# computer corner

## *Data acquisition for microcomputers.* PETER R. RONY, JONATHAN A. TITUS, DAVID G. LARSEN, CHRIS A. TITUS*

THE SOFTWARE IN A PREVIOUS COLUMN provided an example of a program used to acquire a single analog point in digital form. (See October 1977 issue.) We are generally interested in applications in which a series of points are to be acquired, stored, displayed and perhaps manipulated. This month we will explore the use of microcomputers for data acquisition.

It will be assumed that the analog-to-digital converter (A/D) is interfaced as shown in the previous column. The software routine to run the A/D converter is shown in Table 1. The digital value of the analog voltage is returned in the B and C register (register pair B).

In most data acquisition programs, a fixed number of points must be acquired over a fixed period of time. In our example, 100 points will be taken, one every second. The 100 data points will be stored in read/write memory so that they can be used later. In writing data acquisition software we are now faced with three tasks that must be performed in addition to the actual A/D task:

1. **Provide a software counter** to count 100 points.
2. **Provide a one-second timer.**
3. **Provide software** to store the data values.

### Software counter

The software necessary to count the 100 acquired points will actually count 100 passes through the data acquisition software. A general-purpose register within the 8080 IC is well suited for this; conditional jump instructions can be used to detect when the count is decremented to zero. The counter can be either incremented or decremented, but decrementing is probably easiest if you are just starting to program microcomputers.

Storing the data in memory is not difficult. Once the converter value is stored in a register pair, the H and L registers (register pair H) can be used as memory pointers to point to a read/write memory location. Note that a complete 16-bit address must be specified for the MOV M,r instructions. Since the data is acquired from a 10-bit A/D converter, two successive memory locations must be used to store each *point*. The INXH instruction (increment register pair H) provides an easy means of pointing to the next successive memory location. We will store the data by placing the eight least significant bits in location n, and the two most significant bits in location n + 1.

*This article is reprinted courtesy American Laboratories. Dr. Rony, Department of Chemical Engineering, and Mr. Larsen, Department of Chemistry, are with the Virginia Polytechnic Institute & State University. Mr. Titus is president of Tychon, Inc.

```
              /           TABLE 1    TYPICAL ADC INPUT ROUTINE FOR A
              /           10 BIT ANALOG-TO-DIGITAL CONVERTER

                          *100 000
100 000 365  ADC,     PUSHPSW /SAVE REGISTER A & FLAGS
100 001 323           OUT     /STROBE THE ADC TO START A CONVERSION
100 002 037           037
100 003 333  TEST,    IN      /INPUT STATUS BIT AND 2 MSB'S
100 004 066           066
100 005 306           ADI     /ADD 1 TO THE FLAG BIT
100 006 200           200     /TO CAUSE A CARRY IF IT IS SET
100 007 322           JNC     /NO OVERFLOW, CHECK IT AGAIN
100 010 003           TEST
100 011 100           0
100 012 107           MOVBA   /OVERFLOW, FLAG=1, SO SAVE MSB'S
100 013 333           IN      /INPUT THE 8 LSB'S
100 014 065           065
100 015 117           MOVCA   /STORE THEM IN REGISTER C
100 016 361           POPPSW  /RESTORE REGISTER A & FLAGS
100 017 311           RET     RETURN TO MAIN PROGRAM
                       /        TABLE 2  100 POINT DATA ACQUISITION
                       /        ROUTINE FOR ONE POINT PER SECOND

                        DW ADC 100 000

                        *070 000
070 000 061  START,   LXISP   /LOAD THE STACK POINTER
070 001 377           377
070 002 070           070
070 003 041           LXIH    /LOAD THE DATA STORAGE STARTING
070 004 000           000     /ADDRESS IN REGISTERS H & L
070 005 072           072
070 006 315  CONVRT,  CALL    /CALL THE ADC SOFTWARE
070 007 000           ADC     /SHOWN IN TABLE 1
070 010 100           0
070 011 161           MOVMC   /STORE THE 8 LSB'S TO MEMORY
070 012 043           INXH    /INCREMENT THE MEMORY POINTER
070 013 160           MOVMB   /STORE THE 2 MSB'S TO MEMORY
070 014 043           INXH    /INCREMENT THE POINTER AGAIN
070 015 175           MOVAL   /GET THE LOW ADDRESS VALUE
070 016 376           CPI     /COMPARE IT TO THE 201ST ADDRESS
070 017 310           310     /310 = 200 DECIMAL
070 020 312           JZ      /DONE YET?
070 021 047           DONE    /YES, JUMP TO "DONE"
070 022 070           0
070 023 315           CALL    /NO, DO THE 1 SECOND DELAY
070 024 031           DELAY
070 025 070           0
070 026 303           JMP     /AFTER THE DELAY, GET THE NEXT
070 027 006           CONVRT  /ADC DATA POINT
070 030 070           0

                        /THIS IS THE ONE SECOND TIME DELAY
                        /SUBROUTINE

070 031 365  DELAY,   PUSHPSW /SAVE REG A & FLAGS
070 032 325           PUSHD   /SAVE REGISTERS D & E
070 033 021           LXID    /LOAD COUNTER REGISTERS
070 034 000           000
070 035 110           110
070 036 033  DEC,     DCXD    /DECREMENT THE REG PAIR
070 037 172           MOVAD
070 040 263           ORAE
070 041 302           JNZ     /IF NOT ZERO, DO IT AGAIN
070 042 036           DEC
070 043 070           0
070 044 321           POPD
070 045 361           POPPSW
070 046 311           RET

                        /THE PROGRAM WILL CAUSE THE COMPUTER TO
                        /JUMP HERE WHEN IT HAS ACQUIRES ALL THE
                        /DATA POINTS.  A DISPLAY OR OTHER ROUTINE
                        /MIGHT BE PLACED HERE INSTEAD OF THE HALT

070 047 166  DONE,    HLT
```

### Real-time clock

The one-second timer may present some problems, depending upon the type of system used. It is relatively easy to write a one-second software delay program using a series of register decrementing loops, nested one within the other. However, to accurately time a one-second period, the computer must be doing nothing else. In a system dedicated to data acquisition for the 100-second period, such a procedure is valid.

If interrupts occur or if the computer cannot be allowed to "do nothing" most of the time, an alternate solution is needed. One possibility is to use an external clock, often called a *real-time clock*.

Real-time clocks are unaffected by computer execution times, interrupts, slow I/O devices, etc. Once started, they will continue to run at an accurate rate until they have timed the particular period of interest and sent an interrupt to the microcomputer.

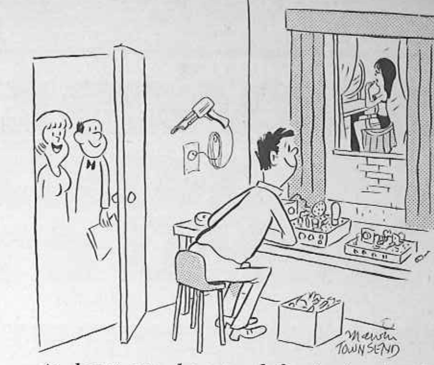
Some clocks are free running, always keeping time; others are programmable or preset for a particular period. The free-running clock interrupts the computer at repetitive intervals while the programmable clock interrupts the computer only once, at the end of its pre-programmed period. Integrated circuits such as the Intel 8253 and Texas Instruments TMS 5501 contain time-keeping circuitry that is easily interfaced to most 8080's.

### Data storage

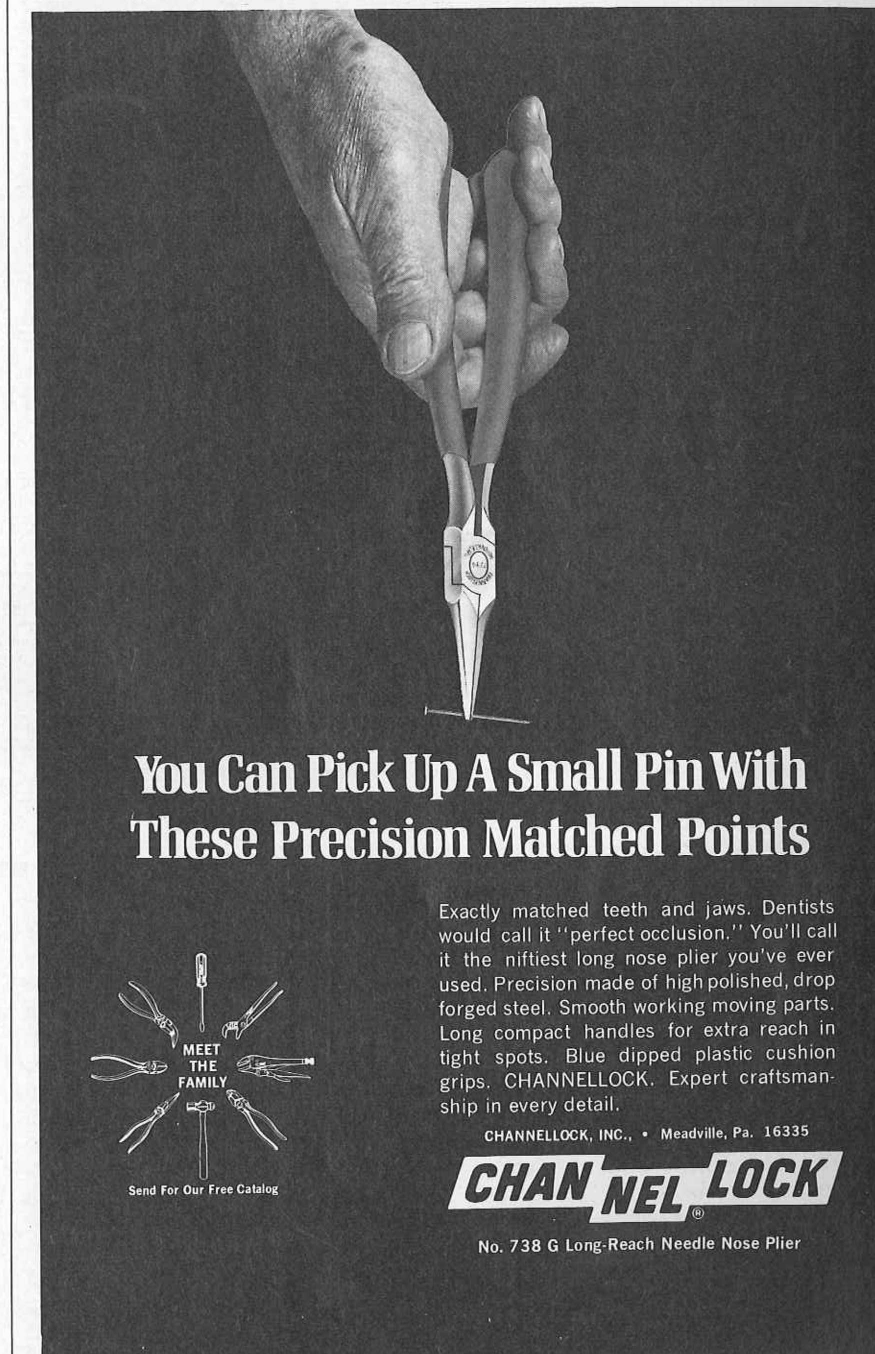
For simplicity, we will use the software clock in our example rather than an interrupt-based real-time clock. The software for the 100-point data acquisition program is shown in Table 2. After completing the program, the computer might be programmed to jump to the type of data display software discussed in the previous column.

If you look at the program carefully, you will not find a separate register to count the 100 passes through the data acquisition software. Since the memory address stored in registers H and L is already a counter, we have chosen to detect the 200th address rather than the 100th loop. This saves an internal register. Instead of decrementing a counter and detecting the zero condition, the contents of register L are compared with the final address and the equality is used to signal the end of the loop.

Analog-to-digital converters are not "instantaneous" devices that take only a few microseconds to perform a conversion. In many real situations, the analog input to the converter will vary while the A/D converter is trying to perform a conversion. This presents the converter with a problem. How does it know what the real value of the voltage is? In most systems the A/D converter module has a *sample-and-hold* device on the analog input. The sample-and-hold circuitry samples the analog voltage when pulsed to provide a steady analog output to the A/D converter for conversion; the A/D converter is then pulsed to start the conversion. The Intersil IH 5110 is a typical sample-and-hold device. **R-E**



*At least we know while junior is tinkering with his electronic equipment his mind is off girls!*

CIRCLE 4 ON FREE INFORMATION CARD