COMPUTER TECHNOLOGY

More on millisecond timing and tachistoscope applications for the IBM PC

ROGER GRAVES and RON BRADLEY

University of Victoria, Victoria, British Columbia, Canada

This paper describes a method of establishing time intervals at a precision of better than 1 msec, using QuickBASIC 4.0, for running real time experiments on an IBM PC without an 8087 coprocessor. This method is far superior to the TIMER function provided by BASIC, which has a precision of only 110 msec. Also described is an assembly language subroutine that corrects a problem in QuickBASIC 3.0 and QuickBASIC 4.0 to allow proper switching between screens in the CGA text mode for use of the PC as a tachistoscope. A tachistoscope program listing shows how to use the screen subroutine and how to establish intertrial intervals and record reaction times using the timing subroutines.

In an earlier paper, Graves and Bradley (1987) described assembly language subroutines, TIMERSET and TMRREAD, that reprogram the IBM PC clock-timer and access its timing information in order to time events (e.g., reaction times) at 1-msec accuracy. This earlier paper also described a method for using these subroutines with BASICA to establish intervals (e.g., intertrial intervals). The interval precision provided by this method was only 35 msec, however, using interpreted BASICA on the standard IBM PC. Although 35 msec is better than the 110-msec accuracy offered by the TIMER function in BASIC (compiled Microsoft QuickBASIC as well as interpreted BASICA), it is insufficient for critical applications.

The TMRREAD subroutine obtains time information from the hardware quickly enough for submillisecond interval timing precision. The practical limitation is that this routine returns three integer values that must be converted to a single value representing the current time. The time required for this conversion sets a lower bound on the precision possible in establishing intervals, since current time cannot be checked more often than once every conversion time period when a program loop determines whether the interval has elapsed. Graves and Bradley's (1987) conversion method employs many double-precision real number additions and multiplications. This method is fast (slightly faster than the method described in this paper) if an 8087 coprocessor and good compiler software support are available. Without the 8087, however, this method still results in conversion times of about 10 msec, even when using compiled BASIC with 8087

emulation (QuickBASIC 4.0). The solution to this problem is to change the method of conversion. QuickBASIC 4.0 (and other programming languages) now provide a "long integer" variable type. When there is no 8087 coprocessor installed, arithmetic functions done with these long integers are considerably faster than calculations done with the floating point reals. With the use of long integer calculations, the conversion time can be reduced to less than 2 msec when running QuickBASIC 4.0 on a standard IBM PC/XT.

In order to establish real time intervals, one obtains the time at the start of the interval and then enters a loop that keeps reading the current time and checking to see if the time has exceeded the start time plus the desired interval. For example, if an interval of 100 msec is desired, the loop looks for a time of 100 msec greater than the start time. Assume, however, that the conversion takes 1.6 msec. If the elapsed time at the latest clock read were exactly 100 msec, then the actual elapsed time at the end of the conversion would be 101.6 msec, and the interval would be 1.6 msec too long. If the elapsed time were 99.9 msec, the loop would check the time again after the conversion, and the next value would be 101.5 msec reported at an actual time of 103.1 msec, resulting in an interval 3.1 msec too long. Since these potential errors range from 1.6 to 3.1 msec, an actual routine would look for a time 2.4 msec less than the desired interval in order to reduce the errors to within 0.8 msec of the desired interval. According to the DLYTEST program (see Appendix A), a correction constant of 2.4 msec is optimum for the interval generation subroutines using a standard IBM PC (with or without an 8087), producing errors within 0.8 msec of the desired interval.

A function, TIME, has been defined which internally calls TMRREAD, does the time conversion using long integer calculations, and then returns the current time in

Preparation of this article was supported in part by Grant A-1021 from the Natural Sciences and Engineering Research Council of Canada. Requests for reprints should be sent to Roger Graves, Department of Psychology, University of Victoria, Victoria, British Columbia V8W 2Y2, Canada.

units of 0.1 msec. This function is included in a QuickBASIC 4.0 subroutine module, TIMESUBS.BAS, shown as Appendix B. This module also contains a subroutine, DELAY, which provides a delay of the desired number of milliseconds from a supplied reference time. TIME and DELAY can very conveniently establish time intervals, as shown in the following code, which establishes a 100-msec interval:

```
T&=TIME          'Get current time.
                  'Do whatever is desired here, as
                  'long as the total execution time
                  'is less than 100 msec.
DELAY T&, 100     'Wait until the interval elapses.
```

Appendix A shows a QuickBASIC 4.0 program, DLYTEST.BAS, which illustrates the use of these subroutines for establishing time intervals. In addition, the program checks the accuracy of the intervals produced by DELAY. DLYTEST.BAS was compiled and linked to the timing subroutines using the QuickBASIC 4.0 package to produce a stand-alone .EXE file. The results over all intervals checked from 5 msec to 10,000 msec showed a maximum measured error of 0.8 msec on a standard IBM PC/XT with an 8087, and the same on a standard IBM PC without an 8087. This precision should be sufficient for the time interval requirements of behavioral applications.

Tachistoscope

Segalowitz (1987) described how the IBM PC can be used as a tachistoscope for presentation of text. The approach involves writing material to the various "screens" available in the CGA text mode, and then switching among these screens. Essentially the PC can be an eight-channel tachistoscope. We used this approach successfully with interpreted BASICA (IBM's version 3.0) and with compiled QuickBASIC 1.0. Unfortunately, newer versions of BASICA (Microsoft GWBASIC version 3.2) and Microsoft QuickBASIC (versions 3.0 and 4.0; 2.0 was not tried) cannot be used in this application without modification. The problem is that the SCREEN function in these languages is very slow and, worse, is variable in the time it takes to switch among the screens. (Apparently the addition of EGA support has resulted in more complex operation.) The consequence is uncertainty as to how many frames are being displayed and how long it has taken to present the stimulus field, rendering these languages useless for tachistoscope applications. The solution to this problem is to replace the faulty SCREEN function. Appendix C shows an assembly language subroutine, SCRIN, which provides the required function. It is for switching the "display page" in the CGA text modes. The SCREEN function should be used for switching the "active page" for writing information to the screens. SCRIN was linked with TIMESUBS, TIMERSET, and TMRREAD to make a library file called TSCOPE.

Segalowitz (1987) also described a method of recording reaction times to tachistoscope stimuli that requires

an add-on board. The assembly language timing routines described by Graves and Bradley (1987) can provide reaction time measurements in this application without requiring additional hardware.

Appendix D shows a QuickBASIC 4.0 demonstration tachistoscope program, TSDEMO.BAS. This program illustrates the use of the SCRIN subroutine, shows how TIME and DELAY can be used to establish inter-trial intervals, and shows how TMRREAD can be used to measure reaction time. (See Graves & Bradley, 1987, for information on obtaining improved reaction timing accuracy by using switch buttons connected to the joystick port

as the response device instead of the keyboard.) TSDEMO.BAS also measures the stimulus presentation time, which was found to agree with the expected time (16.67 msec per frame) for our CGA adaptor, but which was about 1.5% too long for our IBM EGA adaptor.

Bührer, Sparrer, and Weitkunat (1987) described different methods for interval timing and control of stimulus screens. Their timing method has several disadvantages in comparison with our approach; namely, timing accuracy in all applications may be in error by as much as 2 msec since time is recorded only in units of 1 msec, intervals are limited to a maximum of 65 sec, the BIOS time-of-day information is lost, and the frequency of timer interrupts is increased to one every millisecond. Calling the timing interrupt service 1,000 times every second adds substantial overhead whenever any timing is done and slows operations requiring maximum speed. Also, the frequent interrupts may themselves suffer interference from any function that momentarily turns off interrupts. Furthermore, our approach is considerably more "user friendly" in the way the timing routines are called. Bührer et al.'s screen control method is innovative, but most existing tachistoscope applications probably use the more conventional screen control approach described by Segalowitz (1987). Hence, the method of controlling the screen and measuring reaction times described in this paper require much less modification to existing programs than does conversion to Bührer et al.'s approach.

Availability

A diskette containing ASCII and compiled versions of the programs mentioned in this article is available from Roger Graves for \$10 (Canadian) to cover materials, postage, and handling.

REFERENCES

- BUHRER, M., SPARRER, B., & WEITKUNAT, R. (1987). Interval timing routines for the IBM PC/XT/AT microcomputer family. *Behavior Research Methods, Instruments, & Computers*, **19**, 327-334.
- GRAVES, R., & BRADLEY, R. (1987). Millisecond interval timer and auditory reaction time programs for the IBM-PC. *Behavior Research Methods, Instruments, & Computers*, **19**, 30-35.
- SEGALOWITZ, S. J. (1987). IBM PC tachistoscope: text stimuli. *Behavior Research Methods, Instruments, & Computers*, **19**, 383-388.

APPENDIX A

Listing of DLYTEST.BAS

```

'DLYTEST.BAS                                R. GRAVES, UNIVERSITY OF VICTORIA
'USE WITH TSCOPE LIBRARY (TIMESUBS+TIMERSET+TMRREAD+SCRN)
'PROGRAM TO TEST THE ACCURACY OF INTERVAL TIMING

DEFINT A-Z
DECLARE FUNCTION TIME& ( )
DECLARE SUB DELAY (STARTTIME&, DELAYDURATION&)
T1& = 0: T2& = 0                                'TIME VARIABLES, UNITS OF 0.1 MSEC
DLY& = 0                                          'DELAY INTERVAL, UNITS OF 1 MSEC
T# = 0                                           'TIME VARIABLE, UNITS OF 1 MSEC

CLS
INPUT "ENTER DESIRED DELAY INTERVAL (MILLISECONDS) ", DLY&:SET DELAY INTERVAL

CALL TIMERSET                                'SET 8253 TIMER CHANNEL 0 TO MODE 2
T1& = TIME                                    'INITIAL CALL OF TIME SUBRT SPEEDS SECOND CALL
FOR I = 1 TO 20
    T1& = TIME                                'T1& = TIME AT START OF DELAY SUBRT
    DELAY T1&, DLY&                            'DELAY DESIRED NUMBER OF MSECS FROM T1&
    T2& = TIME                                'T2& = TIME AT END OF DELAY SUBRT
    T# = (T2& - T1&) / 10#                    'T# = MEASURED DELAY INTERVAL IN MSECS
    PRINT USING "#####.##"; T#              'PRINT MEASURED INTERVAL
NEXT I

```

APPENDIX B

Listing of TIMESUBS.BAS

```

'TIMESUBS.BAS    TIMING SUBROUTINES    R. GRAVES, UNIVERSITY OF VICTORIA

DECLARE FUNCTION TIME& ( )

SUB DELAY (STARTTIME&, DELAYDURATION&) 'SUBRT TO WAIT UNTIL INTERVAL ELAPSES
    ENDTIME& = STARTTIME& + (DELAYDURATION& * 10&) - 24& 'SUBTRACT 2.4 MSEC
    WHILE TIME < ENDTIME&: WEND
END SUB

FUNCTION TIME&                                'FUNCTION TO RETURN TIME IN UNITS OF .1 MSEC
DEFINT A-Z
CONST d& = 32768                                'add to tmrread variables to get actual counts
CONST e& = 51
CONST f& = 1000
CONST g& = 119318 'number of .1 msecs per res. count =1000/119318 =.00838095
CONST h& = 35995906 'number of .1 msecs per high count =65536*65536*.00838095
CONST l& = 28012 'number of .1 msecs per low count = 28012/51 = 549.25
    CALL TMRREAD(hi, lo, rs)
    hi& = hi + d&                                ' actual high count
    hi& = hi& * h&                                '.1 msecs in high count
    lo& = lo + d&                                ' actual low count
    lo& = (lo& * l&) \ e&                        '.1 msecs in low count
    rs& = rs + d&                                ' actual residual count
    rs& = (rs& * f&) \ g&                        '.1 msecs in res. count
    TIME& = hi& + lo& + rs&                      'TIME IN UNITS OF .1 MSEC
END FUNCTION

```

APPENDIX C

Listing of SCRNR

```

PAGE
    title SCRNR
    name SCRNR ;module name

Comment |
Developed by:    R. Bradley, University of Victoria, January    1988

Environment:    IBM PC, tested under DOS 3.3
Segmentation:   Program segment CODE, byte aligned
Specifications: This module is intended to be called from Microsoft
                QuickBASIC 4.0.
Implementation:  Called from Microsoft QuickBASIC 4.0.
Returns:        No returns from this module.
Suggested invocation:
    Called from Microsoft QuickBasic 4.0 with:
    SCRNR(#) 'where # is the screen number which is to be switched to.
            'Requires linkage to QuickBASIC program file.

```

APPENDIX C (Continued)

```

CODE SEGMENT BYTE PUBLIC 'CODE'
    ASSUME CS:CODE
    PUBLIC SCRNM

SCRNM PROC FAR
    PUSH BP
    MOV BP,SP
    PUSH AX
    PUSH BX
    PUSH DS
    MOV AH,05
    MOV BX,WORD PTR[BP+6]
    MOV AL,[BX]
    INT 10H
    POP DS
    POP BX
    POP AX
    POP BP
    RET 2
;save registers
;BIOS select-display-page function
;get pointer to screen number
;get screen #
;execute BIOS interrupt
;restore registers
;clean up stack upon return

SCRNM ENDP

CODE ENDS
END

```

APPENDIX D

Listing of TSDEMO.BAS

```

'TSDEMO.BAS
'DEMONSTRATION TACHISTOSCOPE PROGRAM
'USE WITH TSCOPE LIBRARY (TIMESUBS+TIMERSET+TMRREAD+SCRNM)
R. GRAVES, UNIVERSITY OF VICTORIA

DEFINT A-Z
DECLARE FUNCTION TIME& ( )
DECLARE SUB DELAY (STARTTIME&, DELAYDURATION&)
T1& = 0: T2& = 0: T3& = 0
ITI& = 4000
NFRAMES = 1
CONST V% = &H3DA
CONST B% = 8
CONST O% = &H60
CONST P% = 255
CONST Q% = 128
IF INP(0) AND 128 THEN AT = 1 ELSE AT = 0
CONST INFO% = 0, FIXN% = 1, STIM% = 2, POST% = 3
CONST CORR& = 184

'=====
'                               MAIN PROGRAM
'=====
GOSUB LOADSTIMULI
GOSUB INSTRUCTS
GOSUB FIELDSETUP
FOR I = 1 TO NTRIALS
    GOSUB STIMSETUP
    GOSUB DISPLAY
    GOSUB SCORE
NEXT I
END

'=====
LOADSTIMULI:
    NTRIALS = 4
    DIM STIMULUS$(NTRIALS), STIMTYPE(NTRIALS)
    STIMULUS$(1) = "          CAT"
    STIMULUS$(2) = "          DAG"
    STIMULUS$(3) = "          PIG"
    STIMULUS$(4) = "          CUW"
    RETURN

'=====
INSTRUCTS:
    SCREEN 0, , INFO, INFO: WIDTH 80: COLOR 0, 7: CLS
    INPUT "DISPLAY STIMULI FOR HOW MANY FRAMES? ", NFRAMES: CLS
    PRINT "PRESS THE SPACE BAR AS QUICKLY AS POSSIBLE WHEN A WORD APPEARS"
    LOCATE 5, 1: PRINT "PRESS THE SPACE BAR WHEN YOU ARE READY TO BEGIN"
    WHILE INKEY$ = "": WEND: CLS
    CALL TIMERSET

```

APPENDIX D (Continued)

```

T3& = TIME          'GET CURRENT TIME
RETURN
'=====
FIELDSETUP:         'CHANGE TO 40 COL TEXT, BLACK ON WHITE
  SCREEN 0, , INFO, INFO: WIDTH 40: COLOR 0, 7: CLS  'CLEAR INFO FIELD
  SCREEN 0, , FIXN, INFO: COLOR 0, 7: CLS  'FIXATION FIELD ACTIVE
  LOCATE 12, 1, 0: PRINT "                  +
  SCREEN 0, , POST, INFO: COLOR 0, 7: CLS  'POST-STIMULUS FIELD ACTIVE
  LOCATE 12, 1, 0: PRINT "                  "
RETURN
'=====
STIMSETUP:          'PLACE NEW STIMULUS IN STIMULUS FIELD
  SCREEN 0, , STIM, INFO: COLOR 0, 7: CLS  'STIMULUS FIELD ACTIVE
  LOCATE 12, 1, 0: PRINT STIMULUS$(I)
RETURN
'=====
DISPLAY:
  DELAY T3&, ITI& - 1000      'WAIT TIL INTERTRIAL INTERVAL MINUS 1 SEC
  WAIT V, B: CALL SCR(N(FIXN)) 'SWITCH TO FIXATION FIELD
  DELAY T3&, ITI&          'LEAVE FIXATION FIELD ON FOR 1 SEC
                          'WAIT FOR VERT RETRACE, SWITCH TO STIMULUS FIELD, & GET TIME
  WAIT V, B, B: WAIT V, B: CALL SCR(N(STIM)): T1& = TIME
  FOR J = 1 TO NFRAMES      'WAIT FOR NFRAMES TO BE DISPLAYED & GET TIME
    WAIT V, B, B: WAIT V, B: NEXT J: CALL SCR(N(POST)): T2& = TIME
RETURN
'=====
SCORE: 'IF NON-WORD THEN WAIT 1 SEC BEFORE DISPLAY INFO, ELSE GET RT
  IF STIMTYPE(I) = 0 THEN T3& = T1&: DELAY T2&, 1000: GOTO SC1
  IF AT = 1 THEN GOTO AT ELSE GOTO PC
  'WAIT FOR KEYPRESS, GET TIME, THEN CORRECT FOR KEYBOARD DELAY
PC: WAIT 0, P: T3& = TIME: T3& = T3& - CORR&: GOTO SC1 'FOR PC
AT: WAIT 0, Q, Q: T3& = TIME: T3& = T3& - CORR&: GOTO SC1 'FOR AT & PS/2
SC1: SCREEN 0, , INFO, POST: COLOR 0, 7: CLS 'INFO FIELD ACTIVE
  LOCATE 14, 1
  PRINT "DESIRED NUMBER OF FRAMES = "; NFRAMES
  PRINT "      EXPECTED DURATION =";
  PRINT USING "#####."; NFRAMES * (1000 / 60); : PRINT " MSEC."
  PRINT "ACTUAL MEASURED DURATION =";
  PRINT USING "#####."; (T2& - T1&) / 10#; : PRINT " MSEC."
  PRINT "      REACTION TIME =";
  PRINT USING "#####."; (T3& - T1&) / 10#; : PRINT " MSEC."
  CALL SCR(N(INFO))          'DISPLAY INFO FIELD
  IF I = NTRIALS THEN DELAY T2&, ITI&      'DELAY BEFORE ENDING PROGRAM
RETURN

```

(Manuscript received April 1, 1988;
revision accepted for publication May 24, 1988.)