Z8671 Seven Chip Computer

Zilog

Hardware Application Note

September 1981

Z8671 Seven Chip Computer

Zilog

Hardware Application Note

September 1981

INTRODUCTION

The Z8601 is a single-chip microcomputer with four 8-bit I/O ports, two counter/timers with associated prescalers, asynchronous serial communication interface with programmable baud rates, and sophisticated interrupt facilities. The Z8601 can access data in three memory spaces: 2K bytes of on-chip ROM and 62K bytes of external program memory, 144 bytes of on-chip Register, and 62K bytes of external data memory.

The Z8671 is a Z8601 with a Basic/Debug Interpreter and Debug monitor preprogrammed into the 2K bytes of on-chip ROM. This application note discusses some considerations in designing a low-complexity board that runs the Basic/Debug Interpreter and Debug monitor with an external 4K bytes of RAM and 2K bytes of ROM. The board stands alone, allowing users to connect it with a terminal via an RS232 connector and run the Basic/Debug Interpreter.

The user of this board can run Basic/Debug with little knowledge of the Z8601. The board, however, derives its power through its ability to execute assembly language programs. To use the board to its full potential, the Z8 Technical Manual (document #03-3047-02) and the Z8 PLZ/ASM Manual (document #03-3023-03) should be read. The Z8 Basic/Debug Software Reference Manual (document #03-3134-00) provides general information, statement syntax, memory allocations, and other material regarding Basic/Debug and the Debug monitor provided by the Z8671. There are also two documents describing the Z6132; these are the Z6132 Product Specification (document #00-2028-A), and the Interfacing to the Z6132 Intelligent Memory Application Note (document #00-2102-A).

Basic/Debug

Basic/Debug is a subset of Dartmouth Basic, which interprets Basic statements and executes assembly language programs located in memory. Basic/Debug can implement all the Dartmouth Basic commands directly or indirectly.

One advantage to programming in Basic/Debug is the interactive programming approach realized because Basic/Debug is interpreted, not assembled or compiled. Modules are tested and debugged using the interactive monitor provided with Basic/Debug. Using Basic/Debug saves program development time by providing higher-level language statements that simplify program development. Using the INPUT and PRINT statements simplify debugging.

The Z8671 Microcomputer

Basic/Debug controls the memory interface, serial port, and other housekeeping functions performed by the assembly language programmer.

The Z8671 uses ports 0 and 1 for communicating with external memory. Port 1 provides the multiplexed address/data lines (AD_0-AD_7); port 0 supplies the upper address bits (A_8-A_{15}). The Z8671 also uses the serial communications port for communicating with a terminal. Serial communication takes two pins from port 3, leaving six I/O pins from port 3 available to the user. The serial communication interface uses one of the two counter/timers on the Z8671 chip.

All other functions and features on the Z8601 are available with the Z8671. The user may reconfigure the Z8671 in software as a Z8601 if desired.

Applying the Z8671

Applications of the Z8671 range from a low-complexity home microcomputer that is memory intensive to an inexpensive, I/O-oriented microcontroller.

For home computer users, Basic/Debug is used like other available Basic interpreters. The Z8671, however, has many advantages over other computers. For example, the programmer can use the available functions such as interrupts to perform sophisticated tasks that are beyond the scope of other computer products. There is also a counter/timer

that is used as a watchdog counter, a time-of-day clock, a variable pulse width generator, a pulse width measurement device, and a random number generator.

As an inexpensive microcontroller, Basic/Debug speeds program development time by calling assembly language subroutines (for time critical applications) and by supplying high-level Basic language statements that simplify the programming of noncritical subroutines.

ARCHITECTURE

Two major design goals were set for this Z8671 Basic board. First, the board was to be simple. Second, the board needed to allow the user to write Basic programs and to utilize the features of the Z8601.

Overview

The board has seven IC packages:

- Z8671 (Z8601 preprogrammed with Basic/Debug)
- Z6132 (4K bytes of pseudo-static RAM)
- 2716 (2K bytes of EPROM)
- 1488 (RS232 line driver)
- 1489 (RS232 line receiver)
- 74LS04 (Hex inverter)
- 74LS373 (octal latch)

With these chips, a complete microcomputer system can be built with the following features:

- 2K byte Basic/Debug interpreter in the internal ROM.
- 4K bytes of user RAM.
- 2K bytes of user-programmable EPROM.
- Full-duplex serial operation with programmable baud rates.
- RS232 interface.
- 8-bit counter/timer with associated 6-bit prescalers.
- 124 general-purpose registers internal to the Z8671.
- 14 I/O lines available to the user.
- 3 lines for external interrupts.
- 3 sources of internal interrupts.
- Sophisticated, vectored interrupt structure with programmable priority levels. Each can be individually enabled or disabled, and all interrupts can be globally enabled or disabled.
- External memory expansion up to 124K bytes.
- Memory-mapped I/O capabilities.

This microcomputer can be used as a microcontroller, in which case a terminal is attached, via the RS232 interface, and Basic/Debug is used to create, test, and debug the system. When the system is debugged, the program is put into the EPROM, the terminal disconnected, and the board run standing alone. The terminal can be reattached at any time to monitor the subroutines running on the board.

This proposed board meets the design requirements of simplicity and of allowing the user to write and debug programs in Basic while maintaining access to the Z8671 on-chip features.

Interfacing the Z8671 with External Memory

Both RAM and ROM are used in this application for program development and to demonstrate the use of components with and without address latches.

The RAM interface is easy to implement when using a Z6132 (Figure 1). No external address latch is needed because the Z6132 latches the address internally. The Z6132 signals $\overline{WE}$ (Write Enable), $\overline{DS}$ (Data Strobe), and AC (Address Clock) are wired directly to the Z8671 signals R/$\overline{W}$ (Read/ Write), $\overline{DS}$ (Data Strobe), and $\overline{AS}$ (Address Strobe). The only other signal required is $\overline{CS}$ (Chip Select). $\overline{CS}$ is provided by the Z8671 by decoding the upper address bit of port 0. This board uses address bit 15 to select the chip. Since there are two memory chips on this board, the upper address bit ensures that the Z6132 is selected for addresses 800-7FFF (Hex) and that the 2716 is selected by addresses 8000-FFFF (Hex).

There are two major advantages to using the Z6132. The interface to the Z8671 is uncomplicated because both components are Z-BUS™ compatible, and it provides 4K bytes of RAM in one package.

The ROM interface is not as simple as the interface to the Z6132. Nevertheless, the circuit is common in microcomputer applications. The ROM does not latch the address from the Z8671 and therefore needs an external address latch. The 74LS373 latches the address for the 2716 EPROM. The Enable pin on the 74LS373 is driven by the $\overline{AS}$ signal via an inverter. The EPROM is also selected by the upper address nibble of port 0. Figure 2 shows the Z8671-to-2716 interface.

Interfacing the Z8671 with RS232 Port

The Z8671 uses its serial communication port to communicate with the RS232 port. Driver and receiver circuits are required to supply the proper signals to the RS232 interface. The circuit of Figure 3 shows the interface between the Z8671 and the 1488 and 1489 for serial communication via the RS232 interface.

The serial interface does not use the control signals Clear to Send, Data Set Ready, etc. It uses only Serial In, Serial Out and Ground, so it is a very simple interface.

The Z8671 uses one timer and its associated prescaler for baud rate control. When the Z8671 is reset, it reads location FFFD and uses the byte

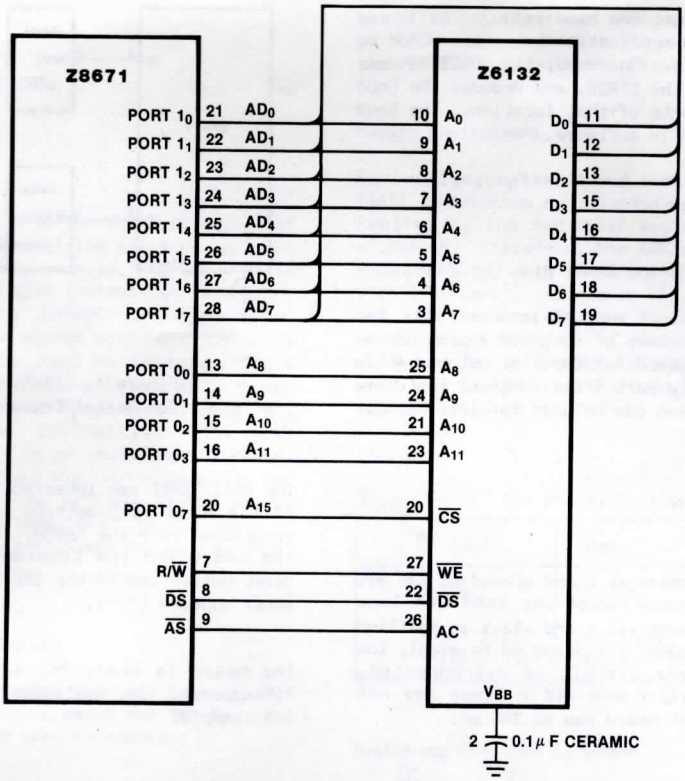

Figure 1. The Z8671 and Z6132 Interface

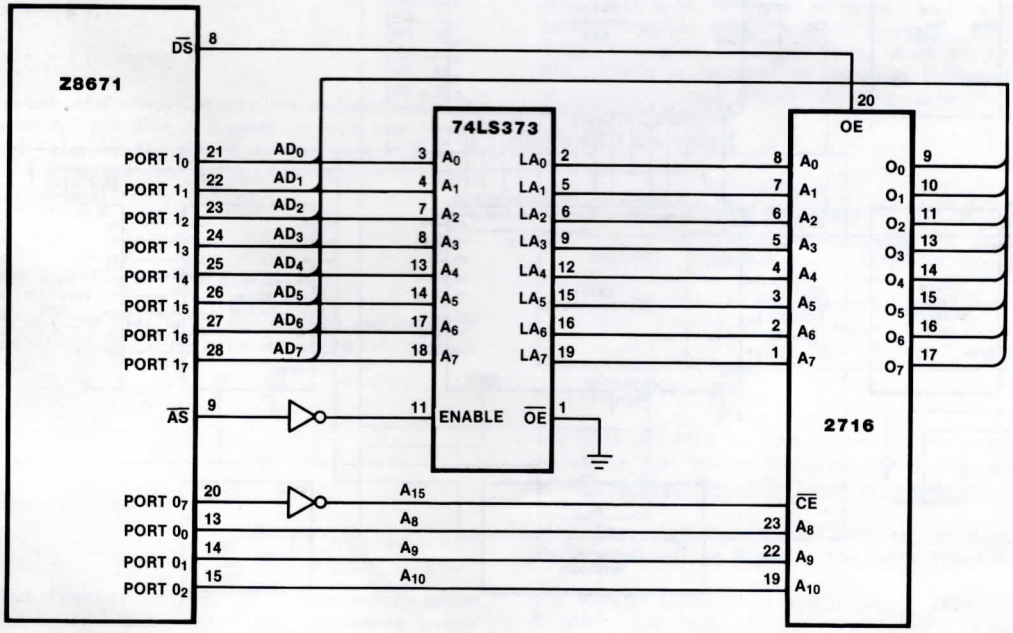

Figure 2. The Z8671 and 2716 Interface

stored there to select the baud rate. The board described in this application note uses EPROM to select the baud rate. On reset, the Z8671 reads FFFD, which is in the EPROM, and decodes the baud rate from the contents of that location. The baud rate can be changed in software.

Figure 4 shows the full board design implemented for this application note.

Uncommitted I/O Pins and Other Pins

Using the above design, port 2 is available for user applications. Any of the port 2 pins can be individually configured for input or output. There are also six pins in port 3 available to the user. The port 3 input pins can be used for interrupts.

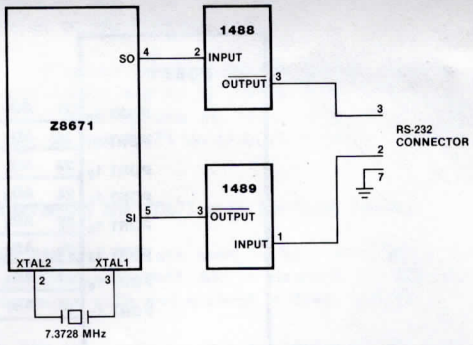

Figure 3. Z8671 Interface for Serial Communications

SOFTWARE

Getting Started

The Z8671 board needs +5 V and ground to run all components on the board except the 1488 EIA line driver. The 1488 needs +12 V and -12 V in addition to the +5 V and ground. (If using no terminal, the EIA driver/receiver circuit is disconnected. Consequently, the +12 V and -12 V lines are not required.) The test board ran at 200 mA.

The RS232 port can interface to any ASCII terminal if the baud rate setting is matched to the value programmed into the EPROM. With power supplied to the board and the terminal connected to it, the reset button resets the Z8671 and the prompt character appears (":").

The board is ready for a Basic command when the ":" appears. The following sequence is a simple I/O example:

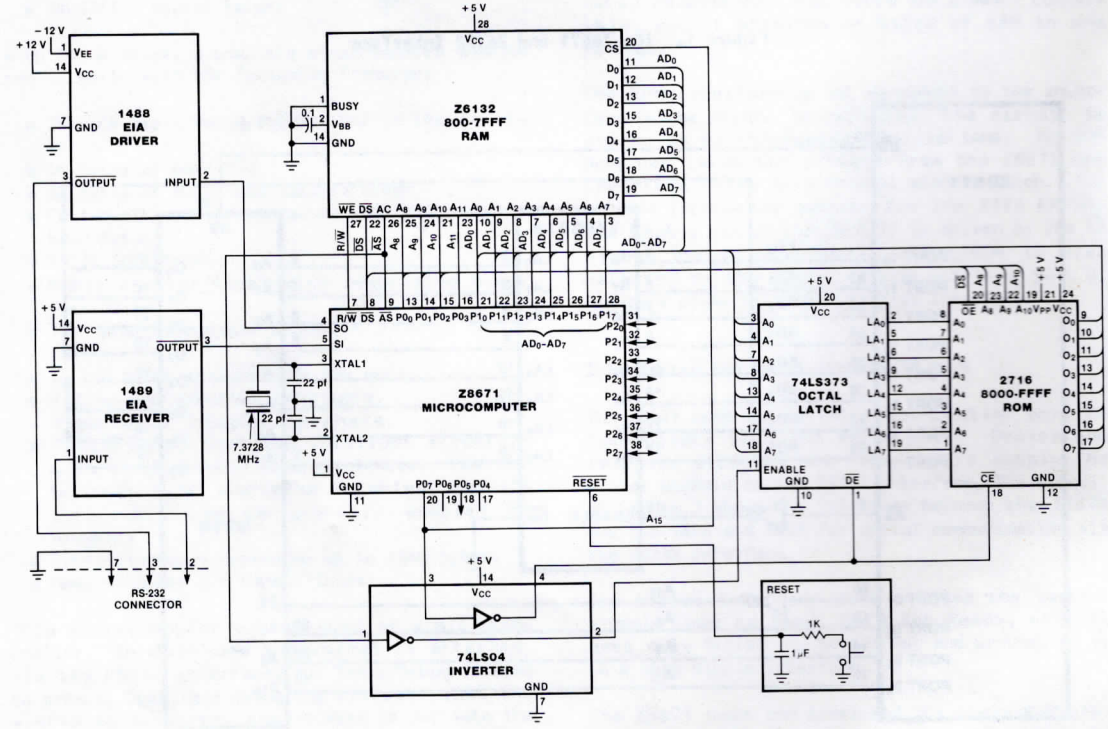

Figure 4. The Z8 System with Basic/Debug

```
:10 input a
:20 "a=";a
:run
 ?5
 a=5
:list
 10 input a
 20 "a=";a
:
```

When a number is entered as the first character of a line, the Basic monitor stores the line as part of a program. In this example, "10 input a" is entered. Basic stores this instruction in memory and prints another ":" prompt. The Run command causes execution of the stored program. In this example, Basic asked for input by printing "?". A number (5) is typed at the terminal. Basic accepts the number, stores it in the variable "a", and executes the next instruction. The next instruction (20 "a=";a) is an implied print statement; writing an actual "print" command is not necessary here. This line of code produced the output "a=5". The command "list" caused Basic to display the program stored in memory on the terminal.

Reading Directly from Memory

Basic lets the user directly read any byte or word in memory using the Print command and "@" for byte references or "↑" for word references:

```
:print @8
 10
:printhex(@8)
 A
:printhex(↑8)
 AF6
:
```

The first statement prints the decimal value of Register 8. The next statement prints the hexadecimal value of Register 8 and the last statement prints the hexadecimal value of Register 8 (0AH) and Register 9 (F6H).

Writing Directly to Memory

Basic lets the user write directly to any register or RAM location in memory using the Let command and either "@" or "↑".

```
:@%a=%ff
:↑4096=255
:print@10
 255
:printhex(↑%1000)
 FF
:
```

The Let command is implied to save memory space but can be included. The first statement loads the hexadecimal value FF into register 10 decimal (AH). The next instruction loads the decimal value 255 into register 4096 decimal (1000H). The print commands write to the terminal the values that were put in with the first two instructions.

Memory Environment

Table 1 gives the memory configuration for the Z8671 application example. Chip Select is controlled by the MSB (most significant bit or A_{15}) of port 0. Therefore, the RAM is selected for all addresses between 800H (2048 decimal) and 7FFFH (32767 decimal). Addresses 8FF, 18FF, 28FF, 38FF, and 78FF address the same location in RAM in this application because of Modulo 4K. EPROM is selected for all addresses from 8000H to FFFFH and, like the RAM, several addresses point to the same location in the PROM.

Table 1
The Memory Environment

Decimal	Hex	Contents
0-2047	(0-7FF)	Internal ROM (BASIC/DEBUG)
2048-32767	(800-7FFF)	RAM (Z6132)
32768-65536	(8000-FFFF)	EPROM (2716)

Switching from RAM to EPROM

Register 8 and Register 9 contain the address of the first byte of a user program or, if there is no program, the address where the Z8671 will put the first byte of a user program. In this application example, when the Z8671 is reset, Register 8 and Register 9 contain 800H, which points into RAM. EPROM is selected by changing the contents of register 8 from 08H to 80H (See Table 2).

Table 2
The Registers

Decimal	Hex	Contents
22-23	(16-17)	Current Line Number
8-9	(8-9)	Address of the First Byte of User Program

For more details on the register assignments, refer to the Pointer Registers-RAM System section of the Z8 Basic/Debug Software Reference Manual.

After the instruction "↑8=%8000" is executed, the Z8671 accesses the EPROM on the Basic/Debug Board.

The example below shows how to switch from RAM to EPROM. The example uses two separate programs, one in RAM and one in EPROM. The RAM program is listed first, then the EPROM.

```
:printhex(↑8)
 800
:list
 10 "executing out of RAM"
:↑8=%8000
:printhex(↑8)
 8000
:list
 10 "executing out of EPROM"
:
```

Baud Control

The baud rate is selected automatically by reading location FFFDH and decoding the contents of that location when the Z8671 is reset (the Z8 Basic/Debug Software Reference Manual contains the baud rate switch settings in Appendix B). This application example holds the baud rate settings in its EPROM. The least significant bits of location FFFD hex will provide baud rates as follows:

Baud Rate	Value Read
110	110
150	000
300	111
1200	101
2400	100
4800	011
9600	010
19200	001

After a reset, the baud rate is programmed by loading a new value into counter/timer 0 (see the Z8 Technical Manual, section 1.5.7). A Reset always changes the baud rate back to the rate selected from the contents of location FFFD.

Burning an EPROM

The EPROM contains the baud rate selection byte in location 7FDH. The other locations in memory are used for program storage. See section 6.3 of the Basic/Debug Manual for the format used to store programs in memory. This format is used to store programs in EPROM.

Example

The following is a printout of the game Mastermind written in Basic/Debug.

```
10 @243=7
20 @242=10
30 @241=14
40 x=usr(84):a=@242-1:x=usr(84):b=@242-1
50 x=usr(84):c=@242-1:x=usr(84):d=@242-1
55 "":i=0
100 "guess ",:in e,f,g,h
110 i=i+1
300 j=%7f22:k=%7f2a
301 l=0
302 r=0:p=0
310 if↑j=↑kp=p+1
320 j=j+2:k=k+2:l=l+1:if 4 > 1310
330 J=%7f22:k=%7f2a
331 l=0
340 if↑j=↑kr=r+1:↑j=↑j+10:l=3
341 j=j+2
350 l=l+1:if4 > 1340
351 j=%7f22
352 l=0
360 k=k+2:if%7f31>k340
363 j=%7f22:k=%7f2a
366 if↑j>9↑j=↑j-10
367 j=j+2
368 if%7f29>j366
370 "right ";r;" place ";p
380 if4>p100
390 y=999
400 "right in ";i;" guesses;";"play another
    y/n":inputx
410 ifx=y10
```

Lines 10 through 50 comprise the random number generator for the program. The three lines:

```
10 @243=7
20 @242=10
30 @241=14
```

initialize counter/timer 1 to operate in modulo-10 count. Refer to the Z8 Technical Manual for complete information on initializing timers.

The "usr(84)" function waits for keyboard input, the ASCII value of the key is returned in a variable with the following command:

```
:10 x=usr(84):""
:15 printhex(x)
:run
 5
 35
:
```

In the above example, the program waits at line 10 until keyboard input, in this case the number 5. The input value is stored in ASCII format in the variable "x". The line:

```
40 x=usr(84):a=@242-1:x=usr(84):b=@242-1
```

waits for input, reads the current value of timer 1, subtracts 1 (to get a number between 0 and 9), and stores the number in variable a. Then it waits for keyboard input at the second user function call, reads the current value of timer 1, subtracts 1, and stores the number in variable b. Line 50 of the example program gets two more random numbers and stores them in variables c and d. The four-digit random number is located in variables a, b, c, and d.

Line 300 assigns the location of variable a to variable j and the location of variable e (the

first variable in the guess string) to the variable k. The strategy is to access these variables indirectly and to increment pointers j and k to access the variables.

A colon is used to separate commands on the same line. This is useful in packing the program into a small amount of memory space. The code, however, is harder to read. See section 5 of the Basic/Debug manual for more information on memory packing techniques.

Below is a sample run of the Mastermind program:

```
:run
(<RETURN> on the keyboard is entered four
 times here)
guess ? 0, 1, 2, 3
right 2 place 0
guess ? 4, 5, 6, 7
right 2 place 1
guess ? 0, 2, 4, 6
right 3 place 2
guess ? 4, 2, 1, 6
right 4 place 4
right in 4 guesses
play another? y/n
?n
:
```

CONCLUSION

The design of this application example met the major design goals of simplicity and functionality. The first goal is accomplished by prudent selection of support components, excluding any unnecessary chips. The board allows the user to exercise the full power and flexibility of the features of the the Z8601 not used by Basic/Debug. The user can write and debug Basic programs without detailed knowledge of the Z8601.

The Basic application example demonstrates a memory interface that is applicable for all Z8 Family members. The case where there is no address latch on the memory chip was discussed, and an example of how to interface the multiplexed address/data bus of the Z8 Family through an address latch was shown.

The software section explains the memory environment and gives several examples of Basic/Debug. These examples are a good introduction to the board and to Basic/Debug.

The Z8671 is a customized extension of the Z8601 single-chip microcomputer. The simplicity of the Basic application example demonstrates the flexibility of the Z8601 microcomputer in an expanded memory environment.

Zilog Sales Offices

West

Sales & Technical Center
Zilog, Incorporated
1333 Lawrence Expressway
Suite 400
Santa Clara, CA 95051
Tele: (408) 446-9848
TWX: 910-338-7621

Sales & Technical Center
Zilog, Incorporated
18023 Sky Park Circle
Suite J
Irvine, CA 92714
Tele: (714) 549-2891
TWX: 910-595-2803

Sales & Technical Center
Zilog, Incorporated
15643 Sherman Way
Suite 430
Van Nuys, CA 91406
Tele: (213) 989-7484
TWX: 910-495-1765

Midwest

Sales & Technical Center
Zilog, Incorporated
890 East Higgins Road
Suite 147
Schaumburg, IL 60195
Tele: (312) 885-8080
TWX: 910-291-1064

Sales & Technical Center
Zilog, Inc.
28349 Chargrin Blvd.
Suite 109
Woodmere, OH 44122
Tele: (216) 831-7040
FAX: 216-831-2957

South

Sales & Technical Center
Zilog, Incorporated
2711 Valley View, Suite 103
Dallas, TX 75234
Tele: (214) 243-6550
TWX: 910-860-5850

Zilog, Incorporated
7115 Burnet Rd.
Suite 207
Austin, TX 78757
Tele: (512) 453-3216

Technical Center
Zilog, Incorporated
1442 U.S. Hwy 19 South
Suite 135
Clearwater, FL 33516
Tele: (813) 535-5571

East

Sales & Technical Center
Zilog, Incorporated
Corporate Place
99 South Bedford St.
Burlington, MA 01803
Tele: (617) 273-4222
TWX: 710-332-1726

Sales & Technical Center
Zilog, Incorporated
110 Gibraltar Road
Horsham, PA 19044
Tele: (215) 441-8282
TWX: 510-665-7077

United Kingdom

Zilog (U.K.) Limited
Babbage House, King Street
Maidenhead SL6 1DU
Berkshire, United Kingdom
Tele: (628) 36131
TELEX: 848609

West Germany

Zilog GmbH
Zugspitzstrasse 2a
D-8011 Vaterstetten
Munich, West Germany
Tele: 08106 4035
TELEX: 529110 Zilog d.

Japan

Zilog, Japan KK
Linden Sky Heights
Bldg. 1F
13-2 Sakuragaoka-Machi
Shibuya-Ku Tokyo 105
Japan
Tele: (813) 496-4428
TWX: 781-23723 Lawright