Part I:

A Superboard II Monitor

Frank Cohen
Pacific Palisades, CA

In the last article I stated my opinion of the OSI Superboard II's video output. The result was an advanced cursor routine to supplement the powerful ROM-based Microsoft BASIC. Super-Cursor adds many functions to the output routine of BASIC. However, Super-Cursor is a very long routine (almost 500 bytes long) and entering it into the Superboard can take forever using the existing monitor program. This gave me the inspiration to write the next couple of programs.

Upon turning on the Superboard and pressing the BREAK key, one is presented with a choice of entering the BASIC, disk or monitor programs. Unless one knows enough about how BASIC works to write a machine code interface, using the monitor program is the only way to program the computer in machine language. It only takes one look at the monitor program to know that one is very limited in the things that can be done.

The monitor will perform only three different things with the Superboard. First, it allows the viewing of one memory location at a time. This gives the capability of seeing one whole byte of memory for every given keyboard entry. Imagine trying to look through a program like Super-Cursor one byte at a time, keeping in mind there are over 500 instructions. Finding one byte incorrectly entered would take hours. Second, the monitor allows the modification of only one location at a time. After a byte has been entered, the next address is displayed. A problem will occur here if you are not sure the byte you just entered was correct. In this case you would have to go back and look at the last memory location. It then becomes necessary to retype the four digits of the previous address. Finally, the monitor allows the entry of a machine language program from a cassette recorder. Of course, it doesn't allow one to store a program on a cassette recorder (which leads one to wonder what one would be loading to begin with).

An Advanced Monitor

It doesn't take long after you start programming in machine language to realize the necessity of a more advanced monitor program. This program (and two others to follow) form an advanced monitor routine. To be quite original I have named the total program Super-Monitor. It is intended for the type of person who knows the basics of machine language programming and wants to expand his knowledge. The three programs are fully documented and in assembly language format. They are written in three separate packages so that you, the user, can modify to your specifications while, at the same time, learn some simple machine language programming techniques.

Before we start, an outline of what a monitor program should do will make the whole thing come together at the end. First, the monitor should be able to display as few or as many memory locations at one time as is desired. Second, it should allow easy entry of many bytes into the Superboard's memory along with the ability to see what was just entered. Third, it should allow the user to fill many memory locations with certain strings or combinations of bytes. Fourth, it should be able to move a whole block of data from one location to another in memory. Fifth, we should have a routine to store memory onto tape. Finally, we should have a routine to read a program from tape to memory with verification. Once all this is accomplished, entering and editing machine language programs will be a snap.

Hexdump

This program allows us to see many locations of memory at once. It will ask you for a beginning memory address and then it will print one line of eight bytes of data from the memory address specified. It will then wait for your command. If you want the next line of eight bytes to be listed, you simply hit the carriage return key. If you want to list another part of memory, you hit the line feed key and HEXDUMP will start again by asking for a new beginning address.

Before we get into seeing just how Hexdump works, here is a small word of warning. If you read the article describing Super-Cursor V1.3 in **COMPUTE!** #18 you should nave noted that my Superboard II has had the video modifications added to give a video display of 26 lines with 48 characters per line, using the Super-Cursor program. The modifications are simple and well described in conversion plans which can be purchased from Elcomp Publishing (Silver Spur Electronics, Chino, California.)

Hexdump does use some of the routines in Super-Cursor so it is necessary that you load Super-Cursor as well. If you don't want to use Super-Cursor, you will have to write your own output routines to allow Hexdump to display information onto the screen. Writing these routines is not very difficult and you probably could copy the individual routines out of Super-Cursor with only a few modifications.

The assembly listing of Hexdump shows that it loads into locations 1D20 through 1E38. This is the area in memory directly below Super-Cursor V1.3. There are seven bytes of memory which separate the two programs. These will be used later for another part of Super-Monitor. Hexdump may be moved to another location in memory by reassembling it; however, if you do not own an assembler, relocating Hexdump may become very difficult as it uses absolute addresses extensively.

Hexdump is an example of structured machine language programming. It uses a main supervisor routine which branches into other routines. The supervisor is labeled Hexdump in the assembly listing, and calls all of the other subroutines into play.

Upon starting Hexdump, the supervisor homes the cursor using the Home routine of Super-Cursor V1.3. This positions the solid block cursor in the upper left position of the screen. The program then goes to the subroutine labeled Padr. This subroutine prints the two byte address held in location 00E7 and 00E8. This address is later used to find the memory location you want to display.

The program continues by jumping to another subroutine. This one called Inadr. This subroutine reads the keyboard four times, allowing the input of a four digit memory address. The resulting two byte address is put into locations 00E7 and 00E8 (ADR). Once the four numbers have been entered, the program jumps back to the supervisor which again homes the cursor. The program then jumps to another subroutine labeled Pline.

Up to this point you probably will not be modifying the program very much as the function of the subroutines are very straightforward. However, Pline will probably need to be modified as it prints the start address of the desired memory locations and then prints the eight bytes of data contained in those locations. The number of data bytes printed is what may need to be changed, if you have not added the video modifications to your Superboard. On a 24 by 24 video display you can only print up to five bytes at a time unless you don't mind the information falling off one line and continuing on the next. The number of bytes printed is controlled by the byte at 1E24. In the assembly listing you can see that the program between 1E22 and 1E30 is concerned only with checking to see if Pline has printed eight characters and, if it has, to return back to the supervisor. To change the number of bytes printed per line, it is necessary to put the desired number at 1E23 and a copy of that number at 1E2B.

Now that Pline is finished and we have returned to the supervisor, you can see that Hexdump checks the keyboard and, if a carriage return is entered (ASCII value of 0D), it will branch back to print the next eight bytes of memory. If a line feed is entered (ASCII value of 0A), it will branch back to the part of the program which homes the cursor and starts it all again.

Next Month

We now have a program which is the first part of a very advanced monitor program. What comes next is two other routines which include the functions listed in our outline. They are smaller routines than Hexdump. The resulting Super-Monitor will allow you to enter large programs in a single bound.

```
                            ;This program uses some subroutines from
                            ;Super-Cursor V1.3  (COMPUTE! Nov. '81)
                            ;
                            ;Zero page usage is limited to only
                            ;two bytes-
                            ;   00E7 - ADR
                            ;   00E8 - ADR+1
                            ;which are the low and high bytes forming an
                            ;address for which HEXDUMP looks into memory.
                            ;
                            ;
*=1D20                      ;Start of program and entry point.
                            ;
1D20 20 80 1E    HEXDUMP JSR HOME           ;Home the Cursor
1D23 20 3C 1D    DIA     JSR PADR           ;Print address
1D26 20 80 1E            JSR HOME           ;Home the cursor again
1D29 20 93 1D            JSR INADR          ;Input address
1D2C 20 00 1E    DAL     JSR PLINE          ;Print one line
1D2F 20 BA FF    BD      JSR KEYIN          ;Reads the keyboard result in A
1D32 C9 0D               CMP #$0D           ;Key pressed = CR?
1D34 F0 F6               BEQ DAL
1D36 C9 0A               CMP #$0A           ;Key pressed = LF?
```

OSI TRS-80 COLOR-80 OSI

GALAXIAN - 4K - One of the fastest and finest arcade games ever written for the OSI, this one features rows of hard-hitting evasive dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. Specify system — A bargain at $9.95 OSI

LABYRINTH - 8K - This has a display background similar to MINOS as the action takes place in a realistic maze seen from ground level. This is, however, a real time monster hunt as you track down and shoot mobile monsters on foot. Checking out and testing this one was the most fun I've had in years! — $13.95. OSI

THE AARDVARK JOURNAL

FOR OSI USERS — This is a bi-monthly tutorial journal running only articles about OSI systems. Every issue contains programs customized for OSI, tutorials on how to use and modify the system, and reviews of OSI related products. In the last two years we have run articles like these!

1) A tutorial on Machine Code for BASIC programmers.

2) Complete listings of two word processors for BASIC IN ROM machines.

3) Moving the Directory off track 12.

4) Listings for 20 game programs for the OSI.

5) How to write high speed BASIC — and lots more —

Vol. 1 (1980) 6 back issues - $9.00

Vol. 2 (1981) 4 back issues and subscription for 2 additional issues - $9.00.

ADVENTURES!!!

For OSI, TRS-80, and COLOR-80. These Adventures are written in BASIC, are full featured, fast action, full plotted adventures that take 30-50 hours to play. (Adventures are interactive 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 8K on an OSI and 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 PLANT. 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.

DEATH SHIP (by Rodger Olsen)

Our first and original ADVENTURE, this one takes place aboard a cruise ship - but it ain't the Love Boat.

VAMPIRE CASTLE (by Mike Bassman)

This is a contest between you and old Drac - and it's getting a little dark outside. $14.95 each.

OSI NEW-NEW-NEW TINY COMPILER OSI

The easy way to speed in your programs. The tiny compiler lets you write and debug your program in Basic and then automatically compiles a Machine Code version that runs from 50-150 times faster. The tiny compiler generates relocatable, native, transportable machine code that can be run on any 6502 system.

It does have some limitations. It is memory hungry — 8K is the minimum sized system that can run the Compiler. It also handles only a limited subset of Basic — about 20 keywords including FOR, NEXT, IF THEN, GOSUB, GOTO, RETURN, END, STOP, USR(X), PEEK, POKE, -, =, *, /, , <,> , Variable names A-Z, and Integer Numbers from 0-64K.

TINY COMPILER is written in Basic. It can be modified and augmented by the user. It comes with a 20 page manual.

TINY COMPILER — $19.95 on tape or disk OSI

SUPERDISK II

This disk contains a new BEXEC* that boots up with a numbered directory and which allows creation, deletion and renaming of files without calling other programs. It also contains a slight modification to BASIC to allow 14 character file names.

The disk contains a disk manager that contains a disk packer, a hex/dec calculator and several other utilities.

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for —

SUPERDISK II $29.95 (5¼") OSI

BARE BOARDS FOR OSI C1P

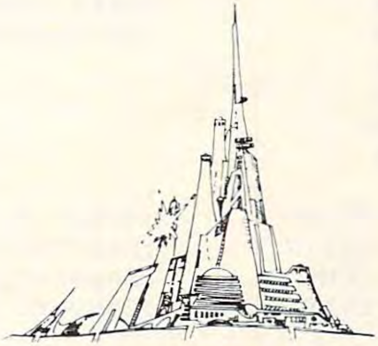
MEMORY BOARDS!!! — for the C1P — and they contain parallel ports!

Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

PROM BURNER FOR THE C1P — Burns single supply 2716's. Bare board — $24.95.

MOTHER BOARD — Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. - $14.95.

16K RAM BOARD FOR C1P — This one does not have a parallel port, but it does support 16K of 2114's. Bare Board $39.95.



WORD PROCESSING THE EASY WAY— WITH MAXI-PROS

This is a line-oriented word processor designed for the office that doesn't want to send every new girl out for training in how to type a letter.

It has automatic right and left margin justification and lets you vary the width and margins during printing. It has automatic pagination and automatic page numbering. It will print any text single, double or triple spaced and has text centering commands. It will make any number of multiple copies or chain files together to print an entire disk of data at one time.

MAXI-PROS has both global and line edit capability and the polled keyboard versions contain a corrected keyboard routine that make the OSI keyboard decode as a standard typewriter keyboard.

MAXI-PROS also has sophisticated file capabilities. It can access a file for names and addresses, stop for inputs, and print form letters. It has file merging capabilities so that it can store and combine paragraphs and pages in any order.

Best of all, it is in BASIC (0S65D 51/4" or 8" disk) so that it can be easily adapted to any printer or printing job and so that it can be sold for a measly price.

MAXI-PROS — $39.95. Specify 5¼ or 8" disk.

SUPPORT ROMS FOR BASIC IN ROM MACHINES — C1S/C2S. This ROM adds line edit functions, software selectable scroll windows, bell support, choice of OSI or standard keyboard routines, two callable screen clears, and software support for 32-64 characters per line video. Has one character command to switch model 2 C1P from 24 to 48 character line. When installed in C2 or C4 (C2S) requires installation of additional chip. C1P requires only a jumper change. — $39.95

C1E/C2E similar to above but with extended machine code monitor. — $59.95 OSI

ARCADE GAMES FOR OSI, COLOR-80 AND TRS-80 (8K OSI, 16K TRS-80 AND COLOR-80)

TIMETREK - A REAL TIME, REAL GRAPHICS STARTRECK. See your torpedoes hit and watch your instruments work in real time. No more unrealistic scrolling displays! $14.95.

STARFIGHTER - This one man space war game pits you against spacecruisers, battlewagons, and one man fighters, you have the view from your cockpit window, a real time working instrument panel, and your wits. Another real time goody. $9.95

BATTLEFLEET - This grown up version of Battleship is the toughest thinking game available on OSI or 80 computers. There is no luck involved as you seek out the computers hidden fleet. A topographical toughie. $9.95

QUEST - A NEW IDEA IN ADVENTURE GAMES! Different from all the others, Quest is played on a computer generated mape of Alesia. Your job is to gather men and supplies by combbat, 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

Please specify system on all orders

This is only a partial listing of what we have to offer. We offer over 120 games, ROMS, and data sheets for OSI systems and many games and utilities for COLOR-80 and TRS-80. Send $1.00 for our catalog.

AARDVARK TECHNICAL SERVICES, LTD.
2352 S. Commerce, Walled Lake, MI 48088
(313) 669-3110

OSI COLOR-80

```
1D38 F0 E9              BEQ DIA
1D3A D0 F3              BNE BD
                                          ;
1D3C A5 E8     PADR     LDA ADR+1         ;Print address on screen
1D3E 29 F0              AND #$F0          ;Start with high 4 bits of
1D40 4A                 LSR               ;ADR +1  (00E8)
1D41 4A                 LSR
1D42 4A                 LSR
1D43 4A                 LSR
1D44 20 72 1D           JSR CVHA          ;Convert Hex to Ascii
1D47 20 40 1E           JSR CURSOR        ;Part of Super-Cursor
1D4A A5 E8              LDA ADR+1         ;Now do the low 4 bits of ADR+1
1D4C 29 0F              AND #$0F
1D4E 20 72 1D           JSR CVHA
1D51 20 40 1E           JSR CURSOR
1D54 A5 E7              LDA ADR           ;Now work on high 4 bits of ADR
1D56 29 F0              AND #$F0
1D58 4A                 LSR
1D59 4A                 LSR
1D5A 4A                 LSR
1D5B 4A                 LSR
1D5C 20 72 1D           JSR CVHA
1D5F 20 40 1E           JSR CURSOR
1D62 A5 E7              LDA ADR           ;work on low 4 bits of ADR
1D64 29 0F              AND #$0F
1D66 20 72 1D           JSR CVHA
1D69 20 40 1E           JSR CURSOR
1D6C A9 2D              LDA #$2D          ;Print '-' on screen after address
1D6E 20 40 1E           JSR CURSOR
1D71 60                 RTS
                                          ;
1D72 A2 00     CVHA     LDX #$00          ;Convert whats in A from Hex to
1D74 8E 78 1D  CVST     STX CON+1         ;ASCII
1D77 C9 00     CON      CMP #$00          ;This value is changed in CVST
1D79 F0 04              BEQ CFIN
1D7B E8                 INX
1D7C 4C 74 1D           JMP CVST
1D7F BD 83 1D  CFIN     LDA CDATA,X       ;Put result in A
1D82 60                 RTS
1D83 30 31 32  CDATA    DATA              ;Data used in both CVHA and CVAHX
1D86 33 34 35           DATA              ;to convert Hex to Ascii and back.
1D89 36 37 38           DATA
1D8C 39 41 42           DATA
1D8F 43 44 45           DATA
1D92 46                 DATA
                                          ;
1D93 20 80 1E  INADR    JSR HOME          ;Input 4 digit (2 byte) Address
1D96 20 BA FF           JSR KEYIN         ;Read Keyboard Routine from ROM
1D99 20 40 1E           JSR CURSOR        ;Super-cursor
1D9C 20 F3 1D           JSR CVAHX         ;Convert Ascii to Hex
1D9F 0A                 LSL
1DA0 0A                 LSL
1DA1 0A                 LSL
1DA2 0A                 LSL
1DA3 8D AC 1D           STA INADC+1       ;Pokes A into 1DAC
1DA6 A5 E8              LDA ADR+1
1DA8 29 0F              AND #$0F
1DAA 18                 CLC
1DAB 69 00     INADC    ADC #$ 00
1DAD 85 E8              STA ADR+1
```

```
1DAF 20 BA FF           JSR KEYIN        ;Get second digit
1DB2 20 40 1E           JSR CURSOR
1DB5 20 F3 1D           JSR CVAHX
1DB8 8D C1 1D           STA INBDC+1      ;Pokes A into 1DC1
1DBB A5 E8              LDA ADR+1
1DBC 29 F0              AND #$F0
1DBF 18                 CLC
1DC0 69 00     INBDC    ADC #$00
1DC2 85 E8              STA ADR+1
1DC4 20 BA FF           JSR KEYIN        ;Get third digit
1DC7 20 40 1E           JSR CURSOR
1DCA 20 F3 1D           JSR CVAHX
1DCD 0A                 ASL
1DCE 0A                 ASL
1DCF 0A                 ASL
1DD0 0A                 ASL
1DD1 8D DA 1D           STA INCDC+1      ;Pokes A into 1DDA
1DD4 A5 E7              LDA ADR
1DD6 29 0F              AND #$ 0F
1DD8 18                 CLC
1DD9 69 00     INCDC    ADC #$00
1DDB 85 E7              STA ADR
1DDD 20 BA FF           JSR KEYIN        ;Get the last digit
1DE0 20 40 1E           JSR CURSOR
1DE3 20 F3 1D           JSR CVAHX
1DE6 8D EF 1D           STA INDCD+1      ;Poke A into 1DEF
1DE9 A5 E7              LDA ADR
1DEB 29 F0              AND #$F0
1DED 18                 CLC
1DEE 69 00     INDCD    ADC#$00
1DF0 85 E7              STA ADR
1DF2 60                 RTS
                                         ;
1DF3 A0 00     CVAHX    LDY#$00          ;Convert contents of A from
1DF5 D9 83 1D  CVCON    CMP CDATA,Y      ;Ascii to Hex
1DF8 F0 04              BEQ CVFIN
1DFA C8                 INY
1DFB 4C F5 1D  CVFIN    JMP CVCON
1DFE 98                 TYA              ;Put result in A
1DFF 60                 RTS
                                         ;
1E00 20 3C 1D  PLINE    JSR PADR         ;Print one line of eight bytes
1E03 A0 00              LDY #$ 00
1E05 B1 E7     PBYTE    LDA (ADR),Y      ;Print one byte from ADR
1E07 29 F0              AND #$F0
1E09 4A                 LSR
1E0A 4A                 LSR
1E0B 4A                 LSR
1E0C 4A                 LSR
1E0D 20 72 1D           JSR CVHA         ;Convert A to Ascii
1E10 20 40 1E           JSR CURSOR       ;Super-cursor
1E13 B1 E7              LDA (ADR),Y      ;Print low 4 bits on screen
1E15 29 0F              AND #$0F
1E17 20 72 1D           JSR CVHA
1E1A 20 40 1E           JSR CURSOR
1E1D A9 20              LDA #$20         ;Print a space " " to separate
1E1F 20 40 1E           JSR CURSOR       ;the bytes
1E22 C8                 INY              ;Are we finished?
1E23 C0 08              CPY #$08
1E25 D0 DE              BNE PBYTE        ;If not display another byte
```

```
1E27 A5 E7              LDA ADR        ;If yes add 08 to ADR
1E29 18                 CLC
1E2A 69 08              ADC #$08
1E2C 85 E7              STA ADR
1E2E 90 02              BCC RCOM
1E30 E6 E8              INC ADR+1
1E32 20 95 1E   RCOM    JSR CR         ;Were finished printing one line
1E35 20 AB 1E           JSR LF         ;so carriage return and line feed
1E38 60                 RTS
                                       ;
                                       ;
                                       ;
;Statistics
;
*=      1D20    ;Start
1D3C    PADR    ;Print ADR and ADR+1 on screen
1D72    CVHA    ;Converts Hex to Ascii
1D93    INADR   ;Input a two byte address for ADR and ADR+1
1DF3    CVAHX   ;Converts Ascii to Hex
1E00    PLINE   ;Print a line of 8 bytes from ADR
1E40    CURSOR  ;Prints what ever is in A to where the cursor is
1E80    HOME    ;Home the cursor
1EC2    CLS     ;Clear screen
```

©