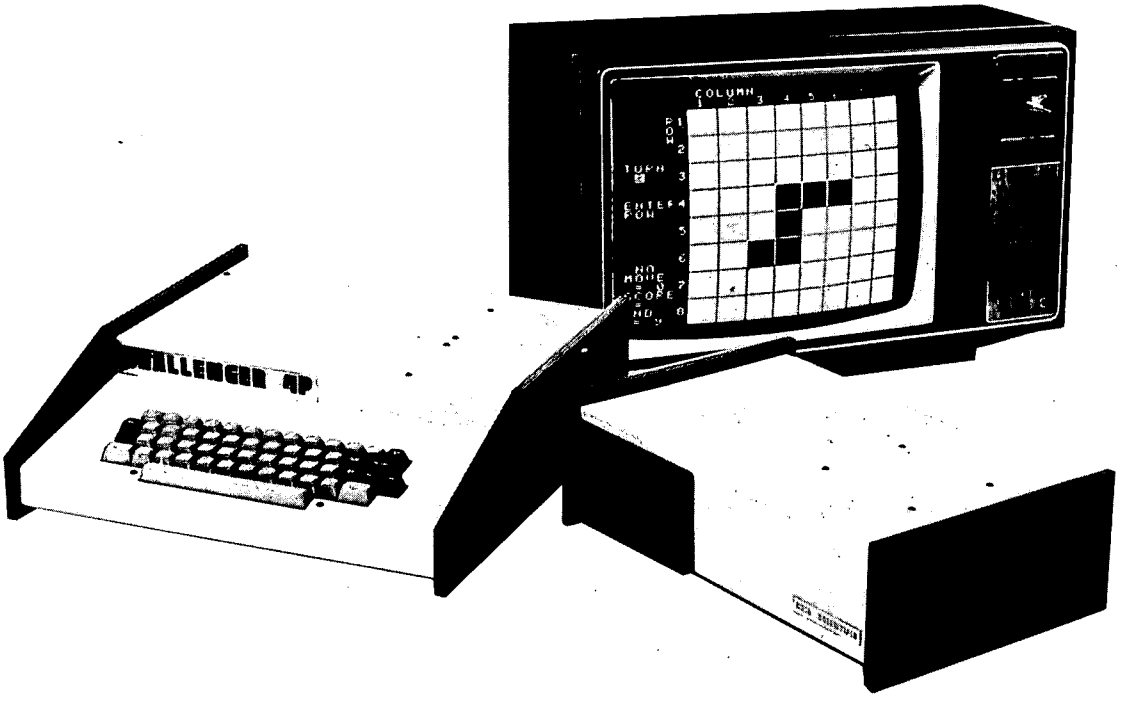
PEEK (65)

Editor: Al Peabody

VOL. 1, NO. 12, DECEMBER 1980

The Unofficial OSI Users Journal

P.O. Box 347
Owings Mills, Md. 21117
(301) 363-3267



INSIDE:

Column One

This issue marks the end of the first year of the life of PEEK(65). I am not sure whether I am more excited about the things which have happened, or the things which will happen soon. It has certainly been an eventful year, but 1981 promises to be even more so for both of us.

PEEK(65) has grown from a tentative effort by two guys operating out of a basement, sent primarily to OSI dealers and printed on little paper, to almost a real magazine, printed big enough to bind in your notebook, sent out to thousands of subscribers worldwide, still including all known OSI dealers. When we started PEEK, Dick and I wondered if: 1) anybody would subscribe; 2) we could break even; 3) the factory would sue us. We are breaking even, though making money in the magazine business is as difficult as everybody told us it would be (but we don't really care -- PEEK(65) is a labor of love!). Lots of folks have subscribed, though not enough to satisfy us. And the factory has been just great, not hassling us but supporting us, answering questions when we asked them. They even (sometimes) return our phone calls!

My arm is sore from patting us on the back. What about next year? Glad you asked. I believe we will never make it by resting on our laurels, so we have planned a blockbuster year for 1981.

First of all, more, more, more. PEEK(65) is now at 20 pages, and we hope to hold at that level through the holiday slow season, then grow still more. What other magazine, even if it is 300 pages long, has 20 pages of info and ads for OSI users? To this end, let me repeat, YOUR ARTICLES AND LETTERS ARE WELCOME AT PEEK(65)!

Secondly, special features will be a big part of our new year plans. I have on my desk right now an article which knocks my socks off. It's called "Assembly Programmer's Guide to OSI Board Interfacing." It clearly and concisely explains the I/O procedures, memory locations, chip types and methods you need to do any kind of I/O with any kind of OSI board: CPU, printer driver, CA-10-X, even disk drive controllers. Want to know how to make your computer share files with another computer, or drive a new kind of printer, or input over the CA-10 from a modem, or emulate a smart terminal? This is the poop you need, well laid out and easy to read. It is designed for assembly language programmers, but there is lots and lots the BASIC programmer can use as well.

Third, our ranks of top-flight columnists will grow. This month's U2 and Casette Corner are examples. There will be more. Let us know what YOU would like to see in a regular column, and we will scour the nation to find it!

And last but not least, Ads. As there are more of you, PEEK(65) will become increasingly THE place to advertise anything you want to sell to OSI users. This means you will see the products and programs you need advertised right here in your journal.

There is only one possible fly in the ointment. You. Who me? Yep, you. Have you renewed your subscription? No, you don't need to wait for it to expire. If you renew early, we will add another 12 months to your term, whenever it expires. And have you talked to your dealer, your friends and your co-members of the computer club about PEEK(65)? Hey, really sorry to bug you about it, but we do need your help to make our joint venture here more worthwhile for all of us!

U2

by Greg Stevenson
TELOS CONCEPTS
8002 Poisettia
Buena Park, CA 90620

One often hears of computers having "bugs", but how about mice? Across the street from a C3-B I work on, they are building a new shopping mall. So far, a momentary power outage has demagnetized various spots on the hard disk, making a re-initialization necessary -- and mice have invaded the building, leaving "tell-tail" traces that they have also frolicked inside the C3. This has led to numerous problems, including the reason for no U2 last month. I now realize that backing up one's column is just as important as backing up one's files, a practice I will put into effect right away! Now, on to this month's U2.

No operating system, regardless of the source, will meet everyone's needs. In fact, systems will normally fall short in certain circumstances even when adequate for all other situations. Thus, it is an asset if an operating system or language allows itself to be temporarily modified to meet these special occasions. Unlike the BASIC-IN-ROM machines and the BASICs of some of the larger systems, 65U gives a user the ability to customize the OS (the term is used loosely) or BASIC to meet particular needs. This is normally accomplished via the POKE command. This command allows a user to directly replace the standard values or code of 65U with new values or code, thus adapting 65U to the new environment. For example, if one wishes to disable Control "C", POKE 2073,96. To restore Control "C", POKE 2073,76. By simply poking a value or two in the right spots, one can enable or disable a whole host of features in 65U or change a system parameter (i.e., change the line-delete character from an "@" to some less frequently used character). Thus, the POKE command can make 65U much more versatile.

However, there are several drawbacks to using POKE. Perhaps the most dangerous (especially in immediate mode) is that the user could accidentally enter the wrong address or value. Depending on where that POKE "landed",

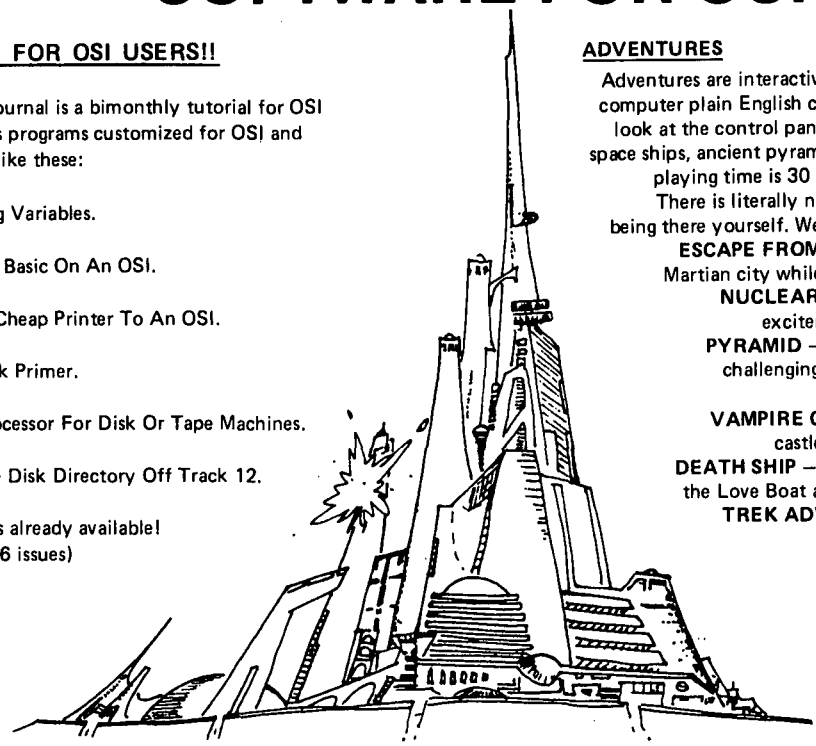
OSI SOFTWARE FOR OSI OSI

A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

1) Using String Variables.
2) High Speed Basic On An OSI.
3) Hooking a Cheap Printer To An OSI.
4) An OSI Disk Primer.
5) A Word Processor For Disk Or Tape Machines.
6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)



ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them – except being there yourself. We have six adventures available.

ESCAPE FROM MARS – Explore an ancient Martian city while you prepare for your escape.
NUCLEAR SUBMARINE – Fast moving excitement at the bottom of the sea.
PYRAMID – Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.
VAMPIRE CASTLE – A day in old Drac's castle. But it's getting dark outside.
DEATH SHIP – It's a cruise ship – but it ain't the Love Boat and survival is far from certain.
TREK ADVENTURE – Takes place on a familiar starship. Almost as good as being there.

$14.95 each

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S – for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line.), Software selectable scroll windows, two instant screen clears (scroll window only and full screen.), software choice of OSI or standard keyboard format, Bell support, 600 Baud cassette support, and a few other features. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48, or 64 characters per line. All that and it sells for a mesly $39.95.

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines.
This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains both an extended machine code monitor and a fix for the string handling bug in OSI Basic!! It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Specify system! $59.95

STRING BUG FIX (replaces basic ROM chip number 3)
All this chip does is to replace the third basic ROM and correct the errors that were put into the ROM mask. $19.95

DATA SHEETS

OS65D LISTING
Commented with source code, 83 pages. $24.95
THE (REAL) FIRST BOOK OF OSI
65 packed pages on how OSI basic works. Our best selling data sheet. $15.95
OSI BASIC IN ROM
Ed Carlson's book of how to program in basic. Now available from Aardvark. $8.95

P.C. BOARDS

MEMORY BOARDS!! – for the C1P. – and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

REAL SOUND FOR THE C1P – and it's cheap! This bare board uses the TI sound chip to give real arcade type sound. The board goes together in a couple of hours with about $20.00 in parts. Bare board, plans, and sample program – $15.95

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves – for fast action. This is our best invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 – Disk - $12.95

TIME TREK (8K) – real time Startrek action. See your torpedoes move across the screen! Real graphics – no more scrolling displays. $9.95

STARFIGHTER – a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

SEAWOLFE – this one looks like it just stepped out of the arcades. It features multiple torpedoes, several target ships, floating mines and real time time-to-go and score displays. – $6.95 in black and white $7.95 in color and sound.

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you are using the standard OSI video display and polled keyboard.

C1P CURSOR CONTROL – A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optional instant screen clear.
$11.95

C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and change lines on any part of the screen. (Basic in ROM systems only.)

FOR DISK SYSTEMS – (65D, polled keyboard and standard video only.)

SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs. The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. – $24.95 on 5" disk - $26.95 on 8".

DISK UTILITIES

SUPER COPY – Single Disk Copier
This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time – up to 12 tracks at a pass. It's almost as fast as dual disk copying. – $15.95

DISK CATALOGER
This utility reads the directory of your disks and makes up an alphabetic list off all your programs and what disks they are on. $14.95

MACHINE CODE RENUMBERER
(C2/4-MF only)
Renumbers all or part of a program at machine code speeds. – $15.95

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

OSI

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110 or (313) 624-6316

OSI

all sorts of things could happen to 65U. Another problem is that the locations changed by this command are often not restored to their original values. (OS-DMS is notorious for leaving Control "C" and "O" disabled). This means that the user must POKE back in the original values and the address and contents are not the easiest to recall (e.g., was it 2676 or 2976 that affects comma input?). So POKEs have their negative points.

Does 65U have another method of enabling and disabling a feature like one can do with POKE, but which, unlike POKE, (1) prevents accidental and uncontrolled changes to other parts of 65U, (2) makes it much easier to remember "what does what", and (3) is much easier to use and consumes much less space in programs? The answer is, yes! The following little experiment will reveal it:

1) Load in any simple program and run it.
2) POKE 15639,00
3) Run the program again.
4) POKE 15639,32
5) Run the program again.

One will notice that the program runs normally in Step 1. But after the POKE, the program runs just as if one had typed a FLAG 7, which enables the statement trace feature. After the POKE in Step 4, the program runs normally again (FLAG 8). The FLAG command, which seems to alter 65U so mysteriously, is nothing more than an automatic POKE, but it is a lot more convenient than POKE. With a POKE one must key in up to 8 digits, correctly delimited by a comma. The current FLAGs have no more than two digits with a maximum of 3 allowed. Thus, typos are greatly reduced and the FLAG values are much easier to remember than POKE values. Also, the FLAG value to disable a feature is usually just one more than the value enabling a special feature. Thus, FLAG can be a far superior method for one to customize 65U to individual requirements, provided that new FLAGs can be added. Fortunately, adding new FLAGs is very simple, and an understanding of how the FLAG command works will aid one in using this command more effectively.

All FLAGs are stored as a series of 4-byte entries in a FLAG table. This table (see table 1) is currently broken into four sections and each section is located in a different part of 65U. The first byte of each entry is the FLAG number. The next byte is the value to be poked, and the last two bytes are the low and high bytes, respectively, of the address of the memory location to be poked. Whenever a FLAG command is encountered, the FLAG routine takes the FLAG number specified, and starting at the beginning of the table, $49E7 (18919), looks at the first entry's FLAG designation and checks it against the designated FLAG number. If they do not match, it skips over this entry and checks the following entry. If there is a match, the FLAG routine takes the FLAG entry's value and "POKEs" it in the address specified by bytes 3 and 4. It then moves on to the next entry and checks it. Each and every entry is checked by the FLAG routine, and it is either processed or ignored, depending on whether or not a match is found. This means that if more than one location needs to be altered for a given feature, one simply places in one entry for each location that needs to be altered. The entry designations do not need to be in any order, nor do entries for a given designation need to be grouped. This means that a feature can be added to an existing FLAG by simply adding entries with the same FLAG designations to the end of the table. There is no need to touch other entries. If a non-existent FLAG number is accidentally specified, the FLAG routine will simply search the whole table (as it always does), but will alter no locations.

There are two types of entries with special FLAG designations that guide the FLAG routine as it searches the table. A designation of $FF(255) tells 65U that this is the last entry in the table and to return to execute the next BASIC command. A designation of $FE(254) tells 65U that it has reached the end of a section of the table. The FLAG routine then bypasses the value byte and loads the "table pointer" with the address specified by the last two bytes. This address is where the first entry in the next section is located. FLAG designations for normal FLAGs can be any value from $00(00) through $7F(127). FLAG values from $80(128) to $FD(253) will be ignored.

One can add one's own FLAG entries to 65U simply by changing the last entry of the fourth section from $FF to $FE and placing the high and low bytes of the location of the "fifth section" of the table into the last two bytes of this entry. Now, place the new FLAG entries in this fifth section, then add an entry having the FLAG designation of $FF(255).

A big question is where to place this and other new sections to the FLAG table. Are there "holes" in 65U that one can use to place tables or routines? For the most part, the answer to this latter question is no. 65U is packed full. The fact that the FLAG table had to be broken down and placed into different sections attests to this. The few remaining locations are used by EDITOR or RSEQ either for code or workspace when they are enabled. The remaining holes are too small for anything other than a couple of bytes of information.

In the light of this, one has three options. One can overlay a part of 65U with the table disabling another function (which EDITOR does on top of using up the holes). However, this is tricky, especially since 65U is full of patches, and this should be done only if you know what every byte of the area to be overlaid does.

A second option is available for those who have a memory at $D000 for running LEVEL 3 or CP/M. This area is totally unused by LEVEL 1, and even under LEVEL 3 there is a large hole from $D340(54080) to $D7FF(55295) that can be used for tables or routines. If the FLAGs can be common to all users, this is an ideal spot. (NOTE: Much of this area is used in systems running NET.) This is the location used by this author.

The last alternative and the most practical for most users of 65U is to lower the top of the BASIC workspace by a page (256 bytes) or two and create a "hole." To do this, simply POKE 133,X where X is 192 minus the number of pages needed. Thus, POKE 133,191 would create a hole of 256 bytes, enough for a FLAG section with 64 entries.

As an example of how to add

FLAGs, the following routine (see listing 1) could be added to BEXEC* giving these additional FLAGs:

31) Allow commas & colons on input
32) Restore commas & colons as delimiters
33) Suppress auto CR and LF
34) Restore auto CR and LF
35) Disable Control "C" and "O"
36) Enable Control "C" and "O"

That's all for this month. All users of 65U are encouraged to write in with problems, comments, and suggestions, especially those working under LEVEL 3 (local and net).

LISTING #1. ADDITION TO BEXEC* TO ADD FLAGS

```
1000 POKE133,191
1010 X=191*256
1020 READ FD,VL,AD
1030 AH=INT(AD/256):AL=AD-(AH*256)
1040 POKE X,FD:POKEX+1,VL:POKEX+2,AL:POKEX+3,AH
1050 IF FD=255 THEN 1090
1060 X=X+4:IF FD<128 THEN 1020
1070 IF FD=254 THEN X=AD:GOTO1020
1080 PRINT "INVALID FLAG DESIGNATOR IN DATA":END
1090 X=8487:POKE X,254:POKEX+1,00:POKEX+2,00:POKEX+3,191
1099 END
1100 DATA 31,13,2972,31,13,2976
1110 DATA 32,58,2972,32,44,2976
1120 DATA 33,00,2676,33,00,2683
1130 DATA 34,13,2676,34,10,2683
1140 DATA 35,96,2073,35,000,14639
1150 DATA 36,76,2073,36,255,14639
1160 DATA 255,00,00,00
```

	FLAG		VALUE		LOW ADD		HGH ADD	
SECTION	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
1	13	19	1C	28	38	56	36	54
$49E7	14	20	04	04	38	56	36	54
(18919)	01	01	AD	173	E1	225	4C	76
	02	02	20	32	E1	225	4C	76
	13	19	F0	240	5D	93	32	50
	13	19	4D	77	5E	94	32	50
	14	20	86	134	5D	93	32	50
	14	20	25	37	5E	94	32	50
	03	03	02	02	90	144	2D	45
	FE	254	00	00	9D	157	3F	63
2	03	03	02	02	91	145	2D	45
$3F9D	04	04	01	01	90	144	2D	45
(16285	04	04	01	01	91	145	2D	45
	15	21	1C	28	6A	106	24	36
	15	21	0B	11	6B	107	24	36
	16	22	7B	123	6A	106	24	36
	16	22	25	37	6B	107	24	36
	FE	254	00	00	80	128	21	33
3	05	05	8D	141	40	64	4A	74
$2180	05	05	3C	60	41	65	4A	74
(8576)	06	06	90	144	40	64	4A	74
	07	07	00	00	17	23	3D	61
	08	08	20	32	17	23	3D	61
	09	09	03	03	7E	126	3E	62
	0A	10	00	00	7E	126	3E	62
	0B	11	FF	255	3E	63	38	56
	0C	12	00	00	3F	63	38	56
	FE	254	00	00	0F	15	21	33
4	0D	13	68	104	30	48	12	18
$210F	0D	13	68	104	31	49	12	18
(8463)	0D	13	60	96	32	50	12	18
	DE	14	A2	162	30	48	12	18
	0E	14	02	02	31	49	12	18
	0E	14	4C	75	32	50	12	18
END	FF	255	00	00	00	00	04	00

SOMETHING FOR NOTHING

We, the willing,
Led by the uncaring,
Have been doing so much,
For so long,
With so little,
That we are now qualified
To do anything,
With NOTHING.

For a long time, this common lament seemed to apply to us OSI owners. Especially when trying to get factory response to a problem! Fortunately, PEEK (65) and magazines such as MICRO and COMPUTE (in its various incarnations) have helped to fill the information gap.

Recently, I was reminded of an experience I had early in 1979 when my C2-4P was a new and almost undocumented mystery.

The program was a simple one. However, one line not only would not execute, but seemed to retype itself after being entered!

A line something like this:

```
150 IF X=WT AND Z=L THEN Y=0
```

when LISTed, came out like this:

```
150 IF X=WTAND Z=L THEN Y=0
```

Which, when I figured it out, made me glad I had used spaces in entering the code. Otherwise, I would not have had that hint that BASIC was combining the T from the variable WT with the AN from AND, to form TAN (TANgent function).

Later, I encountered a similar problem in using long, descriptive variable names. Variable names like RATE, PRINCIPAL, and PAYMENT make a program much easier to follow. But INTEREST, MONTHLY, and COST are disasters looking for a place to happen. Needing a quick way to tell whether a word could be used as a variable name, I hit on the scheme of typing them in as a dummy line of code, with each letter followed by a space. The line is then LISTed, and if any of the spaces have disappeared, the reason will be pretty obvious. For example:

```
OK
10 R A T E
20 P R I N C I P A L
30 P A Y M E N T
40 I N T E R E S T
50 M O N T H L Y
60 C O S T

LIST
10 R A T E
20 P R I N C I P A L
30 P A Y M E N T
40 INT E R E S T
50 M ON T H L Y
60 COS T
```

(Cont'd p. 16)

CASSETTE CORNER
by David A. Jones

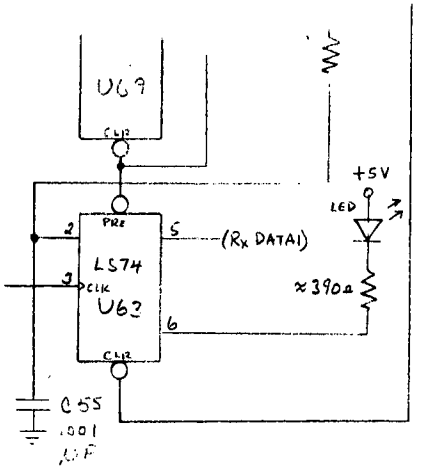
Perusing my last article it appears that I could have worded the statement about reinitializing the input vector a little more clearly. This vector will have to be reset anytime the "BREAK" key is depressed, not only after a cold or warm start. Therefore, if you intend to use it with the Extended Monitor or Assembler/Editor $0218 must be set to $22 and $0219 must be $02.

Also the remarks for the rows and columns are incorrect. The corrected listing follows. Note that there are no changes to the code, only the remarks.

A quick and economical way to increase the speed of the SII/ClP is to purchase a 74LS92 and install it in U29. Then make a small cut in the foil supplying the clock to the CPU (00 U8 pin 37) and run a new wire from U29 pin 9 (W9) to pin 37.

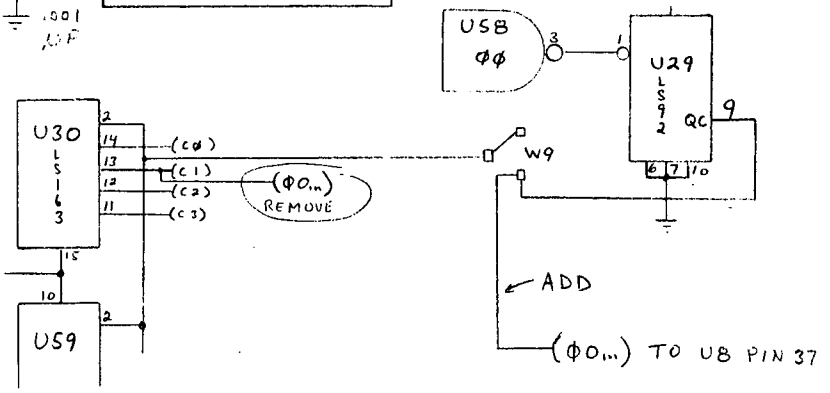
This new clock frequency is 1.31072 MHz which is 33% higher than the original .98304 MHz so of course the system now runs this much faster. Although the 6502 is not guaranteed to work past 1 MHz, the 6502A is only a 6502 which has been tested and certified to operate at 2 MHz over the full temperature range. This means there are many 6502's in use that are capable of operating faster than 1 MHz but not at 2 MHz. This principle applies to the 2114's as well. The standard 2114's are rated at 450 nanosecond access time and we now have a period of only 1/2 1/1.33 MHz or 375 nanoseconds. Slightly too fast, but should not cause any problems except in extreme temperature environments.

Another dollar modification is to install a LED and resistor between pin 6 of U63 AND +5 Volts. The actual value of the resistor is not critical and may be anywhere between 300 and 400 ohms. The lower the value the brighter the LED but try to stick to the upper side so as not to exceed the 8 ma output capability of the 74LS74.



Now, any activity on the cassette port will cause the LED to flicker. This is useful for positioning tapes prior to loading. Especially so, when two or more programs or data files are to be resident in memory at the same time. Noise bursts caused by positioning the tape for the second load can delete or overwrite a line in the first file just as surely as if you had intentionally entered the line numbers.

Position the tape by watching the LED and when the desired location is found type LOAD and start the tape moving. Wait until the LED remains on steadily indicating the 5 to 10 second leader preceding data and then hit return for a nice clean load.




```
 10 0000                ;OUTPUT CONTROL
 20 0000                ;BY DAVID A. JONES
 30 0000                KBRD=$DF00
 40 0000                OUTPUT=$FF69
 50 0222                *=$0222
 60 0222 48             PHA
 70 0223 A9F6           LDA #$F6        SELECT ROWS 1 $ 3
 80 0225 8D00DF         STA KBRD
 90 0228 A9C0           LDA #$C0        SELECT COLUMNS 7 & 6
100 022A 2C00DF         BIT KBRD        STOP? (CONTROL S)
110 022D D00C           BNE CONT        IF NOT, CONTINUE
120 022F A9FC    WAIT   LDA #$FC        ROWS 0 & 1
130 0231 8D00DF         STA KBRD
140 0234 A9C0           LDA #$C0        COLUMN 7 & 6
150 0236 2C00DF         BIT KBRD        RESUME? (CONTROL Q)
160 0239 D0F4           BNE WAIT        IF NOT, KEEP LOOPING
170 023B 68      CONT   PLA
180 023C 4C69FF         JMP OUTPUT      CONTINUE WITH OUTPUT
```

CALL FOR OSI BASIC

Phil Hooper
Box 62
Northfield, VT 05663

Anyone who interfaces OSI BASIC with machine language routines is surely envious of the CALL command provided by APPLE, with which a program can summon a machine language routine from a specified address. The program OSICALL of Listing 1 provides a similar feature for OSI users (with ROM BASIC). It is meant to be run once, to install the code either from Listing 2 or from Listing 3 as a machine language routine embedded in a string located in Line 1. It also sets the USR pointer so it points to the start of the routine just installed. Subsequently, a routine located at, say, $2D30 may be invoked from BASIC using only:

```
<line #> X=USR(X),2D30
```

This command follows the USR pointer to the code installed by OSICALL, which then looks through the rest of the BASIC line, after the comma, decoding ASCII to HEX and storing it in two idle locations, (0013,14), of the line buffer. Finally, control jumps to the address stored at these locations. To eliminate null (00) bytes from the machine code, as they would interfere with loading, the mechanism for transfering the contents of $0013,14 to the program counter uses the stack, rather than the simpler indirect jump, JMP($13), or 6C1300.

The DATA of listing 2 implement the hex version of OSICALL, while those of Listing 3 provide the capability of using decimal address in-line. In fact, the decimal version will accept a BASIC expression for the subroutine address. Though this would be of limited use, it does permit one to save the starting addresses of machine language subroutines in a table, say T, so that the BASIC line:

```
<line #> X=USR(X),T(N)
```

would select subroutine number N from the table and hop to it.

The machine code from both listings is completely relocatable. If you prefer not to embed it in line 1, merely change the POKEs of line 3 and the initial counter value of Line 10 to correspond to your preferred starting address. In this case, Line 1 would be superfluous.

As a simple test you can insert the following BASIC line into your BASIC program (after OSICALL has installed its routine). This calls the keyboard polling routine and will suspend execution until some key is pressed.

```
<line #> X=USR(X),FD00
```

If you wish to be able to put several subroutine addresses in a single line - ►

LISTING #1: to be used with data from either listing #2 or listing #3.

```
 1 ZZ$="23 CHARACTERS"set up dummy string
                       variable to hold
                       object code
 3 POKE11,10:POKE12,3 set USR pointer
                       to $030A, the lst
                       character in the
                       above string
 5 READ N              23 for hex vers;
                       14 for dec vers.
10 FORI=778TO777+N     dec 778 = 30A
15 READ K:POKE I,K     install another
                       byte of obj code
20 NEXT I              do entire DATA
                       list.
```

LISTING #2

```
1000 DATA 23           . . .      . . . .   size of remaining DATA list
1001 DATA 162,23       A217       LDX $17   load X with offset from $FC to $13
1002 DATA 3,23,166     2017A6  NXT JSR $A617 get next character from BASIC line
1003 DATA 32,147,254   2093FE     JSR $FE93 convert to hex, set N if no good
1004 DATA 48,5         3005       BMI OUT   if not hex, we are done - exit
1005 DATA 32,218,254   20DAFE     JSR $FEDA roll hex nibble into $13,14
1006 DATA 240,243      F0F3       BEQ NXT   this branch always taken
1007 DATA 156,20       A514   OUT LDA $14   get hi byte of address
1008 DATA 72           48         PHA       put it on stack
1009 DATA 165,19       A513       LDA $13   get low byte of address
1010 DATA 72           48         PHA       put it on stack
1011 DATA 8            08         PHP       push status (trick to avoid
                                             adjustment to address with
                                             RTS)
1012 DATA 64           40         RTI       pull address into program
                                             counter & go to it.
```

LISTING #3

```
1000 DATA 14           . . . .    . . . .   size of remaining DATA list
1001 DATA 32,173,170   20ADAA     JSR $AAAD evaluate expression from BASIC line
1002 DATA 32,5,174     2005AE     JSR $AE05 put result into $AE,AF
1003 DATA 165,175      A5AF       LDA $AF   get high byte of result
1004 DATA 72           48         PHA       put it on the stack
1005 DATA 165,174      A5AE       LDA $AE   get low byte of result
1006 DATA 72           48         PHA       stack it
1007 DATA 8            08         PHP       as above
1008 DATA 64           40         RTI       follow the pointer to subroutine
```

<line #> X=USR(X), FD00, 2C18, A8C6, FE43, FEB0

- have your subroutines, except the last one, end with JMP $030A instead of RTS, and be certain that the final address in your subroutine list is to a routine that DOES terminate with RTS.

See if you can convince yourself of what will happen with the following:

<line #> X=USR(X), 030A, FD00

Notes: Line 1 must be entered exactly as above, with NO EXTRA SPACES. The actual string value may be anything, as long as it is at least 23 characters long.

When this program is run, with an appropriate DATA list, a machine language program is installed in line 1 of the BASIC program, between the quotation marks which delimit the value of the string. Since there are no null (00) bytes in this code, it may be saved and later loaded without trouble. However, it lists rather oddly. After the program has been run, there is no longer any need for the DATA lines, or for lines 5 through 20, so they may be deleted. The remaining two lines (LINE 1 with the machine language in it and LINE 3 setting the USR pointer to the address of the machine code in LINE 1) may be saved and brought in whenever one wishes to write a program using OSICALL.

Have fun!!

HOW TO EDIT PROGRAMS AND KEEP VARIABLES

By Jim Williams
Calumet City, IL

The variables used in a BASIC program (or pointers to them, for strings) are stored in two blocks of memory immediately after the program. Pointers to those blocks are updated (and one of the blocks is moved) as the program encounters more and more variables. When you either RUN or edit a program, the pointers are reset, though what was stored in memory to represent your variables was not changed. Entering a program with a GOTO n command lets you keep the values that were used previously by not resetting the pointers. But even the most modest editing -- replacing a line with exactly the same thing -- does reset them.

Since the variable data is not destroyed, simply restoring the pointers after editing will let you keep your data. Adding to the program, of course, will make the program extend into the first block of variables, destroying some of them. But shortening the program is just fine. If you must add a line or lines (and want to keep your old data), be sure to delete more than you add, and to do it before you add the lines! (If you really have to count bytes, a line of BASIC takes 5 bytes plus one byte for each keyword (FOR, INPUT, etc.) plus one byte for each other ASCII character in the line.)

To save your data, first use the monitor to find the contents of the pointers in loctions $7B to $82, and write them down. Then do your editing, first deleting, then adding. Use the monitor to restore the pointers to their old values, and be sure to reenter the program with a GOTO!

The machine is in a nonstandard and unstable state at this point. Since its pointers are not pointing where it expects them to point, if further editing is really needed, try recording the values of the pointers, and then point $7B,7C and $7D,7E both to where variable space would normally begin, at the address two higher than the contents of $AA,AB immediately after a LIST. Then try your editing. Restore all pointers to the recorded values. Good luck!

The scheme is generally useful only after you have a lot of text typed in, or after a long number- crunching run that you forgot to put an output routine in!

OSI Sold!!

Ohio Scientific, Inc. and M/A COM, Inc. have signed a letter of intent, under the terms of which M/A COM will purchase OSI, which will become one of M/A COM's family of subsidiary companies. M/A COM is in the business of manufacturing microwave and other communications equipment and systems, including satellite communications systems.

Under the terms of the agreement, the present management of OSI will be largely retained, but will be supplemented by a number of experienced M/A COM people. All OSI products will continue to be manufactured and supported.

The move is a natural one for both companies. OSI has suffered from too-rapid growth and inexperienced management, with most of the executive team having little management experience beyond what has been gained in operating OSI. Also, the additional cash provided by a 300 million dollar company will be welcome as OSI continues to expand. For M/A COM as well, the move is promising. Increasingly, as we move through the '80s, computers will be communication-dependent, and communications equipment will be computer-dependent. Tying the most progressive small computer company with a very progressive and powerful communications firm makes splendid sense.

What does this move mean to the OSI user? We stand to benefit more than anyone. The rather well-founded rumor is that 10 people will be added to the Dealer Support Center, and 6 full-time programmers will be added. These two moves would answer the two most often-heard complaints about OSI, that their software is slow to appear and buggy when delivered, and that the factory is reluctant to communicate with users and dealers. Actually, neither of these complaints is as valid as they perhaps were a couple of years ago, but further improvement will certainly be welcome!

★

LETTERS

ED:

One of the major deficiencies of OSI computers is the lack of high resolution graphics. At the local computer club I keep seeing beautiful graphics on the APPLE. I have been getting the raspberry from APPLE owners that my system can't do graphics. In fact, several former OSI owners have been converted to APPLE by the HIRES feature.

Two independent vendors, GRAFIX and Mittendorf Eng., are now selling add-on graphics boards for the 1P. I would like to see reviews in PEEK from anyone who is using either of these. Since I own an 8P, neither of these new boards would plug directly into my system.

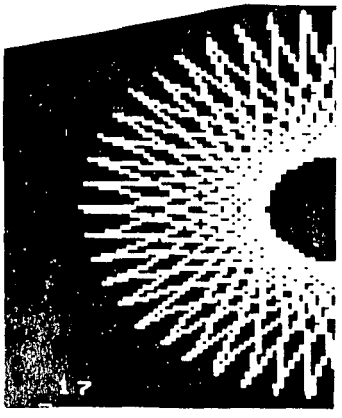
Recently I have upgraded my system from the old 440 video board to the current 540 model. The 440 was put away on a shelf as I am sure many other 440 boards are. However, I have found a good use for the old board; I converted it to 256 by 256 high resolution graphics. The total cost is 8K of 2114 memory chips and $16 for a circuit board to hold the new chips. The memory is usable for program storage when graphics are not used. The modification requires extensive foil cutting and rewiring. The alphabetic display circuitry is removed and the board decoding expanded to 8K. Since the 440 board was now worthless to me in its original state, I figured I had nothing to lose in trying to modify it for this purpose.

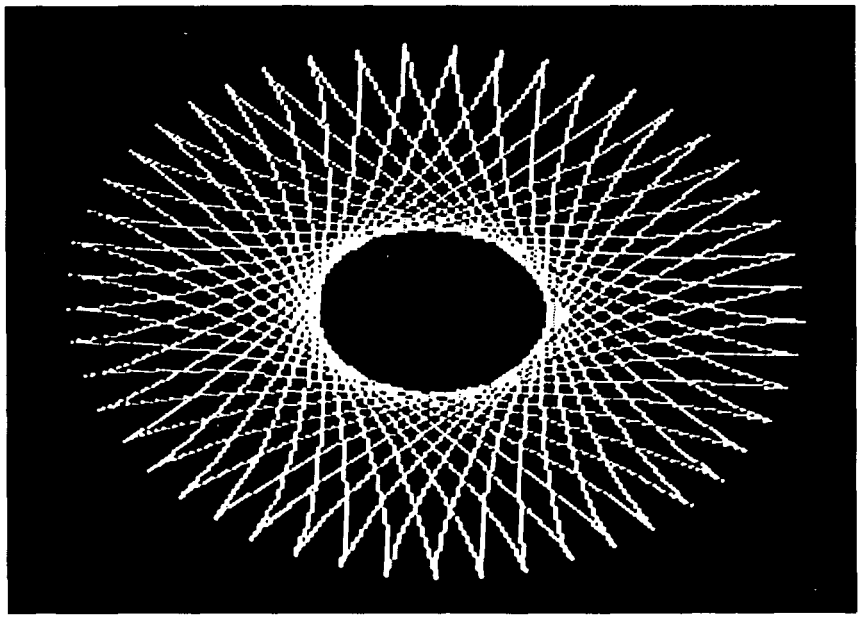
Below is an example of the type of plotting you can do with individual control of 65,536 pixels. If you have an old 440 and are interested in modifying it, please send me one dollar for more information.

PEEK #4 contains an earlier plot with 128 by 128 resolution. If you compare the photos you can see the effect of increased resolution.

Earl Morris
3200 Washington
Midland, MI 48640

* * * * *





ED:

I am a student at Purdue University studying for a BSET degree. As a senior design project is required and I own a C4P, I developed an idea using the OSI in my project. Additional memory above the standard 8K is required and the addition of a 527 board will solve that problem, if money is no object. On my limited project budget, a populated board is a luxury I can't afford. So I decided to add the memory by building up the bare board. Much to my chagrin, OSI has discontinued selling their bare boards which I feel is a direct "stab in the back" to those of us who believed in their commitment to the experimenter and hobbyist.

Question---Is there an alternative source for the 527 and other OSI bare boards? If not, maybe some PC board manufacturer might consider adding the OSI boards to their product line. I know of several that could be sold within the N.E. Indiana area alone.

I have written to OSI about my dilemma, but based on their past willingness to communicate directly with the outside world, I won't hold my breath until an answer arrives.

I hope you or a reader can help me on this problem. Meanwhile I have 48 2114's and no place to stick them.

I also would welcome assistance from anyone who has modified the C4P for a printer output.

Jerry Ryan
Ft. Wayne, IN.

ED:

One problem with Ohio Scientific Systems (most notably C-2) has been the inability to utilize the 6502 IRQ and NMI commands from a BASIC program, via USR routines. The problems originates from the fact that the reset vectors for these commands, contained in the system ROM, point to an area of memory that is heavily used by BASIC (i.e. 01XX). Thus, it is impossible to field either of these interrupts because BASIC rapidly destroys any service routine.

Continued on page 12

ED:

I can answer a question from Dru Cabler in Volume I, No. 9. The routine to open a space in memory is located at $A1CF-A211 for the C1P.

To help answer other questions, attached is a partial list of routines in the Basic ROM (Version 1.0 Rev. 3.2). A list like this may be available already (I don't know).

The two issues I've received have several letters about difficulties with the garbage collection routine. There is a company that is selling a set of ROM's (Basic 3, Basic 4, and Monitor) in which the garbage collection routine has been fixed. In addition, they have added an editing cursor, insert and delete, and some other goodies. They are:

Progressive Computing
3336 Avondale Crt.
Windsor, Ont.
N9E 1X6 Canada

Peter Smith
Lakewood, CO

Peter:

Also, Aardvark now offers new ROM's in several varieties. It gets better and better!

AL

MICROSOFT BASIC VER. 1.0 REV. 3.2 IN OSI C1P AND SUPERBOARD II

Location of Routines

A000-A037 Initial Work Jump Table
A038-A065 Routine Entry Addresses

A084-A163 Keywords in ASCII (high bit = delimiter)
A164-A186 Error Messages

A1A1-A1CE Search Stack For FOR or GOSUB Activity
A1CF-A211 Open Space in Memory
A212-A21E Test: Stack Too Deep
A21F-A24B Check Available Memory
A24C-A273 Send Canned Error Message, Then:
A274-A294 Wait for Command or Warm Start
A295-A31B Handle New Line from Keyboard
A31C-A34D Rebuild Chaining of Basic Lines in Memory
A34E-A3A5 Receive Line from Keyboard
A3A6-A431 Change Keywords to Basic Tokens
A432-A460 Search Basic for a Specific Basic Line Number
A461-A479 Do NEW
A47A-A4A6 Do CLEAR
A4A7- Reset Basic Execution to Start of Basic Program

A556-A5FE Do FOR
A5FF-A619 Execute Basic Statement
A61A-A628 Do RESTORE

A638-A660 Do STOP or END
A661-A67A Do CONT

A691-A69B Do RUN
A69C-A6B8 Do GOSUB
A6B9-A6E5 Do GOTO
A6E6-A70B Do RETURN
A70C-A719 Do DATA
A71A-A71E Scan for Next Basic Statement
A71F-A73B Scan for Next Basic Line
A73C-A74E Do IF
A74F-A75E Do REM
A75F-A77E Do ON
A77F-A7B8 Get Fixed-point Number from Basic Line
A7B9-A828 Do LET

A829-A8C2 Do PRINT
A8C3-A8DF Print String from Memory
A8E0-A903 Print Single Format Character
A904-A922 Handle Bad Input Data
A923-A945 Do INPUT
A946-A94E Prompt and Receive Input
A94F-AA1B Do READ
AA1C-AA3F Messages: "EXTRA IGNORED, REDO FROM START"
AA40-AA9A DO NEXT
AA9B-AAC0 Check Data Type, Print "TYPE MISMATCH"
AAC1-ABF4 Input and Evaluate Expression (string or numeric)
ABF5-ABFA Evaluate Expression within Parentheses()
ABFB-ABFF Check Right Parenthesis)
AC00-AC02 Check Left Parenthesis (
AC03-AC0B Check for Comma
AC0C-AC10 Print "SN"
AC11-AC17 Set up Function for Future Evaluation
AC18-AC26 Set up Variable Name
AC27-AC65 Identify and Set up Function References
AC66-AC6A Do OR
AC6B-AC95 Do AND
AC96-AD00 Do Comparisons, String or Numeric
AD01-AD0A Do DIM
AD0B-AD80 Search for Variable Location in Memory
AD81-AD8A Check if ASCII Character is Alphabetic
AD8B-ADE5 Create New Basic Variable
ADE6-ADF6 Array Pointer Subroutine
ADF7-ADFA 32768 in Floating Binary
ADFB-AE16 Evaluate Expression for Positive Integer
AE17-AF7B Find or Create Array
AF7C-AFAC Compute Array Subscript Size
AFAD-AFC0 Do FRE
AFC1-AFCD Convert Fixed-point to Floating-point
AFCE-AFD3 Do POS
AFD4-AFDD Check if Direct Command, Print "ILLEGAL DIRECT"
AFDE-B00A Do DEF
B00B-B020 Check FNx Syntax
B021-B08B Evaluate FNx
B08C-B0AD Calculate String Vector
B0AE-B114 Scan and Set up String
B115-B146 Subroutine to Build String Vector
B147-B1D3 Garbage Collection Subroutine
B1D4-B217 Check for Most Eligible String for Collection
B218-B24C Collect a String
B24D-B289 Do String Concatenation
B28A-B2B2 Build a String into Memory
B2B3-B2EA Discard Unwanted String
B2EB-B2FB Clean the Descriptor Stack
B2FC-B30F Do CHR$
B310-B33B Do LEFT$
B33C-B346 Do RIGHT$

OS-DMX

OS-DMS EXTENSIONS

digital technology
inc.
P.O. BOX 178590
SAN DIEGO, CA 92117

an advertising supplement to PEEK(65)

What happens when the world's first--and most powerful--microcomputer database management system is superimposed on Ohio Scientific's OS-DMS? The result is an incredibly powerful, eminently usable combination: OS-DMX.

OS-DMX (OS-DMS Extended) is certain to become the new standard in microcomputer database management systems. OS-DMX is, basically, Digital Technology's Microsystems Information Management Program (MIMP) with a new and more powerful DMS-compatible format, extended command language, and a "master control" function, DMX's Program Sequence Executive, which allows the operator to specify a series of programs for the computer to perform automatically.

OSI users now have the best of both worlds. OS-DMX means synergy, standardization, and support. The synergistic effect of merging the best features of the industry's two leading database management systems results in spectacular performance. And standardization will provide a continually expanding list of business application programs for OSI owners. But perhaps most important of all is support.

Digital Technology, Inc., is the largest independent supplier of OSI software with hundreds of business packages in use around the world. Digital Technology software is sold by a growing number of conscientious OSI dealers and OEMs. Every package is backed by the finest support program in the microcomputer industry. All "bugs" are fixed free of charge. Updates (fixes to bugs, minor enhancements, new product announcements) are provided to all dealers and licensed users free of charge. And upgrades to new versions are encouraged (at nominal charge).

Digital Technology's software is user-oriented. In fact, no one else provides such expansive features as on-line documentation, idiot-proof prompting, and operator's manuals that are comprehensive, detailed, and accurate. In addition to the on-line documentation and the reference manual, OS-DMX also features Digital Technology's innovative HELP commands. Problems? Just type HELP. General and technical information is at your fingertips.

What does OS-DMX do? OS-DMX starts where Ohio Scientific's OS-DMS leaves off. The OS-DMX package is designed to replace OS-DMS nucleus, query, and sort modules; in fact, OS-DMX can perform all functions provided by the various OS-DMS modules--and a great deal more. Separate programs or "modules" will be unnecessary in all but a few highly interactive specialized applications. OS-DMX adds a new dimension of flexibility and usability to any DMS-compatible programs.

OS-DMX is comprised of three primary modules and a number of auxiliary programs designed to optimize user interaction with the database system. Information is entered using the Database Input program. Updating database file contents is performed with the Database Edit program. And the most powerful report writer available to microcomputer owners is OS-DMX Database Report. In each module the user controls operation through a powerful command language, telling the computer, in "English", exactly what is to be done.

The Database Create program creates DMS files automatically, allowing the operator to optionally use a model database and add and delete fields as necessary. A database structure mapper is also provided for operator reference. Two database sort programs are provided for in-memory sorts or disk file sorts. A database compress operation allows the operator to remove de-activated records. All program are based on the "command language" concept which allows the user to define program operation in English-like command lines.

OS-DMX allows the operator to create DMX "modules" which may be stored in control files for reuse when needed. All operations are performed by commands which can readily be arranged to perform specific functions. These functions can be repeated as often as necessary without the bother of repeatedly entering the commands. Using the STORE command, the user-defined procedures may be recorded in control files for repeated operation using the RUN command.

Perhaps most innovative of all DMX features is the Program Sequence Executive, a form of job control language which allows the operator to designate a series of operations and BASIC programs which are to be performed automatically by the computer. Executive control files may be used to store the program executive commands. In this way, a program executive control file can supervise the operation of a series of BASIC programs; if DMX programs are included, these programs themselves can be pre-defined using the DMX command language. Control files may contain remarks, comments, and instructions for operator hand-holding. In short, the operator can create a limitless number of "modules" and can instruct the computer to "run itself". (With foresight, the operator can even instruct the computer to perform alternate operations in case of program failures!)

Documentation is on-line at all times in the form of extensive user prompts, an (abbreviated) operator's manual, and HELP commands which describe in detail the DMX command language. In addition, a comprehensive operator's manual, with examples, is included with every OS-DMX system.

All Digital Technology software systems allow the operator to "set" the programs to the type of video terminal and printer used. The operator selects the terminal and printer types from the list provided in the "TERMINAL & PRINTER OPTIONS" program. Screen formatting and printer control are provided automatically yet may be redefined through user subroutines.

Nearly all popular video terminals, including those listed below, are supported and are user-selectable from the Terminal Options menu (drivers for most other terminals available at nominal charge):

NULL (NO SCREEN FORMATTING)
MICRO-TERM MIME
MICRO-TERM ACT-IV
MICRO-TERM ACT-V & ACT-Va
LEAR SIEGLER ADM-3A
BEEHIVE 150/152
BEEHIVE 160/162
HEATH WH-9
HAZELTINE 1420
HAZELTINE 1500
INFOTON I-100
INTERTEC INTERTUBE
ADDS REGENT
SOROQ IQ-120
TELEVIDEO TVI-912/920

Currently supported output devices are user-selectable from the Printer Options menu:

PORT 3	CA-6	SERIAL (RS-232)	OSI 430 OR EQUIVALENT
PORT 5	CA-9	8-BIT PARALLEL	LINE PRINTER INTERFACE
PORT 6	CA-9D	12-BIT PARALLEL	DIABLO & NEC 5500
PORT 8	CA-10	SERIAL (RS-232)	SERIAL PORTS 01-16

A number of additional features are provided by the terminal and printer control program--DEVSET, an exclusive feature of Digital Technology software. DEVSET controls printer operation and paging for all printers and screen formatting through generalized subroutines, so program modification is unnecessary. DEVSET user notes and technical information are provided with every system.

OS-DMX DATABASE INPUT OPERATING COMMANDS

EXCLUDE - specifies fields to be skipped during input operations.

FIELDS - displays for operator reference the characteristics (name and length) of fields in the database.

FIXED - allows the operator to assign fixed field values instead of making repeated entries.

HELP - displays the 'HELP' file for operator reference (includes detailed instructions for all input operations).

INCLUDE - specifies fields to be included during input operations.

INPUT - initiates data-keying operations, prompting the operator for input to all selected fields of each record.

JOURNAL - simultaneously produces a printed record of the database input operation.

OUTPUT - allows the operator to specify line width and output device (system printer or video console) for 'HELP' file listings and database input JOURNAL operations.

QUIT - exits the program; returns to OS-DMX menu.

RESET - resets all selection criteria for key-in specifications.

RUN - initiates an "automatic run" of user-defined input routines recalled as needed from "Control Files".

SCREEN - allows the operator to select programmable format "fill-in the-blank" input routines. (Note: this command operates in conjunction with the OS-DMX Screen Formatter/Editor which will be available April 1980.)

STORE - is used to create user-defined procedures which are stored in "Control Files" for repeated use as needed.

USE - specifies the device and database to be used by the programs.

OS-DMX DATABASE EDIT OPERATING COMMANDS

ADD - is used to add records to the database file, with entries to all fields in the records.

AGAIN - repeats the previous command using the same set of fieldnames and conditions, but a new range of record numbers.

CHANGE - changes individual fields within records according to the range, fieldname(s), and condition(s) specified.

COUNT - counts the occurrence of records meeting the range and condition criteria.

CLEAN - removes all records from the database (confirmation is required).

DELETE - de-activates all records which meet the range and condition criteria.

FIELDS - displays for operator reference the characteristics (name and length) of fields in the database.

HELP - displays the 'HELP' file for operator reference (includes detailed instructions for all input operations).

LIST - displays selected fields, one per line, from records meeting the range and condition criteria.

OUTPUT - allows the operator to display the edit information and 'HELP' files on either the system printer or video terminal.

QLIST - displays a "quick list" of the contents of any field(s) in the database (in horizontal format).

QUIT - exits the program; returns to OS-DMX menu.

RECLAIM - reclaims space used by unneeded records by setting all fields to blank for the specified records.

REPLACE - allows the operator to simultaneously revise the contents of any number of fields in a specified range of records.

RUN - initiates an "automatic run" of user-defined edit routines recalled as needed from "Controal Files".

STORE - is used to create user-defined procedures which are stored in "Control Files" for repeated use as needed.

SUM - provides column (field) totals of selected numeric fields.

USE - specifies the device and database to be used by the programs.

DATABASE EDIT COMMAND MODIFIERS

RANGE - where appropriate, commands may be restricted to specified ranges of record numbers.

FIELDNAME - normally database commands act on all fields in a record, except where field specification is permitted.

CONDITION - where appropriate, command verbs may include relational conditions. Field contents may be compared with either constants or other fields' contents specifying any combination of AND, OR, fn#fn, fn#value, or fn contains "value", where the # symbol represents relationals: =, <=, >=, <, >, <>, ==. The last item, ==, indicates a match if the specified value is contained in the field as a substring.

OS-DMX DATABASE REPORT OPERATING COMMANDS

AVERAGE - prints the average value of totals or subtotals of selected numeric fields.

BREAK - sets up a "control break" operation forcing the report program to separate groups and, optionally, to subtotal selected numeric fields.

COLUMN - defines new columns in the report, created by combining (+, -, *, /) existing database fields.

EXCLUDE - specifies which fields are to be omitted from the report If neither INCLUDE nor EXCLUDE is specified, all fields will be printed in order of occurrence in the database.

FIELDS - displays for operator reference the characteristics (name and length) of fields in the database.

HEADING - allows the operator to change or delete the report heading.

HELP - displays the 'HELP' file for operator reference (includes detailed instructions for all report operations).

INCLUDE - allows the operator to specify which fields will be included in a report and in what order the fields will be printed; also provides horizontal tabs and vertical spaces; allows new column (field) headings. If neither INCLUDE nor EXCLUDE commands are specified, all fields will be printed in order of occurrence in the database.

OUTPUT - allows the operator to specify line width and output device (system printer or video console) for reports.

PRINT - initiates generation of the report in accordance with the designated field specifications, range, and condition(s).

QUIT - exits the program; returns to OS-DMX menu.

RESET - resets all selection criteria for report generation.

RUN - initiates an "automatic run" of user-defined report routines recalled as needed from "Control Files".

STORE - is used to create user-defined procedures which are stored in "Control Files" for repeated use as needed.

TOTAL - specifies which fields are to be totaled (or subtotaled if control breaks are utilized).

USE - specifies device and database to be used by the programs.

DATABASE REPORT COMMAND MODIFIERS

RANGE - where appropriate, commands may be restricted to a specified range of record numbers.

FIELDNAME - normally database commands act on all fields in a record,except where field specification is permitted.

CONDITION - where appropriate, command verbs may include relational conditions. Field contents may be compared with either constants or other fields' contents specifying any combination of AND, OR, fn#fn, fn#value, or fn contains "value", where the # symbol represents relationals: =, <=. >=, <, >, <>, ==. Ths last item, ==, indicates a match if the specified value is contained in the field as a substring.

The OS-DMX DEMO diskette (provided with every order) demonstrates introductory DMX operation.

A MIMP-to-DMX conversion routine is provided without charge to all dealers and licensed users of MIMP, the predecessor to OS-DMX, as part of Digital Technology's on-going support program. Digital Technology software is never obsolete.

A number of optional auxiliary programs (available April 1981) are designed with specific needs in mind. DMX-MAIL will allow conditional mailing label generation. DMX-STAT will supercede the DMS statistical function operations. DMX-COPY will provide the operator with the ability to modify the structure of a DMX database after-the-fact; and, furthermore, to conditionally copy the contents of one database file into another.

Several of the programs previously available as MIMP options will be revised for DMX operation. ECR-1-P Electronic Cash Register Polling and Control interfaces with RS-232 programmable cash registers to provide totally automated business operation. Versions are currently available for certain MKD and NCR cash registers; other models of cash registers can be accommodated if interest warrants. SALES-1 sales analysis program provides detailed breakdown of sales by family groups, departments, etc. And INV-1 Restaurant Inventory and Menu Explosion provides detailed analysis for fast food operations. All of these programs work in concert to provide automated input and analysis for the small businessman.

Currently undergoing final field testing is an interactive Electronic Sales Terminal program which allows the use of video terminals as on-line point-of-sale terminals, controlling cash drawers, ticket printers, and interfacing with DMS-compatible inventory files. OS-65U Level III operation permits multiple sales terminals in addition to multi-tasking functions.

These, and a growing number of DMS-compatible business packages from Digital Technology and others will be available through your Ohio Scientific computer dealer.

OS-DMX INTRODUCTORY OFFER	$ 695
UPGRADE from MIMP (any version)	$ 295

Call for the name of the nearest OSI dealer carrying Digital Technology business software.

B347-B36E Do MID$
B36F-B38B Pull String Function Parameters from Stack
B38C-B391 Do LEN
B392-B39A Go from String Mode to Numeric Mode
B39B-B3AA Do ASC
B3AB-B3BC Input Byte Parameter
B3BD-B3FB Do VAL
B3FC-B407 Get 2 Parameters for POKE and WAIT
B408-B41D Convert Floating-point to Fixed-point
B41E-B428 Do PEEK
B429-B431 Do POKE
B432-B44D Do WAIT
B44E-B454 Add 0.5 to Accumulator #1
B455-B46B Do Subtraction
B46C-B536 Do Addition
B537-B563 Complement Accumulator #1
B564-B568 Print "OV"
B569-B59B Subroutine to Multiply a Byte
B59C-B5BC Function Constants
B5BD-B5FD Do LOG
B5FE-B621 Do Multiplication
B622-B64C Subroutine to Multiply a Bit
B64D-B672 Load Accumulator #2 from Memory
B673-B68F Test and Adjust Accumulator #1 and #2
B690-B69D Handle Overflow and Underflow
B69E-B6B4 Multiply by 10
B6B5-B6B8 10 in Floating Binary
B6C2-B6CC Do Divide by
B6CD-B74A Do Divide into
B74B-B76A Load Accumulator #1 from Memory
B76B-B79A Store Accumulator #1 in Memory
B79B-B7AA Load Accumulator #2 into Accumulator #1
B7AB-B7B9 Load Accumulator #1 into Accumulator #2
B7BA-B7C9 Round off Accumulator #1
B7CA-B7D7 Compute SGN Value of Accumulator #1
B7D8-B7F4 Do SGN
B7F5-B7F7 Do ABS
B7F8-B830 Compare Accumulator #1 to Memory
B831-B861 Convert Floating-point to Fixed-point
B862-B886 Do INT
B887-B911 Convert String to Floating-point
B912-B946 Get New ASCII Digit
B947-B952 String Conversion Constants
B953-B95D Print "IN"
B95E-B96D Print Basic Line Number
B96E-BA95 Convert Floating-point to ASCII
BA96-BAAB Constants for Numeric Conversions

BAAC-BAB5 Do SQR
BAB6-BAEE Do Power Function (^)
BAEF-BAF9 Do Negation
BAFA-BB1A Constants for String Evaluation
BB1B-BB6D Do EXP
BB6E-BBB7 Subroutines for Function Series Evaluation
BBB8-BBBF Manipulation Constants for RND
BBC0-BBFB Do RND
BBFC-BC02 Do COS
BC03-BC4B Do SIN
BC4C-BC77 Do TAN
BC78- Constants for Trig
BF2D- Output Character to CRT and Scroll Routines

Note - String Addresses show location of routines but are not necessarily entry address for that routine.

"PRINT AT" HIDES IN BASIC

by Charles Lundberg
1020 Harrison
Olympia, WA 98502

```
 10 M$="1111111111111111":REM DUMMY STRING USED TO STORE USR PROG.
 15 A$="THIS IS A GREAT MACHINE":REM MESSAGE TO BE PRINTED.
 20 DATA160,0,185,0,0,153,0,0,136,240,3,76,11,3,96:REM USR PROG.
 30 FORQ=0TO14:READD:POKE777+Q,D:NEXT:REM LOAD PROGRAM IN M$
 40 POKE778,23:REM STRING LENGTH
 41 POKE780,32:POKE781,3:REM STRING LOCATION IN MEM.LO/HI BYTE
 42 POKE783,3:POKE784,209:REM SCREEN LOCATION FOR PRINT AT
 50 POKE11,9:POKE12,3:REM SET USR POINTERS
 55 FORQ=1TO30:PRINT:NEXT:REM CLEAR SCREEN
 60 P=USR(P):REM INITIATES USR ROUTINE
 70 GOTO70:REM HOLD UNTIL BREAK OR CTRL/C
```

This program loads a "print at" machine language routine into M$. When the program is run, A$ is printed instantly on the screen. Never edit M$. Edit A$ only if you change line 40 to show the correct length of the new string.

THE PRINT AT ROUTINE

```
 0309 A0 00    LDY   STRING LENGTH
 030B B9 00 00 LDA,Y LOAD CHARACTER FROM STRING
 030E 99 00 00 STA,Y STORE CHARACTER AT SCREEN
 0311 88       DEY   COUNT DOWN TO ZERO
 0312 F0 03    BEQ   WHEN COUNT IS ZERO, SKIP NEXT INSTRUCTION
 0314 4C 0B 03 JMP   GO BACK AND GET ANOTHER CHARACTER
 0317 60       RTS   RETURN TO BASIC
```

(Cont'd from p. 9)
We have proposed to OSI that a new ROM be produced, identical to the old one in all respects but for the IRQ and NMI reset vectors. These would be changed to point to a part of memory that is "stable" (e.g. D0XX or E0XX). However, for such a new ROM to be produced, it must be financially feasible to do so. So, we would like to ask all interested OSI users to drop a quick note to the address below expressing interest (the cost to be in the $25-$50 range). If enough replies are received, we may well see a new monitor chip. Thanks so much!

OHIO SCIENTIFIC COMPUTERS
Attn: Customer Relations
1333 South Chillicothe Rd.
Aurora, OH 44202

Shaun D. Black
Ann Arbor, MI

* * * * *

ED:

We have been working with the OS-DMS Modules with the intent of using them to demonstrate the capabilities of our OSI equipment to area businesses and professionals.

We have discovered a few bugs scattered throughout the modules which we haven't seen in any of OSI's Tech Letters, and we thought we would pass them along to you for possible inclusion in a future issue of PEEK(65). Enclosed find a hardcopy listing of fixes for the following problems:

(1) In the DMS Accounts Receivable Module's Program "ARSTMT" we noticed that any time that alphabetic characters were included in the Invoice Number field (i.e. PO#03221) the invoice would not be printed. The fix for this problem is shown on the attached sheet in line 54000 of ARSTMT.

(2) In the DMS Inventory II Module Program "ORDER" we found that nothing would be posted to the 'Quantity on Order' field unless line 25014 was changed as per attached sheet, and also the changes shown in lines 1000-1020 were made to allow for the trailing '-1' on the invoice number to flag for posting to 'Quantity on Order' as described in the Inventory II Manual.

(3) Lastly, in the DMS Personnel Module, we found that after deleting and re-packing the dummy records that were resident on the disk for initial demo and trying to start the system up to use Checkfile "A" for the first week's records that we couldn't use "A" unless we ran the short program shown on the attached sheet.

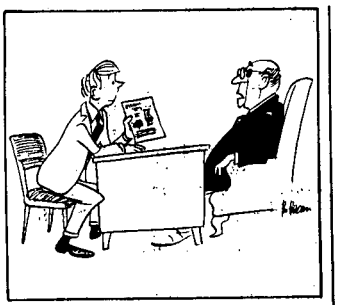
In the Personnel Module Program "PAYROL" we ran into an accumulated round off error in the gross deductions additions which we fixed by a 'brute-force' addition of the individual deductions as shown on line 61152 on the attached sheet.

In the Personnel Module Program "REPORT" when running the Payroll Summary we found some gross discrepancies between the totals of the different types of pay and the total gross pay. The correction for this problem was to line 20040 of the program "REPORT" as shown on the attached sheet.

We assume that these fixes are probably only the 'tip of the iceberg', but we thought they might be of help to those of your readers who have run into these same problems or might not be aware of them at this point. We will pass any more information your way that seems relevant. Keep up the good work!!!

Earl H. Phillips
Columbus, IN

* * * * *



'You Understand, of Course, That We Are Interested in a Computer That Will Eliminate Only Those Workers in the Lower Echelon.'

PROGRAM "ARSTMT"

(type in as one line)

```
54000 G8=1:FORQ=1TONN:
      W4$=L$(K1+9+4*(Q-1)):
      IFW4$<>"0"THENG8=0:RETURN
```

DMS INVENTORY II PROGRAM "ORDR"

```
2514 IFF8=2THEN25022

100 GOSUB 60000
1001 FC$(7,1)=""
1005 PRINT"ENTER THE INVOICE NO.";
     TAB(25);:INPUTIN$
1006 IF IN$="/"GOTO 525
1010 IF LEN(IN$)<2 GOTO 1005
1011 IF RIGHT$(IN$,1)=" "THEN
     IN$=LEFT$(IN$,LEN(IN$)-1)
1013 IF X=0THENF8=2:GOTO1020
1014 IF MID$(IN$,X,1)<>"-"THEN
     X=X-1:GOTO1013
1015 X=VAL(MID$(IN$,X+1)):
     IFX<>1THENF8=2:GOTO1020
1016 F8=1
1020 PRINT"PURCHASE ORDER NUMBER";
     TAB(25);:INPUTPU$
```

DMS PERSONNEL PROGRAM "PAYROLL"

```
61152 GD=FTX+FICA+LT+ST+INS+BU
      :PRINT
```

DMS PERSONNEL PROGRAM "REPORT"

```
20040 FORP=K5TOK8:T(K7)=
      T(K7)+VAL(C$(P,K1)):
      NEXT:REM GROSS PAY (FIX)
```

DMS PERSONNEL
QUICK-FIX TO INITIALIZE

CHECK FILES TO A

```
10 DEV"A"
12 OPEN"DIRFL0","PASS",1
14 A$="96     "
16 INDEX<1>=9
18 PRINT#1,A$
20 CLOSE 1
```

ED:

I recently purchased a C4PMF. It currently is back with the Math Box for repairs on the joy stick and its inability to load or call a track into memory. As soon as I get that straightened out, maybe I can help you, since we will have a mutual problem.
I have (in 6 days):

1) annoyed my wife.
2) added a background color change to colors, so listing/programming can be done with a color background.
3) added shorthand numeric input capabilities to BEXEC*.
4) modified UNLOCK to run"BEXEC*/DIR EXIT to BASIC.
5) half completed Othello from Prof. Peter Frey's algorithm - Byte, Vol 5, No. 7, July 80.

The first four are available for $2.00 including postage and handling.

I am interested in games for my computer

My C4PMF's fan causes 60 Hz interference with channel 5 in my apartment building despite using Radio Shack's Archer line interference filter. Any suggestions would be appreciated.

Stephen J. White
3517 Parkway Terrace Dr.
Suitland, Md. 20023

* * * * *

ED:

The October issue arrived today. It must be time to comment on the September issue!

Page 18: Mike Carroll is to be commended for identifying the bulk move routine. Unfortunately, the description omits that the target address (+1) must be in bytes $A4, A5. Also, your typographical error threw me for a loop -- the other addresses are $A6,7 & $AA,B, etc.

Page 19: Dru Cabler asked about the SEP from Grafix. I ordered one August 2nd. They warned there would be a delay, but said they would hold my check until three weeks before shipment. Apparently they did. The assembly/instruction manual arrived mid-October. The bare board itself, October 29th. I am mildly disappointed.

The board itself looks very well constructed. Layout is reasonable; wading through the schematics, it looks like it should function. But the function is a bit less than I had expected. It does NOT use OSI's character generator and does not include lower case. The four alpha-numeric modes mentioned in the ads might better be called colors, for that is what they are; in all cases 32 columns by 16 rows and upper case only.

The expansion memory is addressable only at $2000-5FFF I happen to have a modified MEM2 at that address. $8000 - $980F are occupied by display memory and the 6522 VIA. (I am running a Superboard with 24k, a 6821 PIA, & a Bunker/Ramo 32-column dot-matrix printer).

The SEP is set up for output through its self-contained TV modulator only. No data were supplied concerning output to a video monitor.

The manual is nicely printed. But 20 of its 33 pages are merely a reprint of the 6522 spec sheet. While this was necessary, it is not sufficient. No spec sheets are provided for the MC6847 & MC1372 chips (the rest are all garden variety chips).

The instructions are a bit sketchy, but adequate. Neophyte hardware builders might be well-advised to buy the manual first (It carries a printed $5.00 price tag). The display function table could use a lot more explanation. Maybe it will be clearer after I have experimented with the functions.

I plan to keep the board, but it will not do all I had hoped/planned for it to do.

Page 17: A comment which may help Gary Wolf. Make sure you don't have shorted address or data lines on your expansion board downstream from the buffers. This will appear as defective memory, but does not migrate as chips are shifted from socket to socket (Lord help you if you didn't use sockets). Furthermore, such address line problems can slip past simple-minded memory diagnostics. Try one which changes one bit (not byte) and then checks that no other bit(s) have changed. Good Luck!

I hope these comments may be useful to some of my fellow readers. I look forward to more useful tidbits from PEEK (65).

Myron E. Williams
Apalachin, N.Y.

* * * * *

ED:

I have a C1P driving the Heath H14 printer. Interfacing was done in the (well known) way of simply populating the C1P board with two transistors and a few resistors and a diode. Hardware handshaking is by routing the printer handshake signal through the C1P "RS232 IN" channel to the 6850ACIA "CTS" pin. Now the printer prints all PRINT statement output if the SAVE command has been executed (at 300 Baud). Complete printer control is easily accomplished with simple BASIC software and with no device driver needed. OSI deserves praise for a design which allows this to be done so easily and cheaply.

Now for the little trick which is really the heart of my message. By executing (program or immediate mode) the following BASIC code prior to printing, the C1P will drive the H14 at 4800 Baud:

POKE 61440,3: POKE 61440,20

In my experience, this increases average printer thru-put by about 50% over the 300 Baud rate. What the POKES do is reset and re-program the 6850 ACIA control register so serial output is at the clock rate/1 instead of the clock rate/16 rate as used for 300 Baud operation. In nine months of use, I have had absolutely no problems with this.

You must of course keep track of what the ACIA control register is set for and ensure it is back to 300 Baud before cassette input/output is done. A warm start sets up 300 Baud or it can be done by using the same two POKES with "17" instead of "20" in the second POKE.

David L. Flannery
Englewood, Ohio

* * * * *

ED:

I do like your journal ... possibly because it makes me feel not so alone with my OSI non-documentation problems.

In #8 a fellow in Missouri asked about interfacing a digitizing tablet to a C3. I've done this, but my solution is not wholly satisfactory. If anybody has a better way, I really want to know about it!

We (the Naval Research Laboratory) have a Houston Instruments Hi-Pad digitizer, for which we're the object of a patent suit, but that's another story. I've interfaced it with the CA-10(550) RS-232 board. It shares port 0 with a Paper Tiger Printer. Now it seems to me that a simple "INPUT #8,A$" ought to fetch the 15-character stream (13 ASCII digits and signs, then CR and LF), just as "PRINT #8,A$" happily sends data out to the printer. No such thing. BASIC goes away to look for input, appears to find none, and never comes back -- a cold-start is necessary to recover. I've tried "IO 80" as well as any number of POKEs to tell it where to look, with zero luck.

What does work, is this (the principal line is 100):

```
010 DIM A(14)
070 REM routine to read stream of 14 HIPAD bytes, ignoring
    the 15th (LF)
080 REM this must be at the start of the program, else it's
    too slow
090 I=0 : A(I)=PEEK(52995) : REM this resets the status register
100 WAIT 52994,1 : A(I)=PEEK(52995) : IF A(I)<>141 THEN I=I+1 :
    GOTO 100
102 REM 141 is carriage return; the HIPAD ASCII is offset by 128
104 REM 52994 is status byte, 52995 the data byte
110 X=0 : FOR I=2 TO 6 : X=X+A(I)-176 : X=10*X : NEXT I
120 Y=0 : FOR I=8 TO 12 : Y=Y+A(I)-176 : Y=10*Y : NEXT I
130 REM the 176 converts the offset ASCII to decimal
140 IF A(1)=173 THEN X=-X
150 IF A(7)=173 THEN Y=-Y : REM sign bytes of input
160 X=X/1E4 : Y=Y/1E4 : REM set decimal point for inches
170 (continue program)
```

Could the problem be that the ASCII is offset by 128? I've tried hooking the HIPAD directly to the RS-232 printer, and that works; it recognizes 141 as CR.

The trouble with this routine is that it is SLOW, barely able to keep up with the paltry 300 baud input. Merely adding program lines around it can slow it enough to make it miss characters. No, the local OSI rep couldn't help; and you know just how much help OSI-Aurora is. I telephoned them for help once with a problem that had the local rep stumped, got coldly brushed off, and to my knowledge OSI has sold no more machines to this laboratory. If anybody knows how to make INPUT #8 work, or perhaps a USR(X) subroutine, and will send it along, I will be truly grateful.

Jack McKay
Washington, DC.

Jack:

See next month!

Al

* * * * * *

ED:

In response to Mr. Beechhold's question on how to load cassette programs onto a disk-based machine (issue #10, Oct.) the following technique works on a C2-4P with a single drive:

Enter NEW to clear the workspace
Enter POKE 8993,1
Load the program.

The POKE sets the I/O input distributor flag to device #1 which is the ACIA used in the cassette port. After entering the POKE, input from the keyboard is ignored. However, by shrewd foresight or happy accident, when an illegal BASIC line is entered, such as the "OK" at the end of the program listing, the input distributor flag is reset to device #2, which is the keyboard.

Note that this will happen any time you get a syntax error message, as when you start the listing in the middle of a line and miss the line number. If this happens, back up, re-POKE and try again. The POKE location may be different in the Picodos operating system of the C1P. If so, perhaps your manual gives the proper input flag location.

Willis H. Cook
Lilburn, Georgia

* * * * *

AD$

ED:

I recently overheard discussions between OSIO users that indicated those individuals were writing to and reading from disk files in a manner that would be incompatible with files I frequently come in contact with. Specifically they write variable length fields (rather than fixed length), and input with no protection against the occasional comma and colon that is imbedded in string data (ex. J. R. Ewing, Esq.). I don't propose to argue the merits of fixed versus variable length fields -- no doubt each has its application. Instead, I will suggest some code which will (1) pad fields with trailing blanks to give them

the desired length, (2) protect these blanks during input and print operations, and (3) insure that imbedded commas and colons don't cause problems when they are not intended to be delimiters.

```
100 INPUT "ENTER FILE NAME,
 PASSWORD,DEVICE"; F1$,PW$,D1$

        Other code as needed

    190 DIM L(12),IP$(12)
    200 FOR I=1 TO 11:
        READ L(I):NEXT I
    220 DATA 10,12,8,7,10,11,
        5,7,9,10,12
    230 POKE 2972,13:
         POKE 2976,13

    400 FOR I=1 TO 11:
        IP$(I)=LEFT$(IP$(I)+
        "          ",L(I)):
        NEXT I
    420 POKE 207,36: FOR I =
         1 TO 11:PRINT%1,
         IP$(I):NEXT:POKE
         207,240

     500 POKE 2972,58:POKE
          2976,44
     520 CLOSE:END
```

Note the following: (1) To read a fixed length field from disk, use the same scheme as shown in line 420, except change PRINT%1 to INPUT%1; (2) The 2972 and 2976 POKES must not precede inputs which use commas or colons as delimiters; (3) There can be no embedded blanks between a POKE207,36 and its companion POKE207,240.

For handling a numeric variable rather than a string variable you might use one of these schemes:

- For the most significant digit to be left justified in the field, try

```
IP$(I) =
MID$(STR$(IP(I)+
"           ",2,L(I)).
```

- To retain the leading blank before the most significant digit, try

```
IP$(I)=LEFT$(STR$(IP(I) =
"           ",L(I)).
```

- Similar methods can be used to move the value to the right side of the field, with blanks padding the leftmost portion.

Incidentally, I would appreciate your commenting on which scheme is preferred if you are planning to use the file with KYUTIL. The first one does not work properly; keying on a left justified field gives results which suggest that the leftmost digits all have the same 'units', 'tens', 'hundreds', etc. value.

Kelsey Goodman
Upper Marlboro, Md.

Kelsey:

KYUTIL uses a string sort and will sort numbers properly only if they are left packed with zeroes; (345 will sort before 35, but 0035 will properly sort before 0345)

AL

* * * * *

ED:

Like Gary Wolf (in the September issue), I too encountered errors when working with programs using large memory. In my case, BS and OM errors resulted. Since I suspected a faulty memory chip, I devised the following memory check:

```
10 X=13050:REM IN BASIC,X=900
20 FOR Y=X TO X+1000:POKE Y,
    100:NEXT Y
30 FOR Y=X TO X+1000
40 Z=PEEK(Y)
50 IF Z<>100 THEN ? Y:STOP
60 NEXT Y
70 X=X+1000:GOTO 20
```

This program ran through to the end of memory, without a break. When I then reloaded the troublesome program, it ran without error. Apparently, the check forced my lethargic memory chip back into action. Surprisingly, this check also corrected the problem with the FRE function, described by John Sohl in the same issue. However, I still do not check for free memory without first recording the program.

Stanley Harshfield
Memphis, TN.

* * * * *

ED:

If PEEK (65) just saved me one hour of time per year, it would be worth twice the subscription. It has in fact saved me several hours.

When I buy a T-bone, I expect some fat... but ain't them T-bones good! Bring on the steaks, and no more tears. Here's a fact I learned from Rick W. at OSI:

The contents of the CD-74 hard disk buffer are arranged as follows:

MEM LOC	CONTENTS	REMS
E000	15 NULLS	SYNC PATTERN
THRU		
E00F	01	"
E010	DISK ID	8000 USUALLY
E011	"	"
E012	CYL LOW	
E013	CYL HIGH	
E014	TRACK	RANGE IS 0-11
E015	SECTOR	RANGE IS 0-4
E016	NOT USED	
E017	CHECKSUM	SUM OF PREV 7 BYTES - I THINK
E018	DATA	
THRU		
EE17	END OF DATA	0D IE <CR> AT 3584TH DATUM
EE18	CK SUM LOW	
EE19	CK SUM HIGH	
EE1A	TRAILING	
EE29	NULLS	16 NULLS
EE2A	UNUSED	
THRU		
EFFF	"	

Now, you CD-23 types send in some info.

Also, it would appear that grounding of pin 10 of U21 on the 592 board will inhibit the disk read causing an ERROR 12. But this buys us the ability to examine the header BEFORE the usually subsequent WRITE VERIFY. You may blow that gate of the inverter, but we did not as TTL is pretty rugged stuff. We did the above to confirm the correctness of the header sync pattern, etc.

Dale H. King
BKM Microsystems
Bryan, TX

TTY

Two BRAND NEW Teletype model 43 printing terminals

- Full ASCII keyboard and printer
- 300 baud serial operation (with computer or time-share network)
- $1100 -- both for $1995

Dave Moore
540 Green River Ct.
Annapolis, MD 21401

ED:

I'll answer my own question from several months ago about what connectors fit the OSI motherboard. They are Molex 09-64-1221 (male) & Molex 09-52-3121 (female). This combination also will fit the extension interface on the CPU board.

Incidentally, I ordered some of these from "Technical Products Co.", Gainesville, Fla. quite some time ago; they don't deliver very fast. I am beginning to wonder if they deliver at all!

Enclosed is a little program that will give entertainment to any of the artistic younger set among our OSI fans.

Neil Dennis
Bliss, NY

```
20 POKE 56900,1
30 GOSUB 300
40 FOR W=1TO32:PRINT:NEXT W
50 POKE 56900,0
60 FORQ=53443TO54979STEP64:POKEQ,188:NEXT
70 FORQ=53468TO55004STEP64:POKEQ,188:NEXT
80 X=53443
90 C=57100
100 FORT=1TO100:NEXT:REM  TIMING LOOP
110 X=X+1
120 IF X > 55004 THEN 290
130 IF PEEK(C)=3 THEN Q=46:GOTO 200
140 IF PEEK(C)=5 THEN Q=42:GOTO 200
150 IF PEEK(C)=0 THEN Q=32:GOTO 200
170 IF PEEK(C)=129 THEN Q=187:GOTO 200
180 IF PEEK(C)=33 THEN GOTO 230
190 GOTO 130
200 IF PEEK(X)=188 THEN X=X+40
210 POKE X,Q
220 GOTO 100
230 IF PEEK (X-1)=188 THEN X=X-40
240 POKE X-1,32
250 X=X-2
260 IFX<53443 THEN X=53443
270 N=N-1
280 GOTO 100
290 END
300 PRINT"DO YOU WISH INSTRUCTIONS?"
310 PRINT
320 INPUT"DEPRESS  'Y' OR 'N'";A$
330 IF LEFT$(A$,1)="N" THEN 500
340 IF LEFT$(A$,1)<>"Y" THEN 320
360 PRINT"PROGRAM WILL DRAW PATTERNS"
370 PRINT"ON THE SCREEN BY USING THESE"
375 PRINT"CHARACTERS"
380 PRINT
390 PRINT"LEFT SHIFT";TAB(20);"."
400 PRINT
410 PRINT"RIGHT SHIFT";TAB(20);"*"
420 PRINT
430 PRINT"CTRL";TAB(20)"BLANK"
440 PRINT
450 PRINT"REPEAT";TAB(20)CHR$(187)
460 PRINT
470 PRINT"'ESC' WILL ERASE BACKWARDS"
480 FOR X=1TO10:PRINT:NEXT
490 INPUT"DEPRESS ANY KEY TO CONTINUE";A$
500 RETURN:END
```

* * * * *

ED:

Try this for structured programming:

```
10 FOR I=1 TO 80
20:   A=I
30:    PRINT I
40 NEXT I
```

The ":" suppresses the automatic space-eating properties of BASIC, maintaining your spacing to indent the body of the subroutine (or whatever) a few spaces, making your program easier to read.

G. Lombard
Olympia, WA

* * * * *

OSI BUYS OKIDATA MEMORY

OSI has announced the purchase of the Okidata Memory Products plant in Santa Barbara, California which has been manufacturing the 74 Mbyte hard disk drive used in the OSI C3-B. This means that the entire C3-B is now manufactured by OSI, the computer at OSI in Ohio, the disk drive at OSI Memory Products, Inc., as the Santa Barbara plant will now be called.

Also, a new product will shortly be produced, based on the famous 4-platter 74 Mbyte disk drive. It will be a smaller version, with 2 platters, yielding 37 Mbytes of hard disk storage in a package which will run as fast and reliably (and probably as noisily) as the CD-74. It will of course be called the CD-37, and the computer in which it will live will be called the C-3C Prime. The word is that when stocks are exhausted, the old C3-C with the 23 Mbyte Shugart hard disk will no longer be offered.

(cont'd from p. 5)

A similar technique can be applied to a whole line of code if it keeps bombing out with no apparent cause.

The BASIC reason for the foregoing is that each line of code is stored in memory in a compacted form, where keywords like FOR, OR, AND, TAN, etc. are stored as single-byte "tokens" (See MICRO 15:20). Since the line of code is scanned from left to right, ignoring spaces, a variable ending in F (or consisting of F) can be combined with OR to become FOR. When the line is LISTed, the single-byte token prints out in long form with no intervening spaces.

Since a space is the absence of a visible alphanumeric

character, the trick I have described for "certifying" long variable names, or whole lines of code, could be considered as being at least the start of doing "anything" with "nothing".

C. Eugene McMurray

5083 Shadycrest Rd.
Columbus, OH

* * * * *

ED:

I have confirmed your observation that something happens when a 6502 executes a JMP(XXFF) instruction. However, the error may be on the part of the programmer rather than the hardware. Enclosed is a copy of a page from the MOS programming manual describing JMP indirect. Note that in cycle 5, the high order jump to byte is fetched from IAH, IAL + 1. No provision is made for incrementing IAH in case of a carry.

Several other instructions also discard the carry. However, these are well documented and explicitly stated in the manual.

Example 9.6: Illustration of JMP Indirect

Cycle	Address Bus	Data Bus	External Operation	Internal Operation
1	0100	OP CODE	Fetch OP CODE	Finish Previous Operation. Increment PC to 0101
2	0101	IAL	Fetch IAL	Interpret Instructions Increment PC to 102
3	0102	IAH	Fetch IAH	Store IAL
4	IAH, IAL	ADL	Fetch ADL	Add 1 to IAL
5	IAH,IAL+1	ADH	Fetch ADH	Store ADL
6	ADH,ADL	Next OP CODE	Fetch Next OP CODE	

E. Morris

Midland, MI

ED:

I have some observations, a request for help and a need to go on record to the general OSI user community, all relating to problems with a C3D hard disk system. This letter was provoked by our general impression of OSI support plus comments in your September 15th issue.

I am in the position of being responsible for maintenance and hardware development for a community of some 13 C2's and one C3-D system. Hence I am painfully aware of the abysmally poor support offered by OSI at sale time with respect to hardware diagnostics. I really find it inexcusable that any system is originally sold without at least elementary instruction verifiers and a RAM integrity tester. Clearly any system with any mechanical peripherals is even more in need of proper diagnostics. I suggest OSI take a look at DEC's X11 Package or Data General's DTOS as an indication of the desired goal.

I realize that small companies with limited software design resources do not have sufficient capacity to generate sophisticated diagnostic operating systems such as X11; however, it is obvious that full diagnostics must be available within OSI Production facilities and hence could be made available to the user community even at a cost.

We have had a hardware fault on our OKIDATA 3061/C-D74/OS-65U system for many months. Recently after numerous attempts at getting a good diagnostic (including writing to OSI via Johnson's of Medina with no response at all) we have accepted the inevitable and began writing our own diagnostic. What the final outcome will be I do not know, but I am sure that three man-months are required to generate a comprehensive diagnostic for this type of disk and I resent being forced to commit that amount of effort after purchasing such a system.

I appreciate your publishing this letter and welcome any response from your readers who may already have solutions to this sort of dilemma.

G.J. Hicks
McMaster University
1280 Main Street West
Hamilton, Ontario, L8S 4M4
Canada

Mr. Hicks:

We also have been baffled by poor/zero responses from OSI in the past, but find them greatly improved of late. When you complete your diagnostic routine, why not share it through PEEK (65)?

AL

* * * * *
ED:

C4PMF patch-up files.

Presented in brief, this is the procedure I followed to initialize a new disk and put it into service complete with random/sequential file capability, done on a single drive, 24K machine, running OS-65D V3.2:

Initialize disk using "init" command.

Load track 12, sectors 1 and 2 with ASCII 35 characters, this is two pages (256 bytes) of "#s". (POKE into memory locations 1 page in length, then SAVE on new disk).

Alter line 20090 of the CREATE program to read IF T0 <1 OR T0 > 39 THEN 20080.

Create new file on new disk and store program.

From old disk load track 12, sector 3 into available memory, 1 page long.

SAVE on new disk, track 12, sector 3.

Repeat with track 12, sector 4.

From old disk load "dir" program and SAVE on new disk.

These two utility programs are the ones I most frequently use, but others may be added to a disk as needs arise.

This will give you 35 free tracks for file storage for programs or data. Without the information located on 12,3&4 the disk will only LOAD and PUT source code, with the information, the disk will respond to GET and PUT, OPEN and CLOSE.

John Kamm
Cambridge, OH

* * * * *

GOODIES for OSI Users!

( ) C1P Sams Photo-Facts Manual. Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just $7.95 $_____

( ) C4P Sams Photo-Facts Manual. Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at $15.95 $_____

( ) C2/C3 Sams Photo-Facts Manual. The facts you need to repair the larger OSI computers. Fat with useful information, but just $34.95 $_____

( ) OSI's Small Systems Journals. The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only $15.00 $_____

( ) Terminal Extensions Package - - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U $25.00 $_____

( ) RESEQ 5.2 - - BASIC program resequencer plus much more. Global changes, tables of bad references, GOSUBs & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for $50.00 $_____

( ) SANDERS MACHINE LANGUAGE SORT/MERGE FOR OS-65U. Complete disk sort and merge, OS-DMS compatible, handles multiple fields, documentation shows you how to call from any BASIC program, then return to it or any other BASIC program on any disk, floppy or hard. Most versatile and fastest sort/merge yet. It should cost more, but Sanders says sell it for just $89.00 $_____

( ) KYUTIL - - The ultimate OS-DMS key file utility package. Creates, loads and sorts multiple-field, conditionally loaded key files, sorting at over 200 entries per second! Never sort another master $100.00 $_____

( ) The Credit System - - An accounts receivable system that accepts and verifies inputs (charges and payments) for a charge account system, then prints monthly statements, ages receivables, maintains complete disk files, produces aged accounts receivable analysis. Takes inputs in any order, prints statements always in date order of transactions Requires 65U $298.00 $_____

( ) SUPERMAIL - - The last word in mailing list packages. Uses DMS and the fastest label-printing technique known to produce zip-sorted labels, complete ABC circulation reports. Includes programs for input, editing, dupe checking, automatic soundex generation, label and report generation and printing of renewal notices. Requires 65U $1,798.00 $_____

( ) Cash enclosed ( ) Master Charge ( ) VISA

Account No. _____ Expiration date _____

Signature _____

Name _____

Street _____

City _____ State _____ Zip _____

TOTAL $_____

Md Residents add 5% tax $_____

Postage & Handling $ 2.00

TOTAL DUE $_____

master charge THE INTERBANK CARD

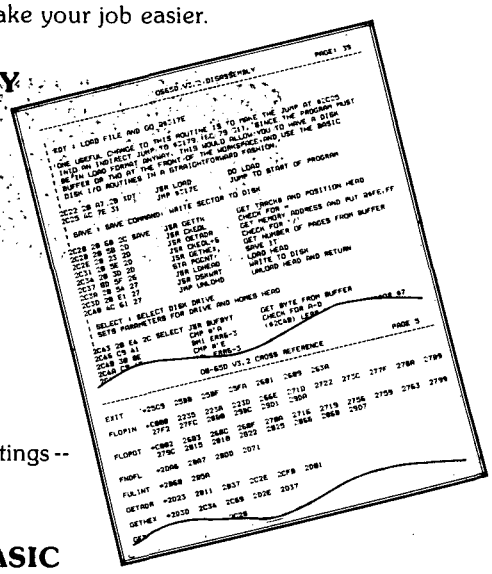
DBMS, INC.
PO Box 347
Owings Mills, MD 21117

VISA

OSI'ers!! **Just for YOU. . .**
from SOFTWARE CONSULTANTS. . .

great products that make your job easier.

1. OS-65D V3.2 DISASSEMBLY MANUAL With CROSS REFERENCE LISTINGS. . .



- 50 pages of Disassembly listings in standard Assembler format.
- FULLY COMMENTED code -- a complete, computer-generated concordance.
- 10 pages of CROSS REFERENCE listings -- not a useless program listing.

2. REF COMMAND UNDER BASIC
A Complete CROSS REFERENCE UTILITY. . .

- HIGH SPEED, memory resident utility available on OS-65D or OS-65U.
- LISTS ANY OR ALL occurrences of an individual BASIC variable, line number, or numeric constant.
- GENERATES A SORTED LISTING of all occurrences of any variable, and sends it either to printer or screen.
- REQUIRES 1K of memory. Can either be automatically loaded into the high area of memory (48, 32, or 24K), or under the OS-65U, can be hidden away in the OS.
- AVAILABLE EITHER on 5¼" or 8" floppy disks.

If this is the kind of help you can use, mail your check today for both great programming tools. Enclose $24.95 for each DISASSEMBLY MANUAL and $29.95 for each CROSS REFERENCE UTILITY. Your order will arrive postpaid from Memphis, TN, USA in 2-3 weeks. Dealer inquiries invited. Please specify disk size for CROSS REFERENCE UTILITY. Tennessee residents add 6% sales tax. Phone us at 901-377-3503 for rush orders.

NAME _____

ADDRESS _____

FIRM _____ DISK SIZE _____

CITY, STATE _____ POSTAL CODE _____

COUNTRY _____

PEEK (65)
The Unofficial OSI Users Journal

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

BULK RATE
U.S. POSTAGE
PAID
Annapolis, MD
Permit No. 563

DELIVER TO:

PLEASE SEND PEEK(65) FOR ONE YEAR (12 ISSUES)

( ) $12 Enclosed. (Domestic, USA.)

( ) $20 Enclosed. (Foreign Air Mail)

Maryland Residents add 5% sales tax.

NAME_____

STREET_____CITY_____

STATE_____ZIP_____

Please send the following back issues. I enclose $1.50 ea.
Domestic and overseas surface rate.
Maryland Residents add 5% sales tax.

( ) Jan. #1, "Welcome to the first issue"
( ) Feb #2, "A month ago in this spot"
( ) Mar. #3, "Peek Continues to grow"
( ) Apr. #4, "We are OSI fans"
( ) May #5, "The continued growth and health"
(This was mistakenly labled #4.)
( ) June #6, "This column should probably be"
( ) July #7, "Several times recently"
( ) Aug. #8, "A few minutes ago"
( ) Sept #9, "Of course. It had to happen"
( ) Oct. #10, "Publishing PEEK (65) is,"
( ) Nov. #11, "As you can see"