Behavior Research Methods, Instruments, & Computers
1987, 19 (1), 30-35

COMPUTER TECHNOLOGY

Millisecond interval timer and auditory reaction time programs for the IBM PC

ROGER GRAVES and RON BRADLEY
University of Victoria, Victoria, British Columbia, Canada

Many types of behavioral research require the determination of elapsed time, for example to establish interstimulus intervals and to measure reaction time. The use of an IBM PC for on-line control of such applications is limited by the poor timing resolution ordinarily available. The IBM BIOS time information that is used for the BASIC TIMER function can result in interval timing errors as great as 110 msec. A machine language subroutine is described that can provide 1-msec accuracy. A BASIC program is also described that employs this subroutine to measure auditory reaction time.

Many applications for a personal computer as an online controller in psychology studies require an interval timing capability with millisecond resolution. Recording of reaction times is a prime example. Unfortunately, the TIMER function available from IBM BASIC provides time readout in 55-msec increments. Using TIMER twice for interval timing would thus result in potential errors as great as 110 msec. (The fact that the values returned by TIMER contain two decimal places suggests a precision of 10 msec; however, actual precision is 55 msec, as evidenced in successive returned values that remain unchanged, and then suddenly change by .05 or .06 sec.) The cause of this poor resolution rests with the IBM BIOS, which updates time information only every 55 msec.

Smith and Puckett (1984) discussed the problem of the IBM PC time resolution and reported that the 8253 timing chip used in the IBM PC actually contains timing information with microsecond resolution. They described machine language programs that can be used to access this information. Their programs, however, are not written in a form that can be used directly for research timing applications using BASIC. Appendices A and B show two machine language programs that incorporate Smith and Puckett's approach in a form that can be called from BASIC.

The first program, TIMERSET, is used to set the timer chip into Mode 2. The second program, TMRREAD, is used to return 6 bytes of data (3 basic integer variables), representing the number of clock counts since the preceding midnight. The Appendix A and B listings include brief documentation of program operation; more complete information can be found in Smith and Puckett's (1984) article. For use with interpreted BASIC, these two programs were assembled and linked with the IBM Macro Assembler into one machine language program, TIMER.BIN. For use with compiled BASIC programs, TIMERSET and TMRREAD are written in a form that allows them to be linked with compiled programs.

The accuracy of the timing provided by the TMRREAD program was checked with an interpreted BASIC program, TESTCRT.BAS, which measures the time for 10 frames of the color monitor display. The correct time is 166.67 msec. The results for 100 trials using an IBM PC/XT were: mean of 167.07 msec, standard deviation of 0.06 msec, and maximum error of 0.61 msec. These data indicate that good timing accuracy can be obtained even with interpreted BASIC. Similar results were obtained with another IBM PC/XT and with several "true-compatible" PC/XT clones. The program did not return correct times on a semicompatible Zenith Z-158 running GW-BASIC. A compiled BASIC version of this program was linked with TIMERSET and TMRREAD to produce an executable program (TESTCRT.EXE). The results on the IBM PC/XT—mean of 166.88 msec, standard deviation of 0.02 msec, and maximum error of 0.34 msec—demonstrate that the decreased overhead of compiled programs will improve accuracy of timing.

Appendix C provides a BASIC program, RT.BAS, which illustrates how TIMER.BIN can be used to establish predetermined time intervals (interstimulus intervals) and to measure event latencies (subject reaction time). The listing of RT.BAS includes descriptive information on the program operation. Basically, the program produces a beep stimulus and then uses TMRREAD to measure the subject's reaction time to press the space bar. Following

Preparation of this article was supported in part by Grant A-1021 from the National Science and Engineering Research Council. This work would not have been possible without the contributions of Tom Allen and Eric Hargreaves to earlier versions of the programs.

Requests for reprints should be sent to: Roger Graves, Department of Psychology, University of Victoria, Victoria, British Columbia V8W 2Y2, Canada.

an intertrial interval, which is also established using TMRREAD, the stimulus is presented again for another trial, and so forth. To prevent the subject from anticipating the stimulus, the intertrial interval is made random in the range of 2,000-8,000 msec. Since the SOUND command in BASIC appears to produce sound in synchrony with timer interrupts, it was necessary to control sound generation directly using a method described by Norton (1985). During the development of RT.BAS, it was also found that the TIMER and ON TIMER functions of BASIC could not be used in the same program as TMRREAD (or TIMER.BIN) since they impair the accuracy of timing using TMRREAD.

Timing performance using the IBM PC was checked against reaction times recorded independently using a Gerbrands digital millisecond timer. A Schmitt trigger circuit connected to the IBM PC speaker leads started the timer when audio voltage appeared at the speaker. A Gerbrands voice-operated relay stopped the timer when a microphone placed at the keyboard detected the sound of the keypress response. Results for 20 trials using the standard IBM PC keyboard showed times that were slow on average by 18.4 msec (*SD* = 4.3 msec). Results for 20 trials using the keyboard from a clone with the same IBM PC showed times that were slow on average by 36.7 msec (*SD* = 2.9 msec). These discrepancies, which presumably arise from delays in the keyboard, indicate that the keyboard is not an ideal response device for critical applications. However, for applications in which random errors of the order of these standard deviations (4 msec) can be tolerated, the times can be corrected by subtracting the mean error. This was done in the RT.BAS program listed in Appendix C. Greatly improved accuracy was obtained by using the buttons on a joystick as the response device (Gravis Mk IV joystick, connected to an AST SixPacPlus game port). With the same independent timing setup and 20 trials, the PC times were slow on average by 0.55 msec (*SD* = 0.51 msec) with no discrepancy greater than 1 msec. These latter results confirm that good reaction time accuracy can be obtained with the IBM PC even with interpreted BASIC. The listing of RT.BAS in Appendix C includes the commands that were employed to detect game button responses.

AVAILABILITY

A diskette containing RT.BAS, TESTCRT.BAS; the ASCII (.ASM) and assembled (.OBJ) versions of TIMERSET and TMRREAD; and the ASCII compiler (.ASC), compiled (.OBJ), and linked (.EXE) versions of TESTCRT (which also serve to illustrate how the timing programs are called in compiled BASIC) is available for $6 (Canadian) from the Neuropsychology Clinic, Department of Psychology, University of Victoria, Victoria, British Columbia V8W 2Y2, Canada.

REFERENCES

NORTON, P. (1985). *The programmer's guide to the IBM PC*. Bellview, Washington: Microsoft Press.

SMITH, B., & PUCKETT, T. (1984, April). Life in the fast lane. *PC Tech Journal*, 63-74.

APPENDIX A
Listing of TIMERSET Program

```
PAGE
     title   TIMERSET
     name    TIMERSET      ;module

comment |

Environment:  IBM PC, tested under DOS 3.1

Segmentation:  Program segment CODE, byte aligned

Specifications: This module is designed to be called with the
                BASIC "CALL" statement. The result of calling
                this routine will be the placing of Counter 0
                of the 8253 timer chip into Mode 2 and setting
                the Counter 0 count value to zero (equivalent
                to 65536). In this mode the timer updates the
                BIOS registers after each 65536 system clock
                cycles and can retain the number of system clock
                cycles ("residual count") since the last update.

Implementation: Called from BASIC

                 example:

                   10 DEF SEG =
                   20 BLOAD"TIMERSET.BIN"
                   30 DEFINT A-Z
                   40 A = 0
                   50 CALL A()
```

APPENDIX A (Continued)

```
Original Code by: Bob Smith, Tom Puckett
                  PC TECH JOURNAL, April 1984

Modified by: R. Bradley, Aug. 1986

|

CODE SEGMENT BYTE PUBLIC 'CODE'

     ASSUME CS:CODE
     PUBLIC TIMERSET

;equates.....

TIMER_0   EQU   040H                        ;8253 Counter 0 Port
TIMER_CTL EQU   043H                        ;8253 Control Port
TIMER_SET EQU   00110100B                   ;8253 Counter 0,
                                            ;set to Mode 2
;timerset module.....

TIMERSET  PROC  FAR

        PUSH    AX
        PUSH    CX
        MOV     AL,TIMER_SET
        OUT     TIMER_CTL,AL                ;set Counter 0 to mode 2
        XOR     AL,AL                       ;zero AL register
        NOP                                 ;8253 recovery time
        OUT     TIMER_0,AL                  ;set low order count
                                            ;byte to zero
        NOP                                 ;8253 recovery time
        NOP
        OUT     TIMER_0,AL                  ;set high order count
                                            ;byte to zero
        MOV     CX,20000                    ;loop to ensure
LOOP:   LOOP    LOOP                        ;previous count ends
        POP    CX
        POP    AX
        RET

TIMERSET        ENDP
CODE ENDS
        END
```

APPENDIX B
Listing of TMRREAD

```
PAGE
     title   TMRREAD
     name    TMRREAD      ;module

comment |

Environment:  IBM PC, tested under DOS 3.1

Segmentation:  Program segment CODE, byte aligned

Specifications: This module is designed to be called with
                the BASIC "CALL" statement. The result of
                calling this routine will be the return of
                microsecond timing data from the 8253 timer
                and the BIOS second and hour fields.
                The microsecond count is obtained from Counter 0
                of the 8253 timer. In order for the timer count
                to be valid, Counter 0 must have been
                initialized for Mode 2 operation by calling the
                TIMERSET module.

     **********************************************************
     "MODE 2 OPERATION OF 8253 MUST BE INVOKED BEFORE THIS
                      MODULE IS RUN"
     **********************************************************
```

APPENDIX B (Continued)

```
Implementation: Called from BASIC

               example:

                   10 DEF SEG =
                   20 BLOAD"TMRREAD.BIN"
                   30 DEFINT A-Z
                   40 A = 0
                   50 CALL A(HIGH, LOW, RESIDUAL)

                   where   HIGH = BIOS timer high
                           LOW  = BIOS timer low
                       RESIDUAL = Counter 0 residual count


Returns: will return BIOS timer fields, and the Counter 0
        residual count. Data is returned to the BASIC
        program using the BASIC stack.

        BASIC can only accept integer values in the range
        of -32768 to 32767 so routine subtracts 32768 from
        the data and passes back valid BASIC integers.

        STACK POINTER + 10  = TIMER HIGH COUNT
        STACK POINTER + 8   = TIMER LOW COUNT
        STACK POINTER + 6   = RESIDUAL COUNT

Original Code by: Bob Smith & Tom Puckett,
                  PC TECH JOURNAL, April 1984

Modified by:      R. Bradley, Aug. 1986

suggested invocation:

   Called from BASIC program. Module is first linked with
   TIMERSET.EXE and can then be run from BASIC with the
   CALL NUMBER(HIGH,LOW,RESIDUAL) command after being
   loaded into the BASIC program.

!

BIOS_SEG   EQU   040H
BIOS_DATA SEGMENT AT 040H

     ORG  06CH

TIMER_LOW   DW   ?          ;bios timer storage
TIMER_HIGH  DW   ?          ;words

BIOS_DATA ENDS

CODE SEGMENT BYTE PUBLIC 'CODE'    ;code segment

      ASSUME CS:CODE
      PUBLIC TMRREAD

;equates.....

TIMER_0   EQU   040H        ;8253 counter 0 port
TIMER_CTL EQU   043H        ;8253 control port
TIMER_LATCH EQU 00H         ;8253 command - save current
                            ;               residual count

;tmrread module.....

TMRREAD PROC FAR

     PUSH BP
     MOV  BP,SP
     PUSH DS
     POP  ES               ;use ES for storing data
     MOV  AX,BIOS_SEG
     MOV  DS,AX            ;new DS points to BIOS

      ASSUME DS:BIOS_DATA,ES:NOTHING

     MOV  AL,TIMER_LATCH   ;save current residual count
```

APPENDIX B (Continued)

```
        CLI                     ;disable interrupts
        OUT  TIMER_CTL,AL       ;command for 8253 timer
        MOV  BX,TIMER_LOW       ;get BIOS timer values
        MOV  CX,TIMER_HIGH
        IN   AL,TIMER_0         ;get 8253 count
        MOV  AH,AL
        NOP
        IN   AL,TIMER_0
        STI
        XCHG AH,AL              ;put timer count in
        NEG  AX                 ;correct order and invert
        XCHG AX,CX
        SUB  AX,32768           ;set up data for BASIC
        SUB  BX,32768           ;integers
        SUB  CX,32768
        MOV  DX,BX              ;move timer low to DX
        MOV  BX,[BP+10]         ;get TIMER HIGH^ from
        MOV  ES:[BX],AX         ;stack and load with AX
        MOV  BX,[BP+8]          ;get TIMER LOW^ from
        MOV  ES:[BX],DX         ;stack and load with DX
        MOV  BX,[BP+6]          ;get 8253 count^ from
        MOV  ES:[BX],CX         ;stack and load with CX
        PUSH ES
        POP  AX
        MOV  DS,AX              ;restore old DS
        POP  BP                 ;restore old BP
        RET  6

TMRREAD ENDP
CODE    ENDS
        END
```

APPENDIX C
Listing of RT.BAS

```
100 GOTO 140 '                          AUDITORY REACTION TIME PROGRAM
110 '                         R. GRAVES, UNIVERSITY OF VICTORIA, VICTORIA, B.C.
120 '
130 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
140 KEY OFF: CLS                    'CLEAR SCREEN
150 DEFINT A-Z                      'DEFINE VARIABLES, CONSTANTS & FUNCTIONS
160 P=&H60                                 'PORT FOR KEYBOARD RESPONSE
165 'P=&H201                                'PORT FOR GAME BUTTON RESPONSE
170 Q=&H61                                 'PORT FOR CONTROLLING SPEAKER
180 N=255                                  'MASK TO DETECT KEYBOARD RESPONSE
185 'N=240                                  'MASK TO DETECT GAME BUTTON RESPONSE
190 I=0                                    'INDEX FOR LOOP
200 K=80                                   'CONSTANT FOR 80 MSEC LENGTH OF SOUND
210 H=0: H1=0: H2=0                        'HIGH BYTES OF TIME COUNT
220 L=0: L1=0: L2=0                        'LOW  BYTES OF TIME COUNT
230 R=0: R1=0: R2=0                        'RESIDUAL BYTES OF TIME COUNT
240 ITI#=0#                                'CONSTANT FOR INTERTRIAL INTERVAL
250 CORRECTION#=18.4#                      'MEASURED MEAN ERROR WITH IBM-PC
                                              KEYBOARD  (KEYBOARD DELAY?)
255 'CORRECTION#=0#                         'NO CORRECTION NEEDED FOR GAME BUTTON
260 K#=41#                                 'MSECS TAKEN BY TIME COMPARISON
270 COUNT=CINT(1193280!/1000!)             'TIMER COUNT FOR A FREQ OF 1000 HZ
280 LO.COUNT=COUNT MOD 256                 'CALCULATE LOW ORDER BYTE
290 HI.COUNT=COUNT/256                     'CALCULATE HIGH ORDER BYTE
300 SNDOFF=INP(&H61)                       'OLD VALUE OF PORT H61 (FOR SPEAKER)
310 SNDON=(SNDOFF OR &H3)                  'NEW VALUE TO TURN SPEAKER ON
320 T1#=0: T2#=0: T3#=0                    'TIME VARIABLES
330 RT#=0                                  'REACTION TIME
340 D#=32768#                              'FOR ADDING TO COUNT VARIABLES
350 E#=65536#                              'FOR MULTIPLYING COUNT VARIABLES
360 F#=.000838095#                         'NUMBER OF MILLISECONDS PER COUNT
370 A%=0: B%=0                             'LOCATIONS OF MACHINE LANG SUBROUTINES
380 DEF FNT#(H,L,R)=(((CDBL(H)+D#)*E#*E#)+((CDBL(L)+D#)*E#)+(CDBL(R)+D#))*F#
390 'FUNCTION TO CONVERT RETURNED COUNT VARIABLES TO NUMBER MSEC SINCE MIDNIGHT
400 DIM A(120)                             'ARRAY USED TO LOAD TIMING SUBROUTINES
410 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
420 A%=VARPTR(A(0))             'A% = START OF MEMORY FOR TIMING SUBROUTINES
430 BLOAD"TIMER.BIN",A%                             'LOAD TIMING SUBROUTINES
440 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
450 PRINT "       PRESS THE SPACE BAR AS QUICKLY AS YOU CAN WHEN YOU HEAR THE
BEEP":PRINT
```

APPENDIX C (Continued)

```
455 'PRINT " PRESS A BUTTON ON THE JOYSTICK AS QUICKLY AS YOU CAN WHEN YOU HEAR
THE BEEP"
460 '
470 OUT &H43,&HB6                'GET 8253 TIMER CHANNEL 2 READY FOR COMMAND
480 OUT &H42,LO.COUNT                   'LOAD LOW BYTE OF SOUND FREQ.
490 OUT &H42,HI.COUNT                   'LOAD HIGH BYTE OF SOUND FREQ.
500 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
510 '                ESTABLISH INTERTRIAL INTERVAL
520 A%=VARPTR(A(0))              'A% = START OF MEMORY FOR TIMERSET SUBROUTINE
530 CALL A%                             'SET 8253 TIMER CHANNEL 0 TO MODE 2
540 B%=VARPTR(A(0))+&H17                'B% = LOCATION OF TMRREAD SUBROUTINE
550 CALL B%(H,L,R): T2#=FNT#(H,L,R)     'T2# = CURRENT TIME
560 ITI#=(INT(RND*(7000))+2000)         'RANDOM NUMBER BETWEEN 2000 AND 8000
570 ITI#=ITI#+T2#-K#                    'TIME AT WHICH TO EXIT DELAY LOOP
580 CALL B%(H,L,R): IF FNT#(H,L,R)<ITI#  THEN GOTO 580
590                                     'DELAY FOR ITI MILLISECONDS
600 'USING INTERPRETED BASIC, THIS METHOD OF ESTABLISHING INTERVALS CAN NOT
610 'BE USED FOR INTERVALS LESS THAN 41 MSEC. - ALSO, THE INTERVALS MAY BE IN
620 'ERROR BY AS MUCH AS 41 MSEC. - THIS IS THE TIME TAKEN FOR THE
630 'TIME COMPARISON IN LINE 580.
640 'THIS IS BETTER THAN THE 110 MSEC POSSIBLE ERROR USING THE "TIMER" COMMAND.
650 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
660 '                 MEASURE REACTION TIME
670 A%=VARPTR(A(0))              'A% = START OF MEMORY FOR TIMERSET SUBROUTINE
680 CALL A%                             'SET 8253 TIMER CHANNEL 0 TO MODE 2
690 B%=VARPTR(A(0))+&H17                'B% = LOCATION OF TMRREAD SUBROUTINE
700 OUT Q,SNDON:CALL B%(H1,L1,R1)       'START SOUND AND GET CURRENT TIME
710 FOR I=1 TO K: NEXT                  'CONTINUE SOUND DURING LOOP DELAY
720 OUT Q,SNDOFF                        'STOP SOUND
730 DEF SEG=0: POKE 1050,PEEK(1052):DEF SEG 'CLEAR KEYBOARD BUFFER
735 '                                       'NO COMMAND FOR GAME BUTTONS
740 WAIT P,N:CALL B%(H2,L2,R2)
745 'WAIT P,N,N:CALL B%(H2,L2,R2)
750 '
760 'LINE 740 WAITS FOR KEYBOARD RESPONSE AND THEN GETS TIME AT RESPONSE
770 'LINE 745 WAITS FOR GAME BUTTON RESPONSE AND THEN GETS TIME AT RESPONSE
780 '
790 T1#=FNT#(H1,L1,R1)                  'T1# = TIME AT STIMULUS
800 T2#=FNT#(H2,L2,R2)                  'T2# = TIME AT RESPONSE
810 RT#=T2#-T1#-CORRECTION#             'RT# = REACTION TIME
820 PRINT USING "######";RT#,           'PRINT REACTION TIME IN MILLISECONDS
830 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
840 GOTO 560                            'REPEAT STIMULUS
850 '!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
860 'NOTE: THE CONVERSION IN LINE 790 TAKES 35 MILLISECONDS WHEN
870 'USING INTERPRETED BASIC.
880 'THEREFORE, THIS SHOULD NOT BE DONE UNTIL AFTER THE TIMED INTERVAL
890 'IN ORDER TO PREVENT ERRORS IN TIMING SHORT INTERVALS.
900 '
910 'TO USE THE AST SixPacPlus GAME PORT AND JOYSTICK BUTTONS FOR THE RESPONSE
915 'REPLACE LINES 160, 180, 250, 450, 730, AND 740 WITH LINES
                 165, 185, 255, 455, 735, AND 745 RESPECTIVELY.
```

(Manuscript received September 26, 1986;
revision accepted for publication December 12, 1986.)