November 1977 $1.50

ohio scientific's
SMALL SYSTEMS JOURNAL VOLUME 1 NO. 5

features page

The magazine for 6502 computer enthusiasts!

Model 510 Trade-In Offer

Ohio Scientific is offering a very unusual opportunity. We will allow owners of factory-assembled 400 CPU Boards to trade up to a fully assembled Model 510 CPU Board with the A-100 option. It comes fully equipped with 6502, Z-80, & 6800 microprocessors. It has a floppy disk bootstrap PROM, and a serial Monitor PROM for the 6800. The 510 features a crystal controlled clock, which can be operated at 4, 2, & 1MHz, and crystal controlled baud rate generation on its RS-232 and 20ma current loop ACIA port from 75 to 19,200 baud. The A-100 option adds a 16-line PIA port containing four additional address lines for a 1 Megabyte address space, and a software switch so that any of the three processors can be programmed to switch execution to any of the other processors, all under software control. The unit powers up as a normal 6502 system. This model also features a 128-byte scratchpad RAM, which is normally located at F200, but which can be swapped up to FF00, so that it can contain new restart and interrupt vectors for advanced programming topics. The unit can be supplied to you with a 65A Monitor PROM or 65V Monitor PROM (specify when ordering).

The list price for the C3-0 510 CPU Board and A-100 option is $509. We are offering a trade-in allowance of $150 on any original Challenger or Challenger I CPU Board in working condition, that is, any factory-assembled 400 CPU Board. This means that your total cost for a triple-processor CPU board with memory management will be $359 plus your old CPU Board.

In all honesty, deliveries on this trade-in program will be slow--typically sixty to ninety days after receipt of order, because it is being done simply as a special service to hobbyists. Delivery of CPU trade-ins will therefore have to take lower priority to that of systems which use the 510 CPU Board. We have accordingly designed a method by which you may keep your old 400 CPU Board until your Model 510 arrives.

Here is how to order. Specify Model 510 trade-in program with A-100 option, and specify 65A or 65V. Include with the order Master Charge or Visa payment for $359 plus four dollars shipping. Otherwise, pay by personal check or money order. Then make a second payment, which you are to specify as 400 CPU Board trade-in deposit, via Master Charge, Visa, or preferably personal check, for $150. When we receive your order, we will get to work on your 510 CPU Board. If you paid the first portion via Visa or Master Charge, this will not be charged against your account until the day of shipment of the 510 CPU Board. On personal checks, however, it will be necessary for us to wait until the check clears (three to four weeks, typically). In either case, however, we will not charge you or cash your check for the deposit on the CPU board. We will simply hold that as a deposit. You will then have sixty days after you receive your Model 510 CPU Board to send back your old Model 400 CPU Board. When we receive your old Model 400, we will return your deposit uncashed.

Hobbyists Apply Now!

update your old system with the model 510 CPU board at an irresistible discount price

***SAVE* $150**

for only $359 plus trade-in of any factory assembled 400 CPU board you will receive the amazing 510 triple-processor board *fully assembled!*

Introduction

As we near the end of the current year, there is quite a lot to report on. In this issue we are introducing a new Word Processor with a treasury of powerful commands for writing and editing text. We are also announcing the release of BASIC in ROM as a separate component for kit owners and an improved, nine-digit precision BASIC. Two video games are presented as well as an unusual type of holiday greeting card, through the courtesy of one of our readers. We have received a good response thus far on the questionnaire we printed last month, and hope that more of you will let us know what you are thinking. Also feel free to send in more programs. If you have some interesting, practical, and/or entertaining applications of your system, use this means to let your fellow Ohio Scientific users know about it.

Ohio Scientific
Small Systems Journal
Box 36
Hiram OH 44234.

Ohio Scientific's Small Systems Journal is issued monthly by Ohio Scientific Inc., P.O. Box 36, Hiram OH 44234. The subscription rate is six dollars per six issues. Individual copies are $1.50. Published in Twinsburg OH by the Twinsburg Bulletin.

Vol. 1, No. 5 November 1977
Editor-in-Chief Gary Deckant
Production Manager Don Muchow
Contributing Editors Mike Cheiky
Bob Coppedge
Eric Davis
Rick Whitesel

1K Corner

Cassette Loader and Memory Block Transfer

With this program the user may record his own programs via the 430B Super I/O Board in a format that is recognizable to the auto-load function in the 65V Monitor PROM. In addition, the program moves any specified block of memory to any other specified location without altering original data.

When the program is entered, a ? appears in the upper left corner of the screen. At this point the user keys either an S (=Save) or an M (=Move). The entry procedure is as follows:

S: ABCD, WXYZ>LMNO

where ABCD is the starting address of the memory to be loaded onto cassette, and WXYZ is the ending address. Note that the ending address itself is not saved. LMNO is the address at which the program will self-start. It it is not desired that the program be written for self-start, LMNO should equal FE00 so that the 65V Monitor will be reentered.

M: ABCD, WXYZ>LMNO

where ABCD is the starting address of the memory to be moved, and WXYZ is the ending address. Again, the data at the ending address itself is not moved. LMNO is the address at which transferred data is to start, i.e., data ABCD=data LMNO, data ABCD+1=data LMNO+1, etc.

After the three hex locations have been entered, the program waits for a G before executing a Save or a Move. Upon completion, a * appears to the right of LMNO.

Any time during address entry, or after *, a / will cause the program to restart and a ? will appear in the upper left corner of the screen.

The program resides from 0EE5 to 0FFF. However, the portion of the program from 0EE5 to 0F81 may be repositioned at any location, provided that the subroutines from 0F88 to 0FFF are not disturbed.

```
:U0EE5,0FFF
      5  6  7  8  9  A  B  C  D  E  F  0  1  2  3  4
0EE5  D8 A9 D0 85 FF A9 C5 85 FE A0 20 98 91 FE 88 D0
0EF5  F8 A9 3F 91 FE 20 ED FE 85 FD C9 53 F0 04 C9 4D
0F05  D0 F3 91 FE E6 FE A9 3A 91 FE A2 00 20 ED FE C9
0F15  2F F0 CD 20 93 FE 30 F4 20 DC 0F E6 FE 91 FE E8
0F25  E0 04 30 E8 D0 04 A9 2C 10 F1 E0 09 30 DE D0 04
0F35  A9 3E 10 E7 E0 0E D0 D4 20 ED FE C9 2F F0 A1 C9
0F45  47 D0 F5 A9 4D C5 FD F0 0F 20 88 0F A2 05 20 8A
0F55  0F A9 47 20 D1 0F 10 1C B1 F0 91 F4 E6 F4 D0 02
0F65  E6 F5 E6 F0 D0 02 E6 F1 A5 F1 C5 F3 D0 EA A5 F0
0F75  C5 F2 D0 E4 A0 02 A9 2A 91 FE 20 ED FE C9 2F F0
0F85  BC D0 F7 A2 01 A9 2E 20 D1 0F B5 F0 20 CA 0F B5
0F95  F0 20 CE 0F CA F0 F3 E0 03 F0 29 10 ED A9 2F 20
0FA5  D1 0F E8 A1 F0 20 CA 0F A1 F0 20 CE 0F A9 0D 20
0FB5  D1 0F E6 F0 D0 02 E6 F1 A5 F1 C5 F3 D0 E5 A5 F0
0FC5  C5 F2 D0 DF 60 4A 4A 4A 4A 20 F4 0F 48 AD 05 FB
0FD5  10 FB 68 8D 04 FB 60 48 A0 04 0A 0A 0A 0A 2A 26
0FE5  F4 26 F5 26 F2 26 F3 26 F0 26 F1 88 D0 F0 68 29
0FF5  0F 09 30 C9 3A 30 03 18 69 07 60 EA EA EA EA EA
```

Two New Software Packages

Ohio Scientific has just released two major new disk software packages specifically for floppy disk users. These two packages are our new Word Processor and 9-Digit BASIC running under OS-65D version 2.0.

Word Processor OS-WP1

Ohio Scientific's new Word Processor is a full text editor which operates at both the character and line levels. The editor is complete with disk files and has formatted printout, which makes it ideal for writing letters, manuals, reports, and all normal, everyday business forms. The new Word Processors's disk files are completely compatible with the disk files used for source code by the Assembler/Editor, so the Word Processor's powerful editing commands can be applied to Assembler source code. The Assembler can actually be linked to the Word Processor as an option. The Word Processor is specifically designed for secretarial or other nontechnical use. The Word Processor disk boots in automatically when D is typed, and asks "MEMORY SIZE?" The user simply types <return> (since the Word Processor will use all the memory available in the computer). The Word Processor features a very elaborate error-checking routine, which makes it virtually crash-proof. It instructs the operator as to what he did wrong via short messages. In order for you to gain an appreciation for the power of the Word Processor, let us discuss its five classes of commands. Not every command of the Word Processor will be covered here--simply its major commands.

LINE EDITING OPERATIONS

The Word Processor has several line number-oriented operations. Each line is assigned a number between 1 and 65,000. It is not necessary for a user to specify a line number when entering text or modifying text. He can avoid the use of line numbers by simply typing 0 for any line he enters. The Assembler will then automatically assume the line numbers are sequential after the last numbered line printed out. The user can print out any line or combination of lines by use of the P (print) command. By typing P<return> he will list the entire file in his workspace. Or he can specify certain lines by any of the following formulas:

mmm-nnn	lines from mmm to nn inclusive
mmm-	lines from mmm to end of file
-nnn	lines from the beginning of file to nnn
mmm	line mmm only

This allows the user to examine any section of the file. It even has an R<return> or RESEQ<return>, which causes a resequence of line numbers starting with line 10. An offset after the RESEQ command will cause the file to be resequenced with an offset, such that R1000 starts the resequencing with line 1000. After the resequencing is completed, the Editor will report the next available line number.

The line editing portion of the Editor has two additional extremely powerful commands, T (transfer) command and the M (move) command. The T command will transfer any block of lines to a new position specified after any given line, without eliminating the original block of lines. For example, a file may have twenty lines specified as lines 10-200. By use of the command T100-200,5 lines 100-200 will appear before line 10 as well as where they did originally, so that the file will now be thirty lines long. Line numbers are stripped off in the transferred file, so that these new lines each acquire a line number of 0. Via the RESEQ command the file can be resequenced such that new line numbers are reassigned to all lines. This command is useful when blocks, or portions of a file will be very similar. The user can transfer similar portions into different areas of the file, then use the powerful character-editing operations to make any slight changes necessary for the new portion.

The M command's function is identical to that of the T command, except that it eliminates the old entry and prevents the file from becoming any larger in size. It simply gets rearranged. The T and M commands are very valuable in putting form letters together and working with Assembler source files. A typical example of the great use of the T command would be to have a disk file containing 15 or 20 typical paragraphs used in letters. A secretary would then perform T commands to construct a given letter with lower line numbers, for instance, and then simply print out the lower line numbers as the letter, not destroying or eliminating the library of paragraphs for subsequent letters during that particular use period.

CHARACTER EDITING OPERATIONS

The Word Processor also has a powerful class of character-oriented commands. Two of these commands are the C (change) and F (find) commands. The C command is the same as the Edit/Replace commands in some timeshare systems. It allows you to change any string of characters, either on a single line, a block of lines, or in the entire working file. For instance the command C"e","ee"<return> will change the character e in the file to the character ee wherever it occurs. This command can be employed in form letters where only a person's or company's name may vary from letter to letter, for example. The C command can have an optional line specifier, so that if a single line or block of lines is specified, the command will operate only upon that line or block of lines. The C command is thus capable of both

local and global operations.

To complement the C command, the Editor also has an F command, which will find any string specified within quotes and print out all lines containing that string within the specified range. With the F command we can search for all occurrences of a string within a block of lines, or within the entire file. This would be useful when looking for a particular name or word which may have different meanings in different contexts. We would thus be able to change the word in only one of its contexts, if desired.

On all terminals the backspace command (shift-O) rubs out the last characters typed, so that commands and text can be corrected immediately after typos. The Editor also has a powerful E (edit) command which can be implemented on terminals that have cursor addressing. The E command is automatically configured for use on the Hazeltine 1500 or the Lear-Siegler ADM-3A with cursor addressing. The Word Processing Editor is optimized for use on the Hazeltine 1500 because of its key layout and automatic repeat features. It can however be utilized with other X-Y addressable terminals. Since virtually all terminals of this type have different protocols, or codes for character addressing, the Editor will have to be modified by the computer dealer and/or end user if it is used on a terminal other than Lear-Siegler or Hazeltine. As a matter of fact, both of these terminals have different codes, and the Editor automatically decides which type of terminal it is talking to by sensing the command utilized for non-destructive backspace. That is, when the Editor sees the Hazeltine's non-destructive backspace code, it decides that the terminal is a Hazeltine, and similarly in the case of a Lear-Siegler. The Editor changes internal parameters accordingly. X-Y addressing allows character insertion and deletion on a line basis. The user types an E, line number<return>. That line is printed out with the cursor located at the end of the line. The user may back up the cursor over characters and insert or delete characters as desired, and non-destructively forward-space. The line is instantly updated to show the character insertion or deletion. This is done without scrolling, so that the line simply appears to change as you type the characters. At any time the user can simply type <return> to re-install the lines as now shown on the screen into the workspace. Any portion of the line may be changed, including the line number, which will cause the line to be inserted at a different place without destroying the old line insertion.

This sophisticated form of character editing must really be seen to be appreciated. It is quite possibly the most convenient character-editing feature now available for use on microcomputers. Keep in mind however that this edit feature can only be used on CRT terminals that have X-Y addressing. If your terminal does not have X-Y addressing, the edit commands will be of no value to you.

Character editing without the E command is accomplished by use of the backspace when a typo is made. Additional editing may be achieved by simply retyping the number of a line containing an error, and typing the line again from the start. Whenever a line number is duplicated, the new line replaces the old line of the same number in the working file. The Word Processor has an I (initialize) command, which completely erases the working space, eliminating the possibility of conflict of line numbers when new text is to be entered. The I command requires you to answer Y (yes) to complete the initialization process. This guards against inadvertent erasing of the working space.

DISK COMMANDS

As stated above, the Word Processor is designed for use by inexperienced operators. Therefore it has its own internal GET and PUT file commands, which do not require the user to leave the Word Processor. The syntax for both the GET and PUT commands includes the naming of disk drive A or B and the starting track of the file. The Word Processor comes on a standard OS-65D diskette, where the first nine tracks are utilized by the Word Processor and the system. This leaves sixty-eight tracks available for files for the Word Processor, which should be placed in drive A of single or dual drive systems. Seventy-six tracks are available on drive B for use as Word Processor files. An individual Word Processor file is limited to the length or size of available memory. Word processing files start at approximately 3200 (hex), so that a 32K machine would be able to support an individual file of approximately 20,000 characters. A 48K machine would be able to support an individual file of about 36,000 characters. The GET and PUT commands transfer individual files from memory to disk and can handle files up to the 48K maximum configuration of the machine. When a person wishes to store a file on disk, he types the command PUT<return>. The Word Processor reports the length in tracks of the file in its workspace. The user must then find an appropriate free space on his disk and type in his selection of disk drive A or B, followed by the number of an available track. The file will then be stored on the disk, and the system will return to the Word Processor command mode.

To retrieve a file from disk, the user simply types GET<return>, followed by the drive (A or B) from which the file is to be retrieved, and the starting track of the file he wishes to input. The Disk Operating System (DOS) transfers that file in, and returns to the Word Processor command mode.

For advanced applications the user can type EXIT, which will exit the Word Processor and go to the DOS. The DOS is a standard OS-65D, which will allow the user to use the L and P formats, and S and C formats. It will enable him to manipulate the I/O Distributor to utilize the Editor under several different I/O configurations. The user can return to the Word Processor from the DOS by typing RA, without disturbing the file in its workspace. He can store 180,000 characters on the diskette containing the Word Processor, and a total of 209,000 characters on a diskette used in drive B of a

dual drive system. Thus, a typical diskette can store about fifty pages of ordinary single-spaced typewritten information.

FORMATTED OUTPUT COMMANDS

The Word Processor has a powerful formatted output mode which is similar to runoff programs found in many larger computer systems. The formatted output is based on the L (list) command. This command lists text in the workspace without line numbers in justified form. It is used to produce Ohio Scientific's Small Systems Journal and the Word Processor manual. It performs both left and right justification to a character width specified by the user. After typing L, the user can designate a justified text width from twenty to seventy characters. He can optionally add a line specifier so that he can limit the output of justified text to a certain block of lines. If he does not specify any lines, then the entire file is listed in justified format. The L function searches through text for the wordspace just under the total number of characters specified. It adds additional spaces between words on an equal basis to expand the line of text to the designated width. The user may force a carriage return by means of the pounds sign. A double pounds sign will force the end of justified output on one line and an additional carriage return--linefeed before text starts again. The effect of this will be a line skipped between paragraphs.

In addition to the justified output and line specifications, the user can specify page format by also typing an F (format), followed by a page number, in the line with the line command. The presence of a format subcommand in the line command causes the output to page, that is, place spaces between pages of text. It adds a page number in the bottom of each page, starting at the page number specified. If the command is F10, then the starting page number is 10, and so on, up to 65,000 pages. If 0 is given as the page number, then the output will page, that is, skip lines between blocks of output, but will not assign page numbers. The formatted output is coordinated to an internal page counter in the Word Processor, which performs the skip function for eleven-inch-long paper on line printers which do not have skip units (e.g., the popular DEC writer, or OKI-DATA 110 line printer). Thus the software performs the skip for these printers. The paging counter can be disabled for line printers such as the OKI-DATA O-22, which have electronic skip units in them, so the formatted output is fully compatible with line printers, whether or not they have internal skip capability. Finally, the user can type an O in the list command, which will turn on the line printer at the beginning of output. The line printer is then turned off immediately upon completion of the list, so that no control characters are ever echoed to the line printer. This will enable the user to construct an entire book on the line printer, if desired, without requiring any cutting and paste-up, since no extraneous commands or characters will be printed.

To review the line command, let's consider a hypothetical situation. Take the line L50,200-1000,F10,O<return>. This command causes a justified listing fifty characters wide, beginning at text line 200. The output will be paged, starting at page 10, and the output will appear both on the CRT terminal and the line printer or word processing printer. This is obviously a very powerful command.

PRINTER CONTROL COMMANDS

The Word Processor has a full set of printer control commands which can be used with virtually any impact or matrix computer printer or word processing printer. As mentioned above, the F and O sub-commands, which are part of the L command, operate the printer. The Word Processor also contains an S (slew) command, which will cause the printer to slew n number of lines. For instance, if we type S40, the printer will slew forty lines. Typing S0 will cause a top-of-form command to be issued, so that the printer will have form and skip controls. It will cause homing to the new top of form. The Word Processor has three additional formatting commands specifically for line printers. These are set up as shipped to be commercial-at B, commercial-at H, and commercial-at W. These three commands, when encountered in an L mode output, with the line printer, are designed to invoke special formatting in line printers. They are currently set up on the OKI-DATA Model O-22 line printer, where commercial-at W will cause the printer to go into double-width character mode. Commercial-at H will cause it to go into double-height character mode. Commercial-at B will cause it to go into double-width, double-height character mode. The Word Processor comes with documentation to change these three commands to perform similar functions on other sophisticated line printers.

* * * *

The Word Processor package is available with an optional Assembler fully integrated into the package. This Assembler has the standard OSI 6502 Assembler directives, A, A1, A2, & A3. These are fully integrated into the command loop of the Word Processor. Minimum requirements to run the Word Processor system are a Model 400, 500, or 510-based 6502 system, with serial port and serial terminal located on the ACIA at FC00, a single or dual drive floppy disk, and a minimum of 16K of memory. The recommended system for using the Word Processor is a 32K Challenger II or Challenger III with single or dual drives, and a Hazeltine Model 1500 CRT and OKI-DATA Model O-22 line printer. The Word Processor system is delivered with its commands set up for these two peripherals. The Word Processor can be used directly with the Lear-Siegler ADM-3A with direct cursor addressing, without software modifications of the Word Processor package. The system can be directly adapted to the OKI-DATA Model 110 line printer by simply

activating the internal page counter, and can be adapted to virtually any other conventional CRT terminal and line printer or word processor printer via the documentation provided. It is recommended, however, that the user have both the CRT terminal and line printer with upper- and lower-case characters, to make full use of the Word Processor. To make use of the powerful E command, it is necessary to have a terminal with X-Y cursor addressing.

NEW SOFTWARE PRICES

WP-1 WORD PROCESSOR

SOFTWARE PACKAGE, TWO DISKETTES AND MANUAL

$79

WP-1A

AS ABOVE, WITH INTEGRATED 6502 ASSEMBLER

$99

OS-65D VERSION 2.0

WITH NINE-DIGIT BASIC, TWO DISKETTES AND MANUALS (BE SURE TO SPECIFY NINE-DIGIT)

$49

Nine-Digit Precision BASIC

Ohio Scientific is releasing a nine-digit precision BASIC integrated into OS-65D Version 2.0. This nine-digit BASIC is functionally equivalent to the normal six-digit precision BASIC, with a few minor exceptions. Even with these exceptions, the new BASIC should run existing programs identically to the older BASIC with an improvement in numerical accuracy, without requiring any modification to the six-digit BASIC's programs.

The nine-digit BASIC is a version recently issued to us by Microsoft and is believed to be very highly debugged, since it takes into account a year's experience with the older six-digit precision BASIC. The only operational difference in the nine-digit precision BASIC is that it allows the user to list with a line specification. In other words, he can list up to and including a line, from a given line number on, or a specified block of lines. This is not possible in the older version of BASIC. Since the nine-digit precision BASIC has greater numerical accuracy, numerical arrays take considerably more space than with six-digit BASIC. For this reason the nine-digit BASIC also has the capability of storing integer arrays which are specified in DIM statements and array statements by the percent sign. This facilitates a tremendous savings in memory when small constants are required in an array. All other commands, including LOAD, SAVE, USR(X), and interfaces to the OS-65D operating system, are the same as before. The nine-digit precision BASIC is slightly larger in size than the older six-digit BASIC. The initialization package was trimmed to a minimum to allow the nine-digit BASIC to fit within the framework of OS-65D Version 2.0.

When this BASIC comes up, it simply reports "OK," no longer reporting "MEMORY SIZE" or "TERMINAL WIDTH." The initialization package of nine-digit BASIC is invoked only when BASIC is initially started or re-started by the VB command. The initialization package includes a one-byte location which specifies the terminal width of the output. This is normally set to seventy-two characters. It also contains two locations which specify the upper memory limit. If these two locations are zero, BASIC will automatically make use of all available memory in the system. If it is not zero, it will utilize these locations for memory size. Since these locations are used only during initialization, it is possible for a program to come up in some standard configuration, POKE these locations to a new value, and re-start BASIC. Thus although the new initialization package is shorter, it is much better suited to business applications, for instance, where we do not want the ultimate end user to have to make decisions about memory size allocated for BASIC, or the terminal width of the computer. The new initialization package also facilitates dynamic changes in memory size, for example, during the course of executing a business software package.

TWO NEW VIDEO GAMES

SAM (Surface-to-Air Missile)

SAM recreates a hypothetical conflict between an attacking air force and a defending missile installation. You will be controlling the missiles, and the computer randomly controls the number of planes flying overhead. Your task is to shoot down as many as possible. You accomplish this by depressing one of three keys, causing missiles to be fired at various angles. The missile-launching keys and their corresponding angles are as follows:

1) 63.4 degrees
2) 45.0 degrees
3) 26.6 degrees

The launch pad appears in the lower left corner of the screen. The game is time dependent, and score is kept, with possible bonuses. In the future this game may have two-player abilities, allowing the program to be changed to fit a user's whims.

The weaponry in this game is visually represented as follows:

```
launch pad        [ ]
                  XXXXX
enemy aircraft     <
SAM                *
```

The game has some catches. For example, you can fire only one missile at a time. This could be changed by using arrayed "shots" and a subroutine for missile progress (movement). Also there is some slagging of plane and missile movement when you have too many moving characters on the screen. You can minimize this problem by using a USR(X) system.

You can vary this program by incorporating any of the following features:

1) have randomly changing altitudes for the airplanes; mid-air collisions will occur.

2) have the planes attack the missile installation; this would also involve changing altitudes, but they would not be totally random.

3) have the planes attack from both sides of the screen.

4) add sound effects via a PIA.

To start the game, you need to type in a random number (seed)--an instruction to this effect appears on the screen. Then be ready for some fast action. This game should either entertain you or drive you insane, depending on how well you do.

Requirements of the game are:

1) Ohio Scientific's 8K BASIC by Microsoft (cassette, paper tape, disk, or ROM version)

2) OSI's 440 Video Board at DXXX with a keyboard at DFFF

3) at least 4K of RAM at 0000 up, in addition to BASIC in ROM or RAM

Modules of the Program

Lines	Function
5-101	Instructions and seed input
102-103	Clears the screen
105-106	Builds the missile's housing
115-140	Sets up initial starting point of new plane
141-148	Checks above values for redundancy
160-172	Command determination
173-185	Plane's movement
188	Timer
700-703	Determines mode (angle) of fire
710-800	Firing sequence
732-736	Hit checking
900-1020	Hit sequence (scoring, etc.)
1050-1080	Bonus sequence
1100-end	Score determination, end messages

```
LIST

 1 REM**ROBERT L.
 2 REM**COPPEDGE
 5 PRINT:PRINT:PRINT
 10 PRINT"THE '<'S ARE PLANES..."
 20 PRINT"SHOOT THEM DOWN BY"
 30 PRINT"PRESSING A '1','2', OR A"
 35 PRINT"'3'. TIME LIMITED."
 90 PRINT:PRINT:PRINT:PRINT"GOOD SHOOTING!!":PRINT:PRINT:PRINT
 100 PRINT"INPUT RANDOM SEED";
 101 INPUT Z1
 102 FOR P=53250 TO 54200
 103 POKE P,32:NEXT P
 105 POKE 54121,93:POKE 54119,91
 106 FOR L=1 TO 5:POKE 54149+L,88:NEXT L
 115 D=INT(3*RND(Z1)+1)
 120 FOR L=1 TO D
 130 A(L)=53406+INT(15*RND(Z1)+4)*32
 135 B(L)=A(L)
 140 NEXT L
 141 FOR M=1 TO D
 142 FOR N=M+1 TO D
 143 IF A(M)=A(N) THEN 120
 144 NEXT N
 148 NEXT M
 160 F=PEEK(57343)
 165 IF F=179 GOTO 172
 166 IF F=177 GOTO 172
 167 IF F=178 GOTO 172
 168 GOTO 173
 172 GOTO 700
```

```
173 FOR L=1 TO D
177 B(L)=B(L)-1
180 IF B(L)<=A(L)-32 THEN B(L)=A(L)
183 POKE B(L)+1,32
184 POKE B(L),40
185 NEXT L
186 GOTO 160
188 T6=T6+1:IF T6=300 GOTO 1100
190 GOTO 160
301 GOTO 900
700 A=1
701 IF F=177 THEN A=2
702 B=1
703 IF F=179 THEN B=2
710 F1=54088
720 FOR V=1 TO 20
725 POKE F1,32
730 F1=F1-32*A+B
731 FOR J=1 TO D
732 IF F1=B(J) OR F1=B(J)+1 OR F1=B(J)+2 THEN 900
733 IF F1=B(J)-1 OR F1=B(J)-2 THEN 900
734 IF F1=B(J)+31 OR F1=B(J)+32 OR F1=B(J)+33 THEN 900
735 IF F1=B(J)-31 OR F1=B(J)-32 OR F1=B(J)-33 THEN 900
736 IF F1=B(J)+64 OR F1=B(J)-64 THEN 900
740 NEXT J
741 IF B=2 AND F1<=53700 THEN V=20
745 IF B*V>28 THEN V=20
749 POKE F1,42
750 FOR K=1 TO D
751 POKE B(K),32
752 B(K)=B(K)-1
767 POKE B(K),40
768 IF B(K)=<A(K)-32 THEN B(K)=A(K)
770 NEXT K
790 T6=T6+1
791 IF T6=300 THEN 1100
800 NEXT V
846 POKE F1,32
850 GOTO 173
900 FOR S9=1 TO 60
901 POKE F1,S9
902 NEXT S9:POKE F1,32
903 POKE B(J),32
904 IF D=1 GOTO 911
905 FOR X=J TO D-1
906 B(X)=B(X+1)
907 A(X)=A(X+1)
908 NEXT X
910 B(D)=0:A(D)=0
911 B7=B7+1:D=D-1
1015 IF D=0 GOTO 115
1020 GOTO 846
1050 FOR L=1 TO D:POKE B(L),32:NEXT L
1051 POKE F1,32
1052 PRINT"BONUS TIME"
1053 FOR N6=1 TO 1000:NEXT N6
1055 B6=B6+B7:B7=0
1056 FOR L=1 TO D:POKE B(L),32:NEXT L
1057 POKE F1,32
1060 FOR N8=54080 TO 54145
1065 POKE N8,32:NEXT N8
1080 T6=0:GOTO 105
1100 IF B7>15 GOTO 1050
1101 FOR N2=1 TO D
1102 POKE B(N2),32:NEXT N2
1103 POKE F1,32
1104 PRINT"YOUR SCORE IS ";:PRINTB6+B7
1110 PRINT"YOU'RE FINISHED!!"
1120 PRINT"(BUT....UH..WANTA TRY"
1130 PRINT"AGAIN?(YES,NO?)";
1140 INPUT A$
1145 B6=0:B7=0
1150 IF A$="Y" OR A$="YES" THEN 100
1160 END
```

Bomber

Bomber is an enjoyable recreation of another computer game, but this one is a little harder to outsmart. In this situation, a terrorist group has placed a bomb in a large parking garage. Everyone knows it's in there, but no one knows exactly where. You are asked to go in and defuse the bomb. It is assumed that once you find the bomb, you can defuse it with no trouble. You are represented on the screen by an O. You enter the parking garage with a device that registers the time remaining before the bomb detonates and the distance between you and the bomb. However, it does not tell you in which direction to look. You may have to move to different floors by means of the elevators, denoted X. Take care not to fall through the windows, and stay away from the boundary of the playing area. The movement of the figure O is controlled by the following keys:

1) down
2) up
3) left
4) right
5) stop
6) elevator ascent
7) elevator descent

The computer gives a constant read out of the following variables:

1) time remaining before the bomb blows (starts at 60 seconds)
2) the number of the floor you're on (26 floors in all)
3) the distance to the bomb

For best results, depress firmly on the key in use. When you have located the bomb, the program prints a message of congratulation. If you fail to find the bomb, you get blown to kingdom come.

The requirements of this program are the same as those of the SAM game above. As in SAM, you are required to enter a seed, i.e., a random group of numbers, prior to start of the game.

Modules of the Program

Lines	Function
1-98	Instructions and random seed input
100-102	Clears screen
103-112	Prints visual display
113-117	Determines random bomb position
200-500	General game sequence
250-357	Movement (via PEEK)
270-275	Upward movement
280-285	Downward movement
290-295	Left movement
300-305	Right movement
310-311	Stop and defuse bomb
320-325	Elevator ascent
330-334	Elevator descent
350-356	Position check
357	POKEs O
400-419	Prints data via POKEs
550-570	Loser statements
700-725	Winner statements
1000-1051	Mistake statements

```
A*RB
OK
LIST

 1 REM** R. L. COPPEDGE
 2 REM** O.S.I. 28-11-77
 10 PRINT"DO YOU NEED INSTRUCTIONS"
 11 PRINT"(YES OR NO)";
 12 INPUT A$
 13 IF A$="NO" GOTO 97
 14 PRINT"YOU BETTER RUN THIS ON      SAVE"
 15 PRINT"HOKAY...SOME TERRORIST"
 16 PRINT"HAS PUT A BOMB IN A VERY"
 17 PRINT"LARGE PARKING GARAGE, AND"
 18 PRINT"AND FOR SOME REASON THEY"
 19 PRINT"HAVE CHOSEN YOU TO"
 20 PRINT"REPRESENT THE 'PEOPLE'"
 21 PRINT"TO TRY TO DIFFUSE IT. "
 22 PRINT"YOU ARE GIVEN A BOX WHICH"
 23 PRINT"TELLS YOU THE DISTANCE, "
 24 PRINT"BUT NOT THE DIRECTION. "
 25 PRINT"IN THIS DIAGRAM ON MY"
 26 PRINT"FACE, THE X'S ARE"
 27 PRINT"ELEVATORS, THE BOUNDARIES"
 28 PRINT"WALLS.  DON'T JUMP!!"
 29 PRINT"TO MOVE ONE'S SELF, ONE"
 30 PRINT"MUST PUSH A: "
 31 PRINT:PRINT:PRINT"1) TO MOVE DOWN"
 32 PRINT:PRINT"2) TO MOVE UP"
 33 PRINT:PRINT"3 GOES LEFT"
 34 PRINT:PRINT"4 RIGHT. "
 35 PRINT:PRINT"A '5' STOPS YOU. "
 36 PRINT:PRINT"A '6' GOES UP IN"
 37 PRINT"THE ELEVATOR"
 38 PRINT:PRINT"AND A '7' GOES DOWN IN"
 39 PRINT"ONE. (PROVIDING THAT YOU"
 40 PRINT"ARE SITTING ON ONE!)"
```

```
41 FOR I=1 TO 300
42 NEXT I
97 PRINT"INPUT RANDOM SEED";
98 INPUT Z1
100 FOR X=53284 TO 54260
101 POKE X,32
102 NEXT X
103 PRINT"DISTANCE:":PRINT"FLOOR:    TIME:";
104 FOR X=53380 TO 53402
105 POKE X,45:POKE X+672,45
106 NEXT X:Q3=1
107 FOR X=1 TO 21
108 POKE 53380+(32*X),9
109 POKE 53402+(32*X),9
110 NEXT X:Q=54000
111 POKE 53413,24:POKE 53433,24
112 POKE 54021,24:POKE 54041,24
113 X=INT((628*RND(Z1))+53413)
114 IF X-53413-INT((X-53413)/32)*32>20 THEN 113
115 X3=INT(26*RND(Z1)+1)
116 X1=INT((X-53413)/32)
117 X2=X-32*X1
200 FOR C=120 TO 1 STEP -1
201 Q1=INT((Q-53413)/32)
202 Q2=Q-32*Q1
210 D=((X1-Q1)↑2+(X2-Q2)↑2+(X3-Q3)↑2)↑(1/2)
211 D(1)=INT(D/10)
212 D(2)=INT(D-D(1)*10)
213 D(3)=-2
214 D(4)=INT((D-INT(D))*10)
215 D(5)=INT((D*10-INT(D*10))*10)
250 Y=PEEK(57343)
255 Y1=Y-176
256 IF Y1<1 THEN 350:IF Y1>7 THEN 350
260 ON Y1 GOTO 270,280,290,300,310,320,330
262 GOTO 350
270 POKE Q,32:Q=Q+32
275 GOTO 350
280 POKE Q,32:Q=Q-32
285 GOTO 350
290 POKE Q,32:Q=Q-1
295 GOTO 350
300 POKE Q,32:Q=Q+1
305 GOTO 350
310 IF D=0 GOTO 700
311 GOTO 350
320 IF Q=53413 THEN Q3=Q3+1
321 IF Q=53433 THEN Q3=Q3+1
322 IF Q=54041 THEN Q3=Q3+1
323 IF Q=54021 THEN Q3=Q3+1
325 GOTO 350
330 IF Q=54021 THEN Q3=Q3-1
331 IF Q=54041 THEN Q3=Q3-1
332 IF Q=53413 THEN Q3=Q3-1
333 IF Q=53433 THEN Q3=Q3-1
334 GOTO 350
350 IF Q>54041 THEN 1000
351 IF Q<53413 THEN 1000
352 IF Q-53413-32*INT((Q-53413)/32)>20 THEN 1000
353 IF Q3>26 THEN 1050
354 IF Q3<1 THEN 1050
355 POKE 53413,24:POKE 53433,24
356 POKE 54041,24:POKE 54021,24
357 POKE Q,15
400 R1=INT(Q3/10)
401 R2=Q3-R1*10
402 N=C/2
404 N(1)=INT(N/100):N(2)=INT((N-N(1)*100)/10)
405 N(3)=N-10*INT(N/10)
406 FOR M=1 TO 3
407 B=54131+M
```

delay

clear

display

display elevator

Bomb

movement

elevator up

elevator DN

Position check

elevator jammed

Q3 = Level

Q = position of man

X3
X1
X2

D = distance

D( )

Keyboard

Down

Up

Left

Right

Stop - check if Hit

R1
R2

N1
N3

C = Time

```
408 POKE B,N(M)+48
409 NEXT M
410 FOR M=1 TO 5
411 E=54093+M
412 POKE E,D(M)+48
415 NEXT M
416 S(1)=INT(Q3/10)
417 S(2)=Q3-10*S(1)
418 POKE 54123,S(1)+48
419 POKE 54124,S(2)+48
500 NEXT C
550 FOR G=1 TO 96
551 POKE X,G
552 NEXT G
553 FOR H=54084 TO 54300
554 POKE H,32
555 NEXT H
568 PRINT:PRINT"YOU BLEW IT...OR IT BLEW"
569 PRINT"YOU....SORRY."
570 GOTO 1100
700 FOR F=54084 TO 54300
710 POKE F,32
711 NEXT F
720 PRINT:PRINT"WELL DONE!!"
721 PRINT"TIME LEFT:::";
722 PRINTN;:PRINT"  SEC.  APPROX."
725 GOTO 1100
1000 PRINT:PRINT"YOU RAN OUT OF THE WINDOW"
1001 GOTO 1100
1050 PRINT:PRINT"ELEVATOR JAMMED"
1051 PRINT"TURKEY!!"
1100 END
OK
```

Time

distance

floor

Bang

Lose

Hit

---

648 bytes

SEASON'S GREETINGS

Gentlemen:

Somewhere in your Small Systems Journal [p. 8, September 1977] you ask for user programs that use the PEEK and POKE instruction. In the spirit of the season I enclose a tape with a program which makes use of the POKE instruction in, what I think, might be an unusual way.

Best regards,
Erich A. Pfeiffer
Box 2624
Sepulveda CA 91343

NOTE: This program will only run on systems containing the 440 Video Board.

```
5 REM*****************
7 REM  COMPUTER X-MAS
9 REM:::::::::**************************:::::::::::
10 DIM A$(20):GOSUB 500
20 FOR I=1 TO 100:GOSUB 800:NEXT
30 DATA"  MERRY CHRISTMAS   "
40 DATA"     HAPPY HANUKKAH"
50 DATA" AND A GOOD NEW YEAR"
60 DATA"    TO OSI AND      "
70 DATA"TO ALL COMPUTER NUTS"
80 DATA"ALL OVER THE WORLD  "
90 DATA"                    "
100 FOR I=1 TO 7
110 READ M$
120 FOR J=1 TO 20:POKE 53733+J,ASC(MID$(M$,J,1)):NEXT J
130 FOR K=1 TO 3
140 FOR L=1 TO 10:GOSUB 700: NEXT L
150 GOSUB 800
160 NEXT K
170 NEXT I:RESTORE
180 GOTO 100
500 FOR I=1 TO 28:PRINT:NEXT:RETURN
600 R=RND(1)*1024+53248
620 RETURN
700 GOSUB 600:POKE R,32:RETURN
800 GOSUB 600:POKE R,42:RETURN
1000 END
OK
```

601 IF R>53732 AND R<53754 Then 600

ASCII Files under OS-65D

BASIC programs are stored in the machine's memory in tokenized form. That is, all the BASIC's key words or reserved words are reduced to a single byte token. Therefore the actual core image or memory image of a BASIC program is not very recognizable when listed. Likewise, the Editors of the Assembler and the Word Processor WP-1 store source and text in memory in a compacted form. Line numbers are reduced to their hexadecimal equivalent, while repeated characters such as spaces are compacted to a two-byte code, in which one byte is the character itself, and the other byte is the number of times the character is successively repeated. Neither one of the Editors, nor BASIC stores sources in conventional ASCII code, nor are they compatible with each other. It would be desirable, particularly with our new Word Processor to be able to apply high level editing to BASIC programs. It would also be desirable to merge modules of BASIC programs and Assembler source codes together as well as other operations. All this can be accomplished by the creation of a secondary ASCII source file in memory, which can also be stored on disk if desired. This secondary ASCII file capability can be added to OS-65D Version 2.0 with a fairly short patch. However, it is important that the user be fully familiar with the Disk Operating System before attempting to utilize the secondary or indirect ASCII file system. Otherwise he may become bewildered in its operation.

Here is how it works. This modification provides three additional global commands to the operating system, which are usable in all software utilizing the system's input and output subroutines. The commands are: [ ]control-X. When the user types in a [, he turns on a status flag, resets the memory output pointer to a pre-determined value, and starts outputing to a memory file as well as to the screen. All subsequent output going to the terminal is also being loaded into a memory file in pure ASCII form. This will proceed until the user types a] which turns off the status flag and turns off the output to memory. Everything between [and] that is outputed to the terminal is also stored in pure ASCII form in a memory file. This file will be retrieved if the user types a control-X. This turns the status flag back on , converts the input from the terminal to memory, and sets the memory pointer to point to the start of the indirect ASCII file. The file then provides input until the] is encountered in the memory file, at which time the flag is turned off and the input routine is returned to the terminal, restoring control to the user. In this way, the user can create an indirect ASCII file in memory and play back that indirect file as input to any program, or to the system, for that matter.

APPLICATION

To understand the operation of the ASCII files, let us examine an actual application. First of all the user must always keep in mind the amount of absolute memory available. Consider a case where the user has a total of 32K memory, and has a BASIC program on which he would like to do extensive editing. He would like to use the new Word Processor/Editor. He decides to allocate 26K of the machine for the operating system and primary file, and subsequently 6K for his indirect ASCII file. This means that he will have to set the variable PNTH located at 2158 in the indirect ASCII file overlay to the value 68 (hex), which sets the indirect file pointers for memory file pointers to the 26th K up. Then when he initializes BASIC, he will have to answer something less than 26624 to MEMORY SIZE? to make sure that his BASIC file does not overwrite his indirect ASCII file. The user then brings in his program of interest via conventional means into BASIC workspace. To load the program into the indirect ASCII file, he simply types LIST [<return>. The program will list out onto the terminal and also be loaded into memory. At the end of the list, he simply types] <return>. The program will output]]<return>, placing one] in the indirect ASCII file. He now has the ASCII listing of his program in memory from 26000 up. He can then bring in his Editor, making sure that its upper memory limit is set to something less than 26K, initialize its workspace, and simply type a control-X <return>. The indirect ASCII file will now input to this Editor, just as if someone were keying it in at the terminal. The indirect file handler will encounter the] in the memory file and exit this routine, returning control to the user, with the ASCII file now loaded into the Editor's workspace as in any other program. The program can then be manipulated as desired, and when a final satisfactory version is obtained, the program can be re-installed in BASIC by the same approach. The user will then type a P[<return>, at which time the program would then list out with line numbers onto the terminal and into the indirect ASCII file. At the end of the listing he would type], terminating the indirect file. He would then bring BASIC back in, taking care not to overwrite the beginning of his indirect file by specifying a memory size less than the beginning of the indirect file. Once in BASIC, he would simply type control-X <return>, and again, the indirect file would load into BASIC as if it were being entered onto the terminal, and would automatically exit when it encountered a], leaving the revised program in memory. Indirect ASCII files can be stored on disk by using the SAVE and CALL commands and storing the absolute memory on disk.

The indirect files are extremely powerful tools for building or merging programs, text, and Assembler source code, since the control-X feature can be invoked at any time. Take, for example, a case where a user would like to build BASIC programs in modular form. He simply assigns unique sets

of line numbers to each module of his program, say lines 10000 to 19000 for input, 20000 to 29000 for calculations, and 30000 to 39000 for disk operations. He then creates indirect ASCII files for each module of code and stores these on disk. At a later date, he wishes to simply install the disk module in a new program on which he is working. He can do this as long as the module has unique line numbers by simply having the rest of the program in the workspace, and bringing in the indirect ASCII files containing the module he wishes to merge and typing a control-X. This will load the module desired into the existing program just as if it were typed on the keyboard.

INSTALLATION

To add indirect ASCII files to your system, follow these steps. Start with a clean copy of OS-65D which has nothing important on it, so that an inadvertent erasure of files will not result in any great loss. Bring in the Assembler, and key in the source code as listed here. Do an A1 Assembly to clean out any typos. Then leaving the source code in the workspace, exit the Assembler and call in track 04 by using the C command <C1800,=04,1>. This will call one portion of BASIC into memory which will not conflict with your Assembly. However, it does wipe out the Extended Monitor so that you will not be able to execute the Extended Monitor while this one BASIC track is in memory. Then return to the Assembler and perform an A3 Assembly, which will assemble the program in memory. Most of the program will be residing in disk on track 04, with just a few bytes at the end residing on track 05. Once the Assembly is complete, exit the Assembler, type S04,1 1800/B. This resaves the third section of BASIC on the disk with the indirect file patch. Then type S05,1 2200/3. This saves the I/O drivers with their patch. You should now have the indirect files completely stored on disk, and those capabilities should be present whenever you put the system in, both in the Assembler and in BASIC. To utilize indirect files under the Word Processor, follow the instructions in the Word Processor manual for activating the indirect file commands.

* * * *

It will be necessary for you to modify the variable PNTH, based on the size of files you wish to manipulate, and the amount of memory you have in your system. Also, do not modify all your diskettes for the indirect ASCII file until you become very familiar with it, because there are several problems you can have if you do not operate it correctly. Also, if your keyboard appears to have no [or] then use shift-K for [and shift-M for].

```
.A

  10 0000                ;INDIRECT ASCII FILE OVERLAY TO OS65D V2.0
  20 0000                ;MCC 11-29-77
  30 0000                ;FOR ACIA PORT ONLY
  40 0000                ;
  50 0000                     INEXIT=$230B
  60 0000                     NXT2=$2E77
  70 0000                ;
  80 0000                     POLL=$22D1
  90 0000                     INFG=$2203
 100 0000                     OTFG=$2204
 110 0000                ;
 120 2158                     *=$2158
 130 2158 60        PNTH     BYTE $60
 140 2159 00        FLAG      BYTE 0
 150 215A 297F       TSTCN AND #$7F
 160 215C C95D             CMP #$5D
 170 215E D00C             BNE CNTX
 180 2160 AD5921           LDA FLAG
 190 2163 C900             CMP #0
 200 2165 D055             BNE RESTOR
 210 2167 A95D             LDA #$5D
 220 2169 4C0B23           JMP INEXIT
 230 216C C918      CNTX   CMP #$18
 240 216E D016             BNE TSTCD
 250 2170 A908             LDA #$08
 260 2172 8D0322           STA INFG
 270 2175 EE5921           INC FLAG
 280 2178 AD5821           LDA PNTH
 290 217B 8D682E           STA $2E68
 300 217E A900             LDA #0
 310 2180 8D672E           STA $2E67
 320 2183 4CD122           JMP POLL
 330 2186 C95B      TSTCD CMP #$5B
 340 2188 D016             BNE RETN
 350 218A A911             LDA #$11
 360 218C 8D0422           STA OTFG
 370 218F EE5921           INC FLAG
 380 2192 AD5821           LDA PNTH
 390 2195 8D552E           STA $2E55
 392 2198 A900             LDA #0
 395 219A 8D542E           STA $2E54
 400 219D 4CD122           JMP POLL
 410 21A0 4C0B23    RETN   JMP INEXIT
 420 21A3                   ;EXIT TEST ON OUTPUT
 430 21A3 C95D       OUTEST CMP #$5D
 440 21A5 F003             BEQ OUTT
 450 21A7 4C772E           JMP NXT2
 460 21AA A900      OUTT   LDA #0
 470 21AC 8D5921           STA FLAG
 480 21AF A901             LDA #1
 490 21B1 8D0422           STA OTFG
 500 21B4 A981             LDA #$81
 510 21B6 8D0322           STA INFG
 520 21B9 4C772E           JMP NXT2
 530 21BC A95D      RESTOR LDA #$5D
 540 21BE 206D22           JSR $226D
 550 21C1 A901             LDA #1
 560 21C3 8D0322           STA INFG
 570 21C6 8D0422           STA OTFG
 580 21C9 A900             LDA #0
 590 21CB 8D5921           STA FLAG
 600 21CE A95D             LDA #$5D
 610 21D0 4C0B23           JMP INEXIT
 620 21D3              OVERLAYS
 630 22DA                   *=$22DA
 640 22DA 4C5A21            JMP TSTCN
 650 232C                   *=$232C
 660 232C 4CA321            JMP OUTEST
 670 232F                    END
```

BASIC in ROMs

Ohio Scientific has just released its popular BASIC in ROM set as a component set for the computer builder. The BASIC in ROM set is composed of four 2616 2K x 8 mask ROMs, which contain our ultra-fast 8K BASIC for the 6502, authored by Microsoft. In addition to these four ROMs, the set contains a 1702 support EPROM containing 256 bytes of code. The four mask ROMs reside at A000 to BFFF and the support EPROM resides at FF00 to FFFF. The 8K of ROM contains complete 8K BASIC plus I/O drivers, such as ACIA I/O, audio cassette I/O, and a 440 video display I/O. The support EPROM contains loops from the BASIC to the appropriate I/O, as well as restart vectors for the system, a jump-to-monitor EPROM. It includes parameters which set up the start of the user workspace, the USR(X) function and other parameters of the BASIC, which one may desire to change from system to system. By use of this support EPROM, it is possible for the BASIC ROMs to be general purpose in nature and be compatible with several different system configurations. The 8K BASIC in ROM set is configured to start the user workspace, i.e., the workspace used by BASIC programs themselves, at 0300 (hex) up. The BASIC in ROM set makes extensive use of page 00 and page 01 for fast variables and the subroutine stack, but leaves most of page 02 empty for user machine code. The 8K BASIC in ROM fully supports audio cassette I/O based on a 430B I/O Board for both program and data storage, by use of LOAD and SAVE commands. The 8K BASIC in ROM set is specifically designed for use on the Model 500 CPU Board, which is now available as a bare board and manual, as a 504A Kit, or a 504V Kit. BASIC in ROM sets are available in two versions: video system 65VB for use on video systems utilizing the Model 440 Video Display Board for keyboard input and video display, along with an audio cassette for optional program and data storage; and serial system 65AB, which makes use of the Model 500's ACIA-based serial port for I/O. Both versions also support a 430B Board-based audio cassette interface. Either version sells for $99 retail together with the four mask ROMs, the support EPROM and manuals on BASIC, information on the subroutines and structure of the support EPROM. Delivery is approximately thirty days after receipt of order. The video ROM set supports the 65V EPROM Monitor, which is accessed by typing M in response to the message that appears after the computer is reset. Likewise, the 65A Monitor can be accessed from a serial set by typing M after the message appears following reset. These ROM sets will therefore not interfere with manual machine code operation of your computer system. Be sure to specify the serial ROM set Model 65AB or the video ROM set Model 65VB when ordering.

NOTE: The current ROM sets are not compatible with floppy disk systems. In the first place, BASIC in ROM does not have disk I/O capabilities; secondly, BASIC in ROM support EPROMs reside at the same address as the floppy disk bootstrap PROM.

Since virtually all important support parameters, such as subroutine calls, the start of user workspace as well as reset and interrupt vectors, are in 1702 EPROM, it is possible for a user to reprogram his 1702 support EPROM to customize the operation of his ROM set to some extent. Information about the operation of the support EPROM is included with the ROM set for such purposes.

BASIC in ROM Set

65AB including

Four ROMs, One EPROM

and manuals for

Serial 6502 System

$99

65VB including

Four ROMs, One EPROM

and manuals for

440-Video-based System

$99

Ohio Scientific 11679 Hayden Hiram, OH 44234
SMALL SYSTEMS JOURNAL

HIRAM OHIO
First Class Mail
U. S. POSTAGE
PAID
PERMIT 12

OHIO SCIENTIFIC
11679 HAYDEN STREET HIRAM, OHIO 44234