PEEK (65)

MAY 1985
VOL.6,NO.5

The Unofficial OSI Users Journal

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

INSIDE

Column One

It's COMDEX time again! To manufacturers and suppliers, that usually means the culmination of months of preparation to meet the deadline for new product announcements. The OSI/DBI world is no exception. The trouble is that everyone is very secretive about what is to be shown until the last minute. In our case, it means that this issue was all but ready for the printers when we received the announcements. That being the case, there is only room for a thumbnail sketch in this column, with the details to follow next month.

To begin with, in the order that we have been informed, Gander will be holding a seminar on May 21st for people who are purchasing the Program Generator. The Program Generator allows the programmer to pick out the required pieces of The Data System in creating his own program, and that's an over-simplification! It will not be released directly to the end user, only to authorized DBI and ISOTRON dealers and systems houses. Rumor has it that they will also show their new MS-UTIL (multi-system utilities) that allows one copy of the utilities to be used in all hard disk subsystems.

It appears that DBI will be using Gander's gathering to make some announcements of its own. The big brother (16 user) to the DM-1 machine is expected to be there. The user's machine is in one box and all disks and storage in another, including 280 MBs of hard disk, floppy and tape streamer. They may even have the newly revised DB-1 CPU boards in it, using the 8/16 processor and 4K x 1K memory. Lastly, if things work out, they will finally release the 65-E op system, which contains a 17 integer digit precision BASIC with twice as many reserved words and faster than "U".

Isotron has two new hardware efforts to announce at COMDEX. First is the Portland Board (utilizing the 6502C at 4 MHz) for the 200 series computers which we commented on in this column last month. The big news is the new lineup of 700 series machines. Some of you have commented that the OSI hierarchy have been hard to reach these past months. The reason is that they have been slogging it out to make the COMDEX deadline with a totally new machine based on the 68000 family of CPUs. Ultimately, there will be three machines: the 720, in the ad, a 710 and a baby, as yet unnamed. They all will be garden variety UNIX 5 based, with the 720 utilizing co-processors (68000 and 68008) to relieve the main CPU of its time-consuming serial duties. The key points to the user seem to be that: it will run most OS-U material (there's a program to do most of the conversion work) and will compile to either FORTRAN or COBOL (both standard versions) so as to allow unlimited software transfers between machine types. At this time, they are reporting that the 720 will comfortably outrun any other UNIX machine on the market. The 10 MHz clock might have something to do with it, but they have a bunch of other tricks up their sleeves too. Try this: ten users, each doing a KeyFile FIND (UNIX version of FIND) in separate files, each with 200,000 records at an average find time of one second. The software bundle will include a DB, EMail, Screen/Menu Builder, WP, Spreadsheet, Spooler and a super-extension of OSI BASIC that is comfortable with most BASICs, OSI and ISAM files, and still is about 3 times faster than "U".

I am embarrassed at the skimpy overviews above, but how much can one do in this short space? If the above cooperate, you should find the June issue packed with the details. Of course, we are also looking for "spill-overs" that will effect the small users - and there will be some.

Our list of "articles needed" seems to have had some effect, but much more is needed. If you have read about one users efforts to lift the disk head and have another approach, jot it down. Out of such efforts comes the "ultimate" result! The best part is that we are hearing from those of you who have been too bashful to write before. Now that the ice is broken, keep it rolling. If they can do it so can you!

BINARY ENCODING AND BIT DETECTION OF KEYBOARD CHARACTERS

A clever way to scan the keyboard quickly and generate Morse Code.

By: Michael J. Goldstein
Courtesty of TOSIE
Toronto Ohio Scientific Idea Exchange
P. O. Box 29
Streetsville, Ont.
Canada L5M 2B7

Having done the "C1 Tape Control" modification (Put Your Tape Cassette Under Software Control), I now had a software controlled relay available for Morse Code transmission.

The easiest way to have software open and close a relay for Morse code is to use a separate subroutine for each keyboard character, with timing loops for opening and closing the relay for the desired code character. This works, but is very memory hungry, and slow.

Don Moore, VE3EVE, suggested I use encoding and bit detection - and when I looked blank, he borrowed all my OSI manuals; not only did he devise this elegant encoding scheme which I shall shortly reveal, he also discovered how to use the machine code keyboard scan routine at $FD00:

```
110 POKE 11,0: POKE 12,253:
    Z=PEEK(531): REM SETS USR
    POINTERS
120 X = USR(X): PRINTCHR$(Z);
    :REM KEYBOARD SCAN
130 GOTO 120: REM  GO GET THE
    NEXT KEYBOARD PRESS
```

This gives you a fast keyboard detection scheme. If you try to scan the whole keyboard using BASIC, and the method shown in the OSI Graphic Manual, you will have to wait with your finger on the key till the keyboard scan routine gets 'round to scanning that key - very awkward typing!

Once you have the keyboard character, you are ready to convert it to Morse Code.

This is how Don encoded each character, first in Binary, then in Decimal:

First, we will use 0 for a "dit", and 1 for a "dah"...

For the letters A, B, and C ...

	128	64	32	16	8	4	2	1	DECIMAL
A= ·- =	0	1							
B= -··· =	1	0	0	0					
C= -·-· =	1	0	1	0					
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	BINARY

We also need to know how many bits each character has, after it is binary-encoded. Three binary digits allow you to count up to 7 in Decimal. The longest Morse character has 8 bits - the "error" sign is ..... Later we will see that using only 3 binary digits for bit counting allows us to easily divide the decimal equivalent of our binary encoded character by 256. So, we will use extra binary digits to count the number of character bits MINUS ONE.

	4	2	1									
	1024	512	256	128	64	32	16	8	4	2	1	DEC.
A(·-)			1	0	1							
B(-···)		1	1	1	0	0	0					
C(-·-·)		1	1	1	0	1	0					
	2^2	2^1	2^0	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	BIN.
	BIT COUNTING			ENCODING								

If the character has four bits (1000) the bit count shows 3 (011). Our data word now consists of up to 11 binary digits: the first 3 (from the left) signify how many actual bits(-1) our encoded character contains. The remaining bits encode our character.

Well, we can't use binary digits easily in BASIC, so let's now convert our unique binary codes to equivalent decimal numbers.

That's easy - just add up the decimal values of all the binary digits we used!

A = 256 + 64 = 320
B = 512 + 256 + 128 = 896
C = 512 + 256 + 128 + 32 = 928

All of these decimal values would be read into an array, to be used later in sending code. The decimal values are stored in DATA statements.

```
10 DIM C(50)
20 FOR Z =1 TO50: READ C(Z):
   NEXT:RESTORE
30.......
1000DATA......
1010DATA.....
1020DATA...
```

Now, we must detect the Number of bits in each character and send the dit or dah for each bit...

```
199 Z =Z-43: REM CONVERTS Z
    FROM ASCII VALUE TO ARRAY
    COUNTER
200 H=256: N= INT(C(Z)/H):REM
    NUMBER OF BINARY BITS(-1)
    IN CHARACTER
210 I+128:FOR L1=0TON:I1=INT
    (B/I):REM IS FIRST BIT A 1
    OR A 0?
220 B=B-I*I1:REM VALUE OF B
    UPDATES TO CHECK NEXT BIT
230 IF I1 THEN 6000: REM IF
    I1(FIRST BIT) IS A 1, GO
    TO "SEND A DAH" SUB-
    ROUTINE.
240 GOTO 7000: REM FIRST BIT
    AIN'T A 1, SO IT MUST BE
    A 0-SO GO TO THE "SEND A
    DIT" SUBROUTINE
250 I=I/2: NEXTL1: REM YOU'VE
    CHECKED THE FIRST BIT- GO
    CHECK THE NEXT.
260 REM WAIT A SPELL AFTER
    YOU'VE SENT THE LAST DIT
    OR DAH, AND GO BACK TO THE
    KEYBOARD SCAN FOR A NEW
    CHARACTER...USE TIMING
    LOOPS FOR DELAYS
270 GOTO 120
```

Oh Yeah .... TOSIE's beginners articles by Roger say you should initialize all those variables before starting into your loops, to keep things running speedily, and to avoid having to initialize each variable each time you go through the loops.

```
6000 REM DAH SUBROUTINE
6010 CLOSE THE RELAY, WAIT A
     WHILE, OPEN THE RELAY,
     WAIT A SPELL, AND BACK
     TO THE PROGRAM
```


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		$19
Canada & Mexico (1st class)		$26
So. & Cen. America	$38	$30
Europe	$38	$30
Other Foreign	$43	$30

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

```
7000 REM DIT SUBROUTINE
7010 CLOSE THE RELAY, WAIT A
     WHILE, ETC ETC
```

Note that, using this technique, you must send a Morse character before typing the next character - there is no "typing ahead" - can someone devise a method to do this?

To make your array counter agree with the keyboard, press ASCII value. Here is the sequence of characters, for which decimal values are read into your array: X indicates a space.

```
,_.?0123456789:;XXXX/XABCDEFG-
HIJKLMNOPQRSTUVWXYZ
```

Would it humble you if I mentioned that Don had never used BASIC before he devised this algorithm? He speaks 4 other computing languages, however!

```
10 REM MORSE CODE TRANSMIT BY MIKE GOLDSTEIN VE3GFN-298 WARDEN AVENUE
20 REM SCARBOROUGH ONTARIO MIN3A4 CANADA, ALGORITHM-DON MOORE VE3EVE
30 REM PROGRAM FOR OSI C1P WITH AARDVARK C1S MONITOR ROM
40 REM COMPUTER TO BE MODIFIED-CONTROL A RELAY WITH SOFTWARE AT 61440
50 DIMC(50):B=255:E=254:F=1:G=8:K=61440:M=85:Y=0
60 H=256:J=2:POKE11,0:POKE12,253:REM SETS POINTERS TO SCAN ROUTINE
70 PRINT"      CODE MASTER":PRINT:PRINT:PRINT:PRINT
80 CO=53381:FORX=1TO23:POKECO+32+X,43:NEXTX
90 PRINT"To type without sending code, press ESC
100 PRINT:PRINT"To return to code, press ESC again
110 PRINT:PRINT"type '?' for (ERROR)
120 PRINT:PRINT"type '-' for (AA)
130 PRINT:PRINT"type ':' for (AR)
140 PRINT:PRINT"type ';' to change SPEED
150 PRINT:PRINT"hit ESC to continue":WAIT 57088,32,254
160 PRINT:PRINT:PRINT
170 GOSUB680:PRINTCHR$(03)
180 PRINT"  CODE/NO CODE-hit ESC":PRINT
200 POKE548,6:POKE549,26:PRINTCHR$(03):REM SCREEN FORMATTING
210 PRINT"ERROR)*?  (AA)*-  (AR)*:":FORX=1TO23:POKECO+32*5+X,183:NEXT
220 FORX=1TO23:POKECO+32*22+X,183:NEXT
230 POKE548,27:POKE549,30:PRINTCHR$(03):PRINT"  WPM";SP;"  SPEED*';'
235 REM CHANGE SCREEN FORMAT TO PRINT your CALLSIGN HERE...
240 PRINT:PRINT"        VE3GFN":POKE548,10:POKE549,24:PRINTCHR$(03)
250 C$=,-.?0123456789:;    / ABCDEFGHIJKLMNOPQRSTUVWXYZ"
260 FORZ=1TO47;READC(Z):NEXT:RESTORE:L=0:REM LINE CHARACTER POSITION
270 X=USR(X):Z=PEEK(531):REM KEYBOARD SCAN
280 IFZ=27THENPRINT:PRINT:PRINT"Now in receive mode***":GOTO730
300 IFZ=59THENGOSUB680  :GOTO230:REM GET NEW SPEED REQUEST
310 IFZ=32THENL=L+1:PRINTCHR$(Z);:FORT=1TOTI*10:NEXT:GOTO270
320 IFZ<44THEN270
330 IFZ>90THEN270:REM ILLEGAL KEYBOARD PRESSES
340 Z=Z-43:REM CONVERT Z FROM ASCII VALUE TO ARRAY COUNTER
350 PRINTMID$(C$,Z,1);:REM PRINT CHARACTER OF KEYPRESS
360 IFL=70THENL=0:PRINT:REM STARTS NEW LINE WITH <RETURN> AND LINEFEED
370 REM CHARACTER DECODING FROM BINARY TO MORSE
380 N=INT(C)Z)/H):REM # OF VALID BITS IN 8-BIT BINARY CODE
390 B=C(Z)-N*H:REM DECIMAL EQUIV.OF BINARY VALUE OF MORSE CODE
400 I=128:FORL1=0TON:I1=INT(B/I):B=B-I*I1:REM CHECK EACH BIT...
410 IFI1THEN470 :REM GO SEND A DAH IF I1 IS A "1"
420 GOTO530 :REM GO SEND A DIT 'CAUSE I1 IS A "0"
430 I=I/J:NEXTL1:REM YOU'VE CHECKED THE FIRST BIT-GO CHECK THE NEXT!
440 FORT=1TOTI*2,3:NEXT:GOTO270 :REM GO GET NEXT KEYBOARD PRESS
450 REM 'SEND A SPACE'ROUTINE
460 L=L+F:FORT=1TOTI*4:NEXTT:GOTO270
470 REM 'SEND A DAH' ROUTINE
480 POKEK,M:REM CLOSE EXTERNAL RELAY
490 FORT=1TOTI*40:NEXTT:REM DAH LENGTH
500 POKEK,Y:REM OPEN RELAY
510 FORT=1TOTI*2:NEXTT:REM SPACE BEFORE NEXT DIT OR DAH
520 GOTO430
530 REM'SEND A DIT' ROUTINE
540 POKEK,M:FORT=1TOTI*15:NEXTT:REM DIT LENGTH
550 POKEK,Y:FORT=1TOTI*3:NEXTT:REM SPACE BEFORE NEXT DIT OR DAH
560 GOTO430
570 REM ERROR ROUTINE
580 GOTO370
630 REM DECIMAL EQUIVALENTS OF CHARACTER BINARY CODES
640 DATA1484,848,1364,1328,1272,1144,1080,1048,1032,1024,1152,1216
650 DATA1248,1264,1104,-1,-1,-1,-1,1168,-1,320,896,928,640,0,800,704
660 DATA768,256,880,672,832,448,384,736,864,976,576,512,128,544,784
670 DATA608,912,944,960
680 PRINTCHR$(03):INPUT"CODE SPEED (5-50 WPM)";SPEED
690 IFSP<5GOTO680
700 IFSP>50GOTO680:REM ILLEGAL SPEED REQUESTS
710 TIME=45/SP:REM CHANGE THIS VALUE FOR CORRECT CODE SPEED CAL.
720 RETURN
730 X=USR(X):Z=PEEK(531)
740 PRINTCHR$(Z);
745 IFZ=13THENPRINT
750 IFZ=27THENPRINT:PRINT:PRINT"Now-transmit mode***":GOTO270
760 GOTO730
ready
```

K=57089
M=75

★ ★

HOSPITAL INSTALLATION

By: C. Culp, Jr.
Comptrol Systems, Inc.
P. O. Box 1305
Parker, CO 80134

Since PEEK(65) has requested articles about successful installations of DBI/OSI gear, we decided to relate one such installation which has been very "typical" of our experiences here in Denver. The users of this system are far too busy to have the time to write such an article, so we will do it on their behalf.

The installation is at one of the largest hospitals in Denver, - St. Anthony's. St. Anthony's purchased one of the first OSI systems to come into the Denver area, approximately 7 years ago. The system was purchased by the Bio-Medical Department of the hospital. Bio-Med is responsible for the maintenance of ALL hospital equipment, from beds to EEG machines to catscanners. All total, this amounts to over 20,000 pieces of equipment in St. Anthony's. The system was to track repairs made to each piece of equipment, schedule routine preventive maintenance, track all parts/costs associated with a piece of equipment, report breakeven/cost analysis figures, and report man power utilization to management.

Bio-Med had tried to get these programs established on the hospital's multi-million dollar mainframe but was told that the cost would be in the hundreds-of-thousands of dollars and it would take 3 to 5 years to complete. This was not acceptable to Bio-Med and thus they decided to get their own system to do these chores.

For two years, they had an in-house programmer who, using OSI's DMS Nucleus, tried to get the applications on-line. His success was limited and he accepted another position at a different hospital. The Bio-Med Department began searching for someone who could complete the project. They found Comptrol Systems by a referral from the distributor who had sold them the machine.

After analyzing the need, preparing cost/time-frame studies, and presenting a proposal to the Bio-Med Department, we began the project. We estimated 6 - 9 months for total completion. Using our Systems Generator (Data Base Management System), we built all the necessary file structures and

data base maintenance programs within 2 weeks. We did this so that St. Anthony's could begin building the data base as soon as possible, knowing that it was going to take a considerable amount of time to enter the 20,000+ pieces of equipment.

While the data base was being built, we spent approximately 8 weeks putting together the custom programs which would capture date and time of service on an item, service technicians initials, type of hours (Reg, OT, Etc.), labor accumulators, Inventory Control, etc.. The hardest part of the whole application was determining how to deal with 5 - 7 years history on each piece of equipment. This history would grow to 100,000+ records very quickly and had to be able to be searched fast! Our Systems Generator handles the data base quite nicely.

After installing the software, we went through 60 days of debugging and fine-tuning the system. It was working great! The system was running so well, and producing such valable cost management and scheduling reports for Bio-Med that the Engineering Department decided it wanted to go on-line with its 5000+ pieces of equipment.

Since the original system was a 2 terminal, 23MB system, there just wasn't any room for expansion. We needed a lot more hard disk space and faster data access for more terminals. At this point, we upgraded the system to a 250JJ - that's twin 80MB drives. We had plenty of disk space now, but adding 3 more terminals to the system caused too much search speed degradation. We installed 5 - DB-1 DBI Processor Boards and realized well over 200% speed increase in our searches.

The system is VERY EASY TO USE; it HAS to be because "non-computer types," such as Service Technicians, need to be able to walk up to a terminal, punch-in an Equipment ID#, and get a report on a piece of equipment. We do this via fully menu-driven software. The results are that the technician can review the equipment's history with his supervisor to determine if it should continue to be repaired or should be replaced. The cost controls in these reports have saved the hospital thousands, maybe even tens-of-thousands of dollars.

The Inventory Control in the system is very "simple," but it allows for effective tracking of stock items. It also lets Bio-Med "bill" other departments for parts they have used to repair that department's equipment.

Probably the most beneficial part of the system is the automatic PM (Preventive Maintenance) scheduling. Each piece of equipment, even a BED, is assigned a PM Code which indicates if that equipment is to be looked-at daily, weekly, biweekly, etc.. Every night (the system runs 24 hours per day), the system goes through each piece of equipment to see if it is to be PM'ed the next day. A list of scheduled equipment is prepared for management to assign technicians to. The technician inspects the equipment and prepares a Work Order which is then returned to the data entry operator. This Work Order indicates how much time was spent on the PM, any materials used, etc.. The operator enters this into the system and the system, using the PM Code, calculates the next PM date.

A secondary reason for keeping so much history on the system is that in Colorado, hospitals must go through a state certification audit in order to continue to operate. To meet the state's requirements, historical data on all equipment must be retained for the state's auditors to review. (This also comes in handy in court cases involving malfunctioning equipment.) Before acquiring the computer system, all these records were kept by hand (theoretically!). The state's certification inspection required St. Anthony's to have from 5 - 7 people available, just to "track-down" equipment history for the auditors. It is now done by the auditors themselves! They just sit down at the system and look-up what they want to see. Hard copy is also available to them. This has saved St. Anthony's a minimum of $100,000 since installing the system (man hours saved during audits). Believe me, this FAR EXCEEDS the amount they have invested in both hardware and software!

All in all, St. Anthony's is very pleased with their system. Hardware down-time over the past 5 years has been less than 24 hours. Software down-time has been less than 2 hours. The system currently supports 5 terminals via DB-1's, 2 printers via DBI's DP-1 Printer Board, twin OSI 80MB drives, and an Alloy Tape Backup Unit. There is talk now of putting a couple more departments onto the system which will probably require going to DBI's micro to gain access to 300MB plus drive size.

This Work Log System is available for other hospitals through Comptrol Systems (Licensed by St. Anthony's). We also have a "scaled-down" and modified version of the system which will work for ANYONE who needs to keep detailed Repair History on equipment (Schools/ Factories/Car Rental Agencies/ etc.). By the way, the system also records the last time a room was painted, who painted it, what color paint was used, who the vendor of the paint was, etc., etc.. (Hotels?) There are lots more "little" things the system does, but time does not permit me to list them all here.

I KNOW there are lots of other successful DBI/OSI installations "out there." Hopefully, others will relate their system experiences to PEEK(65) readers.

★

TURNING FLOPPY DRIVES OFF

By: Ed Richardson
146 York Street
Nundah, Queensland 4012
Australia

As requested in March Colunn One, here are my thoughts on the head load mod on a C4P-MF.

This would have to be about the simplest possible modification. The three problems it solves are:

1. The head will load and unload with disk access.

2. The MPI drive will now eject disks.

3. The activity light will go on and off with head load.

The big plus, of course, is the reduction of wear on the disk and head. A small minus is the increase in noise due to the head relay. The disk eject and activity light are a bonus. There is no really simple way to electronically control the motor without building up a board. The board would need a retriggerable monostable to start and stop the motor, and some

logic to block off the INDEX pulse until the motor was up to speed - say a 1.2 sec delay. The monostable would be triggered off the $COXX line that feeds the disk controller PIA. It would time out and stop the motor after say 3 seconds. However, a manual motor switch would be easy to implement, either a simple toggle, or a microswitch with an operating arm actuated by the drive door when the release is pushed. The latter idea has the advantage of being automatic, plus instant access to disk files. It's a compromised solution. During program development or game playing, it is always a wise precaution to remove the disk in case of power failure anyway. The 1.2 second delay can be annoying when ASCII files are constantly being accessed. Just my thoughts on the matter.

Now to the mod for head load. 65D has all the software needed to operate the load relay, so no problem there. Unlike the 8" systems the MPI drive does not have a separate pin for head load, so some hacking is needed on the disk controller board (on the drive, that is).

Step 1. Locate the shunt 1G, mounted near the edge connector on the drive board. Open circuit all links on the shunt except the one joining pins 2 and 13. Solder a wire between pins 4 and 8, either under the board, or preferably across the shunt links.

Step 2. Perform the track cut modification as in the diagram. Solder a wire between the pad and pin 4 of the 1G shunt. This can be done under the board, or by a flying lead up to the shunt link. (The latter is preferable.)

Step 3. Locate the black wire on J4 pin 5. J4 is the large connector next to the edge connector and numbering starts from that end.

If you remove the connector from the plug, and turn it over, you will see a slot. Push the end of a small screwdriver against the pin in the slot while gently pulling on the wire, and the pin will come out. Bend up the locking tab on the pin slightly, before pushing the pin into the connector socket number 19 (four from the other end), until it clicks into place. Refit the connector onto the plug.

The job is done! Because this is a drive mod, it would work just as well on a C1-MF. A more complex change is needed for a DF.

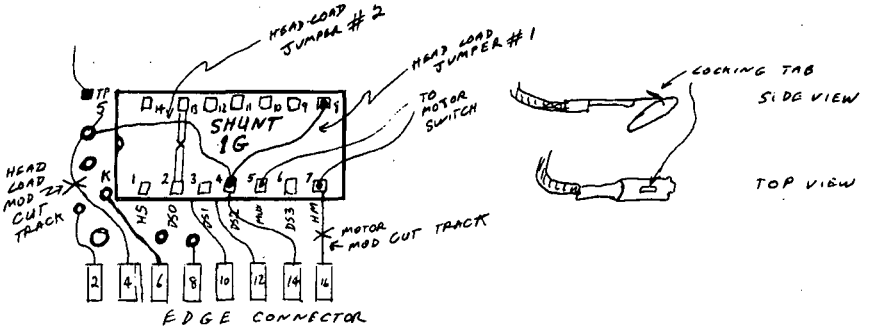
Motor mod for microswitch operation (or toggle switch).

Cut the track from edge connector pin 16 to the 1G shunt as in the diagram below. Solder flying leads to pins 5 and 7 of the 1G shunt. When these leads are connected, the motor will run.

On the OSI A13 board (the one that plugs into J2 on the 505 board) remove the strap (a blue wire on mine) that shorts J2 pin 4 to ground. This is only to remove the short from the outputs of a couple of ICs, a practice I don't like.

Of course, the proper motor control method would be to write software to start/stop the motor. OSI provided the hardware to control the motor, but never implemented the software. It would be a solid job to modify the DOS.

As to the positioning of the motor switch, I will leave it up to individual preference. There are many possibilities. One clever way could use a relay springset mounted above the centre spindle on the door frame arm. This pushes up when the disk is clamped.



A little bit on my background. I am an electronics technician working in the television broadcasting field. (The US calls them engineers.) I have had a C1 since early 1980. In May, 1980, I began a User Group which is still going. The first 25 newsletters were published independently, but since 1982 I have published it in KAOS, a 6502 User Group based in Victoria. Our club has over a hundred members, while KAOS has over three hundred. I now own a C3A, purchased second hand from a business that hardly used it. I have recrunched a BASIC 3 and a BASIC 4 chip. The BASIC 3 mod solves the string bug problem. The BASIC 4 chip offers a fast M/C or BASIC token Save/Load/Verify for cassette with error checking and can auto-run programs from tape. A special Cold/Warm-start function is provided for program hang ups. Standard cassette format is retained. No other functions are lost. Either chip costs A$14.00* including postage world-wide, that's less than US$10. (BASIC 4 is for C1 only, and uses 1200 baud). Full installation and operating details are included, and profits go to the User Group.

* P.S. (Basic 4 is for C1 only, and uses 1200 baud.)

★

BEGINNER'S CORNER

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

STOP THE DWARVE!

Games programs tend to be

lengthy and very difficult to edit if they are written using the traditional method of "One Thousand POKEs." It should be possible to print graphics without having to resort to one POKE statement per graphic character. If the graphics are stored in strings, then several could be printed at once using a BASIC "print at" routine. Under OS65D 3.3 this is particularly easy to do with the "print at" command, PRINT&(X,Y).

A good games program will be short and powerful with plenty of action graphics. It should be quick to write and easy to debug. The best games programs are usually variations on the traditional favorites and so "Stop the Dwarve!" is modeled on "Hangman". Please read on!

The most tedious aspect of many games programs is the enormous number of "POKE" statements that have to be typed in. An alternative method for producing graphics is explained here. Another major problem for games writers is the pseudo-random number generator in BASIC. Ideally, each new game should start with a different random number series and the same random numbers should not repeat. The solutions to both problems are well known, simple, and described later. Another major area that is explored is cursor addressing as supported by DOS 3.3. A good variety of action graphics is implemented.

MAKE A PLAN

The first step to take when writing a program is to devise a plan. For a games program, the first part of that plan is to design the graphics. See Figure 1. Next, design the action.

The graphics action is as follows:

draw the scaffold,

draw figure in horizontal strips,

after 5 guesses the dwarve appears and pushes a step to the scaffold,

the dwarve climbs one step after each wrong guess,

on the top step the dwarve pulls the pin out and throws it away,

the pin is caught by another dwarve at the top of the screen,

the figure's eyes flash,

trap doors open,

figure drops,

mouth turns down,

eyes reverse,

dwarve jumps up and down.

The other graphics are:

messages are printed to the screen at various times,

scores are printed,

correctly guessed letters are printed to make up the word.

THE PROGRAM

The program was written to run under OSI 65D 3.3. The spaces have been inserted for clarity only - do not type them in. All REM and spacing lines can be omitted. The shorter a program is, the faster it will run. Before typing in the program, set one buffer before BASIC's workspace. (Do this using option 7 under DOS 3.3 or run CHANGE under DOS 3.2.)

The first two POKES in line 40 establish null input on <RETURN>. The third POKE can be used to disable CTRL-C - substitute "96" for the "173".

The first block of any program

should declare constants; that is, variables whose values do not change. It is also a good idea to declare, before all others, those variables that are used in time-consuming loops and procedures. These variables are declared in line 50, and the program will run faster as a result.

Long, and/or often used values such as "CHR$(27)+CHR$(20)", should be declared as variables. Doing this speeds up a program in two different ways. First, a statement such as "PRINT C$" means much less work for BASIC; the value for C$ is looked up in a table. This is much faster than having to translate "PRINT CHR$(27)+CHR$(20)". Secondly, six "PRINT C$" obviously take up less space than half-a-dozen "PRINT CHR$(27)+CHR$(20)". The program will be shorter and, therefore, faster.

Location 13026 holds the character for the cursor. The POKE in line 90 sets this character to a less distracting blank! T1 in line 110 is used in the time delay loop in line 1260. T2 is used to speed up the game after 5 wrong guesses have been made. The program chooses its words from array "W$", whose size is fixed by "NW" in line 120. The graphics characters strips are held in array C$(L). In line 110 are the names of three sequential files from which lists of words can read.

RANDOM NUMBERS

Line 130 sets up the array from which numbers will be chosen at random. Each number will be chosen ONCE only. BASIC's random number seed is constantly updated in location 8996 ($2324). This happens when devices #1 and #2 are polled. (The code is at $24F9 and $252E, respectively). PEEKing RAM location 8996 will give a random number (Y) between 0 and 255. This number is then used in RND(-Y) to start a pseudo-random number series different to the previous one. Information on how pseudo-random numbers are produced comes under the heading of "Linear Congruential Generators". It is not easy reading.

CHANGES FOR ClP

For ClP users a "print at" statement can be simulated with,

```
2000 FOR H=1 TO LEN(Z$):POKE
     X+H,ASC(MID$(Z$,H,1)):
     NEXTH: X=X+24:
     RETURN
```

where Z$ is the string to be printed and X+1 is the first screen address. The "24" is screen width. A good example of how the program could be adapted (ClP) to use line 2000 would be as follows. Change lines 150 and 160 to:

```
150 Z*:Y$="----------------":
    X=53670:PRINT C$;:Z$=Y$:
    GOSUB 2000
160 Z$=": STOP the DWARVE! :":
    GOSUB2000:Z$=Y$:GOSUB2000
```

It really works!

The command "Z*" in line 150 may be puzzling. This is an extra command added to "HOOKS". See this month's WAZZAT column for a full explanation and substitute code.

The first FOR...NEXT loop in line 190 reads into arrays X and Y, the X,Y coordinates at which the graphics will be printed. If the BASIC "print at" routine in line 2000 is to be used then only one array (X) would be required. Then in line 190 change "READ X(C), Y(C)", to "READ X(C)". Each pair of addresses X,Y would be substituted for by one 5-figure screen address. For example, in line 1400 the pair "27,0" would be replaced by "53529" (for C4P, 32 by 32 screen mode). The next pair "27,1" would be replaced by "53593" (=53529+64).

THE DATA

Addresses are stored in pairs, ten pairs to a line, from line 1400 to 1440, with four more in line 1450.

All the graphics characters are stored as numbers in DATA statements. The scaffold is in lines 1480-1530. The figure is in line 1550-1610. The final pair of numbers in line 1530 are the noose. The open trap door is the second group of numbers in line 1620. The eyes and mouth are in line 1630.

The total number of graphics strips that are printed is 54. The 54 is stored in "L", in line 50, and used in line 190. Array C$ stores the graphics numbers.

Have a look at line 1480. The first digit is a 5 and counts the number of characters that belong to the first picture strip, i.e., the five characters "ASC(150)" that make up the top bar of the scaffold - see Figure 1.

Loops two and three in line 190 use this graphics data to build up character strips in array C$ - see line 200. The finer details of how this works can be investigated by running the program given in listing 1, and then in Immediate Mode typing "PRINT C$(1)" etc. The scaffold can be printed using:

```
F=1:L=30:FOR Q=F to L:
PRINT&(X(Q),Y(Q)) C$(Q):NEXT
```

ClP users try:

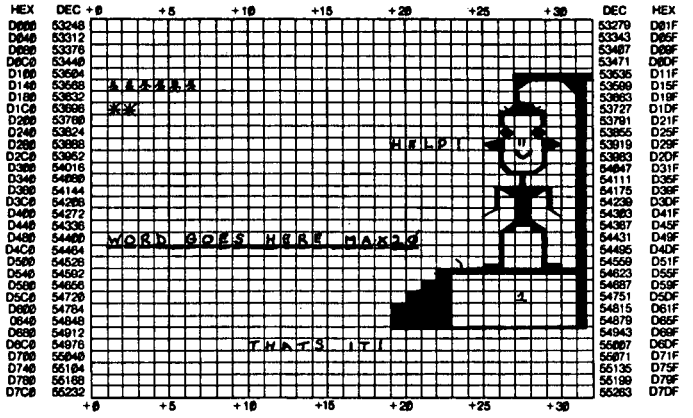
```
F=1:L=30:FOR Q=F TO L:
Z$=C$(Q):X=53400:GOSUB 2000
```

To print the figure let F=31 and L=44. Of course, the scaffold and the figure could both be printed together using F=1 and L=44. This method of printing graphics is very flexible and was particularly useful during program development. A single FOR....NEXT loop does all the work.

Next month, the second part of the program, cursor addressing and how to produce non-repeating random numbers.

Listing on Page 10.

FIGURE 1-2. VIDEO MEMORY MAP - 540 IN 32X32 FORMAT.



1976

THE FIRST TRUE PERSONAL COMPUTER

Ohio Scientific...The Superboard, 400 CPU

1977

THE FIRST MICROCOMPUTER WITH BUILT-IN FLOPPY DRIVES

Ohio Scientific...The Challenger

1978

THE FIRST MULTI-USER MICROCOMPUTER

Ohio Scientific...The 550/CA-10X with 16 port serial I/O Board

1978

THE FIRST MICROCOMPUTER WITH BUILT-IN HARD DISK

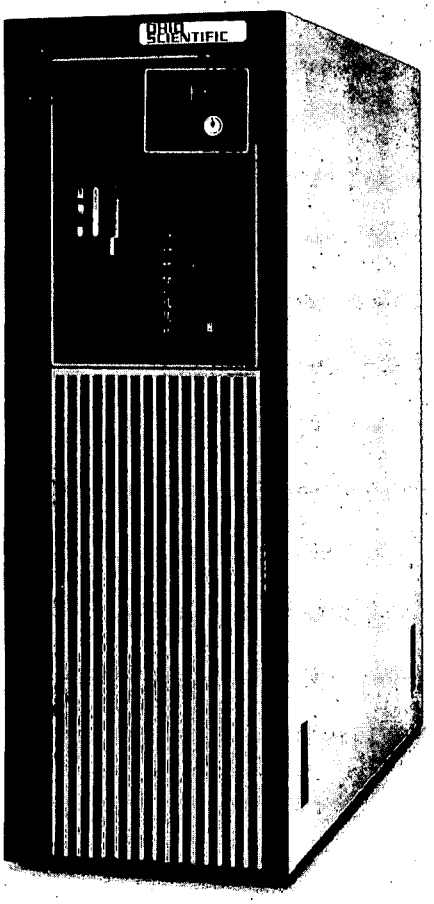
Ohio Scientific...The CD74/C3-B with a 74MB Winchester hard disk drive

1979

THE FIRST MULTI-USER MICROCOMPUTER NETWORK

Ohio Scientific...The CA-10N5/OS-65U with distributed processing hardware and software

1985



THE FIRST REAL-TIME UNIX™ MULTI-USER MICROCOMPUTERS

Ohio Scientific...The new Unix Series. See us at Comdex, Booth 5167, and see what we mean by:

- *Real-time response*
- *Unique communications compatability with IBM 3270, BSC/SNA, IBM 2780/3780, IBM 3770, and others.*
- *Easy program generation with 1 million record capability.*

UNIX is a tradename of Bell Laboratories.

OHIO SCIENTIFIC

THE FIRST NAME IN MULTI-USER MICROCOMPUTERS

Manufactured by ISOTRON, Inc., 140 Sherman Street, Fairfield, Connecticut 06430 (203) 255-7443 Telex 756436

```
10 REM =^=^=^=^=^=^=^=^=^=^) STOP THE DWARVE! (=^=^=^=^=^=^=^=^=^=^=
20 REM Copyright 1985 by L.Z. Jankowski.
30 :
40 POKE 2888,0: POKE 8722,0: POKE 2073,173  : REM Null INPUT, CTRL-C
50 C=0: K=0: L=54: Y=0: P=0: F=0: Q=0: H=0: R$=CHR$(13): B=95: D=127: E=45: G=122
60 R=10: L$=CHR$(8): L1$=L$+L$+L$: H$=CHR$(241): I$=CHR$(161)
70 S$=H$+1$: F$=CHR$(32): G$=CHR$(240): C$=CHR$(27)+CHR$(20)
80 P1$="Good, that's correct.": P2$="Sorry, that's wrong."
90 P3$=" Press a letter key.": DW$=" ": POKE 13026,32
100 P$="You tried '": Q$="' before!": FOR C=1 TO 5: DW$=DW$+G$: NEXT
110 Y$(1)="DWARV1": Y$(2)="DWARV2": Y$(3)="DWARV3": T1=600: T2=T1/2
120 NW=99: DIM R(NW),C$(L),X(L),Y(L),D(26),W$(NW),W(26),P(26)
130 FOR C=0 TO NW: R(C)=C: NEXT : Y=PEEK(8996): Y=RND(-Y): TRAP 1670
140 :
150 Z$: Y$="--------------------": X=L: PRINT C$;&(X,10)Y$
160 PRINT &(X,11)": STOP the DWARVE! :"&(X,12)Y$
170 :
180 REM ---------READ SCR ADDR & PICTURES----------------------
190 FOR C=1 TO L: READ X(C),Y(C): NEXT : FOR C=1 TO L: READ Y: FOR K=1 TO Y
200 READ P: C$(C)=C$(C)+CHR$(P): NEXT : NEXT : PRINT C$;
210 :
1390 REM ----------------SCREEN ADDRESSES----------------------
1400 DATA 27,0,27,1,30,1,31,2,31,3,31,4,31,5,31,6,31,7,31,8
1410 DATA 31,9,31,10,31,11,31,12,31,13,31,14,31,15,31,16,22,17,21,18
1420 DATA 20,19,19,20,19,21,31,17,31,18,31,19,31,20,31,21,25,2,26,3
1430 DATA 25,3,25,4,25,5,25,6,25,7,25,8,25,9,25,10,25,11,25,12
1440 DATA 25,13,25,14,25,15,25,16,22,16,25,17,24,17,30,17,24,18,30,18
1450 DATA 24,19,30,19,26,5,25,16
1460 :
1470 REM ------------------PICTURE DATA-------------------------
1480 DATA 5,150,150,150,150,150, 1,175, 2,177,156, 1,156,1,156
1490 DATA 1,156,1,156,1,156,1,156,1,156,1,156,1,156,1,156,1,156,1,156
1500 DATA 1,156,1,156,1,156, 9,161,151,151,151,151,151,151,151,151
1510 DATA 2,187,187, 3,161,161,161, 4,187,187,187,187
1520 DATA 12,161,161,161,161,128,128,128,128,128,128,128,128
1530 DATA 1,161,1,161,1,161,1,161,1,161, 5,32,32,136,32,32,2,224,225
1540 :
1550 DATA 5,32,179,161,182,32,  5,32,136,32,143,32
1560 DATA 5,32,232,32,232,32,   5,227,136,34,143,228
1570 DATA 5,32,136,191,143,32,  5,32,190,128,189,32
1580 DATA 5,32,32,149,32,32,    5,147,177,161,175,146
1590 DATA 5,147,143,161,136,146,5,195,143,161,136,197
1600 DATA 5,32,176,151,178,32,  5,32,152,32,153,32
1610 DATA 5,32,152,32,153,32,   5,150,152,32,153,150, 2,241,223
1620 DATA 6,32,32,32,32,32,32, 1,218,1,216,1,149,1,149,1,149
1630 DATA 1,149, 5,32,233,32,233,32, 5,32,136,193,143,32
1640 :
1650 PRINT !(12)!(12)!(19)!(26)!(18)
1660 PRINT &(4,0)"$ Disk error. Try Again. $": GOTO 290
1670 PRINT C$;&(4,0)"$ FC Error or SN Error $": GOTO 240
```

★

INTRODUCING THE SYNPHOTHIZER BOARD

By: D. Petersen
3892 Nth 4th St.
Fresno, CA 93726

The Synphothizer Board was designed to bridge the gap in sound capabilities between the OSI and other popular systems. Being based on the advanced SSI-263A speech snythesizer and the Texas Instruments SN-76489A sound generator, the Synphothizer offers capabilities rivaling or exceeding those built into other systems.

Provided on a single 8 by 6.5 inch board are one speech chip and two TI sound chips plus three independent one-watt amplifiers (with volume controls) providing stereo sound and speech effects.

★

Each of the TI sound chips provides three independent tone generators capable of covering a five octave range. One generator is provided for white and periodically varying noise, and may be controlled by the output of tone generator 3. Each of the chip's sound generators also has its own attenuator which may be programmed to cover a 28 dB range from maximum amplitude to off in sixteen -2 dB steps. This most useful feature of the SN76489A makes the synthesis of many interesting sound effects possible. Programming tones may be accomplished by simply POKEing two bytes of data to the chip, and only one byte is required for attenuator and noise updates.

The SSI-263A is a phoneme-based speech synthesizer (phonemes being the basic word sounds from which language is constructed) and is, therefore, capable of producing continuous, unlimited vocabulary speech. But unlike earlier chips of this type, the SSI chip contains five internal registers for continuously varying speech and sound effects. Registers are provided for varying pitch at two addresses (a total of 12 bits in two 8-bit registers), and allow a range of 4096 pitch settings. The remaining four bits of one register provides the capability of programming speech and sound effect rates so that pitch may glide from one setting to another during the pronounciation of several phonemes. This gives the SSI-263A the ability to produce highly intelligible and natural sounding speech. Other registers are provided for varying amplitude, articulation, vocal-tract filter settings, and phoneme output and duration. BASIC programming may be used to create high-quality speech synthesis, and MACHINE LANGUAGE programs are capable of producing remarkable speech and sound effects.

The Synphothizer is available for the OSI sixteen-pin bus, and uses sub-port locations 8 to 15 (C708 to C70F or 50952 to 50959). The first five addresses are used for the SSI-263A. Addresses 50957 and 50958 are used each for one of the two TI sound chips. Addresses 50948 to 50951 are not used by the Synphothizer, and an additional sixteen-pin socket (plus decoding for address bit 3) is provided for connecting to other 16-pin bus devices which use those addresses.

An on-board clock is provided to supply the 2MHz and 1MHz clock signals required by the SN7489As and the SSI-263A. The board will require +5V for the sound chip circuitry, and +9 to +12V for the amplifiers.

Currently available from HG Interfaces as a bare board with assembly and programming instructions, the Synphothizer Board is priced at $40.00

★

HOW TO PUT A BASIC PROGRAM AT ANY ADDRESS

By: John Whitehead
17 Frudal Crescent
Knoxfield 318, Australia

With CMOS memory available with battery backup, you may want to put your most used

BASIC programs in it or put them in EPROM as I have done. This is how to do it.

A BASIC program has to be written at the address it is RUN at or LOADed there from tape. It can't be moved there with the Extended monitor because each BASIC line is stored in memory with the address of the next line. In this example it is assumed 8K of RAM at $0000 to $1FFF and another 8K at $8000 to $9FFF.

The following are the BASIC work space pointers that have to be altered:

Pointer address	cold start value	new value for loading @ $8000	new value for editing or running	new value running only
$0079 text start LO	01	01	01	01
$007A text start HI	03	80	80	80
$007B text end LO	03	03	see note A	see note B
$007C text end HI	03	80		
$0085 memory end LO	00	00	00	see note C
$0086 memory end HI	20	A0	A0	

Note A. This has to contain the address of the next location after the three BASIC nulls that are at the end of every BASIC program. It is updated automatically. If you want to return to a program and edit it then this has to be noted before leaving, then restored when returning.

Note B. This can be the end of the new program or the end of a program that is in $0300 to $2000.

Note C. This is the end of memory and must be the same memory area as in $007B-$007C.

Procedure: - Do a cold start. Break M. Change contents of memory at $007A, $007C and $0086 as above. Put 3 nulls at the start of the new memory, i.e., 00 at $8000, $8001 and $8002.

Type in your BASIC program or LOAD it from tape. (This will be loaded at $8000.)

When it's working correctly then: - If your two sections of RAM are not continuous, as assumed above, you can now cold start to reset the pointers back to normal.

If your RAM is continuous, you will need to reset the pointers to the cold start values using the monitor and then warm start, or do a cold start but answer memory size with a value less than the start of your BASIC program in high memory.

Another program can now be loaded in normal memory. To use the program at $8000 POKE122,128 and RUN. (122=$7A 128=$80). To go back to normal POKE122,3.

★

A HISTORY OF MICRO MAGAZINE

By: Earl Morris
3200 Washington Street
Midland, MI 48640

The first issue of MICRO appeared in October of 1977. The 28 page magazine was subtitled "The 6502 Journal" and its purpose was to bring together users of 6502 machines. The major 6502 machine at that time was the KIM-1. Issue one also marked the first of a long series of articles penned by the infamous Mike Rowe (pun intended). The second issue appearing December 1977 contained an article on the OSI Challenger as well as the just released Commodore PET.

MICRO soon became subtitled "The Magazine of the Apple, Kim, Pet and Other 6502 Systems." Despite the growing number of articles and advertisements, OSI was still classified among the "other" computers. Finally, in December 1979, the OSI name appeared on the front cover of MICRO, courtesy of a design drawn by the publisher's 8 year old son. By the end of 1979, circulation had grown to 3825 with a 78 page magazine. My complaints about lack of OSI recognition were finally heard when starting with January 1980, the banner contained the names of "AIM APPLE KIM PET ATARI OSI SYM," all of which were 6502 machines. Visibility of OSI was even greater in the February 1980 issue with a photograph of the Challenger 4P on the front cover.

The March 1980 issue of MICRO anticipated a 16 bit version of the 6502. This vision has only recently become a reality with the Western Digital 65SC816 chip. More OSI vendors began to notice MICRO, since in April 1980 the first Aardvark Software ad appeared.

Even OSI was aware of the growing amount of OSI related material in MICRO. In that same issue an OSI full color ad appeared on the back cover, and the "Small Systems Journal" appeared within. The OSI ad was on the back cover for every issue through May 1981. The June 1980 issue introduced the cover format of looking out into the world from inside of a terminal. Thus the backwards writing is created. This cover format was continued in all future issues.

The August 1980 issue contained results of the first (and only) MICRO limerick contest as well as the introduction of a column devoted to OSI entitled, "Up From Basements." During 1980 most issues had two or three OSI related articles. The average circulation during 1980 was 10521.

In March of 1981 a new OSI column "Challenges" began. There were many changes in the June 1981 issue. The magazine looked different with a new type of binding and expansion to 112 pages. However, the OSI ad was missing from the back cover and was never to reappear. This corresponded with the buy out of OSI by M/A COM. MICRO announced both the 6502 and the 6809 CPUs would now be covered. The banner was now "The 6502/6809 Journal." The July 1981 issue contained a special section devoted to OSI with five articles. This issue also carried the final "Small Systems Journal" and the last regular editorial by Robert Tripp, the founder of MICRO.

More OSI related vendors dis-

covered MICRO as D&N Microproducts ran their first MICRO ad in August 1981. The September 1981 issue presented the results of the reader survey. The editors of MICRO were astounded to learn 39% of their readers owned OSI machines. The November 1981 MICRO was devoted to games. The editorial questions how many serious computer users would stoop to running games on their machines. The average circulation during 1981 was 13360.

The March 1982 MICRO again contained an OSI special section with five articles. The same issue ran an editorial entitled "Hello, OSI ?." The editor tried to explain the apparent disappearance of OSI and the frustration of trying to contact the company. The July 1982 issue carried a response to the "Hello, OSI ?" editorial. Phillip Johnson wrote, "M/A COM Office Systems intends to continue its presence in the personal computer market and to support our customer base." July 1982 also marked the exit of Mary Ann Curtis who handled many of the OSI articles at MICRO.

During the remainder of 1982, the editors of MICRO pushed the 6809 CPU and requested articles on machines using this processor. The focus of the editors seemed to move towards software and languages in contrast to the earlier hardware and machine code. The scope of MICRO was expanded to include the 68000. During 1982 many OSI related articles and advertisements continued to appear in MICRO despite the absence of OSI itself. The MICRO OSI book was announced in December 1982.

In January of 1983 the banner was changed to "Advancing Computer Knowledge" to reflect the widening scope of the magazine. The editors requested articles about the TRS-80 machine. Many more changes were made early in 1983: The "Learning Center" feature was created for beginners, a new publisher, addition of more color and more pictures, change from MICRO INK to MICRO. The mailing address changed from Chelmsford, MA to Amherst, NH. Between March and May of 1983, MICRO became a rather different magazine. The number of OSI related articles dropped off sharply. In March 1983 the "Micro on the OSI" book was offered for sale.

In the August 1983 editorial, MICRO announced they would no longer carry OSI related articles. On information they obtained from M/A COM and Kendata, it appeared OSI was no longer selling computers into the personal market. MICRO had carried more OSI articles than any other magazine over the last six year period. However, editorial space was now needed to cover the more current, popular personal machines. MICRO suggested OSI users buy the MICRO OSI book and subscribe to one of the OSI newsletters.

At that time, I had no further interest in MICRO and canceled my subscription. Very quickly the OSI vendors withdrew their ads since the magazine was now aimed at an entirely different audience. The copies of MICRO from October 1977 through August 1983 are stored on a shelf over my computer. They are a treasure house of OSI information second only to PEEK(65).

As a postscript, I happened to pick up the October 1984 copy of MICRO. The difference between it and the August 1983 issue was striking. The October 1984 issue had 68 pages and 16 advertisers while the August 1983 issue contained 144 pages and 91 advertisers. MICRO ceased publication with that October 1984 magazine after a total of 76 issues. In March 1985 it was announced that subscribers to MICRO will receive copies of Dr. Dobbs Journal to balance out their remaining subscription. Thus the end of an era.

The rise and decline of MICRO and OSI appear entwined, both occurring in the same time frame. However, I believe this is part of a larger story of the micro-computer industry in general. In the mid 70's industrious individuals started computer related businesses on a shoestring in garages and basements. By the mid 80's the business had grown to the point the original founder could no longer run it. Either he sold out, or at least took a back seat and hired business managers. Business decisions were then often made to change the product line or to concentrate on an expanded market. Very often the changes alienated the original customer base.

Thus ends my history of MICRO magazine as viewed by an OSI user. The computer industry is much changed since MICRO #1 appeared in 1977 to fill the needs of computer hobbyists hungry to read anything about the 6502.

★

TAPE TO DISK PROGRAM CONVERSION: MINOS

By: Jim McConkey
7304 Centennial Rd.
Rockville, MD 20855

This article is the direct result of Editor Eddie's prodding (see, it really does work!) and specifically addresses Gary Florence's letter in the Dec. '84 issue of PEEK on the conversion of the popular maze game "Minos" by Alan Stankiewicz and Bruce Robinson to run on disk-based system. The conversion process is equally applicable to non-games as well. Since I run under HEXDOS, the conversion will be geared to it, though the procedure is directly applicable to OS65D with minor modifications.

The first step in the conversion process is, of course, to load the program from tape into the computer, which has been booted from the disk. Watch out for auto-run games on tape. Hit the space bar to quit the load before the RUN gets loaded. After the program has been loaded, it should be saved onto disk before anything else. The next step should be the obvious - try running it (but remove the disk first, just in case). Many games, especially adventures, will run without modification. If the program won't run, acts funny, or runs but leaves the operating system unable to read the disk, etc., the program must be modified.

The two main causes of tape-disk program incompatibility are machine code subroutines called through USR functions, and data storage accessed by PEEKs and POKEs. Many programs use the empty space under ROM BASIC between $0222 and $02FF to store machine code routines. HEXDOS, for example, uses this space for disk file allocation, and putting a routine there messes up the OS.

If the program uses graphics, HEXDOS users should POKE 227,255 at the beginning of the program and POKE 227,127 at the end. This is necessary because HEXDOS uses the byte at 227 to mask all printed characters to solve OSI's

mysterious half-graphic error messages. If the program does any sort of tape I/O for file storage, etc., this must be changed to the equivalent disk I/O commands. See your OS handbook for details.

The best way to learn is by doing, so on to the conversion of MINOS. One basic rule is to be suspicious of all POKEs. It helps here to be slightly familiar with your machine, know the locations of the memory-mapped screen, the keyboard, joystick, etc.. Also, know the most common monitor ROM routines, like that to get a character from the keyboard at $FD00. This will enable you to know legitimate POKEs from storage and machine code subroutine areas. In lines 10-20 we see some DATA statements which look (at least to me) like a machine code program. The READ : POKE (read: SUSPICIOUS!) combination in line 60 and the USR call in 65 seem to confirm this. The next step is to disassemble this code. Listing 3 shows the subroutine in the DATA statements at 10 and 20. A little experience shows it to be a quick screen clear routine. Knowing that this is a screen clear simplifies matters for HEXDOS users, since they can simply PRINT CHR$(3) to do the same thing. Listing 2 shows the modified listing. The data statements and the USR set up are no longer necessary. Be aware that the USR set up is different under both HEXDOS and OS65D. HEXDOS users must POKE at 240 and 241 instead of 11 and 12. Also, under HEXDOS, all X=USR(X) calls must be changed to X=USR(-7).

The address Q that showed where the subroutine was to be put came from SR in line 30. This indicates an examination of the subroutine at 1310 is in order. SR is set to 600 in line 1310. This is in the area, described earlier, which now contains the file allocation information and is no longer free. So far this is not a problem, since we have eliminated the need for a USR, but checking further, we find several spaces before SR are used in lines 88 and 99 for variable storage. This must be relocated. I simply made SR=10000 in line 1310 to save the variables somewhere in the no-man's land between the program and string spaces. Although this works, it is a kludge at best. The preferred method is to lower the top of free memory by the appropriate POKEs and to put the data in a protected space at the end of memory.

LISTING 1 - TAPE VERSION OF MINOS

```
10 DATA162,0,169,32,157,0,208,157,0,209,157,0,210,157,0,211,
   157,0,212
20 DATA157,0,213,157,0,214,157,0,215,232,208,229,96
30 GOSUB1310:Q=SR
50 POKE12,Q/256:M=PEEK(12):POKE11,Q-256*M
60 READM:POKEQ,M:IFM<>96THENQ=Q+1:GOTO60
65 X=USR(X)

1200 Q=SM+1:PRINT:X=USR(X)

1310 SR=600:VM=20:HM=20:IFPEEK(57088)<127THENM=40

2635 X=USR(X)

9000 POKE240,0:POKE241,253:J=USR(X)
9005 J=PEEK(531)

25009 SM=7000

30005 X=USR(X):V=2:GOSUB1200:POKE11,0:POKE12,253:X=USR(X)
```

LISTING 2 - HEXDOS VERSION OF MINOS

```
10 POKE227,255
30 GOSUB1310:Q=SR
65 PRINTCHR$(3)

1200 Q=SM+1:PRINT:PRINTCHR$(3)

1310 SR=10000:VM=20:HM=20:IFPEEK(57088)<127THENM=40

2635 PRINTCHR$(3)

9000 POKE240,0:POKE241,253:J=USR(-7)
9005 J=PEEK(531)

25009 SM=10000

30005 PRINTCHR$(3):V=2:GOSUB1200:POKE11,0:POKE12,253:X=USR(X)
```

LISTING 3

```
10      0000              .OPTION L 2 S 2
                          ;
                          ;CLEAR SCREEN ROUTINE
                          ;USED IN MINOS GAME
                          ;
100     0000 A2 00        CLS   LDX#   0       ;INIT COUNTER
110     0002 A9 20              LDA#   32      ;PUT ASCII BLANK IN A
120     0004 9D 00 D0     LOOP  STAX   $D000   ;CLEAR SCREEN PAGE 0
130     0007 9D 00 D1         STAX   $D100   ;  "  PAGE 1
140     000A 9D 00 D2         STAX   $D200   ;  "  PAGE 2
150     000D 9D 00 D3         STAX   $D300   ;  "  PAGE 3
160     0010 9D 00 D4         STAX   $D400   ;CLEAR COLOR PAGE 0
170     0013 9D 00 D5         STAX   $D500   ; " PAGE 1
180     0016 9D 00 D6         STAX   $D600   ; " PAGE 2
190     0019 9D 00 D7         STAX   $D700   ; " PAGE 3
200     001C E8               INX            ;INCREMENT COUNTER
210     001D D0 E5            BNE    LOOP    ;DO 256 TIMES
220     001F 60               RTS
```

Continuing down the program, we find X=USR(X) in line 1200. By tracing the program execution, we see that this is a call to the clear screen routine, so this is replaced by a PRINT CHR$(3). The same applies to line 2635. Continuing, at 2520 we find another POKE referencing SM. Like before, it is prefaced by a GOSUB in 2500. Checking this subroutine we encounter SM=7000 in 25009. This seems to be some sort of storage for the maze and it, too, must be moved. Like before, I cheated and changed this to 25009 to have SM=10000. This does not conflict with SR=10000 because the storage using SR is at SR-1, 2 and 3 and the storage using SM is at SM+.

In lines 9000-9005 we first see PEEKs from 11 and 12. Remember that this is the USR vector under ROM BASIC. The address of the screen clear routine is being saved in A and B and a new USR calling address is being set. The POKE 11, 0 : POKE 12,253 is probably the most often used routine in games. This subroutine is in the monitor ROM and waits for a key to be pressed on the keyboard and returns the ASCII code for the key in location 531. In our case, since we are not using the screen clear call, we can eliminate the saving of the USR vector in A and B, although in general, this is a good idea. The USR vector must be changed because HEXDOS expects it at 240,241 and the X=USR(X) must be changed to X=USR(-7).

Finally, at line 30005, we find two USR calls. The first is just called with no vector set up, so we know this is the screen clear and we change it to PRINT CHR$(3). The vector is changed (who has seen this before?) and the USR is called again. This is the GETKEY routine again. It waits for the user to press a key and then runs the program again. The RUN in 30010 could be changed to ask the player if he wants to play again, if you want. Otherwise, the only way to quit is ctrl-C (or <repeat> for HEXDOS users).

To recap: know your machine, watch out for POKEs and USR calls. Also, to be held under suspicion are variables set to a nice round number near 8000. The stock C1P has 8K of memory, user storage is often put near the end of memory. Don't forget to make changes required by your operating system, such as different set up vector locations and different commands for tape and disk I/O.

Next: how to relocate machine code without knowing what it does.

★

OS-65U SELECTIVE SEARCH & PRINT PROGRAM

By: Raymond D. Roberts
P. O. Box 336
Ferndale, WA 98248

Have you ever needed a program that would selectively search and print records? I offer this program in the hope that it will help someone. It is a rather basic (no pun) program and I'm sure some of you expert "hackers" will call it simple. PEEK(65) has asked all of us to share and so this is my contribution. Please remember that some of us are USERS, not programmers. While I find that my desire and interest in programming steadily increases, my time to do so does not. I am deeply indebted to you folks who contribute your knowledge and ideas with us beginners.

"ADS 100"

This OS65U program handles random record data files I have created and filled with the OS-DMS Nucleus. It assumes that the data file is named "ADS". More on this later.

The program searches on a 3 digit numerical variable in the first field of the record, i.e, 000 thru 999. This is controlled by LINE 33035.

In my application, I use a 12 field record with the first field 3 characters long (see diagram 1). I assign a number upon data entry of the record. Suppose I have 300 records on autos. I then classify "red Fords" under 100, "blue Fords" under 101, "red Chevrolets" under 102, etc..

If I want a printout of all records on "blue Fords", I make a search and print with "ADS 100" and it prints all records with 101 in the first field.

You should be able to see many applications for use of such a program.

If you want to search on "Ford", "Robin", "Appaloosa", or some other string variable, change LINE 33035 as follows:

```
FROM: TX<>INDEX(1)THEN INDEX
      <1>=INDEX(1)+1:GOTO3005:
      REMMIDDLE

TO: IFINDEX(1)>TX+17THENINDEX
    <1>=INDEX(1)+1:GOTO3005:
    REMMIDDLE
```

(Also, change first field length from 3 chraracters to 30.)

Another feature of this program is a built in "expiration". In my application, I want records to be kept for 30 days. I use a 12 field record with field 11 assigned to hold DATE, i.e., (YMMDD) as in 40124 = 84 Jan 24th. Be sure you maintain this form of data entry for data file records unless you are prepared to modify ADS 100.

Remember, records are searched and printed IF they are less than 30 days old. You can circumvent this feature by entering 00/00/00 to the request of "todays date?".

Also, only fields containing data are printed. This is controlled by LINES 33300 to 33320. LINE 33310 deletes printing of fields 1, 11, and 12 (see diagram 3). In the diagram samples, field 12 (codes) is an unused field.

This program is highly modifiable by most anyone with some understanding of BASIC. I encourage beginners like myself to experiment with it as I have found I learn more by experimenting.

"ADS 200"

"ADS 200" is an auto delete program used to delete expired records used with "ADS 100". It inserts a "^P" in the first field of any record older than "todays date", while at the same time, printing an audit on console or printer. Data files then can be repacked with "PACK" on OS-DMS Nucleus.

If other than "ADS", "PASS" is preferred for your data file, you will have to delete line 110 from ADS 100 and replace it with lines 96 and 110 from "ADS 200".

"ADS 200" next month.

DIAGRAM 1

```
Sample Record
Created by OS-DMS NUCLEUS

FILE: ADS 0
NUMBER OF RECORDS: 950

CLASSIFICATION  ---
ITEM            ------------------------------
DESC-1          ------------------------------
DESC-2          ------------------------------
DESC-3          ------------------------------
DESC-4          ------------------------------
PRICE           ----------
PHONE           ------------------------------
CITY            ----------
STATE           ---
EXP DATE        -----
CODES           --
```

DIAGRAM 2

```
Sample Record
With Data entered

RECORD:  1          INDEX: 171
CLASSIFICATION      123
ITEM                JOHN DEERE 2040
DESC-1              LOW HOURS
DESC-2              ORCHARD PROFILE
DESC-3              0
DESC-4              0
PRICE               $8950.
PHONE               509-662-1566
CITY                WENATCHEE
STATE               WA
EXP DATE            40124
CODES               0
```

DIAGRAM 3

Record printed via ADS 100

```
JOHN DEERE 2040
LOW HOURS
ORCHARD PROFILE
---> $8950.
WENATCHEE, WA
509-662-1566
```

Listing on Page 16.

"ADS 100"

```
1 REM ===== A D S 1 0 0 ===== Copyright 1983 R.ROBERTS
2 REM  1-5-83 R.D.ROBERTS POB 336 FERNDALE,WA 98248
3 REM ----))) PRINT ADS FOR A SPECIFIC CLASSIFICATION (((--
4 REM - THIS PROGRAM WILL FORMAT AND PRINT ADS FOR A
5 REM - REQUESTED CLASSIFICATION, BYPASSING THOSE WHICH
6 REM - ARE EXPIRED, BASED ON TODAY'S DATE
7 REM -----3/7/82 MARK MOVES WITH ( AND )
8 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6:K7=7:K8=8:K9=9
9 P9=1:FLAG6:FLAG9:FLAG11:FLAG21:FORX=1TO23:PRINT:NEXT
10 CL$="NO"
30 POKE 2976,44:REM ALLOW , TERMINATION
31 GOSUB 62400:REM FILL VARIOUS ARRAYS
35 SP$="--------------------":SP$=SP$+SP$+SP$+SP$+SP$+SP$
36 SD$="                    ":SD$=SD$+SD$+SD$+SD$+SD$+SD$
39 POKE 2976,13
40 CLOSE
50 PRINT"ADS100 - AD PRINT PROGRAM"
55 PRINT
56 PRINT"THIS PROGRAM PRINTS VERTICAL FORMATTED ADS FOR A
57 PRINT"REQUESTED CLASSIFICATION, OR FOR ALL CLASSIFICATIONS"
58 PRINT"BETWEEN GIVEN RECORD NUMBERS, BYPASSING EXPIRED ADS."
59 PRINT
60 T=PEEK(14387):POKE14457,(T):POKE15908,(T):REM PRNTR CNTRL
61 INPUT"PRINT ON CONSOLE(C) OR PRINTER(P) OR QUIT (Q)";D$
62 IF D$="P" THEN DV=5
63 IF D$="C" THEN DV=2
64 IF D$="Q" THEN 51100
65 IF DV=0 THEN PRINT"WHAT!!":GOTO61
66 POKE 14639,255:POKE2073,76
70 PRINT:INPUT"ENTER TODAY'S DATE (MM/DD/YY)";DT$
71 IFLEN(DT$)<>8THENPRINT"*** ILLEGAL ENTRY ***":GOTO70
72 IFMID$(DT$,3,1)<>"/"THENPRINT"*** ILLEGAL ENTRY ***":GOTO70
73 IFMID$(DT$,6,1)<>"/"THENPRINT"*** ILLEGAL ENTRY ***":GOTO70
75 XX$=RIGHT$(DT$,1)+LEFT$(DT$,2)+MID$(DT$,4,2):REM YMMDD
76 EX=VAL(XX$):REM VALUE OF EXPIRATION DATE YMMDD
90 INPUT"ENTER DEVICE THE AD FILE IS ON";MD$
91 IF MD$<>"A"  AND MD$<>"B" THEN GOTO90
92 DV(2)=PEEK(9832): IF DV(2)>127 THEN DV(2)=DV(2)-128+4
94 DEV MD$
110 MN$="ADS  ":MP$="PASS"
118 MN$=MN$+"0":
160 OPEN MN$,MP$,1: REM OPEN AD FILE
170 INDEX(1)=0: INPUT %1,N%:
190 INDEX(1)=6: INPUT %1,TY:
210 INDEX(1)=9: INPUT %1,EODF:
220 INDEX(1)=20: INPUT %1,BODF:
250 INDEX(1)=31: INPUT %1,RL:
260 INDEX(1)=42: INPUT %1,NR:
280 IF(EODF(=BODF)ORNR(1THENERR$="FILE EMPTY":GOTO40000
290 DIML$(20),FP(20):REM CONTENTS & POINTERS
291 DIMA$(20)
300 INDEX(1)=53:N=1:NF=1:TT=0:TF=1
305 INPUT%1,T$:INPUT%1,T
310 A$(N)=T$:FP(N)=TT:REM FIELD LABELS AND DESCRIPTIONS
320 IFINDEX(1))=BODFTHEN360
330 N=N+1:NF=NF+1:REM NF IS NUMBER OF FIELDS
340 TT=TT+T:REM RECORD LENGTH
350 GOTO305
360 MN$="CLASS":MP$="PASS"
361 MN$=MN$+"0":
362 IF CL$="NO"THEN GOTO500
365 OPEN MN$,MP$,2:REM OPEN CLASSIFICATION DESC FILE
370 INDEX(2)=0: INPUT %2,N$
380 INDEX(2)=9: INPUT %2,EODF(2):
385 INDEX(2)=20: INPUT %2,BODF(2)
390 INDEX(2)=31: INPUT %2,RL(2)
391 INDEX(2)=53:C=1:CF=1:TT=0:CF=1
392 INPUT%2,T$:INPUT%2,T
393 C$(C)=T$:CP(C)=TT
394 IFINDEX(2))=BODF(2)THEN500
395 C=C+1:CF=CF+1
396 TT=TT+T:REM RECORD LENGTH
397 GOTO392
500 REM PARAMETERS FETCH
501 TY=0
505 INPUT"ENTER CLASSIFICATION NUMBER OR 'ALL'";CX$
506 REM GOSUB 8000 IF THERE IS A CLASS FILE
507 PRINT "PROCESSING CLASSIFICATION # ";CX$
508 INPUT"IS THIS WHAT YOU WANT? Y/N ";YN$
509 IF YN$<>"Y"THEN505
510 IF CL$<>"NO"THENCLOSE2
550 IF CX$<>"ALL"THEN 700
560 INPUT"ENTER BEGINNING RECORD #";BE
565 IF BE<1 THEN PRINT"WHAT!!":GOTO560
566 BE=BE-1
570 INPUT"ENTER ENDING RECORD #";EN
571 EN=EN-1
700 H1$=RIGHT$(C$,3)+"  "+D$:REM HEADING DESCRI
720 GOSUB30000:REM GET A REC
721 IF NN=1 THEN GOTO 950:REM EOF DONE
723 IF DV=2 THEN 800
724 GOTO 800:REM NO HEADINGS **********
725 IFPEEK(15908)(L+5THENGOSUB735
726 IFPEEK(15908)(L+5THENFORX=1TOPEEK(15908):PRINT#DV:NEXT:REMSKIPPG
727 IFPEEK(15908)((PEEK(14457)-L+5)THEN800
728 PRINT#DV,CHR$(14);TAB(10);H1$;
732 PRINT#DV,"--) PRINTED ON ";DT$
733 GOSUB 735:REM HEADING
734 GOTO 800
735 FOR I=1TO80STEP5:PRINT#DV,"=====";:NEXTI:PRINT#DV
770 RETURN
800 REM ----------FORMAT AD LINES
805 PRINT#DV
840 PRINT#DV,L$(2)
850 FOR I=3TO6
855 IF L$(I)<>""ANDL$(I)<>"0"THENPRINT#DV," ";L$(I)
860 NEXTI
865 IF L$(7)<>""ANDL$(7)<>"0"THENPRINT#DV," ---) ";L$(7)
870 IF L$(9)<>""ANDL$(9)<>"0"THENPRINT#DV," ";L$(9);",  ";
880 PRINT#DV,L$(10)
890 IF L$(8)<>""ANDL$(8)<>"0"THENPRINT#DV," ";L$(8)
940 GOTO 720
950 REM
990 CLOSE1:GOTO51100
5010 IF CL$<>"NO" THEN CLOSE 2
8000 REM  GET CLASSIFICION
8005 IF CL$="NO"THEN C$=CX$:D$="UNKNOWN..NO CLASS FILE":GOTO8060
8015 INDEX(2)=BODF(2)
8020 FIND CX$,2
8030 IF INDEX(2))=EODF(2)THENC$=CX$:D$="?????":GOTO8060
8040 INPUT%2,C$:REM CLASS NUMBER
8050 INPUT%2,D$:REM DESCRIPTION
8060 RETURN
30000 REM----------SEARCH ADS FILE
30001 IF CX$<>"ALL"THEN30005
30002 IF TY(BETHENINDEX(1)=(BE*RL)+BODF
30003 IF TY)EN-1THENNN=1:GOTO33990
30004 GOTO30010
30005 FIND CX$,1:REM FIND AN AD FOR THE NEEDED CLASSIFICATION
30010 IF INDEX(1))=EODF THEN NN=1:GOTO 33990:REM ALL DONE
33021 TY=INT((INDEX(1)-BODF)/RL):REM COMPUTE RECORD WE FOUND
33022 TX=(TY*RL)+BODF: REM THEN FIND OUT IF WE HIT IN THE MIDDLE
33035 IF TX<>INDEX(1) THEN INDEX(1)=INDEX(1)+1:GOTO 30005:REMMIDDLE
33236 FOR I=1TONF
33237 INPUT%1,L$(I):REM GET ALL FIELDS OF AD
33240 NEXTI
33250 IF L$(1)="^P"THEN 38000
33260 EY=VAL(L$(11))
33270 IF EY(EXTHEN38000
33300 REM DETERMINE NUMBER OF LINES REQUIRED THIS AD
33305 L=0
33310 FOR I=2TONF-2:REM ALL BUT CLASSIFICATION, EXDATE,PRINTCODES
33315 IF L$(I)<>"" AND L$(I)<>"0"THEN L=L+1
33320 NEXTI
33990 F1=K1:RETURN
38000 REM
38010 IF CX$<>"ALL"THEN 30005
38015 TY=TY+1
38020 INDEX(1)=BODF+(TY*RL):GOTO30000
40000 REM- ERROR
40010 PRINT: PRINT ERR$: PRINT: CLOSE 1: GOTO 51100: REM COMMON EXIT P
50000 REMDISK ERROR HANDLER
50010 ER=PEEK(10226): EL=PEEK(11774)+PEEK(11775)*256
50025 REM CHK FOR 'CHANNEL ALREADY OPEN ERROR'
50030 IF ER=133 THEN CLOSE: GOTO EL
50040 IF ER=128 THEN ERR$="INVALID FILE NAME": GOTO 51050
50050 IF ER=132 THEN ERR$="END OF FILE ERROR": GOTO 51000
50060 IF ER=130 THEN ERR$="ACCESS RIGHTS VIOLATION":GOTO 51050
50070 IF ER=129 THEN ERR$="CANNOT ACCESS FILE ": GOTO 51050
```

```
50075 REM OTHER ERRORS ARE HARD ERRORS
50080 ERR$="DISC ERROR CODE "+STR$(ER)+" IN LINE "+STR$(EL)
50094 EA=0: FOR I=4 TO 1 STEP -1: EA=EA*256+PEEK(9889+1): NEXT I
50096 DV(3)=PEEK(9832): IF DV(3))127 THEN DV(3)=DV(3)-128+4
50098 PRINT"ERROR ON DEVICE "+CHR$(DV(3)+65)+" AT DISC ADDRESS";EA
51000 REM-RROR EXIT
51020 CLOSE 1
51040 REM ENTRY AT '51050' DOES NOT CLOSE THE CHANNEL
51050 PRINT:PRINT:PRINT"*****  ERROR  *****": PRINT: PRINT
51060 PRINT ERR$
51100 REM------------------- COMMON EXIT
51110 DEV CHR$(DV(2)+65): REM SELECT ORGINAL DEVICE
51120 FLAG 6: REM ENABLE PROGRAM ABORT ON EOF HIT ERROR
51130 FLAG 22: REM ENABLE BASIC'S IMM. MODE
51140 FLAG 12: REM DISABLE SPACE SUPPRESSION
51150 FLAG 10: REM ENABLE PROGRAM ABORT ON DISC ERROR
51152 REM FORI=1TOPEEK(15908):PRINT#DV;NEXTI
51155 INPUT"WANT TO DO ANOTHER RUN? Y/N ";YN$
51156 IF YN$="Y" THEN RUN
51160 STOP
62400 REM --FILL MISC WORK ARRAYS
62410 DIM M1(13),M2(13),MM$(13)
62420 FOR M=1TO13:READ MM$(M),M1(M),M2(M):NEXTM
62430 DATA JANUARY,0,0
62431 DATA FEBRUARY,31,31
62432 DATA MARCH,59,60
62433 DATA APRIL,90,91
62434 DATA MAY,120,121
62435 DATA JUNE,151,152
62436 DATA JULY,181,182
62437 DATA AUGUST,212,213
62438 DATA SEPTEMBER,243,244
62439 DATA OCTOBER,273,274
62440 DATA NOVEMBER,304,305
62441 DATA DECEMBER,334,335
62442 DATA YEAREND,365,366
63000 RETURN
```

★ ★ ★

WAZZAT CORNER!

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

The best utility that enhances DOS 3.2 and DOS 3.3 is "HOOKS" by R. Trethewey, published in PEEK(65), Dec '83. It is possible to add one's own code to "HOOKS". One example is the disk stop/start code listed here. The code was written to control a single 8" drive, toggling it on and off with the command "Z*". (In fact, probably all four drives will be de-selected). The listing supplies all the pieces required for incorporating the code into "HOOKS". If in "HOOKS", change line 1200 to "JMP UPDATE".

The program can be used as it stands. Assemble it and call the routine "DISK", i.e., CALL $A00B. If BASIC does not support CALL then use an X=USR(X) call,

```
POKE 8955,11:POKE 8956,160:
     X=USR(X)
```

The values "11" and "160" would have to be changed if the program was assembled to an address different to $A000.

If it is required to stop/start more than one drive, then experiment with these values. (Selecting a drive is done with "DISK! "SE A"). The first number goes into $C000 and the second into $C002.

Select drive A - $40, $FF

Select drive B - $00, $FF

Select drive C - $40, $DF

Select drive D - $00, $DF.

In BASIC

Is there an easier way? Yes, from BASIC. Incorporate these two lines into your programs, (49154=$C002).

```
95 ZZ=-255

5000   ZZ=NOT(ZZOR254):  POKE
49154,ZZ*-1: FOR C=1 TO  1200:
NEXT: RETURN
```

For a 1 MHz computer change 1200 to 600.

To understand what is happening, try this program:

```
95 ZZ=-255

100 GOSUB 5000: PRINT ZZ;:
    INPUT A$: GOTO 100

5000 ZZ=NOT(ZZOR254): RETURN
```

OFF dear

The program above uses Boolean Algebra, here is a similar "toggle" example,

```
10 T$(1)="OFF dear":T$(2)="ON"
   :V=1

20 GOSUB 100

30 V=(NOT (V and 1))*-1:PRINT
   T$(V)

40 GOTO 20

50 END

100 INPUT A$:RETURN
```

Try different values for "V" in line 10. For example: 3, 4, 10, 100, -100, and even 0. Wazzat!

POKE list

Here is a list of useful POKEs for DOS 3.3.

POKE 916,0 - get rid of "Ok".

POKE 13026,128 - new cursor.

POKE 11241,13 - 13 sectors per track.

POKE 9610,201: POKE 9611,14 - toggle printer on and off with CTRL-N.

POKE 2888,0:POKE 8722,0 - accept null input on <RETURN>.

```
  10                ; DISK STOP/START &
                                  DELAY
  20                ; by LZJ
  30                ;
  35 A000                   * = $A000
  37                ;
  40 A000 C95A              CMP #'Z
  50 A002 F000              BEQ ONOFF
  60                ;
 100 A004 4C0BA0    ONOFF   JMP DISK
 110                ;
1000 A007 00        TOGGLE  .BYTE $0
1010 A008 00        ONE     .BYTE $0
1020 A009 00        TWO     .BYTE $0
1030 A00A 04        THREE   .BYTE $4
1040                ;
1050 A00B AD07A0    DISK    LDA TOGGLE
1060 A00E F01C              BEQ DIOFF
1070 A010 8D02C0            STA $C002
1080 A013 EE07A0            INC TOGGLE
1090 A016 CE08A0    DELAY   DEC ONE
1100 A019 D0FB              BNE DELAY
1110 A01B CE09A0            DEC TWO
1120 A01E D0F6              BNE DELAY
1130 A020 CE0AA0            DEC THREE
1140 A023 D0F1              BNE DELAY
1150 A025 A904              LDA #4
1160 A027 8D0AA0            STA THREE
1170 A02A D006              BNE BK
1180 A02C 8D02C0    DIOFF   STA $C002
1190 A02F CE07A0            DEC TOGGLE
1200 A032 60        BK      RTS
```

This will probably be the final "Wazzat!" column, and so its contents are particularly exotic. The exotica are found in the two listings explained later. But first, a couple of useful tips.

When a file is being created the disk operating system asks how many sectors are required, and offers up to twelve. But the "PUT" command ignores this and writes 11! This is crazy! Eleven sectors of 256 bytes, times 76 tracks, equals 214,016. But if thirteen sectors could be written, then disk capacity would expand by 38,912 bytes, to 252,928. To gain this 18% increase in disk capacity, add this line to BEXEC*:

```
9 POKE 11241,13: REM 13 Sectors.
```

Yup, it's that simple. A check with SECDIR reveals that track headers are now being written with "0D" - thirteen sector tracks. A test with DOS command "EXAM" also shows that a thirteenth sector of data is being written. No ill effects have been noticed after several weeks use of the new disk capacity. (Tested with DOS 3.3, and it should work with DOS 3.2.)

Recently I had a program named "00DEMo" which refused to load. The dreaded, "Error #9" was reported. But I was still able to easily load the program from BASIC - not by name, but by track number. Interestingly, initializing that track and saving the program back to it did not solve the problem. But changing the program name to "DEMO00" did. Coincidence? Here's the procedure to follow when confronted with DOS error #9:

create a new program file

enter the DOS kernel (type "EXIT" in BASIC)

load the program by track number

save the program to the new program file and

curse Murphy if it didn't work!

Now for the exotica. For years I wondered what to make of USR(X) disk operations and finally found a use for them when writing an OS65U single disk copier. Probably a similar use could be found under OS65D 3.2 and OS65D 3.3.

Listing 1 demonstrates how the command could be used to load a BASIC program under DOS 3.2, but not 3.3. Is it true that long, long ago, in the middle seventies, OSI 65D 1.0 users had to use this method to load and save program?!

```
1 REM Listing 1. Use with DOS 3.2
2 :
10 INPUT"Track # ";TN
20 X=9822       : REM first RAM location
30 POKE X,1     : REM Begin with sector one.
40 POKE X+1,11  : REM Pages to read.
50 POKE X+2,121: REM lo-byte $79 of $3179.
60 POKE X+3,49  : REM hi-byte $31 of $3179.
70 POKE X+4,TN  : REM Poke track number.
80 POKE 8917,3  : REM Poke code '3' for READ, Disk Drive A.
90 X=USR(X)     : REM Read in data from disk.
```

```
1 REM Listing 2
2 :
10 REM READ A TRACK HEADER by LZJ NOV 84
20 GOTO 50
30 DISK!"EXAM 7000=61": RETURN
40 :
50 PO=15041: DISK!"HOME": POKE PO,48: POKE PO+1,48: REM $3AC1
60 DEF FN A(X)=10*INT(X/16)+VAL(RIGHT$(STR$(X-16*INT(X/16)),1))
70 PRINT!(28): PRINT TAB(15)" READ A TRACK HEADER.": PRINT
140 INPUT "Starting track # ";N: PRINT !(28): IF N=0 THEN 140
150 PRINT "T BYTE","43","57","T#","58": PRINT : PRINT
170 FOR T=N TO 76: T$=STR$(T)
180 :   IF T<10 THEN POKE PO+1,ASC(MID$(T$,2,1)): GOTO 210
190 :    POKE PO,ASC(MID$(T$,2,1)): POKE PO+1,ASC(RIGHT$(T$,1))
210 :     GOSUB 30
220 :    FOR R=28672 TO 28676: X=PEEK(R)
230 :       IF X/16-INT(X/16)=0 THEN X=10*X/16: GOTO 250
240 :        X=FN A(X)
250 :         PRINT X,
260 :    NEXT R: PRINT
270 NEXT T
```

Listing 1 will not work under DOS 3.3 because the USR(X) code does not add $0900 to all the links stored in the program and its header. Yes, DOS 3.3 saves BASIC programs as if they were DOS 3.2 programs and then converts them back when loading! The idea behind this no doubt is to add some portability between the two disk operating systems.

Listing 2 is the heart stopper - a self-modifying BASIC program! If you intend to use it, omit lines 1 & 2, and type in lines 10, 20, and 30 exactly as shown. The program uses "EXAM" to read the first 5 bytes from each track header, from track 1 to 76. The program was originally written to test drive hardware. On my system the program does not give consistent results. Sometimes the first byte read is not "43", other times it is. Is the byte before "43" a timing byte, or just something left in the PIA registers? - see line 150. OSI literature reveals that the "43" and the "57" are a two byte track start code. The next number is the track number, followed by the track type code, a "58".

★

BIO-COMPATIBILITY PROGRAM

A new twist to an old standby for any OSI machine.

By: R. R. Groome
824 W. Main Street
Richmond, IN 47374

In the Radio Shack book, #62-2068 there was a nice bio-compatibility program that would not run for errors. Here is my revision.

In the listing, the CHR$(29) CHR$(31) type lines are printer commands.

If anyone wants a cassette copy, send me a cassette with a couple programs (anything!), and I will return both on the other side of cassette (C-60).

```
50 POKE12,83
80 K=0
100 DIM A1(30),B1(30)
110 DIM A(12)
120 FOR I=1TO12:READA(I):NEXT I
130 DATA 0,31,59,90,120,151,181,212,243,273,304,334
150 Y=1
155 PRINT"What is the name of person one";:INPUTW$
190 PRINT:PRINT"What is ";W$;"'s birthday (M,D,Y)";:INPUTM,D,Y
200 IF Y(1790THENY=Y+1900
210 E1=M:F1=D:G1=Y
220 GOSUB 770
230 Z2=T:K1=J+1
240 PRINT
250 PRINT"What is the name of person two";:INPUTX$
310 PRINT:PRINT"What is ";X$;"'s birthday (M,D,Y)";:INPUTM,D,Y
320 IFY(1790THENY=Y+1900
330 E2=M:D2=D:G2=Y
340 GOSUB 770
350 P2=ABS(Z2-T)
360 K2=J+1
370 :
375 PRINT
380 INPUT"Printer ready";Z$:SAVE
385 PRINT CHR$(29) CHR$(31)
390 PRINT"C O M P A T A B I L I T Y  A N A L Y S I S"
400 PRINT"----------------------------------------"
410 PRINT
420 PRINT"Compatability analysis of ";W$;" and ";X$;" ."
450 PRINT
470 PRINTW$;" was born on ";:M=E1:GOSUB950
480 PRINTF1;", ";G1;". It was a ";:J1=K1
490 GOSUB 1070
500 PRINT"."
510 PRINT
530 PRINTX$;" was born on ";:M=E2:GOSUB 950
540 PRINT D2;", ";G2;". It was a ";:J1=K2
550 GOSUB 1070
560 PRINT"."
570 PRINT
580 Z=P2
585 :
590 P3=ABS(INT(((Z/23)-INT(Z/23))*23))
600 S3=ABS(INT(((Z/28)-INT(Z/28))*28))
610 C3=ABS(INT(((Z/33)-INT(Z/33))*33))
620 P5=ABS(100-((2*P3)*(100/23)))
630 S5=ABS(100-((2*S3)*(100/28)))
640 C5=ABS(100-((2*C3)*(100/33)))
645 :
650 PRINT"Physical cycle compatability (23 day) is     ";
660 PRINTINT(P5*1000)/1000;"%"
670 PRINT"Emotional cycle compatability (28 day) is    ";
680 PRINT INT(S5*1000)/1000;"%"
690 PRINT "Intelligent cycle compatability (33 day) is  ";
700 PRINT INT(C5*1000)/1000;"%"
710 PRINT,,,"----------"
720 PRINT"Average compatability is ",," ";
730 A5=(P5+S5+C5)/3
740 PRINT INT(A5*1000)/1000;"%"
760 PRINT :LOAD:INPUT"Another";A$:A=ASC(A$)
765 IF A=78 THEN 1140
767 RESTORE:CLEAR:GOTO100
768 :
770 Y1=Y-1800
780 Q1=INT(Y1/4)
790 Q2=INT(Q1/25)
800 Q3=INT((Y1+200)/400)
810 K=0
820 IF Q1*4()Y1 THEN 860
830 IF Q2*100()Y1 THEN 860
840 IF Q3*400-200()Y1 THEN 860
850 K=1
860 T=365*Y1+Q1-Q2+Q3-K
870 T=T+A(M)+D-1
880 IF M(3 THEN 900
```

```
890 T=T+K
900 IF INT(Y1/4)()Y1/4 THEN 930
910 IF M)2 THEN 930
920 T=T-1
930 J=T-7*INT(T/7)
940 RETURN
945 :
950 IF M=1 THEN PRINT"January";:RETURN
960 IF M=2 THEN PRINT"February";:RETURN
970 IF M=3 THEN PRINT"March";:RETURN
980 IF M=4 THEN PRINT"April";:RETURN
990 IF M=5 THEN PRINT"May";:RETURN
1000 IF M=6 THEN PRINT"June";:RETURN
1010 IF M=7 THEN PRINT"July";:RETURN
1020 IF M=8 THEN PRINT"August";:RETURN
1030 IF M=9 THEN PRINT"September":RETURN
1040 IF M=10THEN PRINT"October";:RETURN
1050 IF M=11THEN PRINT"November";:RETURN
1060 PRINT"December";:RETURN
1065 :
1070 IF J1=1 THEN PRINT "Wednesday";:RETURN
1080 IF J1=2 THEN PRINT "Thursday" :RETURN
1090 IF J1=3 THEN PRINT "Friday";:RETURN
1100 IF J1=4 THEN PRINT "Saturday";:RETURN
1110 IF J1=5 THEN PRINT "Sunday";:RETURN
1120 IF J1=6 THEN PRINT "Monday";:RETURN
1130 PRINT"Tuesday";:RETURN
1140 END


1990 REM ORIGIANL IDEA FROM RADIO SHACK BOOK 62-2068
2000 REM PAGE 21....THAT DOES NOT WORK DUE TO ERRORS.
2010 REM OSI VERSION 1.0 BY R. GROOME
2020 REM CHANGES HEREIN ARE RELEASED FOR ALL NON-
2030 REM COMMERCIAL USES.
2040 REM GOOD FOR YEARS 1800 AND LATER
```

★ ★

RESOURCE PROGRAM LISTING DEVIATIONS EXPLAINED SEE PEEK(65) ISSUES FEB & MAR 1985

In the initial stages of preparing the resource article for publication, a copy of my Resource disk was used to reproduce the program listings in the interest of saving typing time. Unfortunately, when I submitted my disk for use, I forgot that my program listings had the recommended listing additions, as well as imbedded offset addresses for disk buffers, and was renumbered.

In the Feb. issue, page 6, midway down in column 3, it was recommended that lines 680, 690, 950, and 960 be added to Resource Two. Also, lines 610 and 620 were to be added to Resource Four. These additions were made in my copy of Resource and are in the published program listings.

The copy of the program listings sent to PEEK(65) has garbage added to the end of Resource Four. Since my copy program has never failed to produce a correct copy of the original, I did not expect anything less this time. In editing the article for errors after publication, this error was found.

In the March issue, page 9, under Resource 4, delete all lines after line 680.

Add the following lines:

```
690 PRINT TAB(10)"Z PAGE
    CROSS REF. FILE:"ZF$
700 PRINT TAB(10)"Z PAGE
    EQUATE FILE:"ZE$
710 PRINT TAB(10)"PASS 4
    COMPLETED"
720 PRINT:PRINT:END
```

All other program lines are correct.

Dana W. Skipworth
2055 W. 87 ST.
Cleveland, OH 44102

★

READER PROFILE

I have owned an OSI C8P-DF CHALLENGER since 1980. It was bought at a New York dealer, Polk's Aristocraft.

On the hardware side it has 48K RAM (three 520 boards) 505 CPU, 540 CRT controller and the 542 polled keyboard. The basic system has a dual 8" floppy, a NEC JB 1201 green monitor, a Zenith color monitor and an old BASE2 printer. Additional hardware is: a 565 (CA-14) VOICE I/O board with VOTRAX, a 572 (CA-21) 48 line I/O, a 570 (CA-20) 8 port I/O board with battery back up clock, a 575 (CA-24) Prototyping board and a 574 (CA-22) 12 bit Analog I/O board. I also have the BSR based AC-12P Remote Control Starter.

On the software side, I have OS-65D V3.2, OS65U V1.2, WP-2 word processor, Plot Basic, OS-Vocalizer I, DAC I and DAC II, various MDMS under 65D. Later, through a local dealer, now out of business, I bought copies of UCSD PASCAL V II.0, OS-65D V3.3, OS-65U V1.3, WP-3 (never worked), 65U DMS, QFS sort in Assembler, and SARGON II chess.

As you can see, an almost full blown system. It gives me a lot of fun, and profit too.

I'm an Electrical Engineer graduated 26 years ago, working in many fields. Among them are: instrumentation for Physics and for Astronomy at various universities, designing and supervising electronic projects, etc.. Some of these works have been published. During the past years, I have been involved with industrial electronics (control, CNC). So you can see that the main use of my OSI is related to hardware. For this use, I have not seen a better machine. The I/O capabilities are almost endless. The protoboard connected to the computer buffered bus is invaluable. In the meantime, I have developed a lot of the software and hardware that I needed. Most of it is in public domain. The following is a partial list:

-Display Date and Time on line #26 (out of 65D 3.3 window) under OS, automatically on boot. Need CA-20 RT Clock.

-Printer set up menu in BASIC.

-EPROM 2716/2732 reader/programmer (needs a simple piece of hardware and 2 OSI's PIA 6821).

-Sun Ephemerides for any geographic location at any time.

-Parallel & Serial interconnection with other small computers.

-Creating a Centronics type port with PIAs, with OS65D and OS65U modifications.

-Interface for a Smith Corona EC-1100 Daisywheel typewriter, to use as a printer on the created parallel Centronics port.

-Various Assembler routines, to be called from BASIC (USR funct.): search RAM, move blocks of RAM, load through parallel port, etc.

-Monitoring and talking with a SDK-85 single board microcomputer kit (INTEL).

-An 8748 developing system, with the capability to program the 8748's EPROM (running single step and more).

-Assembler plot subroutine to drive a DAC at CA-22 and use an Oscilloscope as plotter of software synthetised waveforms.

-Real time I/O handling. Tasks to do in disk files. Uses BSR station, digital & analog signals.

Now a few questions, and to see if anybody around may answer them.

1. With my polled keyboard, it is very hard to use the 65U V1.3 EDITOR. No cursor displayed. You have to remember cursor key movements. Some strange displays of the editing line occurs. Any solution?

2. The DQFLS 6502 WP word processor I bought was supposed to run under 65U, but it is impossible to handle disk files. When you choose EXEC from main menu, you simply get out to the BASIC immediate mode, losing your text.

3. I typed in Rick Trethewey's HOOKS into BASIC, almost everything works OK, but the IF-THEN-ELSE enable. I had to remove BEXEC*'s line #230 to get an error free boot. Any idea?

Needless to say, PEEK(65) is doing a good job. All of us enjoy it very much. Keep on line. My best wishes to all of you.

Roberto Frentzel
Caracas (1041) Venezuela

* * * * *

Roberto:

1. (From Rick Trethewey) "I assume we're speaking of the article I wrote about adding the Editor for video systems. I never tried it with V1.3, although it works with both 1.2 and 1.44. In one of the Tech Notes, there was a patch to V1.2 that added true backspacing for video systems. As I recall, you lose the cursor when you backspace. It may well be that this routine is giving you trouble. In my article, I also showed a method for porting the OS-65D V3.3 video routines to 65U. With that implemented, I don't lose the cursor."

2. DQFLS, we were informed, is no longer in business. We use 1.3 daily on a hard disk with no problems. You might consider Rick Trethewey's Edit-Plus, which is an improved version of WP-2/3 for something like $40.00. It also works with his Term-Plus modem software. Both are free to CompuServe OSI-SIG members.

3. (From Rick Trethewey) "Sorry I can't help much with the Hooks BEXEC* without knowing what error you get. Check the output of the assembler to make sure that the last routines reside in the memory locations that are pointed to by the BEXEC*. The REMs should give ample indications as to what's happening where."

Eddie and Rick

★

PRESS RELEASE

Gander Software, Ltd., the folks who brought you The Data System, the Time and Task Planner, and the Financial Planner, has begun testing its most recent software product, which it expects to have to market within 60 days, either direct from Gander, or, perhaps, from Isotron, Inc.

Multi-System OS-65Utils is a complete replacement for OSI's standard 65U utilities that provides the same flexibility to the partitioned hard disk user now available to a more limited extent with the Data System. Now, for example, you can install FDUMP once and use it on any defined system on your hard disk. Ditto with DIR and any other program you could wish was available to each partition. This means an end to having to install subsets of the standard utilities in each partition. Now, as another example, even folks without The Data System can have the remarkable ability to copy files from one system to another using COPYFI's proven 40% speed increase over COPIER.

In addition, this package gives the user a complete systems manager with two levels of optional passwords. Up to 100 "Log on" passwords can be defined for access to SYSDIR, and each of those can define up to 15 systems. On top of that, each of up to 100 defined systems can have its own password.

Gander's MS-Utils uses no tricks of any kind, but rather pulls together in one place the sort of multi-system logic used in The Data System and in prior disk managers, in a very easy to use form. Password protected editors are provided for defining systems and passwords. The systems definition editor is intelligent, knowing the parameters for any hard disk device released by OSI. Once you specify the amount of space you want in a partition, it calculates the appropriate length based on your cylinder size. It even keeps track of total space used so when you go to define a new system, the base address is already there.

* * * * *

Gander has also announced the release of a Level 5 version of the Data System with all the changes necessary to take full advantage of DBI's extra memory. Except for using that extra memory, it is functionally identical to, and looks and acts the same as, the very popular OSI version, which Isotron, Inc. was the first to bundle. Dealers around the country report to Gander that they have begun to write applications software around TDS and its Type 30 file structure. Now that same power is available to the user with an OSI box running Denver boards.

LETTERS

ED:

I have a C1P-MF with OS-65D V3.3. I need to know how to keep BASIC's input from stripping the following characters: at sign (ASCII $40), square brackets($5B and $5D), curly braces ($7B and $7C), and the vertical bar ($7D).

Last year I needed some enhancements for my text editor, namely how to keep the DOS BASIC input from eating leading blanks, double quotes, commas and colons. These were completely answered by J. L. Pottier in the Jan. '84 issue (pg 16). Lately, I've had the need to use all the ASCII printable characters. A keyboard driver from the old Aardvark Journal (Dec '81, pgs 18-20) lets me generate them and BASIC lets me print them, but the input screens out the above mentioned characters. After a few days of disassembling code, I discovered the device drivers were responsible for the square brackets only. Looking into the BASIC input routine is fruitless. It is very convoluted and

bears almost no resemblance to the BASIC in ROM version described in the Oct. '82 issue.

Thanks for your help.

Frank Glandorf
Cincinnati, OH 45220

Frank:

The square brackets are trapped by 65D, not BASIC, but it is certainly possible to POKE those into oblivion. The "@" and curley brackets and vertical bar are mysteries to me. I can look at BASIC and reply more specifically, but I think the solution lies elsewhere. The best solution is to add a USR(X) routine that latches into the keyboard poll and returns the ASCII value into a numeric variable. Following that, a C$ = CHR$(?) should allow you to build a string, trapping a terminator character en route.

PEEK now does have the OS-65D V3.2 Disassembly manual available for $25.95 including postage, if you want to hunt for an answer there.

Rick Trethewey, Sys Operator
OSI SIG on CompuServe

Lucky you! We have two answers for you.

Regarding 65D 3.3: to allow "@", POKE 1390 with what you want to be the new line-delete character. To allow braces, tilde and stuff, POKE 1386,127. To bypass the trapping of square backets by the O.S. try this: POKE 9014,32: POKE 9015, 103: POKE 9016,35.

Dan Schwartz

* * * * *

ED:

I had written some time ago (PEEK Nov '84) and asked for help concerning a problem that I had with not being able to write to disks because the headers and data were being garbaged. I say had, because I found a solution to the problem, and now everything works fine. The problem seems to have been related to the termination of the write data line on the D&N controller. During pulse width set up, the write data line is jumpered to the read data and SEP clock inputs so that the read pulse widths can be set. What I found was that if I left the jumper in between write data and SEP DATA, I could write to a disk! As a permanent fix, I installed two resistors on the write data line so that it is terminated just like the SEP data line. (Originally writed data has a pull up resistor to +5 volts, and SEP DATA has resistors to both +5 volts and ground.) Anyway, this should be one for your "believe-it-or-not" file!

Second, I have a question regarding another operating system that I have run across. It's called "ASTRO GRAPHICS" and was written by a fellow named Corey Ostman, and runs on a C4P. The DOS operates similarly to OS65D, but patches have been made to BASIC to allow for some pretty impressive graphics. One program is a lunar lander that is equal to the old arcade lander, complete with various landing pads and point multiples for difficulty. Plus, the BEXEC* has been modified for blinking cursor, and a HOME command that clears the screen and homes the cursor to the upper left of the CRT. BASIC was patched to allow for a very good X-Y plot function that will draw graphics on a pixel by pixel level.

I am very interested in finding out if this DOS has been, or could be, adapted for my C1P. Any assistance will be greatly appreciated.

C. J. Hipsher
Virginia Beach, VA 23456

CJ:

We sure thank you for following up on the disk problem. We too were, frankly, swinging in the breeze. As to Astro, another blank. Here's hoping a reader can come to the rescue.

Eddie

* * * * *

ED:

I have recently been having an intermittent problem with my C1P-MF occasionally blitzing disks on cold boot. The problem may have existed before now, but I had so many problems with my old MPI drive that I replaced it with two Shugart SA-455 drives. Now, this problem is the only one remaining, and it has no relation as to which drive is the default A drive.

I called Bill Thompson at Isotron to see if he knew of any common problem like this and he thought it had been discovered and reported in one of the old issues of PEEK(65). I have only been getting PEEK(65) for a couple years, and don't recall seeing it in any of the issues that I have.

Do you, or any of the staff, recognize the problem? Bill's recollection is that there is some kind of "race" condition in the floppy interface circuitry that would cause such problems in a relatively small percentage of the C1P-MF machines.

Also, I had a conversation with Rick Trethewey and some other hackers on CompuServe and asked them about a spreadsheet program for the C1P. The consensus was that there is a Canadian user group that has one, but they knew nothing about it. Do you know of what they are eluding to, or who to contact for more information?

Thank you for any information you can give me on these problems.

Glen Davis
Endicott, NY 13760

Glenn:

Although there are a multitude of references in PEEK indexes to explanations of Disk Boot routines, bugs and fixes, it sounds to us like your problem may be answered by the article in the October 1980 issue, pg. 4, or July 1983, pg. 2, gives an excellent understanding of the boot routine.

Regarding the spreadsheet program, you might try: TOSIE, P.O. Box 29, Streetville, Ontario, Canada L5M-2B7. In any case, let us know how you fare so we can share it with other PEEKers.

Eddie

* * * * *

ED:

A letter in a recent PEEK(65) mentioned using a 65C02. I have been using one for over a year now. Installation is simple. Pull the old 6502 out and plug in the new 65C02. There are two advantages: power consumption drops by about 100ma, and the 65C02 is more tolerant of poor timing. The 65C02 can be bought in different speed versions. A 4 MHz version would probably require 150ns memory. I may have a go at plugging one into my "Superboard" and seeing what happens at 4 MHz (make sure crystal clock is buffered). I estimate Disk BASIC would run as fast as an 80186 runs in the new MS DOS micros. ROM BASIC would, of course, be

50% faster! Does this tell us that Intel stuff is just plain slow?

I am sure many readers leapt to the first article in the February issue of PEEK(65).

I enclose a short program 'DIRDEL' which readers may find useful. I would also advise using DOS 3.2 with the program. DOS 3.3 (and its seq. file read) causes too many problems because of the need to avoid CTRL key codes.

Leo Jankowski
Timaru, New Zealand

* * * * *

```
10 PRINT !(28):   REM Directory selective delete - by LZJ
20 :
40 PRINT TAB( 10)"DELETE DIRECTORY UTILITY FOR 8"CHR$(34)" DISKS"
50 FOR X=1 TO 64: PRINT "=";: NEXT : PRINT
60 PRINT "P is the # of file names --> NOT <-- to be deleted."
70 PRINT : PRINT "Let P=19 to save first 33 Tracks.": PRINT : PRINT
80 INPUT "# of names (from Trk 0) --> NOT <-- to be deleted. P= ";P
85 IF P>31 THEN PRINT "Can't be done": PRINT : PRINT : GOTO 80
90 :
100 M=11897+P*8       : REM TRACK 08,1. Omit first P names.
110 E=12145           : REM $2F78-7=12145
120 DISK !"CA 2E79=08,1": GOSUB 190: DISK !"SA 08,1=2E79/1"
130 :
140 M=11897
150 DISK !"CA 2E79=08,2": GOSUB 190: DISK !"SA 08,2=2E79/1"
160 :
170 PRINT : PRINT "<< Done >>": END
180 :
190 FOR C=M TO E STEP 8: FOR X=C TO C+5: POKE X,35: NEXT X
200 POKE C+6,0: POKE C+7,0: NEXT C: RETURN
```

* *

ED:

Greetings from the Great White North! As a new subscriber to PEEK(65), I would like to take the opportunity to commend you on the content of your publication. In reading through some of the back issues just received, I know I missed a lot by not being on your mailing list over the past few years while working with my OSI C4P-MF, (OS-65D V3.2).

In addition to the above, I am very interested in obtaining a copy of the "Tiny Compiler" or "F-Basic" and "Busicalc" on 5.25" disk perhaps in some exchange with one of your readers if it is not still available from a commercial source. Do you have any contacts in this regard?

Lloyd G. Bunbury
Ontario, CANADA K2H 7V2

Lloyd:

Re F-BASIC: Rick Trethewey says it rings a bell with him as being a product of Pegasus Software in Hawaii, (they may no longer be in business.) No doubt someone still owns the copyright, but who?

Re "Tiny Compiler": available FREE on OSI SIG on CompuServe or, for a fee from David Pitts, 16011 Stonehaven Drive, Houston, TX 77059, known address as of Dec. '83. Also, BUSI-CALC used to be available from Micro Software International, 3300 South Madelyn, Sioux Falls, South Dakota 57106, not sure if they still support OSI, but worth a try.

Eddie

ED:

Do you folks know of a program that can produce different type fonts and sizes of print for use with a C4P and EPSON MX80 with Graftrax+?

Ray Peterson
Edmore, MI 48829

Ray:

That's not much to go on! Your 4P can send the necessary instructions to the printer to make it do anything in the manual, but you cannot make up new fonts.

In BASIC just PRINT #1, CHR$(X), X being the number of the character from the manual. In WP-2/3, type ESC 01B and the character. You could make a new font and print it in the graphics mode, but SLOW. Maybe other readers will have another answer.

If we are off target, please give us more details.

Eddie

AD$

WANTED: C3-B or C3-C in good working condition. Also, tape back-up. Call Richard (201) 666-3250 (NJ).

*** * * OS-65D V3.2 * * ***
DISASSEMBLY MANUAL

Published by Software Consultants, now available through PEEK(65) for $25.95 including postage. Overseas add extra postage (weight 16oz). Make check or money order (in U.S. funds, drawn on a U.S. bank) payable to PEEK(65), P.O. Box 347, Owings Mills, MD 21117.

*** * * * * GIVE AWAY * * * * ***
Multi-Strike Printer Ribbons

What do you currently pay for a multi-strike ribbon cartridge? About $4.00 each in lots of 6?

We have found a solution that may cause you never to use a fabric ribbon again. 1) Did you know that most all multi-strike ribbon cartridges use the same ribbon bobbin? It is just pressed on a different size hub and put in your cartridge type. 2) We have found a source of recently outdated (yes, many are dated) Diablo Hi-Type I cartridges. We took the oldest one we could find, put it in our NEC cartridge and printed this ad. Now, honestly, do you see any difference? We can't either. So we are offering those of you who use Hi-Type I, or are willing to pry open whatever cartridge you are using and replace the bobbin, a deal you can't refuse.

Buy one box of 6 cartridges for $8.00 and we will give you a second box FREE. That's 66.66 cents a piece or 83% off. At that rate, how can you lose? Add $3.00 for postage and handling. Make check or money order (in U.S. funds, drawn on a U.S. bank) payable to PEEK(65). P.O. Box 347, Owings Mills, Md. 21117. Order NOW, supply limited!

* * * * *

MUST SELL. Still in original wrappings, KEYWORD CP/M Word Processor, CP/M v 2.25. Cost was $400.00 each. Will sacrifice $250.00 each, or $400.00 for set. Reply PEEK, Box K, c/o PEEK(65), P.O. Box 347, Owings Mills, MD 21117.

* * * * *

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

* * * * *

Good prices on collection of OSI equipment and accessories. Send SASE for complete list. Ricky Peterson, 206 Pine Valley, Warner Robins, GA 31093.

* * * * *

FOR SALE: C3S1 with almost new FDD 100-8 8" drives, $1300 or offer. 2 new FDD 100-8 drives with box and power supply $425 or offer. Tom McGourin (219) 429-4160 days, (219) 489-6001 eves/weekends.

PEEK (65)
The Unofficial OSI Users Journal

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

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

DELIVER TO:

GOODIES for OSI Users!