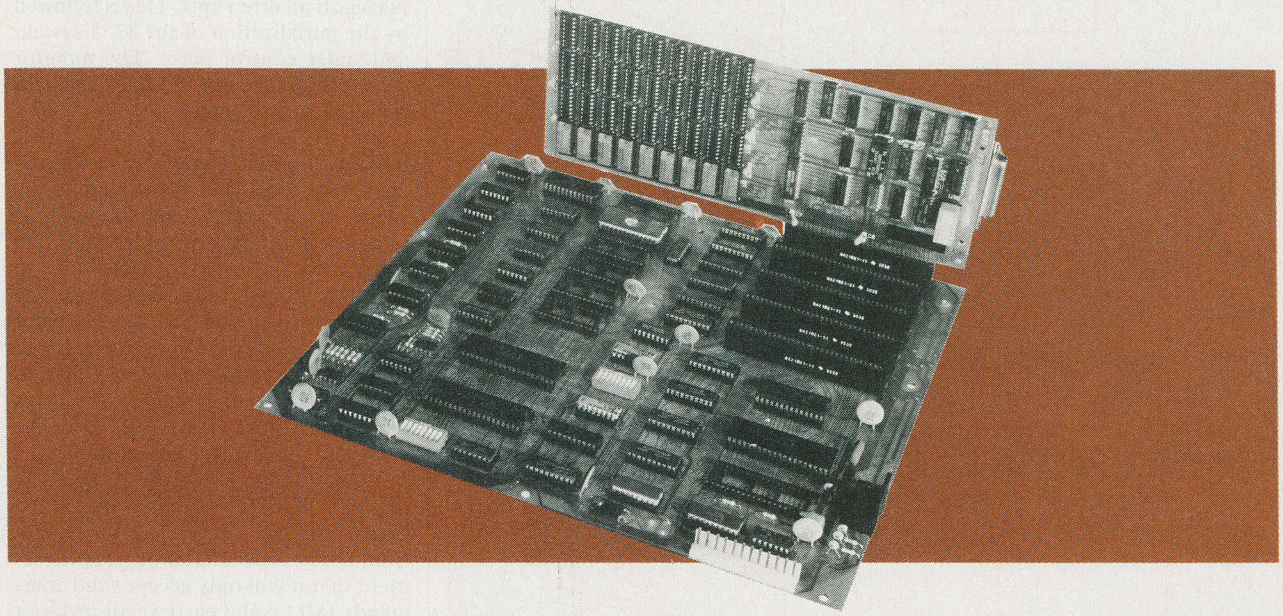
Learning

16-BIT MICROCOMPUTER TECHNOLOGY

Part 6: How to Use Machine Language Programming

By George Meyerle



PREVIOUS articles in this series should have convinced you that, before you can understand, repair, or design microcomputers that use the latest generation of hardware, you must have had at least an introduction to machine/assembly language programming. By using machine language subroutines in conjunction with BASIC or other high-level languages, it is possible to increase execution speed and, in some cases, allow operations that are not possible using the high-level interpreter alone. This month we will discuss how to learn machine-language programming painlessly and we will gain some practice using it with the monitor program resident in the IBM-compatible, 16-bit Explorer 88/PC microcomputer.

Quickly reviewing the design of the Explorer 88, recall that it can be set up to operate in 3 different modes. The first, and least expensive mode ($400), allows accessing the monitor program using a standard terminal connected to the RS232 port. The second choice (about $600 more) is to add an IBM-compatible keyboard, color board, and either a color or black/white monitor. This gets you that much closer to being truly IBM-compatible. The next step involves the addition of a floppy-disk card and up to four floppy drives. At this point, you can run IBM-PC DOS or any of the many IBM-compatible disk operating systems, including Digital Research CP/M-86, all of which include fine monitor or debugging programs. The following applies to any configuration selected.

Monitor Programs. In its simplest form, a monitor program is a collection of routines that allow programs to be generated, tested, modified, and saved for future use. The original microcomputer monitors consisted of a front panel array of switches and LEDs that allowed access to the CPU and memory. However, the flashing LED approach was hard to use and scared away many casual observers. On the other hand, ROM-based monitors, including that used in the Explorer 88, are easy to use and understand; and, with a litle practice, you should be able to write short programs in 16-bit 8088 machine language. Since it is not practical to write long programs or subroutines without the help of an assembler, once the basics are learned, it is suggested that you invest in one of the many available assembler programs.

Monitor Overview. All 8088 microprocessors auto-boot on power-up or reset to location FFFF0H. This location, in the Explorer monitor, contains a jump to location FE000H which is the beginning address of the upper-most 8K of ROM and of the monitor program

TABLE VI—I/O PARAMETER SETTINGS

Baud rate	S1-3	S1-2	S1-1
110	On	On	On
150	On	On	Off
300	On	Off	On
600	On	Off	Off
1200	Off	On	On
2400	Off	On	Off
4800	Off	Off	On
9600	Off	Off	Off

Parity disabled: S1-4 on
Parity enabled: S1-4 off
Parity odd: S1-5 on
Parity even: S1-5 off

TABLE VII—MEMORY MAP

00000-0FFFF	RAM
FE000-FFFFF	ROM
00000-00003	Divide-by-zero interrupt
00004-00007	Single-step interrup
00010-00013	Overflow interrupt
0000C-0000F	Monitor-trap
00010-0002F	System
00030-00033	UART interrupt
00034-0FEFF	System

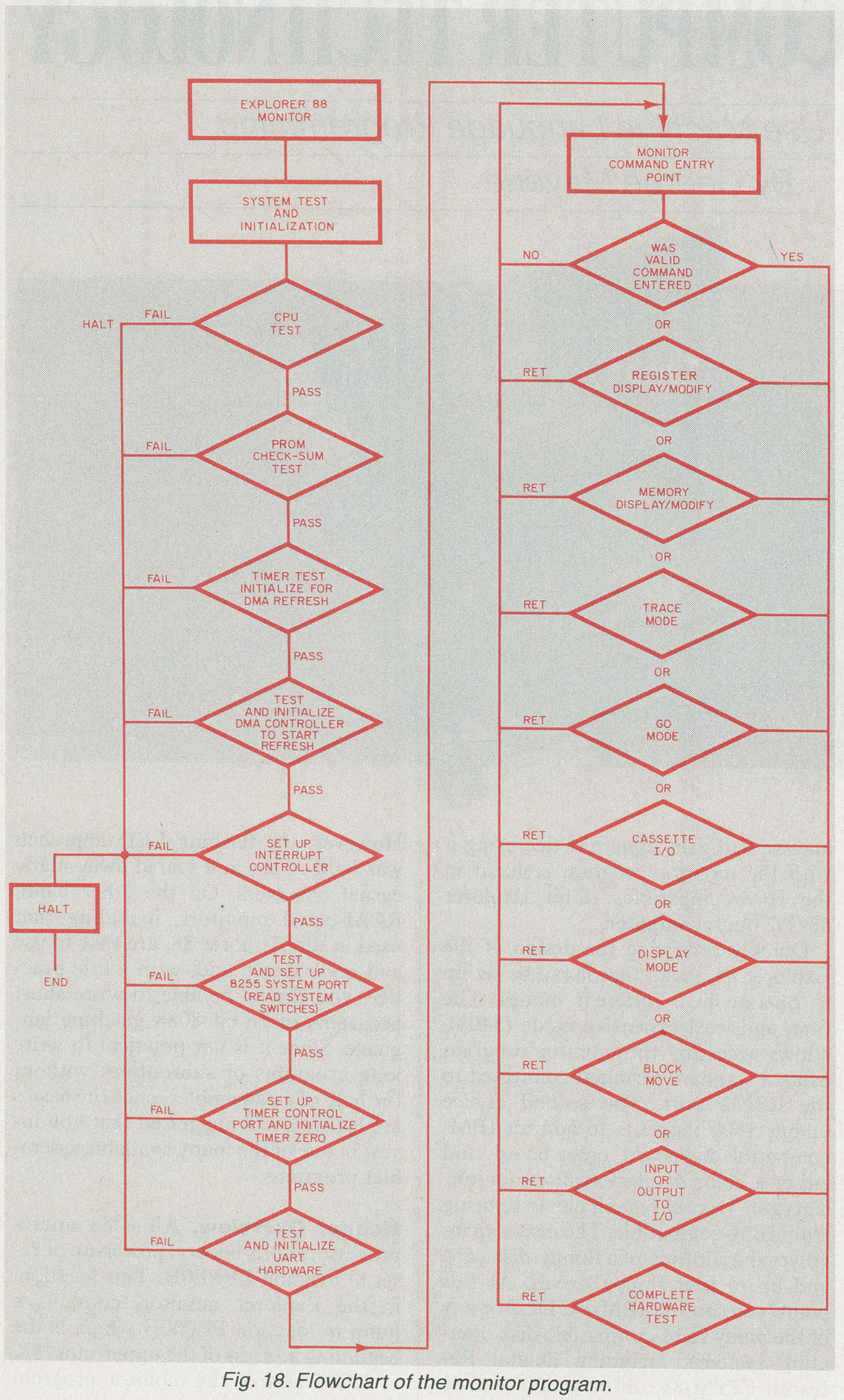

EXPLORER 88 MONITOR
SYSTEM TEST AND INITIALIZATION
HALT
FAIL
CPU TEST
PASS
FAIL
PROM CHECK-SUM TEST
PASS
FAIL
TIMER TEST INITIALIZE FOR DMA REFRESH
PASS
FAIL
TEST AND INITIALIZE DMA CONTROLLER TO START REFRESH
PASS
FAIL
SET UP INTERRUPT CONTROLLER
HALT
END
PASS
FAIL
TEST AND SET UP 8255 SYSTEM PORT (READ SYSTEM SWITCHES)
PASS
FAIL
SET UP TIMER CONTROL PORT AND INITIALIZE TIMER ZERO
PASS
FAIL
TEST AND INITIALIZE UART HARDWARE
MONITOR COMMAND ENTRY POINT
NO
WAS VALID COMMAND ENTERED
YES
OR
RET
REGISTER DISPLAY/MODIFY
OR
RET
MEMORY DISPLAY/MODIFY
OR
RET
TRACE MODE
OR
RET
GO MODE
OR
RET
CASSETTE I/O
OR
RET
DISPLAY MODE
OR
RET
BLOCK MOVE
OR
RET
INPUT OR OUTPUT TO I/O
RET
COMPLETE HARDWARE TEST

Fig. 18. Flowchart of the monitor program.

whose flowchart is shown in Fig.18. Note in this flowchart that, before the program is ready to accept commands from the keyboard, a preliminary hardware check is made. If any hardware test fails, the CPU is automatically halted. The program begins by checking the CPU. All registers, including the flags, are tested. If they are OK, the program continues with a PROM check-sum test. This adds the contents of all 8K of ROM and compares the results with the correct answer.

The next component tested is channel 1 of the timer. If the timer test is OK, channel 1 is programmed to provide the refresh clock signal to the DMA controller. Next the DMA controller is tested. The count and command registers are tested by writing various patterns and then checking the results by reading the registers. If all tests are passed, the count register is set for a 64K system and channel 0 is enabled so that the RAM refresh process can begin.

The program then sets up the interrupt controller mode registers and masks off all interrupts. This is followed by the initialization of the 8255 system and timer control ports. The monitor then sets up the data and stack segment registers and the stack pointer. It also clears the console input and output buffers. The 8250 UART is then initialized. This process involves reading the switch #1 at port B of the 8255. These switch settings determine the baud rate and parity selections (Table VI) to which the UART will be initialized. The monitor also downloads the interrupt jump vectors to the lowest locations in RAM as shown in the memory map in Table VII. Note that the top 256 bytes in the first 64K bank are reserved for the stack and other monitor operations.

After testing the hardware and setting the UART, the monitor idles at the Command Entry Point awaiting entries from the keyboard. The program is set up so that it will only accept valid commands. All invalid entries will result in the display of "??" and the abortion of the process attempted. Valid commands are:

D	Display memory contents
R	Register display/modification
B	Block move of memory
M	Memory modification/display
I	Input one byte from an I/O device
O	Output one byte to an I/O device
CO	Cassette output from memory to cassette
CI	Cassette input from cassette to memory
T	Trace instructions
TR	Trace instructions with register report
G	Go run from memory with optional breakpoint
GR	Go run from memory with optional breakpoint and register report
HT	Hardware tests of ROM, RAM, cassette, interrupt controller, timer, DMA controller and UART.

In future issues, we will show how to use the ROM-based monitor in practical programming.

(To be continued)