# PEEK (65)

The Unofficial OSI Users Journal

★★ $175 ★★

February 1982

Vol. 3, No. 2



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


INSIDE:



# Column One

Also, I recently had a look at a new Basic Reference manual from OSI. It is attractive, with full-color cover and two-column layout, and seems complete, covering ROM, 65D and 65U Basics all in one manual. So nice looking, in fact, that I hesitate to criticise; but I must.

In addition to the inaccuracies and stylistic flaws which creep into anything anybody writes (even my stuff!) this manual suffers from what I consider to be the cardinal sin of manual writing. It was never tested. No one representing the audience for which the manual was written, sat down with the book and a computer and tried to use it. Had they done so, they would have eliminated most of the errors and unclear places, and it would be a fine manual.

As it is, it is several cuts above what we have seen from OSI before, and will doubtless be very useful. The price printed on the front cover was $6.95, which seems quite reasonable.

Concerning this month's issue of PEEK(65): for several months now, we have heard the occasional complaint from a C1P or 65D user that we were paying too much attention to CP/M and 65U. Well folks, this month's issue is almost all C1P and 65D. Not a word about CP/M. Please, you business system users, don't write me letters about how all our stuff is for the smaller hobby machines! What I mean is, it all averages out.

Among the goodies for small system users is a description of a new DOS called HEXDOS, mentioned but not tested previously. The article encourages communication among users of the new DOS. So do I, with one difference. Communicate with each other through PEEK(65). There will be lots more on HEXDOS in these pages, especially if you write something.

Perhaps most exciting of all to me, we have several new authors this issue. The "Call for Articles" is working! But we are missing one more new author -- you! Whatever you are doing with your OSI machine, whatever comment you have about hardware, software, PEEK(65), your favorite computer club, MicroNet, the Source, or anything else in the world which interests you, write it down and send it to us, preferably with some detail and a listing or picture or two, in the form of an article. At least send in a letter to the editor.

We need articles on hardware mods, on new programs or modifications to old ones, new PEEKs and POKEs you have found, equipment you have tried from other manufacturers and how well/poorly it worked with your OSI computer, with as much detail as possible.

*al*

## THE 65U GOODIE BOX

by
Ken Holt
H/B Computers, Inc.
217 E. Main Street
Charlottesville, VA
(804) 295-1975

This month, I'll start off with a few goodies in the form of some useful POKEs; then I'll clean up the loose ends left from the last installment.

Are you tired of fighting 65U file passwords? The protection scheme isn't very secure, I'm afraid. I've personally seen at least three ways to beat the system. People are catching on fast now, so I might as well tell you the really slick way of doing it so you can avoid some of the time-consuming circumventions I have seen. Just do the following: POKE 11193,169: POKE 11194,0: POKE 11195,96. What this does is to tell the system to forget about passwords completely. Now, you need not specify a password on any OPEN, LOAD, or RUN command, regardless of the password or access rights defined in the directory.

Have you ever wanted to limit the number of characters allowed for an INPUT statement? Maybe for a phone number or social security number? Just POKE 1398 with maximum number of characters allowed, then do your INPUT. Be sure to POKE it back to 71 when you are done; if you don't, you won't be able to type very long lines when the program exits. For instance, if you POKE it to 2, then exit to the immediate mode without POKEing it back, you can't type RUN, CONT, LIST, SAVE, or anything else useful. Also, NEVER POKE it to anything higher than 71 or BASIC will probably go down in flames.

Have you ever gotten a "STALLED" message for device 5 even though nothing was wrong with the printer? This is often caused because the printer is very slow. BASIC gives the printer only so much time to respond; if it does not respond in that time, it assumes that the printer has stalled. There are two ways around this (besides buying a faster printer). First, you can lengthen the time limit that BASIC uses to decide when to give up. POKE 15886,14 will give the printer a little more time to respond. You should experiment with the specific value to POKE (12 is normal). The second way is to forget about printer stalls completely. POKE 15896,0 will cause BASIC to patiently wait forever until the printer becomes available; the "STALLED" message will never be issued.

The last POKE is more of a strange curiosity rather than anything useful. POKE 1797,44 will make all the line numbers on the left side of BASIC statements disappear during a LIST or trace (FLAG 7). POKE 1797,32 will make them come back again. Weird!

In the last installment, we covered a routine that enables one to program a series of screen "forms". These "forms" can be filled in by an unskilled operator with little difficulty. I promised to explain how it works this month. Here it is:

The screen formatting routine begins at 19000. It uses three dimensioned variables: FS$, FS, and FV. FS$ contains information on the nature of the field (heading, input, or output), fixed information (such as text in a heading), and directions on what element of the FV$ array the field value is stored in. FS contains information concerning the physical attributes of the field on the screen (X and Y coordinates, and field length). Finally, FV$ contains the actual field values. Values are taken from FV$ and placed on the screen or taken from the screen and placed in FV$, depending on the field type. The specific element of FV$ is determined by one of the items of information stored in FS$.

The routine begins by doing a RESTORE, thus rewinding the READ data pointer to the start, then searching for a special code that marks the beginning of the DATA statements used by the routine. This special code is defined in line 19060; the search is done at line 19080.

Now, the routine is ready to get the definition for the first screen. At 19110 the field count (FC) is initialized to zero. Then a function code is read from the first data statement. If the code is "E", the end of the screen definition has been reached and a branch is taken to 19210. Otherwise, the code is checked to make sure it is one of the four legal ones. If so, the field count is incremented and the specifications for the field are loaded into the FS$ and FS arrays. As seen at 19150, only input fields have a specified length. Next, the program loops to 19120 to read more field specifications. Eventually, the "E" code is hit, and the program proceeds to 19210 to display the "form" on the screen.

At 19210, the screen is first cleared. At 19220 through 19260, all heading and output fields are located and displayed on the screen; all input fields are skipped. For each field, X and Y are set (from the FS array) and the 19900 subroutine is called to move the cursor to the correct spot. For a heading, the text is printed directly from FS$. For an output field, the number of the element in FV$ is taken from FS$.

Now we search for all the input fields (starting at 19270), and skip everything else. When one is found, we branch to 19320 and position to the coordinates on the screen. At 19330, a "change flag" is checked; if it is on, we blank the input field for its entire length. Then, at 19340, we play a trick on BASIC. A normal INPUT statement causes a prompt of a blank followed by a "?". Line 19340 saves these prompt characters for later and sets the prompt to nulls. This, in effect, causes INPUT to not issue any visible prompt characters. Line 19350 cancels the special properties of the colon, comma, and quote. It also allows just a carriage return without resulting in a "?REDO FROM START" message. In 19360, we POKE 1398 with the input field length; this prevents the operator from typing input longer than what the field can handle. Finally, we do the INPUT.

The POKE following the INPUT and the POKEs on the next two lines just put everything back


**PEEK (65)** is published monthly by DBMS, Inc.
Owings Mills, MD 21117.
Editor - Al Peabody
Technical Editor - Dickinson H. McGuire
Asst. Technical Editor - Brian Hartson
Circulation & Advertising Mgr. - Karin Q. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates

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

All subscriptions are for 1 year and are payable in advance in US Dollars.

For back issues, subscriptions, change of address or other information, write to:
**PEEK (65)**
P.O. Box 347
Owings Mills, MD 21117

the way it was, so BASIC can work normally again. In 19390 the value is compared to a slash (/). If not a slash, the value is stored in the appropriate FV$ element and execution branches to 19290 (to get the next input field). If the input value was a slash, line 19410 blanks out the field on the screen. Then, lines 19420 and 19430 search backwards for the previous input field, if any. If none is found, the routine ends with CN (cancel) set to -1 (true) to indicate that the screen was not completed. Otherwise, the program returns to 19320 to re-input the previous field. If line 19290 finds that no more input fields are left, the routine returns to the caller with CN set to 0 (false) to indicate the process was completed.

The routine is built so that it can be entered at one of two alternate points in addition to the main one. If the routine is entered at 19100, the program does not RESTORE the READ data list; this causes the routine to read in field specifications again, thus picking up the next screen in the list. If the routine is entered at 19200, the program does not read any screen specs, but uses the ones already in the FS$ and FS arrays. This allows the re-use of a screen repeatedly.

The DATA statements from 19510 to 19570 comprise the definition of the first screen, while 19580 to 19640 define the second screen. You should, of course, already be familiar with the routine from 19900 to 19950, since you should have written it yourself for whatever type of terminal you are using.

Now that you know how it works, how about improving it? Instead of just a single "input" type, how about "input numeric" (IN) and "input alphanumeric" (IA)? For the numeric, check to make sure the FV$ string has nothing but digits, a decimal point, and maybe a plus or minus sign. Maybe an "input money" (IM) that checks for exactly 2 decimal places. The "output" field type could also be improved: "output money" (OM) could use the $L or $R feature to improve the appearance of a field. And what about something for "heading"? A special X-coordinate value: if the X-coordinate is -1, automatically center the text on the line. There are many more possibilities. The above suggestions are just to get your creative juices flowing. Enjoy yourself!

## USR(X) DEFINED

by: L. Magerman
6 Pumpkin Pine Road
Natick, MA 01760

Talking recently with an OSI user from New Jersey, I was asked questions like "What is a USR function? How does it work? What can it do? Why would I use it?" "You know", I said, "you sure ask a lot of questions for somebody from New Jersey!"

Seriously though, it does point out a recurring problem. The information on the USR(X) function is hard to find and when you do find it, it is sparse, incorrect and incomplete. Let's try to remedy that and answer most if not all of the questions people are asking. I will assume that those reading this article have a basic understanding of BASIC and machine/assembly language programming.

The main purpose of the USR(X) function or call is to access a machine language (referred to hereafter as ML) program or routine DURING the execution of a BASIC program. In addition, it allows control to be returned to BASIC from the ML routine (just make the last ML instruction an RTS). Furthermore, parameters (data) can be passed between BASIC and the ML routine - more about this later.

Consider a screen clear routine in BASIC which is normally about 30 odd PRINTs imbedded in a FOR/NEXT loop. It clears the screen adequately but slowly, compared to one done with ML routine wich can be almost instantaneous. The USR(X) function is the key that provides access to the ML routine.

The procedure is simple:

1) Put the ML routine into upper memory either by CAlling (disk) or POKEing it in the BASIC program or with the assembler using an A3 command. (The ML routine must end with a RTS for control to be passed back to BASIC). [or, put it ahead of your BASIC program in 65U - see PEEK (65) June '81, page 2].

2) In the BASIC program POKE the address of the 1st executable statement of the ML routine into the locations shown below (depends on the system you are running).

3) In the BASIC program access the ML routine with a X=USR(X) statement where X can be any dummy variable.

| | BASIC-in-ROM |
|---|---|
| Lobyte of ML poked to -> | 11 |
| Hibyte of ML poked to -> | 12 |

| | OS65D V3.N |
|---|---|
| Lobyte of ML poked to -> | 574 |
| Hibyte of ML poked to -> | 575 |

| | OS65U V1.M |
|---|---|
| Lobyte of ML poked to -> | 8778 |
| Hibyte of ML poked to -> | 8779 |

For example, if you are using 65D V3.2 and the first executable statement of your ML routine is at $6789:

POKE 574,137 (137=$89)
POKE 575,103 (103=$67)

As I said, if the last ML instruction is RTS, you will return to your BASIC program at the instruction following X=USR(X). Okay, now you can go from BASIC to the ML routine and back to BASIC one time. If you want to repeat the same ML routine elsewhere in the BASIC program you only have to do a USR call as before. If however, you wish to access another ML routine, you must POKE the location of its 1st executable statement to the locations listed above before you do the call.

Now besides accessing the ML routine, it would be nice to be able to pass some parameters (data) between it and BASIC. Consider a player vs. computer game in which the computer must quickly calculate some information about the player's position on the screen and react accordingly. Calculations done in BASIC may slow down the game so much as to make it unworkable, while if done in a ML routine they would appear to be "instantaneous". Perfect for the USR(X) call!

There are two ways one might pass data to the ML routine. The simplest method is to POKE the value (low byte and high byte if greater than 255) to some convenient location in upper memory before performing the USR(X) call as before and pick up the value(s) with the ML routine. Conversely, data may be passed from the ML routine to BASIC by storing the value(s) in memory and

PEEKing at those locations after returning to the BASIC program.

The other method of passing data from BASIC is to put the value to be passed in the argument of the USR(X) function, i.e., Z=USR(value) and pick it up at specific locations shown below after the ML routine is accessed. Some knowledge of how the USR(X) function stores the argument is necessary to use this method to its fullest capacity:

When control is passed from BASIC to the ML routine by the USR(X) statement, the argument of the USR(X) function is stored as a two's complemented 32 bit signed number in locations $AE to $B3 ($AC to $B0 for BASIC-in-ROM). Thus it is possible to pass both fractional and integer values to the ML routine within the range of +/- 1.7014118 E+38 if one is willing to do the necessary decoding of the locations $AE to $B3 (See reference 7).

This decoding is not ncessary however, if the values passed are integers within the range of +/-32767. There is a subroutine which will transform the 32 bit two's complemented values into two 8 bit numbers (two's complement if negative) which can be accessed by the ML routine with an indirect jump to location $6 on page zero by starting your ML routine with:

```
JSR GET       Gosub to input
              transformation
              routine
    /         Body
    /            of
    /             ML
   RTS        Return to BASIC
GET JMP ($6)  Input transf.
              routine (has its
              own RTS)
```

After the return from the JSR GET, the value can be picked up by the ML routine at the locations shown below:

| JSR(X) DEF. | BASIC-in-ROM |
|---|---|
| High byte of input | $AE |
| Low byte of input | $AF |

| | OS65D V3.0 to V3.3 & OS65U V1.1 to V1.3 |
|---|---|
| High byte of input | $B1 |
| Low byte of input | $B2 |

Finally, to pass an integer value back to BASIC one merely loads the accumulator with the high byte and the Y-register with the low byte of the value to be passed, and does an indirect jump to a data output routine at $8 to get back to BASIC like this:

```
/          Body
/             of
/              ML

LDA HIBYT  Put high byte of
           output in accumu-
           lator
LDY LOBYT  Put low byte of
           output in Y-regis-
           ter
JMP (8)    Return to BASIC
```

When BASIC regains control, the value passed back from the ML routine will now be equal to the Z variable in the statement Z=USR(X).

The locations shown above have been tested and do work for all versions of OS65D and OS65U EXCEPT OS65D V3.3 which has a bug. The vector at $8 is incorrect and must be patched (this confirmed by the OSI Technical Support Group in Ohio). Here's how:

Boot the 65D V3.3 system and check location $8. If there is anything other than $18 (=24 dec) and $12 (=18 dec) at memory locations $8 and $9 respectively, exit to the Extended Monitor, !CA 4800=04,1 and open location $51C2. Change the value there to $18 and the next location ($51C3) to $12. Then !SA 04,1=4800/B. This fixes the bug so that when BASIC lays in zero page, the vector at $8 and $9 is correct.

References:


1. C4P Owner's Manual, 1979, pages 126-134, 181

2. OS65D V3.0 User's Manual, October 1978, page 6 of the Appendix

3. 65D Tutorial and Reference Manual (V3.3), August 1981

4. OS65U V1.1 Operator's Manual, June 1978, page 78, 80

5. OS65U V1.3 Reference Manual, 1981, page 50

6. OSI Small Systems Journal, September 1978, page 3

7. OSI As/Ed & Ext/Mon Ref. Manual, 1981, Chapter 9 and Appendix M

8. OSI BASIC Manual, 1981, Chapter 13


★

**HEXDOS REVIEW**


by: Ken Shacter and
Norman D. McMullen
113 Dixie Circle
Slidell, LA 70458


The anouncement of an alternative C1P Disk Operating System to OSI's 65D in these pages in August was greeted by great enthusiasm, I am sure, by the readers of PEEK (65). But alas, there was to be no review! Thanks to the cooperation of Mr. Steven P. Hendrix, author of HEXDOS 2.4, a review will be provided by the members of the Slidell (LA) User's Group (SLUG). We intend to present this review in several installments so the information may be disseminated rapidly, and the individual columns may dwell on specific topics. This month's column will serve as an introduction to the Operating System.

Who Is It For?

HEXDOS 2.4 is a Disk Operating System specifically for disk-based OSI C1P/Superboard II microcomputers. Delivery of the DOS was prompt, and the packaging was adequate to ensure safe handling by the Postal Disservice. The padded semi-rigid envelope contained a 33 page instruction manual (soon to be professionally printed) and a 5-1/4" Verbatim floppy disk with a reinforced spindle hole (nice touch). The write-protect tab was not affixed to the disk as alluded to in the manual, but this was easily remedied. The disk contained the DOS, several utility routines, and some demonstration games/programs to better allow the user to see how to invoke some of the features of HEXDOS.

WHAT CAN IT DO?

The HEXDOS Operating System itself is less than 2K bytes long, but a lot is packed into those 2K bytes. For those of you that missed the features column in the August 1981 issue, let us briefly reiterate what HEXDOS has to offer:

** As previously mentioned, HEXDOS is completely resident in only 2K bytes of RAM.

** Named program and data files are supported, with direct access to specified disk tracks when desired.

** Random Access disk files are supported (with the use of a supplied BASIC subroutine).

**Z-FORTH IN ROM** by Tom Zimmer
5 to 10 times faster than Basic. Once you use it, you'll never go back to BASIC! $ 75.00
source listing add $ 20.00

**OSI FIG-FORTH** True fig FORTH model for 0S65D with fig editor named files, string package & much more $ 45.00

**TINY PASCAL Operates** in fig-FORTH, an exceptional value when purchased with forth.
TINY PASCAL & documentation $ 45.00
FORTH & TINY PASCAL $ 65.00

**SPACE INVADERS** 100% machine code for all systems with 64 chr. video. Full color & sound on C2, 4P & 8P systems. The fastest arcade program available. $ 14.95

**PROGRAMMABLE CHARACTER GENERATOR** $ 99.95
Use OSI's graphics or make a complete set of your own! Easy to use, comes assembled & tested.
2 Mhz. boards $109.95

**PROGRAMMABLE SOUND BOARD** $ 74.95
Complete sound system featuring the AY-3-8910 sound chip.
Bare boards available. $ 29.95

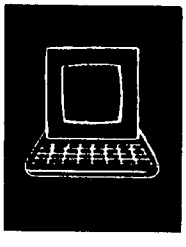
**32/64 CHARACTER VIDEO MODIFICATION** $ 39.95
Oldest and most popular video mod. True 32 chr. C1P, or 32/64 chr. C4P video display. Also adds many other options.

**ROMS!!!**
Augment Video Mod with our Roms. Full screen editing, print at,selectable scroll, disk support and many more features.
Basic 4 & Monitor $ 49.95
Basic 3 $ 18.95
All 3 for $ 65.00

**65D DISASSEMBLY MANUAL.** by Software Consultants
First class throughout. A must for any 65D user.
$ 24.95

*NUMEROUS BASIC PROGRAMS, UTILITY PROGRAMS AND GAMES ALONG WITH HARDWARE PROJECTS. ALL PRICES ARE U S FUNDS. Send for our $1.50 catalogue with free program (hardcopy) Memory Map and Auto Load Routine.*

CHARGEX VISA master charge



## OSI Software & Hardware

3336 Avondale Court
Windsor, Ontario, Canada N9E 1X6
(519) 969-2500

3281 Countryside Circle
Pontiac Township, Michigan 48057
(313) 373-0468

progressive computing

** Debug options for BASIC programs include single-stepping and line-tracing during execution.

** Instant screen clear is available from BASIC programs or the immediate mode.

** Output to the CRT screen may be temporarily "frozen" by depressing a single key during LIST or RUN.

** Multi-port I/O is available (although only one port at a time).

** Hardware modifications are suggested to accomplish the following:

a. Real-time clock (not time-of-day clock).

b. Tone generator.

c. Single key "repeat last command".

d. Automatic power-on reset.

e. Disk motor control.

HEXDOS keeps its compact size by using the C1P's resident BASIC-in-ROM language processor. While running in BASIC, all previous ROM-based commands remain unchanged (even access to the cassette port) except the use of USR(X) and FRE(X) functions. The use of FRE(X) is discouraged for two reasons; 1) the "Garbage Collector" routine in ROM remains unchanged, (thus the bug still lurks, ) and 2) the GC does not recognize your file buffers and will clobber them (unless protected from BASIC workspace). The use of USR(X) will be explained in more detail in a future column, but suffice it to say that USR(X) behaves somewhat akin to Phil Hooper's OSICALL routine described in the December 1980 issue of PEEK (65), page 7.

In addition to the above differences, six keystrokes have functions differing from standard BASIC-in-ROM. These are CTRL, REPEAT, CTRL-L, SHIFT-RETURN, RUB OUT and ESC. CTRL "freezes" screen output while depressed, REPEAT acts as CTRL-C does normally, and CTRL-L clears the screen (also PRINT CHR$(12) in a BASIC program). SHIFT-RETURN places a listed line in the edit buffer, while RUB OUT and ESC move the cursor nondestructively backwards and forwards. ESC and RUB OUT work in the immediate and edit mode; however, no provision is made for character insert/delete from within the edited line.

Documentation

Mr. Hendrix supplies a fairly well written manual with HEXDOS. This is not to say, however, that it could not use some improvements. In order to keep costs down, Mr. Hendrix originally printed the manual with a dot matrix printer (high quality when a good ribbon is used) on paper measuring 4-1/2 by 5-1/2 inches, which was contained in a clear plastic folder bound by a rigid strip of plastic. This has been replaced (so we are told) by a professionally printed manual on a 6 X 8 inch paper. The new manual is a reprint of the original, so all comments should remain valid.

The author's introduction to his Operating System is short, sweet, and to the point. His opening comments serve as a disclaimer for suggested hardware modifications (outlined earlier) that correct several OSI blunders and help enhance HEXDOS flexibility. The manual's Table of Contents is brief and concise. An index and cross-reference section would be handy, along with a summary sheet of system commands and their use, and a table of PEEKs/POKEs to enable/disable certain functions. Perhaps these comments could be included in the next revision to the manual. The body of the manual is well written (one might think Mr. Hendrix a journalist versus a computer science major!) and does an adequate job of explaining control keys, disk I/O commands, USR(X), debugging aids, and program conversions from 65D. We take exception to some of Mr. Hendrix' nomenclature when discussing commands, but the conventions are easily gotten used to.

The Appendices in the manual cover disk format, memory map, and hardware modifications (as mentioned earlier). No schematics or diagrams are provided for the suggested modifications, although it appears to be written well enough that these are not necessary. We will render a verdict later on this last comment. Games and utilities are also discussed, as well as support for random access disk files. For those C1P owners that have Aardvark's CEGMON, version 3.0 of HEXDOS replaces those functions of version 2.4 found in CEGMON with a few extra goodies (Hey! Anybody have a C1P-MF with CEGMON?).

So far only two subtle errors have been found in the manual. First, in the section discussing the CREATE utility, it says that the program prompts for the number of tracks to be reserved. However, the utility actually asks for the number of 256-byte pages (8 pages to a disk track) to be reserved. This is important since no track extents are provided for once a file is allocated. Therefore, one should be sure to allocate extra room for file expansion (we will investigate ways to get around this in a later column). One should note that the preceeding discussion ONLY applies to files set up for DATA or MACHINE LANGUAGE contents. BASIC automatically reserves disk space during SAVE. No provisions are made for storing more than one file per track, so the minimum true disk allocation is one track (2K bytes), no matter how small the program or requested allocation. (This is also true of 65D BASIC files). Second, in the discussion of the real-time clock, the statement is made that user-written programs that are triggered by the clock should not use page zero memory below hex location FO. This should state memory locations below hex F2 are taboo, which leaves memory locations hex F2 through FF for user use.

Support/Future Work

One should not refer to HEXDOS as new, since approximately 100 copies have been sold to C1P owners. For those users of version 2.3, an upgrade to version 2.4/3.0 may be obtained from Mr. Hendrix for $11.00. New owners may purchase HEXDOS for $55.00. Future plans for the DOS include an Assembler, line renumberer, and a 65D-independent version of FORTH. Also in the workings is a newsletter.

Conclusions

The question may (and probably will) be asked, "Why should I spend $55.00 on a DOS that has practically no software support base and (currently) no assembler?" From SLUG's use of HEXDOS so far, it can be truthfully stated that HEXDOS is a pleasure to use. It may not have some fancy bells and whistles as another DOS we all know of, but yet this other DOS doesn't support disk motor/head control or an onboard "true" tone generator (versus the port through the keyboard). The suggested hardware mods may be worth the price of HEXDOS itself, and the RAM freed up by using the resident BASIC language processor (versus disk BASIC) is worth real money... space for more file buffers (HEXDOS will support more than two), larger arrays, or a down payment on a printer.

HEXDOS isn't advertised as "developmental" as is 65D, but need we say more? The source listing for HEXDOS isn't currently available gratis, but one may obtain it for a price comparable to Software Consultants' 65D Disassembly Manual (the price is higher for HEXDOS, and it comes on a 65D compatable ASM formatted disk).

We wish to urge current owners of HEXDOS to submit articles on your experiences with the DOS, as well as programs and modifications you have written. In our next installment we will compare 65D and HEXDOS, along with converting over from 65D or tape. The hardware modifications will be covered as they are attempted. HEXDOS may be purchased for $55.00, from Steve Hendrix at 415 S. Pierce, Enid, OK 73701, or "place your bets" with the 6502 Program Exchange, 2920 W. Moana, Reno, NV 89510.

★ ★ ★

# At last!
# Software Development TOOLS
# for Professional OS-65U Programmers:

## FIND:

If you program in OS-65U BASIC, you need FIND, a machine code overlay which resides permanently in the operating system, extending the FIND command to allow searches for variables, literals, statements, commands, functions, and constants such as line numbers.

FIND is an invaluable tool for writing code and debugging programs — especially someone else's! May be used in the immediate mode with any BASIC program in the user's workspace.

## COPY & DELETE:

These utilities save you from spending hours manually copying and moving BASIC program code. Both reside in the operating system, allowing use in the immediate mode.

COPY copies program lines character-for-character to a new line number location. Tests to make sure no existing lines will be overwritten.

DELETE removes program lines. Any linerange may be specified, although the DELETE command without a linerange is not accepted (to prevent accidental erasure of a whole program).

Using a single COPY-DELETE command with a linerange performs a MOVE of the block of lines to a new location.

COPY & DELETE are available without EDITV3 for video-based systems.

## EDITV3:

Has the usual OSI EDIT features, including Control R, F, P, and Tab, Delete, and Backspace. New features: Control D (erase from cursor to EOL), Auto Upper Case, Bell on All Illegal Characters, Auto -CR- at First-space-closest-to-EOL Flag, Masked Output Flag (prints X's instead of characters for password protection). Underscore and @ symbol are legal characters, replaced with DEL and Control X respectively. Backspace and Delete/Insert work normally. Control T now toggles Insert/Overstrike Character mode, allowing the user to overstrike characters in the middle of a line (without first deleting the old characters and then typing the new). Edit Line command deletes both first space and space between line number and statement, adding one character to editable lines.

Above flags may be set using the calling routine. The Input Editor may also be preloaded with a string to be edited, placing the cursor on the appropriate character within the line (for use in BASIC programs). EDITV3 with COPY & DELETE requires no reserved words.

## MONITR:

Find out what is going on in your Level 3 system! This simple-to-use program allows you to monitor activity while Level 3 is running. MONITR shows which users are 'up,' which are active, what program they're using, the line number (if running), and their input status. Also enables the user to debug multi-user partitions and programs. Runs from any port, and allows any user to reboot any other user. Designed for use by programmers as well as system operators.

## PRICES:

Each program package supplied on an 8-inch flexible disk.

Package 1: FIND $75.00
Package 2: COPY & DELETE (for video-based systems) $75.00
Package 3: FIND, EDITV3 incorporating COPY & DELETE (not for video-based systems) $235.00
Package 4: MONITR w/Talkie $175.00

Brochures available — write or call.


Brown/Collinson Associates
619 "E" Avenue
Lake Oswego, Oregon 97034
(503) 635-5055

**OS 65D3 No. 4 IN A SERIES**

D.R. "Stretch" Manley
5664 East Evans Creek Road
Rogue River, OR 97537

TWO RANDOM ACCESS FILES

We've pretty well stayed on the surface of BASIC and OS65D3 for the first three articles. The next two will get your feet wet, if you are willing to play in the water (machine language).

The column this month will show how to convert BASIC to use device #7 as a random access file device, just like #6. All of the code will fit in the currently allocated space for BASIC and the normal PUT/GET overlay, if the math code speed-up from the May 1981 BYTE is installed (corrections in the Sept. 1981 BYTE). If you don't install the speed-up code (a worthwhile project, by the way), you will have to put the "MULTIP" and "DIVIDE" routines somewhere else, which entails a special area protected from BASIC, extra sector calls or set-up program runs or some other non-elegant way to do the job.

This new code does two things, really. First, it converts device #7 to random access, of course. But it also starts using the "dirty flag" as it should be used, I feel. Most of the explanations are in the code, so I won't say any more, just list the assembler output. I hope you don't have too much trouble getting this to work!

Next time, I will show you what goes in the places reserved for expansion. How about a new BASIC command, "DISK RECORD, D, L", where D is 6 or 7 and L is the length of the record in bytes? The new command also automatically figures the corresponding number of records per track and saves that too. This command only has to be used once, after the file is opened. Nice for really using disk space efficiently.



PROGRAM NAME = "TWO.RAN.FILES"

```
                        ;     SYSTEM LOCATIONS
228A=                   ;       DEVICE=$228A ;00=6,08=7
23AC=                   ;       POINT =$23AC ;POINTER FOR #6 INPUT
                        ;
                        ;  DEVICE #6 PARAMETER LOCATIONS. ADD #$08 FOR #7
2326=                   ;       LOBUFF=$2326
2327=                   ;       HIBUFF=$2327
232A=                   ;       FIRSTK=$232A
232B=                   ;       LASTK =$232B
232C=                   ;       THISTK=$232C
232D=                   ;       DIRTY =$232D
                        ;
                        ; TEMPORARY VARIABLE STORAGE (IN OP. SYS. BUFFER)
                        ;
2E1E=                   ;       TEMP1 =$2E1E
2E1F=                   ;       TEMP2 =$2E1F
2E20=                   ;       TEMP3 =$2E20
                        ;
                        ;  SUBROUTINE ADDRESSES IN BASIC AND THE OP. SYS.
0E1E=                   ;       ERR:  =$0E1E ;BASIC SYNTAX ERROR
10D0=                   ;       ERR4  =$10D0 ;BASIC ERROR 4, FC ERROR
2163=                   ;       GETDEV=$2163 ;RETURNS DEVICE
0E13=                   ;       ISCOMA=$0E13 ;IF CHAR NOT COMMA, SN ERROR
2CF7=                   ;       SWAP01=$2CF7 ;SWAP BASIC AND OPSYS PG 0 & 1
2754=                   ;       DNHEAD=$2754 ;DROP DISK HEAD
2761=                   ;       UPHEAD=$2761 ;RAISE DISK HEAD
2967=                   ;       DOREAD=$2967 ;READ DISK INTO BUFFER
2477=                   ;       WRITEB=$2477 ;THIS DROPS HEAD, WRITES BUFFER
                        ;                    ;TO DISK, RESETS DIRTY FLAG
                        ;                    ;AND RETURNS WITH HEAD STILL
                        ;                    ;DOWN.
2283=                   ;       CLOSE =$2283 ;THIS DOES:
                        ;                    ;JSR GETDEV,
                        ;                    ;JSR SWAP01,
                        ;                    ;JSR WRITEB,
                        ;                    ;JSR UPHEAD,
                        ;                    ;JSR SWAP01,
                        ;                    ;RTS
                        ;
                        ;  MAIN LINE CODE, $2E79 TO $2F78:
                        ;       STORED AS A SECTOR OF 1 PAGE AT:
                        ;           TRACK 8, SECTOR 4 FOR 8" DISKS
                        ;           TRACK 12, SECTOR 4 FOR 5 1/4" DISKS
                        ;         ABOVE LOCATIONS FOR NORMAL OP. SYS.
                        ;
2EA9                    ;       *=$2EA9 ;CODE FROM $2E79 TO $2EA8 UNCHANGED
                        ;
2EA9 D003               ;       BNE ISGET
2EAB 4C8322             ;       JMP CLOSE ;IF PUT, DO A CLOSE (SAME THING).
2EAE C947               ISGET   CMP #'G ;IS IT A "GET"?
2EB0 F003               ;       BEQ GET
2EB2 4C1E0E             ERROUT  JMP ERR:
2EB5 206321             GET     JSR GETDEV
2EB8 20130E             ;       JSR ISCOMA ;IS THE NEXT CHAR A ","?
2EBB 20B90C             ;       JSR $0CB9 ;GET NUMBER FROM PROGRAM TEXT
2EBE 207216             ;       JSR $1672 ;CONVERT TO BINARY INTEGER
                        ;                 ;IN RANGE 0 TO 65535. IF OUT OF
                        ;                 ;RANGE, FC ERROR.
                        ;
                        ; GENERATE RELATIVE TRACK FROM RECORD REQUESTED
                        ;
2EC1 EA                 ;       NOP ;THESE 3 NOP'S ARE FOR LATER EXPANSION.
2EC2 EA                 ;       NOP
2EC3 EA                 ;       NOP
2EC4 205519             ;       JSR DIVIDE ;IF MATH SPEED-UP NOT INSTALLED
                        ;                  ;  THIS JSR MUST BE CHANGED TO
                        ;                  ;  POINT TO THE NEW LOCATION
                        ;                  ;  OF "DIVIDE".
                        ;
                        ; CONVERT TO DECIMAL RELATIVE TRACK
                        ;
2EC7 AD1E2E             DECTK1  LDA TEMP1
2ECA F00A               ;       BEQ ADDTK
2ECC F8                 ;       SED
2ECD 18                 ;       CLC
2ECE AA                 ;       TAX
2ECF A900               ;       LDA #00
2ED1 6901               DECTK2  ADC #01
2ED3 CA                 ;       DEX
2ED4 D0FB               ;       BNE DECTK2
```

This is the code for two random files.

The only BASIC commands that are altered are "DISKGET,N", and "DISKPUT".

"DISKGET,N" is now "DISKGET, D,N", where D is either 6 or 7. "DISKPUT" is now "DISKPUT, D", where D is 6 or 7.

The "GET" is also changed in that it now checks to see if the track is already in buffer before it does a call to the operating system for disk access. If the track is already in buffer, it skips the call and sets the pointers to the correct record.

If the desired track isn't in the buffer, then the dirty flag is checked, to see if the buffer has been written to since the last disk access. If it was written to, then the code performs a "DISKPUT,D" automatically before it reads in the new track.

This feature means that the "DISKPUT,D" command normally doesn't need to be used. Just remember to do a "DISKCLOSE,D" or a "DISKPUT,D" at the end of the program, to force a write of the last buffer back to disk.

The code is set up so that the record length can be POKED into the code at any time after the "OPEN", but before the first "GET". Unlike the O.S.I. version, it will accept any value from 0 to 65535. Only the values from 1 to the number of bytes per track are useful, however. Any other values will cause the math to be done wrong, and the code may hang up.

The records per track must be changed at the same time, to a value that is explained by the formula: INT((BYTES PER TRACK) / (RECORD LENGTH))

If both files are open, and they have different parameters, be sure to set the parameters to the correct values before doing any "GET"s on the file.

Since this code may have a record length of 1 byte POKED to it, byte addressing of the file is available.

When using byte addressing, be sure to set the records/track to the number of bytes/track for your system:

3328 FOR 13 PAGES / TRACK
3072 FOR 12 PAGES / TRACK
2048 FOR 8 PAGES / TRACK.

```
                ; ADD TO DEVICE START TRACK
                ;
2ED6 F8      ADDTK  SED
2ED7 AE8A22         LDX DEVICE
2EDA 18             CLC
2EDB 7D2A23         ADC FIRSTK,X
2EDE D8             CLD
2EDF DD2B23         CMP LASTK,X
2EE2 F002           BEQ ISSAME
2EE4 107D           BPL TOOBIG ;TRIED TO READ PAST END OF FILE
2EE6 DD2C23  ISSAME CMP THISTK,X ;ALREADY IN BUFFER?
2EE9 F03B           BEQ SKIPRD ;YES, SKIP THE READ
2EEB 8D202E         STA TEMP3 ;SAVE DESIRED TRACK
                ;
                ; SET UP FOR OS CALL
                ;
2EEE 20F72C         JSR SWAP01
                ;
                ; CHECK FOR DIRTY BUFFER, AND WRITE IF DIRTY
                ;
2EF1 AE8A22         LDX DEVICE
2EF4 EA             NOP ;THESE 3 NOP'S FOR FUTURE EXPANSION.
2EF5 EA             NOP
2EF6 EA             NOP
2EF7 BD2D23         LDA DIRTY,X
2EFA F003           BEQ READIT ;IF 00 THEN NOT DIRTY
2EFC 207724         JSR WRITEB
                ;
                ; DO THE READ OF THE PROPER TRACK
                ;
2EFF 205427  READIT JSR DNHEAD
2F02 AE8A22  HEADDN LDX DEVICE
2F05 BD2623         LDA LOBUFF,X ;GET START OF BUFFER AND
2F08 85FE           STA $FE      ;PUT IN TARGET POINTER.
2F0A BD2723         LDA HIBUFF,X
2F0D 85FF           STA $FF
2F0F AD202E         LDA TEMP3
2F12 9D2C23         STA THISTK,X
2F15 20BC26         JSR $26BC ;THIS POSITIONS DRIVE TO TRACK
2F18 A901           LDA #01   ;WE WANT SECTOR #1
2F1A 8D5E26         STA $265E
2F1D 206729         JSR DOREAD
2F20 206127         JSR UPHEAD
2F23 20F72C         JSR SWAP01 ;CLEAR FOR BASIC
                ;
                ; FIGURE RELATIVE RECORD IN TRACK
                ;   IT DOES MULTIPLE ADDS TO GET THE RELATIVE BYTE
                ;   POINTER, WHICH IS ADDED TO THE BUFFER START
                ;   ADDRESS TO OBTAIN THE ABSOLUTE BYTE POINTER.
                ;     THE POINTER OBTAINED IS STORED IN THE INPUT
                ;   POINTER OF THE PROPER DEVICE, THEN IT IS
                ;   TRANSFERRED TO THE OUTPUT POINTER OF THE PROPER
                ;   DEVICE.
                ;
                ;   ALL POINTER LOCATIONS ARE FIGURED RELATIVE TO
                ;   DEVICE #6'S INPUT POINTER.
                ;
2F26 A900    SKIPRD LDA #00
2F28 8D1E2E         STA TEMP1
2F2B 8D1F2E         STA TEMP2 ;ZERO ANSWER PRIOR TO CALL
2F2E 206118         JSR MULTIP ;IF MATH SPEED-UP NOT INSTALLED,
                ;                 THIS JSR MUST BE POINTED TO THE
                ;                 NEW LOCATION OF "MULTIP".
2F31 18             CLC
2F32 A000           LDY #00
2F34 AE8A22  SETPNT LDX DEVICE
2F37 F002           BEQ SET6
2F39 A051           LDY #$51
2F3B BD2623  SET6   LDA LOBUFF,X
2F3E 6D1E2E         ADC TEMP1
2F41 99AC23         STA POINT,Y
2F44 BD2723         LDA LOBUFF+1,X
2F47 6D1F2E         ADC TEMP2
2F4A 99AD23         STA POINT+1,Y
                ;
                ; MOVE THE INPUT POINTER TO THE OUTPUT POINTER
                ;
2F4D A017           LDY #$17
2F4F 8A             TXA
2F50 F004           BEQ DUPSET
2F52 A251           LDX #$51
```


JOE F. LINDEN, V.P.

# SOFTWARE TO ENHANCE YOUR COMPUTER's INTEGRITY

## FINALLY! A MULTIPLE USER SOFTWARE SYSTEM FOR THE O.S.I. C3-D COMPUTER!

We are proud to introduce to the OSI community our most recent and significant breakthrough in technical software advancements!
Our software system fills the MARKET GAP for a small, hard disk, multi-user computer.

Following is a brief description of the software standards. ::::

1. Supports all previously generated LEVEL 1 & 3 programs without special modifications.
2. Uses all current factory 65/U V.1.2 programming concepts and standards.
3. NO board CUTS or MODIFICATIONS ARE required for implementation.
4. IMMEDIATE expansion to 2 users in cabinet at present, with up to 15 via our EXPANSION BOX.
5. All corrections to the software have been installed, as listed below in our UPDATE PACKAGE.
6. The addition of a CA-18A and CM-11, along with multi-user implementation and connections is all that is required for a 2 user system with our Multiple User System Software.

Has been in-house tested since MAY, 81, with release in JUNE, 81. To date over 35 installations are using our software.

THE SINGLE USER LICENSE FEE IS - $500.00

## AT LAST! A CURE FOR OS-65/U V.1.2 LEVEL 1 & 3

This is the long awaited cure for those crash, stray writes and floppy disk stepping rate problems, with several UNIQUE ENHANCEMENTS.

OUR SOFTWARE UPDATE will ENHANCE and CORRECT all 26 currently known bugs and errors, of which the 8 most serious are listed as follows. ::::

1. LEVEL 3 STRAY WRITES are COMPLETELY ELIMINATED! - This has been documented and field tested for over 1 1/2 years, and has proven reliable since the fix was implemented.
2. DIREC* file size creation fixed.
3. Cntrl. `S` enhancement implemented.
4. Floppy disk stepper motor problems finally resolved.
5. CREATE file utility corrected for proper file rounding.
6. SHUGART SA-1000/4000 fault clear problem has been resolved!
7. Printer device 5, char. transfer speed now improved 40%!
8. Hard disk transfer speed to user memory now improved 6.3%.

THE SINGLE USER LICENSE FEE IS - $100.00

## SPECIALIZED PROGRAMMER ASSISTANCE UTILITIES

All of our software is self prompting and has been in operation from 6 mos. to 2 yrs.

1. HDFORM / HARD DISK header error, sector reformater. $75.00
2. RECOVR / Same as above, except for FLOPPY DISKS. $55.00
3. SECTST / A utility to test the headers of a floppy or hard disk for integrity! $55.00
4. DISAM1 / Level 3 compatible, MACHINE CODE DISASSEMBLER / ASSEMBLER! $200.00
5. L3COPY / Level 3 COMPATIBLE copier program. $150.00

All of our software and utilities are shipped ppd U.P.S. (USA), overseas frt. as applied.

DISTRIBUTOR, DEALER, OEM pricing available on request.

Keep your eyes open. soon to be released is our implementation of the `ELSE` and `PRINT USING` commands!

**PROFESSIONALS IN ELECTRONICS FOR 25 YEARS!**

**PHONE: (417) 782-1285 BOX 1446 402 WALL JOPLIN, MO 64801**

Since this code uses 2 byte math, only 65535 records are available. If you try to go past record 65535 with a "GET", you will get an FC ERROR, and be kicked out of the program.

* *** * ANNOUNCEMENTS * *** *

Electronic Courseware Systems, Inc., has developed a publication to aid teachers and administrators in planning and evaluating the use of computers in the classroom. The new publications, authored by G. David Peters and John M. Eddins, is entitled, A PLANNING GUIDE TO SUCCESSFUL COMPUTER INSTRUCTION. The book offers criteria and guidelines for assessing the available computer and microcomputer hardware and software for instructional use. Write to Electronic Courseware Systems, P.O. Box 2374, Station A, Champaign, IL 61820, for additional information. Cost of the publication is $19.95 with educational discounts to schools for multiple-copy purchases.

* * * * * * * * * * * * * * * *

* *** * NEW PRODUCTS * *** *

Micro... Publications in Review is a Monthly publication, a quick reference, of titles of article in the 70+ Micro-Mini Computer and Technology Publications. It has a magazine format and is intended to keep the reader abreast of this explosive industry through (1) reprints of the Table of Contents and (2) a Subject Index consisting of 26 major disciplines with each having from 6 to 40 classifications. Approximately 70 publications are covered with over 800 articles per issue.

Where possible, a fourth breakdown by computer and a fifth by language and/or operating system is classified. These later breakdowns will only appear in special issues.

Any title can be classified in one or two classifications and the breakdown is manually coded for computer indexing, since in most cases the title does not offer the necessary information for any keyword search.

For additional information, contact Vogeler Publishing Inc., 455 Crossen Ave., Elk Grove Village, IL 60007, 312/228-0951.

```
2F54 A06A           LDY #$6A
2F56 BDAC23  DUPSET LDA POINT,X
2F59 99AC23         STA POINT,Y
2F5C BDAD23         LDA POINT+1,X
2F5F 99AD23         STA POINT+1,Y
2F62 60             RTS ;ALL DONE, RETURN TO BASIC'S MAIN LOOP
2F63 4CD010  TOOBIG JMP ERR4
                    ;
                    ; THIS SUBROUTINE HIDES IN THE OPEN SPACE CREATED
                    ; BY THE MATH SPEEDUP.
                    ;
                    ; TWO MORE SMALL CHANGES HAVE TO BE MADE TO THE
                    ; MATH SPEEDUP TO LET THIS CODE FIT. THEY ARE:
                    ;
1845                      *=$1845
                    ;
1845 B018           BCS $185F
                    ;
18F6                      *=$18F6
                    ;
18F6 60             RTS
                    ;
                    ; THOSE TWO CHANGES ARE VERY IMPORTANT. YOU CAN
                    ; MAKE THEM, AND NOT INSTALL THIS CODE. BASIC WILL
                    ; STILL WORK OK. HOWEVER, YOU CANNOT INSTALL THIS
                    ; CODE WITHOUT MAKING THE TWO CHANGES ABOVE. BASIC
                    ; WILL NOT WORK RIGHT, IF YOU DO.
                    ;
1955                      *=$1955
                    ;
                    ; THE DIVIDE ROUTINE DOES IT'S MATH BY SUBTRACTING
                    ; UNTIL THE ANSWER IS LESS THAN 0. IT THEN RETURNS
                    ; THE LAST ANSWER GREATER THAN ZERO AS A REMAINDER,
                    ; AND THE NUMBER OF LOOPS AS THE QUOTIENT.
                    ;
1955 A900    DIVIDE LDA #00
1957 8D1E2E         STA TEMP1
195A 8D1F2E         STA TEMP2
195D 38             SEC
195E A519    DIV1   LDA $19
1960 E91A    DIVLO  SBC #26 ;RECORDS PER TRACK, CHANGE TO THE
                            ;CORRECT VALUE FOR YOUR SYSTEM:
                            ;  24 FOR 12 PAGES PER TRACK
                            ;  16 FOR  8 PAGES PER TRACK.
1962 A8             TAY
1963 A51A           LDA $1A
1965 E900    DIVHI  SBC #00 ;THIS IS ZERO FOR ALL DEFAULT
                            ;VALUES OF RECORDS PER TRACK.
1967 900F           BCC DIVDON
1969 851A           STA $1A
196B 8419           STY $19
196D EE1E2E         INC TEMP1
1970 D0EC           BNE DIV1
1972 EE1F2E         INC TEMP2
1975 38             SEC
1976 B0E6           BCS DIV1
1978 60      DIVDON RTS
                    ;
                    ; THIS FITS IN SOME MORE OPEN SPACE CREATED BY
                    ; THE MATH SPEEDUP.
                    ;
1861                      *=$1861
                    ;
                    ; THIS ROUTINE WORKS BY ADDING AND CHECKING THE
                    ; COUNTER TO SEE IF MORE NEEDS TO BE ADDED.
                    ; AT THE END, THE ANSWER IS IN TEMP1 AND TEMP2.
                    ; IF THE ANSWER IS MORE THAN 65535, A JUMP TO
                    ; THE BASIC ERROR "OV" IS MADE.
                    ;
1861 18      MULTIP CLC
1862 A41A           LDY $1A
1864 A619           LDX $19 ;GET REMAINDER FROM DIVISION
                            ;AS INDICES FOR ADD LOOP.
1866 9015           BCC MULT2
1868 88      MULT3  DEY
1869 18      MULT1  CLC
186A AD1E2E         LDA TEMP1
186D 6980    MULTLO ADC #128 ;THIS IS BYTES PER RECORD, LO PART
                            ;SAME FOR ALL DISK TRACK SIZES
                            ;(DEFAULT RECORD SIZE).
186F 8D1E2E         STA TEMP1
```

```
1872 AD1F2E          LDA TEMP2
1875 6900     MULTHI ADC #000 ;THIS IS BYTES PER RECORD. HI PART
                             ;SAME FOR ALL DISK TRACK SIZES
                             ;(DEFAULT RECORD SIZE).
1877 B0A6            BCS $181F ;THIS IS THE "OV" ERROR EXIT
1879 8D1F2E          STA TEMP2
187C CA              DEX
187D D0EA     MULT2  BNE MULT1
187F 98              TYA
1880 F002            BEQ MULTDN
1882 D0E4            BNE MULT3
1884 60       MULTDN RTS
                     .END
```

**AN INTERESTING DATA FILE HANDLER**

by Steve Brown
15856 Ocean Ave.
Whittier, CA 90604

The OS65D mini-floppy data file handling system has 3 major drawbacks:

1) It requires you to determine your data file needs before you even begin writing your program.

2) It takes up 2K of your workspace.

3) Reading in a record, even if only a few bytes, requires a full 2K disk I/O.

All of these drawbacks can be eliminated by using the simple basic routines given below. To use this system, you must write out 8 one-page sectors onto each track of the file. This can be done by a simple program like:

```
10 FOR I = 11897 TO 12152:
   POKE I,32: NEXT
20 FOR I = 11897 TO 12089 STEP
   64: POKE I, 13: NEXT
30 INPUT "START TRACK, END
   TRACK"; ST,ET
40 FOR T = ST TO ET: FOR S=
   1 TO 8
50 DISK!"SA"+STR$(T)+","+CHR$
   (S+48)+"=2E79/1"
60 NEXT S,T: END
```

The key to this data file handler is that it uses the directory buffer for data file I/O. This subroutine handles the I/O:

```
2000 S1=INT(RP/32): S2=INT
     ((RP-32*S1)/4)+1:
     T$=RIGHT$(STR$(S1+ST),2)
2010 S$=CHR$(S2+48)
2020 S3=11897*64*(RP-32*S1-4*
     S2+4): S1=S3 AND 255:
     S2=INT(S3/256)
2030 POKE 9098,S1: POKE 9099,
     S2: POKE9105,S1: POKE
     9106,S2
2040 DISK!"CA2E79="+T$+","+S$:
     RETURN
3000 DISK!"SA "+T$+","+S$+"=
     2E79/1": RETURN
```

where RP= record number ST= first track of the file

The data handling then is just like that for the OSI standard except that:

```
DISK OPEN,6,"FILE" --> ST=
                  first track of file
DISK GET, I --> RP=I: GOSUB
                  2000
INPUT#6,A$  --> INPUT#5,A$
PRINT#6,A$  --> PRINT#5,A$
DISK PUT    --> GOSUB 3000
```

There is only one problem with this system as written, while OSI 'says' that you can put up to eight sectors per track, in practice, their timing gets a little off. This can be corrected under OS65D V3.2 by a POKE 9851,49.

***NOTE 1: The example given is for fixed length 64 character records. The record length (RL) can be set to any power of 2 less than or equal to 256, (ie. 1,2,4,8,16,32,64, 128,256) by replacing the 32 in lines 2000 and 2020 by the number of records/track (32 --> 2048/RL). The 4 in lines 2000 and by the number of records per page (4 --> 256/RL), and the 64 in line 2020 by the record length (64 --> RL).

***NOTE 2: Records can be combined with intersperced carriage returns within a page to give variable length records. As an example, let us say that I wish to have a file of all 64 character records as above except I need one 90 character record followed by a 21 character record, and I need 20 consecutive 3 character records. I dedicate the first page of my file (records 0-3) to these records. The record layout would be:

```
0:(----------------------90
  character record----------
  ---------------)
1:(--------------CR---21 char.
  record---CR
2:(-1-CR-2-CR-3-CR.......
```

To read these records, use:

```
10 RP=0: GOSUB 2000: INPUT#5,
   A90$: INPUT#5,A21$
20 RP=2: GOSUB 2000: FOR I =
   0 TO 19: INPUT#5,A3$(I):
   NEXT
```

★

**THE ULTIMATE SCREEN CLEAR (FOR OS65D3)**

by Martin Ybarra
15856 Ocean Ave.
Whittier, CA. 90604

In almost any magazine, any article that you might find pertaining to OSI machines is probably some form of screen clear or another. Well, I have here what might be ultimate screen clear that OSI should have included in their BASIC. It is a relatively short machine code routine that is located in the middle of the stack. (I found some room there so I thought I would use it so that this code would not take up any actual user memory.)

I feel that this is one of the more versatile screen clears around, (others may think otherwise). First, I took out the BASIC keyword "NULL" and replaced it with "CLRS". However, the actual syntax is "CLRSx,y" where x ranges from 0 to 7 (actually it can go from 0 to 255). This will set the screen display similar to the POKE 56832,x. The second argument y ranges from 0 to 255 of which 0 through 15 will color the screen and anything above that will clear the video screen. Below is the listing of the program:

```
10 DATA 32,102,22,138,201,16,
   16,27,162,0,160,8
20 DATA 157,0,224,232,208,
   250,238,64,1,136
30 DATA 208,244,169,224,141,
   64,1,165,25
40 DATA 141,0,222,96,169,208,
   141,64,1,169,32,208,220
50 FOR X = 306 TO 349: READ
   M: POKE X,M: NEXT
60 POKE 709,ASC("C"): POKE
   710,ASC("L"): POKE 711,ASC
   ("R")
70 POKE 712,ASC("S")+128:
   POKE 546,49: POKE 547,1:
   CLRS1,16
```

Well, I hope this is the end of all those screen clear programs found in those magazines! As an added note: I may add another argument 'z' to home the cursor to the top of the screen and implement reverse scrolling in a future article.

★

**OS-65U V1.2, LEVEL 3**

by: Ron Mosley
Evergreen, CO 80439

1. User Number

The User Number has many uses, such as limiting program usage to a specific terminal or selecting a file for passing data from one program to another. I allow more than one user to pass data simultaneously, so I use separate scratch files named PASS0, PASS1, etc. The following code determines the user number and scratch file name in Level 1 or 3:

```
10 UN=0:REM Level 1 User #
20 IF PEEK(14948)=76 THEN UN=
   PEEK(55381) : REM Level 3
   User #
30 UN$=CHR$(48+UN) : REM Alpha
   User #
40 SF$="SCRAC"+UN$ : REM
   Scratch File Name
```

2. User-Defined Inputs

You have printed interesting discussions of fancy inputs, most recently by Dick McGuire in January, 1981. The goal of these efforts is simply to input a single keyboard character at a time with no screen echo. I don't use the WAIT instruction, because it hogs so much central processor time that Level 3 response time is unacceptably slow. My solution is to use the speedier interrupt-driven INPUT capability which is already available. Let's examine some useful applications which work under either Level 1 or 3.

First, you should disassemble or PEEK at part of your operating system to verify the following (all numbers are DECIMAL):

```
 DISASSEMBLY        PEEKs
                PEEK(1370)= 32
1370 JSR 1415   PEEK(1371)=135
                PEEK(1372)=  5

                PEEK(1407)= 32
1407 JSR 2798   PEEK(1408)=238
                PEEK(1409)= 10

1410 BNE -42    PEEK(1410)=208
                PEEK(1411)=214
```

If you have the 02/80 EDITOR enabled, verify the following:

```
                PEEK(1370)= 32
1370 JSR 23898  PEEK(1371)= 90
                PEEK(1372)= 93
```

If you have another EDITOR enabled, the following applications probably won't work.

The JSR 1415 accepts a keyboard character, the JSR 2798 echoes it to the screen, and the BNE -42 branches to get the next character. I hate having to press RETURN after single-key responses such as menu selections, Y or N INPUTs, etc., especially if I'm

entering a series of them. Responding to the first key pressed, with echo, is done thusly:

```
10 POKE 28,13 :REM Clear INPUT
   Buffer
30 POKE 1410,234:POKE 1411,
   234:REM Quit after 1st Char
40 INPUT X$ :REM Go Get it
50 POKE 1410,208:POKE 1411,214
   :REM Restore Multi-Char
```

If you have the 02/80 EDITOR enabled, add the following:

```
20 POKE 1371,135:POKE 1372,5
   :REM Kill EDITOR
60 POKE 1371, 90:POKE 1372,93
   :REM Revive EDITOR
```

This method accepts only those keys normally accepted by OS-65U. To disable the character echo, change the following lines:

```
30 POKE 1407,76:POKE 1408,132
   :POKE 1409,5 :REM JMP 1412
50 POKE 1407,32:POKE 1408,238
   :POKE 1409,10:REM JSR 2798
```

Now we'll try SECURE INPUT, which I always use for passwords to avoid compromising security. A spy must watch my flying fingers to learn the passwords INPUT as follows:

```
10 POKE 28,13 : REM see above
30 POKE 1407,76:POKE 1408,132:
   POKE 1409,5 :REM see above
40 POKE 2676,0: POKE 2683,0:
   REM Kill CR and LF
50 POKE 2794,16:POKE 2797,8 :
   REM Change "?" to ""
60 P$="" : REM Clear Password
70 INPUT X$ : REM Get a Char
80 IF X$="" GOTO 120 : REM
   GOTO if Done
90 P$=P$+X$ : REM Add to
   Password
100 PRINT CHR$(32+RND(0)*93);
    :REM Echo Random Char
110 GOTO 70 : REM Get another
    Char
120 POKE 2794,32: POKE 2797,63
    :REM Restore "?"
130 POKE 2676,13:POKE 2683,10
    : REM Restore CR and LF
140 POKE 1407,32: POKE 1408,
    238:POKE 1409,10 : REM
    see above
160 PRINT : REM Finish the
    Echo
```

Note that this method doesn't allow for spaces in the password. If you have the 02/80 EDITOR enabled, insert line 20 to Kill it and line 150 to Revive it.

3. Printer Paging

I often use a printer control character to advance paper to the next page, but some printers lack this capability, and the control code can vary from one printer to another. It is better to use the OS-65U paging capability. We should all memorize the following locations:

PEEK(14387) = Maximum # of lines per page, normally 66

PEEK(14457) = # of lines to print per page, normally 60

PEEK(15908) = # of lines left to be printed on this page

These locations apply only to device #5, which is a parallel printer, but I use the OSI Tech Memo #28,12 POKEs to make them apply to a serial device as well. You have probably seen the following paging method:

```
10 LP=PEEK(14457):LL=PEEK
   (15908)
20 IF LL<LP THEN FOR I=1 TO
   LL:PRINT#5:NEXTI
```

There is a much faster method, however. Note that OS-65U senses when PEEK(15908)=0 and automatically PRINTs enough blank lines to get to the top of the next page. Since a machine code is so much faster than BASIC, try the following to "trick" ol' 65U:

```
10 LP=PEEK(14457):LL=PEEK
   (15908)
20 IF LL<LP THEN POKE 14457,
   LP-LL+1:POKE 15908,1:
   PRINT#5
30 POKE 14457,LP:POKE 15908,LP
```

4. Level 3 Printer Paging

Since many of my reports can print a variable number of lines per page, I POKE 14457, 66:POKE15908,66 at the beginning, control my paging internally, then PRINT#5!:POKE 14457,60:POKE 15908,60 before exiting. Try an experiment -- put one terminal in a loop:

```
10 PRINT PEEK(15908):GOTO 10
```

Now load any BASIC program on another terminal and LIST #5. Watch the numbers displayed on the first terminal change. I was amazed when I first saw this. I thought that no user, without fancy coding, could change another user's partition. LEVEL 3 wants every user to know the paper position, but my programs all leave it at top-of-form, and I was POKing in the 66's before checking to see if the printer was busy, so I was messing up paging for the user who was printing. You can disable this feature like I did:

```
LOAD"LEVEL3"
verify that PEEK(24994)=141,
PEEK(24995)=36 and PEEK(24996)
     =62
POKE 24994,234:POKE 24995,234
     :POKE 24996,234
SAVE
```

5. Questions

Jim Sanders (March, 1981) discussed how to change the RUN "RTMON" which is executed when the Level 3 countdown runs down, but I can't make it work. I'd like to know how. Also, I'm looking for a good annotated disassembly of OS-65U, including Level 3, but I can't find Four-State Microcomputers, mentioned by Sanders in June, 1981. [See p.11 - A1].



★

**AN INTERESTING POKE**

For The C4P MF Working Under OS65D V3.2 Operating System

by: Corey Ostman
15856 Ocean Ave.
Whittier, CA 90604

I would like to share an interesting poke that I have discovered. The poke is: POKE 1382,0 (POKE 1382,32 is normal). What this poke does is to enable the following control characters:

```
CTRL J (CHR$(10)) : linefeed
CTRL H (CHR$(8))  : non-de-
                    structive
                    backspace
CTRL L (CHR$(12)) : non-de-
                    structive
                    forward
                    space
```

The control letters can be typed in from the keyboard in the immediate mode or even imbedded in BASIC statements such as REM, PRINT, and IN-

PUT"...". For real tricky programming, you can type several control H's and then space forward to hide whole lines! You will not be able to see the line, but BASIC will recognize it and execute it. Another example is to hide copyright notices or GOTO's to 'non-existent' lines. To be absolutely devious and downright mean, you could hide a counter in the program and after it reaches a certain number, have the program wipe itself out or even worse... INIT the disk! (But we're not like that, are we?)

I personally have used it to make listings clearer by erasing line numbers and the REM statements in my remarks. I have also sprinkled several CTRL J's for spacing or to print 'block' sentences. For example:

10 PRINT" HELLO THERE PEOPLE." Use control J to space down, several control H's to go left, type in something, another control J, etc.

One other possible use is to use a control letter in file names to lock your programs from other people! (Of course you will have to remember where and what control letter.) Perhaps someone will write a program to reveal control letters? Well, have fun with this one...

Warning!!! Be careful not to imbed control letters accidentally in a BASIC line. The program may bomb and you will have a very difficult time of debugging the program. To be on the safe side, repoke the location a 32 to put it back to normal operation.

★

## OSI-C1P/SAVE MEMORY WITH DATE STATEMENTS

by Harry Hawkins - 1981
P.O. Box 4432
Burton, SC 29902

```
9000 FORT=1TO20:?:NEXT:?"PGM
     MAKES A TAPE OF";?
9005 ?"SELECTED MEMORY,
     THAT";?
9010 ?"WHEN LOADED, WILL
     RUN,";?
9015 ?"LOAD THE DATA AND
     THEN";?
9020 ?"ERASE ITSELF, IF
     INPUT";?
9025 ?"IS HEX - PREFIX WITH
     H";?:?
9030 INPUT"START ADD";A$:IFASC
     (A$)<>72THENA=VAL(A$)
     :GOTO9050
9035 A$=MID$(A$,2):A=0
9040 X=ASC(A$)-48:IFX>9THENX=
     X-7
9045 A=A*16+X:A$=MID$(A$,2)
     :IFA$<>""THEN9040
9050 ?:INPUT"LAST ADD";A$:
     IFASC(A$)<>72THENB=VAL
     (A$):GOTO9070
9055 A$=MID$(A$,2):B=0
9060 X=ASC(A$)-48:IFX>9THENX=
     X-7
9065 B=B*16+X:A$=MID$(A$,2)
     :IFA$<>""THEN9060
9070 D=A:C=B-A:E=10:?:?"START
     TAPE! - TAP ESC!":?
9075 IFPEEK(57088)<>222THEN
     9075
9080 SAVE:?:?
9085 ?E;"POKE515,0":E=E+1
9090 ?:?E;"DATA";:E=E+1
9095 FORT=1TO10:?PEEK(A);
     :A=A+1
9100 IFA>BTHEN?:?:GOTO9125
9105 IFT=10THEN9115
9110 ?",";
9115 NEXT
9120 ?:GOTO9090
9125 ?E;"FORT=0TO";C;":READA
     :POKET+";D;",A:NEXT"
9130 ?E+1;"CLEAR:NEW"
9135 ?:?"RUN10"
9140 POKE517,0
```

This program, when run, prepares (saves) a BASIC Tape of any selected part of memory in the form of data statements; followed by a loop to read the data and poke it back to memory. When the Data Statement Tape is loaded, it will execute at the end of the load, poke the data to memory and then erase itself. Control is returned to the keyboard.

When the program is run, line 9080 executes SAVE. After the tape is prepared, line 9140 returns control to the keyboard.

At the end of the Data Statement tape load, RUN 10 (placed on the tape by program line 9135) executes RUN. Line 10 (created by program line 9085) takes the program out of load and last line (created by program line 9130) clears the BASIC program after the data statements have been poked to memory.

If run after load is not desired, delete program line 9135. If erase after data statement load is not desired, delete program line 9130.

The program may be shortened by removing the prompts, lines 9000 thru 9025. It may be further shortened by deleting the Hex to Decimal conversion. In this form the program would start at line 9060 with a "Start add" input request (variable A) followed

# digital technology

**BUS-II LEVEL I BOOKKEEPING & ACCOUNTING SYSTEM**
The BUS-II turn-key multi-client accounting package is the leading OSI business software package. BUS-II Version 3.1 includes five principle modules:

| | Inst. Price | Ref. Price |
|---|---|---|
| GENERAL LEDGER | $1200 | $599 |
| ACCOUNTS RECEIVABLE (a) | 1000 | 599 |
| ACCOUNTS PAYABLE (a) | 1000 | 599 |
| ORDER ENTRY W/ INVENTORY (a)(b) | 1000 | 599 |
| PAYROLL | 1200 | 799 |

The Accounts Receivable, Accounts Payable, and Order Entry W/Inventory are completely interactive with the BUS-II General Ledger. Two optional specialized packages (completely interactive) are also available.

CPA EXTENSIONS (see below)
POINT-OF-SALE TERMINAL W/INVENTORY (see below)

The BUS-II CPA EXTENSIONS Package provides special features for accountants and bookkeepers. The POS-1 Point-of-Sale Terminal package enables the operator to use the computer system's video terminal as an on-line "electronic cash register".

Note: BUS-II V 3.1 operates on floppy-disk or hard disk-based systems running the OS-65U V 1.2 operating system (LEVEL I, II, or III). Multi-client use can accommodate any number of client companies on floppy disk systems or hard disk systems with H/D/E (required for hard disk use). BUS-II LEVEL I files are limited in size for floppy disk back-up; floppy disk operation continues in case of hard disk failure.

**BUS-II LEVEL II ( EXPANSION TO BUS-II LEVEL I )**
BUS-II LEVEL II is designed for much larger businesses. Expanded file size and special operations allow virtually unlimited numbers of accounts and transactions. BUS-II LEVEL II requires BUS-II LEVEL I. Minimal back-up is data cassette (tape) or floppies—although multiple Winchester disk operation is recommended (provides ability to continue computerized bookkeeping functions in case of hard disk failure). H/D/E Hard Disk Executive is required.

| | Inst. Price | Ref. Price |
|---|---|---|
| GENERAL LEDGER | $ 600 | $ 399 |
| ACCOUNTS RECEIVABLE | 600 | 399 |
| ACCOUNTS PAYABLE | 600 | 399 |
| ORDER ENTRY W/ INVENTORY | 600 | 399 |

**POS-1 POINT-OF-SALE TERMINAL** (a)(b)
POS-1 is an on-line multi-store point-of-sale terminal program with integrated inventory designed for cash register emulation. POS-1 controls cash drawer and ticket printer (or system printer). Automates taxable or nontaxable sales, cash transactions, and credit sales (with verification operations). POS-1 is interactive with the BUS-II V 3.1 BOOKKEEPING & ACCOUNTING SYSTEM.

POS-1 Inst. Price $2400 Ref. Price $1199

**CPA EXTENSIONS PACKAGE** (a)
CPA EXTENSIONS is designed for public accounting firms. A number of special operations are provided: a "bankers" Balance Sheet and Profit and Loss statement with summarization and consolidation options, Statement of Changes in Financial Position, Statement of Changes in Components of Working Capital, Cash Flow Analysis, Departmentalized Sales Analysis, Comparative Income Statement, Budgetary Analysis, Asset Depreciation Schedule (compatible with TAXMAN-1040), and Loan Amortization Schedule. CPA EXTENSIONS is interactive with BUS-II V 3.1 BOOKKEEPING & ACCOUNTING SYSTEM.

CPA Extensions Inst. Price $ 3600 Ref. Price $1599

**TAXMAN-1040 PERSONAL INCOME TAX PREPARATION**
TAXMAN-1040 is designed for tax practitioners and public accountants. TAXMAN-1040 is the leading tax package for OSI microcomputers—the package has been installed on OSI, Hewlett-Packard, DEC and IBM systems. Designed and supported by CPA tax experts. This package automatically prepares FORM 1040 and 28 schedules. Individual state tax option available. Support includes annual forms and tax table revisions. Purchasers of 1980 TAXMAN will automatically receive 1981 revisions at no extra charge.

TAXMAN-1040 Inst. Price $3600 Ref. Price $2399

**TAXMAN-1120 CORPORATE INCOME TAX PREPARATION** (a)
TAXMAN-1120 (under development) is a corporate tax preparation package designed to work in conjunction with TAXMAN-1040 to provide full-service tax accounting functions. TAXMAN-1120 requires BUS-II G/L. Individual state tax option available; support includes annual forms and tax table revisions. Purchasers of 1980 TAXMAN will automatically receive 1981 revisions at no extra charge.

TAXMAN-1120 Inst. Price $3600 Ref. Price $2399

**OS-DMX DATABASE MANAGEMENT SYSTEM**
Command-oriented OS-DMS compatible database management system. OS-DMX operates under the OS-65U V 1.2 operating system (LEVEL I, II, or III). Features such as control files, extensive operating commands, and the innovative HELP feature, in addition to Digital Technology's exclusive on-line documentation, make this one of the most usable—as well as powerful—systems available for microcomputers. OS-DMX may be used instead of, or in addition to, OS-DMS Nucleus, Query, Sort; OS-DMX will replace virtually all of the specialized OS-DMS modules—and in most applications will provide greatly improved performance.

OS-DMX Inst. Price $2000 Ref. Price $1199

**ECR-1(P) ELECTRONIC CASH REGISTER POLLING** (c)
ECR-1(P) provides cash register polling and control (including cash register reprogramming) in conjunction with OSI microcomputers. Cash register polling is an alternative to on-line operation which allows the use of regular preset-total style electronic cash registers (with RS-232 communications). Versions are currently available for MKD BANTAM II and certain NCR cash register systems.

ECR-1(P) Inst. Price $1600 Ref. Price $799

**ECR-1(C) DATA CASSETTE POLLING** (c)
ECR-1(C) provides data cassette polling, allowing multi-store cash register polling. ECR-1(C) is recommended when diverse store locations make telephone line communications prohibitively expensive.

ECR-1(C) Inst. Price $1600 Ref. Price $799

**SALES-1 SALES ANALYSIS** (c)
SALES-1 is an OS-DMX-based sales analysis package for use in conjunction with OS-DMX, ECR-1(P), or ECR-1(C). Breakdown is provided by key-hit, family group, etc., indicating totals and percentages of sales. OS-DMX is required; ECR-1 is recommended; manual stand-alone operation is optional.

SALES-1 Inst. Price $1600 Ref. Price $799

**INV-1 RESTAURANT INVENTORY & MENU EXPLOSION** (c)(d)
INV-1, used in conjunction with OS-DMX, ECR-1 and SALES-1, provides a detailed breakdown of sales by family group and menu components. Provides managerial information detailing waste, pilferage, menu costs, stock levels, reorder levels, percentage-of-sales and percentage-of-cost from menu explosion. OS-DMX required; ECR-1 and SALES-1 recommended; manual stand-alone operation optional.

INV-1 Inst. Price $1600 Ref. Price $799

**H/D/E HARD DISK EXECUTIVE**
Digital Technology's implementation of H/D/E is the answer to AMCAP's HDM. Digital Technology's H/D/E provides user functions not found on HDM or similar products: ability to copy from any user "system" to another; automatic recovery in case of "back-up to floppy" or "restore from floppy" utility failures, allowing the user 3 options: (1) ignore error, (2) abort to menu, (3) try again; use of both "A" and "B" floppy drives to back-up hard disk files; and automatic back-up diskette initialization. H/D/E operates on any OSI Winchester disk system from 7 - 80 megabytes. Re-use of hard disk space is provided. Superior to AMCAP's hard disk manager in every respect (and Digital Technology software does not self-destruct). NOTE: H/D/E is required when installing any Digital Technology business applications packages on OSI hard disk systems.

H/D/E Inst. Price $800 Ref. Price $499

**H/D/M MULTI-USER MANAGER** (g)
H/D/M (under development) is Digital Technology's multi-user extensions to OS-65U. Replaces T-MUM by AMCAP. Need we say more?

H/D/M Inst. Price $1200 Ref. Price $499

**DIGITAL TECHNOLOGY, INC.** software is sold through OSI Dealers worldwide. For detailed product information call (714)270-2000. For the name of your nearest OSI Dealer call (toll free) OSI's "hot line" 1-800-321-6850.


digital technology
inc.
P.O. BOX 178590
SAN DIEGO, CA 92117
(714) 270-2000


**REQUIREMENTS**
(a) BUS-II LEVEL I G/L req'd
(b) BUS-II LEVEL I A/R req'd
(c) OS-DMX req'd
(d) ECR-1 recommended
(e) C3 CPU W/56K RAM & OS-CP/M or Lifeboat Associates CP/M req'd
(f) SYNCHRONOUS INTERFACE ASSY req'd
(g) H/D/E Required

by a "last address" request (variable B) at line 9065.

★

## CASSETTE LOADS OSI ASSEMBLER EDITOR IN 1:40 MIN.

by: Alex W. Jackson
1707 Providence Road
Towson, MD 21204

Two new utilities have come to the aid of the long suffering cassette user. Dwo Quong's D/Q Loader and Aardvark's Hi-Speed Loader.

First, the system on which we will test the loaders; it is a Superboard II, 1979 vintage. It has a 6502A Cpu running at 1.392 Mhz. (See PEEK (65) Dec. 1980, page 6.) The only other mod is the Aardvark 600 baud conversion, installed 1-1/2 years ago.

The cassette is a $40 Craig circa 1976. I did an azimuth (vertical) alignment on it a year ago. It is kept very clean and I use Radio Shack's head cleaning tape every week.

The tapes used for testing were first a Microfusion C-20 from Cook Labs. Norwalk, CT. (video tape, CrO2 composition). Second a new Micosette C-20 and last an old C-10 of indifferent quality. All tapes yielded no errors with either loader except when forced to verify error detection on read, i.e. reduce volume control.

O.k., lets start with D/Q Loader. It loads via the Monitor and yields good explicit prompts. A four page instruction sheet comes with the tape. The loader is in mach. lang. and occupies $1C00 to 1FFF. Apparently, it is not portable, but most importantly, could be converted to 600 baud format. It is therefore self reproducing.

Saving a short Basic source program will take as long as the Basic SAVE/LOAD because D/Q saves all of page zero and begins the recording with its own loading preamble. This short routine loads the main program and verifies no error.

A basic source of 5K or more will show a time improvement.

While recording or reading D/Q prints an asterick on the screen for each page (256 bytes) of code. D/Q allows 4 blocks of memory to be dumped for each recording.

This means if you have a hybrid program in BASIC with subroutines at say #0222 to 02FA and another in high memory above $2000 then D/Q will save it all at one time.

The last thing I tried was saving the OSI Assembler Editor at 600 baud. This is the poorest tape I have and it takes 7 minutes to load if nothing goes wrong. It was recorded as one block and took 1:50 minutes. I rewound the tape, did a cold start to sweep all memory, start tape, hit break, M, L in 1:40 m. "iniz" was on the screen. That's impressive.

Hi-Speed from Aardvark is a work horse of another color. It too will save and load BASIC and mach. lang. code. But Hi-Speed must reside in RAM to load its formatted programs. It requires more skill to use than D/Q, but its speed potential is double that of D/Q. Save/load in Hi-Speed format is significantly faster even at 300 baud.

Hi-Speed is a software/hardware modification. I removed my old 600 baud mod. and installed the new one which gives you selection of 600 or 1200 baud. You now get 300 baud by messaging the ACIA buffer. I read in my test program and sucessfully generated 300, 600 and 1200 baud tapes.

At this point I discovered a predictable incompatability between the new and old 600 baud recordings.

Anyone like myself who has made tapes with some other conversion, may not be able to read them on the Hi-Speed system. At 300 baud there will be no problem.

On the Super-II, I reinstalled the original conversion and added another select switch for 1200 baud only. This works very well.

If you buy Hi-Speed you can probably adapt the 1200 baud conversion to whatever you are using. On the other hand if your system is stock the conversion is simple, reliable and can cut load time from 4 minutes to 35 seconds.

Hi-Speed uses less than 400 bytes poked into upper memory and to some degree is portable. It has error detection for loading and will search for a file name, which means you can do continuous recording on one tape without the need for perfect queing.

Below are some time comparisons for the test program in minutes : seconds.

| 3-1/2 K BASIC | FACTORY | AARDVARK (1980) | AARDVARK (1981) |
|---|---|---|---|
| SOURCE CODE | 300 BAUD | 600 BAUD | 1200 BAUD |
| Basic Load/ Save | 4:00 | 2:10 | N/A |
| AARDVARK | ---- | 1:15 | 0:35 |
| HI-SPEED D Q LOADER | --- | 1:50 | 0:55 |

Both these loaders are recommended. For myself, I'll postpone building a floppy disk controller and enjoy writing some new software.

★ ★ ★

## VIDEO BOARD CHARACTER EDITOR

```
1011 REM THIS PROGRAM WILL HELP CREATE THE GRAPHIC CHARACTERS
     FOR
1012 REM  A VIDEO DISPLAY BOARD. THE CHARACTERS ARE STORED IN
     MEMORY
1013 REM  RECALLED FOR ADDITIONS OR CHANGES EACH CHARACTER IS
     CREATED
1014 REM USING ASTERISKS AND PRIODS TO INDICATE WHETHER A DOT
     IS
1015 REM TURNED ON OR NOT IN THE 8 X 8 DISPLAY GRID.
1016 REM
1020 HX$="0123456789ABCDEF"
1022 REM OS 65D ALLOWS DISK TO MEMORY TRANSFERS USING 'CALL'
1023 REM AND 'SAVE' INSTRUCTIONS
1025 INPUT"LOAD FROM TRACK 34 OR 35";TR$
1030 ST$="CALL 6800="+TR$+",1"
1035 DISK! ST$
1040 INPUT"BUFFER ADDRESS ";BA$:AD$=BA$:GOSUB 5000:BA=AD
1050 REM
1051 REM THE MEMORY BUFFER ADDRESS MAY BE ENTERED IN HEX OR
     DECIMAL.
1052 REM  IF IN HEX, SIMPLY BEGIN THE NUMBER SEQUENCE WITH A
     DOLLAR
1053 REM  SIGN '$'. (NOTE THAT THE DISK TRANSFER INSTRUCTION
     IN LINE
1054 REM  1030 & 1720 ARE WIRED FOR $6800.YOU CAN CHANGE THE
     PROGRAM TO
1055 REM ALLOW THE ADDRESS TYPED IN NOW TO BE USED IN THE
     'CALL' AND
1056 REM 'SAVE' INSTRUCTIONS.
1057 REM
1060 INPUT"CHARACTER OFFSET ADDRESS ";CO$:AD$=CO$:GOSUB
     5000:CO=AD
1089 REM 256 IS THE FLAG TO END THE INPUT PROCESS.
1100 IF CO=256 THEN 1700
1105 REM GO FETCH THE CHARACTER NOW IN THE REQUESTED MEMORY
     POSITION
1106 REM AND DISPLAY IT AS A GRAPHICS CHARACTER.
1110 GOSUB 3000
1120 PRINT:INPUT"EDIT OR ADD ";A$:IF A$="E" THEN 1200
1130 REM ALL EIGHT LINES OF THE CHARACTER WILL BE DISPLAYED.FOR
     EXAMPLE
1131 REM AN 'A' WILL LOOK LIKE THIS:
1132 REM ........
1133 REM ..*.....
1134 REM .*.*....
1135 REM *...*...
1136 REM *...*...
1137 REM *****...
1138 REM *...*...
1139 REM *...*...
1140 FOR I=1 TO 8:GOSUB 4000:NEXT I
1200 PRINT:PRINT"ENTER LINE TO EDIT":INPUT"(0=END) ";I
1220 IF I=0 THEN 1400
1240 GOSUB 4000:PRINT:PRINT
1250 PRINT
1260 FOR J=1 TO 8:PRINT"LINE";J,CH$(J):NEXT J:GOTO 1200
1390 REM USE THIS SUBROUTINE TO CONVERT THE CHARACTER STRING
     MATRIX
1391 REM TO 8 BYTES OF THE MEMORY BUFFER.
1400 FOR J=1 TO 8:N=0:FOR I=1 TO 8
1420 IF MID$(CH$(J),I,1)="*" THEN N=N+2^(8-I)
1440 NEXT I:POKE BA+CO*8+J-1,N
1460 NEXT J:GOTO 1060
1700 INPUT"SAVE ON TRACK 34 OR 35 ";TR$
1710 ST$="SAVE "
1720 ST$=ST$+TR$+",1=6800/8"
1730 DISK! ST$:END
2990 REM USE THIS SUBROUTINE TO CREATE A GRAPHICS CHARACTER
     FROM 8
2991 REM BYTES OF THE SELECTED MEMORY POSITION.
2999 REM CLAR DISPLAY BUFFER.
3000 FOR I=1 TO 8:CH$(I)=""
3009 REM CALCULATE 8 BYTE SLOT IN MEMORY BUFFER
3010 N=PEEK(BA+CO*8+I-1)
3020 FOR J=1 TO 8:N1=2^(J-1)
3039 REM IS DOT ON OR OFF?
3040 IF (N AND N1)=N1 THEN 3080
```

CONTINUED

## WHY USER GROUPS

by Al Peabody

We have just received a newsletter from a group called OSMOSUS, the Organization of Southeastern Minnesota Ohio Scientific Users. If there was ever any doubt in my mind about the value of User's Groups, this publication has eliminated it!

Actually, and pardon me for saying this, when I first saw the publication I didn't expect too much from it. I mean, how many OSI users could there be in Southeastern Minnesota, and what could they have to offer, really? Turns out, plenty.

Just take a look at what this little newsletter has to offer.

First of all, the latest business meeting was described. At this meeting:

a new ROM for the Epson MX-80, allowing italic characters and high-resolution plotting, was described;

the "group order of new modem kits" was discussed;

"the 2114 static RAMS and diskette cases were distributed";

a 'packet switching' board for amateur radio was described;

and a sign up sheet was started for group purchase of 2716 EPROMS.

The newsletter also discusses BASF minifloppy drives allowing two-sided, double density operation (a member is working on a controller board), mentions a demonstration of the new high-resolution video board, provides a program for an OS-65D directory utility allowing each disk to contain a volume identifier, and presents a list of the programs available on an OSI disk, plus a demonstration of the print styles available on the stock Epson printer.

This is obviously a very active users group, providing its members with group purchasing power, sharing software and knowledge, demonstrating hardware and giving mutual support and encouragement. If they can do it in Southeastern Minnesota, why not in YOUR town?

★

# OSI AARDVARK OSI
# NOW MEANS BUSINESS!

**WORD PROCESSING THE EASY WAY— WITH MAXI-PROS**

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

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

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

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

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

MAXI-PROS — $39.95

**NEW-NEW—NEW TINY COMPILER**

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

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

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

TINY COMPILER — $19.95 on tape or disk

**THE AARDVARK JOURNAL**

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

1) A tutorial on Machine Code for BASIC programmers.

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

3) Moving the Directory off track 12.

4) Listings for 20 game programs for the OSI.

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

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

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

**ACCOUNTS RECEIVABLE** — This program will handle up to 420 open accounts. It will age accounts, print invoices (including payment reminders) and give account totals. It can add automatic interest charges and warnings on late accounts, and can automatically provide and calculate volume discounts.

24K and 0S65D required, dual disks recommended. Specify system.

Accounts Receivable. $99.95

*** * * SPECIAL DEAL — NO LESS! * * ***

A complete business package for OSI small systems — (C1, C2, C4 or C8). Includes MAXI-PROS, GENERAL LEDGER, INVENTORY, PAYROLL AND ACCOUNTS RECEIVABLE — ALL THE PROGRAMS THE SMALL BUSINESS MAN NEEDS. $299.95

P.S. We're so confident of the quality of these programs that the documentation contains the programmer's home phone number!

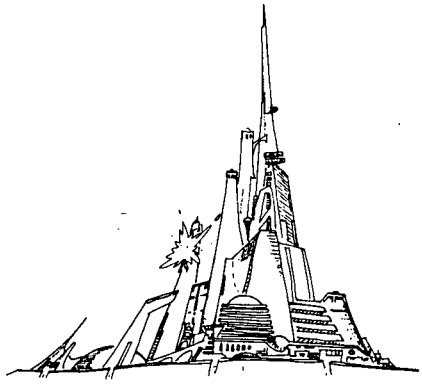
**SUPERDISK II**

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

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

It also has a full screen editor (in machine code on C2P/C4)) that makes corrections a snap. We'll also toss in renumbering and program search programs — and sell the whole thing for — SUPERDISK II $29.95 ( 5 1/4") $34.95 (8").

## AND FUN, TOO!



**BOOKKEEPING THE EASY WAY — WITH BUSINESS I**

Our business package 1 is a set of programs designed for the small businessman who does not have and does not need a full time accountant on his payroll.

This package is built around a **GENERAL LEDGER** program which records all transactions and which provides monthly, quarterly, annual, and year-to-date PROFIT AND LOSS statements. GENERAL LEDGER also provides for cash account balancing, provides a BALANCE SHEET and has modules for DEPRECIATION and LOAN ACCOUNT computation.

GENERAL LEDGER (and MODULES) $129.95.

**PAYROLL** is designed to interface with the GENERAL LEDGER. It will handle annual records on 30 employees with as many as 6 deductions per employee.

PAYROLL - $49.95.

**INVENTORY** is also designed to interface with the general ledger. This one will provide instant information on suppliers, initial cost and current value of your inventory. It also keeps track of the order points and date of last shipment.

INVENTORY - $59.95.

**GAMES FOR ALL SYSTEMS**

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

**NEW — NEW — NEW**

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

**NIGHT RIDER** - You've seen similar games in the arcades. You see a winding twisting road ahead as you try to make time and stay on the road. NIGHT RIDER uses machine code to generate excellent high speed graphics - by the same author as MINOS.

NIGHT RIDER — $12.95 cassette only

**THIEF** - Another machine code goody for the C1P cassette only. You must use mobile cannon to protect the valuable jewels in the middle of the screen from increasingly nasty and trigger happy thiefs. Fast action and fun for one or two players. THIEF $13.95 on C1 cassette only!

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

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

***Please specify system on all orders***

**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.**

VISA

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

MASTER CHARGE

OSI OSI

```
3059 REM DOT IS ON
3060 CH$(I)="."+CH$(I):GOTO 3100
3079 REM DOT IS OFF
3080 CH$(I)="*"+CH$(I)
3100 NEXT J:PRINT"LINE";I,CH$(I):NEXT I:RETURN
4000 PRINT:PRINT"LINE";I;"      ";:INPUT CH$(I):RETURN
4990 REM CONVERT HEX NUMER TO DECIMAL IF '$' LEADS STRING
5000 AD=0:IF LEFT$(AD$,1)<>"$" THEN 5100
5020 AD$=RIGHT$(AD$,LEN(AD$)-1)
5040 FOR I=LEN(AD$) TO 1 STEP-1:FOR J=1 TO 16
5060 IF MID$(AD$,I,1)=MID$(HX$,J,1) THEN 5080
5070 NEXT J:STOP
5080 AD=AD+(J-1)*16^(LEN(AD$)-I)
5090 NEXT I:RETURN
5100 AD=VAL(AD$):RETURN
```

# LETTERS

ED:

I found another interesting idea to put into DMS 9/79, which I would like to share with my fellow OSIers. It's not utopia, but it's better than nothing. In OSI Tech. Note 15, there's some mention of a patch one can install in OS65U Systems disk so that one can output 'today's date', on, let's say a Directory, or for that matter in any program where this is needed. I undertook the task a few days ago, to implement this in DMS 9/79, where I really need it. Of course, one will have to create a field for it in the records, e.g. DATE with a length of 8 characters, e.g. MM/DD/YY including the slashes. I said it's not utopia, because I am not experienced enough to write a routine whereby the date would automatically append to each record that one makes a change on, but in my case, something will have to be input into that field. I have programmed it, so I can input a backslash (\), and the date will automatically fill that field. Saves a lot of typing.

Add these lines to BEXEC* and make these changes in EDMFL

ADD TO BEXEC*:

```
221 REM GET DATE
222 PRINT: INPUT "MM,DD,YY";
    DM$,DD$,DY$: PRINT
223 DM=VAL(DM$):DD=VAL(DD$):
    DY=VAL(DY$)
224 IF DM<1 OR DM>12 OR DM<>
    INT(DM) GOTO 222
225 IF DD<1 OR DD>31 OR DD<>
    INT(DD) GOTO 222
226 IF DY<1 OR DY<>INT(DY)
    GOTO 222
227 POKE 24569,DD:POKE24570,
    DM:POKE 24571,DY
228 GOTO 280

1051 IF A$<"3" OR A$>"4" THEN
     GOTO 1050
1052 IF A$="3" THEN GOSUB 1550
1053 IF A$="4" THEN GOSUB 1520
1055 REM
1056 REM
1060 REM
1065 REM
```

CHANGES TO EDMAFL

```
1130 INPUT S$:GOSUB5500
1131 IF S$="*C"ORS$="*P"GOTO
     62550
1132 IF S$=CHR$(92) THEN GOTO
     11080
1133 IF S$<>"*R" GOTO 1142
1134 S$="^P"
1135 INDEX<K1>=BODF+((RPTR-K1)
     *RL)
1136 IF LEN(S$)=FL(K1) GOTO
     1140
1138 S$=LEFT$(SP$,FL(K1)-LEN
     (S$))+S$
1140 PRINT%K1,S$


11080 A=24569:DT$=RIGHT$(STR$
      (PEEK(A+1)+100),2)+"/"
11090 REM
11100 DT$=DT$+RIGHT$(STR$(PEEK
      (A)+100),2)+"/"
11110 REM
11120 DT$=DT$+RIGHT$(STR$(PEEK
      (A+2)+100),2)
11130 REM
11140 REM ABOVE LINES 11080-
      11120 GETS DATE OUT OF
      MEMORY
11150 IF S$=CHR$(92) THEN
      S$=DT$
11160 GOTO 1133
```

Fred S. Schaeffer
Jamaica, NY 11435

Fred:

Good for you! This is certainly a quick way to enter the date a record was last edited.

A couple of comments are in order:

1. Your routine in EDMAFL re-PEEKs the date and reconstructs DT$ each time you enter the "\". Why not have it do that once, early in the program? Then line 1132 could just be what is now line 11150, and the trip to 11080 could be avoided.

2. EDMAFL keeps track of the field labels as FD$(FP). Just before line 1130 you could put, 1129 IF FD$(FP)="EDIT DATE" THEN S$=DT$:GOTO 1142, and it would fill it in for you! All you have to do is put the field "EDIT DATE" in your master file when you create it, with the proper field length. My EDMAFL even does the creation of DT$ around line 540 or so.

Al

* * * * *

ED:

You asked us to identify ourselves, so I am responding. I don't know if there are many OSI users in my position, but if there are, I would like to correspond with them. I am a physics graduate student doing experimental research at New York University. Our lab bought a C4P DF to monitor and control some of the equipment in our lab. I have been assigned the tasl of interfacing the computer with the apparatus using the CA21 Head End Card and the CA20 board. If there are any PEEK'ers who are using a Challenger (they got that name right) for data acquisition and analysis or other laboratory related roles, I would like to share experiences with them. I can be reached through the Physics Dept., New York University, 4 Washington PL.

Bill Fowlkes
New York, NY 10003

* * * * *

ED:

The article on screen formatting by Ken Holt was excellent and has given me many ideas for my own programs. I would like to make one suggestion which might improve his routine. He defines four screen functions on lines 19900ff using subroutines. However, one could replace those subroutines with string variables and both simplify and speed up the program. For example, on line 19210 of his program a clear screen and GOTO position 0,0 is requested using FU=3 :X=0: Y=0: GOSUB19900. Supposing instead that SP$ was defined at the beginning of the program as SP$=CHR$(12) +CHR$(20). Then line 19210 could be written PRINT SP$CHR$ (0)CHR$(0);. Using this technique, complex screen formatting functions can be generated with quite simple statements. Moreover, the string variables can be combined. Let T0$=CHR$(8)+CHR$(0). You could then create a new variable to position the cursor at the beginning of the eighth

line say, by letting T8$=SP$ +T0$ for repeated use or simply write PRINT SP$T0$. Finally, the string definitions can be isolated from the body of the program as Ken did to allow straightforward modification for different terminals. For example SP$ would be defined as CHR$(26)+CHR$ (61)+CHR$(41) for the Televideo 920. But again, congratulations to PEEK (65) and to Ken for the increasing quality of your journal articles.

Michael Anderson
Arlington, VA 22204

* * * * *

ED:

Several comments on the October issue of PEEK: RE: Your two line solution to the input menu problem GOTO 100*ASC(C$) is known as a computed goto. This is common in Fortran, but not allowed in the several versions of BASIC I know of. Does this REALLY work on your computer??? [Yep... Al (C3D).]

RE: The letter from the dramamine using reader with the "swimming" video display. First a short course in television theory: A television outputs 60 frames per second, each frame is made up of 262-1/2 lines. Thus the horizontal frequency is 60 * 262-1/2 = 15,750. The horizontal and vertical frequencies are related by the magic number 262-1/2.

OSI video boards output 256 lines per frame. This happens because 256 is a nice even power of two. However, if the frame rate is 60 per second, the horizontal frequency is now 60 * 256 = 15,360. The horizontal hold on most B&W sets can be adjusted far enough to lock in on this frequency.

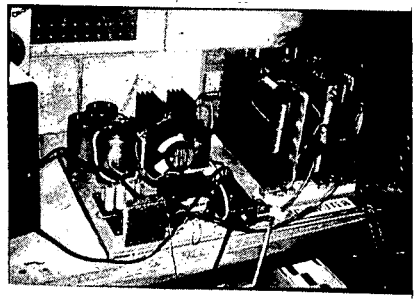
Color television is less tolerant of changes in horizontal frequency because of the additional color information carried just before and after the sync pulse. On color video boards, OSI makes the horizontal frequency exactly correct at 15,750. However, because of the non-standard number of lines per frame, the vertical frequency is 15,750/ 256 = 61.5 frames per second.

The 61.5 frequency beats with 60 cps line frequency to give a resulting "swimming" of the picture at 1.5 beats per second.

The solution is to beef up the power supply in the monitor with more capacitors and perhaps a better voltage regulator. Many televisions have no regulator, but the picture appears steady since the frame rate exactly matches the power supply frequency.

Another solution is to replace the crystal on your video board to produce the 60cps frame rate. Your 540 board will have either a 11.79 or a 12.08 MHz crystal. The crystal frequency is divided by 196,608 ( = 3 * 2^16) to get the vertical frequency. You can do the division to see which crystal gives the proper vertical frequency.

RE: Your comment that PEEK is for all machines which say "Ohio Scientific" on the front. Enclosed is a photo of my computer. Since there is no OSI logo on the front, does that mean I am no longer allowed to subscribe to PEEK?



E. Morris
Midland, MI 48640

P.S. You have no doubt, heard the expression "single board computer" which in my case refers to a piece of plywood.

Earl:

We had a "board" meeting and voted 5-4 to allow you to continue subscribing. Seriously, I like your style!

Al

* * * * *

ED:

We work under 65U Level 3, Version 1.3. We have been working under the Four State's version 1.2 of 65U but the availability of version 1.3 with the ability to transfer data from one program to another and kill inactive arrays made it advantageous to change. (The Four State's version is beautifully done, a real professional job.)

We continue to use the old 'FILLER' program as well as the version 1.3 'INP$' to control field lengths in files. It is easier to include it in the 'NUT' program from which we write; and allows the 'EDITOR' to be on tap at the same time.

In using version 1.3 of 65U, we have had some trouble with the 'wait for / wait clear' commands for access control to the data files. If the 'wait clear' is held until a second program is called, it results in a 'FC' error. It apparently must be used in the same program as the originating 'wait for'. Note that the programs concerned have run for over a year under 65U (versions 1.2 and earlier) with no problem.

We found that the version 1.3 system would not give us backspace control on our ('TEC' and 'Micro-Term Act V') terminals unless a poke of 255 to location 233 was added to the 'EDITOR'. (Yes, we did add the proper controls to 'CRT 0' for the terminal set up for the 'TEC' and 'Micro-Term' terminals.)

One useful trick under any version of 65U, the index of a channel is changed (to '0') only by a reset or a 'OPEN' to that channel. This provides a cheap and dirty way to carry a few data items from one program to another. e.g. let program 1 say 'INDEX<8>=99999: RUN''PRGRM2'' ', program 2 can then say 'I9 = INDEX(8)' and whatdoyouknow I9 is equal to '99999'. This permits carrying up to eight (nine digit) integers from one program to another. A typical use would be to have the time out program go to the line number specified by 'I9' so that 'RTMON' could provide the desired action dependent on the calling program.

C.L. Richards
Indianapolis, IN 46260

## AD$

# OS 65D V3.2 DISASSEMBLY MANUAL

If you've already spent more frustrating hours trying to modify 65D than you'd like to remember, here's help . . . the OS 65D V3.2 DISASSEMBLY MANUAL from Software Consultants.

It's all in one complete document . . . a detailed breakdown of the version 3.2 OS that's packed with the specifics you need to get the OS working for you, not against you.

With our manual, you can customize 65D to meet your own special needs, and not have to hope you're doing it right. Now you can use everything that's there to tie in your own machine language routines. The hundred of hours we spent unscrambling OSI's machine code lets you really use this "developmental" system the way you need it most.

## *FEATURES*

- 50 PAGES of Disassembly listing in standard Assembler format.
- FULLY COMMENTED CODE—Not just another useless program listing, but real explanations of what's going on.
- ALL ROUTINES INCLUDED. Details like:
  - —OSI print routines
  - —Video output routines
  - —Polled keyboard routines
  - —Check input
  - —Dispatch tables
  - —Commands, etc.

## *PLUS,*

- 10 PAGES of Cross Reference Listings—Keyed to Disassembly listing so you can quickly find any specific variable or routine, and, where they are used. A complete, computer-generated concordance.

The Disassembly was written for 8" disks. If you plan to use it with 5-1/4s, please let us know when you order, and we'll include a list of changes at no extra charge.

*PRICE:* **$25.95**

## *OTHER GREAT PRODUCTS*

REF COMMAND UNDER BASIC
Lists line numbers, variables, constants for 65D or 65U. $31.95

SPOOLER/DESPOOLER UTILITY
Super fast. Frees up screen, feeds data to serial or parallel printers. $69.95

FIG FORTH UNDER OS-65U
Runs under multi-user, hard disk systems with all the extras. $89.95

VIDEO ROUTINE
Convenient control of variable screen parameters. May be connected to graphics resolution booster. $25.95 alone. With extensions, $29.95.

GRAPHICS RESOLUTION BOOSTER
Hardware to boost screen resolution by 8 times to 128 x 128. $49.95 alone. With video routine and software extensions, $79.95.

Orders shipped postpaid from Memphis. Foreign orders please add $10 postage fee. Source code for purchased products is available for $12 each.

Dealer inquiries welcome.

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

SOFTWARE CONSULTANTS
HARDWORKING *software*

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

**PEEK (65)**
The Unofficial OSI Users Journal

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

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

**DELIVER TO:**