PROGRAM ABSTRACTS/ALGORITHMS

A software-based millisecond timer for the CP/M operating system

ANDREW C. COYNE

Ohio State University, Columbus, Ohio

One of the most popular operating systems for 8-bit microcomputers is Digital Research's CP/M (see Waite & Lafore, 1983, for an extensive discussion of CP/M). By requiring that programmers and programs make use of a standard set of Basic Disk Operating System (BDOS) calls to handle input, output, and other tasks, CP/M allows for a high degree of transportability across microcomputers based on Z-80, 8080, and 8085 microprocessors. This transportability represents a substantial advantage for university-based researchers who may find themselves dealing with a variety of microcomputers based on the above chips, yet all using slightly different input, output, and mass-storage schemes.

The present article details a simple and short (39-byte) assembly language millisecond timing program that utilizes a call to CP/M's BDOS to monitor a microcomputer's keyboard for a subject's response. Dlhopsky (1983) has provided one means of carrying out such a timing task with the Z-80 assembly language routine, MSEC LATENCY TIMER. However, this particular timer is designed for and will run only on a TRS-80 microcomputer, since it makes use of specific keyboard routines that are coded into the computer's read only memory (ROM). The current routine, CPMTIMER, represents a version of Dlhopsky's MSEC LATENCY TIMER that has been modified to run under CP/M (which employs 8080 code). An additional modification has been made so that a subject's keyboard responses are monitored through the use of a CP/M BDOS call that is constant across all microcomputers running the CP/M operating system. As noted above, this greatly increases the transportability of the program.

Description. CPMTIMER was designed to keep track of how many milliseconds (up to 65,535) elapse between the time that the routine is called or activated from another program and the time that a subject pushes a keyboard key. The routine also keeps track of which key is pressed. CPMTIMER carries out these tasks by continually monitoring the keyboard for a subject's input. If no key is pressed, a series of microprocessor instructions are executed which keep track of elapsed time and increment a counter when 1 msec passes. If a key is pressed, the

hexadecimal value associated with that key is saved, as is the millisecond count, and either the routine terminates or control is returned from CPMTIMER to the program that initially called it.

Monitoring of the keyboard in CPMTIMER is accomplished through the use of CP/M's BDOS system call number 6: Direct Console Input/Output. This system call checks the keyboard (which along with the video display forms what CP/M refers to as the *console*) and returns a hexadecimal value in the microprocessor's A-register. A value of zero in the register indicates that no key has been pushed, whereas a nonzero value indicates that a key has been pushed. In the latter case, the value corresponds to the ASCII code (in hexadecimal notation) of the particular key pushed. This process is handled without echoing the keypress to the screen, thus avoiding any disruption of information previously presented on the video display.

The actual operation of CPMTIMER is straightforward and parallels that of Dlhopsky's (1983) MSEC LATENCY TIMER. Each millisecond of elapsed time is computed on the basis of the total length of time the microcomputer takes to execute the various commands making up the assembly language routine, in addition to the time required to check for keyboard input via the CP/M system call. The exact length of time (in microseconds) required by the routine can be determined by adding the number of microprocessor clock cycles (also referred to as *T-states*) required to execute each individual instruction constituting CPMTIMER as well as each one constituting the CP/M system call utilized and then dividing by the microcomputer's system clock frequency (in MHz). Note that a listing of the number of clock cycles required to execute specific 8080 instructions can be found in Leventhal (1978), and the system clock frequency for a particular microcomputer is typically listed in the unit's technical reference manual. Also, see Dlhopsky (1983) for an extended discussion of how to compute execution times of assembly language routines.

The present version of CPMTIMER was developed and calibrated on the author's Osborne Executive microcomputer, which is based on a Z-80A microprocessor running at 4 MHz. Depending upon the particular microcomputer utilized and the system clock frequency of its microprocessor, CPMTIMER may have to be recalibrated by increasing or decreasing the number of time delay loops the routine carries out, as indicated in the source code (for availability, see below). The accuracy of CPMTIMER was empirically tested by comparing its results against an external millisecond timer (Hunter Klockounter Model 120C), which was started and stopped by the same signal as that which controlled CPMTIMER.

The author's mailing address is: Department of Psychology, Ohio State University, 142 Townshend Hall, 1885 Neil Avenue Mail, Columbus, OH 43210-1222.

This test yielded an accuracy level of 0.2412% over an extended series of trials.

In conclusion, the assembly language routine CPMTIMER provides users of CP/M with millisecond timing capability in a format that can be easily transported across a variety of microcomputers running this popular operating system. The routine can be utilized within other assembly language programs to allow for inexpensive implementation of stimulus and response timing within the laboratory.

Availability. A fully documented listing of the 8080 assembly language source code for CPMTIMER may be obtained without charge from the author. The source code may be assembled into executable object code via the CP/M 2.2 utilities ASM.COM and LOAD.COM (or CP/M 3.0's MAC.COM and HEXCOM.COM) provided

with most CP/M systems. A copy of the source code and object code will also be provided on disk (Osborne single- or double-density format) if a blank 5.25-in. soft-sectored floppy disk, protective mailer, and sufficient return postage are included with a request.

REFERENCES

- DLHOPOLSKY, J. G. (1983). Machine language millisecond timers for the Z-80 microprocessor. *Behavior Research Methods & Instrumentation*, **15**, 511-520.
- LEVENTHAL, L.A. (1978). *8080A/8085 assembly language programming*. Berkeley, CA: Osborne/McGraw-Hill.
- WAITE, M., & LAFORE, R. (1983). *Soul of CP/M: How to use the hidden power of your CP/M system*. Indianapolis: Howard W. Sams.

(Revision accepted for publication September 19, 1986.)