Application By Jan Axelson

Experimenting With the Z8 Single-Chip Microcomputer

It may not be the latest microcomputer chip, but it has plenty of life in it to get specialized jobs done

It's easy to forget that you don't have to use the latest technology in every project. However enticing the newest technological wonders may be, they often aren't required, or even the best suited, for a particular task. Something that's been around a while may be simpler to use, and cheaper as well. In the computer world, the Z8 is a single-chip microcomputer that has been in use for over a decade—a long time in terms of computer years—but still has plenty of life in it.

Zilog, Inc., probably best known for its popular Z80 microprocessor. manufactures the Z8. In contrast to the Z80, the Z8 is a complete single-chip microcomputer, with memory and bit-programmable I/O ports, as well as counter/timers and a UART (universal asynchronous receiver/transmitter) for serial communication. A special version of the chip, the Z8671, contains a BASIC interpreter in ROM for easy program development.

In this article, we'll explore the world of the Z8671 and other Z8s, including their architecture, interfacing, programming and resources for Z8 system design and programming.

About the Z8

The Z8 is especially suited for use in dedicated devices in which a program is embedded in EPROM or other memory device and, along with support circuitry, executes a single function or group of related functions. Possible Z8 projects include data-acquisition systems, motor controllers, programmable pulse or other waveform generators, electronic games, automotive applications, telephone switchers, intelligent instruments, process control and anywhere you might use a single-chip microcomputer or microcontroller.

The Z8 comes in many versions, ranging from ROM-less units to the Super8 series with expanded features and instruction set. Our focus here is on the Z8671, but the main features, with the exception of BASIC/Debug capabilities, are the same as or similar to those in other versions. At less than $10, the Z8671 offers a low-cost introduction to single-chip microcomputing.

For more on the Z8, Zilog publishes a *Z8 Family Design Book* that's several books in one, including data book, technical manual, application-note series, programmer's guide and

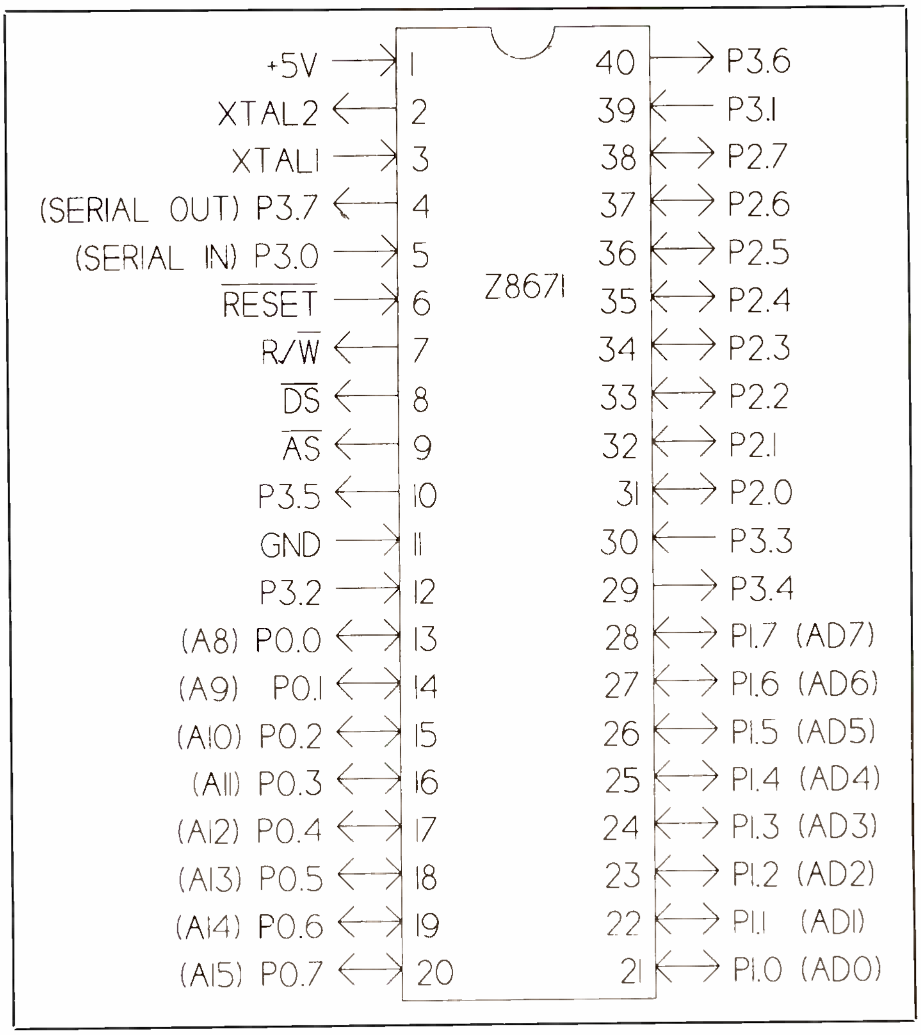


Fig. 1. Pinout of the Z8671 single-chip microcomputer with BASIC/Debug programmed into on-bound ROM.

subroutine library. A separate *Z8671 BASIC/Debug Reference Manual* describes BASIC/Debug. See the Sources box elsewhere in this article for more on these and other resources mentioned.

The Z8671's BASIC/Debug is a bare-bones programming language, with just 15 keywords. Still, it has enough capability for writing many types of programs. In addition, you can use it as a simple debugger for testing programs in BASIC or assembly language. BASIC/Debug makes it easy to examine and modify on-chip and external memory and to load and run programs written in assembly language.

Figure 1 gives the pinout of the Z8671. This IC can address up to 124 kilobytes of external memory: 62K each of data and program memory. An additional 2K of internal ROM contains BASIC/Debug. For smaller systems, the Data Memory Select (DM) control signal can be ignored, for a maximum 62K of combined data and program memory.

Two of the Z8671's four eight-bit I/O ports are dedicated to external memory access. Port 1 is a multiplexed address/data bus, Port 0 is the high-address bus for accessing external memory.

Two of Port 3's pins provide the serial interface to the UART in the Z8671, allowing communication with other serial interfaces, such as an RS-232 port on a PC.

The remaining port pins can be used as desired. Those pins with alternate functions (interrupt request, timer input and output, etc.) can be used for general-purpose input/output applications if the alternate functions aren't needed.

A Development System

Shown in Fig. 2 is a basic system that can be used for experimenting with the Z8671. I built this circuit with Wire Wrap hardware on perforated board. The circuit could also be built using point-to-point wiring, or by designing and making a printed-circuit board for it.

If you'd rather not build your own development system from scratch, the Sources box lists manufacturers of assembled and tested boards that contain a Z8, memory, serial interface and other circuit elements to use as a base on which to build your designs and experiments.

The Fig. 2 circuit contains a Z8671 with BASIC/Debug; 2K of RAM; a socket for a 2-kilobyte EPROM, EEPROM, or nonvolatile RAM; a serial interface for communicating with a desktop computer or terminal; 14 free port pins; and 34K of free memory area for connecting to additional memory devices or other circuit elements.

The circuit's serial interface connects to an RS-232 serial port on a desktop, or host, computer or terminal. By running a communications program on the host computer, you can communicate with the Z8, write and run programs in BASIC, upload BASIC and assembly-language programs from your host computer to the Z8 and download programs from the Z8 to the host.

The crystal frequency of 7.3728 MHz divides down for accurate baud rates for serial communication (according to the formula given in the Z8 manual).

The two memory ICs are RAM and either EPROM, EEPROM or

Professional Tool Case

SPECIAL
$59.95
Reg. ~~$80.00~~

Model ARIM5

- **Two removable pallets hold over 60 tools**
- **Case top has built-in document holder**
- **Case bottom is partitioned into 3 areas**

A handsome black case to organize and transport your valuable tools and instruments. This is the same quality case used by literally thousands of professional field engineers. Case is made of high impact polypropylene, and has snap-action key locks and a padded handle.
Size: 17½″ × 12½″ × 5″.
Tools are not included.
Offer expires December 31, 1991.

To order call:
1-800-225-5370
In MA: (508) 682-2000
Same Day Shipment!
Money Back Guarantee!
Terms: Visa, MC, Amex:
P.O.'s from qualified firms accepted.
Add $6.75 for packing and delivery.

FREE CONTACT EAST CATALOG
Contains thousands of products for testing, repairing & assembling electronic equipment. To get your free catalog, call **(508)682-2000.**

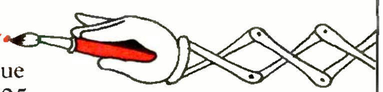
CIRCLE NO. 62 ON FREE INFORMATION CARD

What 8051 compiler would Santa use to automate his workshop?

BCI51™ of course - even Santa knows BASIC!

BCI51™ BASIC cross compiler with integer math • Supplement BASIC-52 or stand-alone • Now includes 'C51F support • BCI51 with assembler and utilities $299 • Powerful Dallas DS5000T extensions $149 • Assembly Language Programmer's Toolkit $99 • superb documentation

Systronix Inc.



754 East Roosevelt Avenue
Salt Lake City, Utah 84105
801-487-7412 FAX: 801-487-3130

CIRCLE NO. 47 ON FREE INFORMATION CARD

battery-backed (nonvolatile) RAM for permanent program storage. The memory interface is typical of many other eight-bit systems. For each byte transferred between external memory and the Z8, the lower eight bits of the address are latched to the memory IC through an 74LS373 transparent latch, and the higher address bits and data bits interface directly to the memory ICs. The Z8's Read/Write (R/W) and Data Strobe (DS) control signals interface to the memory ICs' Output Enable (OE) and Write Enable (WE) pins.

On power-up, BASIC/Debug selects the Z8's slower extended-bus timing mode. So access time of the memory chips isn't critical.

The two halves of a 74LS139 2-to-4-line decoder provide address decoding, which selects and enables the memory ICs, and a baud-rate selector. Unused outputs of the 74LS139 can be used to select additional devices at specific addresses in the system.

The RAM is mapped from 800h to FFFh (h = hexadecimal). Any memory reads or writes to addresses in this range will access the RAM. The EPROM/EEPROM/NVRAM socket is mapped from 1000h to 17FFh. An EEPROM or NVRAM in this socket can be write-protected by jumpering its WE line to +5 volts. Addresses from 0 to 7FFh aren't used in external memory because this is where the internal ROM for the Z8 is located.

A baud-rate selector is provided by three 74LS125 tri-state buffers. When BASIC/Debug boots, it examines memory location FFFDh for a baud-rate setting to use for serial communication. In the Fig. 2 circuit, all memory accesses from C000h to FFFFh will access the baud-rate selector. (To free up portions of this memory area, additional address decoding could be added.) Jumpers or toggle or slide switches set desired baud rate, or you can hard-wire in a single rate, if you prefer.

Free areas of memory include a 2K block at 1800h, and 16K blocks at 4000h and 8000h. These can be used to access additional memory or other components.

The final circuit element is the serial interface, provided by the popular MAX232 chip. This chip converts the TTL-level outputs of the Z8671 to valid RS-232 transmit levels, and in the other direction converts received RS-232 signals to TTL levels for input to the Z8671.

The serial interface is required for program development when using BASIC/Debug. For final projects that don't use the interface, the MAX232 can be left out.

The circuit in Fig. 2 is powered by a regulated +5-volt supply. A 0.5-ampere supply is more than adequate.

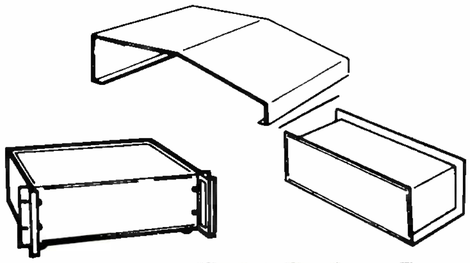
To use the system, you connect its serial interface to a serial connector on the host computer (usually a 25- or nine-pin male subminiature D-type connector). Figure 2 shows the pin connections for a typical 25-pin male D connector. For a nine-pin male D connector, the pinout is typically as follows: pin 2—receive (data in); pin 3—transmit (data out); and pin 5—ground.

Since RS-232 connections are notoriously unpredictable, always verify that your wiring is correct. Check the pinout on the host computer's serial connector to verify that its data output connects to an input on the MAX232 and that the MAX232's

Project Pro

Call For **FREE** Catalog
Call (800) 800-3321 or (216) 425-8888

Project Pro brings a complete line of electronic enclosures, hardware, and tools to both the professional project engineer and the hobbiest.
Everything you need to bring ANY project from development to finished product.

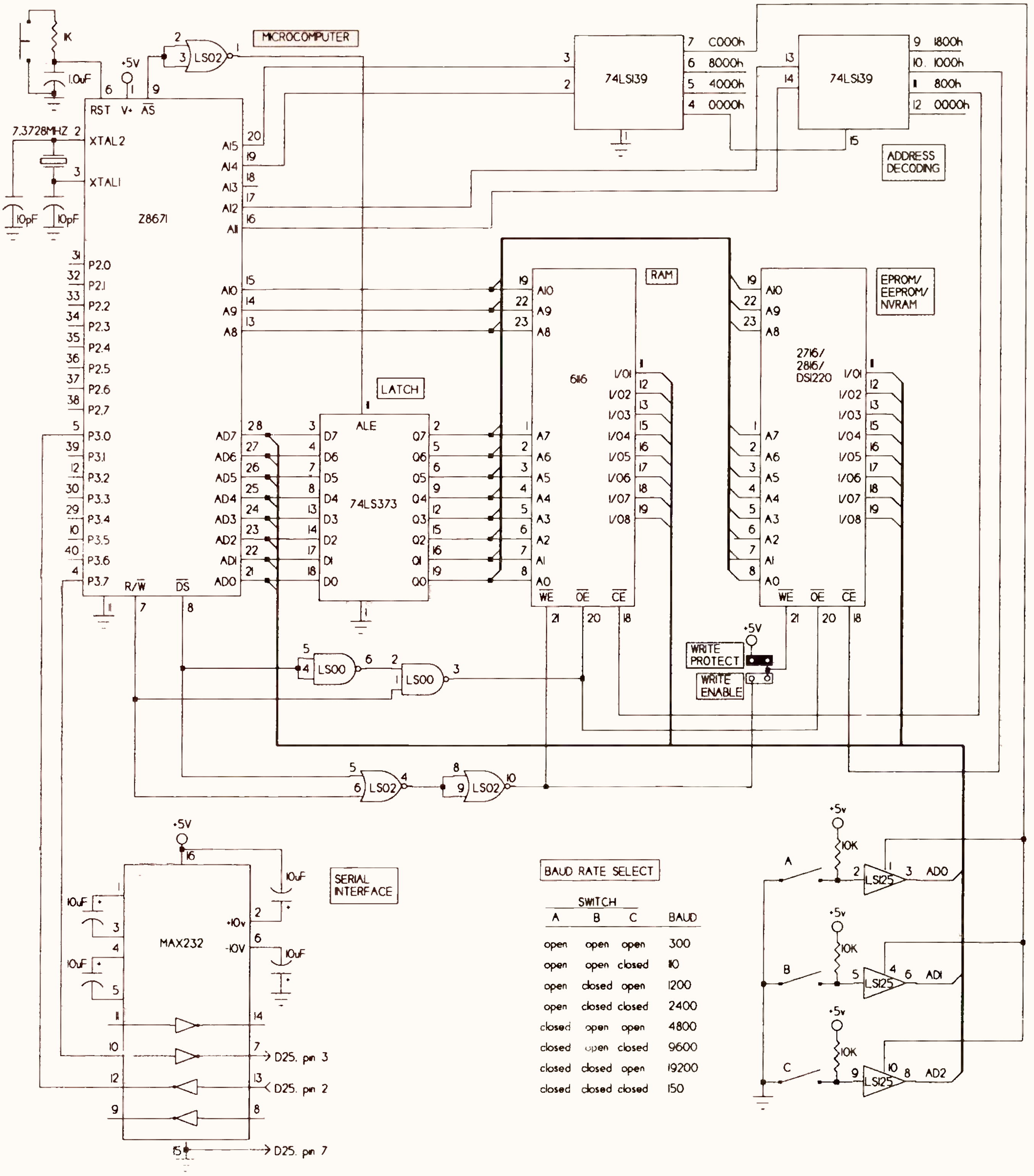


Electronic Enclosures	Custom Services	Fasteners
* Aluminum	* Fabricating	*Nickel Plated
* Steel	* Screening	*Black Zinc
* Plastic (ABS)	* 72 hour-turnaround	
Racks	Chassis	Tools

1710 Enterprise Pky. Twinsburg, Ohio 44087

CIRCLE NO. 70 ON FREE INFORMATION CARD

**Sorry, sir, the computers are busy—
I'll have to help you in person.**



BAUD RATE SELECT

A	B	C	BAUD
open	open	open	300
open	open	closed	110
open	closed	open	1200
open	closed	closed	2400
closed	open	open	4800
closed	open	closed	9600
closed	closed	open	19200
closed	closed	closed	150

Fig. 2. A development system for the Z8671, including RAM, nonvolatile memory and serial interface.

output connects to an input on the host computer.

To communicate with the Z8671, run a communications program (*Procomm, Kermit* or whatever other one you prefer) on the host, setting the communications protocol for the baud rate you've selected on the Z8671 board, with eight data bits, no parity and one or two stop bits. (The Z8 adds two stop bits to transmitted data, and received data must have at least one stop bit.)

Using BASIC/Debug

When the Z8671 powers up or is reset, the ":" BASIC/Debug prompt appears on the host computer's screen. From here, you can write and run programs in BASIC, execute statements in immediate mode and upload and download programs.

Figure 3 shows BASIC/Debug's keywords and operators. If you're used to QuickBASIC, or even GW BASIC, BASIC/Debug will seem primitive. You won't find control structures like WHILE . . . WEND or even FOR . . . NEXT loops. Only integer variables and calculations are allowed, and 26 variables—from A to Z—are the maximum.

BASIC/Debug does, however, permit you to write and execute many simple programs quickly. For its intended use, the elaborate text-formatting, graphics and similar capabilities found in other BASICs aren't needed. Listing 1 shows an example BASIC/Debug program that prompts for a memory location and then prints the contents of the 16 bytes beginning at that location.

In BASIC/Debug, reading from and writing to external and internal memory, including ports and other registers, uses the signal character @, rather than BASIC's usual PEEK and POKE statements. For example, the statement PRINT @2048 causes the contents of memory location 2048 to be displayed on the host computer's screen. A % prefix indicates hexadecimal. Therefore, the statement PRINT @%800 gives the same result as the previous example.

The statement @%900@%F8 writes the value F8h to location 900h.

At times, you may want to examine or modify the value of a 16-bit word, rather than an eight-bit byte.

GO	Unconditionally branches to a machine-language subroutine.
GOSUB	Unconditionally branches to a subroutine specified by line number.
GOTO	Unconditionally branches to a line number.
IF/THEN	Initiates a conditional operation or branch.
LET	Assigns the value of an expression to a variable or memory location.
INPUT/IN	Requests information from the user with "?" prompt, then reads input values (separated by commas) from the keyboard and stores the values in the indicated variables. INPUT discards values remaining from previous IN, INPUT, or RUN statements. IN uses values left in the buffer, then requests new data.
LIST	Displays program listing.
NEW	Resets R4-R5, indicating that RAM is ready to store a new program.
PRINT/ PRINT HEX	Displays text messages or numeric values. PRINT HEX displays values in hexadecimal.
REM	Indicates unexecuted comment or remark.
RETURN	Ends a subroutine by returning control to the line following a GOSUB.
RUN	Causes the current program to execute.
STOP	Ends program execution and clears GOSUB stack.
USR	Unconditionally branches to a machine-language subroutine. Can pass and return up to two variables.
+	Addition
-	Subtraction
*	Multiplication
/	Signed division (range: -32768 to +32767)
\	Unsigned division (range: 0 to 65535)
=	equal
<=	less than or equal
<	less than
<>	not equal
>	greater than
>=	greater than or equal
AND	logical AND
%	hexadecimal (otherwise decimal)
@	indirect byte address
^	indirect word address

Fig. 3. Keywords, operators and special characters available in BASIC/Debug.

For example, BASIC/Debug stores the address of the first location in external RAM in Registers 8 and 9. To determine this value, you could read both registers and add their weighted values, but there's an easier way. The signal character " " references a 16-bit word consisting of the specified byte and the one following it. So the statement PRINT 8 displays the 16-bit value (from 0 to 65,535) stored in Registers 8 and 9.

Although BASIC/Debug has no FOR . . . NEXT loops, you can accomplish the same thing by using an index, or count, variable, an IF . . . THEN statement that tests the value of the index variable and calls a subroutine if the IF statement is true. Listing 1 uses this technique to step through the 16 values it displays.

BASIC/Debug indicates syntax and other errors by number only. The numeric codes are explained in the BASIC/Debug manual; so at first you'll want to keep this handy as you program. You'll soon memorize the codes that pop up often.

BASIC/Debug's line editor lets you backspace to correct typing mistakes, but once you press RETURN, the entire line must be typed again to make a change. Instead of entering a long, involved program from within BASIC/Debug, you can write a program with a text editor that produces pure ASCII output, and then use your communications software to upload the program to the Z8 system's memory. In the other direction, downloading a program to the host computer is an easy way to save your code for re-loading.

Most communications programs include functions for uploading and downloading files. These are the same functions used for sending and receiving files to and from a BBS, and the procedure is similar when transferring files from and to the Z8671. Select ASCII format for the transfer, and use a LIST statement to download the current BASIC/Debug program to the host computer. Uploading a program in ASCII format will store it in RAM as if you had typed it in at the keyboard.

On power-up, BASIC/Debug tests external memory non-destructively, beginning at 800h, to see how much RAM the system contains. A pointer to the high boundary of RAM is stored in an internal register. A small area of RAM is reserved for storing variables, the input line buffer and GOSUB stack.

Circuits that have a program stored in EPROM may have no need for external RAM. In this situation, BASIC/Debug uses the Z8's internal registers for storage, with some operating limitations due to the reduced memory available.

If a program is stored in nonvolatile memory, BASIC/Debug can run it automatically on power-up. On power-up, BASIC/Debug checks external memory location 1020h, and if it finds a program, automatically runs it. This feature allows you to develop a program in RAM, then transfer it to EPROM, EEPROM or nonvolatile RAM for permanent storage and automatic starting.

If you've developed a program in RAM and want to save it to a file for use with an EPROM programmer, you need to save the code exactly as it's stored in the RAM. Although LIST works well for downloading files for later uploading, the LIST statement adds line feeds and translates line numbers from binary format to ASCII; so it's not suitable for downloading files for EPROM programming.

A solution is to download the file in binary format. Listing 2 is a short program that writes the current program in RAM to the serial port in binary format. This listing can be appended to any BASIC/Debug program and called with GOTO 10000. To use this technique, the host computer must be able to receive and save files in the binary format. ASCII downloading protocols strip nulls and/or the eighth bit of each byte; so it won't work for this purpose.

Another option for saving a BASIC program is to copy the program directly into a nonvolatile RAM or EEPROM. Listing 3 is a program that can be appended to a BASIC/Debug program to copy the current program to memory beginning at 1020h. GOTO 10000 causes the program to be copied. After copying, the WE line of the nonvolatile RAM or EEPROM must be jumpered to +5 volts to prevent overwriting and provide autostarting. Or the IC can be removed and inserted into an EPROM programmer that can then copy the contents into an EPROM.

Port 2's pins and the six remaining Port 3 pins can be used to interface to switches, LCD or LED displays, analog-to-digital or digital-to-analog converters or other devices and components. The ports have active pull-ups and pull-downs that are compatible with TTL loads. As Fig. 1 shows, the direction of Port 3's pins is fixed, while Port 2's pins can be programmed individually to serve as inputs or outputs.

Zilog's BASIC/Debug manual contains definitions and examples of each keyword, as well as sections on how BASIC/Debug uses memory and programming tips for maximum execution speed and minimum memory use.

Assembly-Language Programming

For functions that BASIC/Debug can't handle, you can program in as-

Listing 1. BASIC/Debug Program Prompts for a Memory Location and Displays Values Stored in 16 Locations, Beginning With Requested Location.

```
10 PRINT "beginning address?"
20 INPUT X:REM beginning memory location to display
30 A=0
40 IF A<16 THEN 100
50 STOP
100 PRINT HEX(@(X+A)):REM display stored value
110 A=A+1
120 GOTO 40
```

Listing 2. Program Transmits BASIC/Debug Program in Binary Format to the Z8671's Serial Port.

```
10000 X=0
10010 GO @%61,@(%800+X):REM write byte to serial port
10020 IF @(%800+X)=%FF THEN STOP
10030 X=X+1
10040 GOTO 10010
```

Listing 3. Program Copies a Program From RAM Into EEPROM or Nonvolatile RAM for Permanent Storage.

```
10000 X=0
10010 @(%1020+X)=@(%800+X):REM copy RAM byte to NV memory
10020 IF @(%1020+X)=%FF THEN STOP
10030 X=X+1
10040 GOTO 10010
```

sembly language and call the program from BASIC/Debug. For low-cost (free) assembly-language programming, I discovered two versions of a freeware Z8 cross-assembler on the Circuit Cellar Ink BBS (see Sources box). Z8CA1PC.ARC for MS-DOS computers assembles programs in Intel hex, Motorola S-record or a special Z8 file format. Z8CA1AM.ARC is an Amiga version of the same assembler.

After assembling a program, you can program an EPROM or other memory IC with the assembled code or simply upload the file into the Z8's external RAM for testing.

The special Z8 file format is handy for uploading to RAM. A short (five-line) BASIC/Debug program, along with a communications program, uploads a program assembled in Z8 format to the desired location in memory in the Z8 system. Once the program is loaded into memory, you can call it from BASIC/Debug with a GO @ statement. More details on this and how to use the assembler are given in the documentation files.

Full-featured Z8 assemblers are available as well from Zilog and others. Another possibility is to use a universal cross-assembler that supports the Z8. Such a cross-assembler is a single program that assembles programs for a variety of microcomputers from different families.

A convenient feature of the Z8's architecture is that any of its general-purpose registers can be used as an accumulator, address pointer, index register or on-chip stack. This contrasts with many other devices in which specific registers are dedicated to these purposes, and, for example, all calculations must be funneled through an accumulator.

An inconvenient Z8 feature is that several of its internal registers are write-only. To configure the I/O ports, you write values to their mode registers, but there's no way of reading the values back.

Packaging options for the Z8671 include a 40-pin DIP (dual in-line package) and a 44-pin surface-mount chip carrier. Besides the Z8671, other Z8 versions (none of these have BASIC/Debug) include the following:

Z8681/82—A ROM-less version. Like the Z8671, but without BASIC/Debug in ROM. Assembly-language or compiled programs must be stored in external memory.

Z8603/13—A ROM-less version with a piggyback socket for a 2716 or 2732 EPROM. This space-saving version allows you store a program in EPROM without having to wire the EPROM to the Z8. Since standard EPROMs are used, no special adapters are required for your EPROM programmer, unlike many other microcomputers that have embedded into them EPROMs.

Z8601/11—Contains mask-programmed user program in 2K or 4K of ROM. For mass production of chips with a single program.

Z8600—A 28-pin version.

Z86C91—A CMOS ROM-less version.

Z8800—A Super8 version, with improved instruction set that includes multiply and divide instructions, Boolean and BCD (binary-coded decimal) operations, DMA (direct memory access) controller, ability to run at 20 MHz and other improvements.

Z86C27/97 DTC digital television controller. An application-specific version, containing a Z8, an on-screen-display video controller and 13 pulse-width-modulator outputs. It's meant for use in color-television control products. Zilog is continuing to develop other application-specific Z8s meant for specialized markets.

Send comments, suggestions and questions on topics relating to designing, building and programming microcontrollers or other small, dedicated computers to Jan Axelson, *ComputerCraft*, 76 North Broadway, Hicksville, NY 11801. For a personal response, please include a self-addressed, stamped envelope.

Next time: low-power designs for battery-powered projects.

Jan Axelson

Sources

BASICON, Inc.
14273 NW Science Park Dr.
Portland, OR 97229
Voice: 503-626-1012
FAX: 503-643-4686
MC-1z microcontroller board with Z8 BASIC/Debug, expander module, Z8 assembler

Circuit Cellar Ink BBS
Modem: 203-871-1988
300/1,200/2,400 bps, 8-N-1
Z8 cross assemblers

Jameco
1355 Shoreway Rd.
Belmont, CA 94002
Voice: 415-592-8097
FAX: 415-592-2503
Z8 chips

JDR Microdevices
2233 Samaritan Dr.
San Jose, CA 95124
Voice: 1-800-538-5000
FAX: 1-800-538-5005
Z8 chips

Micromint
4 Park St.
Vernon, CT 06066
Voice: 203-871-6170
FAX: 203-872-2204
BCC11 Z8 BASIC Computer, expansion boards, Z8 FORTH chip

PseudoCorp.
716 Thimble Shoals Blvd.
Suite E
Newport News, VA 23606
Voice: 804-873-1947
FAX: 804-873-2154
Z8 assembler, disassembler, simulator

Software Science
3750 Roundbottom Rd.
Cincinnati, OH 45244
Voice: 513-561-2060
ProtoQuick Z8 development board, application notes

Zilog, Inc.
210 Hacienda Ave.
Campbell, CA 95008-6609
Voice: 408-370-8000
Z8 single-chip microcomputer; *Z8 Family Design Handbook* (No. 03-8275-03); Z8671 *BASIC/Debug Reference Manual* (No. 03-3149-03) ■