PEEK (65)

The Unofficial OSI Users Journal

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

$1.75

AUGUST 1984
VOL.5, NO.8

INSIDE

Column One

This must be the "Part Two" issue! This month's issue contains Part Two of no less than four ongoing series of articles - each an in depth presentation with which you should be conversant.

As new owners of old machines join the ranks and even old machines are again being brought out of the closet, our phones ring more frequently with questions that the old hands take for granted. Thus we are grateful to Leo Jankowski for his clear and comprehendible approach in his Beginner's Corner contributions.

At the other end of the spectrum are Leroy Erickson's ROM routines and Rick Trethewey's Assembly Language Classes. These two really go together to open new and unlimited doors to complete utilization of OSI machines.

Then there is Ken Shacter's continued review of DOS/65. Now, there's a really complete and thorough review!

For the Big Boys, at long last, instructions on how to add that second hard disk and a review of a powerful data base.

With all of this great material in one issue, please don't get the idea that we don't need more of your articles and letters. We're keeping a list, and checking it twice, of all subscribers who have not sent us a little something!!

So much for tooting our horn. It's time to remind you of your chance to toot yours! The Software Issue is just around the corner. Get your free listings in soon. Remind your friends, your dealer, or anyone else you can think of who might have a program! The deadline is September 1st - a few short weeks away.

PEEK(65) subscriptions fell somewhat during the last takeover of OSI. While we have made several promotional attempts, which have certainly helped, we all know full well that volume is the name of the game.

What is PEEK's worth to you? One bug fix, a problem solved, etc. is surely worth the cost of a year's subscription.

Believe it or not, there are even those who enfringe upon our copyright and actually make copies. Others "share" a copy!

Do yourself and us a favor. Get your own subscription!!

Also, if you can bring a new subscriber to PEEK, we will extend your subscription one month. In order for you to receive your extension, the NEW subscriber must, of course, supply us with your name and address as it appears on your label, or better still, send a copy of your label.

We have word that the ISOTRON dealer sales contest is over, and the ten winning and hard working firms each get to send someone on a Swedish Spree for two weeks at ISOTRON's expense. We thought you might like to know who is selling so many machines, so many in fact, that ISOTRON almost got behind in filling orders last month. In the Master Dealer category: Fial Computer, Micro World, and Puerto Rico Computer Sales. OEM category: Frank Thornber Co., Medical Business Systems, and Beinfeld Computer Systems. Dealer category: IBA Computer Systems, The Exchange and Computer Applications. Indirect Dealer category: Computer Systems. The folks at ISOTRON tell us that the competition was very keen and went right down to the wire. That means there were a number of others who came in a close second, which in turn means that ISOTRON is doing well and that's good for all of us.

In the same vein, we are trying our darndest to secure a current and up-to-date list of dealers. We feel strongly that you, our subscribers, need to know who the new dealers are if you are to get the support that you deserve, and equally so that these dealers will be supported by our subscribers.

Eddie

DOS/65 REVIEW

PART TWO

By: K.B. Shacter
P.O. Box 61000
New Orleans, LA 70161

BASIC-E/65

The BASIC that comes with DOS/65 is of the compiled/ interpreted variety. By this, I mean the BASIC source is compiled to an intermediate stage, which is then executed by a run-time interpreter. This type of arrangement has the advantage of providing run-time error messages, while sacrificing the ultimate speed of a true compiled source to machine language. Another usual advantage of using a compiled/interpreted language (such as the P-system Pascal) is the compiler and other programs can be written in the high-level language, while only the run-time interpreter must be in native-code to the processor being used. However, this is not the case here, as both the run-time interpreter and the compiler are written in 6502 assembly language. This compiler/ interpreter BASIC does, however, result in smaller run-files, allowing more (or larger) programs to be stored on disk than a true compiled object module.

The compiler supports the standard OSI/Microsoft BASIC key-words except those dealing with interactive debugging, editing, high-speed I/O servicing, and a few others. These are: CLEAR, CONT, LIST, LOAD, NEW, SAVE, and WAIT. The USR function is replaced by CALL (address of ML program); OPEN by FILE; SPC(X) by TAB(POS(0)+X); and PUT and GET by variations of PRINT and READ. In addition, the Boolean exclusive-or (XOR) is provided, as well as the logical/ relational operators LT, LE, GT, GE, EQ, and NE. The old stand by mathematic syntax (<, <=, >, >=,=, and <>) are also supported.

Variable names can be up to 31 characters long (that is quite a bit), and line numbers are optional except when the target of a branch (GOTO or subroutine (GOSUB). The length of the variable name can now make programs self documenting, as long as you use names that mean something. Functions can be either alpha or numeric, and multiple dummy arguments are supported. For example:

```
FN.ADJUST$(STRING$) = . . .
FN.HYPOTENUCE(X, Y) = SQR( X *
X + Y * Y )
```

The IF command has been expanded from:

```
     IF  operand  THEN  object

to:  IF  operand  THEN  object
     ELSE  object

and: IF  END#n    THEN  object
```

The last variation is used to detect an end-of-file condition on a disk read and take appropriate action.

I am disappointed an IF/ON ERROR command was not provided, as well as PRINT AT, PRINT USING, HEX$, RESTORE #, OR KEY. The DOS does provide for a KEYtype function via a CALL from BASIC though. The normal RESTORE feature is available (i.e., reset the DATA pointer to the first DATA statement), but the user cannot direct the pointer to the beginning of a particular DATA statement (as in the North Star BASIC).

PRINT# is available as direction to disk files, but not to the printer. The printer function is handled by another CALL. The RND function is more conventional, with optional use of the RANDOMIZE statement first to reseed the random number generator. Finally, BASIC-E/65 supports the hyperbolic sine function (why?), but not any of the other hyperbolic functions. These can be easily accommodated, however, via FN calls and the EXP function. One may reference any calculus book or standard math table compilation for the hyperbolic function formulae. It is interesting to note the manual (version 2.0) does not reference or explain the EXP function, but it is supported. BASIC-E/65 handles logical operations like Microsoft BASIC, where TRUE = -1 and FALSE = 0. NOT TRUE = 0, and NOT FALSE = -1.

There is no easy way to load a machine language (ML) routine using BASIC-E/65. One must use the DCB and PEM calls (in the manual, but far from detailed) and reserve room for the ML program using BASIC. There is an example of how to reserve room in the BASIC manual, but it stops short of showing how to bring the ML program into memory for execution.

In dealing with ML routines, there is a good section on how to pass variables to and from BASIC. PEM and SIM have calls set-up which appear to make using ML routines (once you get ahold of them) a snap. One nice feature of the compiler is multiple line statements can be used to provide clearer code. The continuation is triggered by a back-slash (SHIFT-L) as the last character on the line, as follows:

```
IF FLAG EQ ANGLE  THEN    \
   BETA = ALPHA / THETA   \
ELSE                      \
   BETA = ALPHA * THETA :\
   FLAG = 0
```

Of course, the 65D commands EXIT, DISK and DISK! do not apply.

BASIC-E/65 performs all math functions in 32-bit floating point; a 24-bit mantissa with an 8-bit exponent. As such, the integer notation of 65D (%) is not supported.

One always hears rumors about compilers being super-fast. Well, BASIC-E/65 is fast . . . at some things. My benchmarks, while not complicated, show you should not switch from OS65D or HEXDOS to BASIC-E/65 to gain great speed in your BASIC programs. The comparisons below are for timing loops run on an OSI C1P running at 1 MHz. Times are given in seconds, except for problem 6, which is in minutes: seconds. All times are also normalized within a sample to 65D results. Next to the actual timed results are relative speed comparisons, normalized to 65D results. If the number is followed by the letter 'S', that particular run is x-times slower than the corresponding 65D run. The letter 'F' means it was faster. The following is a description of the problems.

```
1. Constant Loop   I=1 TO 1000
2. Variable Loop   I=J to K
              (J=1, K=1000)
3. # 2 plus        X=I / I
4. # 2 plus        X=I + I
5. # 2 plus        X=I * I
6. # 2 plus        X=SQR( I )
7. # 2 plus        X=I / 1000
```

Copyright ©1984 by PEEK (65) Inc. All Rights Reserved.
published monthly

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

Subscription Rates

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

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

For back issues, subscriptions, change of address or other information, write to:

PEEK (65)
P.O. Box 347
Owings Mills, MD 21117

Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsements of the product or products by this magazine or the publisher.

8. # 2 plus X=I / K
M7. # 7 per manual timing
M8. # 8 per manual timing

The headings on the columns are:

BIR = BASIC-in-ROM
HXD = HEXDOS 4.Ø
65D = OS65D 3.1
/65 = DOS/65

The MICROSOFT numbers under the 65D column are values from the user's manual. They represent results from problems 7 and 8 as run on a system using a MICROSOFT interpreter. The numbers next to these under the /65 column are values from the manual also, same problems, run using BASIC-E/65.
The MICROSOFT numbers under the 65D column are values from the user's manual. They represent results from problems 7 and 8 as run on a system using a MICROSOFT interpreter. The numbers next to these under the /65 column are values from the manual, same problems, run using BASIC-E/65.

ment, BASIC-E/65 may win hands down. Certainly on the SQR(X) function, 65D can't even compare. The enhanced capabilities of BASIC-E/65 make it a pleasure to code with, and I couldn't get string error (ala BASIC-in-ROM) with multiple concatenations. One utility provided with the DOS (FILESTAT.BAS) does give an out-of-memory error after checking about 7 or so disks, but I have not been able to determine the cause of this problem.

Unlike most basic interpreters, BASIC-E/65 provides real error messages. By this, I mean you get an understandable (for the most part) English message that tells you what went wrong. These error messages are provided during the compile phase as well as the run stage. The compiler has several options which can be used to provide various actions. The options are:

1. List the compile code (not

prob	--- BIR ---			--- HDX ---			--- 65D ---			--- /65 ---		
1	:13	1.23	F	:15	1.Ø7	F	:16	1.ØØ	-	:34	2.13	S
2	:13	1.23	F	:15	1.Ø7	F	:16	1.ØØ	-	:36	2.25	S
3	:47	1.21	F	:5Ø	1.14	F	:57	1.ØØ	-	:66	1.16	S
4	:35	1.17	F	:39	1.Ø5	F	:41	1.ØØ	-	:55	1.34	S
5	:45	1.42	F	:5Ø	1.28	F	:64	1.ØØ	-	:65	1.Ø2	S
6	5:39	2.61	F	5:44	2.57	F	14:45	1.ØØ	-	6:3Ø	2.27	F
7	:84	1.27	F	:92	1.16	F	:1Ø7	1.ØØ	-	:68	1.57	F
8	:52	1.23	F	:58	1.1Ø	F	:64	1.ØØ	-	:71	1.11	F
M7	-	-		MICROSOFT --->			:96	1.11	F	:67	1.6Ø	F
M8	-	-		MICROSOFT --->			:56	1.14	F	:7Ø	1.Ø9	S

As can be seen from the results, for speed, nothing beats BASIC-IN-ROM. HEXDOS comes in a close second. HEXDOS is slower due to a little extra line parsing it must do, and the interrupt capability in controlling the disk motor and head load. When comparing 65D or DOS/65 to the ROM BASICs, one must remember the disk BASICs use longer 'words' for representing numbers in memory. Thus, one would expect slightly slower executions. Examining the results, it is fairly obvious something is wrong with the 65D BASIC SQR(X) function. As I did not check other functions (e.g., EXP, SIN, PEEK, etc.), I cannot vouch for their speed either. BASIC-E/65 does not win any world speed records on your standard loops with simple mathematical operations. Why this is I don't know. I would have thought the compiled BASIC would be at least as fast (if not slightly faster) than interpreted BASIC. As can be seen from the sample problems, this is not necessarily so. In a more real-world environ-

normally useful for the general user).

2. List only lines with errors.

3. Do not produce an .INT file during the compilation. The .INT file is the intermediate compiled code that is executed by the run-time package. The compiler reviewed did open an entry in the directory for the .INT file, but did not place any code into it.

4. Do not convert all lower-case letters outside of strings to upper-case. This is useful for variable names that are the same as BASIC key-words, but are in lower-case.

5. Include code in the .INT file to list a line number along with the error messages. This option did not work in the compiler as received. Shortly after returning the system to PEEK(65), I received a sheet of paper stating what fixes were required to correct the observed non-compliance with the user's manual. I cannot state if the fix worked, but it appeared easy enough to install.

6. Send the compile listing to the printer instead of the video monitor. As of the time of testing the DOS, I did not have a printer, so I was unable to verify this option. The same task can be accomplished by not invoking option 6, but using the cntl-P capability of CCM.

I have noted certain options tend to conflict with each other, with one of the chosen options not being invoked.

I found using the compiled BASIC an enjoyable experience. However, the nature of the personal computer for developing and debugging BASIC programs should be interactive. I lost the sense of intimacy, feeling detached, and found debugging (especially without a printer) to be more tedious than that encountered using an interpreted BASIC. Using BASIC -E/65, if an error occurs, whether syntax or logic, you cannot immediately list the program and the values of the variables in question. You must first load the editor, change the source to insert your debug statements (if the error is not obvious), re-compile, and then re-execute. I prefer the immediacy of the standard OSI BASIC (with all its' shortcomings, when compared to the power of BASIC-E/65) for program developments. It would be nice if you could use OSI BASIC to scope out a program, and then change it over to DOS/65 BASIC -E/65. However, to fully take advantage of the power of BASIC-E/65, you may have a lot of re-writing to do, as the syntax of the two languages is different. Perhaps in the future, an interpreter will be developed that accepts the BASIC-E/65 syntax. This would allow the programmer to develop a BASIC program in an interactive environment, and then compile the source when it is deemed completed. The compiler could then be modified (or a utility provided) to provide a listing of all the variables used in the program (akin to an Assembler's symbol table).

Even without the interpreter, this last option would make debugging easier for the programmer to verify he typed in all the variable names the same throughout the source. It is quite a task to check them all visually!

EDITOR

The editor provided with DOS/65 is what I would call a string-oriented line editor, similar to the awkward single-line editors found for years on mainframes. The editor is not all that bad once you get used to it, though.

The editor is designed to operate on an 'active' buffer, while storing previously edited information on the disk on a different file from the original. Based on my experience with this editor and the accompanying manual, if you do not know how to use the editor, you won't learn from the supplied instructions. Let's call the manual a good reference guide. To learn how to use the editor, I borrowed a copy of Rodnay Zaks' "The CP/M Handbook with MP/M", (Sybex 1980 ISBN 0-89588-048 - 2). The DOS/65 editor does not support all the capabilities of the true CP/M editor as described in the referenced book, but it is compatible on those commands detailed in the DOS/65 EDIT manual.

Let's start at the beginning.

The editor can edit or create a file for you. As with all other utilities provided with the system, DOS/65 allows the EDIT program to be on one disk while the target file can be on another, all independent of the number of drives you have. A warning should be made, however, that the disk that you are targeting the file for should be 'logged in' to the DOS, meaning a warm boot should be done on the disk. This allows DOS/65 to write to an unused portion of the disk without the possibility of clobbering some other program/data. DOS/65 will not let you clobber something easily, however, as if your disk is not 'logged in' and you try to write to it, the system coughs, spits out an error message, and lets you know you goofed. Then, to add insult to injury, the system logs in the offending disk for you. No big problem, just some lost time!

Once you invoke EDIT and have the right disk in to receive the file, DOS/65 takes care of finding the existing file, or automatically creates a new one if it is not resident on the disk. If your existing file is made much larger by your editing session, DOS/65 finds room for the additional data on disk and expands the file. No '<u>out of file room</u>' messages, unless you are unfortunate enough to completely fill the disk. Not to fear, however, as the editor produces a copy of the original and renames it with a .BAK type qualifier, so if you make a terrible mistake (or run out of room), it is easy enough to recover.

Seven basic command groupings are provided for control of the editor. These are:

1. Editor termination
2. Library files
3. Buffer transfer
4. Line oriented
5. Character oriented
6. String oriented
7. Macro execution

The 'editor termination' commands provide for dumping your changed edit file, quitting with no changes, exiting with all changes, or saving all changes and restarting for a second pass or some more modification. 'Library file' commands allow writing lines to a scratch file for later inclusion in your text. You can also read back from this file or another file and have the source lines included in the present file. You may write as many lines as you like, but you read back the entire library file. 'Buffer transfer' allows reading/writing of buffer contents. The buffer's length is determined by the available memory after DOS and program overhead. Changes to the text take place only in the buffer. The buffer doesn't "roll", but is fixed at a given location, until moved off to disk or reclaimed from disk.

'Line oriented' commands move the cursor through the text in the buffer line by line. This allows you to back up or skip over several lines of text in one fell swoop, skip to the beginning (or end) of the text within the buffer, list n lines of text (while not repositioning the cursor), or delete n lines. Again, the cursor only works within the buffer contents. 'Character oriented' commands work on characters (versus lines). The user needs to remember a line is delimited with a (CR) (LF) in the buffer, and the cntl-I tab function also represents only one character. Commands in this group move the cursor within a line on a character basis, or delete individual characters within a line.

'String oriented' commands work on a group of characters within the buffer. One may find character strings, or substitute one string for another. The 'insert text' command also belongs to this group. Finally, there is the 'macro' command. This command makes use of the fact that command lines may be stacked, such as performing one operation right after the other entered on one command line. The macro allows a user to define a command line and have it executed as many times as desired. It is not the conventional macro one normally thinks of when discussing assemblers.

As stated earlier, using this editor takes some getting used to. The manual falls short of teaching a person how to use the system, and the editor on the whole, while powerful, lacks user friendly features and refinements. One error message I had a hard time getting used to was: "BREAK - CAN NOT DO COMMAND SPECIFIED TIMES AT x", where x is the single-letter command being processed. In my happenstances, it usually meant a specified string could not be found, or I asked a string to be changed more times than it existed in the buffer. Other error messages deal with unrecognized commands, buffer memory full, or the requested library file doesn't exist.

The editor has two key-word abbreviations that I believe should be given more prominent location and better coverage: cntl-Z and cntl-L (I shall abbreviate these ^Z and ^L). These command sequences are used in the Find or Substitute commands to denote and 'end of field' or a (CR)(LF) sequence, without having to use the 'return' key. As an example, change the next occurrence of "my tame domesticated carnation.(CR)" to "my wild Irish (CR) rose.". To accomplish this feat, enter the following:

```
Smy tame domesticated carnation. ^L^Zmy wild Irish^Lrose.
(CR)
```

The (CR) indicates a carriage return, which is actually a (CR)(LF) within the text string entered into the buffer. The control characters are shown in upper case for clarity. They are recognized in either upper or lower-case.

Let us remember, however, this editor is not a word processor. The original intent of the CP/M DOS was a programmers tool. The editor may lack fancy bells and whistles, but

TRY US!

WE MEAN BUSINESS

SC5/80-1 Computer System

With 1 Denver Board Multi Processor
Wired for 6 Users
Expandable to 14 Users

$6990.00

For Each Add'l Multi Processor User Add $1,000.00

- **Includes Super Utility Package**

Plus our incredible new super system data base manager. With brand new Disk Tech One, 2 year warranty on internal hard disk components. Beautiful hand finished oak cabinet over steel frame. Instant access to all components. Available with casters or plastic feet

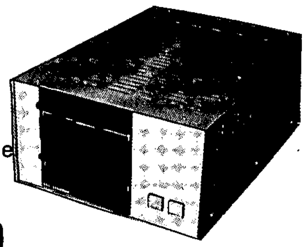
BEAUTIFUL AS WELL AS FUNCTIONAL

CLOSE OUT! C3-OEM

2 MHz/56K*
With 6502 and Z80 Processors

With Centronics Parallel Printer Interface

While they Last!



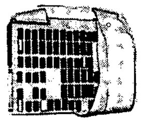
$1499.00

*Includes 8K Exec Memory for CP/M or Multi-User

- Buy for a Spare
- Add to a Spare H.D.
- Develop Programs

Spare 510CPU Board with Z80+6502 $169.00

NEW CONTROLLER/INTERFACE!



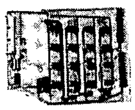
$599.00
Quantity 1

The new 9590 Controller replaces the OSI 590/525 set with 1 single board. Unlike others you've seen, it works perfectly with OSI or Denver boards, and all versions of 65U

It is compatible with old Style 592 Interface with 3 cable adapter, or attaches with single ribbon cable to our new 9592 H.D. Interface Board. Unlike others, our interface completely supports the ready/fault indicator on the 80 meg. hard disk.

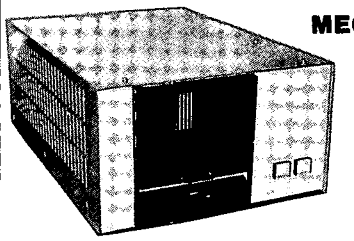
9592
Quantity 1

$269.00



MULTI PROCESSOR!

COMPLETE 10 MEGABYTE SYSTEM



$3999.00

Boot From Hard or Floppy!

- 8″ Hard Disk
- 8″ Floppy Disk

Single user with Centronics parallel printer interface expandable to eight users.

Enclosed in table top cabinet as shown or mounted in deluxe floor cabinet, as above left. Add $400.00

12 SLOT BUS

Fits standard OSI mount holes can be connected in "T" or daisy chained by cable for expansion.

$79.00

10 meg. Subsystem WOW!

$1999.00

Add to any existing OSI floppy based computer.

Just plug in one board and set this little gem on top or next to your computer and voila! Welcome to speed and convenience. Completely self contained with power supply. 9598 Hard Disk Controller also available separately at $699.00

DEALERS! **If you're still alive, call us and ask about our aggressive new price structure. We'll send you an unbelievable spares price list.**

SPACE-COM International
22991 La Cadena Drive, Laguna Hills, CA 92653 (714) 951-4648

it is fast. The .BAK -up feature has saved me more than once during my short experience using DOS/65, and I am sure many users will be glad not to have to worry about pre-allocating disk space, or over-extending a file past the already allocated number of tracks. All in all, this editor may not be a EMACS, ISPF, or TSO, but it's better than anything I have used on my C1P to date. Let's get cracking, and develop a full screen editor using BASIC-E/65!

SUPPLIED SOFTWARE

As mentioned earlier, DOS/65 comes with 10 diskettes. The DOS is scattered all over the 10 diskettes, sometimes duplicated, in order to provide the best coverage for use. A listing of the programs/games on the diskettes is provided with the system. Among the software is:

- Listings for the SIM, LOADER and BOOT routines
- Utilities to help determine where space is on the disk
- Utilities to help determine how large various programs are
- A data base program
- A mailing list program
- The soft keyboard library
- A print routine that dates and titles your pages
- Copy and move routines
- A routine to format diskettes to DOS/65 requirements
- Games, such as football, poker, and blackjack
- Utilities to convert your OS65D programs to DOS/65

These latter programs (called GETASCII.ASM, GETASM.ASM and GETBASIC.ASM) are set up to read the OS65D disk format, de-tokenize BASIC, expand tabs in ASM source files, or just copy an assembled ASM file to the DOS/65 disk. For BASIC, the routine will pad key words and variable names to conform to the requirements of BASIC-E/65. However, it does not check for non-conforming syntax or features that aren't supported.

The documentation for the games, et cetera, are found within the source listings themselves. I tried to compile and execute a few, and had no problems with any of them. As the support utilities are not part of the DOS proper, there is no mention of them anywhere else within the documentation. The only way to find out what you have and what they do is to look at all of them!

SUMMARY

DOS/65 is not your normal run-of-the-mill disk operating system for our kind of OSI machines. It is powerful in many respects, yet has its drawbacks also. I particularly like the file handling capability provided by the DOS, as well as its versatility in handling numerous virtual disks, while really using only one (in my case) or two physical disks. The BASIC compiler provides for a structured approach to writing your programs, and has many advanced features that make interfacing to ML routines easy. However, there is no easy way to load a ML program, and the documentation provides no examples on how this is accomplished.

BASIC in itself is not the "BASIC that screams" variety. In the simple test programs run, BASIC-E/65 came out even with OS65D on the whole, while falling way short of BASIC-in-ROM and HEXDOS. The Assembler has many nice features that the hard core ASM writer will find useful. The Editor provided with the system is simple to use, once you find a source that describes the proper way to operate it. The BAK-up feature also could prove to save many frustrated hours of labor due to the many problems that always seem to crop up when saving your one and only copy of a source.

It is a bit inconvenient to have to boot up OS65D before you can attempt to boot DOS/65. However, if you accidentally hit the break key when in DOS/65, you may re-enter the DOS from the monitor (M) by Go-ing at location $0100.

The documentation provided may intimidate you at first. Each section in the manual has a table of contents and list of figures/tables to help you wend your way through the information. More examples would be useful, and a compilation of the most-used information in one place would also be appreciated.

All in all I liked DOS/65. It is a very different type of DOS than OS65D or HEXDOS. You tend to lose the intimacy and immediate rapport with your system using DOS/65, but it behaves more like the systems I am used to using on more powerful minicomputers and some of the older mainframes. DOS/65 won't replace your current operating system if you have been hacking for a while . . . but if it had been your first operating system, it may have been your one and only.

★

6502 ASSEMBLY LANGUAGE PROGRAMMING CLASS

Part II

By: Richard L. Trethewey
Systems Operator for the
OSI SIG on CompuServe

Before we get any further into programming in Assembly language, I think it is a good idea for you to get familiar with the OSI Assembler/Editor. Most, but not all, OS-65D diskettes have the Assembler/Editor installed on them. The easiest way to test for the presence of the Assembler is to boot up on the OS-65D disk and get to BASIC's "OK" prompt. At the "OK" prompt, enter the command:

```
EXIT
```

That will get you to the OS-65D command kernel's "A*" prompt. At the "A*" prompt, enter:

```
ASM
```

If the Assembler/Editor is not on that diskette, OS-65D will display "ERR #7" or your system will lock up. If the Assembler is present, it will be called from the disk and started. You will then see:

```
OSI Assembler Editor
Copyright 1976 by OSI
```

or something similar. You'll note that the Assembler uses a period (".") as a prompt instead of BASIC's "OK". BASIC programs and Assembler source files are not stored on your disks in the same format. This means that you cannot use the Assembler to compose or edit BASIC programs, nor can you use BASIC to compose or edit Assembler source files.

Like BASIC programs, the OSI Assembler uses numbered lines to make up the source files. So, to enter a line into a file you must first enter a line number followed by the instruction text, as in:

```
10    LDA #$00
```

We will discuss the things you'll be entering in these lines a bit later, but for now we are just getting familiar with composing source files.

You may enter a line into the

current source file by typing a line beginning with a line number. The line is automatically inserted into the source file at the place specified by the line number. If a line is entered with the same number as a line currently in the file, then the old line is replaced with the new line. In addition, you may enter one of the file editing commands to compose the source file. The commands are:

PRINT (line specification)

This command performs the same function as the LIST command in BASIC. It displays the contents of the current file. You may optionally specify that only a certain range of line numbers be displayed. For example, "PRINT" displays all lines in the file. "PRINT1Ø" would display only line number 1Ø. "PRINT1Ø-2Ø" would display all lines beginning with line number 1Ø and ending with line number 2Ø. "PRINT1Ø-" displays all lines starting with line number 1Ø through the end of the file. "PRINT-2Ø" would display all lines from the start of the file through line number 2Ø.

DELETE line specification

This command erases the lines in the line specification. The format for the line specification is the same as with the PRINT command.

INITIALIZE

Clears the workspace, erasing the current file (if any). When you enter this command, you will be prompted "INIZ (Y/N)?". Respond with "Y" to clear the workspace.

RESEQ

Renumbers all of the lines in the current file, starting with 1Ø and increasing by 1Ø for each line in the file.

!text

Sends the "text" to OS-65D as a command. This is the same as the DISK!" command in BASIC. You will use this command to load and save files to and from disk.

You may abbreviate the commands PRINT, DELETE, INITIALIZE, and RESEQ with the first letter of the command (i.e. "P", "D", "I", and "R").

In order to save your source programs, you must have previously created a file on the disk to hold them using the BASIC program "CREATE" or a similar utility program. To load a file from disk into the Assembler's workspace, enter the command:

!LO filnam

at the "." prompt, where "filnam" is the name of the file to be loaded. Similarly, to save the file currently in the workspace to disk, enter the command:

!PU filnam

The command "EXIT" will send you back to the OS-65D "A*" prompt.

BEGINNING TO PROGRAM

In the next chapters, I will probably be beating some cliche's about Assembly language programming into the ground. Put up with these. They may be trite, but they are true. Above all else, remember that nothing is done for you automatically in Assembly language. If your program requires that a memory location holds a specific value, you must put that value in that location. One thing you're going to miss when you start Assembly language programming is <CTRL>'C'. If your program gets hung in an infinite loop, nothing short

MEM *PLUS*

BARE......$ 75	**32K........$300**	**52K....... $415**
16K $200	**40K........$350**	**56K $440**
24K $250	**48K........$390**	**64K $490**

MEM+ Options Include:

• OSI compatible floppy disk controller	add $85
• RTC — Real Time Clock — day, date and time with lithium battery backup	add $85
• Centronics parallel printer interface with software for OS65D and OS65U	add $65
• High reliability sockets for memory chips	add 15%
• RTC only (OSI CA-20 replacement)	$195

All boards feature solder mask, silkscreen, **gold-plated** edge connectors and a one year warranty.

07514 51893

Generic Computer Products

High Resolution Color Graphics

Our new Color Plus board provides 256 x 192 high-resolution graphics with 15 colors. Two 8-bit resolution joystick interfaces are included. Software extensions to OS65-D BASIC provide a superset of APPLE II graphics instructions.

Color Plus connects to the standard 48-pin bus or the 16-pin bus.

Pricing:

CP-8 for C8 or C3 computers:	$195
CP-4 for C4 computers (5V only):	$245
CP-bare Bare board with software:	$ 75

VISA, MasterCard, personal checks and C.O.D.s all accepted. Add $5 per board for shipping and handling.

To order, or for more information, contact:
Fial Computer
5221 S.W. Corbett
Portland, Oregon 97201
(503) 227-7083

of pressing the <BREAK> key is going to stop it. Many of your first programs are going to just "hang". Don't worry about it. It happens to everyone. Therefore, I recommend you take care to always, always, ALWAYS save your program's source code to a disk file before you try to execute it. So right now, prepare an OS-65D disk for your work here, and create two files. The first is a working file to hold the programs we'll be writing here. The second is a scratch file for use in emergency situations.

Let's discuss two of the most common 6502 opcodes, "LDA" and "STA". If you look on the opcode list, you'll see that "LDA" stands for "Load Accumulator". The question is what are you going to load it with? You essentially have two choices. You can either load the accumulator with a specific number or you can load it with the contents of a memory location. The "STA" command stands for "Store Accumulator". This command copies the current contents of the accumulator into a specified memory location. Note that after an STA, the contents of the accumulator are unchanged. The STA command is equivalent to the POKE command in BASIC. The LDA command can be thought of as equivalent to the PEEK command in BASIC when it is used to load the accumulator with the contents of a memory location. LDA is also like an equation in BASIC when loading the accumulator with a specific value. For example:

Assembler	BASIC Equivalent
LDA #$ØØ	ACC = Ø
LDA $DØØØ	ACC = PEEK(53248)
STA $DØØØ	POKE 53248, ACC

You'll note that I used hexadecimal numbers in the examples for the Assembler. I could have just as well used the same decimal numbers as I did in the examples for BASIC. The OSI Assembler understands both. Now we'll see how this works on your machine.

Boot up your system with the OS-65D you made before, get to BASIC's "Ok" prompt, enter "EXIT" to get to "A*" and enter "AS" to invoke the OSI Assembler. You should be at the Assembler's "." prompt now. Enter the following lines of code:

```
1Ø        *=$4ØØØ
2Ø        LDA #$25
3Ø        STA $DØ3F
4Ø        STA $D13F
5Ø        STA $D23F
6Ø        STA $D33F
7Ø        RTS
```

Double check that your listing matches the above text by entering the "P" command to have the Assembler list out your program. If it matches, enter the command "A". The Assembler will pause a moment and then list your program out on your screen again, but this time you'll see numbers printed between the line numbers and the text portion of each line. These numbers are the memory addresses and machine code represented by the Assembly language program you have just entered. If you made a mistake in entering the program, the Assembler would have displayed an error message underneath the line on which the error occurred with a dotted line extending to the approximate point in the line where the actual error was typed in.

If you got an error message, retype the bad line and repeat the "A" command until the Assembler stops printing the error message.

You'll notice that the first line says "*=$4ØØØ". The asterisk tells the Assembler that this is to be an "origin statement". An origin statement is an equation that tells the Assembler what memory address we want to start the program with. I chose $4ØØØ since it is above the operating system and also above where the program source code is stored in memory and yet is within the RAM range of all OSI systems.

Again at the "." prompt in the Assembler, enter the command "A3". This command tells the Assembler to assemble the program and put the result into the specified memory area. The machine code generated by an Assembler is also called "object code". You'll note that this time the Assembler paused, but nothing was printed. This is normal.

Now enter the command "!GO 4ØØØ". This is the OS-65D command that tells the computer to execute the machine code program at memory address $4ØØØ, which is where the Assembler put our program. You should see a "%" printed several times down the far right edge of your screen. Congratulations! You just wrote and ran your first Assembly language program!

The next thing I want to discuss is the concept of an "index". An index in Assembly language is always a value added to an address to determine a resulting new address. Both the X and Y registers can function as indices, although each has some peculiar features. Look at the following program:

```
1Ø       *=$4ØØØ
2Ø;
3Ø       LDY #$ØØ      ; CLEAR Y
4Ø       LDA #$2Ø      ; LOAD ACC. WITH A <SP>
5Ø Pl    STA $D6ØØ,Y   ; SAVE ACC. AT $D6ØØ + Y
6Ø       INY           ; INCREMENT INDEX
7Ø       CPY #$ØØ      ; IS Y A ZERO YET?
8Ø       BNE Pl        ; IF NOT, GO BACK TO Pl
9Ø       RTS           ; YES IT IS, QUIT
```

I'm introducing several new concepts here. The first occurs line #5Ø. The word "Pl" isn't a 65Ø2 opcode. It's called a "label". Labels are for the Assembler's use. In Assembly language programming, we don't refer to points in a program by their line number. That is, there is no "GOTO 5Ø". Rather, we refer to them by label. Line numbers are strictly for the editor's convenience so that it will know where to put lines that you enter from the keyboard.

Next, you'll notice line 1Ø's "LDY" command. This command is the same as "LDA", but loads a number into the Y register rather than the accumulator. Line 6Ø holds the command "INY". As the opcode table says, this command increments the current contents of the Y register by one. Next, in line 7Ø is the "CPY" command. This command asks the 65Ø2 to compare the contents of the Y register with some other value. In our case, we are asking for a comparison of Y with a specific value, but we can also compare with the contents of a memory location. Internally, the 65Ø2 subtracts the value being compared from the Y register and the results of this subtraction set or clear flags in the 65Ø2 that we can look at with other commands. Finally, in line 8Ø we have the command "BNE". This is called a "conditional branch". What that means is that the program branches to the destination indicated if the condition tested for is true. In our case, we've conditioned the flags with the previous "CPY". If Y <> Ø then in our program, control would branch back to the label Pl. When Y does equal zero, the condition tested is not true and control falls through to the next instruction. Remember that Y is an 8 bit register and can only hold values from Ø through 255, thus after 256 passes through the code from lines 50 to 80, Y "flops over" and becomes zero again. This repetitive passing through the same instructions is called "looping" and the area of code itself is called a "loop".

Assemble this program with the "A3" command and execute it with the "!GO 4000" command. A portion of your screen will clear. Fast, wasn't it? So fast you couldn't see it happen. This Assembly language program is equivalent to the following BASIC program:

```
10 ACC=32 : Y=0
20 POKE 54784+Y, ACC
30 Y=Y+1
40 IF Y <> 256 THEN 20
```

If you like, try running the above BASIC program and you'll get a small idea of how much faster Assembly language programs run.

For next time, check your Assembly language books and look up the commands "INC", and "DEC" and the concept of "addressing modes".

OSI ASSEMBLER EDITOR COMMANDS

Mnemonic	Description
ADC	ADd with Carry
AND	logical AND with accumulator
ASL	Arithmetic Shift Left
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on EQual
BNE	Branch on Not Equal
BMI	Branch on MInus
BPL	Branch on PLus
BVC	Branch on oVerflow Clear
BVS	Branch on oVerflow Set
BIT	and with accumulator (no change to acc.)
BRK	BReaK
CLC	CLear Carry flag
CLD	CLear Decimal flag
CLI	CLear Interrupt flag
CLV	CLear oVerflow flag
CMP	CoMPare with Accumulator
CPX	ComPare with X register
CPY	ComPare with Y register
DEC	DECrement memory location
DEX	DEcrement X register
DEY	DEcrement Y register
EOR	Exclusive OR
INC	INCrement memory location
INX	INcrement X register
INY	INcrement Y register
JMP	JuMP to new address
JSR	Jump to SubRoutine
LDA	LoaD Acumulator
LDX	LoaD X register
LDY	LoaD Y register
LSR	Logical Shift Right
NOP	No OPeration
ORA	OR with Accumulator
PHA	PusH Accumulator on stack
PHP	PusH Processor status register on stack
PLA	PulL Accumulator from stack
PLP	PulL Processor status register from stack
ROL	ROtate Left
ROR	ROtate Right
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	SuBtract with Carry
SEC	SEt Carry flag
SED	SEt Decimal flag
SEI	SEt Interrupt flag
STA	STore Accumulator in memory
STX	STore X register in memory
STY	STore Y register in memory
TAX	Transfer Accumulator contents to X register
TAY	Transfer Accumulator contents to Y register
TSX	Transfer Stack pointer to X register
TXA	Transfer X register contents to Accumulator
TXS	Transfer X register contents to Stack pointer
TYA	Transfer Y register contents to Accumulator

★

OSI ROM ROUTINES

(Part 3)

Part 2 published June, 1984

By: Leroy Erickson
Courtesy of OSMOSUS NEWS
3128 Silver Lake Road
Minneapolis, MN 55418

This month's ROM routine is SYNMON page 3, the 65V monitor for the polled keyboard. This routine is at $FE00 in every C2-4P and C4P (cassette and disk), and every video-based C8P. It is reached by doing a

ALTERNATIVE
Data Base Management System

The **Alternative Data Base Management System** is a general purpose software product that allows the user to easily develop on-line applications without the need for computer programming. The system is entirely menu driven. DBMS files are created to meet the user's specifications in a matter of minutes. Data may then be added, changed or deleted quickly and efficiently using the file maintenance programs provided.

Reports from DBMS files can either be generated interactively or by means of a 'Job' containing a stored list of user inputs. The DBMS Report Writer produces sequential listings, sorted reports or exception reports based on conditional tests.

Data management is made easy with the DBMS Merge program. Records may be copied from one DBMS file to another based on multiple conditional tests. Subsets of records with similar characteristics may be merged from several DBMS files into a scratch file for further processing.

The DBMS Arithmetic program allows the user to automatically update numeric data in DBMS files. Calculations may be performed using any DBMS field containing numeric data or numeric constants entered at run time.

System Requirements

OSI Computer System with 48K RAM
OS-65U Version 1.2 or Later (Proof of Purchase is Required)

alternative computer systems
2324 E. 8th St. Davis, CA. 95616
(916) 758-5364

I wish to order:

☐ Alternative DBMS $195
(Includes User's Guide)
☐ User's Guide $20
(For Evaluation)
☐ Product Brochure -FREE-

My system configuration is:

OSI Model: ____________
Terminal Type:
☐ Serial Terminal
☐ Television Monitor
Storage Devices:
☐ Hard Disc
☐ Dual Floppies
☐ Single Floppy

Order Subtotal: ____________
In CA add sales tax: ____________
Total Order: ____________
☐ Check Enclosed ☐ Ship COD

Signature: ____________
Name: ____________
Addr: ____________
City/St/Zip: ____________

'RESET' and typing 'M' on any of these systems, or by entering the command 'RE M' to the operating system on disk systems. It can also be reached by doing a 'GO FE00' command.

When it's entered, the monitor clears the screen and displays a 4-digit hexadecimal number (the current address) and a 2-digit hexadecimal number (the current data) near the top of the screen. The current data is the contents of the memory location pointed to by the current address. The monitor can be put in "Address Mode", in which case the current address can be changed, or in "Data Mode", in which case the current data can be changed, thus modifying the computer's memory. In either mode, any valid hexadecimal digit which is entered is rotated into the bottom nybble of the address or data. For example, when in Address Mode with a current address of 'FE05', if a '7' is entered the displayed address will change to 'E057' and the displayed data will change to the contents of 'E057'. When in Data Mode with a display of '4E11 45', if a '7' is entered the display would change to '4E11 57' and the contents of '4E11' would now be '57'.

Each mode has a few other commands. Address Mode has the following commands:

- '/' - Enter Data Mode.
- 'G' - Go to the current address.
- 'L' - Switch to serial input and enter Data Mode.

Data Mode has the following commands:

- '.' - Enter Address Mode.
- CR - Increment current address.

In a ROM BASIC system, the serial input device is the cassette player. On these systems, a machine code tape may be loaded by going to the 65V monitor, typing 'L', and turning on the cassette player. On the tape would be the following data:

'.'
Command to enter Address Mode.

'aaaa'
A 4-digit hex number - the load address.

'/'
Command to enter Data Mode.

'dd'
The data byte for the current

```
                  ; ****************************************
                  ; ***                                  ***
                  ; ***       C4P BOOT ROM PAGE 3        ***
                  ; ***                                  ***
                  ; ***    65V Monitor for 540 Video     ***
                  ; ***      and Polled Keyboard         ***
                  ; ***                                  ***
                  ; ***    Comments by Leroy Erickson    ***
                  ; ***           April 1982             ***
                  ; ***                                  ***
                  ; ****************************************
                  ;
00FB=             H00FB=$00FB ; In Flg=0=>Keybrd,else Serial
00FC=             H00FC=$00FC ; Current Data
00FD=             H00FD=$00FD ; Current data high (filler)
00FE=             H00FE=$00FE ; Current Addr, low
00FF=             H00FF=$00FF ; Current Addr, high
0130=             H0130=$0130 ; NMI Vector
01C0=             H01C0=$01C0 ; IRQ Vector
D0C6=             HD0C6=$D0C6 ; Screen-4th line-6th column
D0CA=             HD0CA=$D0CA ; Screen-4th line-10th column
D0CB=             HD0CB=$D0CB ; Screen-4th line-11th column
DF00=             HDF00=$DF00 ; Scanned keyboard address
FB05=             HFB05=$FB05 ; 430B Board
FB06=             HFB06=$FB06 ;   Ditto
FC00=             HFC00=$FC00 ; Serial Port Status Reg
FC01=             HFC01=$FC01 ; Serial Port Data Reg
FD00=             HFD00=$FD00 ; Scanned keyboard ROM Address
                  ;
FE00                     *=$FE00
                  ;
FE00 A228         HFE00  LDX   #$28      ; Initialize stack
FE02 9A                  TXS             ;
FE03 D8                  CLD             ; Clear decimal mode
FE04 AD06FB              LDA   HFB06     ; Init the 430 board
FE07 A9FF                LDA   #$FF      ;
FE09 8D05FB              STA   HFB05     ;
                  ;
                  ; *** Clear the screen ***
                  ;
FE0C A2D8                LDX   #$D8      ; Get high page # + 1
FE0E A9D0                LDA   #$D0      ; Get low page #
FE10 85FF                STA   H00FF     ; Store in data ptr
FE12 A900                LDA   #$00      ; Get a zero
FE14 85FE                STA   H00FE     ; Data Ptr=$D000
FE16 85FB                STA   H00FB     ; In Flag=Keyboard
FE18 A8                  TAY             ; Zero the index
FE19 A920                LDA   #$20      ; Get a space
                                         ;
FE1B 91FE         HFE1B  STA   (H00FE),Y ; Store a space
FE1D C8                  INY             ; Bump index
FE1E D0FB                BNE   HFE1B     ; Loop on 256 bytes
FE20 E6FF                INC   H00FF     ;  then bump page #
FE22 E4FF                CPX   H00FF     ; Done ?
FE24 D0F5                BNE   HFE1B     ; No, keep going
FE26 84FF                STY   H00FF     ; Set data ptr=0000
FE28 F019                BEQ   HFE43     ; Do forced branch
                  ;
                  ; *** ADDRESS MODE ***
                  ;
FE2A 20E9FE       HFE2A  JSR   HFEE9     ; Get an input char
FE2D C92F                CMP   #$2F      ; '/' ?
FE2F F01E                BEQ   HFE4F     ; Yes ==> Data Mode
FE31 C947                CMP   #$47      ; 'G' ?
FE33 F017                BEQ   HFE4C     ; Yes,Go to cur addr
FE35 C94C                CMP   #$4C      ; 'L' ?
FE37 F043                BEQ   HFE7C     ; Yes,In Flg=serial
FE39 2093FE              JSR   HFE93     ; Else,valid hex ?
FE3C 30EC                BMI   HFE2A     ; No, get next char
FE3E A202                LDX   #$02      ; Point to cur addr
FE40 20DAFE              JSR   HFEDA     ; Rotate in new digit
FE43 B1FE         HFE43  LDA   (H00FE),Y ; Get new data
FE45 85FC                STA   H00FC     ; Save in data reg
FE47 20ACFE              JSR   HFEAC     ; Display addr & data
FE4A D0DE                BNE   HFE2A     ; Go get next char
                  ;
                  ; *** GO COMMAND ***
                  ;
FE4C 6CFE00       HFE4C  JMP   (H00FE)   ; Go to current addr
                  ;
                  ; *** DATA MODE ***
```

continued

address.

'cr'
A carriage return to step to next address.

'dd','cr'
Successive data bytes for the program.

'.'
When done, return to Address Mode.

'aaaa'
A 4-digit hex number - starting address.

'G'
A 'GO' command to start the program.

Now, look at the code. The monitor is entered at $FE00, it inits the stack, clears decimal mode and initializes the '430B' board. This last step is a remnant of earlier OSI systems on which the cassette port was built around a UART on a circuit board called a 430, revision B.

After initialization, the screen is cleared (540 video style), a flag is set to select input from the scanned keyboard, and Address Mode is entered.

Continued on page 14

DISK DRIVE RECONDITIONING

WINCHESTER DRIVES

NEW OSI CD23 (SA4008)

FLAT RATE CLEAN ROOM SERVICE.
(parts & labor included)

Shugart	SA4008	23meg	$550.00
Shugart	SA1004	10meg	$450.00
Seagate	ST412	10meg	$350.00
Seagate	ST506	5meg	$290.00

FLOPPY DRIVE FLAT RATES
Parts & Labor Included

8" Single Sided Siemens	$150.00
D&E Series	
8" Double Sided Siemens	$170.00
P Series	
8" Double Sided Remex	$225.00
8" Single Sided Shugart	$190.00
8" Double Sided Shugart	$250.00

ONE WEEK TURN AROUND TYPICAL

90 Day warranty on Floppy & Large Winch.
1 Yr. Warranty on 5" & 8" Winchesters.

Write or call for detailed brochure

Phone: (417) 485-2501

FESSENDEN COMPUTERS
116 N. 3RD STREET
OZARK, MO 65721

```
                    ;
FE4F 20E9FE  HFE4F  JSR   HFEE9     ; Get an input char
FE52 C92E           CMP   #$2E      ; '.' ?
FE54 F0D4           BEQ   HFE2A     ; Yes==>Address Mode
FE56 C90D           CMP   #$0D      ; CR ?
FE58 D00F           BNE   HFE69     ; No, test for hex
FE5A E6FE           INC   H00FE     ; Else,incr current
FE5C D002           BNE   HFE60     ;    address
FE5E E6FF           INC   H00FF     ;
FE60 A000    HFE60  LDY   #$00      ; Get data at
FE62 B1FE           LDA   (H00FE),Y ; current address
FE64 85FC           STA   H00FC     ; Save in data area
FE66 4C77FE         JMP   HFE77     ; Skip ahead
                                    ;
FE69 2093FE  HFE69  JSR   HFE93     ; Valid hex ?
FE6C 30E1           BMI   HFE4F     ; No,get a new char
FE6E A200           LDX   #$00      ; Point to data reg
FE70 20DAFE         JSR   HFEDA     ; Shift in the digit
FE73 A5FC           LDA   H00FC     ; Get the new value
FE75 91FE           STA   (H00FE),Y ;Store at cur addr
FE77 20ACFE  HFE77  JSR   HFEAC     ; Display current
FE7A D0D3           BNE   HFE4F     ;   values and loop
                    ;
                    ; *** L COMMAND ***
                    ;
FE7C 85FB    HFE7C  STA   H00FB     ; In Flag=serial
FE7E F0CF    HFE7E  BEQ   HFE4F     ; Enter data mode
                    ;
                    ; *** SERIAL INPUT ROUTINE ***
                    ;
FE80 AD00FC  HFE80  LDA   HFC00     ; Get status register
FE83 4A             LSR   A         ; Carry=Input flag
FE84 90FA           BCC   HFE80     ; Wait until ready
FE86 AD01FC         LDA   HFC01     ; Then get that char
FE89 EA             NOP             ; *** Junk Filler ***
FE8A EA             NOP             ;        ...
FE8B EA             NOP             ;        ...
FE8C 297F           AND   #$7F      ; Strip parity bit
FE8E 60             RTS             ;   and go home
                                    ;
FE8F 00             BRK             ; *** Junk Filler ***
FE90 00             BRK             ;        ...
FE91 00             BRK             ;        ...
FE92 00             BRK             ;        ...
                    ;
                    ; *** TEST VALID ASCII ***
                    ;
FE93 C930    HFE93  CMP   #$30      ;
FE95 3012           BMI   HFEA9     ; < '0' ==> Invalid
FE97 C93A           CMP   #$3A      ;
FE99 300B           BMI   HFEA6     ; '0' - '9' ==> Valid
FE9B C941           CMP   #$41      ;
FE9D 300A           BMI   HFEA9     ; < 'A' ==> Invalid
FE9F C947           CMP   #$47      ;
FEA1 1006           BPL   HFEA9     ; > 'F' ==> Invalid
FEA3 38             SEC             ;
FEA4 E907           SBC   #$07      ;A-F =>'9'+1 to '9'+6
FEA6 290F    HFEA6  AND   #$0F      ;'0'-'F' => 0 - 15
FEA8 60             RTS             ; Go Home
                    ;
FEA9 A980    HFEA9  LDA   #$80      ; Invalid, set hi bit
FEAB 60             RTS             ;  and go home
                    ;
                    ; *** PRINT ADDR AND DATA ***
                    ;
FEAC A203    HFEAC  LDX   #$03      ; Plan on 4 bytes
FEAE A000           LDY   #$00      ; Zero index
FEB0 B5FC    HFEB0  LDA   H00FC,X   ; Get a data byte
FEB2 4A             LSR   A         ; Shift high nybble
FEB3 4A             LSR   A         ;   into low nybble
FEB4 4A             LSR   A         ;
FEB5 4A             LSR   A         ;
FEB6 20CAFE         JSR   HFECA     ; Display high nybble
FEB9 B5FC           LDA   H00FC,X   ; Get the byte again
FEBB 20CAFE         JSR   HFECA     ; Display low nybble
FEBE CA             DEX             ; Decrement counter
FEBF 10EF           BPL   HFEB0     ; Loop for 4 bytes
FEC1 A920           LDA   #$20      ; Get a space
FEC3 8DCAD0         STA   HD0CA     ; Overlay 5th & 6th
FEC6 8DCBD0         STA   HD0CB     ;   characters
FEC9 60             RTS             ; Go home
```

cont. on pg.14

THE DATA SYSTEM
A REVIEW

PEEK(65)
By: Edward T. Gieske, Jr.

Before getting down to the nitty gritty, let's generalize about data bases a bit, and this one in particular. Basically, all data bases do about the same thing: they allow you to collect, store, massage and retrieve data. In their basic element, data bases are probably more alike than they are dissimilar, but each has its own claims and personality.

Flexibility means power and power also means complexity. To put it the other way round, a program that is simple and easy to learn, will probably fall short when it comes to the capability of handling and manipulating large quantities of complex data.

The Data System (TDS) is in the powerful range, but still retains a remarkable degree of operator simplicity. TDS came out of the original OSI DMS mold. DMS did its job, resplendent with monotonous question ad nauseam, but it nonetheless established a file format that has been the standard for OSI. Various people cursed DMS, tinkered and altered the code, tailoring it to their specific needs and thus rendering it near useless to others.

Enter Gary Gesmundo. Gary grew up with early OSI, knew its wonders and pitfalls and thus set out to create his own data base manager. That's about when DMS+ was being talked about. What's that, you say? Simply put, a DMS file, but with a much larger header to hold all that extra stuff that a sophisticated DBM will need to function properly. Months and months went by as the system slowly and methodically took form. Routines were reduced to machine code for speed and file handling expedited tremendously.

Enter John Huntley. John's a lawyer, by trade, and now principal of Gander Software, Ltd., meticulous with words, a stickler for accuracy and frustrated by anything that doesn't do what it is supposed to do, the way it is supposed to do it.

Put both gentlemen in a pot, stir and out comes Gander's version of TDS. Now you can begin to appreciate the quality of the package we are talking about.

To start off with, I scanned the 140+ page manual. Then I took it home to really read it several times more. I didn't have a machine at home to "follow along", so some did fall through the cracks, but that was my fault. It didn't take long, however, for things to begin to fall into place.

THE MANUAL

The manual is broken down into two main areas: a Tutorial Section and a Reference Section. The first is subdivided into an Overview and a Data Skills section that really is a "how to get started" walk through. As I write, I have been told that serious thought is being given to either extending the "walk through" portion of the manual or even a tutorial on disk. Which ever it is, if it bears any resemblance to Gander's other efforts, it will be well worth having.

The second is broken down into 8 subsections and an appendix that includes an extensive index. One of the unique features of this section and the system itself is that every screen presented on the operator's terminal has a number in the top corner that relates, not only to the function being performed, but also to the manual text. If you get lost, you can always go to the manual and retrace your steps.

With any opus of this magnitude, there are bound to be a few spots that could be improved, however, instances of confusion or misleading information are definitely the exception rather than the rule. All in all, the manual is refreshingly well-done. Within its pages, you will find information sufficient to answer almost any question. But should you have an unresolved problem, the folks at Gander will bend over backwards to give you prompt and accurate help - at least they did for me.

THE PROGRAM

Installation is a snap! OSI's INSTALL program is used with clear, simple instructions. It's "goof proof".

As I said, TDS is in the "power" group, but the Gander people have given this program a split personality. The power is there. It can be as intricate and complex as the job requires, but when it comes to day to day use, it is a lead pipe cinch for those entering data, gristing and/or printing out reports, labels, etc. That's because, once you have laborously thought out your problem and entered it into TDS, virtually everything is stored for future use and automatically presented to the "user" in simple user defined menus.

The storing power alone is reason enough to make the switch to TDS. Just think of it: you can store up to nine Job Files. Whoopee! But each of those jobs can store up to 4Ø different report formats, 4Ø sets of conditions, 4Ø file merge routines and 4Ø file posting routines. That's 4Ø^4 possibilities in the first Job File - and there are 8 more to go! Chances are that most users will never get to the 2nd Job File. The real point is that once these jobs are set up and saved, one simply calls in a whole sequence of events from simple menu selections - ones that you personally have named - like: "Do Month-end Update" or "Print Month-end Report". Then go for a coffee while your machine gets on with it.

It would be nigh impossible to list all the gristing capabilities here, but there are some interesting ones that leave old DMS far behind in the dust. Let's have a quick look at MERGE. For a start, forget about simply moving records from file to file, but rather selected fields. They can be Appended to the end in new records, Overlaid so as to up-date a record and the overlay can be qualified upon finding a Match in yet another field.

Lest I forget, Gander has included rules and syntax similar to OSI's Planner Plus (KeyCalc) in setting up calculations. This permits all sorts of mathematical operations on various fields, with the results saved in the same field or elsewhere.

Take all this power, put it together in the right sequence, and you will suddenly realize that you are on the verge of creating your own general ledger, specialized inventory system, billing, etc.

Incidentally, TDS will handle up to 1ØØ fields per record, which should open up new areas of business utilization.

Sorting is all done by machine

From Gander Software, Ltd. *The Ultimate Data Base Manager*

THE DATA SYSTEM

AFTER 2½ YEARS OF DEVELOPMENT, THE MUCH RUMORED & SOUGHT DBM IS HERE!
WITH IMPROVEMENTS OVER THE OTHER SIMILAR ITEM

- Stored Report Formats
- Stored Jobs, Formats, Calcs.
- Multiple Condition Reports
- Multiple File Reports
- Calc. Rules Massage Data
- Up to 100 Fields Per Record
- User Designed Entry/Edit Screens
- Powerful Editor
- Merges - Append, Overlay, Match
- Posting - Batch Input
- Nested Sorts - 6 Deep
- Abundant Utilities

HARDWARE REQUIREMENTS: 48K OSI, Hard Disk, serial system, OS-65U 1.42 or Later; Space required: 1.3 megabytes for programs and data.

TECHNICALITIES: User configurable to either 1.42 or 1.43 and Later; 9 job files, each saving up to 40 report formats, 40 calculations, 40 merge routines, or 40 posting routines, 40 file editors, and all those on up to 9 conditions each! Will convert older Type 10 and Type 20 file headers to the new Type 30; Etc., etc.

PRICE: $650.00 (User Manual $35.00, credited towards TDS purchase). Michigan residents add 4% sales tax. 30 day free trial, if not satisfied, full refund upon return.

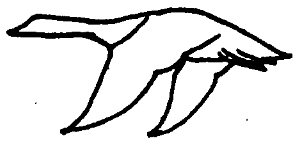
DEALERS: This package is the new standard in DBM's. We are out to make it your standard too! Give us a call to find out how we can make TDS very worth your while.

SIMPLY POWERFUL AND COMPLETE: This DBM has external simplicity, and a manual that is written in English; complete from Tutorial to Definition of Terms. Although Gander provides support by the author, it shouldn't be necessary.

TDS's power is derived from all the things you would expect of a DBM, plus a list of capabilities (menu selectable and self-instructing) which include: Reports in any format, constructed on screen, giving only the desired data from up to three files with key file access; Move fields from one record to another; Sub Totals and Totals where wanted, etc.; Calc. Rules are similar in syntax to OSI's Planner Plus; PRTMAP modified to work with all system printers, with paging and still user independent; Quick File's QF Sort speeds sorting with no record length limits; Many machine language Utilities expedite the system; for instance Pack File is 2850% faster.

GANDER SOFTWARE, Ltd.

3223 Bross Road
"The Ponds"
Hastings, MI 49058
(616) 945-2821


"It Flies"

FROM THE FOLKS WHO BROUGHT YOU:
Financial Planner
Time & Task Planner
AND THERE IS MORE COMING SOON:
Program Generator for TDS
Proposal Planner
Time and Billing A/R

language and is suitably fast, but the nice thing is that you won't have to worry about it. If a change has been made to a file since it was last sorted, and you try to produce a report, it will automatically be done for you.

Reports come in several hues: Vertical, Horizontal, Multi-Line (for those cases where it won't all fit on one line), Multi-File and Labels. Setting up reports is a bit tedious, but once done correctly, it's saved and you needn't bother with it again. Another nice touch here is that they have added the date (automatic) at the top of the report and at the bottom a file report like "3 of 3 records or 100%". Totals and subtotals are possible and are averaged too! The Main Menu allows you to set or reset the date and direct the output to any printer you could put on an OSI machine. Not only that, but pagination, form length, etc. are problems of the past. The actual specs of your report can easily be made to fit on any preprinted form.

Just in case the complexity of power is beginning to get to you, there are Job Specification Reports that will give you every last detail of the job you have constructed; from file name to the number of printable lines on the report. There are separate Specification Reports for: the entry screen, merges, calculations, all reports and postings. For good measure, there is a File Specification Report that's just as detailed.

With any system there are always Utilities. Among them is the necessary Delete & Repack, but its nothing like DMS! Machine code to the rescue along with an extra option that just moves the last record up to fill the deleted record hole. All in all, that option will get the job done 2850% faster than ol' DMS. Here' another. How would you like to enter phone numbers like "3023633268" and then "Reformat a Field" to go back and add "(", ")" and "-" where necessary to make it look like a real phone number? Do you want Spooling? It's OSI, but it's there too! Of course there are the necessay routines to create and back-up data, Job Files, create data disks etc., etc. They work and work fast. Just try creating a master file. Don't blink. It will be created and initialized in nothing flat, ML again.

Even the data entry screen is custom designed by you. It could be what looks nice to you or it could look just like the form the information to be entered was taken from. It's your option and it's saved for future use.

Editing has been made even easier. Naturally, full advantage has been made of the 1.4+ editor, but you will also be assisted by a series of ESC functions: ESC 0 returns you to the main menu from anywhere, ESC 1 will back up one question, ESC 2 will return you to the bottom of the screen where you select your next option, ESC 3 ignores the last edit you made, ESC 7 initiates several kinds of field searches (a powerhouse in itself) and ESC ESC aborts and ESC. Each screen tells you which ESC functions are available at that point.

If all this sounds like you will have to sit down and do a lot of planning, you're right. But if you don't, and I don't care whose DBM you are using, you will probably not achieve the desired results. With TDS it's made easy. You can crank out a full set of planning sheets for everything that you will need to keep your logic straight.

As is often the case, the best is saved for last. Try TDS for 30 days. How many software packages make that kind of offer? It is a superior class product and the folks at Gander obviously have pride in it. We found it to be swift, effective and capable of handling most everything we could think of to throw at it and we think that you will too!

CONTINUED FROM PAGE 11

Look further down in the code. Location $FE80 is the serial input routine. Now it contains code for a 6850 ACIA, but with all of the 'NOP's and 'BRK's around it, it seems obvious that the previous code for the 430's UART was simply patched over.

Further down, at $FEF0, is a routine which the monitor does not even use! Surprise! It's a keyboard single scan routine. When it's called, it returns the values of all 8 rows 'OR'ed together. Simply mask the result to eliminate the 'shift lock' key ('AND' with 254 decimal or 'FE' hex) and if the result isn't zero, at least one other key is being depressed.

At the bottom of the monitor, at locations FEFA through FEFF, the interrupt vectors show up. What this means is that if this page were to be selected for both $FE00 and $FF00, the code would still execute normally for $FE00, but the last 6 bytes would be selected when a 'RESET', 'NMI', or 'IRQ' was received. On receiving a system 'RESET', the monitor would be entered. At this point, presumably, a cassette would be loaded. Original system debug, maybe?

That's it for this routine. Next month I'll cover page 4, the ROM BASIC support boot page. See you then.

```
                    ;
                    ; *** DISPLAY 1 HEX BYTE ***
                    ;
FECA 290F    HFECA  AND  #$0F      ; Strip high 4 bits
FECC 0930           ORA  #$30      ; Convert to ASCII
FECE C93A           CMP  #$3A      ; > '9' ?
FED0 3003           BMI  HFED5     ; No, skip ahead
FED2 18             CLC            ; Yes, convert to
FED3 6907           ADC  #$07      ;   'A' thru 'F'
FED5 99C6D0  HFED5  STA  HD0C6,Y   ; Save at current loc
FED8 C8             INY            ; Bump index
FED9 60             RTS            ; Go home
                    ;
                    ; *** INSERT NEW DIGIT ***
                    ;
FEDA A004    HFEDA  LDY  #$04      ; Set rotate counter
FEDC 0A             ASL  A         ; Move data into high
FEDD 0A             ASL  A         ;    nybble
FEDE 0A             ASL  A         ;
FEDF 0A             ASL  A         ;
FEE0 2A      HFEE0  ROL  A         ; Carry =  next bit
FEE1 36FC           ROL  H00FC,X   ; Rotate carry into
FEE3 36FD           ROL  H00FD,X   ;  data,hi & low
```

Continued on page 15

```
FEE5 88              DEY             ; Decr bit counter
FEE6 D0F8            BNE    HFEE0    ; Loop for 4 bits
FEE8 60              RTS             ; Then go home
              ;
              ; *** COMMON INPUT ROUTINE ***
              ;
FEE9 A5FB     HFEE9  LDA    H00FB    ; Test Input Flag
FEEB D091            BNE    HFE7E    ; Loop back if serial
FEED 4C00FD   HFEED  JMP    HFD00    ; Else,check keyboard
              ;
              ; *** KEYBOARD TEST ROUTINE ***
              ;
FEF0 A9FF     HFEF0  LDA    #$FF     ; Select all rows
FEF2 8D00DF          STA    HDF00    ; Store in row select
FEF5 AD00DF          LDA    HDF00    ; Read column values
FEF8 60              RTS             ; Return
              ;
FEF9 EA              NOP             ; *** Junk ***
              ;
FEFA 3001            .WORD H0130     ; Remnant NMI Vector
FEFC 00FE     HFEFC  .WORD HFE00     ; Remnant RESET Vector
FEFE C001            .WORD H01C0     ; Remnant IRQ Vector
              ;
                     .END
```

BEGINNER'S CORNER

ONE STEP BACK, TWO FORWARD

By: L. Z. Jankowski
Otaio Rd 1 Timaru
New Zealand

Last month's listing of the Otaio Mailing List (OML) was obviously for Disk and DOS 3.3. My original intention was to present the listing piece by piece, explaining how to adapt it to various OSI environments. But the Editor beat me to it and published the Disk demo version! We go on regardless, with apologies for any confusion that may have arisen from the June article.

The OML is, hopefully, adaptable to any BASIC language computer. It should, therefore, be adaptable to C1Ps. The first change C1P users should make is to line 100. Three numbers are changed to 234:

```
100 DATA 72,138,72,152,72,160,
0,169,32,  234,234,234,   153,
0,209,153
```

Also for C1Ps: delete lines 20 and 2010 and in line 90 let X=546 if you want to put the partial screen clear at $0222.

The discussion so far has centered on the listing titled "Sample of 'Trap'" (see June issue). This month we look at horizontal scrolling, variables and the Menu - see Listing.

HORIZONTAL SCROLLING

Although horizontal scrolling is definitely a 'twiddly bit' it was interesting to see if it could be done and whether the string would scroll fast enough. The answer was yes in both cases. OSI software and hardware is still one of the fastest combinations around.

The code of horizontal scrolling is in lines 40 to 70. The string to be POKEd is in B$, line 40. In line 50, X=53509 is the base screen address to which the string will be POKEd. The string in B$ is reconstructed character by character, from the right going left. Each time the forming string lengthens by one character, it is POKEd to the screen. The logic of how it all works is hopefully shown in the diagram. Great Scotsman! A Structure Diagram for BASIC? Why not?!

CONSTANTS AND VARIABLES

When a program is RUN some values will change. They are called variables. Other values do not change, and usually it is crucial that they do not. Such values are called Constants. BASIC cannot distinguish between the two but the concept can still be used. Declare all Constant values at the beginning of the program. Looking at line 130, N is the maximum number of records the program will handle (change to suit), P is the number of fields per record. Declaring Z=0 may seem odd. After all, BASIC sets Z to zero when the program is RUN, so why do it again? Well, Z is an important parameter - it stores the number of records that have been created. Declaring Z near the beginning of the program is a reminder of Z's importance. ST=10 is the number of records printed per page during two-column printout; S=64 is screen-width; F$=CHR$(12) is printer form-feed; TB=40 is second Tab; and V=2 is a DOS device number - tape-users leave this in. CHR$(13) is the code for carriage return. Notice that array N$ has not been formally declared. BASIC doesn't mind as long as the array does not go beyond N$(10). General purpose variables are Q,Y,R,I and Y$. A program will run faster if frequently used Constants and Variables are declared before any other.

MAIN MENU

Think of the Main Menu as the Controller for the program. All program blocks have one entry point from Main Menu. Each program block ideally has only one exit point, back to Main Menu. Any other excursion from the block is controlled with GOSUB. This structure prevents a block from taking control and running amok in the program. If you are frequently using a TRACE utility to discover what your BASIC programs are doing then follow this simple remedy - remove the GOTOs!

A choice from the Menu is selected by number. The method is simple and economical on code. It fits in perfectly with the ON ... GOTO command used in line 280. Selecting '1' forces a branch to line 340, selecting '2' forces a branch to line 410, and so on.

POINTS ARISING

Why the GOTO in line 280 and not GOSUB? BASIC uses a Stack on zero page from $0100 to $01FF, starting at $01EF. FOR .. NEXT pushes 16 bytes on the Stack and GOSUB uses 7. It is evident that the Stack can soon fill when nesting GOSUBs and FOR .. NEXT loops. Experience reveals that for long programs GOTO is best if Stack overload is to be avoided.

To convert a number to a string use STR$ - eg:

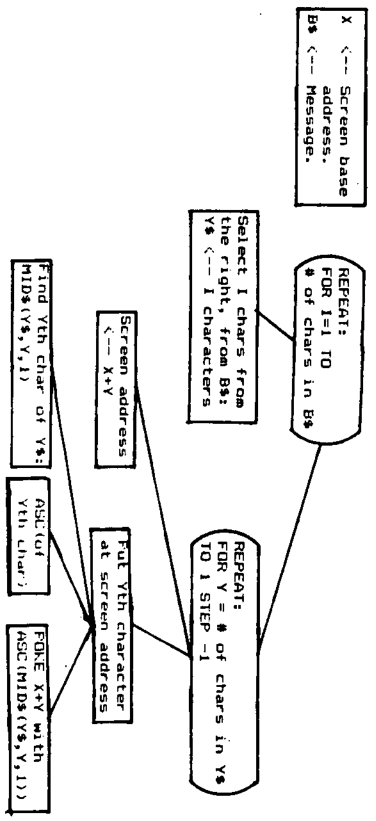
```
10 N=-10 : N$=STR$(N) :  PRINT
"N is " N$
```

POKE can be a mysterious command. It's just a way of making a character appear on the screen. Every character has an ASC number associated with it. For example, 'A' is 65, 'B' is 66 and 'a' is 97, 'b' is 98 and so on. To put a character on the screen we tell BASIC what the character's ASC code is. The first number is the screen address where the character will appear - eg:

20 POKE 54165,65.

Notice that 'A' is at address 54165 for only a fraction of a second - the screen scrolls up several lines before returning control to Immediate Mode.

Horizontal Scroll Logic


X <-- Screen base address.
B$ <-- Message.
REPEAT: FOR I=1 TO # of chars in B$
Select I chars from the right, from B$: Y$ <-- I characters
Screen address <-- X+Y
Find Yth char of Y$: MID$(Y$,Y,1)
ASC(of Yth char)
REPEAT: FOR Y = # of chars in Y$ TO 1 STEP -1
Put Yth character at screen address
POKE X+Y with ASC(MID$(Y$,Y,1))

```
40 B$="        OTAIO MAILING LIST 4/84
   by LZJ"
50 X=53509: FOR I=1 TO LEN(B$)
60 Y$=RIGHT$(B$,I):FOR Y=LEN(Y$)
   TO1STEP-1:POKE X+Y,ASC(MID$(Y$,Y,1))
70 NEXT Y,I
80 REM
130 N=200: P=5: Z=0: ST=10:S=64:
    F$=CHR$(12):R$=" ":S$="STOP":
    H$="HELP"
140 DIM D$(N,P): C$=CHR$(13)
150 N$(1)="Name   ": N$(2)="Address ":
    N$(3)="City    "
160 N$(4)="State ": N$(5)="Country ":
    N$(P+1) = "Record #"
170 REM
180 REM Main Menu
190 PRINT !(28): PRINT TAB(11)
    "* When in trouble type:- HELP *":
    PRINT
200 PRINT:PRINT:PRINT"Records free ==>"
    N-Z" from "N:PRINT
210 REM
220 PRINT "   MAIN MENU":
    PRINT "   ---------": PRINT
230 PRINT "1> LOAD File":
    PRINT "2> SAVE File":
    PRINT "3> PACK Records"
240 PRINT "4> FIND":
    PRINT "5> EDIT":
    PRINT "6> SORT":
    PRINT "7> PRINT"
250 PRINT "8> APPEND":
    PRINT "9> LIST Erased Record #":
    PRINT "-> END"
260 PRINT:PRINT"CHOICE?";:GOSUB 310:
    IF Y$="-" THEN 1970
270 IF Y=0 THEN 190
280 PRINT !(28):ON Y GOTO 340,410,480,
    570,880,1280,1490,1710,1830
1950 REM
1960 REM Restart
1970 PRINT:FORC=1 TO 5:PRINTTAB(10)
     "<<< To RESTART type:-GOTO190 >>>"
1980 NEXT : POKE 2073,173 : END :
     REM CTRL-C (disk) back on.
1990 REM
2020 PRINT : GOTO 200
```

See you next month!

★

NEWDIR

By: Timonthy Hu
1601 E. Lincolnway
Cheyenne, WY 82001

Here's a quick and dirty way to clean up directories when, for instance, you've copied a disk. I got this idea from OSI's V3.3 BEXEC* (this works for V3.3 ONLY!). In this program, lines 5020 to 5050 calls to memory track 11, sectors 2 through 5. Taking a look at this track using the "EXAM" command, you can see that sectors 2 and 3 contain what looks like a blank directory for a disk with no files at all except that for the directory on track 12. Do a "EXAM D200=11" and see for yourself -- 8 inch floppy users should look at track 7, not 11. Sector 6 contains a directory for option 6 in BEXEC*. So, to get a blank directory, do the following:

1. Copy DOS or just initialize. Enter "Exit" to get to DOS command level.

2. Place a disk with OS65D in drive (disk must have a 'valid' track 11 as described above).

3. Enter the following in the order they appear (from the kernal)

```
CALL 4000=11,2  (07,2 for 8
     inch systems)

CALL 4200=11,3  (07,3 for 8
     inch systems)

CALL 4400=11,4  (07,4 for 8
     inch systems)

CALL 4600=11,5  (07,5 for 8
     inch systems)
```

4. Place your new (just copied or initialized) disk in drive.

5. Enter the following in the order they appear:

```
SAVE 12,1=4000/1  (08,1 for 8
   inch)

SAVE 12,2=4200/1  (08,2 for 8
   inch)

SAVE 12,3=4400/1  (08,3 for 8
   inch)

SAVE 12,4=4600/1  (08,4 for 8
   inch)
```

6. Verify the new track.

Presto! What you get is a blank directory with one entry for the directory on track 12. NOTE: If you don't transfer the contents of track 11,4 and 11,5 to 12,3 and 12,4 you won't be able to open files and the machine will hang if you attempt to do so. These two sectors contain the GET-PUT overlays needed by BASIC. So put them on every disk!

This simple method eliminates the need for directory zeroing programs and the need to go through the "DELETE" program routine a million times. I'm sure this will help all OSI (65D) users. I guess the blank directories on track 11 are "undocumented features"!

To make things even quicker, just enter and save this program:

```
10 REM NEWDIR (NEW DIRECTORY)
   By Tim Hu for PEEK(65)
   readers 14JUN84

20 PRINT!(28)"Please insert
   OS-65D disk in drive and
   hit any key";:DISK!"GO
   2336"

30 DISK!"CALL 4000=11,2":REM
   07,2 for 8 inch systems

40 DISK!"CALL 4200=11,3":REM
   07,3 for 8 inch systems

50 DISK!"CALL 4400=11,4":REM
   07,4 for 8 inch systems

60 DISK!"CALL 4600=11,5":REM
   07,5 for 8 inch systems

70 PRINT:PRINT"Please put new
   disk in drive and hit any
   key";:DISK!"GO 2336"

80 DISK!"SAVE 12,1=4000/1:REM
   08,1 for 8 inch

90 DISK!"SAVE 12,2=4200/1:REM
   08,2 for 8 inch

100 DISK!"SAVE 12,3=4400/1:REM
    08,3 for 8 inch

110 DISK!"SAVE 12,4=4600/1:REM
    08,4 for 8 inch

120 END
```

Now user option 2 in BEXEC* to create your files. When making an entry for "OS65D3", DO NOT ANSWER "YES" OR HIT RETURN

Congratulations Europe - You Now Have a DBI Distributor

DBI Welcomes P.M.C. ApS

P.M.C. ApS offers complete Dealer Support:

- Hardware Support (D.B.I. products in stock)
- Software Support (Specialized programs)
- Technical Support (Repair/Installation)

For Replacement and Upgrading O.S.I.* (65U*) 200 Series Computers with D.B.I. products.

P.M.C. ApS Professionals are:
Joergen Clausen
Niels Koldborg
Carsten Sillemann

P.M.C. ApS
Taarnfalkenvej
DK - 2650 Hvidovre
Copenhagen, Denmark

Telefon: 45-1-49 30 66
Telex: 42563 Fulvic DK

*O.S.I. and 65U are Trademarks of ISOTRON Inc.

Wir gratulieren - nun hat auch Europa eine Vertretung fuer DBI

DBI heisst P.M.C. ApS willkommen

P.M.C. ApS liefert vollstaendige Verkaufsunterstuetzung:

- Geraete Service (D.B.I. Componenten an Lager)
- Software (Spezielle Kundenprogramme)
- Technische Unterstuetzung (Reparaturen-Installationen)

Wir ersetzen und ergaenzen O.S.I.* (65U*) 200 Serien Computer mit D.B.I. Componenten.

Das zustaendige Personal der P.M.C. ApS sind:
Joergen Clausen
Niels Koldborg
Carsten Sillemann

P.M.C. ApS
Taarnfalkenvej
DK - 2650 Hvidovre
Copenhagen, Denmark

Telefon: 45-1-49 30 66
Telex: 42563 Fulvic DK

*O.S.I. and 65U sind registrierte Markenzeichen von ISOTRON INC.

Félicitations Europe - Vous avez, maintenant, un distributeur DBI.

DBI souhaite la bienvenue ā P.M.C. ApS

P.M.C. ApS offre un soutien concessionnaire total, que ce soit en:

- machines (matériel D.B.I. en stock)
- ou en software (programmes spécialisés)
- avec appui technique (réparations/installations)

pour le remplacement et l'amélioration des ordinateurs O.S.I.* (65U*) série 200 avec produits D.B.I.

Les agents professionnels P.M.C. ApS sont:
Joergen Clausen
Niels Koldborg
Carsten Sillemann

P.M.C. ApS
Taarnfalkenvej
DK - 2650 Hvidovre
Copenhagen, Denmark

Telefon: 45-1-49 30 66
Telex: 42563 Fulvic DK

*O.S.I. et 65U sont des marques déposées de ISOTRON Inc.

DBi, Inc.
p.o. box 7276
denver, co 80207
(303) 428-0222

when asked if you want these tracks initialized. Otherwise, the new directory will get trashed along with whatever was on tracks 1 to 11 as well as 13 (OS65D3 is 14 tracks long). You may have to delete the file "DIRECT" before you can make the new file, "OS65D3".

Later ya all, and have fun with tracks 7 and 11!

★

DUAL HARD DISK DRIVES

By: ISOTRON, INC.
140 Sherman St.
Fairfield, CT 06430

In response to numerous questions about how to add DEV"F drives to the 250J machines, OSI has passed along the following information. This is not a mod for the uninitiated. If you are not totally familiar with the subject, first find someone who is. Make sure that any HD data is completely backed-up before beginning.

Parts needed for configuring dual hard disk drives for use with 250J:

Qty	Description	Part Number	Where Used
4	DS8831 integrated circuit	100497P1	U9,U15
8	16 pin DIP sockets	100415P3	U9,U15,J4,J6
1	12 pin right angle male molex connector	SC-12ME	J8
1	interconnecting cable assembly for connecting DEV"E to DEV"F	100686G1	

Connectors at J4,J6, and J8 are normally not installed, however, some earlier production runs may have included them.

Refer to figure 1 for the following steps.

Configuring DEVice E:

1. Remove OSI 592 printed circuit board from hard disk drive.

2. Install 16 pin DIP sockets at J4 & J6.

3. Install 12 pin right angle male molex connector at J8. (Not necessary on F drive).

4. Remove integrated circuits at U10 & U16 (IC-75183).

5. Install tri-state line drivers at U9 and U15 (IC-DS8831).

This completes the modifications necessary for DEVice E.

Configuring DEVice F:

1. Perform steps 1 thru 5 as in configuring DEVice E.

2. Cut foil between JP2.2 and JP2.3.

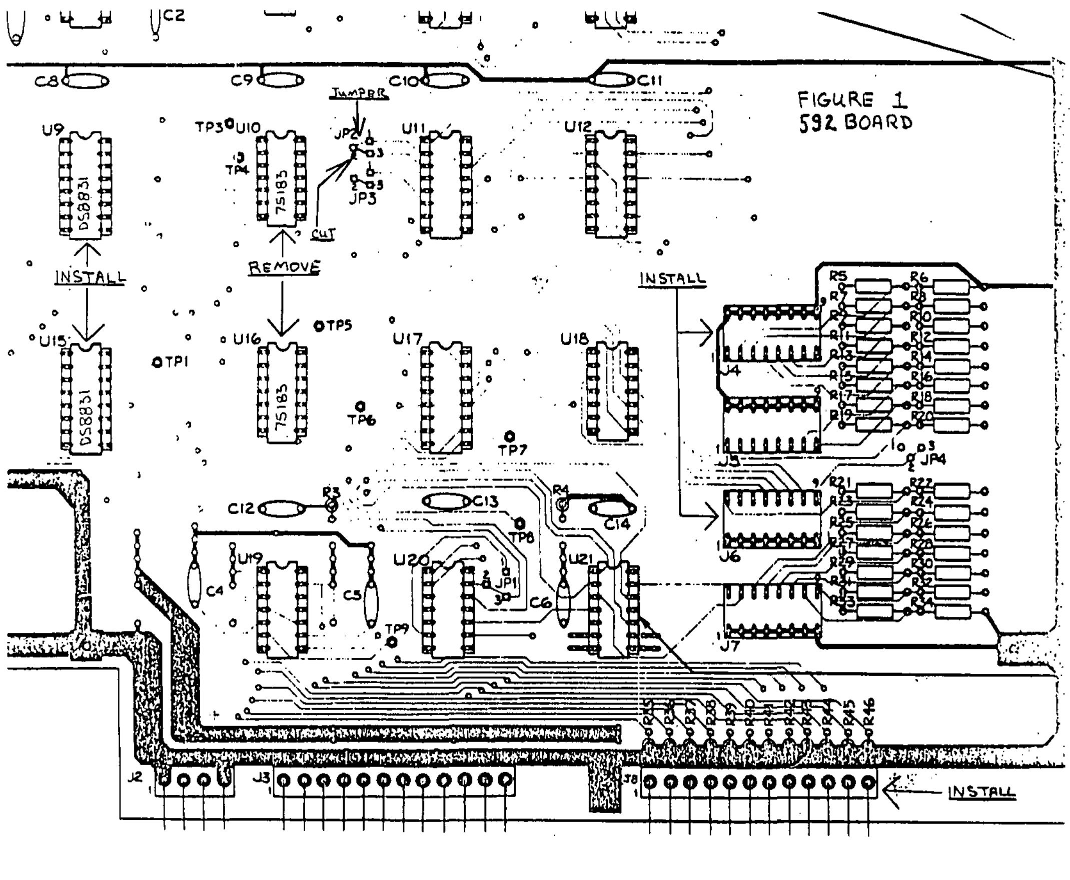
3. Install jumper between JP2.1 and JP2.2.

This completes the modifications necessary for DEVice F.

Connecting the drives:

1. Install one half of the interconnecting cable to DEVice E with the shorter of the two 16 pin DIP connectors going to J6 and the other 16 pin DIP connector going to J4. Be sure to observe polarity, pin 1 of the cable to pin 1 of the socket.

2. Install the twelve pin molex connector to J8 also observing polarity.



D&N MICRO PRODUCTS, INC.

3702 N. Wells St.
Fort Wayne, Ind. 46808
(219) 484-6414

TERMS $3.00 shipping, Foreign orders add 15%, Indiana residents add 5% sales tax.

COMPUTER

MICRO-80 COMPUTER
Z-80A CPU with 4Mhz clock and CP/M 2.2 operating system. 64K low power static memory. Centronics parallel printer port. 3 serial ports. 4" cooling fan. Two 8" single or double sided floppy disk drives. IBM single density 3740 format for 243K or storage, double density format for 604K of storage. Double sided drives allow 1.2 meg on each drive. Satin finish extruded aluminum with vinyl woodgrain decorative finish. 8 slot backplane, 48 pin buss compatible with OSI boards.

MODEL 80-1200 2 8" Single sided drives	$2995
MODEL 80-2400 2 8" Double sided drives	$3495

MICRO-65 COMPUTER
6502 CPU with 2Mhz clock and DOS-65 operating system. 48K of low power static memory. 2 serial ports and 1 Centronics parallel port. 2 8" single or double sided drives. Satin finish extruded aluminum with vinyl woodgrain finish. 8 slot backplane, 48 pin buss compatible with OSI. Will run OSI 65D and 65U software.

MODEL 65-1 2 8" Single sided drives	$2995
MODEL 65-2 2 8" Double sided drives	$3495

BP-580 8 Slot Backplane..... $ 47
OSI 48 pin Buss compatible

MEM-CM9 MEMORY/ FLOPPY CONTROLLER
24K memory/floppy controller card uses 2114 memory chips, 1 8K and 1 16K partition. Supports OSI type disk interface

24MEM-CM9	$325
16MEM-CM9	$260
8MEM-CM9	$180
BARE MEM-CM9	$ 50
Controller on assembled unit add	$ 90

BIO-1600 Bare IO card....... $ 50
Supports 8K of memory, 2 16 bit parallel ports, 5 serial ports, with manual and Molex connectors.

PRINTERS

Okidata

ML82A, 120 cps, 10"	$409
ML83A, 120 cps, 15"	$895
ML84 Parallel, 200 caps, 15"	$1150

C. Ioth

8510AP Prowriter, parallel 120 cps, correspondence quality	$419
8510APD Prowriter, serial	$585
F10-40PU Starwriter, parallel Letter quality daisy wheel	$1319
F10-40RU Starwriter, serial	$1319
F10-55PU Printmaster parallel, Letter quality daisy wheel	$1610
F10-55RU Printmaster, serial	$1610

DISK DRIVES AND CABLES

8" Shugart SA801 single sided	$385
8" Shugart SA851 double sided	$585
FLC-6 6 ft cable from D&N or OSI disk controller to 8" drive	$69
5¼" MPI B51 disk drive with cable, power supply and cabinet. Specify computer type.	$450
FLC-5¼ cable for connection to 5¼ drive and D&N or OSI controller, with data separator and disk switch. Specify computer type	$75

HARDWARE OSI COMPATIBLE

IO-CA10X Serial Printer Port.. $125
Specify Device #3 or #8
IO-CA9 Parallel Printer Port .. $150

CMOS-MEM
64K CMOS static memory board, uses 6116 chips, 3 16K, 1 8K and 2 4K blocks, Partitionable for multi-user, OSI type disk controller, 2 IO mapped serial ports for use with D&N-80 CPU. Ideal way to upgrade from cassette to disk.

64K CMOS-MEM	$490
48K CMOS-MEM	$390
24K CMOS-MEM	$250
16K CMOS-MEM	$200
Controller add.	$ 90
2 IO mapped serial ports add. on assembled memory board	$125
Z80-IO 2 IO mapped serial ports for use with D&N-80 CPU card	$160
FL470 Disk Controller Specify 5¼ or 8" drive	$155

STANDARD CP/M FOR OSI

D&N-80 CPU CARD
The D&N-80 CPU allows the owner of an OSI static memory computer to convert to Industrial Standard IBM 3740 single density disk format and CP/M operating system. Double density disk operation is also supported for 608K of storage on an 8" diskette. When used with a 5¼" disk system 200K of storage is provided. Includes parallel printer and real time clock. Also available for polled keyboard and video systems. Compatible with C2, C3, C4 and 200 series OSI computers.

D&N-80-P	$349
CP/M 2.2	$150
64K CMOS-MEM with D&N-80 CPU card	$450

HARD DISK DRIVER $140
Allows D&N-80 CPU board to control OSI 40 or 80 meg hard disk unit. Will not destroy OSI files. Will also allow for a true 56K CP/M system. Specify 40 or 80 meg drive.

BUSS TRANSFER $135
Allows for D&N-80 and OSI CPU to be in the computer at the same time. Toggle switch provides for alternate CPU operation.

DISK TRANSFER $100
Utility program to transfer OSI CP/M format disk to IBM 3740 single density format. Will also transfer IBM to OSI format.

SYSTEM HARDWARE REQUIREMENTS
D&N-80 CPU, D&N FL470 or OSI 470 controller, 48K memory at 0000-BFFF, 4K memory at D000-DFFF, two disk drive cables.

FORMAT TRANSFER $15
You supply software on 8" diskette D&N will transfer OSI CP/M format to IBM 3740 CP/M format. Can also transfer IBM 3740 CP/M format to OSI CP/M format. Original diskette returned.

3. Install the other half of the interconnecting cable to DEVice F with the shorter of the two 16 pin DIP connectors going to J7 and the other 16 pin DIP connector going to J5. Be sure to observe polarity.

4. Install the twelve pin molex to J8, also observing polarity. Pin 1 of J8 on DEVice E should be the same color wire as pin 1 of DEVice F.

5. The cable assembly connecting DEVice E to the computer remains the same.

This completes all modifications necessary for utilizing dual hard disk drives.

★

COMMUNICATION AND OSI

By: Earl Morris
3200 Washington St.
Midland, MI 48640

I have always been fascinated by the possibilities of communicating via a link between two computers. If the two machines are within 10 feet of each other, the TTL signals can be directly connected. Up to 100 feet, the serial ports of the two machines can be hardwired together. Beyond that distance it becomes difficult to send high speed digital signals as well as impractical to run a wire across town or across country to a friend's computer.

To achieve communication over a greater distance than you can physically string a wire, we need to look at existing communications routes. There are already a number of methods of transmitting audio (voice) signals. The digital logic levels of +5 and 0 volts must somehow be converted into something which will pass over a normal audio communications link. These logic levels are often referred to as "MARK" and "SPACE" in communications articles. In general a logic "0" (space) is converted into one tone while a logic "1" (mark) is converted into a different tone. The frequency of the tones must be chosen to be within the bandwidth of whatever audio link is used. The use of audio tones also restricts the rate at which data can be sent over the audio link. The filter or detector on the receiving end must receive one or often several cycles of the audio tone before being able to determine its frequency. For example, if a 1000 Hz tone is sent, and the filters require four cycles to respond to that tone, each bit sent requires 4 milliseconds. Thus only 250 bits per second can be sent. A higher frequency audio tone and a faster responding filter will allow higher baud (data) rates. There are a variety of different tone combinations in use to send data over audio channels.

One of the most familiar audio devices is the cassette recorder. Back in the early days of home computers before disks were affordable, cassettes were also the medium for program storage. Cassettes were the way to trade programs and communicate with other computers. Just put a cassette in an envelope and drop it in the mail. I have exchanged cassettes with OSI users all over the world. To encourage communications between all computer users, BYTE magazine sponsored a meeting in Kansas City in 1975. From that meeting came the "Kansas City" standard for putting digital data on cassette tape. The March 1976 issue of BYTE explained the standard and urged its use for all home computers. This was a grand idea: every computer could share data with any other by this common format. OSI adapted the "KC standard" cassette interface. Unfortunately, the other major computers of the time, PET and Radio Shack, did not. So in practice the OSI cassette interface can be used only to talk to other OSI users. The "KC" standard consists of using 4 cycles of a 1200 Hz tone and 8 cycles of a 2400 Hz tone to record digital data at 300 baud. Various OSI users have discovered the cassette baud rate can be increased to 600 or even 1200. The tones can be increased from 1200/2400 to 2400/4800 or the number of cycles can be reduced from 4/8 to 2/4.

Another common audio channel is the telephone. Digital data is converted into tones using a "MODEM". The usual baud rate is 300 or 1200. With a modem attached to a serial port, you can communicate with a friend across town, with a remote mainframe computer, or even sign onto one of the many nationwide nets. Of course, you end up paying both the phone company and the net to use this service. Fortunately, standards for the tones used have been set up which everyone adheres to (at least at 300 baud). A telephone is a duplex device: that is the conversion can go both ways at once. With a telephone modem you can be typing a message on the keyboard at the same time the CRT is displaying an incoming message. To allow this, four tones are required. You need one pair of tones to transmit and a second pair to receive data sent from the other end. There is a standard to decide who uses which tone pair. The computer that initiates communications is called ORIGINATE and uses the two tones 1070 and 1270 Hz. The machine answering the phone is called (what else) ANSWER and uses 2025 and 2225 Hz. To communicate, one party must be originate and the other answer. Some inexpensive modems are originate only. These are fine if you are always calling a mainframe computer or network, but will not allow calling a friend across town who also has an originate only modem. Telephone modems have become almost a universal means to communicate computer to computer.

Another audio channel I have become interested in is radio. For many years amateur radio operators have been communicating using discarded teletype machines from the wire news services. These mechanical marvels run at the amazing speed of 45.45 baud using the 5 big Baudot code. This code allows upper case letters, numbers and limited punctuation only. Recently the Federal Communication Commission recognized there was another code called ASCII and has allowed that code to be used in the amateur radio bands. A license is necessary which requires taking a technical exam and demonstrating a knowledge of Morse code. A special license for digital communications has been proposed. But for the present you still need to pass the code test at 5 or 13 words per minute to get an amateur license. The tone standards chosen for the amateur bands are 2125 and 2295 Hz. This is also known as 170 shift since that is the difference in the two tones. If you have a computer, a radio, and a license, you still need a "TU" or teletype unit. This circuit converts digital data into tones similar to a telephone modem. TU's can be purchased or built from plans published in the various ham magazines. The commonly used baud rates are 110 for mechanical models, 33 for teletypes, and 300 for computers. I currently work "RTTY" or radio teletype on a band called two meters (144 to 148 MHz). When using FM and a 2Ø watt transmitter, solid noise free communications can be obtained up to about 5Ø miles. Within that distance is a relay station which amplifies my signal so I can communicate over 1ØØ miles. The relay station is computer controlled and will accept, store and play back files from disk. It also serves as a mailbox for messages and programs. This mode of communications has greatly enhanced my enjoyment of my OSI computer. With more power on different bands, amateurs can communicate computer to computer over large distances with no phone bills to pay.

So enjoy communicating with your computer!!

★

TEC65 REVISITED

By: Michael E. Sherman
12951 Westchester Trail
Chesterland, OH 44Ø26

This is a continuation and expansion of the review of TEC65 from the 65Ø2 Program Exchange that appeared in the October 1983 issue. The original review was written by David A. Jones. Mr. Jones' review made TEC65 sound like it might be a good software investment. In fact, it sounded like a great editor, but that was misleading, it's really much better than that!

As a brief review, TEC65 is a subset of the Digital Equipment Corporation's text editor TECO. It contains the most frequently used commands of TECO with very few changes. It is available from the 65Ø2 Program Exchange at 292Ø West Mona, Reno, NV 895Ø9; and is available for C1P's with HEXDOS, OS65Dv3.2, or OS65Dv3.3. It is also available for the C4P with OS65Dv3.3.

The service is wonderful! It's the kind we all pray for. Questions are referred directly to the author of the OSI version, who is marvelously patient and helpful. He responds quickly, and answers all questions completely and clearly.

SURPRISE!

My first surprise was that I received a smart terminal package with a text editor, not just the text editor that I expected. I never tried the smart terminal program, TERM, that originally accompanied TEC65. Shortly after I received the response to my first question, I received an offer for two other versions of TERM; TERM24, and TERM48. As you might expect, these are specialized for twenty four and forty eight character displays. I have only used TERM48 and found that it fills my needs completely. The top of the screen displays flags for full/half duplex modes, auto linefeed on/off, and data capture on/off. These functions, and the terminal mode, can be toggled on and off with control keys. A display is always maintained showing the amount of memory used and remaining for the storage of captured data. Operating system commands can be issued without exiting the terminal program. Captured data can then be saved to disk for later reference or editing with TEC65 making a complete and useful system.

TEC65

The instructions were adequate for a complete newcomer to TECO. And, with a brief learning curve, it can become quite comfortable to use. The OSI specific instructions were not as complete as they might have been, but have since been upgraded to include more information for altering the behavior of several functions. Among these controllable features are the number of spaces inserted by the tab function and the device selected for I/O by the print command.

One of the most useful features of the system is not mentioned at all. This is the ability to write & edit BASIC or Assembly language programs using TEC65. In order to accomplish this, one must be able to translate files between BASIC's token format and a ASCII text format and back again. Although this is not given anywhere in the documentation, it is quite simple to do. It is accomplished as follows:

BASIC TO ASCII

Load the BASIC program (as if to run)

Set device #5 pointers "DISK! "MEM nnnn,nnnn" *

Type "LIST#5"

Type "PRINT#5,CHR$(26)"(a control Z)

Save the text to disk "DISK! "SA TT,S=nnnn/8" where TT is the location of the text file.

ASCII TO BASIC

Make the last line of the BASIC text file "DISK!"IO Ø2, Ø2"

(this will turn off device #5)

Save the file to disk (TEC65 EW command)

Boot back to BASIC and enter the immediate mode

Set device #5 pointers "DISK! "MEM nnnn,nnnn"

Load the program into RAM "DISK!"CA nnnn=TT,S"

Then transfer the text to BASIC

"DISK!"IO 1Ø,Ø2"

Introducing
SCRIBE
WORD PROCESSOR
OS-65U 1.42 < Floppy/Hard Disk, Level 1 or Level 3
and DENVER BOARDS
✱INTERFACED TO OS-DMS FILES
✱AUTOMATIC WRAP AROUND
✱COMPLETE EDITING CAPABILITIES
FULL CURSOR CONTROL
INSERT & DELETE TEXT
SEARCH/SEARCH & REPLACE
✱USER FRIENDLY MANUAL
✱AND MUCH MORE
IHS COMPUTER SERVICES
Route 1 Box 201B Port Republic, VA 24471
(703) 249-4833
$195.00

OSI repairs
C-2, C-3, & CD Series
200 Series
•board level service on:
•power supplies
•8" floppy drives
•cpu, memories, etc.
•gold molex contacts
•custom printer cables
(1 week turnaround typical)
Sei
Sokol Electronics Inc.
474 N. Potomac St.
Hagerstown, Md. 21740
(301) 791-2562

* note: nnnn must be calculated as the indirect file boundary. That is, about half way through the systems free RAM space.

These two operations are very useful indeed. They can be used for doing fast editing of programs. For example, if you need to merge two programs and they happen to use a common variable name, one of the programs can be transferred to a text file, then the variable name can be changed throughout the program with a single command. When using TEC65 for creating and editing Assembly language source files, care must be taken to insure proper vertical columns in the assembled source listing. Editing a source code is still far easier than what OSI refers to as an editor for the Assembler.

Of course, an obvious application for this is to use it in conjunction with the terminal software. Files can be translated into text files and transmitted over the phone lines with a modem. I have been told that the OSI SIG on CompuServe has a large number of programs for OSI, and will probably welcome any of your software contributions. Share your creations!

WAZZAT CORNER!

By: L. Z. Jankowski
Otaio Rd 1 Timaru
New Zealand

Many OSI users have purchased one of the new Monitors as a 2716 EPROM. If you have two new Monitors it becomes tedious swapping the EPROMS when you want to switch from one Monitor to the other. How about running both Monitors in the same address space, changing from one to the other at the flick of a switch? It can be done using a 4K 2732 EPROM. An EPROM only stores data, not addresses. An EPROM is read from an address provided by the CPU. By switching pin 21 high or low a 2732 EPROM can be 'fooled' into releasing either of its 2K blocks of memory - precisely what is required. This is what you do: program the 2732 with 2 Monitors; bend out pin 21; take a double throw switch; solder a wire from pin 21 to the center pole of the switch; solder a wire from one side of the switch to Ground; solder the other side of the switch to 5V; remove old Monitor and replace with 2732; attach switch to some suitable point. Remember, the 2732 will only function in a socket in which a 2716 is already functioning correctly.

Contributions from readers to the 'Wazzat Corner' are welcomed.

READER PROFILE

I'm no writer, but if you want a run down on my experiences, here goes.

I have a C4P with dual 8" disk -- well my machine keeps taking on various configurations. Right now I'm using GENERIC's MEM+ board with high resolution graphics. I purchased the bare boards and populated them myself. In this way I get to know the workings fairly well. My career was in Electronics and so it has become a hobby and C4P became a hobby too. I started out with an ELF which used RCA's 1802 CPU and had switches for each bit. And so to an 8K C4P with ROM BASIC.

I added a 527 memory board, then installed an RS232 output for my OKIDATA 82A printer. OSI had stopped selling bare boards, so I went to MICRO-INTERFACE for their memory+ board and installed memory to 48K, the disk drive circuit and the parallel printer interface. Things were OK except I had problems with switching between ROM BASIC and DISK BASIC. Then Generic came out with bare boards for their high resolution color graphics board, so I purchased one of those and the memory+ board to go along. My backplane is full of boards, but everything is working well. I installed V3.3 and now have a pretty sophisticated instrument.

Coming up is a MODEM and an EPROM programmer, and maybe I will learn to talk a little in FORTH. Practically all the information I get now-a-days comes from PEEK(65). All the others seem to have abandoned us entirely, so please keep up the good work.

Gerald M. Van Horn
Junction City, OR 97448

LETTERS

ED:

I just got my disk up and running on my C4P 32K. Now I would like to try my luck at writing a word processing program. What I would like to know is where I can find the keyboard and the video routines. I really like the edit functions that are on disk #5. Also, is there a routine that can copy the entire CRT? Boy, would that be a big help!

I'm not too familiar with DISK yet, so I really can't contribute to PEEK(65). I have about a million questions to ask, but will only ask what I can't figure out myself. It really is embarrassing to ask some questions if you know what I mean!! I know we all have to start somewhere, but I would venture to guess that there are others who feel as I do. I will ask more questions next time.

Keep up the good work at PEEK(65)!!

James A. Antonelli
Sugarloaf, PA 18249

James:

Under 3.3 for the C4P, the entry point for the video driver is at $33C0 and the keyboard poll it at $3590. However, those are the specific video device driving routines. For a word processor, it is better to use the standard OS-65D routines which support the active input device # at $2331 and the active output device # at $2332. Therefore, I suggest you use the following:

$2336 INCHNE
Waits for a keypress from the user and returns it in the ACC. but does not echo the character to the console.

$2340 INCH
Same as INCHNE, but DOES echo character to the console.

$2343 OUTCH
Outputs character in the ACC. to all active output devices.

Rick Trethewey
SYSOP, OSI-SIG
CompuServe

* * * * *

ED:

I know my renewal is late, but I have been debating with myself whether to renew or not, as I cannot get much out of my C1P MF. During the past year, I have spent $287 (decimal) in repairs (not counting shipping costs of about $50.) Included in this is an RS232 interface for a printer which Jerry Travis (The Computer Shelter) installed but will not run my printer (Microline 82A) which I bought on his advice. The OKIDATA technicians have been trying very hard to be helpful, but none of the connections they have suggested work. They are now trying to contact Jerry to get it straightened out, but he is away. ISOTRON could not help me when I sent them the computer. They got it to run an EPSON printer, but that didn't do me any good. So you see, unless I get the printer running very soon I will have to say goodbye to my C1P, OSI, ISOTRON, and PEEK(65).

Also, I bought from you the 65V PRIMER, which tries to be very "friendly", but is about as helpful as a friendly moron. I typed in the machine language routines on pages 5 and 7, but they do not run. Thus I am stuck, because I can't continue unless I see what happens with those routines. The author plays it very cute in keeping the results a secret. Can you try it for me?

As for PEEK(65), it seems best suited to those who understand the technical aspects of computers, which I do not, and Assembly Language. In which mode do you type in Assembly Language, and how do you save it? Can you recommend a book which will explain it to someone who does not understand it, unlike those books I have seen?

I hope you can help me. (Do you know of any C1P owners in my area?)

Milton Goodglass
Studio City, CA 91604

Milton:

Trying to resolve printer hook up problems from this distance is nigh impossible, but suffice it to say that one of the glories of OSI machines is their ability to be made to work with just about anything. Thus, there appears to be no reason why your set-up cannot be made to work. Lean on Jerry or one of his associates. You might check "Beginner's Corner" in the July issue, and also Assembly Language Programming by Rick Trethewey.

Re the PRIMER, the first program merely puts the characters you type on the screen. The second displays the ASCII bit values of the letters you type. The results, in both cases, depend entirely upon what you type. We tried them again, and they work. Are you sure you typed ".0000G." to execute the programs?

We are not aware of a C1P owner in your immediate zip code area, but maybe someone reading this and not too far away will make themselves known to you directly or via PEEK.

Peek Staff

* * * * *

ED:

Here are changes for Wizards City to let it run on a system with ASCII keyboard. They allow much easier typing.

Also, I need help on WP6502 cassette ...does anyone know how to change from polled to ASCII keyboard?

```
"
WIZARDS CITY BY AURORA SOFT-
WARE ASSOCIATES 1980

CHANGES BY BOB GROOME SOMETIME
IN 1983

30 POKE11,232:POKE12,31:
   M=30900

900 POKE530,1
905 P=PEEK(KY)
908 SE=RND(1)
910 IF P=160 THEN POKE
    530,0:RETURN
920 GOTO905

20100 POKE 5530,1:P=PEEK
   (KY):V=0:M=0
20110 IF P=177 THEN V=1:
   GOTO20172
20120 IF P=178 THEN V=2:
   M=1:GOTO20172
20130 IF P=179 THEN V=3:
   M=-1:GOTO20172
20140 IF P=180 THEN V=4:
   GOTO20172
20150 IF P=181 THEN V=5:
   GOTO20172
20160 IF P=182 THEN V=6:
   GOTO 20172
20170 IF P<176 OR P>183
   THEN20100
20172 IF V=6THEN PS=4
20173 IF V=5THEN PS=8
20174 IF V=4THEN PS=16
20175 IF V=3THEN PS=32
20176 IF V=2THEN PS=64
20177 IF V=1THEN PS=128

DO YOU KNOW WHAT THE 'PS' IN
LN # 20180 DOES?

9 KY=57343
```

Bob Groome
Richmond, IND 47374

* * * * *

ED:

Just received the June 1984 issue, and I wish that you would publish the complete address of people whose letters you print so that readers could write to the person without having to place a phone call first to get the address.

T. J. Hirasuna wrote in April asking for program mod instructions to make Apple programs run on his OSI, and in June he says you misunderstood - he wants to modify the OSI to directly run Apple programs. This is as big a waste of time as the Apple is. Anyone who hangs in there with his/her OSI knows that the OSI is much faster in every way imaginable, especially in handling disk files if you use the Manley mods. The Apple is really a waste of time to a serious user, and people who claim to be serious users on an Apple just have never used a real computer!! Keep up the good work.

Paul Rainey
Villa Park, IL 60181

Paul:

The reasons we don't publish the complete address of people who write letters to PEEK(65) are: 1. Some do not wish to have it published. 2. It encourages communications through PEEK, allowing us to print comments and answers which benefit all our readers.

Peek Staff

AD$

Send for free catalog, Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005.

* * * * *

C1P SMART TERMINAL ROMS. Replace the Monitor ROM and give your C1P Telecommunication compatibility. ROMTERM has 24 column Display and Disk Boot. ROMTERM II has 48 column display support. Either ROM has corrected keyboard and Basic Program Editing. Only $49.95. MICRO INTERFACE, INC., 3111 So. Valley View Blvd., Suite I-101, Las Vegas, NV 89102, (702) 871-3263.

* * * * *

For sale OSI C3-OEM (1-2 MHZ) with dual double sided 8 inch Shugart floppy drives, 48K plus 8K for CP/M, Televideo 912C terminal, IDS 440 printer, Sola power conditioner, 4 new head cleaner disks, disk storage tray with 73 disks. Software: OS65D V3.2, OS65U V1.42 and V1.44, OSDMS nucleus and sort, C3 utilities, WP-6502 word processing, and misc other programs. Manuals. All in excellent condition. $2500 or best offer. John O'Malley, Miami, Florida (305) 759-6468 evenings or weekends.

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:

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

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

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95 $ ________
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at	$15.00 $ ________
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00 $ ________
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00 $ ________
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00 $ ________
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00 $ ________
( )	**Sanders Machine Language Sort/Merge** for OS-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00 $ ________
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00 $ ________
	BOOKS AND MANUALS (while quantities last)	
( )	**65V Primer.** Introduces machine language programming.	$4.95 $ ________
( )	**C4P Introductory Manual**	$5.95 $ ________
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95 $ ________
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95 $ ________
( )	**How to program Microcomputers.** The C-3 Series	$7.95 $ ________
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95 $ ________

( ) Cash enclosed ( ) Master Charge ( ) VISA
Account No. ________ Expiration Date ________
Signature ________
Name ________
Street ________
City ________ State ________ Zip ________

TOTAL	$________
MD Residents add 5% Tax	$________
C.O.D. orders add $1.65	$________
Postage & Handling	$ 3.50
TOTAL DUE	$________

POSTAGE MAY VARY FOR OVERSEAS