Behavior Research Methods, Instruments, & Computers
1989, 21 (6), 619-622

COMPUTER TECHNOLOGY

TIMEX2: A modified C-language timer for PC AT class machines

MICHAEL M. GRANAAS
University of South Dakota, Vermillion, South Dakota

Emerson (1988) provided a simple C-language timing routine for use with PC AT class machines. Unfortunately, this version of the timing routine makes use of nonstandard functions that are not available in all C-language packages. A modified version of this timing routine that does not make use of one of these nonstandard functions is provided. This version of TIMEX runs under the widely available Microsoft C 5.1. The need and availability of the remaining four nonstandard C functions is discussed. Constants needed to convert the timing routine units into seconds and milliseconds are also provided.

Having recently begun a program of research using PC AT computers for stimulus display and timing, I was rather alarmed to find that the system clock only "ticked" every 54.9 msec. How could a user possibly expect to measure stimulus duration and reaction times precisely? The answer was found in the C-language routine TIMEX, written by Emerson (1988).

Emerson (1988) has provided a badly needed piece of programming for those wishing to control or measure time critical elements on a PC AT class computer. The TIMEX function is written in the C programming language, and it increases the resolution of timing on an AT class machine from approximately 55 msec to approximately .054 msec. This is done by making use of the system's 8254 hardware interrupt timer, which is typically not available to the programmer. Emerson refers to this .054-msec time unit as the "tock." The same convention will be followed here.

Emerson's (1988) routine is limited in that it uses nonstandard C-language functions (Kernighan & Ritchie, 1988) that may not always be available to the programmer. The functions include: enable(), disable(), peek(), inp(), and outp(). If some or all of these functions are missing from the C-language compiler, TIMEX becomes useless. Thus, the desire is to replace any or all of these functions with standard C programming instructions, so that TIMEX can become useful for a wider audience of researchers.

This work was supported in part by Grant NAG2-492 from NASA's Ames-Dryden Flight Research Facility. The author would like to thank Bob Wood of the USD Computer Science Department for his helpful comments on this project. I would also like to thank P. L. Emerson for making available what I consider to be one of the most valuable tools available for psychological research on a PC: a working millisecond timer. In addition, my thanks to an anonymous reviewer whose comments have improved the readability of this paper. The author's address is: Department of Psychology, The University of South Dakota, 414 E. Clark St., Vermillion, SD 57069.

The peek() Function

The only one of the needed functions not available in Microsoft C 5.1 is the peek() function (Microsoft Corp., 1987), which allows the program to directly read a specified location in memory. Such access to memory is a critical feature of TIMEX. This being the case, an available alternative to the peek() function would make the TIMEX routine available to a wider audience of researchers than is currently possible.

Fortunately, the C language provides direct memory addressing as part of the language, through the use of pointers (Kernighan & Ritchie, 1988). The use of pointers is typically restricted to memory locations within the program's data segment (Miller & Quilici, 1986). It is, however, still possible to address and read memory outside of the program's data segment, through the use of far pointers (Miller & Quilici, 1986).

To accomplish the reading of the memory locations containing the clock information needed by TIMEX2, the segment and offset address form used by peek() needs to be converted into an absolute address, which can then be cast into a far pointer. This is done by multiplying the segment portion of the address by 16 and adding the offset to the result (King, 1983). The result is then cast into the far pointer type (Miller & Quilici, 1986). This is done as the first two lines of code within TIMEX2 (see Listing 1).

To mimic the peek() function, the program merely reads the contents of the memory location pointed to by the pointer. This value is assigned to a variable for later use (see Listing 1).

Even when the peek() function is available, the programmer may want to consider using pointers instead. Reading a memory location through the use of pointers

Table 1
Tock Frequency Distribution for 10,000 Trials

Number of Tocks	Frequency
1	310
2	9,304
3	380
4	6

Note—Mean = .108 msec, SD = .014.

is somewhat faster than a function call (Miller & Quilici, 1986).

Replacing the call to peek() with direct memory access is the only coding difference TIMEX2 (Listing 1) and TIMEX (Emerson, 1988). In terms of performance, TIMEX2 running 10,000 trials on a 12-MHz IBM AT clone had a modal execution time of two tocks, and range of three tocks (see Table 1 for a complete summary of these data). So, in the worst case, TIMEX2 is still accurate to plus or minus four tocks ($\pm$0.215 msec), which should be sufficiently accurate for most psychological measurement needs.

The enable() and disable() Functions

The enable() and disable() functions are available in Microsoft C 5.1 as _enable() and _disable() (Microsoft Corp., 1987), and are used in TIMEX2. However, these functions are not available in Lattice C 2.15 (Lattice Inc., 1985).[1] Emerson (1988) cautions against running without these two functions. An empirical investigation using TIMEX2 in Microsoft C 5.1 with the _enable() and _disable() functions removed suggests that Emerson is correct in his warning. Approximately 5 to 10 times in a given set of 10,000 elapsed time trials, where elapsed time is measured between two successive calls to TIMEX2, the elapsed time would be approximately 1,024 tocks. This compares to the typical value of 2 tocks and a range of 1 to 4 tocks (see Table 1) with the _enable() and _disable() functions included.

The inp() and outp() Functions

The inp() and outp() functions are both available in Microsoft C 5.1 (Microsoft Corp., 1987) and Lattice C 2.15 (Lattice Inc., 1985), along with Turbo C (Emerson, 1988). It is arguable that these are widely available functions, at least in major C-language compilers for PC class machines, but this is probably of little concern. In the event that these, or some similar, functions are unavailable, it should be possible to mimic their behavior through the use of pointers.

Conversion of Tocks

The unit of measure provided by TIMEX and TIMEX2, the "tock" (Emerson, 1988), is approximately .054 msec long. Dividing the time in tocks by 18,643.469 yields time in seconds. Dividing by 18.643469 gives the time in milliseconds. The program in Listing 2 has been modified to print times in milliseconds.

Conclusion

The TIMEX timing function (Emerson, 1988) is a generally useful way to measure millisecond times using the C programming language. The modification implemented in TIMEX2 makes TIMEX run under Microsoft C 5.1 (Microsoft Corp., 1987), and therefore makes it available to a larger audience of potential users.

REFERENCES

Emerson, P. L. (1988). TIMEX: A simple IBM AT C language timer with extended resolution. *Behavior Research Methods, Instruments, & Computers*, **20**, 566-572.

Kernighan, B. W., & Ritchie, D. M. (1988). *The C programming language* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

King, R. A. (1983). *The IBM PC-DOS handbook*. Berkeley, CA: SYBEX.

Lattice, Inc. (1985). *Lattice C compiler*. Glen Ellyn, IL: Author.

Microsoft Corp. (1987). *Microsoft C 5.1 optimizing compiler: Runtime library reference*. Bellevue, WA: Author.

Miller, L. H., & Quilici, A. E. (1986). *Programming in C*. New York: Wiley.

NOTE

1. Lattice C is now in version 6.XX, so these functions may now be available in the package.

LISTING 1
The TIMEX2 Function

```
/*  the function TIMEX was originally written by Phillip L.
 *  Emerson of Cleveland State University, and was published in
 *  "Behavior Research Methods, Instruments, & Computers", vol 20
 *  (6), 1988, pgs 566-572.
 *
 *  the current version was modified by Michael M. Granaas of the
 *  University of South Dakota in May of 1989.  The modifications
 *  allow timex to be compiled and run under Microsoft C 5.1.
 */

#include <dos.h>
#define TIMERO 0x40
#define CNTRL 0x43
#define READ_BACK 0xc2
#define BIOSSEG 0x40
#define TICS_LO 0x6c
#define TICS_HI 0x6e
```

LISTING 1 (Continued)

```
void_enable(void);
void_disable(void);

  long timex()
    {
      unsigned tmp1, tmp2, tmp3, tmp4, hi, lo;
      int status;
      unsigned far *add1;
      unsigned far *add2;
      long ticks1, ticks2;

/*  add1 and add2 are pointers that point to the memory location
 *  of the system clock after converting from segment:offset
 *  format to absolute address
 */
      add1 = (unsigned far *) (BIOSSEG * 16 + TICS_LO);
      add2 = (unsigned far *) (BIOSSEG * 16 + TICS_HI);

      _disable();
      outp(CNTRL,READ_BACK);
      status = inp(TIMER0);
      lo = (unsigned) inp(TIMER0);
      hi = (unsigned) inp(TIMER0);

/*  the following two lines of code read the values at the
 *  specified memory addresses.  These replace the peek()
 *  function.
 */
      tmp1 = (unsigned) *add1;
      tmp2 = (unsigned) *add2;
      _enable();

      hi = hi << 2 | lo >> 6;

      _disable();
      tmp3 = (unsigned) *add1;
      tmp4 = (unsigned) *add2;
      _enable();

      if( !(lo & 63) ) hi-- ;
      hi = (~hi & 1023) >> 1;
      if( !(status & 128) ) hi |= 512;

      ticks1 = (long) tmp2 << 16 | tmp1;
      ticks2 = (long) tmp4 << 16 | tmp3;

      if(ticks2 > ticks1 + 1) return(-1);

      if(hi & 512)
       return(ticks1 << 10 | hi);
      else
       return(ticks2 << 10 | hi);
```

LISTING 2
An Example Timing Program Calling TIMEX2

```
/*  This program is virtually identical to the one originally
 *  written by P. L. Emerson (1988).  It reads the timer twice
 *  and calculates the elapsed time.  The two modifications
 *  to this program is that it uses TIMEX2, and converts tocks
 *  into milliseconds before printing the elapsed time
 */
#include <stdio.h>
#include "timex2.c"
long timex();
```

LISTING 2 (Continued)

```
main()
  {
    unsigned long ztime, etime;
    float elapsed;
    while (1)
     {
       printf("press ENTER to start timing\n");
       getchar();
       ztime = timex();
       printf("press ENTER to read timer\n");
       getchar();
       etime = timex();
       if (ztime < 0 || etime < 0) break;
       etime -= ztime;

/*  dividing etime by 18643.469 gives elapsed time in seconds */
/*  dividing etime by 18.643469 gives elapsed time in msec    */

       elapsed = etime/18.643469;
       printf("time was %f milliseconds \n", elapsed);
     }
       printf ("timing error\n");
     }
```

(Manuscript received August 28, 1989;
revision accepted for publication October 16, 1989.)