Behavior Research Methods, Instruments, & Computers
1987, 19 (3), 327-334

Interval timing routines for the IBM PC/XT/AT microcomputer family

MICHAEL BÜHRER, BERND SPARRER, and ROLF WEITKUNAT
University of Munich, Munich, Federal Republic of Germany

We describe a 1-msec software timer for measuring response latencies and controlling delays on the IBM PC/XT/AT without additional hardware requirements. To demonstrate the machine language routines, a short BASIC example program is included. In a simple experimental design, two different stimulus words are presented on screen and keypress response latencies are measured. Precise timing of stimulus presentation is accomplished by direct manipulation of the video controller. The principles of programming interrupt-controlled timing routines are addressed to be easily adapted to other problems or different programming languages.

Designs in experimental behavior research often require exact time-interval measurement and control. Subjects' response latencies or delays of external events often have to be assessed; interstimulus intervals, intertrial intervals, and duration of stimulus presentation need to be timed in order to control the experimental process. Other applications combine two or more of these components (e.g., response latency windows).

Dlhopolsky (1983) proposed two Z-80 machine language millisecond timers that are software designed but based on hardware clock cycles of the TRS-80. Femano and Pfaff (1983) developed an interval handler for the 6502 microprocessor using external circuitry. Although such solutions achieve very high resolutions (less than 50 μsec), they depend on specific hardware requirements.

Currently the standard in microcomputers, in terms of laboratory and analysis applications, tends to be established by the IBM personal computer family. A wide variety of hardware add-ons is available from numerous suppliers. However, adequate add-ons for the laboratory standard of minicomputers (DEC's PDP-11) are unavailable. Particularly, external clocks usually are unable to interrupt the processor, which causes various programming difficulties.

On the other hand, the prerequisites for programming interval timing routines are already built-in features of the IBM PC/XT/AT computer family. In order to demonstrate this, the first part of the present paper describes a simple BASIC program, which makes use of an assembly routine that performs all the difficult steps of timing control. The second part explains the technical details and programming techniques needed to tailor the assembly language routines to the reader's own requirements.

The authors' mailing address is: Institut für Psychologie, Klinische Psychologie, Universität München, Geschwister-Scholl-Platz 1, D-8000 München 22, Federal Republic of Germany.

REACTION-TIME MEASUREMENT

The BASIC program shown in Listing 1 demonstrates an experimental session in which reaction time to two different verbal stimuli is measured. Such a procedure has to fulfill two critical requirements. Delay and reaction times have to be measured precisely, and the stimulus must be presented at a predetermined point in time. These two requirements are met by an assembly language programmed timer, which is able to measure time intervals with a resolution of 1 msec, and a pseudo-tachistoscopic stimulus presentation. The latter is achieved by switching the video display controller off and moving the contents of a prewritten RAM screen into the display memory area. This program design guarantees exact process control even with complex stimulus patterns.

Before presentation of the stimulus, a warning stimulus is given, defined as the moment when the text on the screen disappears. After a predefined delay, the imperative stimulus is presented. It consists of one of two words appearing centered on the video display. The time interval between the warning stimulus and the imperative stimulus is controlled by the delay timer. When the display is turned on and the stimulus appears, the latency timer starts to measure the time until the subject presses a key. The next trial is delayed by a random interstimulus interval. Each word is presented five times in a pseudo-random sequence. At the end of the session, means and standard deviations of response latencies are computed and displayed.

There is a limitation to the exactness of stimulus presentation using a video display. The minimum time required to make a stimulus pattern visible is determined by the refresh rate of the monitor (16.7 msec for a 60-Hz monitor). This could be compensated by synchronizing stimulus presentation with vertical retrace of the electron beam. Since not all display adapters provide the necessary system information, this feature is not implemented in the present program, although the assembly routine offers a

solution to this problem for Color and Hercules Graphics adapters.

The BASIC program is designed for IBM's BASICA interpreter (Version 3.0 or later) or Microsoft's GWBASIC (Version 2.01 or later). The assembly routines should be assembled with Microsoft's MASM (Version 4.0). The program runs on all IBM PC/XT/AT models and on most compatibles. For compatibles, it is a good idea to check the input frequency of the timer/counter chip for substantial deviations from 1.1931817 MHz. (Usually, the computer's technical reference manual provides this information.)

Lines 100–200 initialize all variables used. This is mandatory since interpreter BASIC reorganizes memory when new variables are created. This reorganization must be avoided to ensure that the location of machine language subroutines in memory remains unchanged. Line 160 sets the number of different stimulus words (NWORDS) and the number of presentations of each word (NTRIALS). These values may be changed for other designs (the DATA statements of lines 5030 and 5060 then have to be adjusted). The variable MS in line 140 defines the delay time in milliseconds between the warning and imperative stimulus. It should be long enough to allow for the screen to become completely black. MAXRES defines the maximum response time allowed in milliseconds. This prevents the computer from locking up if the subject fails to respond. The dimension statements in lines 290–310 define the array where the assembly routine will be stored (IASM), a RAM screen (ISCRN) that will be filled with the stimulus text, and the arrays for interstimulus intervals (ISI) and measured latencies (LATB).

Line 320 defines ISUBRT as the starting address of array IASM. ISUBRT will be used to load and call the assembly routine. Loading may be done in two ways: Either subroutine 6000 is called, which reads the DATA statements beginning at line 9000, pokes the data into IASM, and checks for typing errors, or BLOAD is used to read a binary file (produced by MASM; see Listing 2) from disk. The latter, of course, relieves the user from typing lines 6000 to 9480.

The interstimulus intervals (ISIs) are random generated in subroutine 2010. First, the minimum and maximum interstimulus intervals are entered in seconds. The minimum must be greater than 3 sec, because BASIC fills the RAM screen during the interval (see line 440). Compiled BASIC decreases this limit.

Writing to the RAM screen is accomplished in lines 1020–1120. First, the whole 80×25 integer array is filled with blanks, where low bytes contain the ASCII code 20 (hex) and high bytes set the normal screen attribute 07. Lines 1050–1110 center the stimulus word on the screen.

Line 410 starts the main program loop. The ISI is timed by BASIC's simple TIMER function (lines 420 and 450). Next, subroutine 3010 is called, which calls ISUBRT passing the arguments MS (delay between warning and imperative stimulus), ISCRN (starting address of RAM screen), LAT (on input, LAT contains the value of maximum response time allowed [MAXRES]; on output, it returns measured response latency), and KEYP (returns ASCII code of key pressed in low byte and keyboard scancode in high byte). In case the subject fails to respond, both output parameters (LAT, KEYP) are set to zero. When ISUBRT returns to BASIC, the main loop continues. Following the last response latency measurement, statistics are calculated and results are presented (lines 4010–4260).

HARDWARE

The Intel 8253 16-bit timer/counter circuit (located on the motherboard) provides three programmable interval timer channels, of which channel 0 is used to maintain the time of day for the system. The other two channels are reserved for system internal purposes (see Sargent & Shoemaker, 1984). The input clock frequency for the 8253 is derived from the system clock (4.772727/4 = 1.1931817 MHz). The output frequency is determined by a programmable 16-bit counter, which for channel 0 is set to 65536, that is, 18.2 Hz. Every tick causes a hardware interrupt to channel 0 of the 8259A interrupt controller. On system start, the interrupt controller is initialized so that interrupt 8 will keep track of the time of day. This interrupt routine is then redirected to software interrupt 1C (hex). The service routine of interrupt 1C is only an IRET instruction (return from interrupt) and offers an opportunity to the user to get control of the timer interrupt. IBM advises users to utilize this clock for all timing purposes in order to obtain maximum compatibility with all models of the computer family. The company promises to maintain compatibility in future systems (International Business Machines, 1986).

TIMER

Listing 2 contains the program listing in assembly language, which demonstrates the programming of delay and response latency routines. The redirection of the timer interrupt is accomplished by subroutine SETINT. (The use of register BX as relocation factor will be explained later.) First, the address of the old interrupt routine is determined and saved in the two variables INTSEG (segment) and INTOFS (offset). Then the interrupt vector is set so that it points to the routine NEWINT, which then handles the timer interrupt. NEWINT simply increments the counter INTCNT every time it is called by the timer interrupt 1c. In order to obtain a higher resolution than the actual 18.2 Hz, FAST changes the programmed value of counter 0 of the 8253 timer/counter chip to 4A9 (hex). The interrupt frequency is thereby changed to 1000.1522 Hz. The resolution then is 1 msec, which is sufficient for most applications. Since an exact frequency of 1 kHz is not programmable, a deviation of 152 nsec per millisecond is accumulated. If the time intervals desired are considerably larger than 1 sec, measures should be taken in calling the BASIC program to correct

for the error. If a different resolution is desired, another counter value can be calculated by the formula Count = INT(1.1931817/TargetFrequency).

The delay function is implemented in the main procedure. The principle is to initialize the counter INTCNT to zero, and to compare the millisecond argument passed by the caller with the contents of INTCNT. If INTCNT is still lower than delay time, a jump back is performed to continue comparison (see label DELAY).

The latency function (following delay) also clears the counter. Then the keyboard buffer is flushed to ensure that no prior keypress is misinterpreted as a valid response. Next, CX is loaded with the argument RESULT, which on input contains the maximum response time allowed in milliseconds. Then the latency loop is entered, which first checks if maximum response time is exceeded. If so, the loop is terminated, and parameters RESULT and KEY are set to zero. Otherwise, software interrupt 16 (hex) is called. Function 1 of interrupt 16 sets the zero flag if no character is typed. When a keypress (i.e., subject's response) occurs, the actual value of INTCNT is written to the argument RESULT. Any other input source (e.g., Centronics, game ports, or digital I/O devices) can be used to substitute for this method, which might be inappropriate for some applications.

The maximum time period that can be used in the delay and latency routines is 65,535 msec (values must be passed in "two's complement" coding, where numbers above 32767 are represented as negative numbers: IF x > 32767 THEN x = x − 65536). This also holds for maximum response time allowed.

Changing the timer interrupt frequency results in the side effect that the system's time of day is lost. After the program is run, the time should be updated by polling the battery clock.

ADDITIONAL ASSEMBLY FEATURES

To present a warning stimulus, the video controller is disabled by clearing the video enable bit of the CRT mode register that blanks the screen (CRT_DISABLE). Thereby, it is possible to change the contents of the video memory without any visible changes on the display. At this point in the program flow, it is important to clear the delay counter, because the timer interrupt already increments the counter every millisecond. During delay timing the assembly routine swaps the RAM screen to the location of the video screen (SWPSCRN). SWPSCRN determines the kind of display adapter used in the system and copies the RAM area to the display area. Since the starting address of the RAM area is passed as an argument, it is possible to define different RAM screens in the BASIC program and use them as imperative stimuli. When delay has finished, the display is enabled again (CRT_ENABLE) and the latency counter is cleared. From that point, the interrupt routine measures the response latency.

In order to tie response latency measurement to the vertical retrace of the video controller, routine CRT_ENABLE contains a commented area of code, which can be included in the program by deleting the comment signs. This piece of code will wait for the vertical retrace bit to indicate the moment of synchronization. From then on latency will be measured. This feature is not available for the IBM monochrome display adapter, which does not provide this information. Since Hercules Monochrome Graphics Cards use a different port and logic, the code for both Hercules and Color Graphics adapters is included.

Whereas implementing this synchronization suppresses additional artificial variance in the response latency data, another constant bias should be taken into account. The time elapsed from starting the counter to the actual presentation depends on the vertical screen position of the stimulus. This can be easily corrected by the formula:

time=y_position*((1000/monitor_refresh_rate)/No_of_lines).

Multilanguage programming from BASIC requires the following considerations: (1) When loaded by the BLOAD command, a binary file must have a BSAVE header (see BS_Hdr, Listing 2), which contains the identification (0FD hex), four unused bytes, and the module size in bytes. Module size is calculated by the assembler using the HEADER and TRAILER locations. The 7-byte BSAVE header will be the first in the binary file and must be removed when DATA statements are generated. (2) Assembly routines callable from BASIC must be relocatable. This requirement precludes the declaration of variables inside the routines. In order to handle this restriction, the procedure RELOC calculates a relocation factor, which is the offset to BASIC's code segment, and returns it in BX. BX then points to the beginning of the assembly routine. At the end of the listing, three variables are defined. Whenever these variables are addressed, BX is added to the offset of the variables. (This routine assumes that CS and DS point to the same segment.)

REFERENCES

DLHOPOLSKY, J. G. (1983). Machine language millisecond timers for the Z-80 microprocessor. *Behavior Research Methods & Instrumentation*, **15**, 511-520.

FEMANO, P. A., & PFAFF, D. W. (1983). Time-interval acquisition on a 6502-based microcomputer. *Behavior Research Methods & Instrumentation*, **15**, 521-529.

INTERNATIONAL BUSINESS MACHINES (1986). *Technical reference* (IBM Personal Computer AT 6183355). Boca Raton, FL: Author.

SARGENT, M., III, & SHOEMAKER, R. L. (1984). *The IBM personal computer from the inside out*. Reading, MA: Addison-Wesley.

(Manuscript received October 31, 1986;
revision accepted for publication January 30, 1987.)

LISTING 1
BASIC Demonstration Program

```
10 'Demonstration program for calling
20 'assembly language routines for
30 'delay, latency measurement, and
40 'presentation of video stimuli
50 'Program initialization **********
100 OPTION BASE 1:DEFINT I-N
110 CLS:LOCATE 1,1,1:KEY OFF
120 'Declare all variables forward to
130 'avoid problems with machine code
140 MSG$="Attention - Presentation"
150 'Change next line for your design
160 NWORDS=2:NTRIALS=5
170 ISUBRT=0:LAT=0:KEYP=0:I=0:J=0:C=0
180 K=0:L=0:A$="A":MS=2000:MAXRES=3000
190 ISIMAX=10:ISIMIN=2:ISTIM=0:TIME=0
200 CHECKSUM=31968:N=0
210 DIM TEXT$(NWORDS)
220 FOR I=1 TO NWORDS
230   READ TEXT$(I)
240 NEXT I
250 DIM ISR(NTRIALS*NWORDS)
260 FOR I=1 TO NWORDS*NTRIALS
270   READ ISR(I)
280 NEXT I
290 DIM IASM(200):DIM ISCRN(2000)
300 DIM ISI(10)
310 DIM LATB(NWORDS*NTRIALS)
320 ISUBRT=VARPTR(IASM(1))
330 GOSUB 6000
340 'Change 330 for .BIN-file to:
350 'BLOAD "TIMER.BIN",ISUBRT
360 LOCATE 10,19,1
370 PRINT "Stimulus presentation";
380 PRINT " demonstration program"
390 GOSUB 2010:'Determine ISIs
400 '***** main loop *****************
410 FOR ISTIM=1 TO NWORDS*NTRIALS
420   TIME=TIMER:' get actual time
430   LOCATE 4,23,0:PRINT MSG$
440   GOSUB 1020:'Fill RAM screen
450   WHILE TIMER<TIME+ISI(ISTIM)
460   WEND: 'wait
470   GOSUB 3010
480 NEXT ISTIM:'*** end of main loop
490 GOSUB 4000:'Statistics
500 A$=INKEY$:IF A$<>"" THEN 500
510 LOCATE 22,1,1:END
1000 '*** Subroutines *****************
1010 '*** Fill RAM screen *************
1020 FOR I=1 TO 2000:' Clear RAM screen
1030   ISCRN(I)=&H720
1040 NEXT I
1050 L=LEN(TEXT$(ISR(ISTIM)))
1060 J=80*12+INT((80-L)/2)
1070 FOR I=1 TO L:' write text
1080   A$=MID$(TEXT$(ISR(ISTIM)),I,1)
1090   ISCRN(J)=7*256+ASC(A$)
1100   J=J+1
1110 NEXT I
1120 RETURN:'*************************
2000 '*** Interstimulus intervals *****
2010 LOCATE 22,5,1
2020 PRINT "Enter min,max ISI in sec ";
2030 INPUT ISIMIN,ISIMAX
2040 IF ISIMIN>=3 THEN 2060
2050 PRINT"min ISI too small":GOTO 2010
2060 CLS:RANDOMIZE TIMER
2070 FOR I=1 TO NWORDS*NTRIALS
2080   L=INT(RND*(ISIMAX-ISIMIN+1))
2090   ISI(I)=L+ISIMIN
2100 NEXT
2110 RETURN:'*************************
3000 '*** Experimental presentation ***
3010 LAT=MAXRES:'max response time
3020 ISUBRT=VARPTR(IASM(1))
3030 CALL ISUBRT(MS,ISCRN(1),LAT,KEYP)
3040 CLS:LATB(ISTIM)=LAT
3050 RETURN:'*************************
4000 '** means & standard deviations **
4010 DIM ICNT(NWORDS),SUM(NWORDS)
4020 DIM RMEAN(NWORDS),SQ(NWORDS)
4030 DIM SDV(NWORDS)
4040 '*** Response Latency Means ******
4050 FOR I=1 TO NWORDS*NTRIALS
4060   ICNT(ISR(I))=ICNT(ISR(I))+1
4070   SUM(ISR(I))=SUM(ISR(I))+LATB(I)
4080   SQ(ISR(I))=SQ(ISR(I))+LATB(I)^2
4090 NEXT I
4100 FOR I=1 TO NWORDS
4110   RMEAN(I)=SUM(I)/ICNT(I)
4120   X=(SQ(I)-(SUM(I)^2)/ICNT(I))
4130   SDV(I)=SQR(X/ICNT(I))
4140 NEXT I
4150 CLS:LOCATE 5,30:PRINT "RESULTS"
4160 LOCATE 6,30:PRINT"=======":PRINT
4170 PRINT "Stimulus","Mean Response";
4180 PRINT " Latency (msec)    ";
4190 PRINT "Standard Deviation":PRINT
4200 FOR I=1 TO NWORDS
4210   PRINT TEXT$(I);:LOCATE 9+I,22
4220   PRINT USING "#####.###";RMEAN(I)
4230   LOCATE 9+I,45
4240   PRINT USING "#####.###";SDV(I)
4250 NEXT
4260 RETURN:'*************************
5000 '*********************************
5010 'DATA for presentation-material.
5020 'Number of words = NWORDS
5030 DATA left,right
5040 'DATA for order of presentation.
5050 'Number of values = NWORDS*NTRIALS
5060 DATA 1,2,2,1,1,2,2,1,1,2
6000 '*** Basic-Loader ***************
6010 '*** for assembly routines *******
6020 FOR I=0 TO 292
6030   READ J:POKE(ISUBRT+I),J
```

LISTING 1 (Continued)

```
6040   C=C+J:' Checksum
6050 NEXT
6060 IF C=CHECKSUM THEN RETURN
6070 PRINT CHR$(7);"Check sum error",C
6080 END
9000 DATA &H90,&H90,&H55,&H8B,&HEC,&H06
9010 DATA &HE8,&H0B,&H01,&HE8,&HA5,&H00
9020 DATA &HE8,&HD4,&H00,&HE8,&H87,&H00
9030 DATA &HC7,&H87,&H26,&H01,&H00,&H00
9040 DATA &H8B,&H76,&H0A,&HE8,&H4B,&H00
9050 DATA &H8B,&H76,&H0C,&H8B,&H0C,&H39
9060 DATA &H8F,&H26,&H01,&H75,&HFA,&HE8
9070 DATA &H57,&H00,&HC7,&H87,&H26,&H01
9080 DATA &H00,&H00,&HB8,&H00,&H0C,&HCD
9090 DATA &H21,&H8B,&H76,&H08,&H8B,&H0C
9100 DATA &H3B,&H8F,&H26,&H01,&H7C,&H0C
9110 DATA &HB4,&H01,&HCD,&H16,&H74,&HF4
9120 DATA &H8B,&H8F,&H26,&H01,&HEB,&H04
9130 DATA &H33,&HC9,&H8B,&HC1,&H8B,&H7E
9140 DATA &H08,&H89,&H0D,&H8B,&H7E,&H06
9150 DATA &H89,&H05,&HE8,&H96,&H00,&HE8
9160 DATA &H6D,&H00,&H07,&H8B,&HE5,&H5D
9170 DATA &HCA,&H08,&H00,&H53,&HB4,&H0F
9180 DATA &HCD,&H10,&H5B,&H3C,&H07,&HB8
9190 DATA &H00,&HB0,&H74,&H03,&HB8,&H00
9200 DATA &HB8,&H8E,&HC0,&H33,&HFF,&HB9
9210 DATA &HD0,&H07,&HF3,&HA5,&HC3,&H1E
9220 DATA &HB8,&H40,&H00,&H8E,&HD8,&HBE
9230 DATA &H63,&H00,&H8B,&H14,&H83,&HC2
9240 DATA &H04,&HBE,&H65,&H00,&H8A,&H04
9250 DATA &HEE,&H1F,&HC3,&H1E,&HB8,&H40
9260 DATA &H00,&H8E,&HD8,&HBE,&H63,&H00
9270 DATA &H8B,&H14,&H83,&HC2,&H04,&HBE
9280 DATA &H65,&H00,&H8A,&H04,&H24,&HF7
9290 DATA &HEE,&H1F,&HC3,&H53,&HB8,&H1C
9300 DATA &H35,&HCD,&H21,&H8B,&HC3,&H5B
9310 DATA &H89,&H87,&H28,&H01,&H8C,&H87
9320 DATA &H2A,&H01,&HBA,&H0E,&H01,&H03
9330 DATA &HD3,&HFA,&HB8,&H1C,&H25,&HCD
9340 DATA &H21,&HFB,&HC3,&H8B,&H97,&H28
9350 DATA &H01,&H8B,&H87,&H2A,&H01,&H1E
9360 DATA &H8E,&HD8,&HFA,&HB8,&H1C,&H25
9370 DATA &HCD,&H21,&HFB,&H1F,&HC3,&H50
9380 DATA &HFA,&HB0,&H36,&HE6,&H43,&HB8
9390 DATA &HA9,&H04,&HE6,&H40,&H8A,&HC4
9400 DATA &HE6,&H40,&HFB,&H58,&HC3,&H50
9410 DATA &HFA,&HB0,&H36,&HE6,&H43,&HB8
9420 DATA &H00,&H00,&HE6,&H40,&H8A,&HC4
9430 DATA &HE6,&H40,&HFB,&H58,&HC3,&H9C
9440 DATA &H53,&HE8,&H08,&H00,&H2E,&HFF
9450 DATA &H87,&H26,&H01,&H5B,&H9D,&HCF
9460 DATA &HE8,&H05,&H00,&H81,&HC3,&HE2
9470 DATA &HFE,&HC3,&H5B,&H53,&HC3,&H00
9480 DATA &H00,&H00,&H00,&H00,&H00
```

LISTING 2
Assembly Language Timing Routine

```
        TITLE   Timer
; To built .BIN-file for BLOAD (BASIC) type:
;       MASM TIMER;
;       LINK TIMER;
;       EXE2BIN TIMER TIMER.BIN
;
        .radix 16               ; all numbers are defined as hex
;-------------------------------
; EQUATES
;-------------------------------
bios_dseg       EQU     40      ; BIOS Data Segment
addr_6845       EQU     63      ; Base-address of 6845 regs
crt_modset      EQU     65      ; Video-Mode-Register
timer0          EQU     40      ; port address of timer 0
cntrl           EQU     43      ; port address of 8253 control word regis
divfast         EQU     4A9     ; Divisor for 1 kHz (1000.1522 Hz)
divslow         EQU     0000    ; i.e. 65536 (18.2 Hz)
;
KEY             EQU     06      ; 4th argument (which KEY was pressed)
RESULT          EQU     08      ; 3rd argument (RESULT=Latency)
RAM             EQU     0A      ; 2nd argument (start-addrs of RAM-screen
MS              EQU     0C      ; 1st argument (No of MilliSeconds)
;-------------------------------
BS_Hdr  segment para            ; BSAVE-Header
        db      0fdh            ; bsave id
        dw      0,0             ; unused
        dw      trailer-header  ; module size
BS_Hdr  ends
;-------------------------------
basic   segment byte public 'code'
        assume  cs:basic,ds:basic
;
header  equ     $
```

LISTING 2 (Continued)

```
stimi   proc    far             ; main procedure
        nop                     ; for debugging purposes
        nop                     ; -> insert INT 3 for break point
        push    bp
        mov     bp,sp
        push    es              ; BASIC assumes es unchanged on return
        call    reloc           ; bx is our relocation factor
        call    setint          ; redirect timer interrupt (0CH)
        call    fast            ; set clock to 1kHz
        call    crt_disable     ; turn video off (-> warning stimulus)
        mov     word ptr [bx+intcnt],0  ; init counter
        mov     si,[bp+RAM]     ; get address of RAM-screen
        call    swpscrn         ; copy RAM-screen to video-screen
        mov     si,[bp+MS]      ; get address of MS-parameter
        mov     cx,[si]         ; MS= no of ms to wait
delay:  cmp     [bx+intcnt],cx  ; & wait
        jnz     delay           ; until counter=cx
        call    crt_enable      ; turn video on (-> imperative stimulus)
        mov     word ptr[bx+intcnt],0   ; init counter
        mov     ax,0c00         ; flush keyboard buffer with
        int     21              ; MS-DOS function 0CH
        mov     si,[bp+result]  ; get address of RESULT-Parameter
        mov     cx,[si]         ; cx = maximum response time allowed
latency:cmp     cx,[bx+intcnt]  ; Maximum response time exceeded ?
        jl      exceed          ; yes - terminate loop
        mov     ah,01           ; else use function 01 of INT 16
        int     16              ; KeyPressed? No Check for CTRL-C!
        jz      latency         ; jump, if no char typed !
        mov     cx,[bx+intcnt]  ; no of millisecs into cx
        jmp     short back      ; back to BASIC
exceed: xor     cx,cx           ; time exceeded: RESULT = KEY = 0
        mov     ax,cx
back:   mov     di,[bp+result]  ; get adrs of RESULT-parameter
        mov     [di],cx         ; RESULT=cx
        mov     di,[bp+KEY]     ; get address of KEY-parameter
        mov     [di],ax         ; KEY=(Extended) Code of key pressed
        call    slow            ; clock back to 18.2 Hz
        call    resint          ; & reset interrupt
        pop     es              ; restore es (BASIC wants it)
        mov     sp,bp
        pop     bp
        ret     8               ; back to BASIC (drop 4 args from stack)
stimi   endp
;-------------------------------
swpscrn proc    near            ; swap video- and RAM-screens
; on input: ds:si points to Seg:offs of RAM-area
        push    bx              ; int 10 changes bh !
        mov     ah,0f           ; First check monitor type
        int     10              ;    with BIOs Interrupt 10H
        pop     bx
        cmp     al,7            ; 7 => Monochrome display
        mov     ax,0b000        ; video segment of monochrome
        jz      swp1
        mov     ax,0b800        ; video segment if not monochrome
swp1:   mov     es,ax           ; segment to es
        xor     di,di           ; Cursorposition (0,0) Offset=0
        mov     cx,80D*25D      ; counter= 80 cols x 25 lines
        rep     movsw           ; copy RAM- to Video-area
        ret
swpscrn endp
;-------------------------------
crt_enable      proc near       ; enable video-controller 6845
        push    ds              ; save BASICs data segment
        mov     ax,bios_dseg    ; point ds to BIOS
        mov     ds,ax
        mov     si,addr_6845    ; get base-address
        mov     dx,[si]         ;  of 6845-register
;
; The following lines synchronize the start of latency measurement
; to the vertical retrace signal.
;
; for Hercules Monochrome Adapter
;       add     dx,6            ; Offset of CRT-Status-Register
;crt_1: in      al,dx           ; read CRT-Status
```

LISTING 2 (Continued)

```
;       test    al,80           ; test VSYNC-Bit
;       jnz     crt_1           ; wait for vertical retrace
;       sub     dx,6            ; adjust port-address
;
; for Color Graphics and Enhanced Graphics Adapters
;       add     dx,6            ; Offset of CRT-Status-Register
;crt_1: in      al,dx           ; read CRT-Status
;       test    al,8            ; test VSYNC-Bit
;       jz      crt_1           ; wait for vertical retrace
;       sub     dx,6            ; adjust port-address
        add     dx,4            ; Offset of CRT-Mode-Register
        mov     si,crt_modset   ; get value of actual CRT-mode
        mov     al,[si]         ; into al
        out     dx,al           ; and output to turn video on
        pop     ds              ; restore data segment
        ret
crt_enable      endp
;------------------------------
crt_disable     proc near       ; disable video-controller 6845
        push    ds              ; save BASICs data segment
        mov     ax,bios_dseg    ; point ds to BIOS
        mov     ds,ax
        mov     si,addr_6845    ; get base-address
        mov     dx,[si]         ;  of 6845-register
        add     dx,4            ; Offset of CRT-Mode-Register
        mov     si,crt_modset   ; get value of actual CRT-mode
        mov     al,[si]         ; into al
        and     al,0F7          ; clear Video-Enable-Bit (3)
        out     dx,al           ; and output to turn video off
        pop     ds              ; restore data segment
        ret
crt_disable     endp
;------------------------------
setint  proc    near            ; set "newint" as timer interrupt routine
        push    bx              ; save relocation factor
        mov     ax,351c         ; get old timer interrupt
        int     21              ; DOS function 35H
        mov     ax,bx           ; offset to ax
        pop     bx
        mov     [bx+intofs],ax  ; save old offset
        mov     [bx+intseg],es  ; & old segment
        mov     dx,offset newint ; offset of int routine
        add     dx,bx           ; add our relocation factor
        cli                     ; disable interrupts
        mov     ax,251c         ; set new interrupt vector
        int     21
        sti                     ; enable interrupts
        ret
setint  endp
;------------------------------
resint  proc    near            ; restore old timer interrupt routine
        mov     dx,[bx+intofs]  ; get old offset
        mov     ax,[bx+intseg]  ; & old segment
        push    ds
        mov     ds,ax
        cli
        mov     ax,251C
        int     21
        sti
        pop     ds
        ret
resint  endp
;------------------------------
fast    proc    near            ; change frequency of timer channel 0
        push    ax              ;   interrupt to 1 kHz
        cli                     ; disable interrupts
        mov     al,36           ; mode 3 for timer channel 0
        out     cntrl,al        ; output
        mov     ax,divfast      ; new divisor for 1 kHz
        out     timer0,al       ; output low byte
        mov     al,ah           ; high byte to low byte
        out     timer0,al       ; output high byte
        sti                     ; enable interrupts
        pop     ax
```

LISTING 2 (Continued)

```
        ret
fast    endp
;------------------------------
slow    proc    near            ; change frequency of timer channel 0
        push    ax              ;  back to 18.2 Hz
        cli                     ; disable interrupts
        mov     al,36           ; mode 3 for timer channel 0
        out     cntrl,al        ; output
        mov     ax,divslow      ; old divisor for 18.2 Hz
        out     timer0,al       ; output low byte
        mov     al,ah           ; high byte to low byte
        out     timer0,al       ; output high byte
        sti                     ; enable interrupts
        pop     ax
        ret
slow    endp
;------------------------------
newint  proc    far             ; new timer - interrupt handler
        pushf                   ; push flags
        push    bx              ; & register bx
        call    reloc           ; get relocation-factor
        inc     cs:word ptr [bx+intcnt] ; increment counter
        pop     bx              ; restore register
        popf                    ; & flags
        iret                    ; return from interrupt
newint  endp
;------------------------------
reloc   proc    near            ; return relocation factor in bx
        call    setbx
        add     bx,offset header-7-$    ; bx=relocation-factor
        ret
setbx:  pop     bx              ; get return address (from call)
        push    bx              ; bx=RET-adrs (Offs)
        ret
reloc   endp
;------------------------------
intcnt  dw      0               ; Interrupt-counter
intofs  dw      0               ; Offset of old INT 0C
intseg  dw      0               ; Segment - " -
;
trailer equ     $
basic   ends
        end
```

(Manuscript received October 31, 1986;
revision accepted for publication January 30, 1987.)