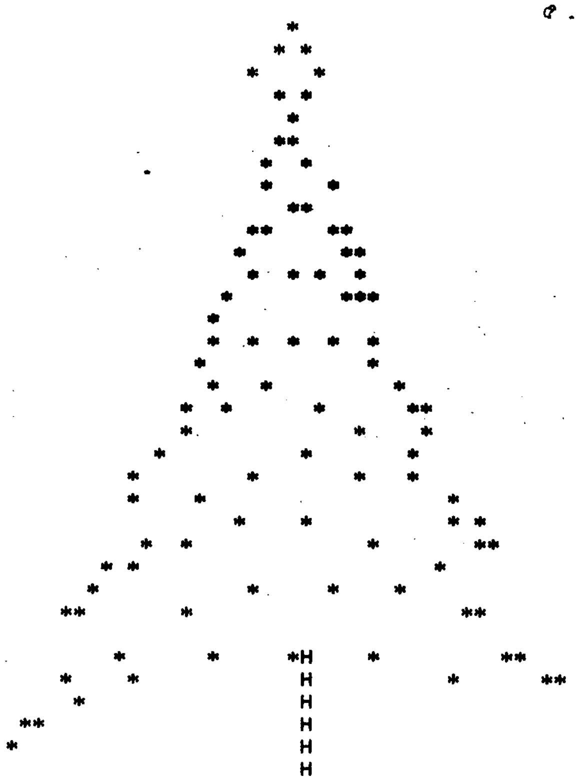
PEEK (65)

The Unofficial OSI Users Journal

$1.75
December 1982
Vol. 3, No. 12

P.O. Box 347
Owings Mills, Md. 21117
(301) 363-3268



Column One

This is a hard column to write. Not because the news is so bad, but because I don't know exactly what to say. I'll try my old fallback position when confused: say what I know and let the chips fall.

As most of you know, PEEK(65) has been owned and published by the Baltimore area OSI Dealer, an outfit called DBMS, Inc., since it's inception. In fact, I founded both, though I now own neither.

DBMS has fallen on hard times, while PEEK(65) has continued, as a labor of love, to usually make ends meet. The result is that DBMS has gone out of business, and PEEK(65) will now be published by PEEK(65), Inc. In some ways, that is a better arrangement, since it makes the magazine really and completely independent (in fact as well as spirit) from OSI and all OSI dealers and distributors. But when you put as much time as I have into a company like DBMS, you hate to see it go down when the economy stagnates...

Please be assured that PEEK(65) has not been zapped by Ray-gunomics. As long as you, our faithful readers, keep subscribing, and writing articles and letters, and you, our faithful advertisers, keep buying ads, we will soldier on. I like to think we are even more needed in these hard times.

We have recently received another one of those letters. You know the ones... "you always write all about the ABC system, why don't you write more about the XYZ system?" In this case, the gentleman was wanting more articles for the polled keyboard hobby systems, less C3 and business articles. Okay, sir, have a look at this issue.

Steve Hendrix continues his excellent series on the Basic-in ROM interpreter. This series should be treasured by all hackers who want to understand what really goes on in their machines, by all assembly programmers who would like to interface with the many K of code already resident in their machine, and by all hobbyist computers in general. We have two more sections in this excellent series in hand, and will publish them in coming issues. Congratulations, Steve, on an excellent series.

Just in case large system users might think we are abandoning THEM this month, have a look at the "Denver Board" ad. I am not sure if you realize what this means. Quite simply, we OSIers now have available to us the most advanced method of multi-user computing -- multiprocessing.

Multiprocessing means that each and every user has his own computer, his own CPU chip and RAM, which he shares with no one else. This in turn means that no matter how many users are up and running, there is virtually no speed degradation, since each new user gets his own computer. HOWEVER, each new user need not buy the really expensive stuff like hard disk drives -- that is shared on the system bus.

In the Denver Board system, each user also has an extension of the 48-pin bus, to plug in (for example) a printer driver board, so that EACH USER could have an independent printer, or modem, or whatever, the operation of which will not slow down or disturb the other users. Sounds like the best of both worlds. We are anxious to hear from a user of this board just how well it works!

[illegible]

THE WORKINGS OF BASIC
THE IMMEDIATE MODE

by: Steven P. Hendrix
Route 8, Box 81E
New Braunfels, TX 78130

This month's routine begins to tie together some of the earlier pieces covered in the earlier columns. The immmediate mode processor is the main loop of the Basic interpreter, calling directly or indirectly every other routine in the interpreter. The main part of this routine lies in the Basic 1 ROM at $A24E thru $A34A.

The first section of this block is not part of the immediate mode proper, but interacts closely enough to cover as a part of this column. The section from $A24E thru $A272 handles error messages, falling through into the immediate mode processor. Upon entry to the error handler at $A24E, the X register contains an error number, which will always be an even number on a stock C1P. Your machine language routines may call this code if you wish to have them return to BASIC after an error and if you can make a suitable error code.

The LSR at $A24E is actually used simply to clear bit 7 of the page zero location $64, which is the ctrl-O flag. When you type ctrl-O, this bit is toggled on and off. The character output routine in another area of Basic tests this flag; if it is cleared (0), characters are printed normally; if it is set (1), no action is taken for attempted character output. This results in the desired action for ctrl-O, namely that it turns output on and off. Your machine language programs may toggle this flag to disable output and keyboard echo, but the INPUT routine and the immediate mode processor both

Copyright ©1982 by **PEEK (65)** Inc. All Rights Reserved.
published monthly

Editor - Al Peabody
Technical Editor - Brian Hartson
Circulation & Advertising Mgr. - Karin Q. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates

US (surface)	$15
Canada & Mexico (1st class)	$23
So. & Cen. America (Air)	$35
Europe (Air)	$35
Other Foreign (Air)	$40

All subscriptions are for 1 year and are payable in advance in US Dollars.
For back issues, subscriptions, change of address or other information, write to:
PEEK (65)
P.O. Box 347
Owings Mills, MD 21117
Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsements of the product or products by this magazine or the publisher.

LISTING 1

THE IMMEDIATE MODE PROCESSOR

```
$A24E LSR $64         ; ENABLE OUTPUT
$A250 JSR $A86C       ; CR-LF
$A253 JSR $A8E3       ; PRINT "?"
$A256 LDA $A164,X     ; 1ST CHAR
$A259 JSR $A8E5       ; PRINT IT
$A25C LDA $A165,X     ; 2ND CHAR
$A25F JSR $A8E5       ; PRINT IT
$A262 JSR $A491       ; CLEAR STACKS
$A265 LDA #$86        ; PRINT " ERROR"
$A267 LDY #$A1
$A269 JSR $A8C3
$A26C LDY $88         ; CHECK FOR IMMEDIATE
$A26E INY
$A26F BEQ $A274       ; YES
$A271 JSR $B953       ; PRINT "IN <line number>"
$A274 LSR $64         ; ENABLE OUTPUT
$A276 LDA #$92        ; PRINT "OK"
$A278 LDY #$A1
$A27A JSR $0003
$A27D JSR $A357       ; ACCEPT INPUT
$A280 STX $C3
$A282 STY $C4
$A284 JSR $00BC       ; GET 1ST CHAR
$A287 BEQ $A27D       ; NULL LINE
$A289 LDX #$FF        ; FLAG IMMEDIATE MODE
$A28B STX $88
$A28D BCC $A295       ; STARTS WITH DIGIT
$A28F JSR $A2A6       ; COMPRESS TEXT
$A292 JMP $A5F6       ; AND EXECUTE IT
                      ; RETURNING TO IMMEDIATE MODE
$A295 JSR $A77F       ; GET LINE NUMBER
$A298 JSR $A3A6       ; COMPRESS REMAINING TEXT
$A29B STY $5D         ; NUMBER OF BYTES
$A29D JSR $A432       ; FIND LINE WITH SAME NUMBER
$A2A0 BCC $A2E6       ; NOT FOUND
$A2A2 LDY #1          ; SET UP $71-$74
$A2A4 LDA ($AA),Y     ; AND X FOR MOVE
$A2A6 STA $72
$A2A8 LDA $7B
$A2AA STA $71
$A2AC LDA $AB
$A2AE STA $74
$A2B0 LDA $AA
$A2B2 DEY
$A2B3 SBC ($AA),Y
$A2B5 CLC
$A2B6 ADC $7B
$A2B8 STA $7B
$A2BA STA $73
$A2BC LDA $7C
$A2BE ADC #$FF
$A2C0 STA $7C
$A2C2 SBC $AB
$A2C4 TAX
$A2C5 SEC
$A2C6 LDA $AA
$A2C8 SBC $7B
$A2CA TAY
$A2CB BCS $A2D0
$A2CD INX
$A2CE DEC $74
$A2D0 CLC
$A2D1 ADC $71
$A2D3 BCC $A2D8
$A2D5 DEC $72
$A2D7 CLC
$A2D8 LDA ($71),Y     ; MOVE THE REST OF
$A2DA STA ($73),Y     ; THE PROGRAM DOWN
$A2DC INY
$A2DD BNE $A2D8
$A2DF INC $72
$A2E1 INC $74
$A2E3 DEX
$A2E4 BNE $A2D8
$A2E6 LDA $13         ; CHECK FOR A LINE DELETION
```

reset it to normal printing. You might use this for a machine language prologue to accept a password for use of a program, without echoing it to the screen.

The JSR at $A250 outputs a carriage-return line-feed pair, and handles housekeeping such as the POS counter at the same time. The JSR at $A253 prints a question mark (?) for the beginning of the error message. I see no particular usefulness to the question mark nor do I see that it clarifies the error message at all. This might be changed to call some other, more useful routine.

The code from $A256 prints the two-character error code, based on the value in the X register. The error codes are stored from $A164 thru $A185. X is used as an index into this table; hence, X will take on only even values since each code takes up two bytes. In each of the character pairs, the second character has bit 7 set, resulting in printing a letter and a graphic character for the error messages. Masking off bit 7 (as would be done by a 7-bit interface to a terminal) yields the following error codes, which seem to make sense and are easier to understand than the actual error codes.

```
$A2E8 BEQ $A319      ; YES - SKIP INSERTION
$A2EA LDA $85        ; CLEAR STRING SPACE
$A2EC LDY $86
$A2EE STA $81
$A2F0 STY $82
$A2F2 LDA #7B        ; SET UP POINTERS FOR BLOCK
$A2F4 STA $A6        ; MOVE TO MAKE SPACE FOR
$A2F6 ADC $5D        ; THE NEW LINE
$A2F8 STA $A4
$A2FA LDY $7C
$A2FC STY $A7
$A2FE BCC $A301
$A300 INY
$A301 STY $A5
$A303 JSR $A1CF      ; CALL BLOCK MOVE ROUTINE
$A306 LDA $7F        ; CORRECT THE END-OF PROGRAM PTR
$A308 LDY $80
$A30A STA $7B
$A30C STY $7C
$A30E LDY $5D        ; INSERT THE LINE
$A310 DEY
$A311 LDA $0F,Y
$A314 STA ($AA),Y
$A316 DEY
$A317 BPL $A311
$A319 JSR $A477      ; CLEAR VARIABLES, STACKS, ETC.
$A31C LDA $79        ; GET START OF PROGRAM POINTER
$A31E LDY $7A
$A320 STA $71        ; SAVE IN TEMPORARY
$A322 STY $72
$A324 CLC
$A325 LDY #1         ; END OF PROGRAM ?
$A327 LDA ($71),Y
$A329 BNE $A32E      ; NO
$A32B JMP $A27D      ; TO IMMEDIATE MODE
$A32E LDY #4         ; SKIP 1ST 4 BYTES
$A330 INY
$A331 LDA ($71),Y    ; END OF LINE ?
$A333 BNE $A330      ; NO
$A335 INY
$A336 TYA            ; COMPUTE START OF NEXT LINE
$A337 ADC $71
```

REPLACE UP TO 6 OSI* BOARDS WITH MEM+. SAVE ROOM. SAVE POWER. SAVE MONEY.

Now you can have the memory and peripherals you want with out sacrificing valuable backplane space or overloading your power supply.

* MEM+ (56K all options) replaces these OSI[†] products for $675:
3 520 16K memory boards
1 CM10 8K memory board
1 CA9 CENTRONICS
1 470 Disk Controller

FEATURES:

- Up to 64K low power static RAM.
- These memory chips use 40 times less power than chips used on 24K boards by OSI[†] and D&N.[††]
- Divided into 3 16K blocks + 2 individually addressable 4K or 8K blocks
- 2716 EPROM plug-in compatible.
- OSI compatible floppy disk controller 8 or 5¼, single or double sided.
- CENTRONICS Printer Interface
- Real time clock calendar.
- 10 year lithium battery back up.
- Accurate to 1/1000 sec.
- Versatile programmable interrupts.
- 1 year full warranty.

[†]OSI is a trademark of MA/COM Office Systems Inc.
[††]Trademark of D&N Micro Products Inc.

00319 20700

Generic Computer Products

MEMORY OPTIONS

16K	$275
24K	$325
32K	$370
40K	$410
48K	$450
56K	$490
64K	$530

PERIPHERALS

DISK CONTROLLER (specify 5¼ or 8, single or double sided.)	add $95
CENTRONICS PORT	add $45
CLOCK CALENDAR	add $45

VISA, MASTER CARD, checks, money orders and c.o.d.s accepted. Add $5 per board shipping and handling. For more information contact:

FIAL COMPUTER
11266 S.E. 21st Ave
Portland, Oregon 97222
(503) 654–9574

"Computer Business Software"
"CBS"

BUSI-CALC

"The Businessman's Calculator"

Do you want the power
of an electronic worksheet
without giving up your hard disk
and multi-user capabilities?

BUSI-CALC FEATURES

Local and General Formatting
Replication
Variable Column Widths
Editing
Insertion/Deletion of Rows and Columns
Protected Entries
Help Screen
Flexible Printing
Complete User Manual

Busi-Calc is available for
M/A Com OSI Business Computers.

MICROSOFTWARE
INTERNATIONAL

3300 South Madelyn, Sioux Falls, South Dakota 57106
1-800-843-9838

```
$A339 TAX
$A33A LDY #0        ; AND STORE IN CURRENT LINE'S PTR
$A33C STA ($71),Y
$A33E LDA $72
$A340 ADC #0
$A342 INY
$A343 STA ($71),Y
$A345 STX $71       ; AND IN THE TEMPORARY PTR
$A347 STA $72
$A349 BCC $A325     ; UNCONDITIONAL BRANCH
```

NF -	Next without For
SN -	SyNtax
RG -	Return without Gosub
OD -	Out of Data
FC -	Function Call
OV -	OVerflow
OM -	Out of Memory
US -	Undefined Statement
BS -	Bad Subscript
DD -	Doubly Dimensioned
/0 -	Divide by zero
ID -	Illegal Direct
TM -	Type Mismatch
LS -	Long String
ST -	String Temporaries
CN -	ContiNuation
UF -	Undefined Function

Note that by setting the value of X to anything from 0 to 255, you can select from 256 possible different error messages. Not all of the possible settings yield letters, but by choosing odd values of X you can create error messages from the standard list, using the second letter of one message and the first letter of the next. You can also use letters from some text which is stored immediately after the error messages. All that is necessary is for your routine to load X with the appropriate pointer to the two characters, and to mask off bit 7 when printing the characters. This is how HEXDOS generates its error messages.

The JSR $A491 at $A262 clears the machine's hardware stack and Basic's string stack, and does some other minor housekeeping to insure that Basic will start up properly. Notice that this is NOT done for a warm start, which enters the routine a little further along. This is why the first command to Basic after a warm start yields an error message: the Reset routine points the hardware stack pointer to an area which is off limits to Basic, causing an OM ERROR. After trapping that error, this routine is called, resetting the stack pointer into Basic's permitted range for the stack ($0133 - $01FE). This routine should perhaps have been moved to the beginning of the warm start, where it would be called even without an error.

$A265 thru $A267 preload the A and Y registers for the very useful routine called by the JSR $A8C3 at $A269. This routine prints a string stored in memory, starting with the character whose address is given by Y (high byte) and A (low byte), and ending with an ASCII NUL ($00) character. In this case, the string to be printed starts at $A186 and $A18B (an ASCII blank and the word ERROR). This routine is called many places throughout the Basic interpreter.

The code at $A26C thru $A270 tests to see if the line which caused the error was typed as an immediate line from the keyboard, versus a line which was part of a program. The byte at $88 is the high byte of the current line number, which is set to $FF for an immediate mode line. The branch at $A26F is taken if it is an immediate line, skipping the JSR $B953 at $A271, which prints the word "IN" and the line number, with appropriate spacing. That routine is also used by the "BREAK IN xxxx" message.

Now for the real meat of the immediate mode processor, starting at $A274. A cold start sets up a JMP at $0000 to this location, and then jumps through that JMP. A warm start jumps directly to $0000, so you can store any code there you wish to define the results of a warm start. The first instruction at $A274 is a LSR $64, used for the same purpose as the identical instruction at $A24E. Next comes the setup for and call of the routine which prints the "OK" prompt. $A276 thru $A279 load the Y and A registers as for the earlier "ERROR" string. The JSR $0003 at $A27A jumps to RAM, which was set up in the cold start to jump to the "print-string" routine at $A8C3. This is a prime spot to intercept the immediate mode processor for your own purposes, such as a locked end-user system. All Basic routines eventually return and come back through the JSR, whether they end successfully or cause an error.

The JSR $A357 at $A27D calls the line input routine discussed last month. As you will recall, this routine accepts keyboard input, echoes it to the screen, stores it in a buffer at $0013 and handles backspace and line-cancel functions. It returns with X and Y holding a pointer to the byte before the buffer, high part in Y and low part in X. $A280 thru $A283 store this pointer in the "current byte" pointer of Basic's GETBYTE routine, discussed a few months ago. The JSR $00BC at $A284 calls that routine, returning in this case the first character in the terminal buffer with appropriate flags.

The first task after receiving the input line is to determine if it was a null line. If it was, the first character will be a $00, and the JSR $00BC above will have returned with the Z flag set. The BEQ at $A287 branches back to get another line if this is the case, without repeating the OK prompt. However, if you will recall from the discussion of the GETBYTE routine a few months ago, that routine also sets the Z flag if the character returned is a colon, so that the Z flag is set in either case for the end of a Basic statement. In this case, this ambiguous signal causes any immediate line starting with a colon to be totally ignored (try it!).

Next, the routine sets a flag to indicate that the line which will shortly be executed was entered in the immediate mode and is not resident in a program as a program line. This Basic disallows INPUT statements and DEF statements in immediate lines. The reasons for this become clear if you think through the mechanisms which will be required to implement these statements. The INPUT statement will load the characters of the response into the terminal buffer, destroying the remainder of the INPUT statement and any statements which were input with it, causing the whole process to crash. The DEF statement will need some provision for saving the text of the function definition, and there are no provisions for assigning memory for this purpose. If either of these statement types are encountered in a program line, there is no conflict: in the case of the INPUT statement, the buffer is

MULTI-PROCESSING
with the Denver Board

The Denver Board (Model DB-1) is an assembled and tested terminal expansion circuit board for expanding terminal usage on any Ohio Scientific, Inc. (OSI*) Series C2 and C3 computer system. The DB-1 is designed to reduce terminal speed loss from 80 to 90 percent when two or more terminals are added to the computer. Each terminal is also provided with an additional 16K bytes of memory.

Each DB-1 comes with a full 90-day parts and labor warranty, and a factory repair/exchange program is also available should a DB-1 that is out of warranty ever need servicing.

FEATURES

- 64K Bytes Random Access Memory (RAM)
- One Programmable Read Only Memory (PROM) for BUS arbitration and interprocessor communications.
- Six light emitting diodes (LED's) for power, master BUS indicator, transmit and receive.
- Automatic system boot switch.
- Auxiliary BUS for expansion printer I/O circuit board.
- Four reset modes:
  Power-on reset.
  Master reset (front panel).
  Individual reset from terminal with BREAK key.
  Individual reset from DB-1 with pushbutton switch.
- Memory expansion capability of 4K bytes common memory using standard OSI memory expansion circuit board.

SOFTWARE

95 percent of existing OS-65u* software is compatible with the DB-1. An OSI operating system patch program is supplied on 8-inch floppy disk as required. The patch program is copied to the user disk that contains the OSI operating system; and when the computer is turned on, the patch program will automatically tie-in.

for more information call or write:

DBi, inc.
p.o. box 7276
denver, co 80207
(303) 364-6987

Dealer Inquires Invited

* OSI and OS-65u are trademarks of M/A-COM Office Systems, INC.

not used to store the program line; in the case of the DEF statement, the text is already assigned a permanent place in memory (as a part of the program line). The flag is set by crashing the "current line number" stored from the last operation. The LDX - STX combination at $A289 - $A28C sets the high-order byte of that stored line number to $FF, and since that results in a line number which is greater than 65279 and the largest permitted line number is 63999, there can be no confusion between an immediate line and a high-numbered program line.

It would appear at first glance that the BCC at $A28D is testing for some result of the last operation, setting the immediate mode flag. In fact, however, neither LDX nor STX affect the C flag. The last operation which could have affected that flag took place in the GETBYTE routine, where the C flag indicates whether the byte returned was an ASCII digit. In this case, the purpose should be apparent after a moment's thought: if the first byte of the input line is a digit, we are typing a line to be entered in a program, and if it is not a digit, we are typing a line of instructions to be executed in the immediate mode. If the carry is set (meaning the first byte is not a digit), the routine falls through to $A28F; if not, it branches to $A295.

The first thing that happens if the branch is not taken (the first character was not a digit) is that the line is compressed, converting it to a form more useable to Basic. The routine at $A3A6 (called by the JSR at $A28F) scans the buffer, searching for matches to its table of Basic reserved words. If it finds a match, it replaces the word with a single byte which represents that word. Then, during interpretation, the interpreter has an easier time recognizing its reserved words, and since the line may be interpreted many times but will be compressed only once, the interpreter will run faster. This also saves some storage space, since each full word is represented by only one byte.

The compression routine also handles the conversion of a question mark to the PRINT byte. Since this is stored the same way regardless of whether you type a question mark or the word PRINT, there is nothing gained by using the question mark - it is merely a typing convenience, since PRINT is used so often in most Basic programs.

Returning to the next instruction after calling the compression routine, we find a JMP to another routine. Notice this is a jump with no possibility for a return. In fact, the routine to which this jumps at $A5F6 is the major executive routine for running a Basic program, breaking each line up into individual statements and calling the appropriate routines to carry out the desired functions. If that routine runs into the end of the program it does a jump back to the start of the immediate mode, returning control to the terminal. That routine will be the subject of next month's column.

If the line did start with a digit, the interpreter will instead branch to $A295, which is the line delete/insert routine. Starting there, the JSR $A77F converts the line number to an internal form as a two-byte integer, with the high byte stored at $0012 and the low byte stored at $0011. Then the JSR $A3A6 compresses the text of the line as before (the line number was discarded by $A77F, so only the text of the line is considered). When this routine returns, it leaves the number of bytes in the line in the Y register. This value is then saved at $005D for later processing.

The JSR $A432 at $A29D is a line-search routine. It seeks the line whose number is given by the contents of $0011-$0012, leaving a pointer to the beginning of the line in $00AA-$00AB. It returns with the C flag set if the line was found, and cleared if it was not. The logic used by that routine can cause some problems if you merge programs or otherwise tamper with Basic's internal storage. It starts at the beginning of the program, comparing each line number with the number at $0011. If the line number is less, the search continues with the next line. If it is equal, the search is complete and the line has been found. If it is greater, the search is complete and the line does not exist. This logic works fine for programs entered normally from the keyboard or tape, because Basic inserts each line in numerical order. Those of you with disk, however, especially those merging programs by the methods suggested for use with HEXDOS, can possibly get line numbers entered out of sequence by merging a program after another with line numbers which are lower than some line numbers in the first program. Now consider the result of the above logic in such a case, where the line numbers are not strictly increasing: the routine may not be able to find lines which appear after higher-numbered lines. Since this routine is used also by the GOTO, GOSUB, and IF-THEN constructs, your program logic can behave strangely.

The BCC at $A2A0 is testing to see if the line was found. If the line exists, we must delete it before continuing; if not, we may jump directly to the insertion routine at $A2E6. Notice that the line will be deleted if it exists before considering the contents of the new line; thus, the logic does not need anything special to contend with deleting a line by just typing the line number with a carriage-return.

The code from $A2A2 thru $A2D7 is rather messy and difficult to understand. If you wish to fully understand it, try walking through it with various programs. The general logic of it is to use the "next line" pointer contained in the first two bytes of the stored line to determine how much of the program to delete. It leaves a pointer to the deleted line at $0073, and a pointer to the beginning of the remainder of the program at $0071, with a number in X which determines how much of higher memory is part of the program. The number is the number of bytes divided by 256, rounded upward. This information is then used by the code from $A2D8 thru $A2E5 to move the remainder of the program downward in memory, overwriting the deleted line. This process leaves the links from each line to the next messed up, but that is dealt with later in the routine.

The line insertion routine lies from $A2E6 thru $A318. It starts by checking for a null line (line number followed immediately by carriage-return). In that case, $0013 will contain a zero, and the BEQ at $A2E8 is taken, jumping to the pointer-correction routine at $A319. $A2EA thru $A2F0 de-allocates all string space by setting the string space pointer ($0081-$0082) to match the end-of-memory pointer ($0085-$0086). The code from $A2F2 thru $A302 copies the end-of-program pointer ($007B-$007C) to a temporary pointer at $00A6-$00A7, and also adds the number of bytes in the new line (from $005D) to it to determine the new end-of-program, storing the result at $00A4-$00A5. The address of the line which should follow the new line is already stored at $00AA-$00AB, and now we call $A1CF with the JSR at $A303.

$A1CF is a fairly general memory-move routine. It moves the block of memory whose start is given by the contents of $00AA-$00AB and whose end is given by the contents of $00A6-$00A7, to the area starting at the address given in $00A4-$00A5 and downward. It moves the highest memory first, progressing downward through the block until the move is complete. Given the previous settings for these pointers, the net result is that the program lines after the point where the new line should be inserted, are all moved up in memory far enough to allow space for the new line. Again, the link pointers from line to line are incorrect after this operation, but this will be addressed later.

$A306-$A30D move the end-of-program pointer, which was stored at $007F-$0080 by the block move routine above, back to its normal location at $007B-$007C. This is the corrected pointer, allowing for the bytes added by the new line. And then, at last, the code from $A30E-$A318 actually moves the new line into its proper place in the program. However, as noted above, the pointers which link one line to the next are incorrect at this point.

The code from $A319 thru $A34A takes care of this problem by rebuilding the complete set of pointers linking the lines. In the first two bytes of each line is the address of the next line, low byte first. These pointers are used to scan rapidly for a desired line, without needing to look at every character of each line to find its end, after determining that it is not the correct line. This is the source of the infinite loop if you attempt to list a program in which these pointers have been changed: a pointer may point back to a previous line, causing the LIST routine to keep repeating a set of lines over and over until you break or reset the system.

The JSR $A477 at $A319 takes care of some preliminary housekeeping. It jumps to a portion of the NEW routine, which does everything but actually destroy the program itself. The portion called clears the variable space, resetting the variable and array pointers to the end of the program so that new variables and arrays may be created; sets the currect-byte pointer in GETBYTE to the beginning of the program; RESTOREs the data pointer to the beginning of the program; and clears the hardware stack and the simulated stacks used for arithmetic and string processing.

The final portion of the routine, from $A31C thru $A34A, corrects the pointers. The general logic runs something like this: Starting at the beginning of the program (pointed to by $0079-$007A), skip over the first four bytes of each line (the link pointer and the line number). Then scan the remainder of the line for the end-of-line mark ($00). Assuming that the next line starts at the byte after that mark, set the current line's link pointer to the byte after the end of the line. Then use that location as the start for the next line. Detect the end of the program by finding the high byte of the link field (the second byte after the end of the previous line) a $00, which is impossible. Upon finding the end of the program, return to the immediate mode by jumping to $A27D, which skips over printing the OK prompt (thus you don't get the OK between each line as you type in a program). $0071-$0072 is used to store the pointer keeping track of the current position in the program during this process. After completing all this, the new line has been inserted correctly and the program is ready for execution or further editing.

Next month: the main executive routine and verb dispatcher.

★

C1P CORNER

By: David A. Jones
8902 SW. 17th Terrace
Miami, FL 33165

Now that I have had my disk up and running for a couple of months I couldn't resist the temptation to experiment once again with the hardware. Lo and behold, I find that not all of the code from $F800 to $FD00 is required by the C1P disk system. Actually it is only $FC00 to $FD00 that the C1P needs, the rest being monitors for other configurations. I programmed a new EPROM monitor using this extra space and haven't had any adverse effects in over a month now. Frank Aguilar (Mar. '82) take note.

I guess this is probably common knowledge to some but no one bothered to write about it. At least not in the publications that I subscribe to. I should have figured this out earlier myself with all of the replacement PROM monitors that are available for the C1P though. I hope some of you older 'Peekers' don't cringe at the following, as much of this article seems so elementary and I'm probably reinventing the wheel but all of it was developed independently by me. Anyway, I have never seen any of these ideas in print before, so here goes.

CREATE

In the create utility of version 65D 3.1 adding line;

```
530 FOR I = 0 TO 39: IF AL%(I)
    = 0 THEN PRINT I;
535 NEXT I: PRINT
```

will list all of the unused tracks on a disk prior to asking you for the starting track of the file you are going to create. Great for those who can't remember for more than a few seconds or so, things like, which tracks are open.

BEXEC*

BEXEC* doesn't do much except POKE a few constants into memory to enable some functions that we may or may not want all of the time, set up the input and output vectors and ask us what next. The new user soon gets tired of answering the prompt to unlock the system and so starts bypassing most of the code with a GOTO 10000 in line 50.

DIR

Later it may dawn on him that the DIR routine could fit into BEXEC* if all of the now unused code were removed. The system would boot up with the directory and an extra track

Super Sale!

NOW!! $149.95!!

60% ~~40%~~ Off On Ohio Scientific Superboard II
A Complete Computer System On A Board

Includes full-size 53-key keyboard, video and audio cassette interfaces; SWAP, Modem, sampler cassettes; manual; 8K BASIC-in-ROM, with 8K RAM. Requires 5-V/3 amp regulated DC power supply. 30-day limited warranty. Supply is limited. ONLY ~~$200.00~~ NOW! $149.95!!

Plus Sensational Limited-Time Savings On Ohio Scientific C1P Series personal computers, Superboard and C1P accessories, spare replacement parts, printers, monitors, integrated circuits, and other computer-related components.

To Order
Call us directly or return order coupon with your check, money order, or Mastercard or Visa Account Number. Orders will normally be shipped within 48 hours after receipt. $100.00 minimum order.

FREE
Sampler Cassettes with each Superboard II and C1P series order!

Taxi (Game), Electronic Equations, Loan Finance, Straight and Constant Depreciation, Uneven Cash Flows

Tiger Tank, Flip Flop, (Logic Game), Hectic, Black Jack, Master Mind

Cleveland Consumer Computers & Components
1333 S. Chillicothe Road, Aurora, OH 44202
TO ORDER: CALL 1-800-321-5805 TOLL FREE
(Ohio Residents Call 216-562-4136)

☐ SUPERBOARD II, $149.95 ~~$200.00~~

☐ Send Detailed Catalog/Order Form

Name ______________________

Address ______________________

City ____________ State ________ Zip ________

Payment by enclosed check or money order or charge to:

☐ MasterCard ☐ VISA

Account # ____________ Expiration Date ____________

Total Amount Charged or Enclosed $ ________

Ohio Residents Add 5.5% Sales Tax. All Orders Will Be Shipped Insured By UPS Unless Requested Otherwise.

OSI Disk Users

**Double your disk storage capacity
Without adding disk drives**

Now you can more than double your usable floppy disk storage capacity—for a fraction of the cost of additional disk drives. Modular Systems' DiskDoubler™ is a double-density adapter that doubles the storage capacity of each disk track. The DiskDoubler plugs directly into an OSI disk interface board. No changes to hardware or software are required.

The DiskDoubler increases total disk space under OS-65U to 550K; under OS-65D to 473K for 8-inch floppies, to 163K for mini-floppies. With the DiskDoubler, each drive does the work of two. You can have more and larger programs, related files, and disk utilities on the same disk—for easier operation without constant disk changes.

Your OSI system is an investment in computing power. Get the full value from the disk hardware and software that you already own. Just write to us, and we'll send you the full story on the DiskDoubler, along with the rest of our growing family of products for OSI disk systems.

Modular Systems

Post Office Box 16 D
Oradell, NJ 07649.0016
Telephone 201 262.0093

™DiskDoubler is a trademark of Modular Systems.

TACKLE YOUR PROGRAMMING PROBLEMS HEAD ON!

If operating bugs and "unsolvable" coding problems have got you dodging a big project, here's help . . . Quality software products that'll let you tackle those problems and move on to something easier.

At Software Consultants, we know the kinds of problems you often face with OSI gear, so we designed our software accordingly. Most of our products are modifications and/or extensions of OSI packages, but some are complete replacements. All are easily extensible for your own custom applications. We feel they're the finest OSI software products around—and the most reasonably priced.

PRODUCT ROUNDUP

OS-65D V3.2 DISASSEMBLY MANUAL

A super-complete manual that has it all. 50 pages of disassembly listings, complete and clear comments on all lines, 10 pages of computer generated cross reference listings, and more! Praised by many OSIers who couldn't believe it til they bought one. A deal at $25.95.

REF COMMAND UNDER BASIC

A complete, cross reference utility that'll find and list any BASIC line number, variable, or numeric constant. It's available under 65D or 65U and comes on 8" floppies. This one will save your sanity, and cut out hours of wasted time. Yours for $31.95.

SPOOLER/DESPOOLER UTILITY

A useful utility that feeds backed-up data to your printer for normal output, and leaves your screen free for other work by intercepting data bound for the printer and temporarily storing it on hard disk. Written in super fast machine language. Interfaces with serial and parallel printers. Only $69.95.

FIG FORTH UNDER OS-65U

The only other language available for your 65U system. You get lots of extras, like terminal oriented editor, printer and terminal tools, and much more. Under multi-user , runs BASIC simultaneously too. Unbelievable at $89.95.

VIDEO ROUTINE

This convenient extension for OS65D V1.2 really spices up your video system with little niceties like 24 separate control codes, horizontal and vertical plotting, and many variable screen parameters. Software extensions are available to connect this with the graphics resolution booster. The routine alone is $25.95, with extensions, $29.95.

GRAPHICS RESOLUTION BOOSTER

An ingenious piece of hardware that increases the graphic resolution per character by 8 times. With this device, your circles will be rounder and your curves smoother. A slick addition to your C4/8P system. Priced at $49.95, or, with the video routine and extensions, $79.95.

Unlike the majority of other software vendors, we offer our customers copies of source code (on floppies) for any of our products they've purchased. For a nominal fee of $12, covering postage & handling, we'll send you the source code you choose.

Foreign orders please add $10 postage fee for air mail. Dealer inquiries welcome. Tennessee residents add 6% sales tax. COD orders will incur a $1.50 handling charge. We ship via UPS, so please include street address with your order.

Write or call us today with your order, or ask for our free product catalog and get all the details.

SOFTWARE CONSULTANTS
HARDWORKING *software*

6435 Summer Avenue
Memphis,TN 38134
901/377-3503

would be free on a system disk.

RENAME

After a while one begins to wonder why he didn't compress the code a little more and include RENAME in BEXEC*. Easy to do and another track becomes available.

DELETE

We start getting smarter faster now and examine what happens when a file is deleted. Not much really. The entry in the directory is written over with #'s. Just one # is sufficient to keep 'DIR' and 'CREATE' from reading the rest of the name so renaming the file 'TEST' to '#EST' does the job but leaves enough to identify the file should you change your mind later on and try to locate it. Providing, of course, you didn't create another file in the same directory space. Adding a line to RENAME to ask you whether you wish to delete or rename followed by 'IF R$=D THEN AN$="#"+RIGHT$(A$,5) does this and another track is freed.

HEX

We now have 4 of the most useful utilities on one track and if we were conservative in the use of REM statements we still have room for a compact number converter ala Harry Hawkins and Dale Mayers. Change the prompt line mentioned above to include this and also 'CREATE' like INPUT 'LIST/PRINT/RENAME/DELETE/CREATE/NUMBER';R$ followed by 'IF R$=L THEN xxx, etc., renumber the whole thing and praise ourselves for being so efficient every time a disk is booted.

BREAK

I've seen several articles (Mar.'82 page 6 for example) on ways to save a program if the 'BREAK' key is accidentally depressed before a program being developed was saved, but none that recommends what at first glance seems obvious. Assume the file you are working on is called 'TEST' and is on tracks 30-34. If the BREAK key is hit and you need to save the file, enter the monitor and type 2547G. This puts you back in the kernal and you will have the A* prompt. Now just type PUT TEST or PUT 30. This seems so simple that maybe I'm missing something but it does work on C1P's.

```
100 REM BEXEC*/UTILITY 9-03-82
110 POKE 8993,02:POKE 8994,02
120 POKE 741,76:POKE 750,78:POKE 2073,173
130 POKE 2893,55:POKE 2894,8
140 DEF FNA(X)=10*INT(X/16)+X-16*INT(X/16)
150 DV=2
160 :
170 PRINT #DV : PRINT #DV,"OS-65D VERSION 3.1.2"
180 PRINT #DV,"--DIRECTORY--":PRINT #DV
190 PRINT #DV,"FILE NAME    TRACK RANGE"
200 PRINT #DV,"----------------------"
210 PN=11897
220 DISK!"CA 2E79=12,1
230 GOSUB 290
240 DISK!"CA 2E79=12,2
250 GOSUB 290
260 PRINT #DV
270 DV=2: GOTO 370
280 :
290 FOR I=PN TO PN+248 STEP 8
300 IF PEEK(I)=35 THEN 350
310 N$=""
320 FOR J=I TO 1+5: N$=N$+CHR$(PEEK(J)): NEXT J
330 PRINT #DV,N$;TAB(12);FNA(PEEK(1+6));TAB(16),"-";
340 PRINT #DV,TAB(17);FNA(PEEK(1+7))
350 NEXT I:RETURN
360 :
370 PRINT:INPUT"LIST/PRINT/RENAME/DELETE/CREATE/NUMBER";R$
380 IF R$="L" THEN 170
390 IF R$="P" THEN DV=4: GOTO 170
400 IF R$="C" THEN RUN"CREATE
410 IF R$="N" THEN 780
420 IF R$<>"R" AND R$<>"D" THEN 370
430 :
440 FLAG=0
450 PRINT:INPUT "FILE NAME";A$
460 IF LEN(A$)>6 THEN 450
470 IF LEN(A$)<6 THEN A$=A$+" ": GOTO 470
480 DISK!"CA 2E79=12,1
490 GOSUB 540 :IF FL=1 THEN DISK!"SA 12,1=2E79/1":GOTO 370
500 DISK!"CA 2E79=12,2
510 GOSUB 540 :IF FL=1 THEN DISK!"SA 12,2=2E79/1":GOTO 370
520 GOTO 740
530 :
540 REM CHECK DIRECTORY
550 FOR I=PNT TO PN+248 STEP 8
560 FOR J=I TO I+5
570 IF PEEK(J)<>ASC(MID$(A$,J-I+1,1)) THEN 710
580 NEXT J
590 :
600 IF R$="R" THEN 640
610 PRINT:PRINT"DELETE";CHR$(34);A$;CHR$(34);:INPUT "OK";AN$
620 IF AN$="Y" THEN AN$="#"+RIGHT$(A$,5): GOTO 690
630 :
640 PRINT:PRINT"RENAME";CHR$(34);A$;CHR$(34);:INPUT "TO";AN$
650 IF LEN(AN$)>6 THEN 640
660 IF LEN(AN$)<6 THEN AN$=AN$+" " : GOTO 660
670 IF MID$(AN$,1,1)<"A" OR MID$(AN$,1,1)>"Z" THEN 640
680 :
690 FOR J=I TO I+5 : POKE J,ASC(MID$(AN$,J-I+1,1)) : NEXT J
700 FLAG=1 :RETURN
710 NEXT I
720 FLAG=0 :RETURN
730 :
740 PRINT
750 PRINT"* ";CHR$(34);A$;CHR$(34);" NOT FOUND IN DIRECTORY *
760 FOR X=1 TO 1500:NEXT:RUN
770 :
780 PRINT:INPUT "H/D";H$:IF H$="H" THEN 830
790 H$="":PRINT:INPUT"DEC";D
800 A=INT(D/16):B=D-A*16:H$=CHR$(B-7*(B>9)+48)+H$:D=A:IF
    D<>0THEN 800
810 PRINT:PRINT"HEX  "H$:GOTO 370
820 :
830 PRINT:INPUT"HEX";HEX$
840 N=0:H=1:FOR X=1 TO LEN(HE$):D=ASC(RIGHT$(HE$,X))-48
850 D=D+7*(D>9):N=N+D*H:H=H*16:NEXT
860 PRINT:PRINT"DEC "N:GOTO 370
OK
```

A BAD EXPERIENCE

Observing a few soft errors reading disks, I started shopping around for a manual for the MPI 51 drive. With no success locally I let my fingers do the walking and found that Jade Computers would send me one for $20.00 plus shipping. A little steep I thought, but when you have problems, any port in a storm! Much to my surprise, I received a photocopied manual, and a poor copy at that, without a schematic. Feeling ripped off, I promptly called Jade and got my second surprise. Of course, it is a photocopy, what did you expect? Well, I did expect an original copy and I did expect a schematic. No amount of persuasion on my part could convince them that they had done wrong but they did agree to refund the purchase price because I was dissatisfied but I would be out the shipping costs both ways. Makes one a little leary of ordering by mail or phone, even when you think you know exactly what you are ordering. As the Apple ads say you may get a 'Samurai Ichiban'.

A PUZZLE

Speaking of mail order, I'm still wondering what Modular Systems has to offer. After 2 requests for information plus intervention by Peek(65) I still don't know any more that what's promised in the ads. With only one disk drive I really am looking for expansion options. Has anyone else had better luck hearing from them or actually tried the doubler?

SAVE MACHINE LANGUAGE PROGRAMS ON OS65D

by: Richard List
2104 Village Drive
Pittsburgh, PA 15221

(C1P-4PMF)

In the May 1982 issue of PEEK(65), Gary Wolf had an interesting program to re-locate a tape based machine language program so it could be saved and run on disk. Without being relocated (temporarily), it would have been wiped out by the disk system.

Gary's program, as written, was for a machine language program at a particular location. For someone who understands machine code it was easy to modify for any other machine language program. I had many machine language programs on tape, written in various locations, that could not be easily transferred to disk. Since Mr. Wolf's program was for a particular case, I re-wrote the program, called RELOCATOR, to make it general, so any taped machine language program at any location could be transferred to disk. Also, no knowledge of machine language is necessary and the relocator talks you through.

The easiest way to show how it operates is to step through a f'rinstance. Suppose you had a cassette machine language program of length 4K, named JUNK, that you want to transfer to disk. It starts loading at $0200 and execution starts at $0300. You want to save it on tracks 24 and 25. Here is the procedure:

1. Create a 1 track file named "JUNK". This will actually be a command file when we're finished.

2. Load and RUN RELOCATOR, selecting option #1 (load relocator). Program asks for a beginning location (hex). Answer, in this case, 0200.

3. BREAK M and load JUNK from cassette in the normal way. If your program starts executing automatically, you may have to do another BREAK M.

4. Still in the monitor mode, type: .5000G. This executes the relocator which moves JUNK from $0200 to upper memory where it's safe and cozy.

5. BREAK D, load and run RELOCATOR again, this time selecting menu option #2 (save on disk).

6. RELOCATOR will ask for start of execution location (may be same as start of program). Answer 0300.

7. RELOCATOR will ask for the starting track that you want your machine language program saved on. Answer 24.

8. RELOCATOR will ask for number of tracks required. Answer in this case, 2. Your program is now stored on disk, but it still needs a short program to dump it back into RAM.

9. RELOCATOR will now list a short program of several lines, consisting of a series of CALL commands, and a GO 5000 command. Type this program in exactly as listed on the screen and save it under the file JUNK (DISK!"PU JUNK"), which you previously created.

From now on, to run the machine language program JUNK, simply RUN JUNK. That program will load the tracks holding the machine language program to high memory, along with the relocator, transfer control to the machine language re-locator, which will relocate it to low memory (disk system no longer needed so it's safe to use low memory now), and run it.

When broken down into itty-bitty steps, it looks messy, gut it's really easy to use.

The first run of RELOCATOR loads a machine language program which can move a block from any location, to high memory. Then the program we want to save is loaded using the monitor. The relocator is run from the monitor, which moves your program to high memory where it will be safe from the clutches of the disk system. On the second run of the BASIC relocator program, the machine language relocator program is modified so it can move your program from high memory back to where it belongs.

The combination of the relocator and your machine language program is stored on disk from high memory. A command file is created which will load the relocator and your machine language program back into high RAM, then automatically transfer control to the re- locator. The relocator will move your program to its proper location and start execution. The assembly listing for the two options are as shown in Mr. Wolf's article, except correct addresses are generated by the BASIC program. As suggested by Mr. Wolf, if the command file (JUNK in this case) is made the BEXEC*, the program will automatically execute when the disk is booted.

Another version would be to have loading commands for several programs in the command file, the proper program selected from a menu, using IF statements or ON--GOTO statements, to run the proper set of loading

```
1    REM RICHARD W. LIST
2    REM 2104 VILLAGE DR
3    REM PITTSBURGH PA 15221
4    REM (412)371-6275
10   DIMWX$(16)
20   FOR I=0TO15:READWX(I):NEXT
30   DATA 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
40   INPUT"(1) LOAD RELOCATOR, (2) SAVE ON DISK";OP
45   REM LOAD RELOCATOR
50   FOR I=20480TO20504:READX:IFOP=IOROP<20484THENPOKEI,X
60   NEXT
70   ONOPGOTO80,160
80   INPUT"BEGINNING LOCATION";BL$:GOSUB540:GOSUB450
85   REM MODIFY RELOC PROG ADDRESSES TO MOVE PROG AT SELECTED LOC
90   POKE20485,LB:POKE20486,HB:POKE20505,LB:POKE20506,HB
100  IFOP=ITHENPRINT"LOAD ML PROG TO BE SAVED & TYPE .5000G."
110  PRINT"THIS WILL BE MOVED TO A TEMP STORAGE AREA.
120  PRINT"THEN, BREAK D, AND RERUN THIS PROGRAM USING MENU ITEM 2.
130  STOP
140  DATA162,0,160,42,189,0,3,157,0,83,232,208,247
150  DATA238,6,80,238,9,80,136,208,238,76,0,254
160  INPUT"START OF EXECUTION LOCATION";BL$:GOSUB540:GOSUB450
170  L=PEEK(20505):H=PEEK(20506)
175  REM CHANGE ADDR PARTS OF RELOC TO MOVE PROG BACK
180  POKE20488,L:POKE20489,H:POKE20485,O:POKE20486,83
190  POKE20503,LB:POKE20504,HB
200  INPUT"READY TO PUT ON DISK";A$
210  IFLEFT$(A$,1)<>"Y"THENPRINT"DUMMY!":STOP
220  INPUT"STARTING TRACK";ST
230  INPUT"NUMBER TRACKS";NT
240  READD1$:FORI=1TO6:READD2$(I):NEXT
250  DATA"SA ",",1=5000/8"
260  DATA",1=5800/8",",1=6000/8",",1=6800/8",",1=7000/8",",1=7800/8
265  REM CREATE DISK COMMAND STRING
270  FORI=ST TO ST+NT-1
280  D$=D$+STR$(I)+D2$(I-ST+1)
290  PRINT"DISK COMMAND: ";D$
295  REM EXECUTE DISK COMMANDS:
300  DISK!D$
310  NEXT
320  PRINT"ML PROGRAM SAVED.":PRINT
330  PRINT"CREATE A 1 TRACK FILE W/ YOUR PROGRAM NAME."
340  PRINT"SAVE THE FOLLOWING PROGRAM IN THAT FILE:"
350  LN=10
360  FOR I=IT06:READD2$(I):NEXT
370  DATA5000=,5800=,6000=,6800=,7000=,7800=
375  REM PROG TO BE SAVED ON COMMAND FILE TRACK:
380  PRINT:PRINT"------------------------"
390  FORI=STTOST+NT-1
400  PRINTLN;" DISK!";CHR$(34);"CA ";D2$(I-ST+1);I;",1"
410  LN=LN+10:NEXT
420  PRINTLN," DISK!",CHR$(34);"GO5000"
430  PRINT:PRINT"------------------------":STOP
440  REM HEX-DEC CONVERTER
450  FORI=0TO15
460  IFRIGHT$(LB$(1)=HX$(I)THENL1=I
470  IFLEFT$(LB$,1)=HX$(I)THENL2=I
480  IFRIGHT$(HB$,1)=HX$(I)THENH1=I
490  IFLEFT$(HB$,1)=HX$(I)THENH2=I
500  NEXT
510  LB=16*L2+L1:HB=16*H2+H1
520  RETURN
530  REM LO-HI BYTE SEPARATOR
540  HB$=LEFT$(BL$,2):LB$=RIGHT$(BL$,2):RETURN
```

commands. This would display a menu, just like the normal BEXEC*. For someone with a collection of machine language tapes, Mr. Wolf's concept saves a lot of rewriting. With these modifications, all programs can be stored on disk very easily without having to customize the relocator.

A GOTCHA!!

There is a minor goof in the original article. The call and save commands do not agree. The CALL shows tracks 23 to 28 whereas the SA shows tracks 21 to 26. Either may be correct, but the CA and the SA must be the same.

★ ★ ★

OSI-FORTH

OSI-FORTH 3.0 is a full implementation of the FORTH Interest Group FORTH, for disk-based OSI systems (C1, C2, C3, C4, C8) Running under OS65D3, it includes a resident text editor and 6502 assembler. Over 150 pages of documentation and a handy reference card are provided. Requires 24K (20K C1P). Eight-inch or mini disk $79.95. Manual only, $9.95. "OSI-FORTH Letters" software support newsletter $4.00/year.

Other Software for Ohio Scientific Computers:

VIDEO EDITOR

Video Editor is a powerful full screen editor for disk-based C2, C4, C8 systems with the polled keyboard and color video boards (b&w monitor ok). Allows full cursor-control with insertion, deletion and duplication of source for BASIC or OSI's Assembler/Editor. Unlike versions written in BASIC, this machine-code editor is co-resident with BASIC (or the Assembler), autoloading into the highest three pages of RAM upon boot. Video Editor also provides single-keystroke control of sound, screen format, color and background color. Eight-inch or mini disk: $14.95. Specify amount of RAM.

SOFT FRONT PANEL

Soft Front Panel is a software single-stepper, slow-stepper and debugger-emulator that permits easy development of 6502 machine code. SFP is a fantastic monitor, simultaneously displaying all registers, flags, the stack and more. Address traps, opcode traps, traps on memory content and on port and stack activity are all supported. This is for disk systems with polled keyboard and color (b&w monitor ok). Uses sound and color capabilities of OSI C2/C4/C8 systems (not for C1P). Eight-inch or mini disk $24.95. Specify amount of RAM. Manual only, $4.95 (May be later credited toward software purchase). Six page brochure available free upon request.

TERMINAL CONTROL PROGRAM

OSI-TCP is a sophisticated Terminal Control Program for editing OS-65D3 files, and for uploading and downloading these files to other computers through the CPU board's serial port on OSI C2, C4, and C8 disk-based systems with polled keyboards. Thirteen editor commands allow full editing of files, including commands for sending any text out the terminal port and saving whatever text comes back. INDUTL utility included for converting between BASIC source and TCP file text. Eight-inch or mini disk $39.95. Manual only, $2.95.

WRITE FOR FREE CATALOG!

Prices shown are postpaid

Specify computer model & RAM.

NEW ADDRESS

Technical Products Company

P.O. BOX 9053

Boone, NC 28608

USE YOUR COMPUTER FOR TV CONVERGENCE ADJUSTMENTS

(ALL VIDEO MACHINES)

by: Robert S. Baldassano
4045 Ashbrook Circle
San Jose, CA. 95124

The figures on my TV screen had multicolor fringes attached to them and I knew it was time to adjust the color convergence. I had the forethought to order a repair manual from the manufacturer, but to do the job, I needed a pattern generator and I didn't own one. A simple hand held one would cost $72.95 in a kit and I would still have to put it together - I already had a backlog of projects on my workbench.

But wait a minute - I had a versatile pattern generator already - my OSI C8PDF computer!

To do the static and dynamic color convergence I only needed a dot pattern and a crosshatch pattern, but other patterns could be generated as well.

So I hooked my TV up to my computer, after first plugging the TV into an isolation transformer for safety. I then typed the following simple program into my computer:

```
10 POKE 56832,4 :REM SELECT 32
   X 32 DISPLAY, SOUND OFF
20 INPUT Y :REM SELECT CHAR-
   ACTER CODE
30 FOR X=53248 TO 55263
40 POKE X,Y
50 NEXT X
60 GOTO 60
```

I then ran the program inputting a 165 to the ? prompt to get a dot pattern so I could check and adjust the static convergence (superimposing the red, blue and green dots in the center of the screen). I then entered a CTRL C to stop the program and I ran it again this time entering a 219 at the prompt to get a cross hatch pattern. I then used this pattern to do a dynamic convergence (getting the red, blue and green dots superimposed at the edges of the screen). I also used this pattern to check for a tilted or off center image.

The number of patterns is almost unlimited, but the following inputs for Y will give the standard patterns indicated:

Y	PATTERN
165	DOTS
219	CROSS HATCH
144	HORIZONTAL LINES
149	VERTICAL LINES

Of course you could use the same procedure to adjust a color monitor as well.

★

LETTERS

ED:

In the past few years, I have seen some computers that have displayed some neat graphics. I thought, why can't my OSI do that? After hours of experimenting, I came up with this program that comes close to a 'plot routine'. The first part of the program, lines 170 to 210, is the screen and color clear routine. The next part of the program, lines 250 to 630, are the functions I plotted. The last part, line 670, is the 'plot' routine. It works like this. First you find the middle of the screen. Then you calculate the X value and the Y value. Next you multiply the Y value times 64 to get the Y direction.

Finally you add the memory location to the X value and the new Y value then poke all that to the color memory. Those who don't have color, can poke to the graphic memory using a graphic character. This gives you the same effect. You can change the functions around to give you different patterns. With a few changes this can be run on any OSI, mine is a C4P. I hope you enjoy my program as much as I have. Happy computing!!!

```
100 REM JOHN FRANKFORTHER
110 REM
120 REM PLOTTING FUNCTIONS
130 REM
140 REM
150 REM *** COLOR AND SCREEN
    CLEAR ***
160 REM
170 FORX=0TO36:READA:POKE16384
    +X,A:NEXT
180 POKE11,0:POKE12,64:X=
    USR(X)
190 DATA169,208,141,18,64,169,
    224,141,23,64,162,0,160,8,
    169,32
200 DATA157,0,216,169,14,157,
    0,232,232,208,243,238,18,
    64,238,23
210 DATA64,136,208,234,96
220 REM
230 REM PLOT FLOWER
240 REM
250 PRINTTAB(10)"THIS IS THE
    FLOWER PLOT":
260 FORX=0TO10:PRINT:NEXT
260 FORX=0TO5000:NEXT
270 S=USR(S):M=58336:D=0:A=15:
    POKE56832,5
280 FORT=1TO7STEP.2
290 FORI=0TO20STEP.1
300 R=A*COS(T*I)
310 X=INT(R*SIN(I))
320 Y=INT(R*COS(I))
330 GOSUB670:NEXTI:D=D+1:X=USR
    (X):IFD>12THEND=0
340 NEXTT
350 REM
360 REM PLOT CIRCLE
370 REM
380 X=USR(X)
390 PRINTTAB(10)"THIS IS A
    CIRCLE PLOT":FORX=0TO10
    :PRINT:NEXT
400 FORX=0TO5000:NEXT
410 S=USR(S):M=58016:D=0
420 FORA=0TO20
430 FORI=0TO30
440 R=A*COS(I)
450 X=INT(R*SIN(I))
460 Y=INT(R*COS(I))
470 GOSUB670:NEXTI:S=USR(S):
    D=D+1:IFD>12THEND=0
480 NEXTA:FORX=0TO5000:NEXT:
    X=USR(X)
490 REM
500 REM PLOT SPIRAL
510 REM
520 X=USR(X)
530 PRINTTAB(10)"THIS IS A
    SPIRAL PLOT":FORX=0TO10
    :PRINT:NEXT
540 FORX=0TO5000:NEXT
550 X=USR(X):M=58336:D=0:A=15
    :POKE56832,5
560 FORB=1TO2
570 FORT=0TO10STEP,5
580 Y=INT(K*SIN(T))
590 X=INT(K*COS(T))
600 K=K+.03
610 GOSUB670:NEXTT,B
620 D=D+1:IFD>12THENFORX=0TO
    5000:NEXT:X=USR(X):END
630 GOTO560
640 REM
640 REM
650 REM
660 REM  ***  PLOTTING ROUTINE
    *****
670 Y=Y*64:Z=M+X+Y:POKEZ,D:
    RETURN
```

John Frankforther
Maumee, OH 43537

* * * * *

ED:

For your readers that want to try their hand at programming, this program can come in handy. It has a lot of useful things in it such as the BASIC screen clear, the BASIC print at; real back space using print at and also checks and sets up both the screen clear and the screen width for the 600 and the 540 board.

I use a lot of strings and

string manipulations in the program. Anyone having only 8K of RAM might have a problem with the garbage collector unless you install the fix for the garbage collector.

If you do not have it, you can find one in the January '80 issue of PEEK (65), page 4 and in Vol. #1 #2 June 1980 of the Aardvark Journal, page 12.

I have installed REM's throughout the program and you can remove them when you load it to save memory.

The program was made as a tutorial and is called resistor color code.

Lines 10010 and 15105=a check for type of computer C1/2/4.

Lines 15100 through 15120 =screen clear.

Lines 10202 through 10220 =print at.

Line 10215 =check for carriage return.

Line 10221 =back space.

Lines 10240 through 10265 =add up print ats to make color.

Lines 10010 through 10028 =set up screen and graphic variables.

Lines 10060 through 10080 =set up resistor value at random.

Robert F. Corwin
Portville, NY 14770

* * * * *

RESISTOR COLOR CODE # 1 @

```
8900 REM ROBERT CORWIN
8910 REM RD#1 PROTVILLE
8920 REM NEW YORK 14770
8940 REM      1980
8980 REM PROGRAM FOR C1/2/4
8990 POKE11,186:POKE12,255
8992 REM SET USR(X) FOR
8993 REM HIT ANY KEY TO CONTINUE
9000 GOSUB15100:PRINTTAB(3)"COLOR CODE #1"
9020 FORX=1TO11:PRINT:NEXT:FORX=1TO2000:NEXT
9040 PRINT"COLOR VALUE  MULTIPLER":PRINT
9050 PRINT"BLACK  0  X1"
9060 PRINT:PRINT"BROWN  1  X10"
9070 PRINT:PRINT"RED    2  X100"
9080 PRINT:PRINT"ORANGE 3 X1000
9090 PRINT:PRINT"YELLOW 4 X10000"
9100 PRINT:PRINT"GREEN  5 X100000"
9110 PRINT:PRINT"BLUE   6 X1000000"
9120 PRINT:PRINT"VIOLET 7 X10000000"
9130 PRINT:PRINT"GREY   8 X100000000"
9140 PRINT:PRINT"WHITE  9 X1000000000"
9150 PRINT:PRINT"HIT ANY KEY TO CONTINUE":X=USR(X)
9160 GOSUB15100:PRINT"REMEMBER"
9180 PRINT:PRINT"IN THIS PROGRAM"
9190 PRINT:PRINT"WE WILL ASSUME THAT"
9200 PRINT:PRINT"ALL RESISTOR'S"
9210 PRINT:PRINT"WILL HAVE A TOLERANCE"
9220 PRINT:PRINT"OF 20% WITH NO"
9230 PRINT:PRINT"FORTH BAND,BUT"
9240 PRINT"PRINT"REMEMBER THE TOLERANCE"
9250 PRINT:PRINT"CODE";:PRINTCHR$(20)
9260 PRINT:PRINT"HIT ANY KEY TO CONTINUE":X=USR(X)
9270 GOSUB15100:PRINT"COLOR  TOLERANCE"
9290 PRINT:PRINT"NO COLOR =20%":PRINT:PRINT"SILVER     =10%"
9310 PRINT:PRINT"GOLD      = 5%":PRINT:PRINT"RED        = 2%"
9330 FORX=1TO6:PRINT:NEXT
9340 PRINT:PRINT"HIT ANY KEY TO CONTINUE":X=USR(X)
9345 GOSUB15100:PRINTTAB(9)"NOTE!!"
9352 PRINT:PRINT:PRINT"IN RUNNING THIS PROGRAM"
9354 PRINT:PRINT"YOUR ENTRIES WILL BE"
9356 PRINT:PRINT"ENTERED MID SCREEN"
9360 PRINT:PRINT"HIT RETURN AFTER EACH"
9364 PRINT:PRINT"COLOR IS ENTERED"
9370 PRINT:PRINT:PRINT:PRINT"HIT ANY KEY TO CONTINUE":X=USR(X)
9400 GOSUB15100:PRINTTAB(6)"NOTE!!!"
9410 PRINT:PRINT:PRINT"IF YOU MISSPELL A"
9420 PRINT:PRINT"WORD YOU HIT THE":PRINT:PRINT"LEFT SHIFT KEY
     AND"
9430 PRINT:PRINT"THE"CHR$(34);"O";CHR(34);"AT THE SAME TIME"
```

LISTING CONTINUED ON PAGE 17

MnM Software Technologies, Inc.
9701 Fields Rd., Suite 1904
Gaithersburg, Maryland 20878

INTRODUCING OUR NEW PRODUCT LINE

The missing tools for the OS-65U system. Our products are written in 6502 native code and are compatible with 65U, single, time-share or network modes. Floppy or hard disk systems.

Ky. ASM V1.1-ASSEMBLER (Virtual source files, superfast, many extra features including a label table) ...$129 (manual $25)(50 pgs.)

Ky. COM V1.5-COMPILER (Configures itself to V1.2 or 1.42, dynamic variables and arrays DIM A (N), supports machine language routines at hex6000, last 2 pages in high memory accessible, debug with interpreter and compile in 2-3 minutes. Protect your valuable source routines, gain as much as 2-10 times on average programs in execution speed. Supports 'INPUT[' and 'PRINT[' on the 1.42 system. ....$395 (manual $25)(110 pgs.)

Ky. DEV I-ASSEMBLER AND COMPILER TOGETHER....$474(manual $40)

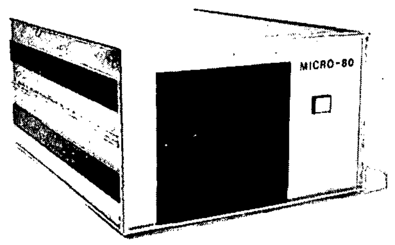
KEYMASTER I V1.0-The word processing missing link for OS-65U based systems. KEYMASTER I is screen oriented, menu driven, simple to use yet highly advanced. KEYMASTER I contains most of the best features only found in dedicated work processing systems. Ask for the features you have been looking for and the answer will most likely be "YES!" We are working on a good manual for your use. Check with us sometime in October....Introductory price $475

All software comes with license agreement, registration card, manual, binder, diskette holder and 8" diskette.

Manuals are available by themselves and are deductible from full purchase price of software within 60 days after purchase.

ALLOW 2 WEEKS FOR DELIVERY AFTER RECEIPT OF CHECK OR MONEY ORDER **CALL-301/963-2325**

NEW FROM D & N MICRO PRODUCTS, INC.



MICRO-80 COMPUTER

Z80A CPU with 4MHz clock and CP/M 2.2 operating system. 64K of low power static RAM. Calendar real time clock. Centronics type parallel printer interface. Serial interface for terminal communications, dip switch baud rates of 150 to 9600. 4″ cooling fan with air intake on back of computer and discharge through ventilation in the bottom. No holes on computer top or side for entry of foreign object. Two 8″ single or double sided floppy disk drives. IBM single density 3740 format for 243K of storage on each drive. Using double density with 1K sectors 608K of storage is available on a single sided drive or 1.2 meg on a double sided drive. Satin finish extruded aluminum with vinyl woodgrain decorative finish. 8 slot backplane for expansion. 48 pin buss is compatible with most OSI boards. Uses all standard IBM format CP/M software.

Model 80-1200 2 8″ single sided drives, 1.2 meg of storage	$2995
Model 80-2400 2 8″ double sided drives, 2.4 meg of storage	$3495
Option 001 Serial printer port, dip switch baud rate settings	$ 95

Software available in IBM single density 8″ format.

Microsoft		**Digital Research**		**Micropro**	
Basic-80	$289	PL/1-80	$459	Wordstar	$299
Basic Compiler	$329	Mac	$ 85	Mail-Merge	$109
Fortran-80	$410	Sid	$ 78	Spellstar	$175
Cobol-80	$574	Z-Sid	$ 95	Super Sort I	$195
Macro-80	$175	C Basic-2	$110	**Pascal**	
Edit-80	$105	Tex	$ 90	Pascal/MT +	$429
Mu Simp/Mu Math	$224	DeSpool	$ 50	Pascal Z	$349
Mu Lisp-80	$174	**Ashton-Tate**		Pascal M	$355
		dBase II	$595		

Convert almost any static memory OSI machine to CP/M® with the D & N-80 CPU Board.

Z80A CPU with 4MHz clock. 2716 EPROM with monitor and bootstrap loader. RS-232 serial interface for terminal communications or use as a serial printer interface in a VIDEO system. Disk controller is an Intel 8272 chip to provide single or double density disk format. 243K single density or 608K double density of disk storage on a single sided 8″ drive. A double sided drive provides 1.2 meg of storage. DMA used with disk controller to unload CPU during block transfers from the disk drives. Optional Centronics type parallel printer port complete with 10 ft. cable. Optional Real Time Calendar Clock may be set or read using 'CALL' function in high level languages. Power requirements are only 5 volts at 1.4 amps. Available with WORDSTAR for serial terminal systems.

Includes CPM 2.2

D & N-80	serial	$695
D & N-80	serial w/Wordstar	$870
D & N-80	video	$695
Option 001	parallel printer and real time calendar clock	$ 80

D & N-80 CPU BOARD

OTHER OSI COMPATIBLE HARDWARE

IO-CA10X Serial Printer Port $125
Compatible with OS-65U and OS-65D software

IO-CA9 Parallel Printer Port $175
Centronics standard parallel printer interface with 10 ft. flat cable

BP-580 8 Slot Backplane $ 47
Assembled 8 slot backplane for OSI 48 pin buss

24MEM-CM9	**$380**	**24MEM-CM9F**	**$530**
16MEM-CM9	**$300**	**16MEM-CM9F**	**$450**
8MEM-CM9	**$210**	**8MEM-CM9F**	**$360**
BMEM-CM9F	**$ 50**	**FL470**	**$180**

24K memory/floppy controller card supports up to 24K of 2114 memory chips and an OSI type floppy disk controller. Available fully assembled and tested with 8, 16, or 24K of memory, with floppy controller (F). Controller supports 2 drives. Needs separated clock and data inputs. Available Bare (BMEM-CM9F) or controller only (FL-470). Ideal way to upgrade cassette based system

C1P-EXP Expansion Interface $ 65
Expansion for C1P 600 or 610 board to the OSI 48 pin buss. Requires one slot in backplane. Use with BP-580 backplane

BIO-1600 Bare IO card $ 50
Supports 8K of memory, 2 16 bit parallel ports may be used as printer interfaces. 5 RS-232 serial ports, with manual and Molex connectors

DSK-SW Disk Switch $ 29
Extends life of drive and media. Shuts off minifloppy spindle motor when system is not accessing the drive. Complete KIT and manual

D & N Micro Products, Inc.
3684 N. Wells St.
Fort Wayne, Ind. 46808
(219) 485-6414

VISA MasterCard

TERMS $2.50 shipping, Foreign orders add 15%.
Indiana residents add 4% sales tax.

Disk Drives and Cables

8″ Shugart SA801 single sided	**$395**
8″ Shugart SA851 double sided	**$585**
FLC-6 6ft. cable from D & N or OSI controller to 8″ disk drive	**$ 69**
5 1/4″ MPI B51 with cable, power supply and cabinet	**$450**
FLC-5 1/4 8 ft. cable for connection to 5 1/4 drive and D & N or OSI controller, with data separator and disk switch	**$ 75**

Okidata Microline Printers

ML 82A Dot Matrix Printer **$534**
120 CPS, 80/120 columns, 9.5″ paper width, friction or pin feed

ML 83A Same as 82A except **$895**
16″ paper width, 132/232 columns with tractor feed

ML 84 Same as 82A except 200 CPS, **$1152**
16″ paper width, 132/232 columns, 2K buffer, dot addressable graphics, with tractor feed

CONTINUED FROM PAGE 15

```
9440 PRINT:PRINT"THIS WILL REMOVE THE":PRINT:PRINT"LAST LETTER
     YOU"
9450 PRINT:PRINT"TYPED AND ALLOW YOU":PRINT:PRINT"TO CORRECT
     IT"
9460 PRINT:PRINT:PRINT"HIT SPACE BAR":X=USR(X):GOSUB15100
9999 INPUT"TYPE IN YOUR NAME";W$
10005 T=1:V=0
10008 REM SET SCORE TO 0
10010 A=53380:L=32:IFPEEK(57088)<129THENA=53441:L=64
10015 RESTORE:FORX=2TO10:L(X)=L*X:NEXT:C=A+6+L(4)
10020 D=A+16+L(4):F=A+2+L*15:G=A+1+L*20:H=A+8+L(9):E=0:N=0
10025 REM C1/2/4-VARIABLES:A=UPPER CORNER:L=LINE LENGTH
10028 REM A+=RESISTOR:C=YOUR ANSWER:D=CORRECT ANSWER:G=QUESTION
10030 GOSUB15000:GOSUB15100
10032 REM GET DATA
10050 FORX=5TO1STEP-1:PRINTTAB(L/2-1)A$(X):NEXT
10052 FORX=10TO6STEP-1:PRINTTAB(L/2-1)A$(X):NEXT
10053 REM PRINT COLORS
10055 FORX=1TOL-23:PRINT:NEXT:GOSUB14500
10060 I=INT(RND(20)*10):IFI<2ORI>10THEN10060
10070 J=INT(RND(29)*10):IFJ<0ORJ>10THEN10070
10080 K=INT(RND(20)*10):IFK<1ORK>6THEN10080
10085 REM RESISTOR RANDOM VALUE
10090 GOSUB14400:I=I+48:J=J+48
10092 REM ADJUST FORM DEC. TO ASCII
10094 FORX=1TOLEN(B$(K)):J$(X)=MID$(B$(K),X,1):NEXTX
10100 POKEG+10,I:POKEG+11,J
10102 FORX=1TOLEN(B$(K)):POKEG+11+X,ASC(J$(X)):NEXT
10104 POKEC+2,63:POKEC+2+L(2),63:POKEC+2+L(4),63
10110 POKEC+4,22:GOSUB10200:I=I-48:J=J-48
10111 REM ADJUST FROM ASCII TO DEC.
10112 IFA$(I+1)=G$ANDA$(J+1)=H$ANDA$(K)=I$THENV=V+1:GOTO16000
10113 M=A+14+L(4):N=A+L(5)-9
10114 FORX=MTON:POKEX,32:IFX=A-9+L*14THEN10117
10115 REM CHECK FOR CORRECT ANSWER LINE 10112
10116 NEXT:M=M+L:N=N+L:GOTO10114
10117 GOSUB10300:I=I+1:J=J+1
10118 REM CLEAR PART OF SCREEN AND ADJUST FROMDEC. TO ASCII
10120 FORX=1TOLEN(A$(I)):F$(X)=MID$(A$(I),X,1):NEXT
10125 FORX=1TOLEN(A$(J)):G$(X)=MID$(A$(J),X,1):NEXT
10130 FORX=1TOLEN(A$(K)):J$(X)=MID$(A$(K),X,1):NEXT
10135 FORX=1TOLEN(A$(I)):POKED+X,ASC(F$(X)):NEXT
10140 FORX=1TOLEN(A$(J)):POKED+L(2)+X,ASC(G$(X)):NEXT
10145 FORX=1TOLEN(A$(K)):POKED+L(4)+X,ASC(J$(X)):NEXT
10150 GOTO16200
10200 GOSUB10270
10202 E=E+1:D$=" ":N=0
10205 FORX=4TO6:F$(X)="":NEXTX
10210 POKE11,0:POKE12,253:X=USR(X):P$=CHR$(PEEK(531))
10215 IFPEEK(531)=13THENC=C+L(2):POKEC+4,22:ONEGOTO10240,
      10250,10260
10218 REM PRINT AT & CHECT FOR RETURN KEY
10220 N=N+1:D$=D$+P$
10221 IFPEEK(57100)=250THENN=N-2:POKEC+N+2,32:GOTO10223
10222 GOTO10225
10223 D$=" ":FORY=1TON:D$=D$+F$(Y):NEXT:GOTO10210
10224 REM LINES 10221 AND 10223=BACK SPACE
10225 FORY=1TOLEN(D$):POKEC+Y,ASC(MID$(D$,Y,1))
10230 F$(N)=P$:NEXTY
10235 GOTO10210
10240 G$=F$(1)+F$(2)+F$(3)+F$(4)+F$(5)+F$(6)
10245 GOTO10202
10250 H$=F$(1)+F$(2)+F$(3)+F$(4)+F$(5)+F$(6)
10255 GOTO10202
10260 I$=F$(1)+F$(2)+F$(3)+F$(4)+F$(5)+F$(6)
10261 REM ADD UP PRINT AT'S TO MAKE COLOR
10262 IFPEEK(H+2+L)=22THENPOKEH+2+L,82
10265 RETURN
10270 POKEH,16:POKEH+L-1,89:POKEH+L,79:POKEH+1+L,85
      :POKEH+2+L,82
10275 POKEH+L(2)-2,65:POKEH+L(2)-1,78:POKEH+L(2),83
      :POKEH+1+L(2),87
10280 POKEH+2+L(2),69:POKEH+3+L(2),82
10282 REM GRAPHICS = YOUR ANSWER
10285 RETURN
10300 POKEF+1+L(2),87:POKEF+2+L(2),82:POKEF+3+L(2),79
10302 POKEF+4+L(2),78
10305 POKEF+5+L(2),71:POKEF+7+L(2),84:POKEF+8+L(2),82
10308 POKEF+9+L(2),89:POKEF+11+L(2),65:POKEF=12+L(2),78
10310 POKEF+13+L(2),79:POKEF+14+L(2),84:POKEF+15+L(2),72
```

CONT.

ED:

Thanks a lot for your excellent magazine. I should have bought it earlier! Some of my problems are solved now and I have a lot of new ideas also for my Superboard II.

I have a few questions also:

1. In the November 1980 issue, there is a program in BASIC on page 12 to stop listings. In the article, David A. Jones speaks of the output vector which is at locations 538 and 539 (decimal) as is seen in program line 50006. But in the December 1980 issue, he gives an explanation on this vector, but now he calls it the input vector. Which one is to be used? If it is the input vector, then line 50006 should be: POKE 536,34:POKE 537,2!

2. In the November 1980 issue, page 2, is an excellent article on the video modification, which I certainly will try out. In reading it, I saw the timing caps of the 74L5123 is 5pF (as far as I can read it!). Is this correct? (I don't think so!)

Furthermore, I have two projects for the Superboard:

1. An 8K RAM card with dip-switches switchable between (starting) address 2000 to 8000 in steps of 1K. The advantage is that you can now make this RAM card (without having to buy the 610 board for additional RAM) and if you have or buy a 610 afterwards, you simply move your 8K RAM card to another address when filling up the RAM's of the 610. The RAM on the 610 is from 2000 to 7FFF. As far as I can see, there is still 8K left and unused (is this so???) from 8000 to 9FFF, so this RAM can still be used.

2. EPROM card with fixed addresses from 8000 to 9FFF (=5K EPROM: 1x2708 + 2x2716: I happen to have these chips already). These two projects I have on paper but I can't try them out now as long as I am in Mexico. In July '83 I am returning to my country (Belgium) and will start working on it right away. If you're willing to publish the diagrams without the confirmation that it works good, let me know and I'll send a full explanation with the diagram.

Last question: if you connect a RS232 output on the 600 board, does this mean that you can't use the cassette any-

```
10312 POKEF+16+L(2),69:POKEF+17+L(2),82
10320 POKEH+10,16:POKEH+7+L,67:POKEH+8+L,79:POKEH+9+L,82
10322 POKEH+10+L,82:POKEH+11+L,69:POKEH+12+L,67:POKEH+13+L,84
10330 POKEH+8+L(2),65:POKEH+9+L(2),78:POKEH+10+L(2),83
10335 POKEH+11+L(2),87:POKEH+12+L(2),69:POKEH+13+L(2),82
10338 REM GRAPHHICS = WRONG TRY AGAIN & CORRECT ANSWER
10340 RETURN
14400 POKEG,82:POKEG+1,69:POKEG+2,83:POKEG+3,73:POKEG+4,83
14410 POKEG+5,84:POKEG+6,79:POKEG+7,82:POKEG+8,61
14420 POKEG+18,79:POKEG+19,72:POKEG+20,77
14430 POKEG+2+L(2),89:POKEG+3+L(2),79:POKEG+4+L(2),85
14440 POKEG+6+L92),73:POKEG+7+L(2),78:POKEG+8+L(2),80
14444 POKEG+9+L(2),85:POKEG+10+L(2),84:POKEG+12+L(2),67
14450 POKEG+14+L(2),76:POKEG+15+L(2),79:POKEG+16+L(2),82
14455 POKEG+13+L(2),79
14458 REM GRAPHICS = RESISTOR=OHMS & YOU INPUT COLOR
14460 RETURN
14500 Q=143:POKEA+2+L(3),208:POKEA+1+L(4),Q:POKEA+1+L(5),Q
14505 POKEA+1+L(6),Q:POKEA+1+L(7),Q:POKEA+L(8)+1,Q
      :POKEA+1+L(9),Q
14510 POKEA+2+L(10),207:POKEA+3+L(4),207:POKEA+3+L(5),208
14520 POKEA+2+L(5),128:POKEA+3+L(6),Q:POKEA+3+L(7),208
14530 POKEA+2+L(7),128:POKEA+3+L(8),Q:POKEA+3+L(9),208
14540 POKEA+4+L(4),61:POKEA+4+L(6),61:POKEA+4+L(8),61
14550 POKEA+7+L,82:POKEA+8+L,69:POKEA+9+L,83:POKEA+10+L,73
14560 POKEA+11+L,83:POKEA+12+L,84:POKEA+13+L,79:POKEA+14+L,82
14570 POKEA+8+L(2),67:POKEA+9+L(2),79:POKEA+10+L(2),76
14580 POKEA+11+L(2),79:POKEA+12+L(2),82;POKEA+9+L(3),67
14585 POKEA+10+L(3),79:POKEA+11+L(3),68:POKEA+12+L(3),69
14587 REM GRAPHICS=RESISTOR
14590 RETURN
15000 FORX=2TO10:READW(X):NEXT
15010 FORX=1TO10:READM(X):NEXT
15020 FORX=1TO7:READB$(X):NEXT
15030 FORX=1TO10:READA$(X):NEXT
15040 RETURN
15050 DATA1,2,3,4,5,6,7,8,9
15060 DATA0,1,2,3,4,5,6,7,8,9
15070 DATA".0","0","00","000","0000","00000","000000"
15080 DATABLACK,BROWN,RED,ORANGE,YELLOW
15090 DATAGREEN,BLUE,VIOLET,GRAY,WHITE
15100 CL=PEEK(129):SC=PEEK(130):ST=0:SU=212:CC=7
15105 IFPEEK(57088)<129THENST=192:SU=215:CC=62
15110 POKE129,ST:POKE130,SU:C$=" ":FORS=1TOCC:C$=C$+C$+" ":NEXT
15120 POKE129,CL:POKE130,SC
15122 REM C1/2/4-SCREEN CLEAR
15150 RETURN
16000 POKEF+4+L(3),71:POKEF+5+L(3),79:POKEF+6+L(3),79
16005 POKEF+7+L(3),68:POKEF+9+L(3),87:POKEF+10+L(3),79
16008 POKEF+11+L(3),82:POKEF+12+L(3),75:POKEF+13+L(3),33
16010 REM GRAPHICS = GOOD WORK
16015 FORX=1TO2000:NEXT:GOTO16200
16200 POKEF,72:POKEF+1,73:POKEF+2,84
16205 POKEF+4,83:POKEF+5,80:POKEF+6,65:POKEF+7,67:POKEF+8,69
16210 POKEF+10,66:POKEF+11,65:POKEF+12,82:POKEF+14,79
16215 POKEF+15,82:POKEF+17,47
16218 REM GRAPHICS = HIT SPACE BAR OR /
16220 GOTO16400
16400 P=57088:POKE530,1:POKEP,253
16402 M=239:N=247
16404 IFPEEK(P)<129THENM=M-223:N=N-239:POKEP,2
16406 IFPEEK(P)=MTHEN16420
16410 IFPEEK(P)=NTHEN16500
16415 GOTO16406
16417 REM C1/2/4-CHECK KEY BOARD
16420 POKE530,0:T=T+1:GOTO10010
16500 GOSUB15100
16510 PRINT"WELL ";W$;" OUT OF"
16520 PRINT:PRINTT;" YOU GOT ";V
16530 IFT>VTHEN16550
16535 PRINT:PRINT"GOOD WORK ";W$
16538 IFT=<VTHEN16570
16550 PRINT:PRINT"I THINK YOU SHOULD"
16555 PRINT:PRINT"TRY IT AGAIN WHAT"
16560 PRINT:PRINT"DO YOU THINK ";W$
16570 POKE530,0
16572 FORX=1TO8000:NEXT
16575 GOSUB15100
```

★ ★ ★

OSI—AFFORDABLE DATA BASE MANAGER

Now you can own a full featured DB Manager that doesn't cost more than your computer!

B&W FILE MASTER runs under OS65D V3.3, (video only). Single or dual drive.

FEATURES: User and/or pre defined files with coding options, formatted screen viewing and inputting, find, edit, update, delete & page. 'Screen', 'quick' and 'format' dump. Manual included.
.... only $55.00

ADD ON FEATURES:

Label print option	$45.00
Report generator	(Jan. '83)
Manual only (applied towards purchase)	$10.00

SPECIAL INTRODUCTORY OFFER!
B&W File Master & Label print option (incl. manual) $80.00

For more information contact:
BUNIN & WARD COMPUTER SERVICES
P.O. BOX 895 CHURCH STREET STA.
NEW YORK, NY 10008
(212) 434-5760

more? If so a (rather complicated) switch could be a solution.

Hope to get my problems solved by you or one of the readers. Thanks in advance.

Guy Vanderwaeren
Mexico

Guy:

1. The program in the November issue appears to be correct. The author must have made a mistake as to what to call it. The correct nomenclature is "OUTPUT" vector.

2. Yes it is!

PROJECT 1. As far as we know, only 4K is left for you to use.

RE-RS232. You would need a switch and also a -12v (-9v) power supply to provide true RS232. Check your intended peripheral - many, but not all require a swing through 0 to function.

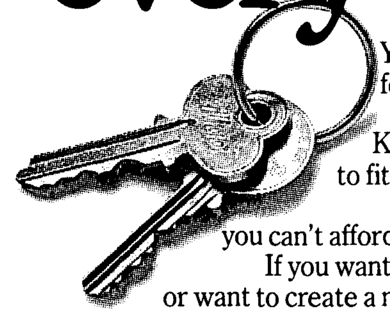
Brian.

* * * * *

ED:

My question concerns two programs produced by OSI, which I am using on a cassette-based C1P series II. The first is a modem routine SCX-107, designed for use on the C1, C4 or C8, which allows me to use the C1P as a remote terminal to campus comuters here at the University of Arizona. The second program SCX-106, designed to modify the C1P video

New KEYFAMILY™ small business computers for every size business.



You'll find that a KEYFAMILY computer is the small business system for you, no matter what size your business or your budget.

Whether you are your own boss or the boss of hundreds, the KEYFAMILY line is so complete, you can choose exactly the equipment to fit your needs.

And KEYFAMILY systems are so affordable, you can't afford to be without one.

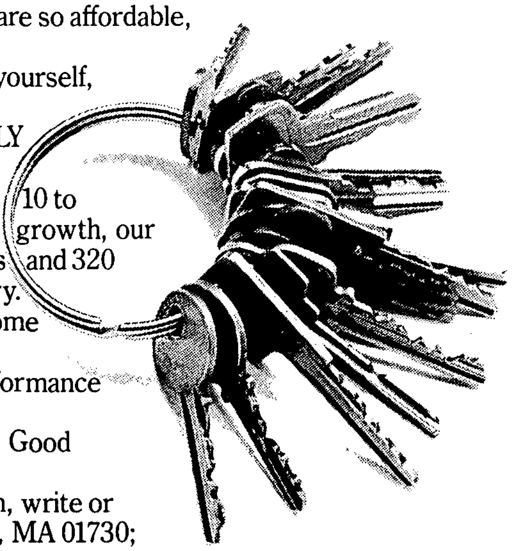
If you want a single-user system just for yourself, or want to create a multi-user system that will support up to six users, we have the KEYFAMILY system that will fit into your business.

Our multi-user systems allow you to access from 10 to 160 megabytes of hard disk storage. And for even more growth, our network clusters support up to 16 intelligent work stations and 320 megabytes of storage—the largest capacity in the industry.

So, before you commit to any computer system, come in and talk to us. We'll plan a small business system around KEYFAMILY specifically for you. It's the best deal for performance and price you can get anywhere.

That's what we mean when we say, "KEYFAMILY. Good for business."

For the dealer nearest you, or for more information, write or call M/A-COM Office Systems, Inc., 7 Oak Park, Bedford, MA 01730; 1-800-C•A•L•L•O•S•I, in Massachusetts call 617-275-4440.



M/A-COM OSI

KEYFAMILY. Good for business.

OSI COMPATIBLE PRODUCTS

56K 2-MHz Ultra Low Power CMOS Static Memory BoardMEM-56K $850

Partially Populated Boards (Specify address locations required) ...	**MEM-48K $750**
MEM Board uses the new 2K-Byte Wide Static RAM chips which are	**MEM-32K $550**
2716 EPROM compatible. Any 2K byte memory segment can be	**MEM-24K $450**
populated with RAM or EPROM (or left empty for use of Address Space	**MEM-16K $350**
by another board). Fully expandable to any memory size you will ever	**MEM- 8K $250**
need. No special addressing requirements, just solder in extra sockets	**MEM- 4K $200**
Extra 2K RAM Memory Chip	**$24**
Optional Parallel Printer Port	**-P $120**
Optional Calendar/Clock Software available in EPROM)	**-T $ 25**
Both options (Disk software mods provided for use of 6522 VIA on printer).	**-PT $125**

EXAMPLE USES:

C4P & C8P: Expansion to **4K RAM** of Basic workspace.
Parallel Printer Port — Reserve Serial Port for MODEM
Calendar/Clock Displaying on unused portion of screen.
Space for 5.75K of **Enhanced System Monitor EPROMS.**

All of this on 1 Board, using only one of your precious slots. Software for Enhanced System Monitor capabilities is continuously being developed and improved. As new EPROM Monitors are available, you may upgrade to them for any price differential plus a nominal $10 exchange fee. Another possibility is to fill any portion of the memory with Basic Programs in EPROM for **Power-on Instant Action.** This custom EPROM programming service is available at $25 per 2716 (Includes EPROM). Extra copies at $15 for each EPROM.

C4P-MF & C8P-DF:

Memory expansion to **48K.**
Add 6K **Memory** above BASIC for special software requirements.
Parallel Printer Interface and/or **Displaying Calendar/Clock.**
Add 1.75 K **Enhanced System Monitor ROM.**

C3: Up to **56K of Memory** Expansion — can be addressed for **Multiuser.**
(Optionally, each user can have his own **Dedicated Printer Port).**

C1P,C4P & C8P FLOPPY DISC CONVERSIONS:

Memory/Floppy Board (Includes MI48P1 ROM) .... **MEM F-16K $450**
C1P-600 Board Adapter & Cable .... **A600/48 $ 50**
Additional Memory/Printer/Times .... **(See MEM Board Prices)**
5¼" Drive/Case/Power Supply & Cable to MEMF Board **FD5 $399**

IEE-488 INTERFACES AND SOFTWARE:

The General Purpose Instrumentation Bus (GPIB Controller interface is available for all OSI Computers. Machine code GPIB Drivers are linked to Basic to provide easy control of IEEE-488 instruments which is equal to the best of Hewlett-Packard Controllers and far superior to most others. Basic Commands for Serial Poll, Parallel Poll, IFC Clear, full Local/Remote Control, Respond to SRQ Interrupts, Send Trigger, do Formatted Input/Output, Direct Memory Input/Output and MORE. Interface includes IEEE-488 Ribbon Cable/Connector.

GPIB Controller Interface for C2, C3, C4 and C8 Systems .... **GPIB 4-488 $395**
GPIB Software for **OS-65D** (Add -8 for 8" or -5 for 5") .... **GPIB 488-D $ 70**
GPIB Software for **OS-65U** .... **GPIB 488-U $100**
GPIB Software on two 2716 EPROMS for**ROM** based systems ... **GPIB 488-R $100**
Add Optional **Parallel Printer Interface** to GPIB 4-488 .... **-P $120**
Add Optional **Calendar/Clock** to GPIB 4-488 .... **-T $ 25**
Add **2K RAM** to GPIB 4-488(Specify location, $4000-$BFFF & $DOOO-$$EFFF available)**-M $25**
GPIB Controller for C1P, Includes Software, Clock, All Features of ROMTERMS, & space for 6K EPROM .... **GPIB 6-488R $395**
Add Optional Parallel Printer Interface to GPIB 6-488R .... **-P $120**

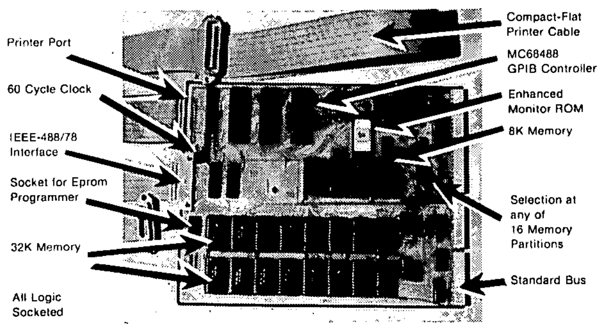
EPROMS:

C1P ROM with 24/48 Col Display for Series II, Smart Terminal, Line Editing, Corrected Keyboard Screen Clear and More .... **ROM-TERM II $59.95**
C1P ROM with 24 Col Display. Other ROM-TERM II Features, Disk Boot, and ROM/Disc Basic Interchange .... **ROM-TERM $59.95**
C4P-MF/C8P-DF Disk warm start, changed IRQ Vector and just flip switch for Serial or Video System with Corrected Keyboard .... **SYNKEY $39.95**

ENHANCED MONITOR ROMS FOR USE ON GPIB 4-488 & MEM BOARDS:

Expanded Support for **C4P & C8P** Featuring Calendar/Clock, Line Edit, Smart Terminal, Memory Files, Parallel Printer Control, Corrected Keyboard, All Features of ROMTERMS, **Disk Support** with Warm Start and More **MI48P1 $59.95**
Expanded C2 Monitor with Calendar/Clock Software, Hard Disk Boot, Warm Start and Multi-User Control for C2 Systems **MIC2-1 $59.95**

IEEE-488 CONTROLLER INTERFACE



THE GPIB 4-448 INTERFACE BOARD CONVERTS ANY OSI COMPUTER INTO AN IEEE-488 INSTRUMENT BUS CONTROLLER!

BENEFITS — Provides a Sophisticated Instrumentation Controller at very low cost (often saving thousands of Dollars). The combination of IEEE-488 Instrumentation Controller and High Capacity Hard Disk file storage available on OSI Computer systems is available at a fraction of the cost required by the nearest competitor. The IEEE-488 Bus, also known as the GPIB,HP-IB or IEC-625 is the most popular International Standard for connecting instrumentation systems. This 16-line bus is designed to interconnect and control up to 15 instruments at a time. Currently, over 2000 different instruments are available to work on this bus. They include: Plotters, Digitizers, Printers, Graphic Displays, Recorders and a multitude of specialized Test/MeasurementControl Equipment.

EPROM-ABLE — Can be used with a C4-P to create a dedicated IEEE-488 controller.

C2-D MULTIPLE USER SYSTEMS

SAVE — 2 and 3 user Time Sharing Systems are available on the C2-D Winchester Disk Computer at a considerable cost savings from C3 Multiple User Systems. The 3 user C2-D System can be expanded to include a word processing printer, 4 other parallel printers and 3 serial printer interfaces.

COMPATABLE — The special C2-D Multi-User Executive Program is 100% compatable with OS-65U V1.4. The Multi-User Real Time Clock, Memory Partition Control and IRQ Interrupt Management are done on the Micro Interface Memory Board. Thus, the CPU board is not modified and remains in factory condition.

CONVERSIONS — The Up-Grade of your existing C2-D Computer to Multiple User Configuration is also available. Call for details.

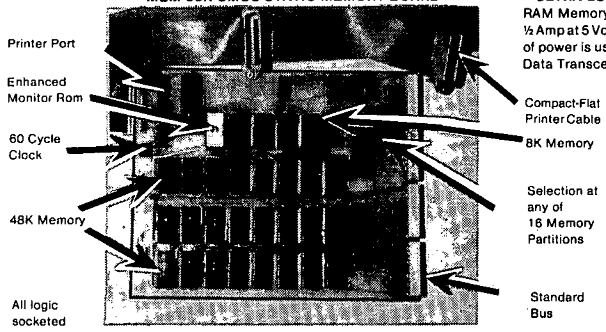
FLOPPY DISK UPGRADES FOR C1P, C4P & C8P

Our Memory/Floppy Board provides easy conversion of 502 and 600 CPU Computers to Floppy Disk Operation. The **MEMF** Board has a floppy disk interface which includes a data separator and the ability to automatically lift the disk drive heads — your floppy disk lifetime will be extended many times. You will retain the cassette interface for your existing software; which can easily be converted to Disk.

This **MEMF-16K** Board is populated with **16K RAM** (50K possible) and has features of the MEM CMOS Static Memory Board with an added floppy interface. The low power memory means extra power supply not required. **ROM Basic** is retained even when Board is populated for 48K Disc Basic. An optional Parallel Printer Port and Real Time Calendar/Clock is on board.

Complete Ready to Run conversion kits with 5¼" or 8" Disk Drives are available.

MEM-56K CMOS STATIC MEMORY BOARD



ULTRA-LOW POWER — By using CMOS Static RAM Memory, the total power consumption is about ½ Amp at 5 Volts when populated for 48K. In fact, most of power is used by the Address Line Buffers and the Data Transceivers.

MULTI-USER — Can be addressed for any of the 16 multi-user memory partitions. The low power and single memory board/partition simplify installation and provide a typical $1400 saving for a 3-user system.

MICRO-INTERFACE
3111 SO. VALLEY VIEW BLVD., SUITE I-101
LAS VEGAS, NEVADA 89102
Telephone: (702) 871-3263

Check with your local Dealer or Order Direct.
Phone orders accepted.
TERMS: Check/Money Order/Master Charge/VISA
Sent POSTPAID ON PREPAID ORDERS.
Foreign Orders: Prepaid only.
Add 5% for handling/shipping.

driver, allows video swapping between the normal 24X24 character display format (POKE 251,0) and a text format (POKE 251,1) of 12X48 char./line... ....except when the C1P is operating in the modem mode. The problem then being that although characters of reduced width are presented to the screen, there is a blank swath down the display center (columns 53405 thru 53412 and below of the video RAM). Further, the start position of each line alternates to the right and left of this swath as the text is scrolled, as though the 24X24 video map is still in use. In short, the same effect as when poking the control register 55296 with 1 when the system is first booted up in the local mode.

Since the dealer from which I purchased the C1P and both programs has not supplied much documentation on either, other than the mode toggling, is it possible that there are further POKE parameters that I'm unaware of to overcome this apparent incompatibility between the MODEM and VIDEO SWAP software?

Any information you might provide would be greatly appreciated.

Scottie Cantrell
Tucson, AZ 85715

* * * * *

ED:

I will try to keep this letter rational and refrain from the use of profanity. I am totally outraged. At this point I don't know whether the rage is directed at you or at M/A-COM. 'Column One' of the July and August issues of PEEK (65) were dedicated to the wonderful things that M/A-COM were developing; the fantastic documentation that was being produced; the fierce dedication to their dealers and their progressive view of the future.

'MIS Weekly' reported that M/A-COM announced that they would divest themselves of the OSI division by the end of the year. Someone is lying to the OSI users! Whether it is PEEK (65) or M/A-COM probably doesn't really matter. The charade exists.

Owners of OSI hardware are probably a unique bunch of 'nuts'. We have persevered with all the atrocites that a manufacturer could bestow on a user: poor documentation, marginal software and factory support that was practically non-existent. The reason for purchasing and staying with OSI, at least in my case, was the hardware. It was reasonably priced, rock solid and gave me the opportunity to write software for an excellent microprocessor with a logical instruction set.

I don't care if my system doesn't look pretty. Moulded plastic cases and multi-colored logos do not impress me at all. It is annoying that the Basic-in-ROM garbage collector doesn't really work. I regret that the Assembler/Editor code is a hodge-podge of patches. The ROM Monitor should have been written with more subroutines rather than in-line code making those routines available to user programs. All of these things could have been corrected and many of us would have been pleased. But, we live with the little problems. The hardware is great! My C2-4P has demonstrated 100% up-time since late 1979. The unforgivable sin is being lied to.

I have been in the computer business for over twenty-five years (all with the same major main-frame manufacturer). I believe that we are number two in this industry through dedication to several premises. One of these is integrity. We, like everyone else have released marginal hardware or software, but when it happens there is an all-out effort to correct the problem. We do what is necessary to keep the customer happy. M/A-COM does not appear to be dedicated to the same ideals. It seems as though their motto should be taken from the Woody Allen movie, "Take the Money and Run".

It is hard to believe that M/A-COM spent approximately one year formulating plans, improving the hardware/software and then just a few months after their 'New Products' announcements, decide to get out of this segment of the business. Did they expect that after a year of virtual silence that the whole world would beat a path to their door? Not likely! Any farmer will tell you that it takes time for seeds to sprout and grow. Maybe they bought some technology; picked the brains of their experts and decided to start from the beginning without the OSI legacy (good or bad). At any rate the next owner of OSI will have a more difficult time. Until significant changes are demonstrated by management, I could not recommend OSI hardware to anyone.

Please do not misunderstand the reasons behind this letter, I like PEEK (65) and will continue my subscription. I am just outraged over the way that OSI users are treated.

Harry B. Pye
Lansdale, PA 14446

Harry:

I am very glad you wrote to us. Here at PEEK(65) we are not feeling the anger you are, but are also quite confused. We were delighted at what we heard from M/A COM about their big plans for OSI, very happy with the new machines, looking forward to still more newer and better machines, then ... BOOM! We were as surprised as you were. Fortunately, we did have a couple of contacts at OSI, and were able to determine that the divestiture does not mean as much as one might think. OSI will continue. New machines will be forthcoming. In fact, as you read this, COMDEX will be history and it is rumored that OSI will have announced their multi-processing machine. The ambitious program to develop new documentation will go on ... but under new ownership. Why M/A COM continued for so short a time we do not know any more than you do.

Flat Rate

DISK DRIVE OVERHAUL

One Week Turnaround Typical

Complete Service on Current Remex, MPI Siemens and Shugart Floppy Disk Drives.

FLAT RATES

8" Double Sided Remex	$170.00*
8" Single Sided Siemens	$150.00*
5¼" M.P.I. Drive	$100.00*

Other Rates on Request.

*Broken, Bent, or Damaged Parts Extra.

YOU'LL BE NOTIFIED OF

1. The date we received your drive.
2. Any delays and approximate time of completion.
3. Date drive was shipped from our plant.
4. Repairs performed on your drive.
5. Parts used (# and description).
6. Any helpful hints for more reliable performance.

90 day warranty.
Ship your drive today.
Write or call for further details.

We Sell Parts

PHONE (417) 485-2501
FESSENDEN COMPUTER SERVICE
116 N. 3RD STREET OZARK, MO 65721

Now that I have made lots of brave statements about what "will" happen, let me clarify that PEEK(65) has no magic wand, no deep throat style inside source into what is happening in the complex corporate structure called M/A COM. We talk to them, we interview them by phone and in person, we report to you what have we heard. We make observations and try to make intelligent predictions and projections. But when things break really fast, remember what you read in PEEK(65) is a month out of date because of publishing lead times, and is just what we were told.

And in any case, we don't know yet that we have been "lied to." Bill Chalmers, based on our contacts with him, is still very determined to continue OSI's development. The new machines are here. The manuals come out regularly. The whole thing might be one of those corporate financial transactions in which new names appear on the letterhead but nothing really changes at all. Let's wait and see.

As far as PEEK(65)'s "complicity" in any changes at M/A COM goes, you over estimate us. We calls 'em like we sees 'em, or like they (and you) tell us, nothing more. I hope we sometimes here more (just because we have more sources) or hear quicker than some others, but if we don't, we can't tell you! Anyway, we will keep trying to keep you up to date. And while we might be dead wrong, we will not lie to you. If I know it, you know it.

AL.

PEEK(65) INDEX FOR 1982

ITEM	MO	PG
1-DISK COPY LISTING	JUL	8
2716 REPLACES CG ROM	JAN	19
32 CHAR MOD FOR C1P	MAY	25
64 CHR FOR C1P	AUG	13
65D 3.2 HEAD UNLD	JUN	10
65D DATA FILE HANDLR	FEB	13
65D ML PROG SAVE	DEC	12
65D RANDOM FILES	FEB	9
65D V3.3 REVIEW	JAN	2
65D3 DIRECTORY SORT	APR	4
65D3 SCREEN CLEAR	FEB	14
65D3 SERIES PT 4	FEB	9
65D3.2 STEP RATE	AUG	20
65D3.3 COMMENTS	OCT	23
65D3.3 GUIDE	SEP	19
65D3.3 KEYBOARD RTN	JUL	20
65D3.3 NOTES	MAY	15
65D3.3 PT 5	MAR	2
65D3.3 REVIEW	MAR	13
65U 1.3 TERM INDEPEN	APR	11
65U 1.42 CODE FIX	SEP	23
65U 1.42 REVIEW	MAY	2
65U 1.42 REVIEW	NOV	9
65U GOODIE BOX	FEB	2
65U LEVEL 3 INFO	FEB	14
65U PASSWORD HIDER	MAY	2
65U PASSWORDS	FEB	2
65U POKES	FEB	2
65U QUIRKS	JUL	23
65U QUIRKS (CONT'D)	OCT	23
65U SCREEN FORMAT	FEB	21
65U TERMINAL EMULATR	JAN	18
65U USER-DEFINED IPT	JUN	18
65U WAIT FOR/CLEAR	FEB	22
8" DRIVE FOR C4	SEP	9
8" DRIVE FOR C4 (2)	OCT	2
8K SWITCHED RAM	DEC.	17
A/R INSTALLATION	NOV	16
AARDVARK EDITOR	SEP	21
ADD DATE IN DMS	FEB	21
ADD MEMORY FOR C4P	JUN	10
ASSEMBLER NOTES	MAY	13
ASSEMBLER ERROR MSGS	APR	3
AUDIT TRAIL IN DMS	NOV	25
AUTO EXECUTE IN CP/M	JAN	18
AUTOMATIC DIR	DEC	8
BACK SPACE	DEC	14
BASIC BACKSPACE	DEC	14
BASIC IMMEDIATE MODE	DEC	2
BASIC & INTEGERS	NOV	27
BASIC INPUT ROUTINE	OCT	14
BASIC PRINT AT	DEC	14
BASIC ROM FIXES	SEP	4
BASIC SCREEN CLEAR	DEC	14
BASIC WARM START	MAR	6
BEGINNER'S COLUMN	SEP	2
BEXEC* MODS	DEC	8
BEXEC* NAME CHANGE	AUG	13
BIGBAG JEWELS	AUG	7
BLIND PASSWORD ENTRY	JUN	22
BREAK & SAVE	DEC	11
BREAK SWITH MOD	MAR	6
BURGER KING SYSTEM	SEP	7
BUSI-CALC	SEP	8
BUSI-CALC REVIEW	OCT	6
BYEBYE YASUO	AUG	22
C1P 32 CHAR MOD	MAY	25
C1P 64 CHR VIDEO	AUG	13
C1P AND OKI 82A	APR	2
C1P ASM-ED	NOV	2
C1P BEXEC* MODS	DEC	8
C1P CONTROL CHARS	AUG	15
C1P EXMON	NOV	2
C1P FAN	MAY	23
C1P GAMES	MAR	15
C1P GRAPHICS	MAR	18
C1P INFO	MAR	6
C1P KEYBOARD PROG	JAN	2
C1P LONG LINES	MAY	5
C1P MEMORY MAP	JUN	17
C1P MOD SMORGASBORD	JUN	5
C1P SAVE MEMORY	FEB	16
C2 8" DRIVE	SEP	9
C2 8" DRIVE (2)	OCT	2
C3 GAMES PROGS	JAN	21
C4 8" DRIVE	SEP	9
C4 8" DRIVE (2)	OCT	2
C4P 1-DRIVE COPY	JAN	4
C4P CASSETTE LOAD	JAN	21
C4P CONTROL POKE	FEB	15
C4P GRAPHICS	DEC	14
C4P MEMORY ADD	JUN	10
C4P MODS	APR	2
C4P MONITOR	APR	16
C4P POWER SUPPLY	APR	2
C4P RS232 MOD	JAN	20
C4P SCREEN WINDOW	JAN	7
C4P SERIAL INTERFACE	JAN	19
C4P SYSTEM MODS	MAR	17
C4P TV ADJUSTER	DEC	14
C8 8" DRIVE	SEP	9

yum!

BACH-MANN

ONLY $27.50 and FULLY GUARANTEED! My kids say it is BETTER THAN ATARI PAC MAN!®

FAST ASSEMBLY PROGRAM 'RUN' LIKE BASIC. BEWARE THE INTELLIGENT ROBOTS WHILE YOU TRY TO EAT THE MAZE.

REQUIRES OS-65U AND SERIAL TERMINAL WITH CURSOR ADDRESSING. SUPPLIED ON 8-INCH DISK. EASILY CONFIGURED TO YOUR TERMINAL OR STATE TYPE WHEN ORDERING. COMPLETE INSTRUCTIONS INCLUDED... YOU CAN'T GO WRONG.

ALL OF SANDERS' SOFTWARE INCLUDES 30 DAY SATISFACTION GUARANTEED OR MONEY BACK WARRANTY

J AND T ASSOCIATES
2338 RIVIERA DRIVE
VIENNA, VA. 22180

®trademark Midway Mfg. Co.

SMART TERMINAL SOFTWARE

$35.00

A COMPLETE MODEM COMMUNICATIONS PACKAGE FOR OS65U SERIAL SYSTEMS. VERY EASY TO USE. INCLUDES A FULL RANGE OF KEYSTROKE CONTROLLS FOR DUPLEX, DELAY, FILE HANDLING, AND SIMULTANEOUS PRINTER OPERATION. OVER 200 USERS ARE SENDING AND RECEIVING PROGRAM AND DATA FILES, OR JUST COMMUNICATING. IN DAILY USE WITH CDC AND IBM MAINFRAME WORK. SUPPLIED ON 8 INCH DISK WITH COMPREHENSIVE MANUAL. ASSEMBLY PROGRAM WITH BASIC DRIVER.... EASILY CONFIGURED TO ANY ACIA OR UART ADDRESS, OR STATE DESIRED CONFIGURATION WHEN ORDERING.

INCLUDES HEAVILY COMMENTED ASS'Y PROGRAM LISTING!

C8 8" DRIVE	OCT 2
C8PDF IN BUSINESS	OCT 23
C8P TV ADJUSTER	DEC 14
CALENDAR PROG	AUG 15
CASSETTE LOAD IN C4P	JAN 21
CASSETTE LOADER	FEB 18
CASSETTE CORNER	MAR 15
CASSETTE CORNER	APR 6
CASSETTE CORNER	MAY 7
CASSETTE CORNER	JUN 13
CASSETTE CORNER	AUG 14
CASSETTE CORNER	NOV 2
CASSETTE CORNER	DEC 8
CBBS FOR HEARING IMP	JAN 22
CEGMON FINAL WORD	NOV 20
C1P/MODEM	APR 6
CLUTTER CLEARER	OCT 10
CMOS LOGIC DESIGN	JAN 10
CODE FIX FOR U 1.42	SEP 23
CODE TO DOS RAM	MAY 3
COMMO PROG FOR 65U	JAN 18
COMPUTED GOTO CANDO	FEB 22
COMPUTER CUBE	MAR 8
CONDITIONAL BRANCH	SEP 22
CONFIG BUS. SYS RVEW	OCT 9
CONTROL POKE FOR C4P	FEB 15
COPY FOR 1 DISK	JUN 15
COPY FOR 1 DISK	JUL 8
CP/M AUTO-EXECUTE	JAN 18
CP/M SUPPORT	JAN 15
CP/M UPDATE TO 2.2	JUN 21
CPM PUBLIC PROGS	NOV 23
CUBIST ART	OCT 27
DATA SELECTOR	JUL 17
DATE ADDED IN DMS	FEB 21
DBASE II REVIEW	AUG 9
DEC-HEX CONVERTER	JAN 22
DELETE BY RENAME	DEC 11
DIAGNOSE RAM	APR 25
DIGITAL DESIGN	JAN 10
DIR AUTOMATED	DEC 8
DIRECTORY OF DP	MAY 26
DIRECTORY SORT	NOV 5
DIRECTORY SORT - 65D	APR 4
DISK COPY FOR 1 DRIV	JAN 4
DMS REPORT WRITER	JAN 3
DMS/DBASE II (2)	AUG 9
DMX WITH LEVEL 3	JAN 22
DOS RAM GETS ML CODE	MAY 3
DP DIRECTORY	MAY 26
DQ SECRETARY	SEP 21
EDMAFL AUDIT TRAIL	NOV 25
EDMAFL DATE MOD	FEB 21
EDUCOMPUTER PLANNER	FEB 12
EPROM PROGRAMMING	MAR 22
ERROR MSGS FOR ASM	APR 3
EXAMINE BASIC PROGS	NOV 11
FAST CASSETTE LOADER	FEB 18
FBASIC CRITIQUE	MAR 21
FDSAP REVIEW	APR 7
FIX FOR 1.42 CODES	SEP 23
FIXES FOR ROM BASIC	AUG 2
FORMAT INPUTS	MAR 17
GAMES FOR C1P	MAR 15
GAMES FOR C3	JAN 21
GAMES REVIEW	MAY 7
GAMES REVIEW	JUN 15
GC BUG	JAN 21
GC FIX WORKS	MAR 22
GRAPHICS ON C1P	MAR 18
GRAPHICS PLOTTER	DEC 14
GRAPHICS TO PRINTERS	JAN 22
HAPPY USER	NOV 16
HEXDOS REVIEW	FEB 4
HEXDOS REVIEW II	MAY 8
HEXDOS USER REPORT	JUN 25
HI-RES GRAPHICS	DEC 14
HYBRID DISKDIR SORT	NOV 5
IF-THEN-ELSE SIMUL	APR 25
INDEX TO MICRO LIT	OCT 27
INDIRECT COMMAND FIL	APR 20
INPUT IN BASIC	OCT 14
INTERNAL CONVERSION	MAY 25
INTERVIEW WITH OSI	JUL 2
INVISIBLE PASSWORDS	MAR 10
JAPANESE CHARACTERS	JAN 1
JOYSTICKS ON OSI	SEP 22
KEYBOARD INPUT	MAR 17
KEYBOARD ROUTINE	JAN 2
KYUTIL & 1.3 UP	NOV 27
KYUTIL MODS	MAY 4
LEVEL 3 + DMX	JAN 22
LEVEL 3 INFO	FEB 14
LEVEL 3 TOF FIX	JUL 18
LEVEL 3 WAIT FOR/CLR	FEB 22
LIFEBOAT SUPPORT	JAN 15
LOAD CASETTE - C4P	JAN 21
LOG ON AND OFF	NOV 14
LONG LINES FOR C1P	MAY 5
LOW POWER RAM CHIPS	APR 26
MACHINE LANG SAVE	JAN 9
MEMORY ADD FOR C4P	JUN 10
MEMORY MAP - C1P	JUN 17
MEMORY SAVE - C1P	FEB 16
MICRO PUBS IN RVIEW	FEB 12
ML PROG SAVE	DEC 12
MODEM PROGRAM HINTS	OCT 26
MODEM/C1P	APR 6
MODS FOR C1P - LOTS	JUN 4
MODS TO C4P	APR 2
MONEY ROUNDER	JUN 23
MONITOR DISASSEMBLY	APR 16
MONITOR ROM LOCS	JUN 22
MONOHYBRID CROSSES	AUG 18
MORRIS GC FIX	MAR 22
MPI DATA SELECTOR	JUL 17
NBTEL P-SYSTEM	NOV 21
NEW COMPUTER PICS	APR 1
OKI 82A AND C1P	APR 2
OS-DMS DESCRIPTION	JUL 3
OS/DMS VS DBASE II	JUL 3
OSI MODEM PROG HINTS	OCT 26
OSI VIP INTERVIEW	JUL 2
P-SYSTEM ON OSI	AUG 25
PASSWORD HIDER	MAR 10
PASSWORD HIDER	JUN 22
PASSWORDS IN 65U	FEB 2
PASSWORDS IN 65U	MAY 2
PHILA OSI USR GRP	JAN 19
PICTURES - NEW MCHNS	APR 1
PLOTTER	DEC 14
POKE FOR SPACE SUPPR	APR 19
POKES FOR SCRN WNDOW	JAN 7
POKES IN 65U	FEB 2
PRINT AT	DEC 14
PRINTER SWAP PROG	APR 26
PRINTER TESTER	SEP 3
PRINTER UN-STALL	APR 10
PROGRESSIVE MON MODS	OCT 25
PROXY-80 BOARD	AUG 19
PUBLI PROGS IN CP/M	NOV 23
PUNNETT SQUARE PROG	AUG 18
RAM 8K	DEC 17
RAM - LOW POWER	APR 26
RAM DIAGNOSTIC	APR 25
RAM FROM ROM	MAY 19
RANDOM FILES IN 65D	FEB 9
RANDOM NUMBER TEST	MAY 26
RANGE FOR KYUTIL	JUN 4
RENAME = DELETE	DEC 11
REVIEW - 65D3.3	MAR 13
REVIEW - 65D3.3	MAY 15
REVIEW - 65U 1.42	MAY 2
REVIEW - 65U 1.42	NOV 9
REVIEW - ASSEMBLER	MAY 13
REVIEW - BUSI-CALC	OCT 6
REVIEW - CONFIG B S	OCT 9
REVIEW - DBASE II	AUG 9
REVIEW - FBASIC	MAR 21
REVIEW - FDSAP	APR 7
REVIEW - GAMES	MAY 7
REVIEW - GAMES	JUN 15
REVIEW - HEXDOS	FEB 4
REVIEW - HEXDOS (2)	MAY 8
REVIEW - SUPERSPOOLR	JUL 20
REVIEW - TERM PROG	JUL 18
REVIEW - TMAKER-II	APR 8
REVIEW - WP 3.2	JAN 13
REVIEW - WP6502 1.3A	AUG 13
ROM BASIC FIXES	SEP 4
ROM BASIC FIXES (1)	AUG 2
ROM TO RAM	MAY 19
ROUND $ AMOUNTS	JUN 23
ROUNDING ERRORS	NOV 23
RS232 FOR C4P	JAN 21
RUBIK DIAGRAM	OCT 27
RUBIK'S PROGRAM	MAR 8
SAVE AFTER BREAK	DEC 11
SAVE FOR C4P	JAN 9
SAVE ML PROGS	DEC 12
SAVE MEMORY - C1P	FEB 16
SCREEN CLEAR	DEC 14
SCREEN CLEAR FOR 65D	FEB 14
SCREEN CLUTTER CLEAR	OCT 10
SCREEN FORMAT IN 65U	FEB 21
SCREEN FORMAT INPUTS	MAR 17
SCREEN WINDOW -C4P	JAN 7
SELECTRIC INTERFACE	OCT 17
SERIAL PORT FOR C4P	JAN 19
SINGLE 8" DRIVE COPY	JAN 4
SMARTERM TERMS	JUN 25
SORT DISK DIRECTORY	NOV 5
SPACE SUPPRESS POKE	APR 19
STARTUP FOR L 3	NOV 13
STAT02 EXTENDED	JAN 2
SUPERSPOOLER REVIEW	JUL 20
SWAP PRINTER PROG	APR 26
SWIMMING VIDEO FIX	FEB 22
TERM INDEPENDENT 1.3	APR 11
TERMINAL PROG FOR U	JAN 18
TERMINAL PROG REVIEW	JUL 18
TMAKER-II REVIEW	APR 8
TOP OF FORM IN L 3	JUL 18
TV ADJUSTER	DEC 14
UN-STALL PRINTER	APR 10
UNLOAD HEAD FOR 65D	JUN 10
UPDATING CP/M	JUN 21
USER # IN LEVEL 3	FEB 14
USER GROUPS	FEB 19
USER-DEFINED INPUTS	JUN 18
USR(X) EXPLAINED	FEB 3
VARIABLE RECLEN	MAR 2
VIDEO SWIM FIX	FEB 22
WAIT FIX POKE CHANGE	APR 19
WAIT FOR/CLEAR L 3	FEB 22
WARM START BASIC	MAR 6
WP 3.2 REVIEW	JAN 13
WP3.3 RESPONDS	JUL 22
WP6502 1.3A REVIEW	AUG 13
WP6502 RESPONDS	AUG 20
XMON GRAPHICS DUMP	JAN 9
YOUNGSTOWN CBBS	JAN 22

AD$

FOR SALE - OSI C4P 8K Cassette Color Computer, $350.00, Don Colin, P.O. Box 1238, Defuniak Springs, FL 32433. Evenings (1-904-892-5295).

* * * * *

O.S.I. C4P with CA-20 board (populated), D&N BMEM-CM9F board with 8K RAM, socketed but unpopulated disk controller, new D&N 96 protoboard. Lambda 10-A supply. SAMS/OSI manuals, $625. Paul Pagel, 4 Roberts Rd.,Enfield, CT 06082.

PEEK (65)
The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117

BULK RATE
U.S. POSTAGE
PAID
Owings Mills, MD
PERMIT NO. 18

DELIVER TO:

PEEK (65)
P. O. BOX 347
OWINGS MILLS, MD. 21117

SUBSCRIPTION RATES FOR 12 ISSUES (ONE YEAR), EFFECTIVE WITH THE JULY, 1981 ISSUE. ALL RATES QUOTED IN U. S. DOLLARS.

PLEASE FILL OUT AND RETURN WITH CHECK OR MONEY ORDER.

( ) $15.00 ENCLOSED. U. S. (MARYLAND RESIDENTS ADD 5% SALES TAX.)
( ) $23.00 ENCLOSED. CANADA AND MEXICO, 1ST CLASS, SURFACE.
( ) $35.00 ENCLOSED. SOUTH AND CENTRAL AMERICA. AIR MAIL.
( ) $35.00 ENCLOSED. EUROPE. AIR MAIL.
( ) $40.00 ENCLOSED. ALL OTHER. AIR MAIL.

NAME _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ STREET _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

CITY _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ STATE _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

ZIP CODE _ _ _ _ _ _ _ _ _ _ _ _ _ _ COUNTRY _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

PLEASE SEND THE FOLLOWING BACK ISSUES. I ENCLOSE:

( ) $2.00 EA. U. S. SURFACE. (MARYLAND RESIDENTS ADD 5% SALES TAX.)
( ) $2.50 EA. CANADA AND MEXICO. SURFACE.
( ) $3.00 EA. SOUTH AND CENTRAL AMERICA. SURFACE.
( ) $3.00 EA. EUROPE. SURFACE.
( ) $3.50 EA. ALL OTHER. SURFACE.

Vol 1. 1980

( ) JAN #1	( ) FEB #2	( ) MAR #3	( ) APR #4
( ) MAY #5	( ) JUN #6	( ) JUL #7	( ) AUG #8
( ) SEP #9	( ) OCT #10	( ) NOV #11	( ) DEC #12

Vol 2. 1981

( ) JAN #1	( ) FEB #2	( ) MAR #3	( ) APR #4
( ) MAY #5	( ) JUN #6	( ) JUL #7	( ) AUG #8
( ) SEP #9	( ) OCT #10	( ) NOV #11	( ) DEC #12

Vol 3. 1982

( ) JAN #1	( ) FEB #2	( ) MAR #3	( ) APR #4
( ) MAY #5	( ) JUN #6	( ) JUL #7	( ) AUG #8
( ) SEP #9	( ) OCT #10	( ) NOV #11	

INDEXES ARE INCLUDED IN THE JANUARY & DECEMBER 1981 ISSUES.