Microprocessor Control With BASIC (Part I)

A Development System for Microprocessor- Controlled Projects

By Jan Axelson & Jim Hughes

Microprocessor control can greatly expand the capabilities of many electronic instruments. A microprocessor-controlled "smart" thermometer, for example, can do more than just measure temperature. With proper programming, it can be made to keep track of maximum and minimum temperatures and when they occurred, or perform other temperature-related functions. With microprocessor control, it is also feasible to use panels that display informative data and messages.

Designing microprocessor-controlled devices is a complex task. In addition to building the required circuitry, you have to write the software that will control the circuits. Fortunately, you do not have to start from square one in this area. With the Microsys project presented here the task is greatly simplified because the project serves as a base system for projects like the smart thermometer. The Microsys simplifies the hardware and, particularly, the software design.

This article is presented in two parts. This month, we'll describe the basis Microsys that allows you to write, run and save programs. Next month, we'll describe the Tempwatch instrument that's created by adding a few components to the Microsys and which monitors temperature and displays present temperature, time, maximum and minimum temperatures and the times these occurred, all on a 16-character dot-matrix LCD panel.

The Preliminaries

The key to implementing successful Microprocessor Control of instruments is the Intel 8052AH-BASIC microcontroller. A microcontroller is a microprocessor, or a single-chip CPU, that is optimized for use in single-purpose (dedicated) controllers or instruments. The 8052AH-BASIC has several features that make it especially easy to use, including a built-in BASIC language interpreter, serial port,and automatic programming of EPROMs. This is the microcontroller used in the Microsys. By adding real-world inputs and outputs (sensors, displays, switches, and the like) you can program the Microsys to collect and display data or to act as a controller or other intelligent device.

While you're developing a project, the serial port of the Microsys connects to the serial port of a computer terminal. Just about any type of personal computer can be used for this in terminal-emulation mode. You can then experiment by writing, running, and debugging programs on the Microsys, using the BASIC interpreter in the 8052AH-BASIC chip and the keyboard and video display of the terminal. This frees you from having to write programs in machine code or assembly language and from having to invest in assemblers, debuggers, or other costly development tools.

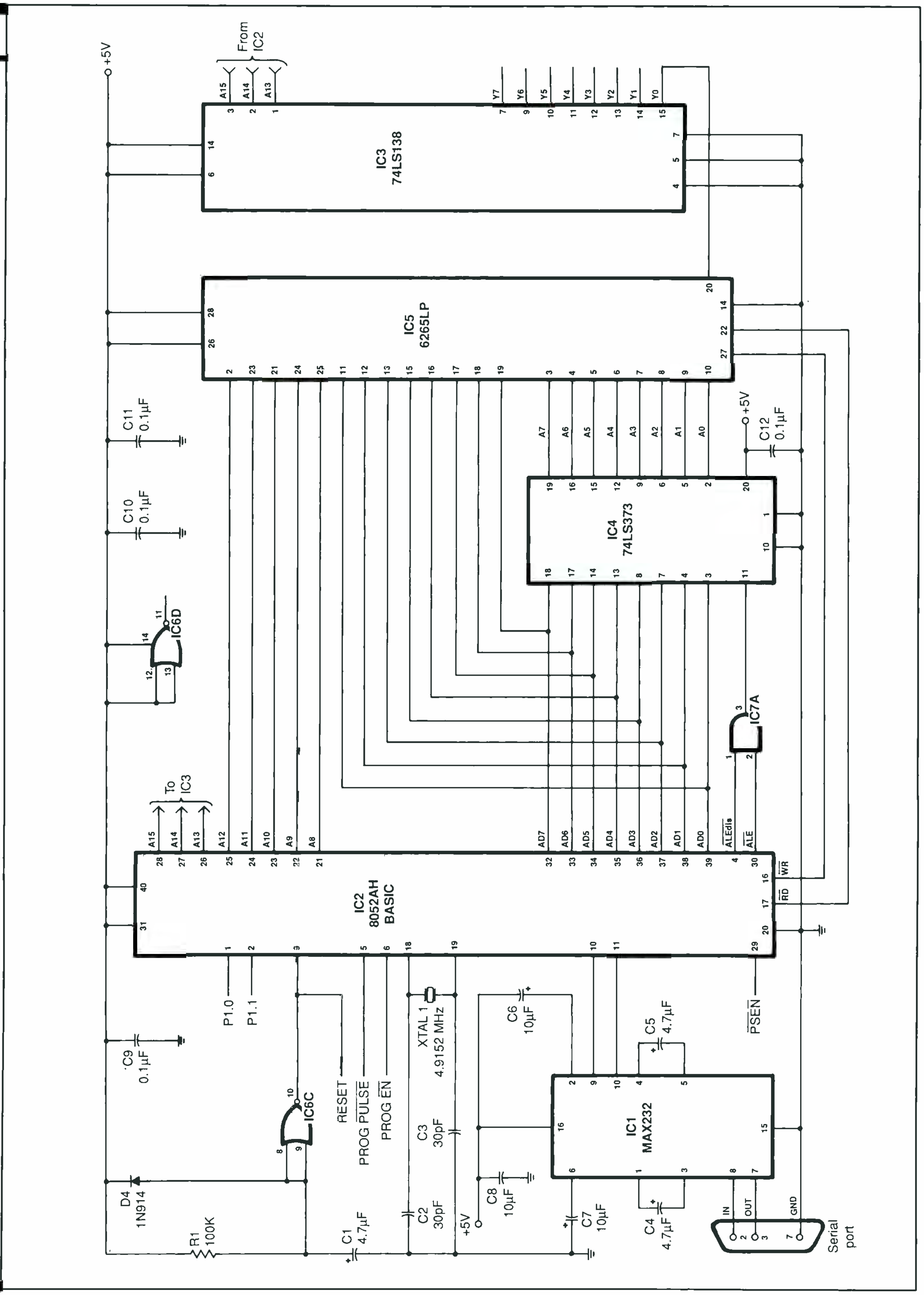
Programs you develop on the Microsys can be stored in EPROM (one command in BASIC does this automatically). When your design is completed and the EPROM is programmed, the "umbilical cord" to the serial port can be disconnected. You then have a complete, standalone project.

All components in the Microsys are readily available from mail-order suppliers. The extra capabilities of the 8052AH-BASIC IC do mean that it carries a premium price tag, $20 to $30 more than it costs for a "plain-vanilla" microcontroller without BASIC. Still, with careful shopping, component cost for the basic Microsys (minus the cost of the components that make up the power supply) runs only around $50.

To program the Microsys, you'll need a computer terminal, or a computer that can be configured to emulate a terminal, with a serial port and cable. Modem communication is an example of terminal emulation. If your computer has a serial port and can communicate by modem, it should be able to communicate with the Microsys.

We use an IBM XT-compatible computer running the Procomm communications software. With Procomm, we communicate with the

Fig. 1. Schematic diagram of the basic Microsys, minus the EPROM circuitry and dc power supply.


IC1 MAX232
IC2 8052AH BASIC
IC3 74LS138
IC4 74LS373
IC5 6265LP
IC6C
IC6D
IC7A
+5V
From IC2
To IC3
C1 4.7µF
C2 30pF
C3 30pF
C4 4.7µF
C5 4.7µF
C6 10µF
C7 10µF
C8 10µF
C9 0.1µF
C10 0.1µF
C11 0.1µF
C12 0.1µF
D4 1N914
R1 100K
XTAL 1 4.9152 MHz
P1.0
P1.1
RESET
PROG PULSE
PROG EN
PSEN
ALEdis
ALE
WR
RD
IN
OUT
GND
Serial port

PARTS LIST

Semiconductors
D1—1N914 silicon switching diode
D2—1N270 general-purpose germanium diode
D3 thru D6—1N4001 or similar 50-volt rectifier diode
IC1—MAX232 RS232 driver/receiver (Jameco, JDR)
IC2—8052AH-BASIC microcontroller (Jameco, JDR)
IC3—74LS138 3-to-8-line decoder
IC4—74LS373 octal D-type latch
IC5—6264LP 8,192 × 8 static RAM
IC6—4001 CMOS quad NOR gate
IC7—74LS08 quad AND gate
IC8—LM317 3-terminal adjustable regulator
IC9—2764 8,192 × 8 EPROM with 12.5-volt programming
IC10—7407 hex open-collector buffer
IC11—7805K +5-volt regulator in TO-220 package
Q1—2N4403 pnp switching transistor

Capacitors (25-WV)
C1,C4,C5—4.7-μF electrolytic
C2,C3—30-pF ceramic
C6,C7—10-μF electrolytic
C8 thru C13—0.1-μF ceramic
C14,C16,C17—1.0-μF tantalum electrolytic
C15—470-μF electrolytic
C18—100-μF electrolytic

Resistors (¼-watt, 5% tolerance)
R1—100,000 ohms
R2—2,000 ohms
R3—240 ohms
R4,R6,R8 thru R16—10,000 ohms
R5—4,700 ohms
R7—1,000 ohms

Miscellaneous
B1,B2—9-volt battery
F1—1-ampere slow-blow fuse
S1—Dpst toggle switch
T1—12.6-volt, 1-ampere power transformer
XTAL1—4.9152-Megahertz crystal
Perforated board with copper-ringed holes on 0.1" centers and interleaved power and ground buses; heat sink for TO-3 package; two 9-volt snap-type battery connectors; suitable enclosure; one set each of 2- and 3-pin interlocking connectors; panel-mount, solder-type 9-pin D-type connector; in-line fuse holder; rubber grommet; ac line cord with plug; heat-shrinkable tubing; Wire-Wrap IC sockets and hardware; machine hardware; hookup wire; solder; etc.

Addresses of Suppliers

Digi-Key Corp.
701 Brooks Ave. S.
Thief River Falls, MN 56701-0677
1-800-344-4539

Intel Literature Sales
P.O. Box 58130
Santa Clara, CA 95052-8130
1-800-548-4725

Jameco Electronics
1355 Shoreway Rd.
Belmont, CA 94002
415-592-8097

JDR Microdevices
110 Knowles Dr.
Los Gatos, CA 95030
1-800-538-5000

Datastorm Technologies
(For Procomm communications software)
P.O. Box 1471
Colombia, MO 65205
1-800-626-2723

Sources of Information

MCS BASIC-52 User's Manual
Intel Part No. 270010-003
Essential reference to the version of the BASIC language interpreter contained in the 8052AH-BASIC microcontroller. Also contains schematics of sample system configurations.

Embedded Controller Handbook
Intel Part No. 210918 (Jameco Cat. No. 210918)
Hardware description, specifications, and timing diagrams for the 8052 microcontroller (and many others).

Microsys and can also easily print program listings, save programs to disk and load programs from disk to the Microsys. This is just one possible configuration. Many other computer/software combinations are also feasible.

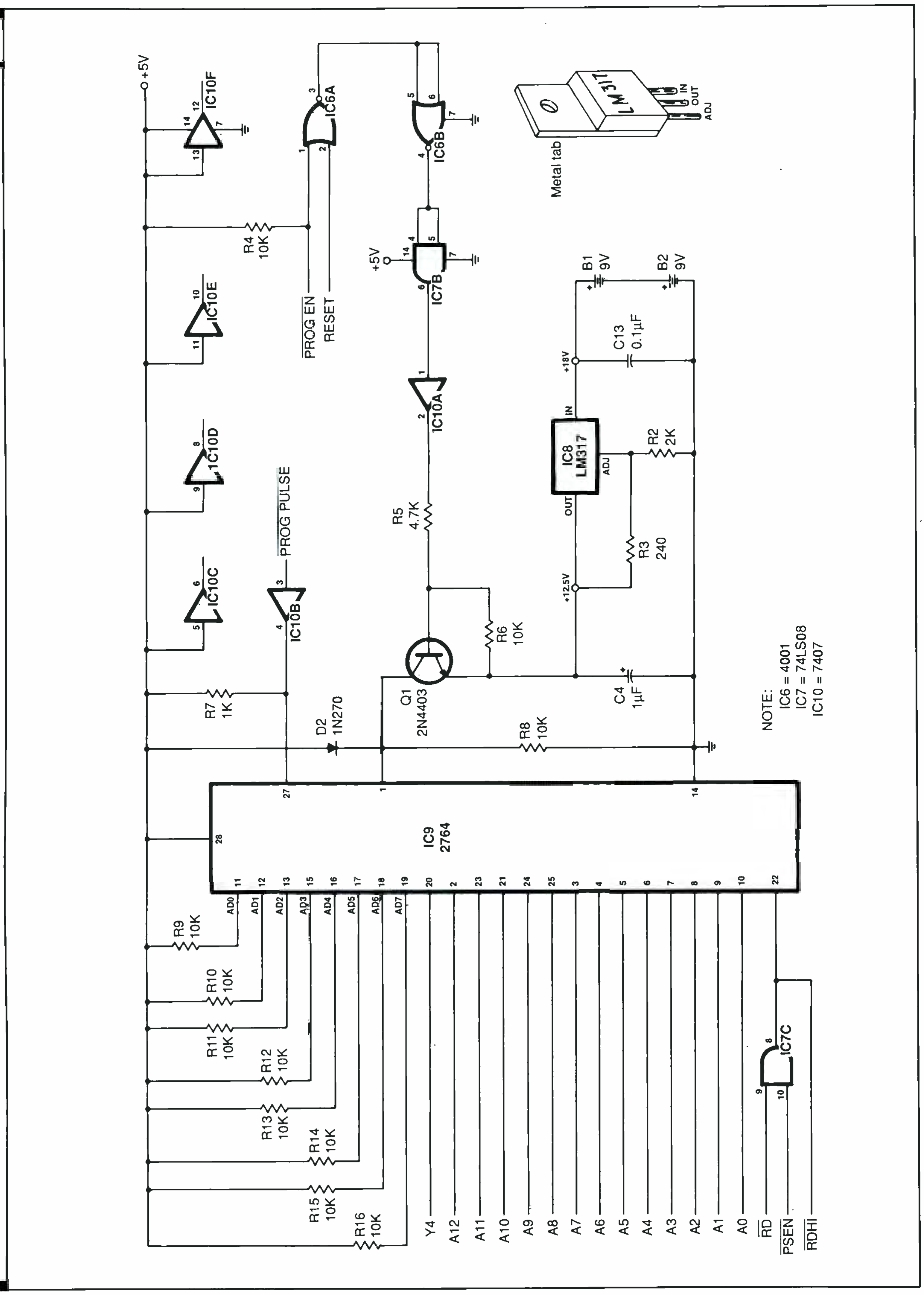
A few other capabilities and resources are highly recommended for experimenting with the Microsys. One is some familiarity with the BASIC programming language. Intel's MCS BASIC-52, which is the BASIC interpreter used in the 8052AH-BASIC, is similar to other versions of BASIC. Consequently, any BASIC experience will make programming the Microsys that much easier.

A second helpful resource is access to an EPROM eraser. This will allow you to experiment with different programs in EPROM, then erase and try new ones. EPROM erasers consist of a controlled, enclosed source of ultraviolet light, and are widely available. An EPROM eraser project appeared in the May 1987 issue of this magazine. Also, check out the *Modern Electronics* advertisers for sources of erasers.

If you're familiar with computer-relevant subjects, such as hexadecimal and binary number systems, this will also help you in understanding the Microsys. If not, the Microsys is a good place to gain some experience and practice with these and other topics related to computer hardware and software.

Finally, to experiment on your own with the Microsys, more complete documentation than can fit in this article is recommended. The user's manual for MCS BASIC-52 is a particularly important source of such documentation. This and other sources of information and components is given in the section following the Parts List.

Fig. 2. Schematic diagram of the EPROM circuitry. A battery-powered 12.5-volt dc supply provides the EPROM's programming voltage.



+5V
IC10F
IC10E
IC10D
IC10C
IC10B
IC10A
IC6A
IC6B
IC7B
IC7C
R4 10K
R5 4.7K
R6 10K
R7 1K
R8 10K
R9 10K
R10 10K
R11 10K
R12 10K
R13 10K
R14 10K
R15 10K
R16 10K
R2 2K
R3 240
PROG EN
RESET
PROG PULSE
D2 1N270
Q1 2N4403
C4 1µF
C13 0.1µF
B1 9V
B2 9V
+18V
+12.5V
IC8 LM317
IN
OUT
ADJ
LM317
Metal tab
IC9 2764
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
Y4
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
RD
PSEN
RDHI
NOTE:
IC6 = 4001
IC7 = 74LS08
IC10 = 7407

About the circuit

Figure 1 is the schematic diagram of the Microsys' circuitry, minus its EPROM and power-supply circuitry. The system shown in Fig. 1 has just five main components: *IC1* through *IC5*. The 8052AH-BASIC microcontroller is shown as *IC2*. From here on, we'll refer to the microcontroller as either the 8052 or *IC2*.

A major task of *IC2* is reading from and writing to memory and other devices. To do this requires a bus to carry the selected addresses, another bus to carry the data to be read or written, and control signals to initiate and time the processes.

The address bus of the 8052 is 16 bits wide, allowing it to access up to hexadecimal address FFFF. The low address byte is carried on eight multiplexed address/data lines (AD0 through AD7). Eight additional address lines (A8 through A15) carry the high address byte.

One of the components that *IC2* reads from and writes to is 6264 static RAM *IC5*. This RAM provides temporary storage of programs and data and stores up to 8,192 eight-bit words. Thirteen address lines (A0 through A12) are required to address all 8,192 bytes in the RAM. Addresses A0 through A7 are latched from *IC2* to *IC5* through *IC4*, a 74LS373 octal D-type latch. Address lines A8 through A12 connect directly from *IC2* to the corresponding inputs of *IC5*. Data is transferred between *IC2* and *IC5* directly on AD0 through AD7.

Reading and writing to *IC5* can occur only when its chip select input (pin 20) is low. CHIP SELECT is generated by IC3, a 74LS138 3-to-8-line decoder. Data inputs to *IC3* are A13, A14 and A15. These are the three highest address lines. For each combination of these, a unique output of *IC3* goes low. When A13, A14 and A15 are all low, Y0 at pin 15 of *IC3* is also low and *IC5* is enabled. This places the RAM at address 0 in the system. Other components of the Microsys will connect to other outputs of *IC3*.

To write a byte of data to an address in RAM, *IC2* places the low byte of the address on lines AD0 through AD7. It also places the five high bits of the address on A8 through A12. Lines A13 through A15 are low, to enable *IC5*.

A low at pin 3 of *IC7A* latches the low address byte through *IC4* to *IC5*. Then *IC2* places the data on lines AD0 through AD7, and a low at pin 16 of *IC2* (WRITE) causes the data to be written to the RAM at the selected address.

Reading a byte in RAM is a similar process. To do this, *IC2* again places an address on AD0 through AD7 and A8 through A12, with A13 through A15 held low. Pin 3 of *IC7A* again latches the low address byte through *IC4* to *IC5*. A low on pin 17 of *IC2* (READ) then causes *IC5* to place the requested data on AD0 through AD7 for *IC2* to read.

In practice, all of this is easily accomplished on the Microsys. The 8052 provides a special operator that allows you to read or write to external memory with a single statement in BASIC.

The interface for serial communication between the Microsys and the terminal is provided by *IC1*, a MAX232 RS232 driver/receiver. The MAX232 contains two charge-pump voltage converters that generate ± 10 volts from the + 5-volt supply. Pin 7 of *IC1* is the output of an RS232-compatible driver at + 9 volts, and pin 8 is the input to an RS232-compatible receiver. Pins 9 and 10 of *IC1* interface this chip to *IC2* at TTL-level voltages.

A 4.9152-MHz crystal (*XTAL1*) at pins 18 and 19 of *IC2* provides the system clock. On power-up, the slow charging of *C1* through *R1* causes a reset pulse on pin 9 of *IC2* to initialize the system.

The five ICs described in Fig. 1 are the building blocks of the basic Microsys. With this much circuitry, you can write and run programs from your terminal. However, when you power down, any programs stored in RAM will be lost.

For permanent storage of programs, the EPROM circuitry shown in Fig. 2 must be added to the Fig. 1 circuitry. In this circuit, *IC9* is a 2764 EPROM that has a storage capacity of 8,192 eight-bit words (the same as the RAM in Fig. 1). Its data and address lines are wired the same as *IC5*'s, except that pull-up resistors *R9* through *R16* on lines AD0 through AD7 are required for EPROM programming.

As was the case for *IC5*, the pin 20 CHIP SELECT line for *IC9* is controlled by *IC3*. Chip *IC9* is enabled when line A15 is high and lines A13 and A14 are low, causing Y4 at pin 11 of *IC3* to go low. This places the EPROM at hex address 8000.

Reading and programming of *IC9* are controlled by the 8052. Pin 17 (RD) and pin 29 (PSEN, or Program Store ENable) of *IC2* are ANDed in *IC7C* to provide RDHI, the READ signal for *IC9* (and other devices located above hex address 8000).

To program the EPROM, + 12.5 volts must be applied to pin 1 of the device. Since this voltage is used only for EPROM programming, it's practical to use battery power to generate it. In this circuit, two 9-volt batteries in series (*B1* and *B2* provide a + 18-volt input to LM317 adjustable voltage regulator *IC8*. These batteries can be inserted for EPROM programming and can be removed during "normal" operation of the Microsys.

A reference potential of 1.25 volts is developed by *IC8* across *R3*. This determines the amount of current that flows through *R2*, whose value is chosen so that the output of *IC8* is + 12.5 volts. Input bypassing is provided by capacitor *C13*, while transient suppression is provided by *C14* at *IC8*'s output.

When PROG EN (PROGram ENable) pin 6 and RESET pin 9 of *IC2* are

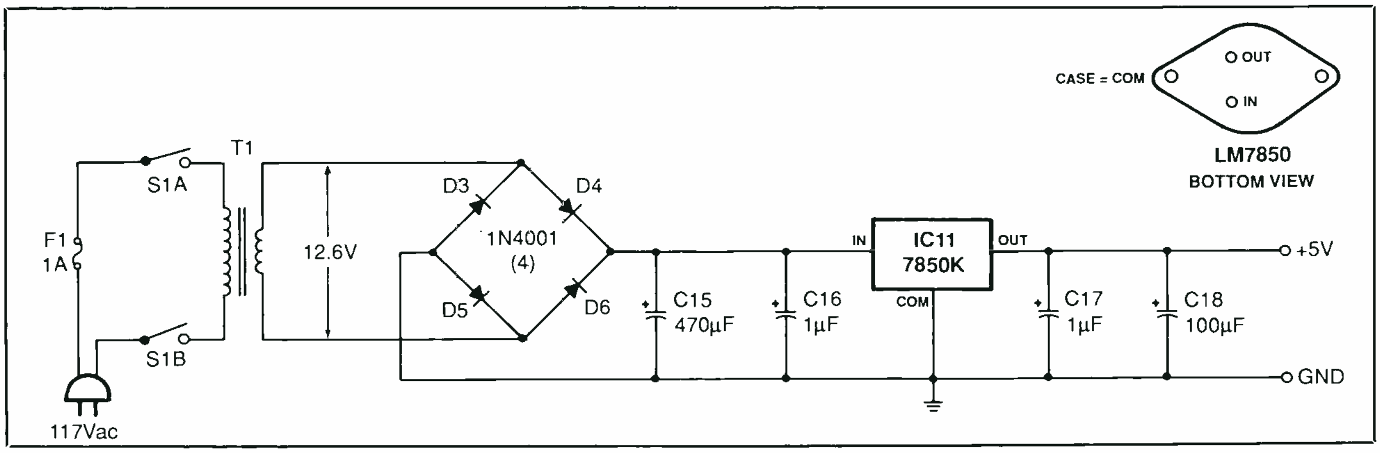


Fig. 3. Schematic diagram of the Microsys' 5-volt dc power supply.

both low, pin 2 of *IC10A* is also low. This sends transistor *Q1* into conduction and switches the + 12.5-volt output of *IC8* to pin 1 of *IC9*.

Series NOR gates *IC6A* and *IC6B* "OR" the PROG-EN and RESET signals. Note that *IC7B* drives *IC10A*, a 7407 open-collector buffer, which allows the 5-volt output of *IC2* to control the programming voltage. Programming pulses from *IC2* are buffered at *IC10B* and are then fed to input pin 27 of *IC9*.

For normal (non-programming) operation, diode *D2* and resistor *R8* hold pin 1 of *IC9* at about + 4.7 volts. Diode *D2* is a germanium type, selected to keep the potential at pin 1 as close to + 5 volts as possible.

Figure 3 schematically shows the circuitry of the + 5-volt power supply recommended for powering the Microsys. Transformer *T1* steps down the incoming 117 volts from the ac line to 12.6 volts ac. Then the bridge rectifier made up of diodes *D3* through *D6* rectify this voltage to pulsating dc, and *IC11* regulates it to + 5 volts dc.

Construction

Wire Wrap is the recommended method of construction for the Microsys. (See the "Wire-Wrapping Tips" box for details on wrapping techniques and Wire Wrap equipment.) Use a perforated board with copper-ringed holes on 0.1-inch centers and interleaved power and ground buses. A board that measures 8 by 4.5 inches will accommodate components with room to spare. (The extra room on the board will be used for the components for the Tempwatch that will be described next month, or other components for your own circuit designs.)

The upper photo in Fig. 4 shows the prototype Microsys assembled on perforated board, while the lower photo shows the project wired using the Wire Wrap method.

Addresses of a few suppliers that stock the components used in this project are given at the end of the Parts List. Other suppliers than those cited may also offer most or all of these components.

For connection to the serial-port cable of the terminal, the Microsys requires a female 25-pin D-type connector. In most cases, this connector must be wired to transmit on pin 3 and receive on pin 2, and pin 7 should be used as signal ground. This makes the Microsys' serial port compatible with the DTE (Data Terminal Equipment) type of RS-232 serial ports found on the majority of personal computers. If your computer's serial port is configured differently, you must determine the wiring scheme required and adjust the wiring on the serial-port connector on the Microsys accordingly.

In addition to the transmit, receive, and ground conductors, some serial ports require extra "null-modem" connections for direct communication with the Microsys. These normally involve connecting together pins 4, 5, and 8 and pins 6 and 20 somewhere in the link. If your system requires these connections, they can be wired at the serial-port connector on the Microsys.

Molex or locking-type connectors are recommended for use on the conductors that connect the serial-port connector and power transformer to the circuit board. These will allow you to easily remove the circuit-board assembly from its enclosure, for changes or/and additions you later decide to make to the circuits. One set each of two- and three-pin connectors are required.

The 4.9152-MHz specified for *XTAL1* is one of several frequencies recommended in the MCS BASIC-52 User's Manual to assure best accuracy of the 8052's real-time clock (which will be used in next month's smart thermometer). It's also possible to replace *XTAL1, C2,* and *C3* with a TTL crystal oscillator of the same frequency. In this case, the oscillator's output connects to pin 18 of *IC2*, with pin 19 connected to ground.

Begin construction of the Microsys

Tips On Wire Wrapping

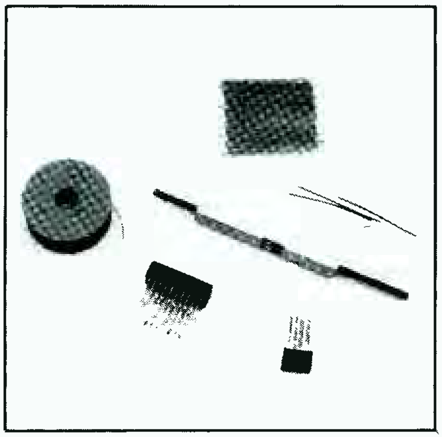
Because of its simplicity and flexibility, the Wire Wrap technique is the recommended method for interconnecting components in the Microsys. Microsys' multiplexed data and address buses can easily be Wire Wrapped, whereas printed-circuit wiring would most likely have required intricate routing of traces between IC pins, or a difficult-to-fabricate double-sided pc board.

Another advantage is that Wire Wrapped circuits can easily be changed. Modifications are made and errors corrected quickly and easily, simply by unwrapping the connections and wrapping new ones.

If you're new to Wire Wrapping, here are some tips on how to use this technique to wire circuits, and on when (and when not) to use it.

Wire Wrap is a largely solder-free method of wiring circuits. Connections are made by tightly wrapping several turns of wire onto special, square Wrap posts, the corners of which bite into the wires to form solid electrical and mechanical connections without the need for solder.

For the beginner, the investment in Wire Wrap equipment is small—$15 or so for a multi-purpose Wire Wrap tool and a spool of the special wire used with it. Perhaps the biggest continuing expense is the special Wire Wrap IC sockets required by the medium. These have long, square posts suitable for several levels of wraps. They are usually more expensive than the solder-type sockets used with printed-circuit boards.


Wrapping wire is available on spools, or in packets of precut and pre-stripped lengths. A single tool strips, wraps, and unwraps the wire onto special Wire-Wrap IC sockets with long, square posts.

A multi-purpose tool will strip, wrap, and unwrap the wires. Electrically powered Wire Wrap tools are also available, but a simple "finger-powered" tool is all you need for occasional use. To make successful Wire Wrap connections, you must use Kynar-insulated No. 30 wire, either in spool form or in packets of pre-cut, pre-stripped lengths.

Wire Wrap tools, hardware and other materials are available from many sources. Shown in the photo is an assortment of such paraphernalia.

A convenient type of circuit board to use for Wire Wrap projects is perforated board with holes on 0.1-inch centers, with copper pads around each hole and interleaved power and ground buses that help ensure that the supply and ground connections are low in impedance and less prone to noise. The pads make it easy to solder the IC sockets to the board. Some boards also include rows of three-hole pads for making soldered connections between components.

You can also Wire Wrap on perforated board that has copper-ringed holes but no buses or multi-hole solder pads. A plain perforated board is another possibility, though the copper-ringed holes are convenient and are recommended for use in large projects.

When building a circuit using the Wire Wrap technique, plan board layout by trial-inserting the IC sockets and other large components you'll be using. As much as possible, arrange the components that interconnect with each other in close-proximity to each other. When you've finished with the layout, install the IC sockets in their planned locations and solder their pins to the pads on the circuit board. If you're prototyping and plan to dismantle the circuit later, tack-solder only two pins diagonally opposite each other to the pads. When you're through using the circuit, desolder the pins and recycle the sockets into other projects. For a permanent installation, however, solder each socket pin to the board.

by preparing two 8-inch and eight 4-inch lengths of hook-up wire. Strip ½ inch of insulation from the ends of each. If you're using stranded hook-up wire, tightly twist together the fine conductors at both ends and sparingly tin with solder. To one end of five of the 4-inch wires, crimp female pins for the locking connectors. Crimp male pins to one end of the other three 4-inch wires. Determine which are the secondary leads of *T1* and crimp and solder pins to these. Insert the pins on *T1*'s wires into a 2-pin connector, and insert the other pins into the other three connectors.

Next, plan your preliminary parts layout on the circuit board. All components except *T1*, *F1*, and *S1* should mount directly on the board. Group the power-supply components together, and be sure to reserve enough space for *IC11*'s heat sink.

The TO-3 package of *IC11* requires two mounting holes in the circuit board and two slightly enlarged holes for its input and output pins. The case provides the ground connection, so if possible, locate the IC on the board so that at least one of its mounting screws overlaps a ground trace on the board.

A mounting hole is also required for *IC8*'s TO-220 case. Because this voltage regulator is required to deliver just 30 milliamperes, no heat sink is required for *IC8*.

When you have a preliminary parts layout planned, drill the holes for *IC11*, *IC8*, and for mounting the circuit-board assembly inside the selected enclosure. These holes can be

Liberal use of labels on the circuit board helps keep you oriented as you wire the circuit. Small adhesive labels marked with the IC number can be mounted between the rows of pins of each socket on the wiring side of the board. Also, place a dot in one corner of each label to identify pin 1's location.

An alternative to homemade labels is preprinted Socket Wrap ID labels that slip onto the socket pins before wrapping. Pin numbers are labeled on these, and you can add your own identifying information.

To wrap a wire between two socket posts, choose or cut a length of wire long enough to make the connection comfortably and add 2 inches. Avoid routing wires between adjacent posts on a socket, but routing between the two rows of posts on a socket is fine. Route the wires around the rows, even if this means using a slightly longer wire.

Unless you're using pre-stripped wire, strip 1 inch of insulation from both ends of the wire. Most Wire Wrap tools include a slot for stripping the wire. Insert the wire 1 inch into the slot, push the wire to the bottom of the slot, and pull the end to be stripped through the slot. Repeat the process at the other end of the wire. The slot has a sharp blade in it that severs the insulation without nicking the wire.

To wrap the wire onto a post, insert the wire into the smaller of the two holes on the wrap end of the tool. Push the entire stripped end and about ¼ inch of insulated wire into the tool to ensure that the wrap will begin with a turn or two of insulated wire to reduce the chance of a bare wire shorting against an adjacent post. Bend the wire 90 degrees where it leaves the bottom of the tool.

The other hole on the wrap end of the tool fits onto a socket post for wrapping the connection. Push the tool onto the post until it's bottom is flush with the circuit board or a previous wrap. Grip the free end of the wire to keep it from spinning with the tool and rotate the tool several turns, until it spins freely and the wire is wrapped onto the post.

Lift the tool from the post, and follow the same procedure for the other end of the wire. The posts on most Wire Wrap sockets accommodate three or more levels of wraps to permit multiple connections to a given point.

For connections to resistors, capacitors, and other components, you have a couple of options. One is to use special slotted Wire Wrap posts to mount the components. The posts friction-fit into perforated board (soldering is optional). On the component side of the board, the component leads are pressed into the slots on the posts. On the reverse side, connections are wrapped as usual. The advantage of this method is that components can easily be lifted out and changed. The disadvantage is the added cost of the special posts.

Another option is to insert and solder the components directly into the pads on the perforated board. You can Wire Wrap connections onto the leads of most components, but it's a good idea to also solder any such wrap to ensure a sound connection. Some component leads will be too thick or thin to accommodate the wrapping tool and will have to be hand-wrapped with longnose pliers and soldered.

Soldered connections aren't as easily changed as the ones on the Wire Wrap posts, but either method is workable. You might use the posts for components you may have to change and solder the ones that you're confident are permanent.

As you wrap each connection in a circuit, check it off on your schematic diagram (or a photocopy of it) to save you from having to mentally keep track of what you have and haven't yet wired.

To unwrap a connection, push the unwrap end of the tool onto the post and rotate the tool in the opposite direction from the wrap until the connection is free.

Finally, Wire Wrap isn't always the appropriate method to use. High-current circuits (more than 200 milliamperes) aren't suitable for Wire Wrap's small-diameter wires. Also, if you need more than a few of the same circuit, printed-circuit-board construction may be more economical. Quite often, though, Wire Wrap is a good choice. If you haven't yet, or haven't recently, given the Wire Wrap technique a try, consider using it the next time you have to build a circuit.

made by enlarging existing holes in appropriate locations on the circuit board.

Now begin wiring the +5-volt power supply, using Fig. 3 as a guide. The power supply output is several hundred milliamperes, so Wire Wrap (with its small-diameter No. 30 wire) shouldn't be used for these components. Instead, use solid hookup wire of 24 gauge or larger, and solder the wires into place.

Place a small amount of heat-transfer compound between the heat sink and regulator *IC11* to provide efficient heat conduction. Then mount the heat sink and regulator on the circuit board. Ensure that *IC11*'s case makes sound electrical connection to ground on the circuit board. You can make this connection either directly from a mounting screw to the ground bus or via a separate jumper wire to which is attached a spade or ring lug at the mounting screw and a soldered connection to ground at the other end.

Solder a connection between the output of *IC11* and the +5-volt bus on the circuit board. To aid you as you wire the rest of the project, label the buses on the circuit board with indelible marker or adhesive labels and make photocopies of the schematics.

Insert and make appropriate connections for *C15* through *C18*, *D3* through *D6*, and *IC11*. Mark off on the photocopy of Fig. 3 each wire run as you make it, and take care to orient all components correctly. Off-

(Continued on page 80)

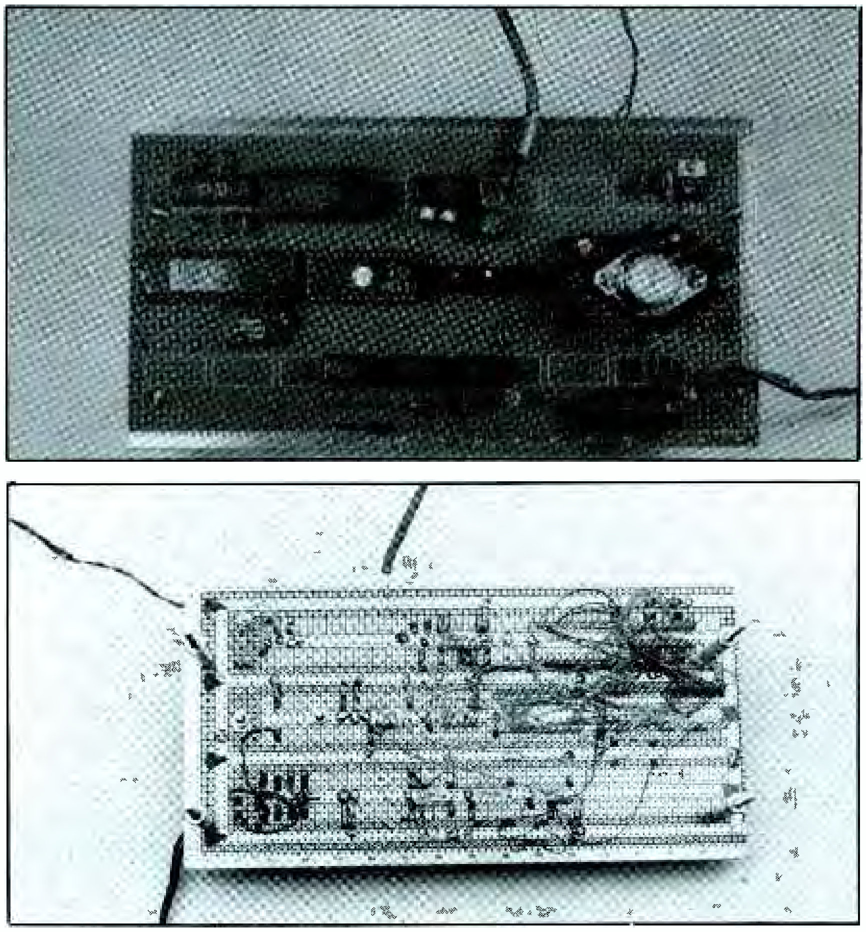


Fig. 4. Microsys components can be mounted on perforated board (upper photo) and be Wire Wrapped together (lower photo).

board connections to *T1*, *S1*, and *F1* will be added later.

Next, wire the + 12.5-volt supply, marking off each component and conductor run as you make it on the photocopy of Fig. 2. Mount *IC8* on the circuit board by bending its legs 90 degrees backward (toward the metal tab) and inserting them so that the mounting hole on the IC lines up with the prepared hole in the circuit board. A small machine screw and nut through the IC's mounting hole secures the IC to the board. Then wire into the circuit *IC8*, *C13*, *C14*, *R2*, and *R3*.

Slip a 1-inch length of small-diameter heat-shrinkable tubing over the red-insulated lead of one battery snap connector. Twist together this lead and the black-insulated lead of the other connector. Solder the connection and then slide the tubing over the connection and shrink it solidly into place. Solder the free red-insulated lead on the connector pair to the IN of *IC8*; solder the remaining free black-insulated lead to a ground point on the circuit board.

Now wire the rest of the Microsys. Following your preliminary layout, install and solder into place the sockets for *IC1* through *IC7*, *IC9*, and *IC10*. (Do *not* install the ICs in their respective sockets until you've wired the entire circuit and checked for proper voltage distribution with the power supply turned on.)

Use self-adhering or socket-identification labels to identify the number of the IC and indicate the location of pin 1 in each case on the wiring side of the circuit-board assembly.

Connect the power-supply and ground pins on the sockets to the power and ground buses on the circuit board. Be aware that the *IC1* socket is the only one with unconventional V+ or ground pin locations. In this case, pin 15 is to be grounded and pin 16 is to be at + 5 volts. For all other sockets, ground is at the lower-right pin and V+ is at the upper-left pin on the wrap side of the board (pins 7 and 14 of a 14-pin socket, pins 8 and 16 for a 16-pin socket, and so on).

Install and solder into place bypass capacitors *C8* through *C12* at roughly even intervals between the + 5-volt and ground buses on the circuit board. Then use Fig. 1 and Fig. 2 as guides as you continue wiring the circuit. As necessary, insert and solder *XTAL1*, *Q1*, *R1*, *R4* through *R16*, *D1*, *D2*, and *C4* through *C7*. Observe proper orientation for *XTAL1*, *Q1*, *D1*, *D2*, *C1* through *C5*, and *C12*.

At *IC1*, notice especially the orientations of *C6* and *C7*. In a properly wired circuit, pin 6 of *IC1* will be at − 10 volts; so the positive (+) lead of *C7* must connect to ground. Similarly, pin 2 will be at + 10 volts; so the negative (−) lead of *C6* must connect to + 5 volts on the circuit board.

Select an enclosure for the project that is large enough to accommodate the circuit-board assembly and off-the-board components without crowding. Make certain that the enclosure is deep enough to ensure that the Wire Wrap socket pins on the circuit-board assembly will not contact the floor panel.

Begin preparing the enclosure by planning parts placement. This done, drill mounting holes in the floor panel for mounting the circuit-board assembly, power transformer *T1*, and two 9-volt battery clips. Drill a hole through the back panel for the ac line cord and cut the slot for the serial-port connector. An opening for a D-type serial-port connector can be made by first drilling a ⅜-inch starter hole and then using a nibbling tool to complete the task. Two small holes are also required for the D connector's mounting hardware.

Drill a hole through the front panel for POWER switch *S1*. Locate this hole to one side of the panel to leave room for adding more controls and displays later as needed.

Next, wire *T1*, *S1*, and *F1* into the circuit, using heat-shrinkable tubing to insulate all exposed connections. Connect the two prepared 8-inch-long wires to the primary leads of *T1*. Twist each pair of wire ends together, solder the connections and insulate them with heat-shrinkable tubing. Slide another length of tubing over each wire, solder connections to *S1A* and *S1B*, and shrink the tubing over the switch terminals.

Insert a rubber grommet into the ac line cord's hole. Feed the free end of the line cord through the grommet into the enclosure and tie a strain-relieving knot in it several inches from the inside end. Tightly twist together the fine wires in each line-cord conductor and sparingly tin with solder. Then solder the in-line fuse holder's leads to a terminal of *S1A* and to the "hot" conductor of the ac line cord. Solder the free end of the other line cord conductor to the remaining lug on *S1B*. Use heat-shrinkable tubing to insulate all connections.

Crimp and solder the unconnected wires at the two-pin locking connector to the junctions of *D3/D4* and *D5/D6*.

Three wires connect the Microsys to its serial-port connector. Solder the wires on one of the prepared three-pin locking connectors to the appropriate pins on the serial-port connector, using either Fig. 1 or your own configuration as a guide. Solder the three wires on the matching connector to pins 7 and 8 of *IC1* and to ground on the circuit board, as appropriate.

Carefully check over your wiring. If you're satisfied with your work, mount the circuit-board assembly in place, using suitable-length spacers and machine screws. Then mount into place the battery clips, serial-port connector, *T1*, and *S1*. Install *F1* and connect the locking connectors.

Checkout & Use

The first step in circuit checkout is to verify power-supply operation. Plug in and turn on the +5-volt supply and verify that its output is within 0.2 volt of +5 volts. Also check for +5 volts between the V+ and ground pins of each IC and for +12.5 volts at the output of IC8. If your Microsys fails any of these tests, locate and rectify the problem before proceeding.

When everything looks okay, turn off the power supply and allow the charges to bleed off the electrolytic capacitors. Then install the ICs in their respective sockets. Make sure you plug each IC in the correct socket and in the proper orientation and that no pins overhang the socket or fold under between IC and socket.

Connect the Microsys' serial port to your terminal's or computer's serial-port cable. Turn on the terminal or computer and configure it for operation with 8 data bits, no parity, and one stop bit. Baud rate used is your choice. On boot-up, the 8052 senses the transmission rate and matches it. A good initial selection is 1,200 bits per second. (If you use Procomm communications software, these parameters are set in the Line Settings menu, with selection #8.)

Turn on the Microsys, and press the space bar at your terminal or computer. If everything is okay, the sign-on message:

```
*MCS-51(tm) BASIC V1.1*
READY
```

will appear on-screen. This indicates that you can now use the project. If you don't see the sign-on message, don't despair. It just means it's time to troubleshoot.

First off, the space bar at the keyboard *must* be the first key you hit after powering up the Microsys. If you don't hit the space bar first, the project will not respond properly. However, if you should press any other key first, turn off the Microsys, wait 10 seconds, and turn it back on. Press the space bar. If you still don't see the correct sign-on message, check to make sure the terminal or computer is configured correctly and that the serial connector is wired correctly.

Recheck the other wiring in the Microsys. Be sure all connections are made correctly and that no inadvertent short circuits or open connections have crept into the project. Verify that no connections have been left out. With persistence, you'll soon find the cause of the problem that's keeping your system from booting up and repair the situation.

When the system has booted to BASIC, a few simple tests will verify that your Microsys is functioning properly.

After seeing the sign-on message, key in:

```
PRINT MTOP
```

and press RETURN. The terminal or computer should respond with:

```
8191
```

which tells you the amount of user RAM available. Following this, key in:

```
XTAL=4915200
```

and press RETURN. (From here on, each line you key in is to be followed by a RETURN, and a RETURN is to be used at the end of each entry.) This tells the 8052 the frequency of the crystal installed in your Microsys. The 8052 uses this information to time signals such as its real-time clock and EPROM programming pulses.

Type:

```
PRINT XTAL
```

The terminal should respond with:

```
4915200
```

Listing 1. Sample BASIC Program

```
1     REM Program 1, for-next loop
10    FOR I=1 TO 10
20    PRINT I
30    NEXT I
40    END
```

which shows that your previous message was received. Now try typing in and executing a program. Listing 1 is a simple program you can type in at the keyboard, then run on the Microsys.

Key in Listing 1. If you make an error while typing in a line, whether on this or any other program, erase characters with the DELETE key. Once you press the RETURN key, however, the line can be changed only by retyping it completely. To execute the program, type:

RUN

The terminal should respond by listing the numbers from 1 to 10. If the program doesn't run properly, type:

LIST

to review the program entered. Retype the entire line that has an error in it. Then type:

LIST

again to ensure that it's correct. If it is, type:

RUN

to execute the corrected program.

The command "NEW" erases all lines previously entered for the current program and allows you to begin fresh.

The program in Listing 2 prompts you to enter a temperature in Fahrenheit degrees and then converts it to Celsius and displays both on-screen. Type in this listing and run the program as you did for Listing 1.

Listings 3, 4, and 5 are more sample programs. The program in Listing 3 lets you read or write data to a location in RAM.

Program 3 first asks you if you want to read or write. It then jumps to the appropriate subroutine and prompts you for the information needed. If reading is selected, the special operator XBY in line 80 reads the data stored at the address requested, and line 90 displays the data read on-screen. If writing is selected, line 140 writes the data into the address requested.

Writing to addresses below 1000 isn't advised, as you might over-write the program you're executing or other essential data required by the system. This could "crash" the system. If you do inadvertently cause a system crash, power down momentarily, then power up and press the SPACE bar to reboot.

Listing 4 shows the somewhat unconventional way that MCS BASIC-52 handles string, or text, variables. The STRING statement in line 10 sets aside a portion of memory, and is required if string variables are to be used. The second number in the STRING statement is the maximum length of string variables allowed plus one. The first number is the number of string variables allowed plus one, multiplied by the second number. Line 10 allocates space for one eight-character string variable, which is declared in line 20. Line 30 displays the string on the screen, and lines 40 through 60 select each character in the string individually and display each on a separate line on the screen.

Listing 5 uses the 8052's real-time clock that creates a 60-second timer on the terminal's or computer's screen. Line 20 turns on and initializes the clock, which automatically counts seconds in 5-millisecond steps and stores the count in the variable TIME. The ONTIME statement in line 50 causes an interrupt when TIME is equal to or greater than 1 second. The program then jumps to line 80, where it subtracts one from TIME, increments the seconds count, and displays the count on the screen. The program then returns to the main loop at lines 40 through 60. The program ends after 60 seconds have been counted.

These sample programs illustrate that MCS BASIC-52 is in many ways similar to other BASICs. There are some differences, of course: MCS

Listing 2. BASIC Temperature-Conversion Program

```
1      REM Program 2, Fahrenheit to Celsius
10     DO
20     INPUT "Enter a Fahrenheit temperature (0 to quit):  ",FTEMP
30    CTEMP=(FTEMP-32)*5/9
40     PRINT FTEMP," degrees F = ",CTEMP," degrees C"
60     UNTIL FTEMP=0
70     END
```

Listing 3. BASIC Program For Writing/Reading Data to a RAM Location

```
1      REM Program 3, read and write to RAM
10     DO
20     INPUT "Enter 0 (read), 1 (write), or 2 (quit):  ",RW
30     IF RW=0 THEN  GOSUB 70
40     IF RW=1 THEN  GOSUB 120
50     UNTIL RW=2
60     END
70     INPUT "Enter an address to read (0-8191):  ",ADDRESS
80    BYTE=XBY(ADDRESS)
90     PRINT BYTE," is stored in address ",ADDRESS
100    PRINT
110    RETURN
120    INPUT "Enter an address to write to (1000-8191):  ",ADDRESS
130    INPUT "Enter data to be written (0-255):  ",BYTE
140   XBY(ADDRESS)=BYTE
150    PRINT BYTE," has been written to address ",ADDRESS
160    PRINT
170    RETURN
```

Listing 4. BASIC Program That Shows Unconventional Way MCS BASIC-52 Handles Text Variables

```
1     REM Program 4, string variable
10    STRING 18,9 :  REM reserve string space in memory
20   $(0)="MICROSYS" :  REM create string variable
30    PRINT $(0) :  REM display string
40    FOR I=1 TO 8 :  REM display each character separately
50    PRINT CHR($(0),I)
60    NEXT I
70    END
```

Listing 5. BASIC Program For On-Screen 60-Second Timer

```
1     REM program 5, real-time clock
10   XTAL=4915200
20    CLOCK 1 : TIME=0 :  REM start and initialize timer
30   SEC=0 :  REM initialize seconds
40    DO
50    ONTIME 1,80 :  REM when time=1 second, go to line 80
60    WHILE SEC<60
70    END
80    REM increment time subroutine
90   TIME=TIME-1 :  REM decrement time
100  SEC=SEC+1 :  REM increment seconds
110   PRINT SEC
120   RETI  :  REM return to main program
```

BASIC-52 is in some ways more limited, and in other ways, more powerful, than other BASICs.

Enhancements include the specialized statements and commands that allow you to check for interrupts, read and write to memory and I/O ports, and program EPROMs. Limitations of the language are mainly due to the fact that the entire BASIC interpreter must fit into the 8k bytes of ROM available in the 8052.

We don't have enough space here to give more than give a few examples of MCS BASIC-52 programs. The User's Manual describes the language fully and is essential for serious experimenting with the Microsys.

When you want to save a program, you can store it (and up to 254 other programs, space permitting) in the EPROM. The BASIC command PROG writes the program currently in RAM to the EPROM. To save the program that you've just run, install *B1* and *B2* in the project via the two battery snap connectors and then type:

PROG

Microsys will then program the EPROM with the program currently in RAM. The terminal or computer displays a "1" to show that this is the first program to be stored in the EPROM. When programming is done, you'll see the READY prompt at the terminal.

The 8052 has two modes of operation. In RAM mode (the default), the RUN command executes the current program in RAM. In ROM mode, the RUN command executes the requested program in EPROM. To specify the mode, type "RAM" or "ROMn," where *n* is the number of the EPROM program wanted.

To execute the program now in EPROM, type:

ROM1
RUN

This switches to ROM mode, selects the first ROM program (the only one, so far), and executes it. To return to RAM mode, type:

RAM

Powering down doesn't affect the programs saved in EPROM. On powering up again, you can run any program saved in EPROM by switching to ROM mode with the appropriate program selected.

Perhaps you'd like to make changes to the program stored in EPROM. The information in the EPROM can't be changed (except by erasing *everything* programmed into the chip's ROM section), but you can edit a stored program if you transfer it back into RAM. To do this, type:

ROM1
XFER

This selects program 1 in the EPROM, transfers it to RAM, and switches to RAM mode.

You can now make any changes you wish to the program by adding, deleting, or re-entering program lines. To save the new, edited program to EPROM, type:

PROG

The terminal or computer will display a "2" to show that this is the second program to be stored in the EPROM.
To run the new program from EPROM, type:

ROM2
RUN

To return to RAM mode, type:
RAM

Closing Remarks

Your complete, functioning Microsys allows you to write programs, save them to EPROM, and run them from RAM or from EPROM. Feel free to experiment by writing and running your own programs on the Microsys.

Next month, we'll show you how to add such "real-world" inputs and outputs as sensors and displays, as you transform the Microsys into the Tempwatch smart thermometer. Of course, the Tempwatch is just one example of what you can do with the Microsys. The Microsys has wide capabilities, yet it is user-friendly, providing a good introduction to the topic of microprocessor control. ME

LETTERS

Project Updates

• Here are a few clarifications to the schematics for my "Microprocessor Control With BASIC, Part I" article in the April 1989 issue: In Fig. 2, be sure to wire AD0 through AD7 pins 11 through 19 of IC9 to AD0 through AD7 pins 39 through 32 (note the reverse order here) of IC2 in Fig. 1. In Fig. 1, IC5 is a 6464LP and in Fig. 3, IC11 is a 7805K. Also, the telephone number for Datastorm Technologies is now 314-474-8461.

Jan Axelson

LETTERS

Project Updates

• Here are a few clarifications to the schematics for my "Microprocessor Control With BASIC, Part I" article in the April 1989 issue: In Fig. 2, be sure to wire AD0 through AD7 pins 11 through 19 of IC9 to AD0 through AD7 pins 39 through 32 (note the reverse order here) of IC2 in Fig. 1. In Fig. 1, IC5 is a 6464LP and in Fig. 3, IC11 is a 7805K. Also, the telephone number for Datastorm Technologies is now 314-474-8461.

Jan Axelson