the rabbit's nictitating membrane response. *Behavior Research Methods & Instrumentation*, **11**, 67-70.

SOLOMON, P. R., WEISZ, D. J., CLARK, G. A., HALL, J., & BABCOCK, B. A. (1983). A microprocessor control system and solid state interface for controlling electrophysiological studies of conditioning. *Behavior Research Methods & Instrumentation*, **15**, 57-65.

WILLEN, D. C., & KRANTZ, J. I. (1984). *8088 assembler language programming: The IBM-PC* (2nd ed.). Indianapolis, IN: Howard W. Sams.

(Manuscript received February 10, 1989;
revision accepted for publication June 23, 1989.)