Part I of this article appeared in the March, 1982, issue and Part II appeared last month. This completes the routines which comprise the Supermonitor for the Ohio Scientific Superboard Computer.

Supermonitor: Part III

Frank Cohen
Pacific Palisades, CA

Here is the conclusion to a long and complex program which adds functions to a Superboard II which you would normally find only in an advanced operating system. These functions make it easy to display, move, and modify machine language programs and data.

The programs listed so far have made up the framework of the Supermonitor. The first program presented was called Hexdump and did nothing more than dump an address and eight bytes of data onto the screen. Hexdump was listed first because subsequent programs use some of its subroutines. The second article included two programs. Indata prints a line of eight bytes of data and allows you to modify the contents. After you have modified a byte, Indata allows you to move forward, backward, or skip over subsequent memory locations. Bmove is a simple block move program. Bmove moves a whole block of memory to another location in memory. With just these three programs, entering and editing maching language data is much more efficient and easy than using the ROM Monitor program OSI supplies.

Without a disk system, loading Supermonitor in its entirety takes about five minutes with the Superboard's 300 baud cassette interface. With the assembly listings of Supermonitor you can use only the programs you find interesting. By doing this, you can limit the size of Supermonitor. The listing of the main menu program shows all the equates for all the programs.

All of the programs of Supermonitor use a program called Supercursor V1.3 (**COMPUTE!**, December, 1981, #19, p. 124) to handle its video output. Supermonitor is installed directly below Supercursor at the top of an 8K byte Superboard II. If you don't want to use Supercursor, you can write your own video output routines. To use Supercursor V1.3 a program puts the ASCII character in the CPU's accumulator and executes a JSR to its start address, located at $1E80. Supercursor also has routines to "Home" the cursor and clear the screen. To use the Home functions, jump to the subroutine at $1E80 using a JSR. Use the same instruction to clear the screen at $1EC2.

A Brief Review

Let's go over some terms. An *assembler* is nothing more than a program which takes programs called *source code* and converts them into machine instructions (called *object code*) which can be directly executed by your computer. Assembly language is made up of three-letter codes which abbreviate what the CPU [*Central Processing Unit*] executes. For example, one commonly used instruction is the "load the accumulator" instruction. In machine language, the code is an A9 followed by the byte to be loaded into the accumulator. In assembly language, the instruction looks something like this: LDA. This stands for LoaD Accumulator. But load it with what?

The 6502 microprocessor has twelve different addressing modes. So, following the LDA instruction, the assembler looks for the type of addressing to use. One of the most common is the *immediate mode*. To load the accumulator with the value 00 (hex) the assembler instruction looks like this: LDA #$00. The pounds sign (#) stands for immediate addressing and the dollar sign ($) tells the assembler that this is a hexadecimal number. If you left out the pounds sign, the assembler would think that you want to load the accumulator with a byte residing at location $00 in the zero page of memory. Executing an instruction like LDA $1000 tells the assembler to load the accumulator with the byte at location $1000 in memory. Labels may be used instead of the actual numbers.

These labels are called *equates*. Before entering the program into the assembler labels can be defined. By defining the labels, specific numbers are assigned to alphanumeric names. In the listing of the main menu program, the major equates are shown. For example, the equate named *cursor* is assigned the value $1E40. So, when we tell the assembler to jump to a subroutine called *cursor* (JSR CURSOR) the assembler will execute the subroutine starting at $1E40. Using equates, assembly language becomes easier to read.

Main Menu

This is by far the simplest of the programs. By entering at $1A7B (called SPMON) the program first clears the screen, then homes the cursor and reads the keyboard. When a key is pressed, it checks to see if it is a valid character. If it is, we jump to the correct program. If not, the screen is cleared and we return to the beginning of the program. The valid characters are listed below:

G – EXECUT, transfers control to a machine language program
I – INDATA, displays and modifies memory

QUALITY SOFTWARE FOR TRS-80 COLOR AND OSI

ADVENTURES!!!
For TRS-80 COLOR and OSI. These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are inter-active fantasies. It's like reading a book except that you are the main character as you give the computer commands like "Look in the Coffin" and "Light the torch.")
Adventures require 16K on COLOR-80 and TRS-80. They sell for $14.95 each.

ESCAPE FROM MARS
(by Rodger Olsen)
This ADVENTURE takes place on the RED PLANET. You'll have to explore a Martian city and deal with possibly hostile aliens to survive this one. A good first adventure.

PYRAMID (by Rodger Olsen)
This is our most challenging ADVENTURE. It is a treasure hunt in a pyramid full of problems. Exciting and tough!

TREK ADVENTURE (by Bob Retelle)
This one takes place aboard a familiar starship. The crew has left for good reasons — but they forgot to take you, and now you are in deep trouble.

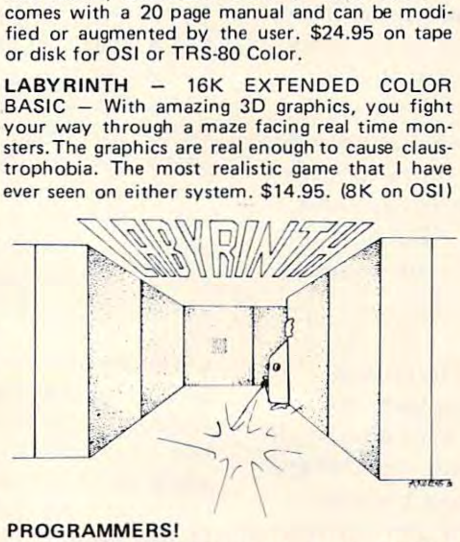
NEW!!
CIRCLE WORLD — We got Kzinti and puppeteers and problems. Our newest and biggest adventure. Requires 12k on OSI and 16K on TRS-80 Color.

NUCLEAR SUB — You are trapped in a nuclear sub at the bottom of the ocean. Escape and even survival is in doubt. Plotted by three of the most sadistic — I mean "creative" minds in adventure programming.

VENTURER!—A fast action all machine code Arcade game that feels like an adventure. Go berserk as you sneak past the DREADED HALL MONSTERS to gather treasure in room after room, killing the NASTIES as you go. Great color, high res graphics, sound and Joystick game for the TRS-80 Color or OSI machines. (black and white and silent on OSI.) Tape only. $19.95.

BASIC THAT ZOOOMMS!!
AT LAST AN AFFORDABLE COMPILER FOR OSI AND TRS-80 COLOR MACHINES!!! The compiler allows you to write your programs in easy BASIC and then automatically generates a machine code equivalent that runs 50 to 150 times faster.
It does have some limitations. It takes at least 8K of RAM to run the compiler and it does only support a subset of BASIC—about 20 commands including FOR, NEXT, END, GOSUB, GOTO, RETURN, END, PRINT, STOP, USR(X), PEEK, POKE, *, /, +, -, X, X, =, VARIABLE NAMES A-Z, A SUBSCRIPTED VARIABLE, and INTEGER NUMBERS FROM 0 - 64K.
TINY COMPILER is written in BASIC. It generates native, relocatable 6502 or 6809 code. It comes with a 20 page manual and can be modified or augmented by the user. $24.95 on tape or disk for OSI or TRS-80 Color.

LABYRINTH — 16K EXTENDED COLOR BASIC — With amazing 3D graphics, you fight your way through a maze facing real time monsters. The graphics are real enough to cause claustrophobia. The most realistic game that I have ever seen on either system. $14.95. (8K on OSI)



PROGRAMMERS!
SEE YOUR PROGRAM IN THIS SPACE!! Aardvark traditionally pays the highest commissions in the industry and gives programs the widest possible coverage. Quality is the keyword. If your program is good and you want it presented by the best, send it to Aardvark.

QUEST — A NEW IDEA IN ADVENTURE GAMES! Different from all the others. Quest is played on a computer generated map of Alesia. Your job is to gather men and supplies by combat, bargaining, exploration of ruins and temples and outright banditry. When your force is strong enough, you attack the Citadel of Moorlock in a life or death battle to the finish. Playable in 2 to 5 hours, this one is different every time. 16K COLOR-80 OR TRS-80. ONLY $14.95.

SPACE ZAPPER — Protect your central Star Base from ships that attack from all four sides. Fast reflexes are required as the action speeds up. Great for kids or Dads. This game has high speed high resolution graphics and looks as if it just stepped out of the arcades. — 16K extended or 32K disk. BASIC TRS-80. Color only. $14.95.

Please specify system on all orders

This is only a partial listing of what we have to offer. We have arcade and thinking games, utilities and business programs for the OS1 and TRS-80 Color. We add new programs every week. Send $1.00 for our complete catalog.

AARDVARK - 80
TRS 80 COLOR **2352 S. Commerce, Walled Lake, MI 48088** **OSI**
(313) 669-3110

C – CLEAR, clears the screen
D – HXDMP, dumps memory to the screen
M – BMOVE, moves a block of data
S – TAPOUT, saves a block of data to cassette tape
F – FILL, fill a block of memory with a specified byte

As it is listed, SPMON fits directly under all the other programs. It uses the clear screen and home cursor functions of Supercursor and a subroutine in the ROM monitor (at $FFBA) to get a key from the keyboard.

Execut

If SPMON is the simplest of the programs, EXECUT is the smallest. Most of EXECUT is devoted to input the starting address of the machine language program. EXECUT prints "G=" on the screen and expects you to type in the four digit address. An infrequently used instruction is applied to jump to the address. This instruction is called the "jump indirect" instruction. EXECUT uses the INADR subroutine in HXDUMP to input the address to locations $00E7 and $00E8. We then use the jump indirect instruction to use these addresses.

Fill

This program is similar to BMOVE. FILL loads a block of memory with some value you input. It starts by asking the beginning address of memory by printing "S=" on the screen. Type in the four digit hexadecimal address. FILL then asks for the ending address by printing "E=". Again, input the address. Then it asks what the block of memory is to be filled with. FILL is a very fast program and will fill all 64K in about two seconds. FILL is listed to fit after the main menu and before the cassette tape program.

Tapout

This is the most valuable program. There exist programs that save from machine language to tape, but the problem is that BASIC uses almost all of the zero page memory locations and some of the main memory making it difficult to work around. Since the Superboard's ROM monitor already has a tape input routine, this program only stores data onto cassette.

TAPOUT makes use of BASIC's cassette output subroutine stored in ROM. By setting location $0205 to FF (hex) a jump to subroutine instruction outputs the contents of the accumulator to the cassette interface at 300 baud. After TAPOUT is finished, it resets location $0205 to 00. If you want to use TAPOUT from a machine language program, put the starting address at location $00E9 and $00EA and also the ending address at location $00E7 and $00E8. Then execute the program.

After you install the three programs in this issue, it is necessary to make some slight modifications so that all the programs will return control to the main menu program. To do this you will need to enter the following modifications:

```
1C36  4C  7E  1A   ;For BMOVE
1CB0  4C  7E  1A   ;For INDATA
1D1D  4C  7E  1A   ;For HEXDUMP
1D38  F0  E3
```

```
;EQUATES
CURSOR    =$1E40
CLS       =$1EC2
HOME      =$1E80
INADR     =$1D93
CR        =$1E95
LF        =$1EAB
KEYIN     =$FFBA
EXECUT    =$1ABA
FILL      =$1ACA
TAPOUT    =$1B3F
BMOVE     =$1BC6
INDATA    =$1C56
HXDMP     =$1D20
ADR       =$E7
EBAD      =$E9
SBAD      =$EB
TMP       =$ED
CVAHX     =$1DF3
CVHA      =$1D72
OFLAG     =$0205
AOUT      =$FFEE
```

```
1A7B 20 C2 1E 20 80 1E A9 24
1A83 20 40 1E 20 BA FF C9 47
1A8B D0 03 4C BA 1A C9 49 D0
1A93 03 4C 56 1C C9 4C D0 03
1A9B 4C 7B 1A C9 44 D0 03 4C
1AA3 20 1D C9 4D D0 03 4C C6
1AAB 1B C9 53 D0 03 4C 3F 1B
1AB3 C9 46 D0 C4 4C CA 1A A9
1ABB 47 20 40 1E A9 3D 20 40
1AC3 1E 20 96 1D 6C E7 00 20
1ACB 80 1E A9 53 20 40 1E A9
1AD3 3D 20 40 1E 20 96 1D A5
1ADB E7 85 E9 A5 E8 85 EA 20
1AE3 95 1E 20 AB 1E A9 45 20
1AEB 40 1E A9 3D 20 40 1E 20
1AF3 96 1D 20 95 1E 20 AB 1E
1AFB A9 42 20 40 1E A9 3D 20
1B03 40 1E 20 BA FF 20 40 1E
1B0B 20 F3 1D 0A 0A 0A 0A 85
1B13 ED 20 BA FF 20 40 1E 20
1B1B F3 1D 18 65 ED 85 ED A5
1B23 ED A0 00 91 E9 E6 E9 A5
1B2B E9 D0 02 E6 EA A5 E9 C5
1B33 E7 D0 EC A5 EA C5 E8 D0
1B3B E6 4C 7E 1A 20 80 1E A9
1B43 53 20 40 1E A9 3D 20 40
1B4B 1E 20 96 1D A5 E7 85 E9
1B53 A5 E8 85 EA A9 45 20 40
1B5B 1E A9 3D 20 40 1E 20 96
1B63 1D A9 FF 8D 05 02 A9 2E
1B6B 20 EE FF A5 EA 20 A5 1B
1B73 A5 E9 20 A5 1B A9 2F 20
```

©