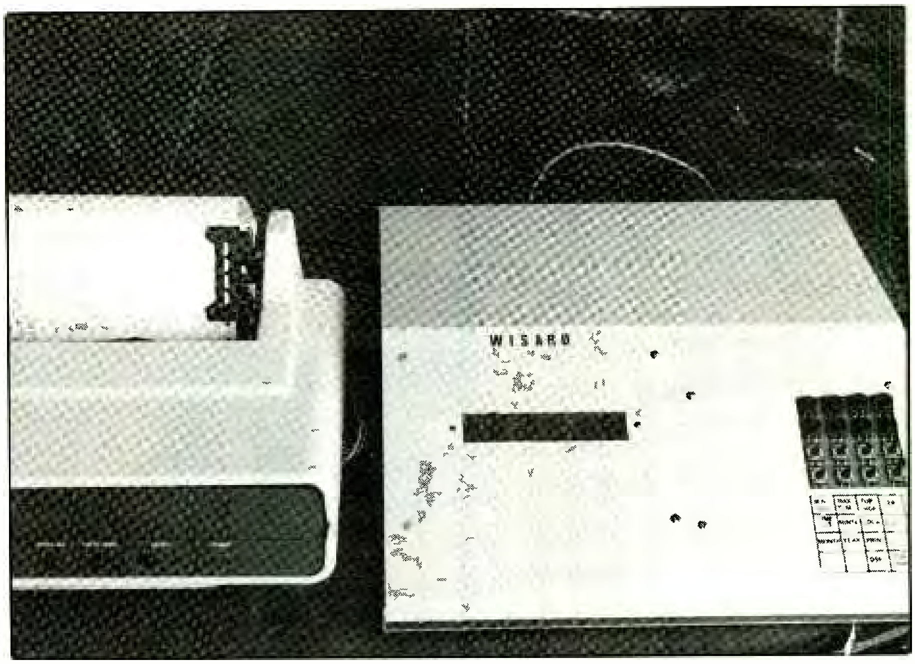
A Smart Weather Monitor

(Conclusion)

How to use and variations on the basic project

By Thomas R. Fox

In the first four installments of this article, we described construction and initial testing of the WISARD, a stand-alone expandable multi-function "smart" thermometer and temperature predictor. In this final installment, we discuss instrument operation and give hints on modifications you can make to expand it. We will conclude with a brief look at a down-to-earth variation on the basic WISARD project.



Sensor Placement

As has previously been stated, the basic WISARD has five sensors, one each to detect rain/dew, daylight and sunshine and two to track temperature variations.

For the project to accurately sense the presence of moisture, the rain/dew sensor must be mounted in a location where rain will strike it directly. An area of the roof of your house that has no tree branches or other structures that obstruct a clear line of sight overhead is one suitable location for the rain/dew sensor.

Sunshine and daylight sensors *Q1* and *Q2* in the schematic diagram in Part 1 are simple phototransistors. One method of protecting these sensors from the weather was described in Part 4. Daylight sensor *Q2* must be mounted outside, such as on a roof. A nearly perfect location for sunshine sensor *Q1* is on a 1,000-foot tower, well above surrounding trees and buildings. This is obviously impractical in real-world situations. Hence, the site that gives perhaps the largest accuracy/cost ratio is the highest point of the roof of the structure in which the WISARD project is located.

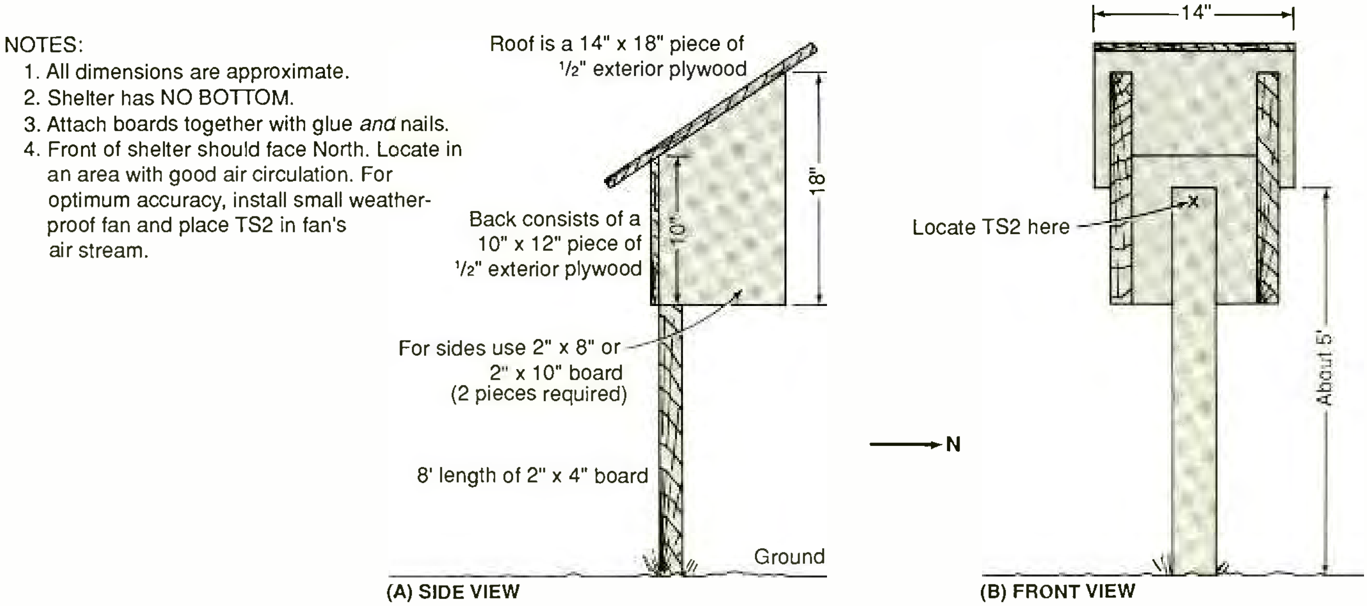
The firmware for the project treats temperature sensor *TS1* as the "inside" monitoring device and temperature sensor *TS2* "outside" monitoring device. Most of the records and temperature analysis pertain to outside sensor *TS2*.

Assuming you are using standard ME1 firmware, *TS1* should be located inside the structure, close to where the WISARD is located. It may be of interest to you to monitor the internal temperature of the project. If so, you can mount *TS1* directly to the A/D Memory-Expansion module in the holes assigned for this sensor.

The purpose of *TS2* is to measure outside air temperature. Therefore, locate this sensor in a suitable thermometer shelter, such as illustrated in Fig. 18. Mount this shelter a minimum of 6 feet from any building or other structure that is heated during the winter and cooled during the summer.

Operating WISARD

Once the project has passed its initial tests and you have mounted the various sensors in suitable locations, it is time to put WISARD on-line. Unlike most electronic devices, WISARD should be allowed to undergo a full


Roof is a 14" x 18" piece of ½" exterior plywood
18"
Back consists of a 10" x 12" piece of ½" exterior plywood
10"
For sides use 2" x 8" or 2" x 10" board (2 pieces required)
8' length of 2" x 4" board
Ground
(A) SIDE VIEW
N
14"
Locate TS2 here
X
About 5'
(B) FRONT VIEW

NOTES:
1. All dimensions are approximate.
2. Shelter has NO BOTTOM.
3. Attach boards together with glue *and* nails.
4. Front of shelter should face North. Locate in an area with good air circulation. For optimum accuracy, install small weather-proof fan and place TS2 in fan's air stream.

Fig. 18. Details for building a shelter for the outside-temperature sensor.

24-hour period of initialization. Self-taught tutoring is required inasmuch as the firmware that predicts minimum temperature requires the full 24-hour period that comprises a day, as well as sunrise and sunset times. Other reasons for the required "schooling time" are that WISARD requires mean and hourly temperature data before it can "intelligently" predict temperatures.

After setting the project in its permanent location, it is best to start it up in the morning, after dawn. A nighttime start-up may cause the project to act erratically at first. However, if you do start the project operating at night, do not be concerned by the various error messages and/or sluggish operation that result.

If you have a printer connected to it when WISARD is first turned on, a warning message will be printed that informs you that power has been off, the clock/calendar requires setting and data may be in error. This warning message is printed every time power is interrupted and then restored *and* whenever the battery has discharged.

In Part 4, we described how to set the clock. The DISPLAY key on the keypad is used to flash up the various data in the LED display. Pressing this switch once causes a "d" to flash intermittently in position 5 of the display. This "d" indicates that the minimum and maximum temperatures, day, month and year can be accessed by pressing the appropriate keypad keys.

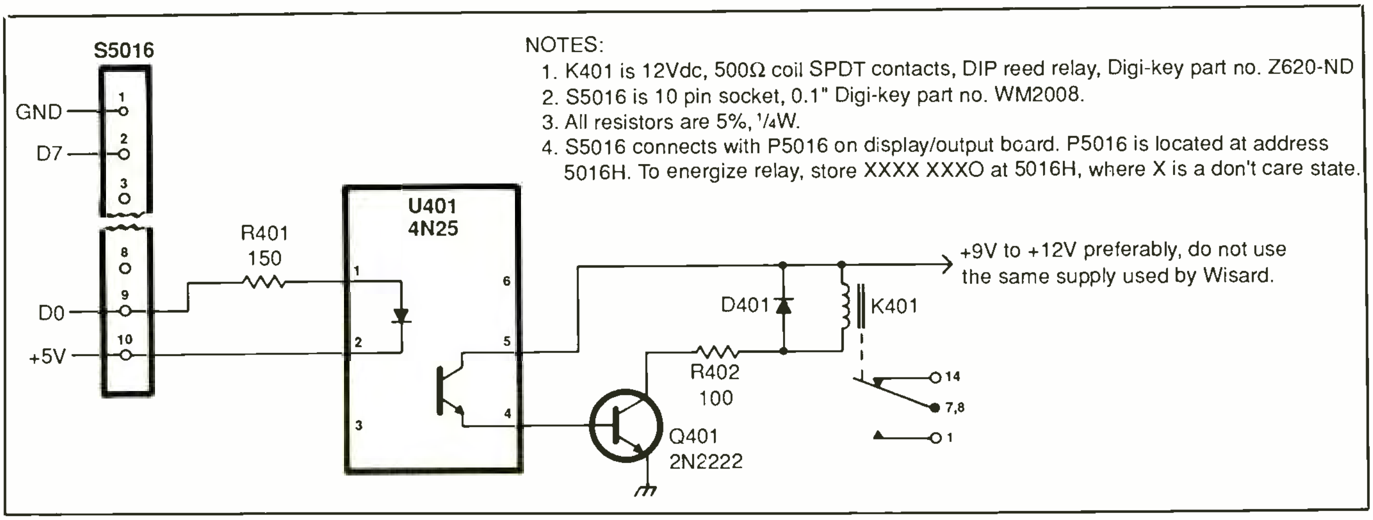
Pressing the DISPLAY key on the keypad once again causes the "d" to disappear from the display. Now if one of the keys previously mentioned is pressed, the display will show "ErrOr.'

WISARD has built-in pagination that assumes a 66-line page. At the top of each new page, the project prints out the date and time it started printing that page. This information permits simple organization of data. At the beginning of the first page, press the TOP PAGE, key. Thereafter, WISARD will automatically keep track of the other pages for you.

Pressing the FORECAST key causes WISARD to print out the high temperature forecast for the day and the low temperature for the night. Notice that if this key is pressed while daylight conditions exist, the forecast minimum temperature will be "???." This simply means that WISARD does not have enough data to forecast a temperature yet. It waits until the daylight sensor is clothed in darkness. Also, WISARD waits until 6:00 AM before it estimates how warm the temperature will be for any given day.

Pressing the PRINT key forces a printout of various information, including the hourly temperature readings for the day. The purpose of the VERSION key is to provide a printout of the version of the firmware in use.

As has previously been mentioned, if a printer is connected to WISARD and is on at 9:00 AM, the project automatically prints out temperature



NOTES:
1. K401 is 12Vdc, 500Ω coil SPDT contacts, DIP reed relay, Digi-key part no. Z620-ND
2. S5016 is 10 pin socket, 0.1" Digi-key part no. WM2008.
3. All resistors are 5%, ¼W.
4. S5016 connects with P5016 on display/output board. P5016 is located at address 5016H. To energize relay, store XXXX XXXO at 5016H, where X is a don't care state.

Fig. 19. Schematic diagram of a tested control circuit for loads that require more than 25 milliamperes of current.

and other information from the previous day. If for some reason the power or printer is off at 9:00 AM, printout will occur at 10:00 AM. If you do not desire a printout, you can either turn off the printer or disconnect it from the project.

ME1 firmware instructs WISARD to printout a rain/dew message when rain first hits the moisture sensor and a "dried-off" message when moisture has evaporated from the sensor. Normally, this does not cause a problem. However, on a day in which scattered sprinkles occur, printouts may become an annoyance because of the deluge of messages that will be printed out. If the annoyance is too great, you can simply turn off the printer. Alternatively, you can install a switch in series with the moisture sensor to allow you to disable it under such conditions.

Adding Versatility

A simple circuit connected to The Display/Output module of the project will allow WISARD to control just about anything imaginable, such as an alarm at a certain temperature or a pump upon detection of rain. Another possibility is to have WISARD turn on its own printer before it prints out a message and shut off the printer after printout is complete. This "printer controller" saves on the cost of electricity and reduces the possibility of the printer overheating.

A somewhat more sophisticated chore would be for WISARD to use two temperature sensors in a differential temperature controller for a solar heating system. Another use for a sophisticated differential temperature controller is in a cooling system that uses a large exhaust fan to keep a whole house cool at a fraction of the cost of air conditioning.

On the Display/Output module are 16 outputs that can be used to control 16 separate devices. These outputs are tied to connectors *P5016* and *P5017* in the project. Pin 1 of these connectors is tied to circuit ground, pin 10 to the +5-volt rail (see Fig. 19). Pins 2 through 9 are associated with data lines D7 through D0, respectively.

The "cleanest" way to access the pins of these connectors is with a socket like Digi-Key's Part No. WM2008 (crimp terminals Part No. WM2200 are also needed). This type of socket can also be used for the PM210 connector on the A/D Memory-Expansion module and *P301* on the Keyboard module.

The "output" pins of *P5016* and *P5017* are connected to 74LS373 octal D-type transparent latches whose outputs can sink about 25 milliamperes of current and can be used to control a solid-state buzzer or LED directly. You can hook this output to a low-power sensitive relay that has high-current contacts to control loads that require more current. Bear in mind, though, that direct connection to a relay has the potential of being problematical. When MCUs and MPUs are used, it is good design policy to isolate the logic supply from the power supply with an optical isolator or similar device.

Shown in Fig. 19 is the schematic diagram of one control circuit that works. Relay *K1* can be a Digi-Key Part No. Z620-ND, which has a 12-volt dc coil resistance of 500 ohms and has contacts rated at 0.25 ampere at 28 volts dc. Just about any similar 12-volt dc relay with a coil resistance greater than 400 ohms will probably work in this circuit. The optical isolator shown is the popular 4N25. Clamping diode *D1* eliminates negative spikes on the power line.

The Fig. 19 circuit works okay with power supplies that can deliver between 9 and 12 volts dc. Ideally, the power supply used should be

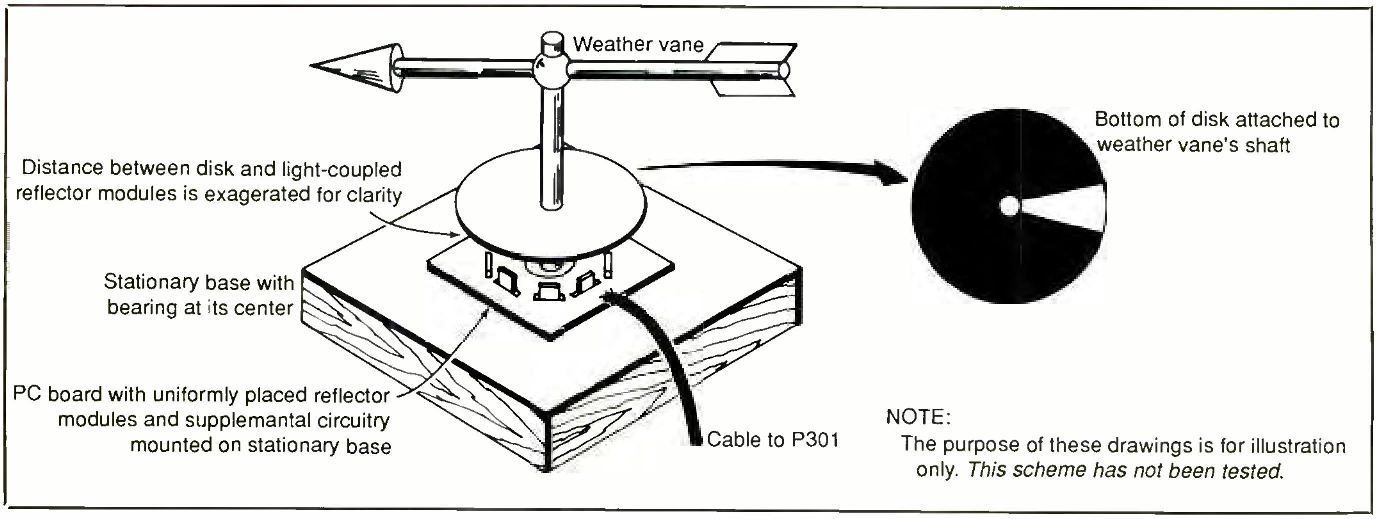

Weather vane
Bottom of disk attached to weather vane's shaft
Distance between disk and light-coupled reflector modules is exagerated for clarity
Stationary base with bearing at its center
PC board with uniformly placed reflector modules and supplemantal circuitry mounted on stationary base
Cable to P301
NOTE:
The purpose of these drawings is for illustration only. *This scheme has not been tested.*

Fig. 20. Typical example of a weather vane arrangement using eight reflector modules to indicate wind direction.

completely isolated from the supply that powers the WISARD project.

Most real-world applications require far more than 0.25 ampere of current. This should present no problems because all you have to do is simply connect a power relay and suitable power source to the contacts of *K1* in the Fig. 19 circuit. In this case, *K1* should be able to control a relay that can handle more than 20 amperes at 240 or more volts ac.

Other Modifications

If you are familiar with the previous four parts of this article, you are probably aware that about 25 percent of WISARD's circuitry is not used by the "standard" configuration of the project. The purpose of the surplus circuitry is to provide a simplified means of expanding the project. In previous installments, we have only touched upon wiring WISARD to control power devices. Now let us look briefly at what you can do with the optional inputs for the project.

Five sensors have been described so far for the basic project, but 14 more can easily be added. Six sensor inputs connect to the A/D converter, which transforms analog voltages into eight-bit digital "words." The remainder of the inputs must be connected to sensors that have one-bit (i.e. high/low) resolution—typically a switch.

As an example of an addition you can make to WISARD, assume you want the project to monitor barometric pressure. Obviously, the project cannot actually read barometric pressure. However, barometric pressure, converted into an electrical signal, can be processed into a form that WISARD does understand—in this case, a varying voltage.

To implement a barometer function, you begin with a solid-state pressure transducer. Then you add a circuit that has an output of 0 volt at a barometric pressure of 28.44 inches of mercury and 5.0 volts at 31.00 inches of mercury and connect it to pin 1 of the *PM210* connector on the A/D Memory-Expansion module. Then you enable this input with the following short routine:

```
EXPRESS PSHB
        CLRB
        STAB  5002H
WAIT3   DECB
        NOP
        NOP
        NOP
        BNE   WAIT3
        PULB
        RTS
```

To actually load the barometric pressure information into WISARD, simply call up the "LOADA" subroutine in the firmware. After it is called, Accumulator A of the 6803 processor will contain the barometric pressure information. Assuming linearity, one LSB (least-significant bit) will correspond to 0.01 inch of mercury. Inputs to pins 2 through 6 of the PM210 connector are treated similarly. However, change the address in the enable subroutine from 5002H to 5003H, 5004H, 5005H, 5006H or 5007H, respectively.

Now let us look at a rather unique example of the use of the logic inputs connected to *P301* of the Keyboard module—a wind-direction indicator.

You implement the wind-direction indicator starting with a weather vane, to the bottom of which has been fastened a disk that has a single white radial strip on the bottom. Under the disk, on a stationary circuit-board assembly, you have eight light-coupled reflector modules arranged in a circular pattern with equal angular spacing with reference to the center of the disk. (See Fig. 20)

Coupling each reflector module to a special circuit allows the system to develop a voltage that can be fed to a pin of *P301*. By providing (via firmware) some basic data concerning the disk, WISARD can infer which way the wind is blowing. This simple arrangement may not be the best for a wind-direction sensor, but it does demonstrate that *P301* can be used as more than just a means for connection of option switches to the project. (Note: *P301* is located at address 5011H, as is *LED2*, but no conflict exists. Thanks to the R/W line and decoding circuitry, *LED2* can only be written to and *P301* can only be read.)

Predicting Temperatures

Firmware for the temperature-forecasting capability of WISARD is contained in the 2732A EPROM in the A/D Memory-Expansion module. Two separate routines are supplied for minimum and maximum temperature forecasts.

The algorithm for minimum temperature forecast is primarily algebraic. The project uses the length of the day, times at sunrise and sunset and four preceding temperature measurements to formulate a quadratic equation that is solved for one hour after sunrise, which is assumed to be the low point of the temperature-versus-time curve. This algorithm results in extreme accuracy if sky or wind conditions are unchanged during the night. If these conditions do change, the predicted low temperature will be revised at the next hour. The algorithm is not "location-sensitive" and, thus, is accurate in all geographic locations on Earth.

The algorithm for the maximum temperature forecast routine modifies average measured data for specific conditions. The particular routine used in the project has been optimized for Grand Rapids, MI but is accurate in similar climates. However, it may be somewhat inaccurate —especially in winter—south of 30 degrees north latitude.

The basic data in the maximum temperature forecast routine consists of the average temperature spread (which is maximum to minimum) for each month. This program can easily be customized for a particular geographical location, as long as the average temperature spread is known for each month.

The program modifies the "average" temperature spread to take into account such variables as sunshine, cloudiness, rain, mean temperatures, time of day, time of year, and so on. This modified spread is added to the minimum temperature to obtain a forecast maximum, which itself may be modified, under certain conditions, before being printed out.

If you plan to modify—but not totally rewrite—WISARD's firmware, you need the source code. This and the Intel hex and object files are available on a 5.25-inch PC-compatible disk from the source in the Materials Availability box. The source code is written in 6801/6803 assembly language that, except for a few additional instructions, is identical to 6800/02/08 assembly language. Several built-in subroutines can help you in writing modified code. A few are enumerated below:

- **PRINT.** Used to simplify printing of messages. Before this subroutine can be called, the Index Register must be loaded with the starting address of the message. Generally, labels are used to indicate this starting address. For example, the following program segment will print out, "Long live WISARD!":

```
        LDX #MES
        JSR  PRINT
        .... .......
        .... .......
MES DB  'Long live WISARD!',0
```

Notice that the message is terminated in a 0.

- **ENASEN1, ENASEN2.** These subroutines "enable" temperature sensors *TS1* and *TS2*, respectively. To load the data into accumulator A, call the "LOADA" subroutine.
- **DELAY1S.** Creates a delay of about 1 second.

Materials Availability

A number of items that can save you time and effort in locating components and other materials for the WISARD and assembling the project are available from Magicland, 4380 S. Gordon, Fremont, MI 49412. They include: Kit of components for the A/D Memory-Expansion module (does not include programmed 2732A programmed EPROM), No. ADME1, $65; double-sided A/D Memory-Expansion module pc board, No. 680X-ADM-2, $22; complete set of precision resistors for A/D module, No. PRAD89, $5; LM335, $2 each; ADC0809, $4; 6116LP, $7; 74LS541, $1.45; 6803, $3.75; LM1830, $2.25. Available separately from same source: 27128 EPROM programmed with firmware ME1, V1.9 or later, $18; 2732A programmed EPROM, $12; both EPROMs, $25. Firmware assembly-language source code and Intel hex file on PC-compatible 5.25-inch floppy disk, $2.75. Michigan residents, please add 4% sales tax.

- **DIGITAL.** Converts the data in Accumulator A into digital format. An error results if the data exceeds 99.
- **DISPLF.** This subroutine assumes Accumulator A holds temperature data. When it is called, the display shows the temperature in degrees Fahrenheit. (**DISPLC** is virtually identical except that the display shows degrees Celsius.)
- **PRNTIME, PRNDATE.** When called, these subroutines print the time and date, respectively.
- **PRNTTMP.** This subroutine assumes temperature data is in Accumulator A and prints temperature in degrees Fahrenheit. (If OPTION switch 3 in the project is set to "off," this routine also prints out temperature in degrees Celsius.)
- **DIVIDE.** This subroutine stores the most-significant bit (MSB) of the dividend at DIVIDNM and least-significant bit (LSB) at DIVIDNL. The MSB of the divisor must be stored at

DIVISRM and LSB at DIVISRL. The quotient resides at QUOTINT.

If you are new to 680X assembly language, the A/D Memory-Expansion module has a "polite" optional 2716 EPROM on it that can contain software "bugs" that will not completely throw off WISARD. When OPTION, switch 1 is "on," the MCU does not know this EPROM exists because the program bypasses the "JSR EPR2716" instruction. Setting the switch to "off" causes the program to jump to starting address 400H of the 2716. To make use of this feature, start the "2716 program" with "ORG 0." Be sure to end the program with an RTS instruction that returns from the subroutine.

If you want to modify the firmware, you need a translator program called an "assembler." One reason for this is that the source code for WISARD's firmware is written in assembly language. Since few personal computers use the 6801/03 series MPU/MCUs, you must use a cross-assembler to assemble source code written for an MPU that is distinctly different from the MPU used by your computer. Use a text editor or word processor to write the source code and cross-assembler to translate this into a form the MCU/MPU understands. A "linker" is usually used to link the object file produced by the cross assembler into a format that is understood by most PC-based EPROM programmers. Linker programs are usually supplied with cross-assemblers.

After assembling the source code, "burn" it into an EPROM with an EPROM programmer like the stand-alone model featured in the February and March 1987 issues of *Modern Electronics*. Alternatively, you can obtain programmed EPROMs from the source given in the Materials Availability box.

A Note About Printers

WISARD has a strictly serial printer interface and will not work with a printer that has a parallel interface. This is no hardship if you own a printer that has both serial and parallel interfaces. If you plan on dedicating a printer exclusively for use with WISARD, low-cost RS-232-compatible surplus printers are available.

You do not need an actual printer for the WISARD. If you wish, a terminal or computer, the latter with terminal-emulation software, can be used in place of a printer. The only requirement for whatever print-out device you decide upon is that it can communicate at 600 or 4,800 baud.

This concludes our series on the sophisticated WISARD weather monitor and temperature predictor. With this project, you will never be without current weather-condition reports. Additionally, you will also be able to follow weather trends simply by reading earlier printed reports, and you get predictions on what the high and low temperatures will be for the day based upon those trends. In effect, you have your own abbreviated version of the weather monitoring and forecasting equipment used by professional weather-watch services. And as we have pointed out, you can also use WISARD's optional expansion capabilities to monitor for flooding in your basement, loss of refrigeration in your freezer, etc. and to initiate remedial action. ME