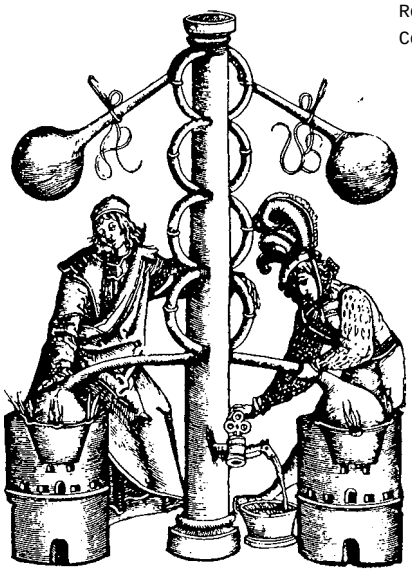

Microcomputer interfacing

Real-time clocks

AN INTRODUCTION

By Peter R. Rony, Jonathan A. Titus, Christopher A. Titus, and David G. Larsen

IN MANY microcomputer applications it is necessary to have the computer perform actions at accurately timed intervals. This allows the computer to make accurate measurements of an analog signal at 100-msec intervals, for example. The period of 100 msec may be timed through the use of a time-delay loop in which software commands are used, or through the use of an external clock.

At this point, it must be realized that while a time-delay software routine may generate a delay of the required accuracy, the computer can do nothing else while it is performing the timing software steps. This is a serious limitation. Although probably less obvious, the time-delay software steps may be interrupted by an external device that requires immediate servicing by the computer. The overall effect is to lengthen the time required for the time-delay software steps. The actual time delay would be the sum of the time spent in the software steps and the time spent servicing the external interrupt.

In most instances in which accurate periods are required, an external circuit is used to time the necessary periods with as little interaction between the computer and the external clock as possible. Such clocks are immune to external interrupts and to changes in the normal flow of a program. Once started, they continue to time a period until it is completed and the time is up. In this way, the clock runs parallel with the computer, allowing the computer to perform other tasks and service interrupts while the clock is running. This type of an external clock is often called a *real-time clock,* since its time is real, in that it cannot be altered or delayed by events that would normally affect a program.

There are several different types of real-time clocks:

1. *Programmable real-time clock.* The actual period that is required is preprogrammed within the clock through either hardware or software. Once the clock has been started it will continue timing until the period has ended. At the end of the period, the clock will signal the computer that the timing task has been completed.

2. Free-running real-time clock. The clock runs continuously, signalling the computer at the end of each period. The periods are generally of equal length, e.g., 10 msec.

3. Time-of-day clock. This type of clock will provide the computer with the actual time, for example, 16:20 hours. This type of clock is not used frequently in small computer systems.

Although you may not have realized it, you have already been exposed to the concept of real-time in previous columns. We described the operation of an 8085-based computer system in previous columns, and the use of the 14-bit timer contained within the 8155 read/write memory and interface chip was described in terms of its real-time operation. In that 8085-based computer application, the 14-bit counter obtained its time-base from the crystal clock that was used to control the 8085 chip. An interrupt was used to signal the end of the timing period.

The 14-bit timer was an excellent example of a programmable real-time clock. If it is assumed that a frequency of 1 MHz was used to control the clock, a 14-bit counter could provide a total count of 16,384 μsec, or just over 16 msec. This might be somewhat limiting if periods of several seconds are required, but the scheme is fairly flexible. If longer periods are required, the 14-bit counter could be programmed to time an integer fraction of the period and the computer used to total the number of shorter periods that

Dr. Rony, Department of Chemical Engineering, and Mr. Larsen, Department of Chemistry, are with the Virginia Polytechnic Institute and State University. Mr. J. Titus is *President, and Dr.* C. *Titus* is *Consultant, Tychon, Inc.*

are required for the total period to elapse. The computer would only have to increment and test a count each time that the clock interrupted it. One drawback is that additional software is required, something that we tried to avoid through the use of the real-time clocks initially, but the additional program steps are minimal.

In many cases, it would be valuable if the real-time clock could be preset for the clock's basic frequency, i.e., I MHz, 10 KHz, etc., as well as for the actual count. If these various intervals were available, the timing of longer periods would be relatively easy and no additional software steps would be required. A simple series of divide-by-ten counters such as the SN7490 or SN74390 could be used to divide a high-frequency clock signal into lower frequencies for use by the real-time clock's counters. Various frequencies could be selected readily through the use of jumper wires on the computer board. A more sophisticated real-time clock could use an electronic switching circuit that would allow the computer itself to select the frequency required. Thus, a programmer could select the basic period and the actual count through software commands to the real-time clock.

The free-running real-time clocks are preset to time a period of predetermined length, such as 10 msec. This period is timed over and over again, interrupting the computer each time that a period has been completed. In many computer systems, the line power frequency, 60 or 50 Hz, is used to provide a very stable, fixed-length period that may be used with equal success. The free-running type of real-time clock is not as independent of the computer, e.g., software control, as the programmable real-time clock. Software steps to accumulate the number of periods are still required, and total timing period may have an error of up to two of the basic frequency periods.

Since the free-running real-time clocks have a regular period, they are often used to signal the processor that it is time to start a software routine that will check various input/output (I/O) devices to determine whether or not they require some computer service. Through the use of a software table, the computer can check to see what devices are able or disabled and it can also determine the frequency at which they must be checked. It is useless to check a 1O-character-per-second teletypewriter every 10 msec, so it is only checked every 80 or 90 msec instead, while a faster device is checked at the end of each lO-msec period. Such a scheme allows the computer, and the programmer, to have a great deal of flexibility in the way that real-time operations are handled, particularly in situations in which the computer is required to perform many real-time operations simultaneously.

In nearly all cases, the computer and real-time clock are connected by an interrupt signal. In this way, the clock can immediately signal the computer that the current period has been completed or timed out. Because interrupts can be quite complex, as we have described previously in our columns, only one real-time clock should be used with a microcomputer. It will be up to the user to determine the priority and thus the importance of the real-time clock. Often the real-time clock is assigned to the highest priority.

Unfortunately, there are few, if any, real-time clock interfaces available for the Intel/National Semiconductor 8080-based microcomputer boards, or for the popular S-IOO-type microcomputer systems. In a future column we will describe further the hardware interface for a programmable real-time clock. The software will also be.listed and discussed.