The following material is excerpted from:

# *The Microcontroller Idea Book*

***Circuits, Programs, & Applications***

***featuring the 8052-BASIC Microcontroller***

by Jan Axelson

ISBN 0-9650819-0-7
Published by Lakeview Research
Distribution by International Thomson Publishing (ITP) in arrangement with
Peer-to-Peer Communications.

For more information, contact:

Lakeview Research
2209 Winnebago St.
Madison, WI 53704
USA

Phone: 608-241-5824
Fax: 608-241-5848
Email: jaxelson@lvr.com
World Wide Web: http://www.lvr.com

# 2

# Inside the 8052-BASIC

This chapter introduces you to the 8052-BASIC chip, including the kinds of projects you can do with it, what equipment, materials, and skills you need in order to design and build an 8052-BASIC project, and a pin-by-pin look at the chip and its abilities.

## Possibilities

The 8052-BASIC microcontroller is an easy-to-use, low-cost, and versatile computer-on-a-chip. It’s ideal for projects that require more than an assortment of logic gates, but less than a complete desktop computer system with a full keyboard, display, and disk drives. If you’re interested in doing more with computers than simply running applications programs, the 8052-BASIC gives you a chance to design and build a system from the ground up.

With a few support chips and a program stored in memory, you can use the 8052-BASIC to sense, measure, and control processes, events, or conditions. Here are just a few examples of the uses you can put it to:

- data collection
- machine control
- test equipment
- wired and wireless links for communications and control

The 8052-BASIC is actually two products in one: it's an 8052 microcontroller, with the BASIC-52 programming language on-chip. To begin using the 8052-BASIC, you need a minimum circuit consisting of the 8052-BASIC and some support components, plus a personal computer. This book contains specific instructions for use with "IBM-compatible," or MS-DOS, computers, but you can use any computer that has an RS-232 serial port and communications software to go with it. Figure 2-1 shows the basic setup.

With an 8052-BASIC circuit connected by a serial link to a personal computer, you have a complete development system with these abilities:

- You can write and run BASIC programs. You use the keyboard, video display, and other resources of the personal computer to type and view the programs and commands that the 8052-BASIC system executes. BASIC-52 is an interpreted language whose programs do not require an additional assembling or compiling step. You can run programs or execute commands immediately after you write them.

- You can use BASIC-52's programming functions to permanently store your programs in EPROM or other nonvolatile memory. You don't need a separate EPROM programmer.

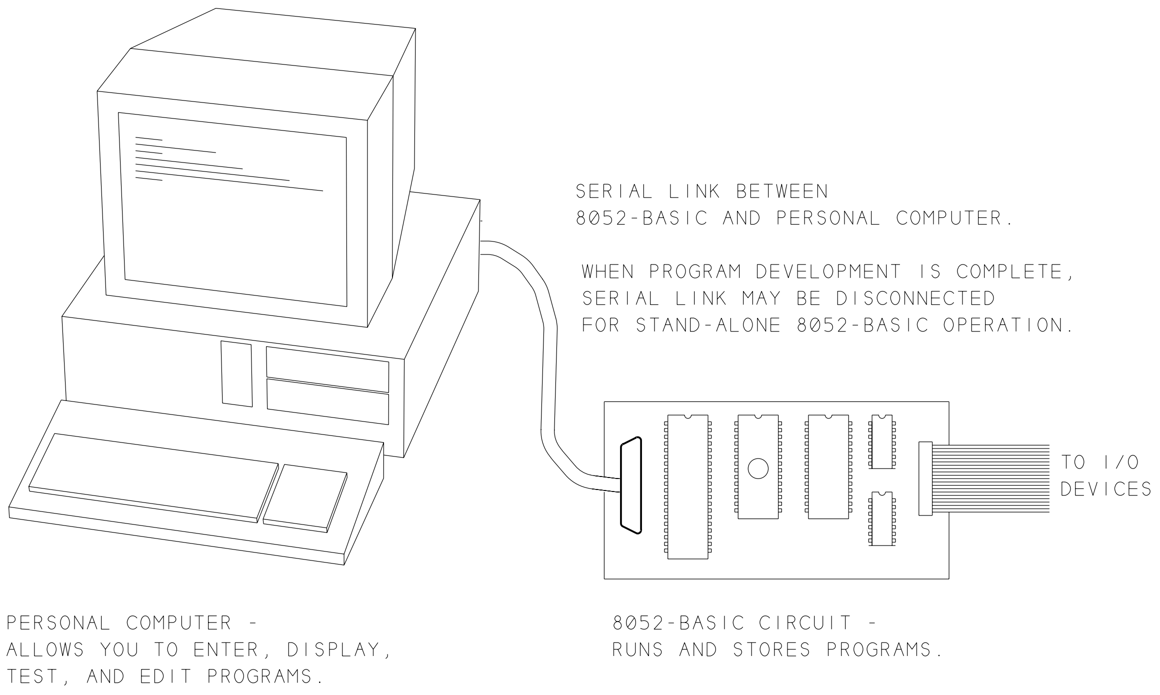
SERIAL LINK BETWEEN
8052-BASIC AND PERSONAL COMPUTER.
WHEN PROGRAM DEVELOPMENT IS COMPLETE,
SERIAL LINK MAY BE DISCONNECTED
FOR STAND-ALONE 8052-BASIC OPERATION.
TO I/O
DEVICES
PERSONAL COMPUTER -
ALLOWS YOU TO ENTER, DISPLAY,
TEST, AND EDIT PROGRAMS.
8052-BASIC CIRCUIT -
RUNS AND STORES PROGRAMS.

**Figure 2-1** Setup for working with the 8052-BASIC.

- You can also store programs on your personal computer’s disk. You can write or edit programs on your personal computer, and then upload them to the 8052-BASIC system.

- To the basic circuits, you can add displays, switches, keypads, relays, and other components, depending on the needs of your project.

- After program development, you can disconnect the link to the personal computer and let the 8052-BASIC system run its stored program on its own.

## Limits

No single product is ideal for every use. These are some of the limitations to the 8052-BASIC:

- Program execution can be slow, compared with programs that run on more powerful computers, or programs written in assembly language. A typical program line in BASIC-52 takes several milliseconds to execute. Because of this, there are some tasks that BASIC-52 just can’t handle—for example, detecting and responding to an interrupt within a few microseconds. But for many control, monitoring, and other tasks, BASIC-52 is fine. For example, a weather station that senses conditions once per minute and stores or displays the results doesn’t need super-fast response. And, if necessary, you can call an assembly-language routine for a portion of a program where speed is critical.

  Even if you write your programs in assembly language, C, or another language, you can use the 8052-BASIC system as a development system that enables you to upload your program to memory, run the program, and test and debug your programs and circuits.

- Another limitation of the 8052-BASIC is that a complete project requires additional components. If you’re looking for a true single-chip solution, the 8052-BASIC isn’t it. Even a minimal system requires an external RAM chip, and most systems also have an external EPROM or other non-volatile memory. The serial link and other optional functions also use some of the on-chip timers and input/output ports, so these may not be available for other uses.

  Still, the 8052-BASIC lets you to do a lot with a little. When needed, you can easily add chips to expand the input/output ports, timers, and other functions.

- And finally, don’t expect BASIC-52 to have the abilities of *QBasic, Visual Basic* or other BASIC programming languages that you may use on your personal computer. BASIC-52 is more capable than many other single-chip BASICs. It includes features like loops, subroutines, string handling, and even floating-point math for handling fractional quantities. But there are some primitive aspects to the language. For example, the on-line editing functions are limited. Once you write a program line, you can change it only by retyping from the beginning. The limitations are understandable, because the entire programming language has to fit in the 8052's 8 kilobytes of ROM. Fancy editing and other features just aren't feasible in this small space.

There are solutions here as well. You can get around many of the editing limitations by writing and editing programs off-line, using your personal computer and text editor, and then uploading to the 8052-BASIC system. And, there are software and hardware products that enhance BASIC-52 and make it easier to use, especially for longer, more complex programming jobs.

## What You Need

To use the 8052-BASIC chip, you need the following equipment, materials, and skills:

### Components

The 8052-BASIC chip and supporting components are widely available. Appendix A lists sources for the components used in the circuits described in this book.

### Power Supply

You'll need a regulated +5-volt power supply to power the circuits. Output capability of at least 500 milliamperes is recommended for general experimenting. The power supply can be powered by batteries or AC line voltage, but it must have a regulated output between 4.75 and 5.25 volts.

### Construction Materials

To build the circuits, you'll need circuit-construction materials and the skills to use them. Wire-wrapping is an effective, quick way to build the circuits described, but if you prefer, you can use point-to-point soldering or design and make a printed-circuit board, or use any method that you're comfortable with. Another option is to buy one of the available kits or prebuilt 8052-BASIC boards. You can then use this book as a guide to using and expanding the abilities of your board. Appendix A lists board suppliers and books on project-construction techniques.

### Documentation

Using just the information in this book, you can build and begin using your system. For serious experimenting, two additional references are recommended: programming and hardware manuals. For programming, you have two choices: Intel's *BASIC-52 User's Manual*, or Systronix's *BASIC-52 Programming*. Each of these describes the BASIC-52 programming language in detail. The Intel manual includes a few schematics, while Systronix's version has more programming examples and is better organized in general. Intel's *Embedded Microcontrollers* data book is a hardware reference that describes the 8052 chip, including electrical specifications and timing requirements. It also includes an assembly-language reference. Appendix A tells where to get these.

Other useful documentation includes data sheets for the other components in your projects. For a small charge, many component vendors will send along data sheets for the parts you order.

## Host Computer

To program the 8052-BASIC, you connect its circuits to a host computer, using an RS-232 asynchronous serial port and terminal-emulation software. The computer can be any type, as long as it has a serial port and appropriate software.

The serial port is the same connector where you plug in an external modem, serial printer, serial mouse, or other RS-232 serial device.

Terminal-emulation software is the same type of software that you may use for modem communications with an on-line BBS. Examples for MS-DOS computers are Datastorm

**Table 2-1.** Differences among 8051-family chips.

| Chip | Program Memory | | Ram (bytes) | Timers |
|---|---|---|---|---|
| | Type | kilobytes | | |
| 8051 | ROM | 4 | 128 | 2 |
| 8052 | ROM | 8 | 256 | 3 |
| 8031 | none | - | 128 | 2 |
| 8032 | none | - | 256 | 3 |
| 8751 | EPROM | 4 | 128 | 2 |
| 8752 | EPROM | 8 | 256 | 3 |

- 80C51, 80C52, 80C31, and so on are CMOS versions of above.
- 80C51FA/B/C add more versatile timers and an enhanced serial channel.
- 8052-BASIC has the BASIC-52 programming language in ROM.
- Packages include 40-pin DIP, 40-lead PLCC, and 44-pin QFP.

Technologies' *Procomm Plus* and the Terminal accessory in Microsoft Windows. At minimum, the software must enable you to do the following: set the baud rate and other communications parameters, serially transmit the characters that you type at the keyboard, and display the characters received at the serial port. Also useful, but not essential, is the ability to upload and download text files from your disk, over the serial link. If you don't have a favorite communications program, look in shareware catalogs or the file areas of online services or BBS's, where you can try out the offerings for a small disk-copying or downloading charge.

### Test Equipment

Some basic test equipment will help you monitor, test, and troubleshoot your circuits. Minimum requirements include a multimeter capable of reading volts, ohms, and milliamperes. Just about any basic meter will do for this. A logic probe is convenient, but not essential, for monitoring logic levels and transitions. Best of all, an oscilloscope lets you view the actual waveforms on one or more channels.

### Knowledge

This book assumes that you have a basic knowledge of electronic circuits, including digital logic. It does not assume that you know a lot about computer programming and computer circuits. Appendix A lists some books that cover the basics, if you want to review or learn these. Appendix C is a review of hexadecimal, binary, and decimal number systems.

## The 8051 Family

At the core of the 8052-BASIC is an 8052 microcontroller, a member of the 8051 microcontroller family. Intel Corporation introduced the 8051 in 1980. Since that time, 8051-family chips have been used as the base of thousands of products. Many other companies, including Philips, Siemens, Dallas Semiconductor, OKI, Fujitsu, and Harris-Matra now also make 8051-family chips. Some companies have expanded the 8051 family by offering compatible chips with additional features.

Table 2-1 summarizes the differences among popular 8051-family chips. The 8052 is an enhanced 8051, with an extra timer and more RAM and ROM. The 8031 and 8032 are identical to the 8051 and 8052, except that the ROM area is unused, and program code must be stored in an external EPROM or other memory chip.

The 8052, like other 8051-family chips, is available in NMOS and CMOS versions. Figure 2-2 shows the pinout of the 8052 and 8052-BASIC, and Table 2-2 describes the pin functions.

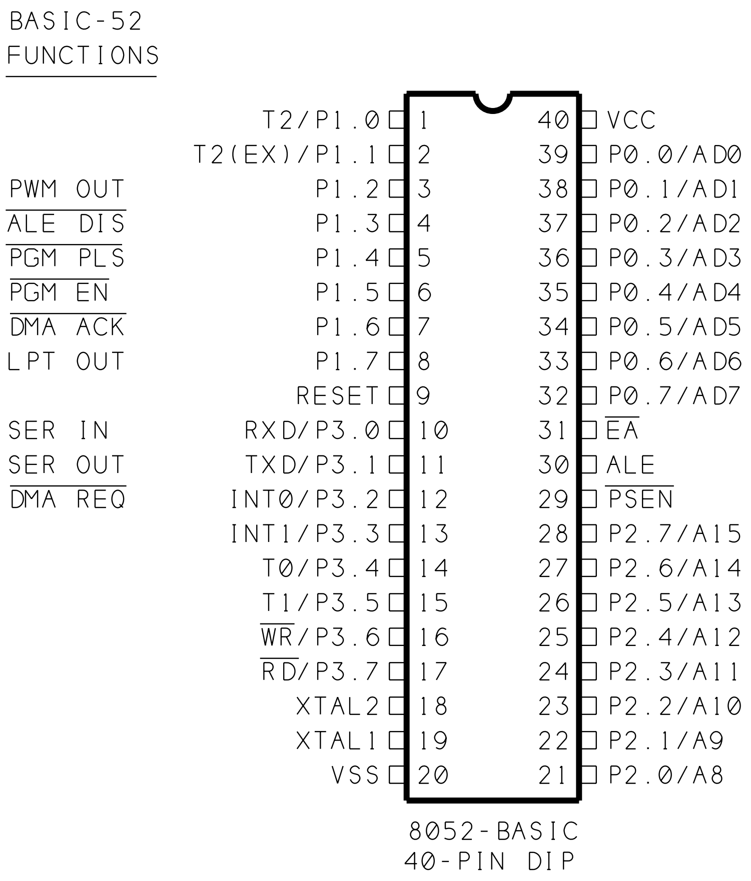

**Figure 2-2** Pin functions of the 8052 and 8052-BASIC microcontrollers.

## Elements of the 8052 and 8052-BASIC

These are the major elements of the 8052, plus the enhancements included in the 8052-BASIC:

### CPU

The CPU, or central processing unit, executes program instructions. Types of instructions include arithmetic (addition, subtraction), logic (AND, OR, NOT), data transfer (move), and program branching (jump) operations. An external crystal provides a timing reference for clocking the CPU.

### ROM

ROM (read-only memory) is the read-only memory that is programmed into the chip in the manufacturing process. In the 8052-BASIC, the ROM contains the BASIC-52 interpreter program that the 8052 executes on boot-up. As far as the hardware is concerned, this is the only difference between the ordinary 8052 and the 8052-BASIC.

**Table 2-2. (page 1 of 2)** Pin functions of the 8052 microcontroller and 8052-BASIC additions.

| Pin | Symbol | Input/ Output | 8052 Function | 8052-BASIC Additions | |
|---|---|---|---|---|---|
| | | | | Symbol | Function |
| 1 | P1.0<br>T2 | I/O | Port 1, bit 0;<br>Timer 2 external input | | |
| 2 | P1.1<br>T2(EX) | I/O | Port 1, bit 1;<br>Timer 2 external reload/capture | | |
| 3 | P1.2 | I/O | Port 1, bit 2 | PWM | Pulse-width-modulated output |
| 4 | P1.3 | I/O | Port 1, bit 3 | $\overline{\text{ALE DIS}}$ | Address latch disable |
| 5 | P1.4 | I/O | Port 1, bit 4 | $\overline{\text{PGM PLS}}$ | Program pulse |
| 6 | P1.5 | I/O | Port 1, bit 5 | $\overline{\text{PGM}}$ $\overline{\text{EN}}$ | Programming voltage enable |
| 7 | P1.6 | I/O | Port 1, bit 6 | $\overline{\text{DMA ACK}}$ | DMA acknowledge |
| 8 | P1.7 | I/O | Port 1, bit 7 | LPT | Line printer out |
| 9 | Reset | Input | Reset system | | |
| 10 | P3.0<br>RXD | I/O | Port 3, bit 0<br>Serial receive | SER IN | Serial port in |
| 11 | P3.1<br>TXD | I/O | Port 3, bit 1<br>Serial transmit | SER OUT | Serial port out |
| 12 | P3.2<br>$\overline{\text{INT0}}$ | I/O | Port 3, bit 2<br>External interrupt 0 | $\overline{\text{DMA}}$<br>$\overline{\text{REQ}}$ | DMA request |
| 13 | P3.3<br>$\overline{\text{INT1}}$ | I/O | Port 3, bit 3<br>External interrupt 1 | | |
| 14 | P3.4<br>T0 | I/O | Port 3, bit 4<br>Timer 0 external input | | |
| 15 | P3.5<br>T1 | I/O | Port 3, bit 5<br>Timer 1 external input | | |
| 16 | P3.6<br>$\overline{\text{WR}}$ | I/O | Port 3, bit 6<br>Write strobe for external memory | | |
| 17 | P3.7<br>$\overline{\text{RD}}$ | I/O | Port 3, bit 7<br>Read strobe for external memory | | |
| 18 | XTAL1 | Input | Inverting oscillator amplifier (crystal) | | |
| 19 | XTAL2 | Output | Inverting oscillator amplifier (crystal) | | |
| 20 | VSS | Input | Circuit ground | | |

**Table 2-2. (page 2 of 2)**

| Pin | Symbol | Input/ Output | 8052 Function | 8052-BASIC Additions (none on pins 21-40) |
|---|---|---|---|---|
| 21 | P2.0<br>A8 | I/O | Port 2, bit 0<br>Address bit 8 | |
| 22 | P2.1<br>A9 | I/O | Port 2, bit 1<br>Address bit 9 | |
| 23 | P2.2<br>A10 | I/O | Port 2, bit 2<br>Address bit 10 | |
| 24 | P2.3<br>A11 | I/O | Port 2, bit 3<br>Address bit 11 | |
| 25 | P2.4<br>A12 | I/O | Port 2, bit 4<br>Address bit 12 | |
| 26 | P2.5<br>A13 | I/O | Port 2, bit 5<br>Address bit 13 | |
| 27 | P2.6<br>A14 | I/O | Port 2, bit 6<br>Address bit 14 | |
| 28 | P2.7<br>A15 | I/O | Port 2, bit 7<br>Address bit 15 | |
| 29 | $\overline{\text{PSEN}}$ | Output | Program store enable<br>Read strobe for external program memory | |
| 30 | ALE | Output | Address latch enable | |
| 31 | $\overline{\text{EA}}$ | Input | External access enable for program memory | |
| 32 | P0.7<br>AD7 | I/O | Port 0, bit 7<br>Address/data bit 7 | |
| 33 | P0.6<br>AD6 | I/O | Port 0, bit 6<br>Address/data bit 6 | |
| 34 | P0.5<br>AD5 | I/O | Port 0, bit 5<br>Address/data bit 5 | |
| 35 | P0.4<br>AD4 | I/O | Port 0, bit 4<br>Address/data bit 4 | |
| 36 | P0.3<br>AD3 | I/O | Port 0, bit 3<br>Address/data bit 3 | |
| 37 | P0.2<br>AD2 | I/O | Port 0, bit 2<br>Address/data bit 2 | |
| 38 | P0.1<br>AD1 | I//O | Port 0, bit 1<br>Address/data bit 1 | |
| 39 | P0.0<br>AD0 | I/O | Port 0, bit 0<br>Address/data bit 0 | |
| 40 | Vcc | Input | Supply voltage | |

## RAM

RAM (random-access memory) is where programs store information for temporary use. Unlike ROM, the CPU can write to RAM as well as read it. Any information stored in RAM is lost when power is removed from the chip. The 8052 has 256 bytes of RAM. BASIC-52 uses much of this for its own operations, with a few bytes available to users.

## I/O Ports

I/O (Input/Output) Ports enable the 8052 to read and write to external memory and other components. The 8052 has four 8-bit I/O ports (Ports 0-3). As the name suggests, the ports can act as inputs (to be read) or outputs (to be written to). Many of the port bits have optional, alternate functions relating to accessing external memory, using the on-chip timer/counters, detecting external interrupts, and handling serial communications. BASIC-52 assigns alternate functions to the remaining port bits. Some of these functions are required by BASIC-52, while others are optional. If you don't use an alternate function, you can use the bit for any control, monitoring, or other purpose in your application.

**Accessing external memory.** The largest alternate use of the ports has to do with accessing external memory. Although the 8052 is a single-chip computer, a complete 8052-BASIC system requires additional components. It must have external RAM in addition to the 8052's internal RAM, and most systems also have EPROM, EEPROM, or battery-backed RAM for permanent storage of BASIC-52 programs.

Accessing this external memory uses all of Ports 0 and 2, plus bits 6 and 7 of Port 3, to hold data, addresses, and control signals for reading and writing to external memory. Data here refers to a byte to be read or written, and may be any type of information, including program code. The address defines the location in memory to be read or written.

During a memory access, Port 0's eight pins (AD0-AD7) first hold the lower byte of the address, followed by the data to be read or written. This method of carrying both addresses and data on the same signal lines is called a *multiplexed address/data bus*. It's a popular arrangement that many devices use, since it requires fewer pins on the chip, compared to giving each data and address line its own pin. Port 2's eight lines hold the higher byte of the address to be read or written to. These lines make up the high address bus (A8-A15). Together, the 16 address lines can access 64 kilobytes (65,536 bytes) of memory, from 00000000 00000000 to 11111111 11111111 in binary, or 0000h to FFFFh in hexadecimal.

Besides pins to hold the data and addresses, the 8052 must also provide control signals to initiate the read and write operations. Control signals include $\overline{\text{WR}}$ (write), $\overline{\text{RD}}$ (read), $\overline{\text{PSEN}}$ (program store enable), and ALE (address latch enable). Some of the address lines may also function as control signals that help to select a chip during a memory access.

**Code and data memory.** To understand the operation of the control signals, you need to know a little about how the 8052 distinguishes between two types of memory: data and code, or program, memory. By using different control signals for each type of memory, the 8052 can access two separate 64K areas of memory, with each addressed from 0000h to FFFFh, and each using the same data and address lines.

The 8052 accesses code memory when it executes an assembly-language program or subroutine. Code memory is read-only; you can't write to it. The only instructions that access code memory are read operations. Code memory is intended for programs or subroutines that have been previously programmed into ROM or EPROM. The 8052 strobes, or pulses, $\overline{\mathrm{PSEN}}$ when it accesses external code memory. Accesses to internal code memory (the BASIC-52 interpreter in ROM) do not use $\overline{\mathrm{PSEN}}$ or any external control signals.

Data memory is read/write memory, usually RAM. Instructions that read data memory strobe $\overline{\mathrm{RD}}$, and instructions that write to data memory strobe $\overline{\mathrm{WR}}$. The term*data memory* may be misleading, because it can hold any information that is accessed with instructions that strobe $\overline{\mathrm{RD}}$ or $\overline{\mathrm{WR}}$. In fact, BASIC-52 programs are stored in data memory, not code memory as you might think. This is because the 8052 does not execute the BASIC programs directly. Instead, the BASIC-52 interpreter program reads the BASIC programs as data and then translates them to machine code for execution by the 8052.

If you don't need all of the available memory space, you can combine code and data memory in a single area. With combined memory, $\overline{\mathrm{WR}}$ controls write operations, and $\overline{\mathrm{PSEN}}$ and $\overline{\mathrm{RD}}$ are logically ANDed to create a read signal that is active when either $\overline{\mathrm{PSEN}}$ or $\overline{\mathrm{RD}}$ is low. Combined data/code memory is handy if you want the flexibility to store either BASIC or assembly-language programs in the same chip, or if you want to be able to upload assembly-language routines into RAM for testing.

ALE is the final control signal for accessing external memory. It controls an external latch that stores the lower address byte during memory accesses. When the 8052 reads or writes to external memory, it places the lower address byte on AD0-AD7 and strobes ALE, which causes the external latch to save the lower address byte for the rest of the read or write cycle. After a short delay, the 8052 replaces the address on AD0-AD7 with the data to be written or read.

**Timers and Counters.** The 8052 has three 16-bit timer/counters, which make it easy to generate periodic signals or count signal transitions. BASIC-52 assigns optional functions for each of the timer/counters.

Timer 0 controls a real-time clock that increments every 5 milliseconds. You can use this clock to time events that occur at regular intervals, or as the base for clock or calendar functions. Timer 1 has several uses in BASIC-52, including controlling a pulse-width-modulated output (PWM) (a series of pulses of programmable width and number); writing to a line printer or other serial peripheral (LPT); and generating pulses for EPROM programming ($\overline{\text{PGM PULSE}}$). Timer 2 generates a baud rate for serial communications at SER IN and SER OUT. These are all typical applications for timer/counters in microcontroller circuits.

If you don't use the optional timer functions, you can program the timers for other applications. In addition to timing functions, where the timer increments at a defined rate, you can use the timers for event counting, where the timer increments on an external trigger and measures the time between triggers. If you use the timers for event counting, T2, T2(EX), T0, and T1 detect transitions to be counted.

**The serial port.** The 8052's serial port automatically takes care of many of the details of serial communications. On the transmit side, the serial port translates bytes to be sent into serial data, including adding start and stop bits and writing the data in a timed sequence to SER OUT. On the receive side, the serial port accepts serial data at SER IN and sets a flag to indicate that a byte has been received. BASIC-52 uses the serial port for communicating with a host computer.

**External interrupts.** $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are external interrupt inputs, which detect logic levels or transitions that interrupt the CPU and cause it to branch to a predefined program location. BASIC-52 uses $\overline{\text{INT0}}$ for its optional direct-memory-access (DMA) function.

**Programming functions.** BASIC-52's programming commands use three additional port bits ($\overline{\text{ALEDIS}}$, $\overline{\text{PGM PULSE}}$, and $\overline{\text{PGM EN}}$) to control programming voltages and timing for storing BASIC-52 programs in EPROM or other nonvolatile memory.

## Additional Control Inputs

Two additional control inputs need to be mentioned. A logic high on RESET resets the chip and causes it to begin executing the program that begins at 0 in code memory. In the 8052-BASIC chip, this program is the BASIC-52 interpreter. $\overline{\text{EA}}$ (external memory access) determines whether the chip will access internal or external code memory in the area from 0 to 1FFFh. In BASIC-52 systems, $\overline{\text{EA}}$ is tied high so that the chip runs the BASIC interpreter in internal ROM on boot-up.

## Power Supply Connections

And, finally, the chip has two pins for connecting to a +5-volt DC power supply (VCC) and ground (VSS).

That finishes our tour of the 8052-BASIC chip. We're now ready to put together a working system.