PEEK (65)

The Unofficial OSI Users Journal

$1.75

NOVEMBER 1984
VOL.5, NO.11

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

INSIDE

USER PROG. ERROR RECOVERY 2
6502 ASSEMBLY LANG. PROG. CLASS 6
OSI ROM ROUTINES 8
BEGINNERS CORNER 10
SOFTWARE LISTINGS 11
WAZZAT CORNER! 13
READER SURVEY 14
POS VERT. APPLICATION REVIEW 17
KPS BUSINESS SYSTEMS REVIEW 20
LETTERS TO THE EDITOR 21

Column One

Last month I promised news of DBI and OSI's 515 boards. You will get it, but not in as much detail as I would have liked. Regrettably, Column One is the last thing to go in PEEK and thus is restricted to the space you see here. So, hopefully, this will whet your appetite for the December issue, where we can devote more space to these subjects.

But first a sparce few words about this issue. This is the last of Leroy Erickson's ROM articles. Leroy will be back with more! Beginners, please let us know if we are on the right track.

If you have been following along with the KPS series, you will find it joined by the piece on Jim Sileo's POS system. Does all this sound like we are leaning toward the business users? Well, yes and no! From the business point of view, we have been neglecting them and must steer in that direction. Does that mean that we are forsaking the "P" users. Heavens no! You are the ones who got us where we are.

On that subject, OSI reports that the sale does not mean they are closing out "P" things. They will have parts and things for sometime to come. Now to the promised subjects.

Over the months, more and more comment has come our way about DBI's multi-processor board mods for the OSI box. Rumors say that machines with the Denver Boards (DB-1) "just fly". We had the privilege of visiting DBI's Denver headquarters and were treated to an all-day session from factory to the last technical detail. All very impressive. This is the tight-knit organization that came up with the multi-processor idea.

Close behind the DB-1 came the DP-1 (printer board) with 4 serial and 2 parallel ports for either user or system use, depending upon which end is plugged into the backplane. The big boost came with the introduction of the DS-1 (host adaptor) that converts the OSI buss to SASI for disk storage. This means that most of the garden variety disks (hard and floppy - 2 each) can be added.

Next came the real surprise. Well, not really, It seems only natural to put all this tech. in a box. That's the DBM-1. A tiny (13hx15x15) box will hold 160 MB of hard disk, two 5.25" floppies and eight users. And it really is quiet. It comes with their own DB-DOS operating system which picks up where "U" left off, adds more reserved words and functions and is planned to be the stepping stone into the 3rd generation machine with UNIX now on the drawing boards.

The above just scratches the surface, but you will be reading more about DBI and their machine in future articles. But from what I saw, they are well organized, energetic and conscientious about the well-being of the end-user. One certainly gets the feeling that they will be successful and we wish them well.

Meanwhile, things have not been quiet at OSI either. In their booth at Comdex will be the 515 boarded machines, a new release of TurboDos with multiple random sized block transfers that will speed up (several times) the handling of larger files and the FIND command, the 5.25" hard disks that run faster than the 14" because of smart controllers and hopefully a sneak preview of the new 3rd generation machine to put OSI back in the lead. Delivery is not fixed, but we are betting on spring. It will have a 68000 family CPU, run UNIX 5.2 (garden variety ATT) and OS-U (no POKEs, PEEKs or ML please). OSI says that your 2 MHz OS-U machine will out-speed any micro running UNIX on the market. Well, this new effort might just change that too.

Oops! For now, the 515 is either a 2 or 4 - user computer on a board, except for hard disk controller. That's right, CPU, memory, FD controller and I/O. The result - cheaper, more reliable and easy to repiar machine. I haven't checked, but it might even go in your C4P.

Oh, for more room.....!

USER PROGRAMMABLE ERROR RECOVERY

By: David A. Weigle
108 N. Missouri Ave.
Morton, IL 61550

The capability to direct both BASIC errors and disk errors to a routine at line 50000 of a BASIC program was introduced with release 1.3 of OS-65U. This paper discusses the ways in which an error recovery routine may be employed to either recover from a BASIC or disk error or report its occurrence. It also presents a method which can be used to recognize a printer "not ready" condition and communicate with the operator.

Before proceeding, it is necessary to point out that the information contained in this paper is not based on an exhaustive study of how the many types of BASIC and disk errors could possibly be handled by an error recovery routine. While the author has spent much time developing and testing such routines, he has not attempted to deal with all of the possible errors one might encounter when a program is executed. Furthermore, the applicability of the methods presented to releases of OS-65U other than 1.43 should be determined by the reader. The reader is encouraged to research the subject of error recovery and develop the routine(s) which best satisfy his/her needs. Hopefully, this paper will serve as a good starting point for such research and development.

Figure 1 shows the routine the system manuals present as an example of how to decode BASIC and disk errors. This is an error reporting routine, not a recovery routine. It shows the memory locations which contain the data the programmer needs in order to identify error conditions:

Figure 1
Error Reporting Routine (from system manual)

```
50000 PSN=PEEK(11774) + 256*PEEK(11775) : REM get error line
50010 EN=PEEK(18176) : REM get error number
50020 IF EN=23 GOTO 50210 : REM is this a disk error?
50030 :
50100 REM Decode BASIC Error
50110 Z$=CHR$(PEEK(867+EN)) + CHR$(PEEK(868+EN)) : REM error code
50120 ER$="BASIC " + Z$ + "Error in line" + STR$(PSN)
50130 GOTO 50310
50140 :
50200 REM Decode Disk Error
50210 EN=PEEK(10226) : REM get disk error number
50220 Z=PEEK(9832) : IF Z>127 THEN Z=Z-124 : IF Z>63 THEN Z=Z-58
50230 ER$="Device " + CHR$(65+Z) + " Disk Error" + STR$(EN)
50240 ER$=ER$ + " in line" + STR$(PSN)
50250 :
50300 REM Report the Error
50310 PRINT : PRINT ER$ : PRINT
50320 END
```

Table 1
BASIC Error Numbers and Codes

Number	Code	Description
1	NF	NEXT without FOR
2	FS	Full stack; too many nested FORs and GOSUBs
3	SN	Syntax error
4	NR	Not ready; printer
5	RG	RETURN without GOSUB
7	OD	Out of data; more READs than DATA statements
9	FC	Function call
11	OV	Arithmetic overflow
13	OM	Out of memory
15	US	Undefined statement
17	BS	Bad subscript; outside array dimension range
19	DD	Double dimension; array DIMensioned twice
21	/0	Division by zero
23	DV	Disk error; error code is in location 10226
25	TM	Type mismatch; string mismatched to numeric
27	LS	Long string
28	SS	Semaphore stack overflow
29	ST	String temporaries; expression too complex
31	CN	Continuation error
33	UF	Undefined function

Notes:
Error number (EN) is in location 18176
Error code is in locations 867+EN and 868+EN

1. The program statement number (PSN) being executed when the error was encountered can be determined from locations 11774 and 11775.

2. The error number (EN) is stored in location 18176.

3. If the error number is 23, then a disk error is being reported. The code associated with the disk error can be found in location 10226.

4. A two character alphanumeric code for the error (e.g., SN for syntax error) can be found in a table using the error number as an offset from locations 867 and 868.

Table 1 relates the error numbers to the two character error codes and provides a brief description of each error condition.

FLAG 9 and FLAG 23 commands are used to "activate" the line 50000 error recovery routine. If FLAG 9 is used, only disk errors will be directed to the error routine; BASIC errors will result in entry to the immediate mode after the system has displayed an error message. FLAG 23 will make both BASIC and disk errors available to the error routine. Note: FLAG 23 must be specified in order to use the error recovery routine for handling printer not ready situations as this is treated by the system as a BASIC error.

Continued

published monthly
Editor - Eddie Gieske
Technical Editor - Brian Harston
Circulation & Advertising Mgr. - Karin Q. Gieske
Production Dept. - A. Fusselbaugh, Ginny Mays

Subscription Rates	Air	Surface
US		$15
Canada & Mexico (1st class)		$23
So. & Cen. America	$35	$27
Europe	$35	$27
Other Foreign	$40	$27

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 (301) 363-3268

An error condition can be classified as being either recoverable or non-recoverable. Further, we can consider non-recoverable errors to be either fatal or bypassable.

If the door of a disk drive is open when the drive is accessed to OPEN a file, a not ready condition occurs. This is a recoverable error as the error routine can return to the line of the program being executed at the time the error was encountered and re-attempt the file OPEN process after the operator has closed the door of the disk drive.

A syntax error is a non-recoverable, fatal error. The error recovery routine may inform the operator of the error, but it cannot resume program execution.

An arithmetic overflow condition may be a non-recoverable, but bypassable error. To keep the program running, the error routine might log information about the error (on the terminal, the printer, or in a disk file); alert the operator to the error; and, bypass the record (assuming an input disk file containing bad data).

Although the possibilities of recovering from a BASIC error and resuming program execution would appear limited, the error recovery routine can be used for such things as:

1. Notifying the operator of the error and where in the program it occurred.

2. Saving and/or displaying vital information.

3. Instructing the operator as to what steps to take in gathering documentation for problem analysis.

Directing disk errors to the error recovery routine provides the programmer with a means of not only analyzing error conditions and recovering from some of them, but also preventing OS error messages from destroying formatted screens. The programmer can identify such conditions as a disk drive not ready or the wrong disk in a drive; notify the operator of the situation via a message displayed in a pre-determined screen area; and, proceed with program execution following corrective action by the operator. In the event of a fatal error, the disk error routine can do the same things as those described above for BASIC errors. (Note: the examples of error recovery routines in this paper do not show messages displayed in special screen areas.)

It goes without saying that an error recovery routine will be only as good as the data it has to work with. In many cases it is not sufficient to merely know the type of error encountered. It may be necessary to know in what portion of the program the error occurred, what variables or array elements are involved, how many times the error has previously occurred during this execution of the program, etc. These kinds of data could dictate the error recovery routine's course of action.

For example, to access a particular disk file, the operator must enter the password of the file which is used in the OPEN command (refer to Figure 2). If the correct password is entered, the file is OPENed and processing begins. If the password is incorrect, a file error 131 (executability violation) occurs when the OPEN command is issued, and the error recovery routine is invoked. We will assume that the operator has three chances to supply the correct password. The first two times an incorrect entry is made, the error routine informs the operator of the mistake; it then proceeds to the routine for entering the password. On the third occurrence of the error, the error routine terminates the program.

Although we would more than likely handle data security and file access in a better manner, the above example serves to illustrate how the error recovery routine might handle the same error differently depending upon the circumstances in which the error was encountered. Note that for the first two times an incorrect password was entered the error routine did not return to the program statement in which the error was detected (the OPEN command), but rather to the password entry routine.

Upon entry to the error recovery routine, the number of the program statement being executed at the time of error can be determined as shown in the examples. The error routine has the options of returning to this program statement number (GOTO PSN in our example) to re-try the operation or of proceeding elsewhere (to the password entry routine as in the example). Suppose in our example (Figure 2) the disk drive was not ready when the OPEN was issued. The error routine would inform the operator the drive was not ready then return to the statement in which the OPEN command was issued after the operator acknowledged the drive had been readied.

When designing error recovery routines there are some important items to be kept in mind to avoid unintended or even disasterous results in later processing. Among these are:

1. The error routine cannot return to the command (instruction) executed upon encountering an error. Instead, the return is to a program statement number. All instructions in the program line will be re-executed. The question is, "Do you want to have these instructions again?" For example:

```
100 A = A+B : C = 2*A : INPUT%1, A$
```

Assume the disk drive is not ready. The error routine is invoked when the INPUT%1, A$ command is reached. The operator is notified of the situation and readies the drive. The error routine returns to program line 100. The commands A = A+B and C = 2*A will be re-executed, most likely producing undesired results. The program should have been coded:

```
100 A = A+B : C = 2*A
200 INPUT%1, A$
```

Then, if the not ready condition occurs, the error routine will return to program line 200 and variables A and C will contain the desired values.

2. If an error occurs in a FOR ... NEXT loop and the error routine is not going to return to the loop, it should terminate the loop. For example:

```
100 FOR K = 1 TO 10
...
...
700 NEXT K
```

If an error occurs, the error routine is to proceed to program line 2000 after letting the operator know there was an error:

```
50123 K = 10 : NEXT K : REM terminate the loop

50124 GOTO 2000
```

3. If an error occurs within a subroutine (GOSUB xxx command

issued) and the processing done by the subroutine is not to be continued, the error routine may need to set an indicator to let the routine which called the subroutine know the error occurred:

```
100 E = 0 : REM clear error indicator

110 GOSUB 500 : REM perform computations

120 IF E <> 0 GOTO ... : REM did error occur?
...
...
50123 E = 1 : RETURN : REM set indicator and exit
```

The above is by no means a complete list of considerations for writing error recovery routines. It is intended to be a "starter" list. As you develop error recovery routines, you will add your own items to the list.

Prior to release 1.43, OS-65U informed the operator that the printer was in a not ready state by issuing the message "PRINTER STALLED." This message appeared on the terminal screen at wherever the cursor happened to be positioned and could cause problems if formatted screens were being used. [A modification applicable to releases 1.2 and 1.3 to suppress the message was published in the April 1982 issue of PEEK(65). See the article entitled "Unstall Your Printer" if you are using one of these releases.]

With release 1.43 the message has disappeared. Instead of a message, the terminal alarm is sounded when the system determines that the printer is not ready. The operator then presses return (or enter) to proceed. The printer driver uses a timing loop in identifying the not ready condition. If a PRINT #5 command is issued or the operator has acknowledged a not ready signal and the printer driver has been unsuccessful in sending data to the printer for a period of several seconds, the terminal alarm is sounded to signal that the printer is not ready. (The interval is about 11.5 seconds on the C2-OEM machine. I do not know if it varies based on CPU speed or is fairly constant; i.e., the timing loop being adjusted to match the CPU.)

After the terminal alarm is sounded, the operator has two choices for a response: carriage return (enter) or CONTROL-C. If the entry is a carriage

Figure 2

Operator Supplied Password Required to OPEN a File

```
100 CNT=0 : REM number of times password entered
110 FS="MASTER" : REM file name
120 PWS="" : REM password provided by operator
...
200 CNT=CNT+1 :REM increment password entered counter
210 PRINT "Enter password ===> ";
220 INPUT (4,"A") PWS : REM operator enters password
230 PRINT
...
300 OPEN FS,PWS,1 : REM open master file
...
50000 PSN=PEEK(11774) + 256*PEEK(11775) : REM get error line
50010 EN=PEEK(18176) : REM get error number
50020 IF EN=23 GOTO 50200 : REM if a disk error
.....
50200 EN=PEEK(10226) : REM get disk error number
50210 IF EN=1 GOTO 50300 : REM if drive not ready
50220 IF EN=131 GOTO 50400 : REM if incorrect password on OPEN
.....
50300 Z=PEEK(9832) : IF Z>128 THEN Z=Z-124 : IF Z>63 THEN Z=Z-58
50310 PRINT "Drive "; CHRS(65+Z); " not ready "; CHRS(7)
50320 PRINT "Press <CR> when ready to proceed ";
50330 RS="" : INPUT (0,"A") RS : REM get operator acknowledgement
50340 PRINT
50350 GOTO PSN : REM retry the disk operation
.....
50400 PRINT "Incorrect password entered"; CHRS(7)
50410 IF CNT<3 GOTO 200 : REM has operator had three chances?
50420 PRINT "Processing terminated" : END
```

Figure 3

OS-65U V1.43 Changes for "Printer Not Ready"

The Change	Original Code
POKE 15913,162	POKE 15913,169
POKE 15914,3	POKE 15914,7
POKE 15915,76	POKE 15915,141
POKE 15916,78	POKE 15916,182
POKE 15917,4	POKE 15917,56

Figure 4

Handling Printer Not Ready Condition

```
10 FLAG 25 : REM permit program to handle CONTROL-C
..
100 PRINT #5, ... : REM print a line
110 PRINT CHRS(0); : REM print null to permit CONTROL-C check
120 IF PEEK(15006)>0 GOTO ... : REM if CONTROL-C
...
50000 PSN=PEEK(11774) + 256*PEEK(11775) : REM get error line
50010 EN=PEEK(18176) : REM get error number
50020 IF EN=23 GOTO ... : REM if a disk error
.....
50100 IF EN=4 GOTO 50400 : REM if printer not ready
.....
50400 PRINT CHRS(0); : REM print null to permit CONTROL-C check
50410 IF PEEK(15006)>0 GOTO ... : REM if CONTROL-C
50420 U1=PEEK(8778) : U2=PEEK(8779) : REM save user vector
50430 POKE 8778,135 : POKE 8779,5 : REM single character input
50440 PRINT "Printer not ready "; CHRS(7);
50450 X=USR(X) : REM to single character input routine
50460 X=PEEK(14518)AND127 : REM ASCII code of character entered
50470 IF X<>3 AND X<>13 THEN PRINT CHRS(7); : GOTO 50450
50480 REM (line above) if entry not CONTROL-C or carriage return
50490 PRINT
50500 POKE 8778,U1 : POKE 8779,U2 : REM restore user vector
50510 IF X=13 GOTO PSN : REM if carriage return - retry print
50520 GOTO ... : REM CONTROL-C - to interrupt routine
```

return, the printer driver attempts to send the print line data to the printer. If this is unsuccessful, the timing loop is entered.

If the entry is CONTROL-C, a BASIC error "NR" occurs (error number 4). Interesting. "NR" just might stand for "Not Ready." Since this is a BASIC error, why not use the error recovery routine to communicate printer status to the operator rather than sounding the terminal alarm to signal a not ready printer?

Quite obviously, it is absurd to require the operator to enter CONTROL-C to generate a printer not ready message. It would be preferable to use CONTROL-C to terminate (or interrupt) the printing process and a carriage return to proceed or re-try printing the line. Figure 3 describes a modification to the printer driver which forces a BASIC error number 4, causing entry into the error recovery routine whenever the timing loop expires. If a CONTROL-C is entered before expiration, it can be acknowledged when control is passed to the error routine. Figure 4 is an example of using the error recovery routine with the printer.

In this example CONTROL-C is used to cause entry into an interrupt routine (or possibly the program termination process). [See the article entitled "Printed Report Checkpointing and Restart for OS-65U" in the Nov 1983 issue of PEEK(65) for a discussion concerning the use of an interrupt routine with report printing.] FLAG 25 is set to permit the program to indentify that a CONTROL-C has been entered and determine how it should be handled rather than allowing the system to enter the immediate mode if such an entry is made. Upon expiration of the timing loop, the error recovery routine is given control. The error routine first determines if CONTROL-C was entered during the timing loop, and, if so, it passes control to the interrupt routine. If CONTROL-C was not entered, the "Printer not ready" message is displayed. The operator may then either enter a carriage return to re-try the PRINT operation or a CONTROL-C to proceed to the interrupt routine (the single character input routine is used to obtain the operator's entry).

In the print driver modification, I have not attempted to shorten the amount of time consumed in the timing loop. Reducing the amount of time, for me anyway, would increase the frequency of the not ready messages - annoying when aligning forms or making minor adjustments. Possibly the time interval should be longer when aligning forms, but shorter during report printing. Can someone supply the modification to accomplish this?

When you consider the variety of errors your programs can encounter and the circumstances in which the errors occur, you readily develop an appreciation for the challenge presented in creating error recovery routines. In many instances the logic of the error routines may be more complex than the programs they support. Sometimes, the error routine contains more code than the rest of the program.

Is it worth the time and effort to develop error recovery routines? If you are marketing software, the answer is definitely "yes" because of the extra touch of quality they add - just ask your customers. Even in the programs you write for your own use, they can be lifesavers. If nothing else, think of the challenge they can provide for you.

★

6502 ASSEMBLY LANGUAGE PROGRAMMING CLASS

Part V

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

In the last lesson, we examined how some of the instructions (opcodes) in the 6502 instruction set are able to refer to memory. There are many instructions that do not address memory at all, but instead directly affect either the registers in the 6502 or program branching.

Let's take the latter case first. There are 8 instructions that do conditional branching. These instructions test the condition of particular bits (or flags) in the Status register. We looked at these instructions in lesson 3 and you are referred to that lesson for details. These conditional branch instructions use what is called "relative" branching. The term "relative" is derived from the fact that the destination address of the branch is based upon the memory address where the actual instruction resides, rather than a specific address. The second byte of this two-byte instruction is added to the LSB of the program counter to determine the destination address.

Those instructions that do not refer to memory addresses, but rather, only affect one of the registers in the 6502 use what is called the "implied" addressing mode. We have used some of these instructions in the screen clear program presented in earlier lessons and they are all explained in the August '84 issue. To be specific, the instructions which use implied addressing are:

CLC CLD CLI CLV DEX DEY INX INY NOPPHA PHP PLA PLP RTI RTS SEC SED SEITAX TAY TSX TXA TXS TYA

As you can see if you check the August '84 issue, none of these instructions refer to any memory locations. Their tasks always appear very simple, although this is somewhat misleading. Some of these instructions have effects beyond what their mnemonics imply.

We have seen that the contents of the Status register is important to the relative branching instructions. We will also soon see that the Status flag is vital in performing mathematical calculations. The Status flag is not conditioned only by the instructions which clear and set flags, but is also affected by most of the other instructions as well. The following table demonstrates which flags are affected by instructions in the 6502:

INS:	N	V	*	B	D	I	Z	C	
ADC	N	V	.	.	.	.	Z	C	See Note 1
AND	N	.	.	.	.	.	Z	.	
ASL	N	.	.	.	.	.	Z	C	
BIT	N	V	.	.	.	.	Z	.	See Note 2
BRK	.	.	.	1	.	1	.	.	
CLC	.	.	.	.	.	.	.	0	
CLD	.	.	.	.	0	.	.	.	
CLI	.	.	.	.	.	0	.	.	
CLV	.	0	.	.	.	.	.	.	
CMP	N	.	.	.	.	.	Z	C	
CPX	N	.	.	.	.	.	Z	C	
CPY	N	.	.	.	.	.	Z	C	
DEC	N	.	.	.	.	.	Z	.	
DEX	N	.	.	.	.	.	Z	.	
DEY	N	.	.	.	.	.	Z	.	
EOR	N	.	.	.	.	.	Z	.	
INC	N	.	.	.	.	.	Z	.	
INX	N	.	.	.	.	.	Z	.	
INY	N	.	.	.	.	.	Z	.	

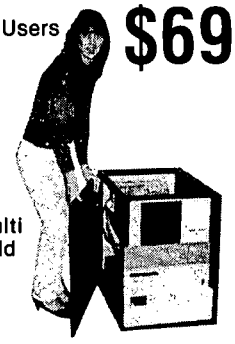
TRY US!

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

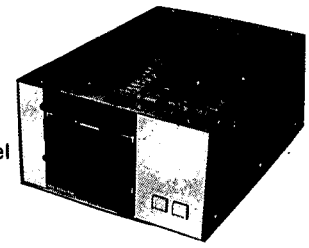
- **Includes Super Utility Package**

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

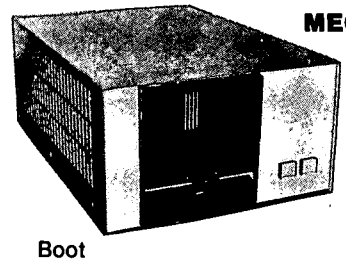
*Includes 8K Exec Memory for CP/M or Multi-User

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

Spare 510 CPU Board with Z80 + 6502 $169.00

MULTI PROCESSOR!

COMPLETE 10 MEGABYTE SYSTEM



$3999.00

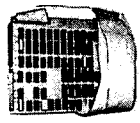
Boot From Hard or Floppy!

- 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

NEW CONTROLLER/INTERFACE!



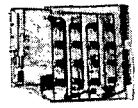
$599.00
Quantity 1

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



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

12 SLOT BUS

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

$79.00

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

```
LDA N . . . . . Z .
LDX N . . . . . Z .
LDY N . . . . . Z .
LSR Ø . . . . . Z C
---------------------
ORA N . . . . . Z .
---------------------
PLA N . . . . . Z .
PLP N V . B D I Z C
---------------------
ROL N . . . . . Z C
ROR N . . . . . Z C
RTI N V . B D I Z C
---------------------

SBC N V . . . . Z C See Note 3
SEC . . . . . 1
SED . . . . 1 . . .
SEI . . . . . 1 . .
---------------------
TAX N . . . . . Z .
TAY N . . . . . Z .
TSX N . . . . . Z .
TXA N . . . . . Z .
TYA N . . . . . Z .
---------------------
```

Note 1:
If in DECIMAL mode,
Z flag is invalid.

Note 2:
N = Data bit 7, V = Data bit 6, Z = AND result.

Note 3:
C = borrow.

★

OSI ROM ROUTINES

(Parts 6 & 7)

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

Before OSI provided their scanned keyboard, they supported a standard ASCII encoded keyboard. Such a keyboard is interfaced through a single 8-bit port (located at $DFØ1) just like the scanned keyboard. Bit 7, however, is used as the keydown flag rather than for data. When a key is depressed, bit 7 goes low. While bit 7 remains low, bits Ø-6 contain the ASCII value of the depressed key. Depending on the keyboard hardware, bit 7 may remain low until the key is released (a simple key-down detector), it may remain low for a specified length of time (a "strobe"), or it may even generate an auto-repeat (such as a 1 millisecond strobe, a one-half second delay and then more 1 ms strobes every 33 ms while the key is depressed). Such things as key debouncing and multiple key conflicts are assumed to be handled by the hardware.

Such an intelligent keyboard

LISTING 1

```
194                ;
195                ; *** ASCII Keyboard Input ***
196                ;
197 FEED AD01DF  HFEED  LDA   KEYBRD  ; Test keyboard port
198 FEF0 30FB           BMI   HFEED   ; No key is down
199 FEF2 48             PHA           ; Key is down,save it
200 FEF3 AD01DF  HFEF3  LDA   KEYBRD  ; Loop until flag is
201 FEF6 10FB           BPL   HFEF3   ;  clear
202 FEF8 68             PLA           ; Then regain char
203 FEF9 60             RTS           ;  and go home
```

LISTING 2

```
 1                 ;*************************************
 2                 ;***                               ***
 3                 ;***       C4P BOOT ROM PAGE 1     ***
 4                 ;***                               ***
 5                 ;*** ROM BASIC Support for 540 Video ***
 6                 ;***        and ASCII Keyboard     ***
 7                 ;***                               ***
 8                 ;***   Comments by  Leroy Erickson ***
 9                 ;***           August 1982         ***
10                 ;***                               ***
11                 ;*************************************
12                 ;
13 0000=           H0000  = $0000 ;BASIC Warm Start
14 0200=           CRSPOS = $0200 ;Video Cursor Pos
15 0201=           H0201  = $0201 ;???
16 0203=           LOADFG = $0203 ;LOAD Flag
17                                ;Set ==> Ser In
18 0205=           SAVEFG = $0205 ;SAVE Flag
19                                ;Set ==> Ser Out
20 0206=           TDELAY = $0206 ;Time Delay
21 020F=           H020F  = $020F ;???
22 0212=           CTLCFG = $0212 ;CTRL/C Flag
23                                ;Non-zero==>Off
24                 ;
25                 ; *** BASIC-IN-ROM Routines ***
26                 ;
27 A628=           HA628 = $A628 ;RTS
28 A633=           HA633 = $A633 ;CTRL/C Test
29 BD11=           HBD11 = $BD11 ;Cold start entry
30 BF15=           HBF15 = $BF15 ;Serial output
31 BF22=           HBF22 = $BF22 ;Init serial port
32 BF2D=           HBF2D = $BF2D ;Video Driver
33                 ;
34 D000=           SCREEN = $D000 ;Addr of video mem
35 DF01=           KEYBRD = $DF01 ;Addr of keybrd
36 FC00=           SERPRT = $FC00 ;Addr of ser port
37 FE00=           HFE00  = $FE00 ;ROM Monitor Entry
38                 ;
39 FF00                   * =  $FF00
40                 ;
41 FF00 D8        HFF00  CLD            ;Clear decimal mode
42 FF01 A228             LDX   #$28     ;Set stack pointer
43 FF03 9A               TXS            ;
44 FF04 2022BF           JSR   HBF22    ;Init Serial Port
45 FF07 A000             LDY   #$00     ;Initialize Flags
46 FF09 8C1202           STY   CTLCFG   ;- CTRL/C Flag
47 FF0C 8C0302           STY   LOADFG   ;- LOAD Flag
48 FF0F 8C0502           STY   SAVEFG   ;- SAVE Flag
49 FF12 8C0602           STY   TDELAY   ;- Time Delay
50 FF15 ADE0FF           LDA   HFFE0    ;Initialize Cursor
51 FF18 8D0002           STA   CRSPOS   ; Position
52                 ;
53                 ; *** Clear the Screen ***
54                 ;
55 FF1B A920             LDA   #$20     ;Get a blank
56 FF1D 8D0102           STA   H0201    ;Clear 2 Indicators
57 FF20 8D0F02           STA   H020F
58 FF23 9900D7  HFF23    STA   SCREEN+$700,Y ;Clear last 8th
59 FF26 9900D6           STA   SCREEN+$600,Y ;Clear next 8th
60 FF29 9900D5           STA   SCREEN+$500,Y ;ditto
61 FF2C 9900D4           STA   SCREEN+$400,Y ;ditto
62 FF2F 9900D3           STA   SCREEN+$300,Y ;ditto
63 FF32 9900D2           STA   SCREEN+$200,Y ;ditto
64 FF35 9900D1           STA   SCREEN+$100,Y ;ditto
65 FF38 9900D0           STA   SCREEN,Y ;Clear top 8th
66 FF3B C8               INY            ;Increment index
67 FF3C D0E5             BNE   HFF23    ;Loop for whole page
68                 ;
69                 ; *** Display Boot Msg ***
70                 ;
71 FF3E B965FF  HFF3E   LDA   BOOTMS,Y ;Get a char
72 FF41 F006             BEQ   HFF49    ;Exit if 0
73 FF43 202DBF           JSR   HBF2D    ;Else, display it
74 FF46 C8               INY            ;Increment index
75 FF47 D0F5             BNE   HFF3E    ;Loop until all done
76                 ;
77                 ; *** Get & Test Response ***
78                 ;
79 FF49 20ABFF  HFF49   JSR   HFFAB     ;Get an input char
80 FF4C C94D             CMP   #'M      ;M ?
81 FF4E D003             BNE   HFF53    ;No, skip
82 FF50 4C00FE           JMP   HFE00    ;Go to ROM Monitr
83                                      ;
84 FF53 C957     HFF53   CMP   #'W      ;W ?
85 FF55 D003             BNE   HFF5A    ;No, skip
86 FF57 4C0000           JMP   H0000    ;Go to Warm Start
87                                      ;
88 FF5A C943     HFF5A   CMP   #'C      ;C ?
```

Continued

would obviously be helpful to reduce programming overhead, but, unfortunately, it costs a little bit more than a simple scanned keyboard so OSI chose to go with the latter. Also, some "game features" would be lost with an ASCII keyboard (you could only receive one key value at a time so you could turn your tank left or right but couldn't move ahead and 'TIGER' would getcha!).

The SYNMON ROM pages 0 & 1 are equivalent to pages 3 & 4, but support the ASCII keyboard rather than the scanned one. In fact, page 0 is identical to page 3 except for lines 194 to 203 (locations $FEED to FEF9). (Note the page 3 listing in the August '84 issue of PEEK(65) and Listing 1). Page 3 contains a jump to $FD00, to get a char from the scanned keyboard routine and go home, and an all rows keyboard scan at these locations. Page 0 contains an ASCII keyboard "get char" routine. In this routine, a first loop is entered which waits until bit 7 goes low. At that point the key value is saved and a second loop is entered to wait until bit 7 goes high again. Then the key value is reloaded and control is returned to the calling routine.

SYNMON page 1 is very similar to page 4 but not nearly as identical as that above. Compare the page 1 listing (Listing 2) to that of page 4 (Sept. '84 PEEK(65)). There are 4 major differences:

1) Locations $201 & $20F are initialized to spaces in page 1, but are not referenced in page 4.

```
 89 FF5C D0A2          BNE   HFF00     ;No, start over
 90 FF5E A900          LDA   #$00      ;Yes, clear A,X & Y
 91 FF60 AA            TAX             ;
 92 FF61 A8            TAY             ;
 93 FF62 4C11BD        JMP   HBD11     ;Go to Cold start
 94                                    ;
 95 FF65 43       BOOTMS .BYTE 'C/W/M?',0 ; * Boot Message *
 95 FF66 2F
 95 FF67 57
 95 FF68 2F
 95 FF69 4D
 95 FF6A 3F
 95 FF6B 00
 96               ;
 97               ; *** Display a char ***
 98               ;
 99 FF6C 202DBF  HFF6C  JSR   HBF2D     ;Display to screen
100 FF6F 48             PHA             ;Save the char
101 FF70 AD0502         LDA   SAVEFG    ;Test SAVE Flag
102 FF73 F024           BEQ   HFF99     ;Skip is clear
103 FF75 68             PLA             ;Else regain char
104 FF76 2015BF         JSR   HBF15     ;Write to Ser Port
105 FF79 C90D           CMP   #$0D      ;Is it CR ?
106 FF7B D01D           BNE   HFF9A     ;No, Go Home
107 FF7D 48             PHA             ;Yes, save it
108 FF7E 8A             TXA             ;  and X
109 FF7F 48             PHA             ;
110 FF80 A20A           LDX   #$0A      ;Get a 10
111 FF82 A900           LDA   #$00      ;  and a Null
112 FF84 2015BF  HFF84  JSR   HBF15     ;Write to ser port
113 FF87 CA             DEX             ;Do 10 Nulls
114 FF88 D0FA           BNE   HFF84     ;
115 FF8A 68             PLA             ;Then regain X & A
116 FF8B AA             TAX             ;
117 FF8C 68             PLA             ;
118 FF8D 60             RTS             ;And Go Home
119               ;
120               ; *** Handle LOAD Command ***
121               ;
122 FF8E 48      LOADCM PHA             ;Save A
123 FF8F A901           LDA   #$01      ;Set LOAD Flag
124 FF91 8D0302         STA   LOADFG    ;
125 FF94 A900           LDA   #$00      ;Clear SAVE Flag
126 FF96 8D0502  HFF96  STA   SAVEFG    ;
127 FF99 68      HFF99  PLA             ;Regain A
128 FF9A 60      HFF9A  RTS             ;And Go Home
129               ;
130               ; *** Handle SAVE Command ***
131               ;
132 FF9B 48      SAVECM PHA             ;Save A
133 FF9C A901           LDA   #$01      ;Get a 1
134 FF9E D0F6           BNE   HFF96     ;Go share code
135               ;
136               ; *** CTRL/C Test ***
137               ;
138 FFA0 AD1202  CTCTST LDA   CTLCFG    ;CTRL/C Enabled ?
139 FFA3 D003           BNE   HFFA8     ;No, Go Home
140 FFA5 4CAEFF         JMP   HFFAE
141               ;
142 FFA8 4C28A6  HFFA8  JMP   HA628     ;Jump to RTS
143               ;
144 FFAB 4CC0FF  HFFAB  JMP   HFFC0     ;Go to CHARIN code
145               ;
146 FFAE AD01DF  HFFAE  LDA   KEYBRD    ;Test the keyboard
147 FFB1 30F5           BMI   HFFA8     ;Nothing, go home
148 FFB3 4C33A6         JMP   HA633     ;Go test CTRL/C
149               ;
150 FFB6 AD01FC  HFFB6  LDA   SERPRT+1  ;Get ser input
151 FFB9 297F           AND   #$7F
152 FFBB 60             RTS
```

```
153                     ;
154 FFBC 68             PLA             ;* Unreachable *
155 FFBD A8             TAY
156 FFBE 68             PLA
157 FFBF AA             TAX
158                     ;
159                     ; *** Get CHAR Routine ***
160                     ;
161 FFC0 AD01DF HFFC0   LDA   KEYBRD    ;Test keyboard
162 FFC3 300D           BMI   HFFD2     ;Nothing, skip
163 FFC5 48             PHA             ;Else,save char
164 FFC6 A900           LDA   #$00      ;Clear LOAD flag
165 FFC8 8D0302         STA   LOADFG    ;
166 FFCB AD01DF HFFCB   LDA   KEYBRD    ;Wait until strobe
167 FFCE 10FB           BPL   HFFCB     ; is done
168 FFD0 68             PLA             ;Then regain char
169 FFD1 60             RTS             ; and go home
170                     ;
171 FFD2 AD0302 HFFD2   LDA   LOADFG    ;Test LOAD flag
172 FFD5 F0D4           BEQ   HFFAB     ;Clear,get keyin
173 FFD7 AD00FC         LDA   SERPRT    ;Else,test serial
174 FFDA 4A             LSR   A         ; input flag
175 FFDB 90CE           BCC   HFFAB     ;Nothing,loop
176 FFDD 4CB6FF         JMP   HFFB6     ;Else,get the char
177                                     ;
178 FFE0 40       HFFE0 .BYTE $40       ;Init cursor pos
179                                     ;
180 FFE1 3F             .BYTE $3F,$01,$00,$03 ; * JUNK (?) *
180 FFE2 01
180 FFE3 00
180 FFE4 03
181 FFE5 FF             .BYTE $FF,$3F,$00,$03
181 FFE6 3F
181 FFE7 00
181 FFE8 03
182 FFE9 FF             .BYTE $FF,$3F
182 FFEA 3F
183                                     ;
184 FFEB 4CABFF HFFEB   JMP   HFFAB     ;Char In Routine
185 FFEE 4C6CFF HFFEE   JMP   HFF6C     ;Char Out Routine
186 FFF1 4CA0FF HFFF1   JMP   CTCTST    ;CTRL/C Test
187 FFF4 4C8EFF HFFF4   JMP   LOADCM    ;LOAD Cmd Handler
188 FFF7 4C9BFF HFFF7   JMP   SAVECM    ;SAVE Cmd Handler
189                                     ;
190 FFFA 3001    NMIVCT .WORD $0130     ;NMI VECTOR
191 FFFC 00FF    RESVCT .WORD $FF00     ;RESET VECTOR
192 FFFE C001    IRQVCT .WORD $01C0     ;IRQ & BRK VECTOR
193                                     ;
194                     .END
```

2) The entry to the CTRL/C test is at a JSR to a get char routine in page 1 but is just after that when called from page 4.

3) The boot message has a space deleted in page 1.

4) The LOAD flag is 'set' to 1 rather than $FF (BEWARE)! Thus any compares are done as zero/non-zero rather than as plus/minus.

Other than the above differences, the instructions end up moved around a little, but the sequence and logic are still the same.

★

BEGINNER'S CORNER

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

LOAD and SAVE

When a desperate user types 'HELP', a good program should respond helpfully! Unfortunately, extensive help would require an overlong program, and is therefore impractical, but some minimum help should be provided. It may return the user to the previous program step, or go right back to the Main Menu. This is the approach taken in the Otaio Mailing List (OML - see June '84 issue). The idea is exemplified in lines 340 and 410 (see Listing 1). Typing 'HELP', or tapping the <RETURN> key, forces a return to Main Menu.

★

In the SAVE block, in line 410, the user is asked for the file name. This line can be easily changed to always save to a specific file name. For example:

```
410 Y$="filename"
```

The message in line 420 is important. There is nothing more unnerving than being confronted by a computer which appears to have suddenly developed a will of its own, seems to be merrily working on something, and is definitely not telling YOU, the user, what's going on!

Or, worse still, the computer has frozen up and the user is presented with the dilemma, 'Should I use the <BREAK> key or should I wait?' The way to avoid frazzled nerves is to use helpful messages!

The OML uses sequential files. What are they? Imagine a long piece of string with knots tied every so often. To find knot 45 it is necessary to count from knot 1, until knot 45 is reached. To add one more knot to the string the last knot must be found towards the end of the string. Using sequential files is rather like coping with the knotted string. To access a string of data from the file it is necessary to read the file, beginning with the first character. Adding data to the file requires the whole file to be read into BASIC. The new data is appended to the end of the file, followed by the carriage return character (knot!). The file can now be written back to disk. Nulls are written to disk as carriage returns.

This method of creating and using files is simple and easy to understand. Call each string of data a field, and let a certain number of fields constitute one record. Each field (string of data) on the disk is separated from the next field by a carriage return. So, fields and records are not distinguished on disk. The number of fields required to make up one record is determined by program logic. The program logic of OML sets P=5 fields to a record, as counted, for example, in the FOR...NEXT loop in line 440.

A sequential file can be viewed directly. To do this, type 'EXIT' to enter the DOS kernel. Now type 'EXAM D000=TR,1', where TR is the number of the track on which the sequential file is stored. The carriage return character will be visible on the screen, separating the strings of data (fields). Any track on a disk can be viewed in this way. Prove these facts to yourself by creating a sequential file and using the command EXAM, and the SEQLST utility. (Don't worry about DOS error message 'Error #D', which occurs when using SEQLST).

Problems arise when a sequential file becomes so large that it cannot all fit into computer memory. A different type of file is then required, the only choice usually being the Random File. When using a Random File it is possible to access any record (off the disk) without having to read any other record. The disadvantage here is that all records have to be a fixed size, though field length

Continued on page 19.

SOFTWARE FOR OSI

EXPLANATION OF LISTING CODES

BASIC Version No./
Minimum computer/
1=SB,SBII,C1P,C2/4P 4=C4P
8=C8P
O=C2/3OEM
D=C2/3-D
2=C200,C3A/B
3=C300

Minimum Storage required/
C=Cassette
5=5 1/4" MF
8=8"FD
7=CD-7
2=CD-23/28/36/74
digit following indicates number of devices required

Systems Supported/
S=Single User
M=Multi-User
H=Hard Disk
R=Record Locking
record lock assumes multi-user. Two may be specified.

Software Support by/
D=Dealer
P=Phone
M=Modem
N=None
O=Other

Sold by/
A=Author
D=Dealer
M=Mail order
O=Other

Copies in Circulation/
No. divided by 10, ie.
1=Less than 11
11=100-110

Price/
Dollars only, no cents, tax, shipping, etc.

<> <> <> <> <> <> <> <> <> <>
OS65-D*UTILITY*SERIAL & VIDEO
<> <> <> <> <> <> <> <> <> <>

ASM-PLUS
3.3/1/51/S/P/A/?/ $50
Author:
RICK TRETHEWEY
8 DURAN COURT
PACIFICA, CA 94044
Seller:
RAINBOW ELECTRONIC REVIEWS
8 DURAN COURT
PACIFICA, CA 94044

FAST DISK-BASED ASSEMBLER FOR D 3.3 ON ALL SYS EX C1P-MF. ALLOWS LINKING MULTIPLE SOURCE FILES (REMOVES MEM CONSTRAINT) 8 TIMES FASTER THAN OSI. FILE EDITOR - FULL LINE & GLOBAL SEARCH. FULL OSI COMPAT + MORE COMMANDS. MENTION PEEK(65) AND GET "REBUG" TOO (A SYMBOLIC DISASM FOR ASM-PLUS FROM MACH)

EAP
ALL/O/81/S/P/D/1/ $125
Author:
JOAN TIRINO
14 MAPLE AVE.
W. NYACK, NY 10994
Seller:
NORTHEAST FINANCIAL SYSTEMS
14 MAPLE AVE.
W. NYACK, NY 10994

TRANSIENT UTILITY REPLACES NULL COMMAND - PROVIDES 13 DIGIT PRECISION WITH ROUNDING FOR 65U & 65D BASIC. ADDITION, MULTIP, DIVISION & SUBTRACTION IN USE 3 1/2 YRS, REQUIRES 2K OF UPPER RAM AREA.

EDIT-PLUS
3.3/1/51/S/P/A/?/ $40
Author:
RICK TRETHEWEY
8 DURAN COURT
PACIFICA, CA 94044
Seller:
RAINBOW ELECTRONIC REVIEWS
8 DURAN COURT
PACIFICA, CA 94044

SIMILAR TO WP-3 UNDER D 3.3 ON ALL SYS EX C1P-MF. FIXES WP-3 FOIBLES LIKE "A" CMD, RESETS LINE COUNTER BEFORE EA OUTPUT. DOES RIGHT JUST (NOT PROPORTIONAL). EDIT-PLUS DOES GLOBAL EDITS, BLOCK MOVES & COPIES. FREE ENTRY OF TEXT WITHOUT LINE # OR <CR>. SIGNIFICANTLY FASTER THAN WP-3.2

TERM-32
3.2/1/51/S/P/A/?/ $25
Author:
RICK TRETHEWEY
8 DURAN COURT
PACIFICA, CA 94044
Seller:
RAINBOW ELECTRONIC REVIEWS
8 DURAN COURT
PACIFICA, CA 94044

SAME AS TERM-PLUS, BUT RUNS UNDER OS-65D V3.2. ALLOWS VIDEO SYSTEMS TO COMMUNICATE AT 1200 BAUD.

TERM-PLUS
3.3/1/51/S/P/A/?/ $25
Author:
RICK TRETHEWEY
8 DURAN COURT
PACIFICA, CA 94044
Seller:
RAINBOW ELECTRONIC REVIEWS
8 DURAN COURT
PACIFICA, CA 94044

SMART TERM. PROG. UNDER D 3.3 ON ALL SYS EX C1P-MF. WORKS ON ANY SER PORT. COMPUSERVE VIDTEXT(TM) COMPAT. FULL ACCESS TO DISK FILES. XMITS BASIC & ASM SOURCE FILES AS ASCII & WP-2/3 FILES FORMATTED.10 PROGRAMABLE FUNCTIONS. 10 PROG. KEYS ON VID. SYS. MANY SUPPORT UTILITIES INCLUDED.

<> <> <> <> <> <> <> <> <> <>
OS65-D*OTHER*VIDEO
<> <> <> <> <> <> <> <> <> <>

BRX-10 HOME CONTROL
3.2/4/51/S/P/A/2/ $25
Author:
F. C. SCHWIERSKE
5014 LAKEFIELD RD.
CEDARBURG, WI 53012
Seller:
SAME

HOME CONTROL OPERATING SYSTEM ALLOWING A C4PMF TO TIME UP TO 16 BSRX10 LOADS. SOFTWARE IS COMPLETELY MENU DRIVEN WITH EXCELLENT GRAPHICS. REQUIRES BSRX10 TRANSMITTER WITH SERIAL INPUT AVAILABLE FROM OSI OR USE PLANS INCLUDED TO MODIFY A STANDARD RADIO SHACK UNIT. FOR PLANS ONLY SEND $5.00.

<> <> <> <> <> <> <> <> <> <>
OS65-U*BUSINESS*SERIAL
<> <> <> <> <> <> <> <> <> <>

TIME & TASK PLANNER
1.3/O/82/MH/P/D/2/ $300
Author:
JOHN HUNTLEY
106 E. STATE ST.
HASTINGS, MI 49058
Seller:
GANDER SOFTWARE, LTD.
3223 BROSS ROAD
HASTINGS, MI 49058

A PERSONAL SUCCESS TOOL! USER DEFINED DAILY SCHEDULER. + TO DO LIST, FUTURE PLANNING LIST. WORK SHEETS AND PRINTED CALENDARS FOR ANY MOS OR YEAR. SEPARATE FILES FOR 5 USERS (5400 APPTS!). VERY EASY TO USE, BUT FLEXIBLE; FULL SUPPORT; 30 DAY RETURN PRIV. YOU'LL NEVER USE DESK-TOP CALENDARS AGAIN.

CAPITAL NEEDS ANALYSIS
/8/81/S/D/A/1/ $49
Author:
WAYNE R. COLE, CLU
805 CHUMLEIGH RD.
BALTIMORE, MD 21212
Seller:
SAME

ILLUSTRATE THE NEED FOR NEW LIFE INSURANCE BY ANALYZING THE INCOME POTENTIAL FROM THE CLIENTS OTHER ASSETS. IT SHOWS THE NEEDS UP AND THROUGH THE YOUNGEST CHILD'S COLLEGE AND FOR THE SURVIVING SPOUSES LIFE THEREAFTER.

DMS-KP FIELD CHANGE
1.42/O/81/S/P/A/1/ $15
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

ALLOWS YOU TO CHANGE CONTENTS OF ANY DMS FIELD IN A SPECIFIED REC RANGE. CAN CHANGE DEPENDING ON THAT FLDS CONTENTS (IF CITY FLD=W THEN MAKE IT=WASHINGTON) OR IT CAN BE CHANGED UNCONDITIONALLY. (SET ALL STATE FLDS=TO VA). REQ'S

EXT INPUT. SEND $1.00 & SASE FOR HYPOTHET USES.

DMS-KP HORIZTL MASTR FILE DUMP
1.42/O/81/S/P/A/1/ $30
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

PROVIDES A HORIZONTAL FORMATTED MASTER FILE DUMP. NO LIMIT ON PRINTER WIDTH OR NUMBER OF FLDS! WILL DO AS MANY LINES AS IT TAKES TO DISPLAY ALL FIELDS. MASTER OR KEY ACCESS, REC RNGE AND CNTRL-BRK PROCESSING AVAIL REQ'S EXT INPUT. SEND $1.00 & SASE FOR SAMPLES. IS A REAL PAPER SAVER!!

BUSINESS VALUATION
/8/81/S/D/A/1/ $49
Author:
WAYNE R. COLE, CLU
805 CHUMLEIGH RD.
BALTIMORE, MD 21212
Seller:
SAME

ILLUSTRATE THE VALUE OF A BUSINESS TO ESTABLISH THE NEEDS IF ANY OF LIFE INSURANCE TO BE USED IN A BUY AND SELL OR STOCK REDEMPTION ARRANGEMENT.

DMS-KP MASTER FILE SPECIFCTNS
1.42/O/81/S/P/A/1/ $10
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

PROVIDES A FORMATTED LISTING OF ALL INFO CONTAINED IN A CHOSEN DMS MASTER FILE HEADER. SHOWS FIELD NAMES, LENS, AND OFFSETS ALONG W/END OF DATA, BEGIN OF DATA, CURRENT NUMBER OF RECS, ETC. OUTPUT TO TERM OR PRINTER. REQ'S EXT INPUT. SEND $1.00 & SASE FOR SAMPLES.

DMS-KP SIMPLE MERGER
1.42/O/82/S/P/A/1/ $25
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

MIMICS SEQUENTIAL MERGE OF OLD DMS MERGER PRGM. SIMPLE TO USE WORKS FASTER THAN ORIG. ALLOWS YOU TO SPECIFY A RECORD RANGE, IF DESIRED. <RETURN> DEFAULTS TO MAX ALLOWABLE RECS YOU CAN TRANSFER. IF TO FILE LENS SHRT WILL AUTO TRUNCATE. REQ'S EXT INPUT.

DMS-KP TRACK MERGER
1.42/O/82/S/P/A/1/ $25
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

CAN DO ALL THAT MY SIMPLE MERGE DOES, BUT W/ A NEAT TWIST! IT WILL INPUT AS MANY RECS AS CAN FIT IN DISK BUFFER BEFORE WRITING TO NEW FILE. IS AT LEAST 40% FASTER THAN ORIG OSI PRGM. MUST BE SEEN TO BE FULLY APPRECIATED. REQ'S EXT INPUT.

DMS-KP ZERO-OUT A DMS FILE
1.42/O/81/S/P/A/1/ $15
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

WILL QUICKLY ZERO-OUT A RANGE OF UNWANTED DMS MASTER FILE RECS. YOU SPECIFY START & END RECS. EXISTS AS AN ALT. TO OLD DMS DELETE & REPACK; WORKS FASTER! REQ'S EXT. INPUT. SEND $1.00 & SASE FOR HYPOTHET USES/EXAMPLES.

PIGGY BACK OR PREM LEVERAGING
/8/81/S/D/A/1/ $150
Author:
WAYNE R. COLE, CLU
805 CHUMLEIGH RD.
BALTIMORE, MD 21212
Seller:
SAME

USE UP TO SEVEN OLD POLICIES TO ILLUSTRATE THE PURCHASE OF A NEW ONE. CHOOSE THE "TOTAL INSURANCE YEARLY OUTLAY" THAT IS COMFORTABLE WITH THE CLIENT. THIS OUTLAY CAN BE LESS THAN HE NOW PAYS, EQUAL TO WHAT HE NOW PAYS, OR MORE THAN WHAT HE NOW PAYS.

ZENEREX CLIENT MANAGER
/O/82/S/D/D/33/ $1495
Author:
ZENEREX CORP.
1301 E. 79TH ST.
MINNEAPOLIS, MN 55420
Seller:
COMPUTER APPLICATIONS CO.,INC.
3004 CENTER RD.
POLAND, OH 44514

CLIENT ACCOUNTING PACKAGE FOR PUBLIC ACCOUNTANTS. PREPARES AND MAINTAINS JOURNALS, G/L, FIN. STMT. INCLUDES COMPARATIVES, BUDGET, HIST. PAYROLL, STMT. OF CHANGES & MORE.

ZENEREX TIME MANAGER
/O/82/S/D/D// $1195
Author:
ZENEREX CORP.
1301 E. 79TH ST.
MINNEAPOLIS, MN 55420
Seller:
COMPUTER APPLICATIONS CO., INC
3004 CENTER RD.
POLAND, OH 44514

TIME & BILLING PACKAGE. MAINTAINS W1P AND A/R RECORDS FOR 999 CLIENTS. 99 DIFFERENT WORK AND BILLING CODES. 3 DIFFERENT INVOICES AND FORMATS. VERY COMPLETE SYSTEM.

<> <> <> <> <> <> <> <> <> <>
OS65-U*UTILITY*SERIAL & VIDEO
<> <> <> <> <> <> <> <> <> <>

FILE FIND UTILITY
1.42/8/81/S/P/A/1/ $10
Author:
KENNETH PORTER
P. O. BOX 1803
SPRINGFIELD, VA 22151
Seller:
SAME

YOU ENTER FILE NAME (BASIC OR DATA) & PASS, PRGM CHECKS DIR FOR A MATCH. IF FOUND, RETURNS ITS LEN & REL DISK ADDR. PRGM EXISTS AS A SUBSTITUTE TO CHECKING A LONG DIR. INVALUABLE IN A HARD DISK ENVIRONMENT. NO SPECIAL REQUIREMENTS NEEDED.

<> <> <> <> <> <> <> <> <> <>
OS65-U*UTILITY*SERIAL
<> <> <> <> <> <> <> <> <> <>

BASIC CROSS REFERENCE (BASXR)
1.2/8/81/SH/D/A/3/ $29
Author:
ELECTRONIC INFO. SYSTEMS
P. O. BOX 5893
ATHENS, GA 30604
Seller:
SAME

BASXR IS A PROGRAMMING TOOL WHICH FACILITATES MODIFICATION AND DEBUGGING. A MENU ALLOWS LISTING OF (1) ALL VARIABLES AND THEIR LINE NUMBER IN ORDER OF OCCURRENCE (2) 12 DISK RELATED OPERATIONS WITH LINE NUMBERS FOR EACH APPEARANCE, AND (3) ANY BASIC COMMAND WITH COMPLETE LINE PRINTOUTS.

DISASSEMBLER
1.44/O/81/H/D/D/00/ $90
Author:
KPS BUSINESS SYSTEMS
P. O. BOX 719
PARKERSBURG, WV 26101
Seller:
SAME

DUMP MEMORY. HEXADECIMAL TO HEX. HEX TO HEXADECIMAL. OSI FLAG TABLE.

EMULATOR - TRACER
1.2+/O/81/SH/O/A/1/ $75
Author:
CARL EIDBO
1509 12 ST, NO.
FARGO, ND 58102
Seller:
EFFECTIVE PROCESSING
1509 12 ST. NO.
FARGO, ND 58102

ALLOWS TRACING OF 6502 MACHINE CODE. DISPLAYS CONTENTS OF ALL REGISTERS AFTER EACH INSTRUCTION. SUPPORTS NAMED

SUBROUTINES AND NAMED MEMORY LOCATIONS. SIMILAR IN CONCEPT TO FLAG 7 IN OSU.

HARD DISK BACK-UP (HFCOPY)
1.2/8/81/SH/P/A/3/ $75
Author:
ELECTRONIC INFO. SYSTEMS
P. O. BOX 5893
ATHENS, GA 30604
Seller:
SAME

A SERIES OF PROGRAMS WHICH ALLOW A HARD DISK TO BE BACKED UP TO FLOPPIES BY TWO METHODS. ONE ALLOWS MULTIPLE FLOPPIES TO BE USED FOR LARGE FILES. THE SECOND ALLOWS MANY SMALL FILES TO BE BACKED UP ON ONE FLOPPY. ALSO, PERMITS RESTORING FROM FLOPPY(IES) TO HARD DISK.

TERM-65U
1.2+/8/81/S/P/A/?/ $50
Author:
RICK TRETHEWEY
8 DURAN COURT
PACIFICA, CA 94044
Seller:
RAINBOW ELECTRONIC REVIEWS
8 DURAN COURT
PACIFICA, CA 94044

SMART TERMINAL PROG FOR ANY VERS OF 65U. SIMILAR TO TERM-PLUS BUT WILL DO SELECTIVE TRANSMISSIONS OF OS-DMS MASTER FILE RECORDS & XMITS WP-3.3 FILES AS FORMATTED TEXT. MANY SUPPORT UTILITIES INCLUDED. COMPUSERVE VIDTEXT (TM) COMPATIBLE.

TRANSFER
1.44/2/71/SH/D/D/0/ $90
Author:
KPS BUSINESS SYSTEMS
P. O. BOX 719
PARKERSBURG, WV 26101
Seller:
SAME

AUTOMATIC BACKUP BETWEEN DISK. TRANSFER FILES BETWEEN COMPUTERS. SPEEDS 300 BAUD TO 500 KBPS. AUTO FILE CREATION. USES STANDARD OSI NETWORK HARDWARE. WILL RUN WITH DBI. READY OCT. 84.

★

WAZZAT CORNER!

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

From time to time it is necessary to convert a machine code program from hex numbers to DATA statements. For example, a machine code utility with a BASIC driver program that is regularly copied to new disks. The program in Listing 1 does such a conversion, but with a difference.

LISTING 1

```
1 REM Listing 1.
2 :
10 PRINT !(28): PRINT TAB( 15)"HEX TO DATA STATEMENTS - LZJ"
20 Z=15: W=13: D$="DATA ": S$=CHR$(34): PRINT : PRINT
30 INPUT "Begin Line Number ";L: input "Line Increment     ";I
40 INPUT "Begin HEX ";N$: GOSUB 180: B=M: M=0
50 INPUT "End   HEX ";N$: GOSUB 180: E=M: M=0 PRINT : PRINT
60 :
70 GOSUB 200: DISK !"MEM 8000,8000": DISK !"IO ,12": PRINT "NEW"
80 :
90 PRINT L"FORX="B"TO"E":READN:POKEX,N:NEXT"
100 FOR X=B TO E STEP Z: L=L+I: PRINT L;D$;
110 FOR Y=X TO X+W: IF Y=E THEN X=E: GOTO 130
120 N$=STR$(PEEK(Y)): PRINT RIGHT$(N$,LEN(N$)-1)",";: NEXT Y
130 PRINT PEEK(Y): NEXT X: PRINT L+I"NEW": PRINT
140 :
150 PRINT "DISK!"+S$+"PUT "+F$+S$+":DISK!"+S$+"LOA "+P$+S$;
160 PRINT ":DISK!"+S$+"IO 02,02"+S$CHR$(13): DISK !"IO 10,02"
170 END
180 N=ASC(N$)-48:N=N+7*(N>9):M=M*16+N:N$=MID$(N$,2):IF N$<>""THEN180
190 RETURN
200 INPUT "Save DATA to File named ";P$: PRINT
210 INPUT "Then LOAD File named    ";P$: PRINT : RETURN
```

First of all, the program is FAST. So fast, in fact, that for short programs, don't look away from the screen - or you will miss the action!

Disk users, create a file on disk to hold the new program of DATA statements. Tape users add these two lines to the program

65 SAVE: INPUT"TAPE ON "; A$: GOTO 90

135 POKE 517,0: END :REM turn SAVE off.

Now, how does it all work? In line 20, the value in variable 'Z' equals 1+13+1, i.e. the first value 'X', as counted by 'Y' in line 110, + 'W', + the value of 'PEEK(Y)' in line 130. This gives 15 data items per DATA line. Variable 'S$' is the speech-mark, '"'. Lines 40 and 50 input the first and last RAM addresses of the program to be converted. The subroutine in line 180 converts hex numbers to decimal numbers - see Vol. 2/6 pg. 7.

The program uses Device 5 to store commands and the new DATA program. Device 5 is just a file in computer memory.

The action begins in line 70. First, the command 'MEM 8000,8000' prepares device 5 to: a) input to $8000, and b) output from $8000. Second, as soon as the command 'DISK!"IO ,12"' is executed, data can be 'printed' to the file at $8000, and also to the screen. The first command 'printed' to Device 5 is 'NEW', and this will be the first command to be read back and executed. Next, lines 90 to 130 'print' to the file the new BASIC program of DATA statements. Lines 150 and 160 'print' a Command File. The final command in line 160 switches input to BASIC from the keyboard to the file at $8000. The '10' in 'DISK!"IO 10,02"' is hex for decimal 16. (The '02' is just 2 - output to screen). The '10' sets bit 5 in the Input/Output distributor flag byte. (Two to the power of 5 is 16!)

BASIC now reads the first command from Device 5, and it is 'NEW' and the old program is gone! Gadzooks, what's this! Here comes the BASIC program, just as if it were being typed in at the keyboard! Next comes the Command File, and it is accepted by BASIC as if from Immediate Mode. The commands are interpreted and executed as follows: 'PUT' the new program to disk, 'LOAD' another program, reset the Input/Output distributor to its normal state, (i.e. accept Input from the keyboard, send Output to screen).

Phew, it's done!

★

READER SURVEY

A number of subscribers have referred to PEEK as "their/our" magazine - meaning the OSI community! Whilst we do receive many compliments, we do, however, ponder if PEEK is reaching out to the OSI community as a whole! Maybe PEEK's direction is getting too technical for some! Not technical enough for others! Too much emphasis on hardware/software! Your comments and suggestions are most welcome. If they are important to you, they are important to us. PEEK(65)'s objective is to serve you, so please take a few moments and give these areas some thought and jot your comments down using as much paper as necessary. Constructive criticism is always welcome. **But please send it in promptly.**

Listed are some specific questions. Please answer those that apply to you. **Your input is important** - after all PEEK(65) is **your** magazine.

1. Is PEEK too technical? ☐ Not tech. enough? ☐ Just right? ☑

2. More reviews? /Y/ (/N/) What kind?_____

3. Do you enjoy non technical articles about OSI applications? (/Y/) /N/

4. Have you written an article for PEEK? (/Y/) /N/

5. Would you write an article for PEEK? If so, what subjects? /Y/ /N/

_____

6. Do you know someone else who would write articles for PEEK? /Y/ /N/
   Please give name, address, phone number, and possible subject.

7. Please list subjects on which you would like to see articles written.

   1. _____ 2. _____

   3. _____ 4. _____

8. To date ads have been restricted to OSI type products. Would you like to see ads for other peripherals, accessories, etc.? /Y/ (/N/)

9. Have you purchased from PEEK advertisers? (/Y/) /N/

   Were you satisfied? (/Y/) /N/ If not, please explain.

10. Did you take advantage of last year's free listing of software? (/Y/) /N/

11. Have you sent in a free listing this year? /Y/ (/N/)

12. Would you like to be able to buy software listed in the software issue directly from PEEK? /Y/ ? /N/

13. If PEEK were to provide merchandise for sale via mail order, what would you like PEEK to carry? Please list.

_____

14. Who is your nearest dealer? Please give name, address, and phone number.

Tech-1 Comp sup 749 River ave
Eugene, OR 97404 (503) 688 7072

15. Where do you turn for repairs?

   Name _____

   Address _____

16. What other computer magazines do you read?

_____

17. What kind of computer do you own/use?

_____

18. How would you rank (1-10) the desirability of having OS-65U available in a 5.25" version? ☐

19. Your name, address, and phone number:

_____

_____

THE DATA SYSTEM

- 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.

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.

TIME & TASK PLANNER

30 DAY FREE TRIAL — IF NOT SATISFIED, FULL REFUND UPON RETURN

- "Daily Appointment Schedule"
- "Future Planning List" - sorted
- "To Do List" - by rank or date
- Work Sheets for all Aspects
- Year & Month Printed Calendar
- Transfers to Daily Schedule

A SIMPLE BUT POWERFUL TOOL FOR SUCCESS

HARDWARE: 48K OSI, 8" floppy or hard disk, serial terminal system, OS-65U v. 1.3 or later.

PRICE: $300.00 (User Manual, $25.00, credited toward TTP purchase). Michigan residents add 4% sales tax.

FINANCIAL PLANNER

- Loan/Annuity Analysis
- Annuity 'Due' Analysis
- Present/Future Value Analysis
- Sinking Fund Analysis
- Amortization Schedules
- Interest Conversions

HARDWARE REQUIREMENTS: 48K OSI, 8" floppy or hard disk, serial terminal system, OS-65U v. 1.2 or later.

PRICE: $300.00 (User Manual, $25.00, credited toward Planner purchase). Michigan residents add 4% sales tax.

DEALERS: Your Inquiries Most Welcome

GANDER SOFTWARE, Ltd.

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

"It Flies"

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

BECOME A SUCCESSFUL

OHIO SCIENTIFIC^ VAR

WITH

POINT-OF-SALE SOFTWARE

COMPLETELY INTEGRATED SOFTWARE INCLUDES:

- POS INTERACTIVE WITH INVENTORY, A/R & GENERAL LEDGER
- CASH TICKETS OR COMPLETE INVOICES
- HANDLES CASH, CHARGES, LAY AWAYS, SPECIAL ORDERS & TRADE-INS
- PRICE LOOKUP, SALES ANALYSIS, PASSWORD PROTECTED
- COMPATIBLE WITH BAR CODE WAND

Now! You Can . . .

✓ **SELL TO THE NO. 1 VERTICAL MARKET - RETAIL BUSINESS**
Literally thousands of retail stores need inventory control with complete integrated accounting. Checkpoint-Of-Sale allows you to sell to the following retail stores: **HARDWARE, SPORTING GOODS, AUTO PARTS, SHOE, HOBBY AND TOY, FURNITURE, CLOTHING** and **GIFT SHOPS.** All these applications from the same software package.

✓ **UNDERSELL COMPETITION BY THOUSANDS OF DOLLARS**
Sell complete turnkey multi-terminal systems including hardware and software for under $15,000 and make over 50% profit. (Try that on an **IBM** [or compatible], **DEC, TRIAD** or **DATA GENERAL.**)

✓ **DELIVER PROVEN, TIME-TESTED SOFTWARE - FULLY SUPPORTED**
Your customer will be buying software that has been installed and running for 4 years, not just promises.

✓ **PROFIT BY SELLING THE SAME SYSTEM OVER AND OVER**
CHECKPOINT-OF-SALE is not customized, it's standard. However, if you choose to customize, the code's open.

✓ **PURCHASE UNLIMITED LICENSE FOR $6995**
License the software to as many end-users as you have time to sell.

MEET WITH US TO DISCUSS THE TREMENDOUS MARKETING POTENTIAL OF RETAIL SOFTWARE IN ISOTRON BOOTH AT COMDEX - LAS VEGAS NOVEMBER 14-18

FOR INFORMATION ON DISTRIBUTORSHIPS AND TERRITORIAL EXCLUSIVES, WRITE OR CALL

CHECKPOINT OF SALE Software by

SILEO, INC.

381 SO. BROADWAY
DENVER, COLORADO 80209 [303] 777-3222

POS "ON LINE" COMPUTING POWER

POS VERTICAL APPLICATION REVIEW

By: J. Dean Ross

Without question, computer systems give the most bang for the buck when they process large quantities of information. In the competitive business world of today, it is too inefficient and costly to allow overhead employees to manually handle menial tasks. In no other business is that fact more evident than in retail.

Retail business provides probably the largest source of business computer end users of any of the applications. Due to the high cost of computerizing prior to microcomputers, this segment of the market place has been reluctant to move toward automating. Now, retailers are seeing the light.

This particular vertical market is, however, no bed of roses. These business people need to be educated about the reasons to computerize. They are continually being bombarded by their trade associations and publications to install computer systems. Many franchisers require their franchisees to be on a computer because their research has dictated that's how to operate successfully.

Due to the large spectrum of operations contained within running a retail business, there are a limited number of software solutions available to them. Integrated systems that provide complete retail business management are few and far between. Checkpoint-of-Sale is a software package designed specifically for this vertical market. The software was written by Sileo, Inc., a Denver based Ohio Scientific dealer since 1978.

The hardware utilized in the system reviewed was an Ohio Scientific 200 series hard disk based computer. It was configured with one CRT and a high speed line printer for the office. Two CRTs and associated ticket printers were located at the sales counter along with two cash drawers. An additional invoice printer was located at the front counter for preparing complete invoices, quotations, lay aways and special orders.

One of Sileo's clients, Fred Jacobsen, granted me an extensive interview and tour through his full-line Sportline sporting goods store in Arvada, Colorado. After talking to the youthful entrepreneur, it was easy to see why he is so successful in retail.

Fred spent nine years cutting his teeth in retailing with Target, Inc., a division of Dayton Hudson Corporation. Then, in 1978 with an extensive background in merchandising, he purchased a bankrupt sporting goods business and reopened its doors.

After spending nearly a year getting his feet on the ground in the sporting goods business, he started the unenviable task of searching for a method to control inventory. Previous experience had made it apparent that his inventory was the largest asset in his business. Its proper management would be first and foremost if he wanted a good bottom line.

The better part of Fred's second year in business was spent talking to numerous computer vendors, software houses, and service bureaus discussing the various methods of retail management. "I must have checked out two dozen different systems" explained Fred. "Probably the most negative aspect of buying your own computer system is sticker shock." Like so many other retailers, with personal computers being so popular, he felt he should be able to solve all his data processing problems for $5000.

"I was looking for a system that would permit us to make an entry once at point-of-sale and not have to touch it again", Fred recalls. Cash registers, as their name implies, handle cash and keep totals on a limited number of departments or classifications. A retail manager needs to know more specifically which items sell fastest, and further, in what sizes, styles, and colors.

For years retailers have struggled with cash registers. That's all that was available to handle a retail sale. Since a large portion of the transactions involved only cash, they were quite adequate. Sportline's business comprises many types of transactions including cash, charges, lay-aways, special orders, quotations, CODs and trade-ins. To handle all these, a point-of-sale terminal is required to be "on line" with a computer.

To do this at point-of-sale, CHECKPOINT utilizes a video terminal (keyboard and screen) to enter sales. The terminal is connected directly to a computer to operate "on line" and interact with inventory and pricing, customer files and general ledger.

Instead of tracking just money and broad classifications of merchandise, the computer is able to track individual items and specific customer accounts and maintain the information by classification, vendor and other criteria.

While employed by Target, Jacobsen received considerable training and exposure in the point-of-sale area. Therefore, the system he would buy for his own store had to be sophisticated enough to look up prices and perform on-line credit and lay away checks. Because he realized that most cash register systems require the salesperson to enter department and classification in addition to the item number and price, he wanted to be able to enter only the item number into the point-of-sale terminal. With price look-up on board, he knew that it would be able to support discounts. Special sales prices for certain categories of merchandise could be controlled enabling him to tell the computer what items to put on sale and for how long.

In June, 1980, Jacobsen purchased a hard disk based Ohio Scientific microcomputer and CHECKPOINT sporting goods software from Sileo. His criteria for the selection of the computer was keeping cost down (his investment was less than $20,000) while being able to operate more than one terminal simultaneously. He wanted the software to not only handle point-of-sale transactions, but to combine this capability with purchasing, inventory and accounting functions.

With CHECKPOINT software the salesperson enters item number and quantity into the video terminal. The computer does the rest, totals the sale, figures tax, calculates change, and prints a cash ticket or complete invoice. The next time you see that sale, it will be in one of the sales analysis reports or the general ledger.

Jacobsen's year of research paid off; he got what he wanted. However, along with the new purchase, there were a few elements he had not antici-

pated. He realized that before you could have an inventory system, you had to input each and every inventory item into the computer, which means looking through vendors' catalogs and price sheets, and assigning numbers to each item. Once items were stored in the computer, everything was great, but it took him a month just to input all eight thousand numbers.

"I don't believe I'd recommend anyone buying his own computer system unless he is truly dedicated", relates Jacobsen. "It's a tremendous amount of work." However, it was during this month-long task that he discovered considerable information about his own inventory. It certainly forces you to become a better businessman because you must think about your costs, selling prices and gross margins on each product. Further, he found it hard to believe some of the items the store was carrying as inventory. The first item on the agenda, after the computer contained the entire inventory, was to have a huge sale to dump merchandise that had been around for years.

The computer has had two noticeable effects on Sportline's relationship with vendors and manufacturers. First, Jacobsen claims that it helps him work closer with each vendor in determining which items and quantities he should be stocking. Prior to implementing this system, everything was a guess. For example, a warm-up suit vendor might recommend a size stocking ratio of 2,4,4,2 (small, medium, large, extra-large). But, due to an individual dealer's demographics, he might really be selling a ratio more like 1,3,5,1. With active wear warm-ups, four suits could represent $20,000 over a year. Secondly, the item and vendor ranking reports, which list dollar sales volume by item or vendor for a range of weeks or months, are useful in planning what to advertise.

Customer reaction to the computer is both positive and negative. A much more professional attitude is presented when a customer recieves a computerized sales ticket or invoice. They like seeing the entire sale unfold on the screen as the salesperson enters each item and are confident pricing information is correct. On the other hand, the customer sometimes may not understand if the computer is down for a few minutes, but those occurrences seem to have been minimal.

One frustration to the novice computer owner is that the computer won't do everything itself. Someone must enter the information and if entered incorrectly, the result is garbage in - garbage out. Because full descriptions are printed out on all sales tickets, invoices, purchase orders and price tags, spelling and pricing must be correct and color and size information valid. The store's incidence of tag switching is minimal now because of the descriptive tags so it appears the effort might have been a small price to pay.

There is another aspect of owning your own computer that is difficult to accept. The computer isn't going to change for you, but rather, you will need to become somewhat flexible in conforming to the system as it's designed. With computerization it becomes quite uneconomical to have the system conform to each businessman's desires.

After owning the system four years, Jacobsen believes this software allows ample latitude. He points out that information is there for his use now. He has control. It allows him daily views of his business so that should a problem arise, it may be corrected right away.

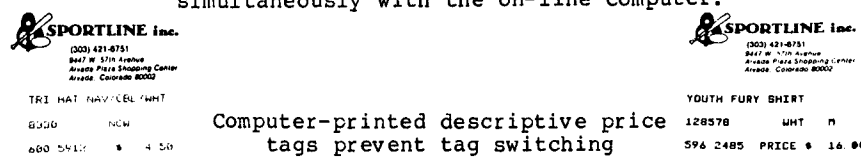
According to this retailer, the point-of-sale computer is a complete management tool, not just a method to handle cash. With today's low cost micro-computers, more retail sporting goods dealers are able to afford owning their own system. Jacobsen's opinion is that successful retail managers in the 80's will automate.

During conversation with officials at Sileo, I learned that they have broadened their package within the last four years to extend to other types of retailers. Their intention is to deliver one standard software package that is broad enough so it may be installed in a variety of different markets. In addition to that they have incorporated the philosophy that a retailer may utilize many types of different numbering systems, all within one store and even may elect to use bar code reader wands for faster checkout capability.

After seeing the same software package installed in three other completely different types of retail stores, I am confident they are achieving their many goals.

The author is a free-lance writer and consultant for retail business.

Multiple point-of-sale terminals can operate simultaneously with the on-line computer.



Computer-printed descriptive price tags prevent tag switching

Continued from Page 10.

within the record may vary. Record size is set by DOS at 128 characters. Accessing records usually involves constant use of the disk-drive; each record has to be read from the disk. Another way of storing data on disk is to use Relative Files, as in 65U DOS. These files have a pointer (INDEX) which can find any byte in the file. Set the INDEX to equal 2001, and byte 2001 can be recovered from the file. Size of fields is determined entirely by program logic. Any field can be accessed at random by calculating its INDEX value. The idea is as simple as can be, but as powerful as one could wish.

Next, in line 430, the Disk Operating System (DOS) opens 'device number 6', for the file whose name is in Y$. 'Device number 6' is the name for the first of two file buffers. The other file buffer is called 'Device number 7'. A buffer is just a place in computer memory where sections of a file are stored, 3072 bytes at a time. (Either buffer could be located elsewhere, in any part of free RAM). BASIC can write to device #6 with the command, 'PRINT#6,'. Clever eh?!

The first piece of information written to the buffer is the number of records, stored in variable 'Z'. Next, in line 440, all the records are written to the buffer, one after the other. Variable C$ equals CHR$(13), and is used to separate fields and records. The semicolon stops an extra carriage return from being written to disk. (The 'C$;' can be omitted from line 440. BASIC will then write the carriage return). DOS throws linefeeds away, so CHR$(10) cannot be used! If it is required that linefeeds be written to disk, do: POKE 9139,255 and POKE 9220,255, for devices 6 and 7, respectively. (The number 255 replaces 10).

Getting back to line 440, variable 'Q' is counting the records, and variable 'C' is counting the fields within each record. When 'P' fields have been written to the buffer then one record has been written, and it is time to write the next record. When the buffer is full, the DOS takes over from BASIC. DOS writes the contents of the buffer to disk, and then control passes back to BASIC. BASIC continues writing records to the buffer, and the process continues until 'Z' records have been written. The 'CLOSE' command in 450 forces the final write to disk, whether the buffer is full or not.

When a sequential file is loaded from disk the process happens in reverse. One track of the required file is loaded into the buffer. BASIC reads the data out of the buffer and into the appropriate arrays. When the end of the buffer is reached, DOS takes over, and if there is more data, the next track is read into the buffer. This continues until the whole file has been loaded.

In line 350, variable 'Z' stores the number of records that have been created and stored in RAM (computer memory). The next free record must be number 'Z+1', and this value is stored in 'Y', ready for use in line 370. In line 370, each record is loaded, one field at a time, into array D$(Q,C). It is instructive to observe the contents of array D$ from Immediate Mode. Run the OML program and load a file, enter Immediate Mode and enter these commands: PRINT D$(1,1), PRINT D$(2,1), and so on.

Because of the way 'Z' is used and recorded on disk, it is possible to load a file and then load another file to sequentially follow the first one. The program would now hold, in array D$(Q,C), two files combined as one.

Tape users see Listing 2 for tape load and save. If Listing 2 is unsatisfactory, have a look at the method used by Charles Stewart - Vol. 4 #7, page 8.

★

```
320 REM LISTING 1.
325 :
330 REM LOAD A FILE
340 INPUT "* Sequential File Name ";Y$: IF Y$=H$ OR Y$="" THEN 190
350 PRINT : PRINT "* Loading from DISK now *": Y=Z+1
360 DISK OPEN,6,Y$: INPUT #6,X: Z=Z+X: IF Z>N THEN Z=Z-X: GOTO 380
370 FOR Q=Y TO Z: FOR C=1 TO P: INPUT #6,D$(Q,C): NEXT C,Q
380 DISK CLOSE,6: GOTO 190
390 :
400 REM SAVE A FILE
410 INPUT "* File Name ";Y$: IF y$=H$ OR Y$="" THEN 190
420 PRINT : PRINT "* Saving to DISK now *"
430 DISK OPEN,6,Y$: PRINT #6,Z
440 FOR Q=1 TO Z: FOR C=1 TO P: PRINT #6,D$(Q,C)C$;: NEXT C,Q
450 DISK CLOSE,6: GOTO 190
```

```
320 REM LISTING 2 - LOAD & SAVE FOR TAPE
325 :
330 REM LOAD A FILE
340 INPUT "* TAPE ON ? ";Y$: IF Y$=H$ OR Y$=" " THEN 190
350 PRINT : PRINT "* Loading from TAPE now *": Y=Z+1: LOAD
355 INPUT X: IF X=0 THEN 355
360 Z=Z+X: REM IF Z>N THEN Z=Z-X: GOTO 380
370 FOR Q=Y TO Z: FOR C=1 TO P: INPUT D$(Q,C): NEXT C,Q
380 POKE 515,0: GOTO 190
390 :
400 REM SAVE A FILE
410 INPUT "* TAPE ON ? ";Y$: IF Y$=H$ OR Y$=" " THEN 190
420 PRINT ; PRINT "* Saving to TAPE now *"
430 SAVE: FOR Q=1 TO 10: PRINT 0: NEXT: PRINT Z
440 FOR Q=1 TO Z: FOR C=1 TO P: PRINT D$(Q,C): NEXT C,Q
450 POKE 517,0: GOTO 190
```

KPS BUSINESS SYSTEMS PART 2

By: Russell D. Daugherty
P. O. Box 719
Parkersburg, WV 26101

Transaction data is stored on disk in a sequential file. The first entry is the type of transaction (cash, charge, payment etc.) ID, and number of items to follow. Each item sold listing product number, quantity and selling price is recorded and followed by a summary of the total transaction. Every transaction that occurs in a store is recorded in a similar manner. At the end of the day, data is copied to two (2) backup disks, with one being retained on location and the other sent to the main office. Backup and resetting pointers is automatic after menu selection. The program requires a bank deposit entry before the backup will start.

Additional POS functions are refunds, credits, payments, record creation, change records (except account balance), defective stock, trade in's, stock orders, special orders, paid out, deposits and messages.

Data from floppies is copied to a hard disk by a modified COPYFI program to permit auto copy of all disks to be processed. Each disk is identified by location and serial number to make certain the disks do not get out of order or lost. Data is processed sequentially, updating inventory files, C/R Journals, bank balances, account files, transaction files, vendor orders, salesman records (amount and number sales) and audit trail. Reports are generated to restock stores, price errors, negative stock items, special orders, special stock orders, changes in account balances, new and changed records and summary of merchandise sold. Provision has been made for restarts to handle disk errors. However, a power failure will cause severe problems if backup was not done prior to processing.

Vendor orders to restock the warehouse are automatically generated when inventory files are updated. Whenever an item goes below minimum, the order file is checked to see if an order has been started for the vendor. If one exists, the part number is then added. If none exists, one is created. This feature allows instant inspection of all pending orders and their dollar value. Vendor records contain the minimum order value to get prepaid freight. While inspecting pending orders, dollar value is compared to minimum; when equal or greater, order is highlighted with reverse video. Orders can be changed manually at any time prior to ordering. When order is complete, it is printed and posted to master inventory. Posting lets inventory know that stock has been ordered, i.e. don't order again. Automatic generation of orders saves countless hours in record keeping, checking stock and of course more accuracy.

Master inventory contains fields for two vendors (primary and secondary), minimum order quantity, three order quantity levels and prices, broken package price, sales per month for twelve months, sales per year for two additional years and package quantity. This data is used to calculate an economic order quantity which takes into consideration costs to generate and process an order, warehouse, handling, sales and payment cost. Some items are packaged by us, so fields exist to indicate which other products make up a package. Data carried on each inventory item is one of the important sources for management information.

Incoming merchandise is checked off on a receiving list indicating those items, if any, which do not match the order. Only deviations are re-entered. This we call, management by exception. Items not received or shorted, are placed into a back order file until received or cancelled. Additions or substitutions can be made prior to posting received order to master.

Invoices are processed by entering PO number and dollar amount. The computer checks the value of the received order and compares it with the invoice amount. When they do not match, each item is displayed for a visual check against the invoice until error(s) are located and corrected. If cost has changed, new selling prices are calculated, edited, posted to master and stored for price changes of stock. The important point is, only those items that need changing are changed, reducing operator input. Payment date(s) and terms are then entered and stored for accounts payable. Input is limited to the absolute minimum. Time spent checking invoices against packing list has been reduced by about 90%.

When payables are run, the operator enters payable date and all bills due to be paid this date will be paid. The computer calculates payables and displays totals due by 10 day increments for 3 periods and by 30 day increments for the next 5 months. It then calculates projected cash flow for the same periods, warning of any expected shortfall and in which period. The operator has an option of selecting certain accounts to be paid, if necessary, due to cash flow. An additional function is comparison of prompt payment terms with current money market rate. When it's to our advantage to hold money in interest bearing accounts, the payable date is pushed ahead until the absolute due date. The only input required is the last date to pay and current money market rate. When checks are printed, bank balance, C/D Journal and the check register are updated. Human involvement with payables has been reduced by 80%.

Check reconciliation is also management by exception. We enter date of last recorded deposit and indicate outstanding checks.

Numerous other programs were written to provide management information:

1. Aging of receivables: Purchases, year to date and average per month
2. Aging of payables
3. Cost of sales and inventory valuation
4. Dept sales by individuals

5. Components of individual sales by selected dollar amount
6. Monthly, quarterly, semi-annual and annual sales by department
   A. Turn rate
   B. Gross profit
   C. Dept vs total
7. Sales reports, current month vs previous month and previous year.
   A. Average sale
   B. Average number per day
   C. Commercial to total
   D. Charges to total
   E. Collection to charges
   F. Sales year to date, 12 months to date and projected annual sales
8. Zero sales by item, in 3, 6, 9, or 12 months
9. Purchases by vendor
10. Customer profile
11. Status of special orders
12. Defective stock
13. Vendor back order inquiries
14. Stock out analysis: Automatic revision of min/max based on sales and frequency of outage

15. Inventory lists by department and vendor

Preparing monthly statements has been a great time saver. Prior to the computer, we would spend 14-16 hours totaling invoice copies and preparing and mailing statements. Now, thirty minutes after closing we have processed the day's data, updated files, sorted account file and are printing statements. Printing takes 30 to 40 minutes for the main store, while we are having dinner, but stuffing envelopes takes 2-3 hours! We are now looking for a reasonably priced presealed form to whack down the mailing time. Results, saving 10-12 hours per month and far more accuracy. In addition, our receivables were reduced 30% after system went into use.

The next greatest joy is bookkeeping. Manual posting and closing consumed a minimum of 80 hours per month, if posting errors were few. In twelve years they only balanced twice, on trial closing. As a practical matter, we would wait until the 10th (until all bills were paid) to total our payable liabilities. Now, books can be closed immediately after processing the last day's business and statements, which is the first day of following month! Closing now takes 2.5 hours including sales and tax reports. The only manual postings are a few General Journal entries. This is what we call timely management information! Manual effort was reduced by 75 hours.

Our big problem has been, and continues to be, inventory count. We spent 14 years moving inventory around, helter skelter, not knowing what we had or were it was. The computer is imposing a new discipline; accuracy. I would guess that we have consumed over 2000 man hours numbering, counting and loading the master inventory file and it still is not completely accurate. Of course the mass of data we are carrying on each item has contributed to the problem. If all we wanted to know was how many widgets we own, it would have consumed less time, but then why use a computer? A card file would serve the same purpose. We are having difficulty adjusting to this discipline but making progress. We counted all locations twice and one location three times trying to get an accurate count. We are going to do it again and keep counting until we get it right. Since no one enjoys inventory, the re-counts are starting to drive the point home that accuracy is essential. If all goes well, the next general inventory should be the last. We plan a weekly random sample, starting with 100 items, varying the frequency and quantity by statistical analysis of the error rate. We will establish an acceptable error factor where cost and benefit balance each other. We have yet to prove the point, but we expect to increase inventory turn rate by 50%.

The conclusion, next month.

LETTERS

ED:

It seems that most of my correspondence with PEEK(65) concerns Editor/Extended Input and terminal usage. This letter is no exception. It discusses the initialization of variables containing CRT control codes when Extended Input is being used.

When Extended Input is activated, most of the CRT control codes are placed in memory beginning at address 6345. These control codes come from the record for the terminal contained in the "CRT 0" file. The control codes for forward space and backward space cursor are stored in memory beginning at addresses 23734 and 23741 respectively. The system manuals for releases 1.3 and 1.4 of OS65U and the GETCRT program contain model subroutines which can be used to retrieve the CRT control codes from the operating system. The following code can be added to the subroutines to retrieve the control codes from forward space and backward space cursor (line numbers follow the sequence in the reference manual):

```
63940 Z=23734 : FS$="" : REM
      forward space cursor
63941 Z1=PEEK(Z) : Z=Z+1
63942 IF Z1<>0 THEN FS$=FS$+
      CHR$(Z1) : GOTO 63941
63943 Z=23741 : BS$="" : REM
      backward space cursor
63944 Z1=PEEK(Z) : Z=Z+1
63945 IF Z1<>0 THEN BS$=BS$+
      CHR$(Z1) : GOTO 63944
```

I have some systems in which "global" variables are defined in the menu programs. Global variables are common to all (or most) of the programs in the system. These variables are passed from the menus to

the other programs by means of the common variables features. CRT control codes are among the global variables.

To retrieve CRT control codes in the menu programs, I have used a routine similar to that provided by OSI. When the menu programs were run, it seemed to me that a considerable amount of time was being consumed in variable definition/initialization, particularly for the CRT control codes.

When you study the routine for retrieving control codes, you will notice that a lot of processing is taking place due to the generalized nature of the code. Naturally, the routine was written this way in order to accommodate the control code structure of the many types of terminals which can be defined.

During this processing the addressing scheme of the terminal is determined, which takes time. There are a number of loops involved with retrieving control codes as the number of characters comprising each code is unknown. The GOTOs in the loops introduce additional system overhead due to the line number searching which is done. Once these codes have been determined, why not store them some place where they can later be retrieved with less processing (and less system overhead)?

Within the OS65U nucleus there is an area of memory which is used only by the resequencer (RSEQ) program. This area extends from addresses 23552 through 23695 and can be a convenient place to store data which is frequently used by programs. This area can be used for saving CRT control once they have been determined by the generalized routine. If you are not interested in saving all of the codes and/or there are not that many characters in the codes, they can be stored back in the place from whence they came; i.e., beginning at address 6345.

The routines which follow can be used to store what I call the "refined" CRT control codes. It is assumed that the generalized code retrieval routine was added to INP$ (Extended Input activation) or some other program which is run as part of the initialization process for your system:

```
Variable names and definitions (used in OSI sample routines)
 AR = CRT coordinate addressing scheme indicator
 YF = Y-coordinate (row) offset
 XF = X-coordinate (column) offset
AD$ = cursor addressing lead-in sequence
DL$ = cursor addressing delimiter
DE$ = cursor addressing delimiter
CS$ = clear screen
CE$ = clear to end of screen (or page)
CL$ = clear to end of line
FG$ = foreground
BG$ = background
FS$ = forward space cursor
BS$ = backward space cursor

63960 REM SAVE CRT CONTROL CODES.
63961 Z=23552 : REM storage area address (could also be Z=6345)
63962 POKE Z,AR : POKE Z+1,YF : POKE Z+2,XF : Z=Z+3
63963 R$=AD$ : GOSUB 200 : R$=DL$ : GOSUB 200
63964 R$=DE$ : GOSUB 200 : R$=CS$ : GOSUB 200
63965 R$=CE$ : GOSUB 200 : R$=CL$ : GOSUB 200
63966 R$=FG$ : GOSUB 200 : R$=BG$ : GOSUB 200
63967 R$=FS$ : GOSUB 200 : R$=BS$ : GOSUB 200

100 REM SERVICE ROUTINE FOR SAVING CRT CONTROL CODES.
200 Z=LEN(R$) : REM length of control code (number of characters)
210 POKE Z,L : REM store control code length
220 IF L=0 THEN Z=Z+1 : RETURN : REM if no control code
230 FOR K=1 TO L : REM set loop iterations
240 POKE Z+K, ASC(MID$(R$,K,1)) : REM store a character
260 NEXT K : REM loop control
270 Z=Z+K : REM set next storage area address
280 RETURN
```

Now that the "refined" control codes have been stored, they can be retrieved by other programs. The following routines can be used for the retrieval:

```
1000 REM RETRIEVE CRT CONTROL CODES.
1100 Z=23552 : REM storage area address (could also be Z=6345)
1200 AR=PEEK(Z) : YF=PEEK(Z+1) : XF=PEEK(Z+2) : Z=Z+3
1300 GOSUB 200 : AD$=R$ : GOSUB 200 : DL$=R$
1400 GOSUB 200 : DE$=R$ : GOSUB 200 : CS$=R$
1500 GOSUB 200 : CE$=R$ : GOSUB 200 : CL$=R$
1600 GOSUB 200 : FG$=R$ : GOSUB 200 : BG$=R$
1700 GOSUB 200 : FS$=R$ : GOSUB 200 : BS$=R$

100 REM SERVICE ROUTINE FOR RETRIEVING CRT CONTROL CODES.
200 R$="" : REM clear work area
210 L=PEEK(Z) : REM number of characters in control code
220 IF L=0 THEN Z=Z+1 : RETURN : REM if no control code
230 FOR K=1 TO L : REM set loop iterations
240 R$=R$+CHR$(PEEK(Z+K)) : REM retrieve control code character
250 NEXT K : REM loop control
260 Z=Z+K : REM set next storage area address
270 RETURN
```

Using this approach to managing CRT control codes I have reduced variable definition/initialization time by upwards of 50%. Another benefit has been less code required to retrieve the codes; I can always use the extra room in programs. Hopefully, other OSI users will notice a similar performance improvement.

David A. Weigle
Morton, IL 61550

★ ★

ED:

I have recently upgraded my 5 year old OSI C1P to disk, and have come up against a rather perplexing problem: The system will not save to disk.

Preliminary investigation has revealed that upon initialization, the track headers are not being written correctly. The first two bytes are OK, but the track number and the last byte are wrong. Here are the results of an initialization of a brand new blank disk:

```
TK 1:CW(CHR$24)(CHR$96OR32-
     BLANK)
TK 2:CW(CHR$13Ø)(CHR$255)
TK 3:CW(CHR$75)(CHR$254)
TK 4:CW(CHR$13Ø)(CHR$96OR32)
```

Subsequent headers were equally fouled up, and re-initialization changed the last two bytes of the headers, but not to the correct format. Also, the disk was not erased after the "header" was written onto the disk.

For hardware, I currently have a C1P, D&N's Data Separator, D&N's disk controller board, and a Tandon TM 1ØØ-1 disk drive. For software I have OS65DV3 and HEXDOS 4.Ø, HEXASM 1.Ø, and a modified C1S ROM to support HEXDOS.

I feel that the problem is in the controller, but much IC swapping and 3 cans of IC cooler have not isolated the problem. Initial board setup

for the read/write circuit was done with a Tecktronic's 425-(V) dual trace O-Scope, and the pulse widths are adjusted to specs and have not shifted.

Under OS65D, #5 errors are displayed for any disk write, and the format program on HEXDOS hangs up at line 260 during a disk write, obviously because the track header does not match the target track number.

I am currently at my wits end and desperately need any information or advice that you could give me that would help me solve this problem.

C. J. Hipsher ICC
New York, NY 09565

CJ:

While I can't comment directly on your problem, one thing you should be aware of is that according to the factory, OSI's are very very vulnerable to salt in the air near the ocean. At first glance, the track headers you report are so fouled up as to defy reason. The bit patterns of the numbers reported don't fall into a discernable pattern which would lead me to suspect noise as opposed to an out and out hardware failure. We in the software game have a term for this - ".... sucker's broke".

Aloha!

Rick Tretheway

Readers:

If any one else can help, please write to us.

Peek Staff

* * * * *

ED:

In the Sept 1983 issue of PEEK(65), you replied to Steve Rydgig's inquiry about FORTH by saying that FORTH was fast, logically organized, and appealing to hard-core hackers, but about as hard to program in as ASSEMBLY language.

Although I agree with your remarks in general, I would respectfully take issue with your last comment regarding difficulty in programming.

As a long-time FORTH user I can tell you FORTH. once thoroughly learned, is one of the easiest of all "languages" to program in. FORTH at first seems awkward and difficult, but this feeling soon passes.

One of the nicest attributes of FORTH is its ability to grow with the programmer while at the same time revealing more and more of its inner nuances. The longer one programs in FORTH the "easier" the process becomes. A FORTH awareness creeps in, but unlike our experience with languages such as BASIC, FORTRAN, PLI, PASCAL, etc., this increasing awareness seems to grow almost without end or diminution. This may in part explain the enthusiasm of dyed-in-the-wool FORTH addicts.

Actually FORTH isn't a language at all. It is a set of tools out of which languages can be created.

Whereas BASIC, let us say, is about equally efficient when applied to widely differing or related problems. FORTH is very much more efficient for coding related applications. The FORTH one develops tends to become personal (both a strength and a weakness of the language) and specialized towards one's own particular applications.

In summary:

FORTH is adaptable and extensible.

FORTH is logical and understandable.

FORTH can be very high level or very low.

FORTH is memory efficient and fast.

FORTH was developed on small computers for small computers.

FORTH is flexible because, although FORTH has rules, the rules can be easily changed.

FORTH can be both compiled and interpreted.

FORTH refuses to pander to the tyro and rewards the proficient.

FORTH is fun.

FORTH is powerful, and

FORTH is beautiful.

I would advise anyone interested in getting started in FORTH to buy Leo Brodie's book "Starting FORTH" and to join the FORTH Interest Group, San Carlos, CA.

Newton C. Fawcett, Ph.D.
Hattiesburg, MS 39406

* * * * *

AD$

HELP WANTED: Full and Part-time. New York authorized OSI Dealer is looking for programmers and hardware technicians. Attractive salary and good working conditions. Must have strong background in OS65U with hard disk experience. Call 212-926-7634 or send resume to Crescent Computer Systems, Inc., Box 119, New York, NY 10037

* * * * *

FOR SALE: OSI C2-8S, 16K, one disk & Centronics interface. OS-65D V2.0 & V3.3, mailing list, WP-1B, misc. - 33 disks. $320.00. Act IV Micro-Term terminal, working with above $315.00. Both for $600.00. John Childs, 4719 Twin Pine Dr. NE, Cedar Rapids, IA 52402, (319) 363-8987, (319) 395-7557.

* * * * *

FOR SALE: OSI 23mb hard disk, works well. Make offer. Bill Brown (503) 357-7132.

* * * * *

FOR SALE: C3 w/Dual 8" Double-Sided Drives - $600. C8P-MF, 48K, 2Mhz w/RS Line Printer - $500. Software included, call Craig (616) 399-3109.

* * * * *

C2-OEM, $800; Hazeltine 1500 Terminal, $200; Centronics 779 Printer, $100; ALL THREE, $1000. Call (216) 747-1803 or write David Skaggs, P. O. Box 556, Youngstown, OH 44501.

* * * * *

FOR SALE: OSI C3-B Computer (48K RAM, two 8" floppies, 74MB hard disk), all in excellent working condition. OS-65U 1.42, OS-DMS included. Best Offer. Write PICS, Box 3117, Berkeley, CA 94703 or call (415) 654-2259 or leave message at (415) 654-5900.

* * * * *

SB II/Series 1; 8K; $60. SB II/Series 2; 8K; $75. 610 board; 24K; $100. PS-005 5V/3A; $20. Aardvark 8K+PIA; $20. A13 board; data separator; $10. CPU case; $20. C12 cassettes; 12/$10. EPROM programmer; 2516/2716; $50. (Add shipping). J. Hays, 2401 53rd Ave. SW, Seattle, WA 98116 (206) 935-3548.

* * * * *

Send for free catalog, Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005. Phone (312) 259-4071.

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 0S-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