PEEK [65]

The Unofficial OSI Journal

September, 1986
Volume 7 Number 9

Inside This Issue:

Column One

It's hard to believe this issue is so near and yet so far to being on schedule. Still, the response to the summer issue was gratifying and I hope it has renewed everyone's confidence in PEEK.

As I write this, Apple Computer has just announced its new Apple IIgs. For those of you who haven't read about it yet, the IIgs is a 65C816-based system that is designed to be compatible with 90% of existing Apple II software, but adds the power of a 16-bit CPU, 256K of RAM, a detatchable keyboard the now-obligatory mouse, a high-resolution graphics chip, and the 15-voice Ensoniq sound chip. I mention the IIgs not to debate the virtues or vices of the machine, but instead to highlight the expanding interest in the new microprocessor it employs and that the 8-bit OSI community is beginning to discover.

Again, very soon every OSI owner will have the opportunity to upgrade the brains of his system to a more powerful processor and I intend to keep PEEK(65) intimately involved with this evolution. While the new 65802 and 65816 will run our ancient 6502 software, there isn't a lot of software available to take advantage of the new chips' abilities. That means the OSI community will again have to band together to provide its own solutions.

Whether you call them hobbiests, personal computer users, or video users, the C1/C4/C8 community simply isn't large enough or enthusiastic enough to provide a vendor a reasonable return on new software. Unless the software will also appeal to the serial system market, you aren't going to see a new commercial product. It seems to me, therefore, that we are back to a situation where there will be no software unless there are enough hardware purchases to warrant it, and there won't be enough hardware purchases if there is no software. Clearly there has to be something to get the ball rolling.

To help things out in the video market, we are all once again indebted to Paul Chidley of the Toronto-based user group TOSIE for developing a new CPU board that uses the 65816. For the high-end users, an imminent announcement will bring you along as well, although I expect the divergence in 65U/65D software between the two communities to increase, rather than diminish.

For my part, I have committed to produce a version of my ASM-Plus assembler for the 65816. But to really make this chip attractive, there must be more than just an assembler. Therefore, I propose that the new 65D project I have been touting be shifted to specifically designed as a 16-bit operating system. Stand up and be heard now, people. Take the time to write in with your comments, suggestions, and opinions or this project will languish and die.

Please check out the mailing label on this issue. As noted in the Summer issue, all subscribers current through June of 1986 had 2 extra months added to their subscriptions. Some people's subscription will have expired despite this extension, for many others, this is their last issue. In any event, if your subscription is about to expire, please don't wait for the reminder postcard to renew. It will be some time before I can do another mailing and this may cause you to miss an issue if you neglect to renew now. So take a moment and see when your subscription expires. Your support is very much appreciated.

Lots of good stuff in this issue. Among other things, Scott Larson shows us how to add 8K of RAM to the C1P, I have included the mailing label program for my DMS65D software presented in the Summer issue, as well as a product description of the new word-processor from SofTouch, Dave Livesay presents his new disk interface that allows you to connect the newer high-density mini-floppies to your system (which he is offering fully assembled for $50), and Richard Reed gives us SAM, the Self-Aware Microcomputer. Enjoy!

Rick

How to add 5.25" 40 or 80 Track Double Sided Drives to Your OSI

by David Livesay
Ave de la Resistance 6
B-4920 Embourg, Belgium

This article will cover several subjects dealing with the problem of 5.25" drives for the OSI. Covered will be the conversion of a new MPI drive to replace your old drive, how to build a data separator and a motor control circuit to use with any industry standard disk drive and last we will cover how to use double sided 80 track drives. By implimenting the suggestions here you can increase your disk drive capacity to either 328K or 656K for about $300 or even less if you only impliment part of this.

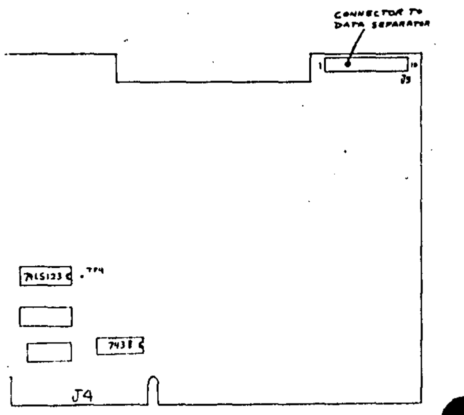
Those who are faced with the problem of adding more drive capacity or replacing existing drives have several choices available. These choices include purchasing a new MPI drive and adapting your old data separator to it, building a new data separator and using standard 40 track drives either single or double sided, or using one of the 80 track double sided drives. Some of the information presented here has been published before but I think it worthwhile to place all of this information into one article.

NEW MPI DRIVES

First we will discuss the adaptation of a new MPI drive to replace an existing OSI MPI drive. As those of you who have 5.25" disk systems know OSI used a single sided 40 track MPI drive with built in data separator. During the past few months I have seen double sided MPI drives advertised for about $89. To make the required modifications you will need to remove the main circuit board from the new drive. Make note of all of the connector positions and if need be mark them before removal. Now remove the connectors and the screws holding the board in place. For either a double or single sided drive there is only one modification required. The line which goes to pin 30 of the drive interface connector (J4) is connected to pin 3 of the 7438 nand gate shown in Figure 1. This is normally the raw data output. Pin 1 of this nand gate is connected to pins 5 and 9 of the 74LS123 (see fig. 1) and must be cut. Make sure that the connection between pin 1 of the nand gate and the drive separator connector remains intact. Now you will need to remove the small data separator circuit board, which is located at the front right hand side of the main circuit board, from your old drive. Install this on the new drive and replace the board and connectors. You now have a new single or double sided MPI drive. For double sided drives the circuitry on the OSI controller and paddle board will also need to be modified.

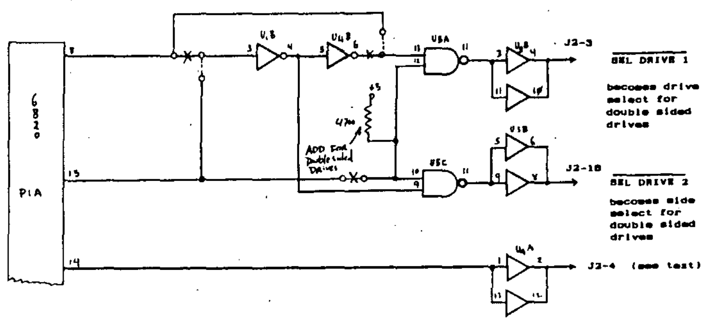
MODIFICATION OF THE OSI CONTROLLER FOR DOUBLE SIDED DRIVES

The OSI disk drive selection circuitry as used on the 505 board is shown in fig. 2. Other OSI floppy controllers use the same basic circuit but you will have to trace out the signals starting with the PIA or get a copy of the schematic for your board. For double sided operation the traces marked with an "X" must be cut and the jumpers shown in dashed lines must be added. The drive selection logic for single and double sided drives is shown in table 1. As shown in table 1 selecting drive A or C will select side



MPI FLOPPY DRIVE CIRCUIT BOARD

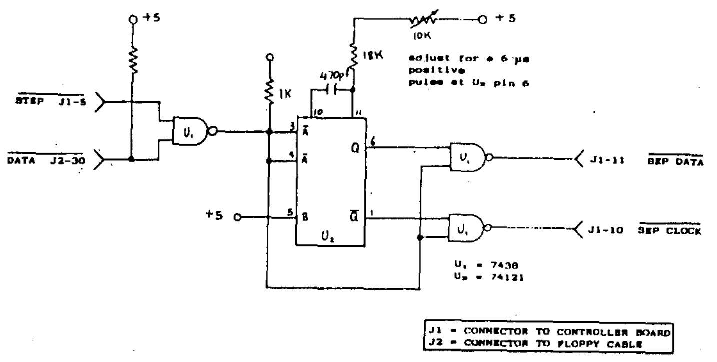
FIG. 1


6820
PIA
U1B
U4B
U5A
U4B
J2-3
SEL DRIVE 1
becomes drive select for double sided drives
+5
47K
ADD For Double sided Drives
U5C
U1B
J2-10
SEL DRIVE 2
becomes side select for double sided drives
U4A
J2-4 (see text)

X=CUT, - - - = ADD (FOR DOUBLE SIDED DRIVES)

OSI DRIVE SELECT SCHEMATIC (505 BOARD)

FIG. 2


+5
STEP J1-5
DATA J2-30
U1
1K
470pf
18K
10K
+ 5
adjust for a 6 µs positive pulse at U2 pin 6
A
A
B
Q
Q
U2
+5
U1
J1-11 SEP DATA
U1
J1-10 SEP CLOCK
U1 = 7438
U2 = 74121
J1 = CONNECTOR TO CONTROLLER BOARD
J2 = CONNECTOR TO FLOPPY CABLE

DATA SEPARATOR
FIG. 3

Copyright 1986 PEEK[65] All rights reserved
published monthly
Editor: Richard L. Trethewey

Subscription Rates	Air	Surface
US		$22
Canada & Mexico (1st class)		$30
Europe	$42	$40
Other Foreign	$47	$40

All subscriptions are for one year and are payable in advance in US dollars. For back issues, subscriptions, or other information, write to:
PEEK[65]
P.O. Box 586
Pacifica, CA 94044 415-359-5708

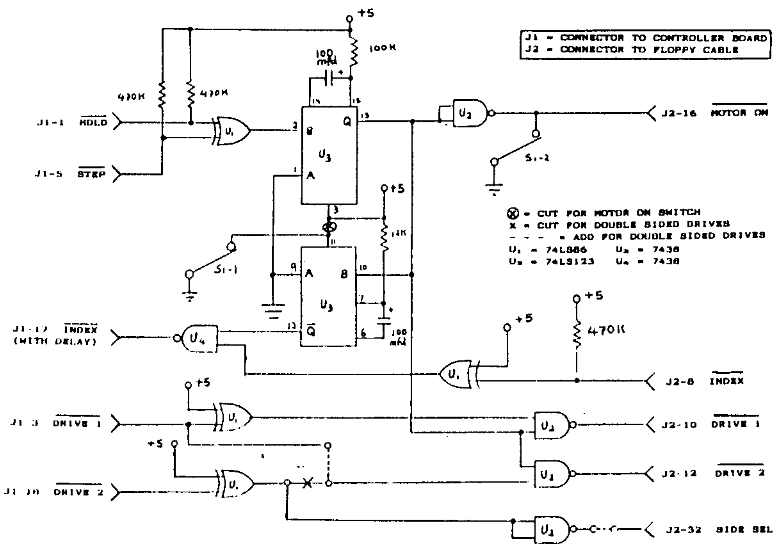
Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by this magazine or the publisher.

one or two of the first drive and selecting drive B or D will select side one or two of the second drive. The only problem now is that the drive select signal for drive two must be the inverse of the signal for drive select one. We have two choices. The first is to add an inverter on the paddle board between pin 3 of the controller connector and pin 12 of the connector going to the drive. You can use a 7438 nand gate for this. Pin 18 of the OSI controller which was conected to pin 12 of the disk drive needs to be connected to pin 32 of the disk drive.

The second choice is to use the fault reset driver in the OSI controller circuit which is not used. This driver is shown in fig. 2. The input should be connected to pin 13 of the nand gate (U5A) which drives the driver for the select drive 1 line. The line coming from pin 14 of the PIA will also need to be cut. Now pin 4 of the OSI controller connector will need to be connected to pin 12 of the disk drive cable connector and the connection between pin 12 of the disk drive connector and pin 18 of the controller connector removed. Pin 18 of the controller connector now needs to be connected to pin 32 of the drive connector. Table 2 shows the final connections. If you build the motor control circuit described later you will only need to make the changes for conversion to double sided drives without making the changes needed to add the inverter for drive select 2.

USING INDUSTRY STANDARD 5.25" INTERFACE DRIVES

Now what options do we have? The first thing is that we will use an industry standard drive. Almost all of the 5.25" drives on the market today use the same interface. The only differences you will find are that some drives have more user installed options available than others. Table 3 shows the pin asignments for the 5.25" interface. The OSI MPI drive had the industry standard interface with two changes. Instead of using pin 30 as the read data line it became the separate clock line and pin 34 became the separate data line. Some history is perhaps in order. The first disk drive that OSI offered in 1976 was the GSI model 105. The 470 board was designed to interface to this drive. This is the reason that some of the lines of the drive controller such as fault reset are no longer used. The fact that this drive and the latter Siemens and Shugart 8" drives also had data separators meant that OSI never did develop an interface board that included the data separator. As the 8" drives all used AC motors they ran continuously and when OSI came out with the 5.25" drive systems they never bothered to provide any control for the motor and had to use a drive with a built in data separator. It may also be that they decided to let the motor turn constantly to increase the access speed. So much for history but thats how we became stuck with today's problems with the disk drives.



MOTOR CONTROL

FIG. 4

DATA SEPARATOR

If we're going to use the industry standard interface we will need to build a data separator. There have been several articles published in PEEK[65] on building data separators all of which should work. In fig. 3 is another data separator circuit which is the one used by Siemens on the 8" drives. The timing values have been adjusted to conform to the 5.25" timing requirements. This separator uses fewer components than most of the others that I have seen and works fine. You can adjust the timing by connecting pin 9 of the OSI controller to pin 30 of the data separator. Adjust the potentiometer for a 6 microsecond low pulse at pin 1 of the 74121.

DRIVE MOTOR CONTROL

I would suggest that you build a disk motor on/off control circuit unless you want your drives to turn continuously. The motor control circuit is shown in fig. 4. The motor control has been set-up so that anytime a head load or step pulse is detected the one-shot will trigger for about 5 seconds. Anytime a new step or head load pulse is detected the one shot will retrigger for the 5 second period. In this way the motor will stay on if the drive accesses several tracks in succession. The second one-shot is used to inhibit the output of the index pulse until the motor has had time to come up to speed.

In general, almost all of the disk drives available will start in .5 seconds or less. We therefore need to inhibit the index pulses for about .5 seconds. If you wish, you can experiment with less delay to provide for quicker access. In some cases you can decrease the delay to around .25 second. If you wish to sometimes have your motor on continuously for faster access then a switch can be added as shown. This switch can be located either in the computer or on the disk drive case.

THE OTHER DRIVE SIGNALS

The connections between the OSI and the disk drive are shown in table 2. Note that if you have a drive with the ready signal then you can connect this line to pins 22 and 24 of the OSI controller, otherwise these pins should be grounded.

BUILDING THE DATA SEPERATOR AND MOTOR CONTROL

If you wish to build your own data separator and motor control you can build one using a small prototype board about 4" X 3". The biggest problem is that the Molex connectors for the connection to the OSI use a .156" spacing between pins. You can drill new holes in your prototype board to match. You will find that you can use some of the existing holes if you enlarge them. If you use the standard connectors as used on the computer boards you will need to straighten the leads so that the connector is at right angles to the board. Molex also builds a version of the connector with straight leads.

I would recommend that you epoxy the connectors to the proto board. I used a 34 pin right angle ribbon cable header for the connector to disk drive. This alows you to easily replace the cable. The 470 pf capacitor in the data separator circuit shoud be a stable type such as a silver mica unit.

HEAD LOAD CONTROL

In general there are two types of head load control for 5.25" drives. Some drives such as the MPI use a head load solenoid and have options for head load with motor on or head load with drive select. Others such as Tandon and Panasonic load the head when the drive door is closed and don't have a head load solenoid. Some of the newer drives have other options such as using pin 2 which is a spare or pin 4 the In Use line to load the head. All of the newer drives (if they have head load solenoids) also have the load head constant, with motor on and with drive select options for loading the head. For those who don't wish to build their own separator and motor control I will have one available assembled and tested without disk drive cable but with a header for the drive cable. This will sell for under $40 plus about $5 for airmail shipping.

TABLE 1

DRIVE SELECTION LOGIC FOR OSI DISK DRIVES

FOR SINGLE SIDED DRIVES

PIA PIN #		CONTROLLER PIN #		DRIVE SELECTED
PIN 8	PIN 15	J2-3	J2-18	
X	LOW	HIGH	HIGH	NONE
HIGH	HIGH	LOW	LOW	DRIVE A
LOW	HIGH	HIGH	LOW	DRIVE B

X = DON'T CARE

FOR DOUBLE SIDED DRIVES

PIA PIN #		CONTROLLER PIN #		DRIVE SELECTED
PIN 8	PIN 15	J2-3	J2-18	
HIGH	HIGH	LOW	LOW	DRIVE A
LOW	HIGH	HIGH	LOW	DRIVE B
HIGH	LOW	LOW	HIGH	DRIVE C
LOW	LOW	HIGH	HIGH	DRIVE D

-NOTE- FOR DOUBLE SIDED DRIVES

DRIVE A = DRIVE 1 SIDE 1
DRIVE B = DRIVE 2 SIDE 1
DRIVE C = DRIVE 1 SIDE 2
DRIVE D = DRIVE 2 SIDE 2

DRIVE SELECTION

Now that we have the problem of the data separator taken care of what do we do about the drive. For someone looking for a bargin in disk drives you should be able to find some single sided Tandon 100 drives as used in the early IBM PCs. Most of these were removed and replaced with double sided drives a few years ago. If you look in the ads in Byte magazine you will find many double sided half height 5.25" drives advertised for under $90. Suitable drives are identified as 40 track double sided. Some known drives which conform to this are Teac FD-55B, Shugart SA455, Mitsubishi 4851 and Qume 142. Panisonic and others also build drives which can be used. If you wish to use two of these double sided drives you will need to modify the circuit of your disk controller board as described earlier. If you built the motor control circuit then it must be configured as shown in fig. 4 for double sided operation. The first drive must be set-up as drive one and the second drive as drive two. The manual that you should have obtained when you got the drive will explain how to do this.

USING 80 TRACK DOUBLE SIDED DRIVES

Now we come to the real heart of the article. We have another choice for the type of drive to use. If you would like to increase your drive capacity at very little cost consider using 80 track drives instead of the 40 track units. You can use drives built by the above mentioned manufacturers which are usually identified as 80 track or 96 tpi (tracks per inch) drives. These usually cost about $10 more than the 40 track drives. The only change that we need to make is to modify 65D and some of the utility programs so that they know we have 80 tracks available. The 65D memory locations to be changed are shown in table 4. We will get to the utility programs later.

How do we set all of this up? First of all you must have one 40 track drive available. This should be set up as drive one. If you have an OSI MPI drive you will need to modify it to provide the raw data to pin 30 of the drive. If you look at the drive controller board you will find that pin 30 goes to a 7438 nand gate. In the OSI version the trace from pin 1 of the 7438 which goes to pin 5 and 9 of a 74LS123 has been cut. We need to remove the small data separator from the disk drive and reconnect the cut trace.

Another easier way is to remove the data separator and jumper pin 3 of the data separator connectorto pin 5. With this change made the drive will now output the raw data at pin 30. At this point you should connect the drive to the new data separator and computer and attempt to boot. If everything was done correctly it should boot. If all goes well you can connect the 80 track disk drive. Set it up as drive 2 as instructed in the manual. Reboot and try to initialize the 80 track drive. The first side of the 80 track drive will be device B. Enter DISK!"SE B" <RETURN> then POKE 9930,128 ,POKE 10089,121 and POKE 1015,121 . Next enter DISK!"INIT". If all goes well the computer will initialize 80 tracks.

If you were able to initialize a disk in the 80 track drive you can now proceed to make a copy of your operating system disk with your utilities on it. Now remove the 40 track drive and connect the first 80 track drive in its place. You should now find that the disk that you just made will boot.

At this point, if you've decided to convert to two 80 track drives, you should use the combination of one 40 track drive with one 80 track drive to copy all of your programs to disks in the 80 track drive. You can then make the permenant change to two 80 track drives. Before you can do this you will need to change the CREATE utility. So that more than 40 tracks can be used

TABLE 2

OSI CONTROLLER - DISK DRIVE CONNECTIONS AND FUNCTIONS
(CONFIGURED FOR DOUBLE SIDED DRIVES)

OSI CONTROLLER PIN*	DRIVE PIN *	FUNCTION
1	N.C.	HEAD LOAD
2	N.C.	LOW CURRENT
3 (THROUGH CONTROLER)	10 & 12	DRIVE SELECT 1 & 2
4	N.C.	FAULT RESET
5	20	STEP
6	18	DIRECTION
7	N.C.	ERASE ENABLE
8	24	WRITE GATE
9	22	WRITE DATA
10 (TO CONTROLLER)	N.C.	SEPARATE CLOCK
11 (TO CONTROLLER)	N.C.	SEPARATE DATA
12 & 13	TO ALL ODD PINS	GND
14	N.C.	+5V
15	N.C.	-9V (NOT USED)
16	N.C.	N.C.
17 (THROUGH CONTROLLER)	8	INDEX
18 (THROUGH CONTROLLER)	32	SIDE SELECT
19	28	WRITE PROTECT
20 GROUND THIS PIN *	N.C. *	READY DRIVE 2
21	N.C.	SECTOR (NOT USED)
22	N.C.	FAULT (NOT USED)
23	26	TRACK 00
24 GROUND THIS PIN *	N.C. *	READY DRIVE 1
N.C.	2	SPARE
N.C.	4	IN USE
N.C.	6	DRIVE SEL 4
N.C.	14	DRIVE SEL 3
N.C. (TO CONTROLLER)	16	MOTOR ON
N.C. (TO CONTROLLER)	30	READ DATA
N.C.	34	SPARE OR READY *

-NOTE- (THROUGH CONTROLLER) INDICATES THAT THE SIGNAL IS MODIFIED BY THE DATA SEPARATOR/MOTOR CONTROLLER.

(TO CONTROLLER) INDICATES THAT THE SIGNAL IS USED BY THE DATA SEPARATOR/MOTOR CONTROL OR GENERATED BY THE CONTROLLER.

* SOME OF THE NEWER DRIVES HAVE READY LINES (PIN 34) WHICH MAY BE CONNECTED TO OSI CONTROLLER PINS 20 AND 24.

change the following lines to read as follows:

```
   490 DIM ALX(79)
   20090 IF T0<13 OR T0>79 THEN
20080
   20110 IF NT<1 OR NT+T0>80 THEN
20100
```

* NOTE * If you wish to use tracks lower than number 13 on a data only disk you can change line 20090 to read as follows:

```
   20090 IF T0=1 OR T0=12 OR T0>79
THEN 20080
```

You should also make the three 65D memory changes permenent.

CHANGING 65D FOR 80 TRACK DRIVES

There are three memory locations in 65D which need to be changed. There are two ways that we can do this. The first one is to poke the correct values into memory from BEXEC*. The second way is to make the changes and save them back to disk. To do this exit BASIC to 65D and load the track zero read/write utility. Follow the instructions to read track zero into memory at $6200. Load the extended monitor and change the three memory locations list . in table 4. Remember you will use an offset of $4000 when making the changes (i.e. use $66CA instead of $26CA). Reload the track zero utility and follow the instructions to write the data at $6200 back to track zero. Remember that we will read and write 8 sectors each time. At this point you should have a disk that will boot and be able to use all 80 tracks. It should be mentioned that the 80 track drives will step at a rate of 3 ms so you can modify the step rate in 65D if you wish.

ARE THERE ANY DRAWBACKS TO USING 80 TRACK DRIVES ?

At this point we should consider the problems which could exist with this system. The first problem is that the disks you create in 80 track format will not be readable by 40 track drives. If you never exchange programs with others then this should not be a problem. If you sometimes have a need for exchanging programs then you should keep one 40 track disk that you can substitute for one of the 80 track drives when you need to send someone a 40 track disk. If you only need to read 40 track drives then you can write a program in BASIC or machine language to read a 40 track disk on a 80 track drive. Another option would be to write a disk copy program in BASIC or machine language to read a 40 track disk on an 80 track drive. To do this the drive must double step to move one track.

WHAT WILL THIS COST?

Two 40 track double sided half height drives will cost a maximum of $180. If you use your existing case for a single floppy system you can install two half height drives in it. You may or may not be able to get away with using your existing power supply. Since both drive motors can be on at the same time the power supply must be able to furnish the power for both drives at the same time.You can also replace the power supply with a small switching supply.

If you wish to purchase a new case it will cost about $50 with power supply. If you build the data separator yourself it will cost you a maximum of $20 plus another $15 for the cable to the disk drives. If you purchase the data separator and motor control it will be about $40 plus the cable. So the price range for this modifiction will be between $215 and $290. If you use the 80 track drives then you will spend another $20 for two drives. This is not too bad an

TABLE 3

INDUSTRY STANDARD 5.25" INTERFACE PIN DESIGNATION

PIN#	SIGNAL TYPE	FUNCTION
2	INPUT	SPARE
4	INPUT	IN USE
6	INPUT	DRIVE SELECT 4
8	OUTPUT	INDEX
10	INPUT	DRIVE SELECT 1
12	INPUT	DRIVE SELECT 2
14	INPUT	DRIVE SELECT 3
16	INPUT	MOTOR ON
18	INPUT	DIRECTION SELECT
20	INPUT	STEP
22	INPUT	WRITE DATA
24	INPUT	WRITE GATE
26	OUTPUT	TRACK 00
28	OUTPUT	WRITE PROTECT
30	OUTPUT	READ DATA
32	INPUT	SIDE SELECT
34		SPARE OR READY ON NEWER DRIVES

-NOTE- ALL ODD PINS ARE GROUND

TABLE 4

THE FOLLOWING MEMORY LOCATIONS MUST BE CHANGED IN DOS TO USE THE EIGHTY TRACK DRIVES. THE NUMBERS IN ( ) ARE THE DECIMAL VALUES FOR POKING.

MEMORY LOCATION	EXISTING DATA	CHANGE TO
$26CA (9930)	$40 (64)	$80 (128)
$2769 (10089)	$39 (57)	$79 (121)
$2779 (10110)	$39 (57)	$79 (121)

investment to obtain 656K of disk drive storage to replace the 82K that you have with one single sided MPI drive.

WHAT ELSE COULD YOU DO?

What I haven't mentioned is that you could select a 3.5" drive. The 3.5" drives use the same interface as the 5.25" drives and the controler can't tell the difference. Another choice for those who have both 5.25" and 8" systems is to use the newer high density 5.25" drives as used in the IBM AT.

You must use one of the two speed versions. In the low speed mode these drives can be used to replace the standard 80 track 5.25" drives. In the high speed mode they can replace 8" drives. Using these drives to replace 8" drives will be the subject of another article.

WRITE FOR PEEK!

Mailing Label Utility for DMS-65D

by Richard L. Trethewey

As promised, this month I am presenting a follow-up to the random file system for OS-65D V3.3 that I wrote about in the Summer issue. You'll recall that one of the primary uses for database managers is for mailing lists. The program presented here incorporates most of the editing functions of its predecessor, but goes on to add a mailing label printer and a simple report generator.

As with any program using data files under OS-65D, it is vital that you run the program "CHANGE" before you enter the program into the computer so that BASIC will reserve the appropriate amount of space in front of the workspace for the disk buffer(s). MAILER requires only one disk buffer, even though it does include token support for a second data file to be opened simultaneously.

The mailing label printer is written to allow you to print any number of

```
10 REM- Mailing List Manager for DMS-65D
20 GOTO 1000
30 :
40 REM- Construct Device 6 Current Track String
50 c6 = FNa(PEEK(9004)): t6$ = RIGHT$(STR$(c6+kh), k2): RETURN
60 :
100 REM- Get Record #r6 for Device #6
110 i6 = bodf + ((r6-k1)*rl): wt = INT(i6/ts) + st(k6)
120 GOSUB 50: IF c6 = wt THEN 160
130 d6 = PEEK(9005): IF d6 = k0 THEN 150
140 DISK!"sa " + t6$ + ",1=3a7e/" + pg$: POKE 9005,k0
150 DISK!"ca 3a7e=" + t6$ + ",1": POKE 9004, FNb(c6)
160 i = i6 - ((wt-c6)*ts) + bs(k6): ih = INT(i/pg): il= i - ih*pg
170 POKE ip(k6),il: POKE ip(k6)+k1,ih
175 POKE op(k6),il: POKE op(k6)+k1,ih
180 RETURN
190 :
200 REM- Set Device 6 I/O Pointers to Index(6)
210 i = i6+bs(k6) - (FNa(PEEK(9004)) - st(k6))*ts
215 ih = INT(i/pg): il = i - ih*pg
220 POKE ip(k6),il: POKE ip(k6)+k1,ih
225 POKE op(k6),il: POKE op(k6)+k1,ih: RETURN
270 :
300 REM- Fetch Record from Device #6
310 GOSUB 100:FOR k = k1 TO nf: i6 = bodf + ((r6-k1)*r  +i6(k)
330 GOSUB 200: INPUT#k6,a$(k): NEXT k: RETURN
340 :
400 REM- Put Record Out to Device #6
410 GOSUB 100
420 FOR k = k1 TO nf: i6 = bodf + ((r6-k1)*rl) + i6(k) :GOSUB 200
430 PRINT#k6,a$(k): NEXT k: RETURN
440 :
700 REM- Display Record Contents
710 PRINT#dv," #"; TAB(k4); "Field Name"; TAB(32); "Contents"
720 PRINT: FOR k = k1 TO nf
730 PRINT#dv,k; TAB(k4); n$(k); TAB(32);a$(k): NEXT k: PRINT#dv
740 RETURN
750 :
800 REM- Main Menu
810 :
820 PRINT !(28); &(k9,k0); "DMS-65D Data File Manager"
830 PRINT &(k5,k2); "(1) Directory"
840 PRINT &(k5,k3); "(2) Print Mailing Labels"
841 PRINT &(k5,k4); "(3) Report Writer
850 PRINT &(k5,k5); "(4) Edit a DMS-65D Master File"
900 PRINT &(k9,k7); "Your Choice ";: INPUT y$: k = VAL(y$): TRAP 0
910 PRINT !(28);: IF k = k0 THEN END
920 IF k<k1 OR k>k3 OR k<>INT(k) THEN 820
930 ON k GOTO 2000,7000,20000,4000
998 :
1000 k0=0: k1=1: k2=2: k3=3: k4=4: k5=5: k6=6: k7=7: k8=8: k9=9: kt=10
1010 aa=ASC("A"): az=ASC("Z"): a0=ASC("0"): a9=ASC("9"): kh=100
1020 pg=256: hex$="0123456789abcdef": sx=16: tt=32: di=11897
1030 POKE 2972,13: POKE 2976,13: REM- Disable Comma & Colon
1040 DEF FNa(x) = kt*INT(x/sx) + x - INT(x/sx)*sx
1050 DEF FNb(x) = sx*INT(x/kt) + x - INT(x/kt)*kt
1060 ht = FNa(PEEK(11687)): dt = FNa(PEEK(11716)): e=35
1070 DIM index(k7), bs(k7), be(k7), st(k7), et(k7), cu(k7), df(k7)
1080 DIM ip(k7), op(k7), f$(ht), ut(ht)
1090 bs(k6) = PEEK(8998) + PEEK(8999)*pg :REM- Buffer Start Address
1100 bs(k7) = PEEK(9006) + PEEK(9007)*pg
1110 be(k6) = PEEK(9000) + PEEK(9001)*pg: REM- Buffer End Address
1120 be(k7) = PEEK(9008) + PEEK(9009)*pg
1130 ts = (be(k6) - bs(k6)): pg$ = MID$(hex$, ts/pg+k1, k1)
1140 dt$ = RIGHT$(STR$(dt+kh), k2) + ","
1150 ip(k6) = 9132: op(k6) = 9155: ip(k7) = 9213: op(k7) = 9238
1160 GOTO 800
1999 :
2000 REM- Directory Printer
2010 GOSUB 50000: GOSUB 11100
2020 PRINT !(28); TAB(21); "Directory": PRINT
2030 FOR k = k0 TO ht: IF LEN(f$(k)) = k0 THEN 2080
2040 PRINT TAB(x*19); LEFT$(f$(k),k6);
2041 p = k8: IF k > k9 THEN p = k7
2050 PRINT TAB(x*19+p); ASC(MID$(f$(k), k7, k1));
2051 p = 12: IF k > k9 THEN p = 10
2060 PRINT TAB(x*19+p); ASC(RIGHT$(f$(k), k1));
2070 x = x + k1: IF x = k3 THEN x = k0: PRINT
2080 NEXT k: PRINT: PRINT
2090 INPUT "Press <RETURN> to Continue "; y$
2100 PRINT !(28);: GOTO 800
2110 :
3000 REM- Create Random Character Record
3010 FOR f = k1 TO nf: a$(f) = ""
3020 FOR  k= k1 TO fl(f) - k1
3030 c = INT(RND(k1) * ASC("z"))
3040 IF c => a0 AND c<=a9 THEN 3070
3050 IF c = >aa AND c<=az THEN 3070
```

labels across each page (i.e. 1-up, 2-up, 3-up, etc.). In addition, you may choose the number of fields to be printed on each line of each label, a character to be printed between each field (like the comma between CITY and STATE fields), and where on each page the labels are to be printed.

The setting for the character to be printed between each field is handled at line number 7190. Sometimes you may want to be able to print more than a single character between each field. To do so, you will have to modify both this input routine and the output routine at line number 7918 Don't forget that in order to print all <SPACE>s, your input routine will have to recognize a null entry as being all <SPACE>s and that it will have to ask for the number of them to be printed or impliment some other form of delimiter.

I know one of the hardest parts of deciphering someone else's program is trying to figure out what each variable represents. In MAILER, I have tried to use variable names that help describe their function, even though they're only 2 characters long For example, "nl" is the Number of Lines on each label, "ac" is the number of labels to be printed ACross each page, and so on.

Both the mailing label printer and the report writer suffer from a poor selection of terminology I have been prone to. When the program asks if you want to do any "sorts", it really means to ask if you want the program to search each record for a string in a selected field. This "sorting" allows you to print only selected records from the file, instead of relying solely on a range of record numbers, although you can certainly do that as well.

In producing PEEK[65], I use a program similar to this one to print out the mailing labels. The best advice I can give you is to make several runs that print only a single page of labels so that you can figure out the proper settings so that the printer doesn't stray from the labels. Until you get it down to a routine, in my experience it's all a matter of trial and error.

```
3060 IF (c<ASC("a")) OR (c>ASC("z")) THEN 3030
3070 a$(f) = a$(f) + CHR$(c): NEXT k, f: RETURN
3080 :
4000 REM- Edit DMS-650 Master File
4010 GOSUB 13000
4020 PRINT !(28) ;"DMS-650 Master File Editor":PRINT
4030 PRINT "(1) Add a New Record"
4040 PRINT "(2) Change an Old Record"
4050 PRINT "(3) Delete a Record"
4051 PRINT "(4) Return to Main Menu"
4060 PRINT: INPUT "    Your Choice "; y$: k = VAL(y$)
4070 IF k<k1 OR k>k4 OR k<>INT(k) THEN 4020
4080 ON k GOTO 4100, 4400, 4800, 4900
4090 :
4100 REM Add a Record
4110 IF tn = nr THEN PRINT "FILE FULL": GOSUB 60000: GOTO 4020
4120 FOR k = k1 TO nf: PRINT
4130 PRINT "Enter "; n$(k): PRINT TAB(k2);
4140 FOR l = k1 TO fl(k) - k1: PRINT "-";: NEXT l: PRINT
4150 INPUT a$(k): l = LEN(a$(k))
4160 IF l < fl(k) THEN NEXT k: GOTO 4180
4170 PRINT "TOO LONG !": PRINT: GOTO 4130
4180 PRINT !(28); " #"; TAB(k4);"Name"; TAB(32);"Contents": PRINT
4190 FORk = k1 TO nf: PRINT k; TAB(k4);n$(k); TAB(32);a$(k): NEXT k
4200 PRINT: INPUT "Are These Alright "; y$: y$ = LEFT$(y$+" ",k1)
4210 PRINT: IF y$ ="y" THEN 4300
4220 INPUT "Which one did you want to change " ;y$: k = VAL(y$)
4230 IF k<k1 OR k>nf THEN PRINT"WHAT ??": PRINT: GOTO 4180
4240 PRINT "Enter ";n$(k): PRINT TAB(k2);
4250 FOR l = k1 TO fl(k) - k1: PRINT"-";: NEXT l: PRINT
4260 INPU Ta$(k): l = LEN(a$(k)): IF l < fl(k) THEN 4180
4270 PRINT "TOO LONG": PRINT: GOTO 4240
4280 :
4300 tn = tn + k1: r6 = tn :GOSUB 400: GOTO 4020
4380 :
4400 REM- Change an Old Record
4410 PRINT: PRINT"File Contains"; tn; "Record(s)": PRINT
4420 IF tn=k0 THEN PRINT"NO RECORDS ON FILE":GOSUB 60000:GOTO 4020
4421 PRINT "(1) Edit by Record Number"
4422 PRINT "(2) Edit by Searching File": PRINT
4423 INPUT "    Your Choice "; y$: k = VAL(y$): PRINT
4424 IF k<k1 OR k>k2 OR k<>INT(k) THEN 4410
4425 ON k GOTO 4430, 4600
4430 INPUT "Which RECORD NUMBER did you want to see "; y$
4440 PRINT: k=VAL(y$): IF k<k1 OR k>tn OR k<>INT(k) THEN 4430
4450 r6 = k: GOSUB 300
4460 PRINT !(28);: dv = PEEK(8993): GOSUB 700
4480 INPUT "Did you want to change this record "; y$
4490 PRINT: IF LEFT$(y$+" ",k1) <> "y" THEN 4560
4500 INPUT "Enter the FIELD NUMBER you wanted to change "; y$
4510 PRINT: k=VAL(y$): IF k<k1 OR (k>nf) OR k<>INT(k) THEN 4500
4520 PRINT "Enter "; n$(k): PRINT: PRINT TAB(k2);
4530 FOR l = k1 TO fl(k)-k1: PRINT "-";: NEXT l: PRINT
4540 INPUT a$(k) :PRINT: l = LEN(a$(k)): IF l<fl(k) THEN 4560
4550 PRINT "TOO LONG!": PRINT: GOTO 4520
4560 GOSUB 400: GOTO 4020
4570 :
4600 REM- Search File for Editing
4610 GOSUB 8000: PRINT
4620 INPUT "Which FIELD NUMBER did you want to search in "; y$
4630 PRINT: k=VAL(y$): IF k<k1 OR (k>nf) OR k<>INT(k) THEN 4610
4640 PRINT "What STRING did you want to find in "; n$(k);
4650 INPUT " ";ss$: PRINT: l=LEN(ss$): IF l<fl(k) THEN 4670
4660 PRINT"TOO LONG !": GOSUB 60000: GOTO 4610
4670 sf = k: sl = LEN(ss$)
4671 GOTO 6000: REM- Remove this if Searches FAIL
4675 FOR r6 = k1 TO tn: GOSUB  300
4679 x = LEN(a$(sf)): FOR l = k1 TO x
4680 IF MID$(a$(sf), l, sl) = ss$ THEN l=x: NEXT l: GOTO 4700
4681 NEXT l
4690 NEXT r6: PRINT"STRING NOT FOUND": GOSUB 60000: GOTO 4020
4700 PRINT!(28);: dv=PEEK(8993): GOSUB 700
```

The report writer portion of the program is structured identically like the mailing label printer. You can select the range of record numbers to be printed, and select to do "sorts". I had planned on making it more sophisticated, but it does compliment the mailing label printer in its present form in that you can produce clean file dumps from your database.

HAVE YOU RENEWED YOUR SUBSCRIPTION?

Don't Miss an Issue! Renew Now!

```
4710 INPUT "Is this the right record "; y$
4720 IF LEFT$(y$+" ",k1) <> "y" THEN 4690
4730 x = r6: r6 = tn: NEXT r6: r6 = x: GOTO 4460
4740 :
4800 REM- Mark a Record for Deletion
4810 PRINT "File contains"; tn; "record(s)": PRINT
4820 IF tn=k0 THEN GOSUB 60000: GOTO 4020
4830 INPUT "Which RECORD NUMBER did you want to delete ";y$
4840 PRINT: k=VAL(y$): IF k<k1 OR k>tn OR k<>INT(k) THEN 4830
4850 r6 = k: GOSUB 300: a$(k1) = "^P": GOSUB 400: GOTO 4020
4860 :
4900 REM- Close DMS-65D Master File
4910 DISK get,k0: eodf = bodf + (tn*rl)
4920 i6 = bs(k6) + k9: ih = INT(i6/pg): il = i6 - ih*pg
4930 POKE op(k6),il: POKE op(k6)+k1 ,ih
4940 PRINT#k6, eodf: DISK close,k6: RUN
4950 :
6000 REM- Fast Device #6 Search Routine
6010 r6 = k1: GOSUB 100: REM- Initialize Pointer to BODF
6020 TRAP 6200: DISK find, ss$
6030 i6 = PEEK(ip(k6)) + (PEEK(ip(k6)+k1)*pg) - bs(k6) - k1
6040 i6 = i6 + (FNa(PEEK(9004))-st(k6)) * ts
6050 r6 = INT((i6-bodf)/rl) + k1
6052 GOSUB 300: l = LEN(a$(sf))
6060 FOR k = k1 TO l
6070 IF MID$(a$(sf),k,sl) = ss$ THEN 6090
6080 NEXT k: r6 = r6 + k1: GOSUB 100: GOTO 6020
6090 k = l: NEXT k: dv = PEEK(8993): GOSUB 700
6100 INPUT "Is this the correct record "; y$
6110 IF LEFT$(y$+" ",k1)<>"y"  THEN r6 = r6+k1: GOSUB 100: GOTO 6020
6130 TRAP 0: GOTO 4460
6140 :
6200 TRAP 0: PRINT "STRING NOT FOUND": GOSUB 60000: GOTO 4020
6210 :
7000 REM- Mailing Label Printer
7010 PRINT "Mailing Label Printer": PRINT: GOSUB 13000
7020 INPUT "How many labels will be printed across "; y$
7030 PRINT: ac=VAL(y$): IF ac<k1 OR ac<>INT(ac) THEN 7020
7050 INPUT "How many lines will be printed on each label "; y$
7060 PRINT: nl=VAL(y$): IF nl<k1 OR nl>nf THEN 7050
7070 REM- #of fields, field #, separation character
7080 DIM lf(nl), l$(nl,k2) ,p$(nl,ac), ta(ac)
7090 FOR ln = k1 TO nl :PRINT "Line #";ln: PRINT
7100 f1=k0: f2=k0: f3=k0: f4=k0
7110 INPUT "How many FIELDs will be printed on this line "; y$
7120 PRINT: lf(ln)=VAL(y$): IF lf(ln)>nf THEN 7110
7121 IF lf(ln) = k0 THEN 7250
7130 FOR l = k1 TO lf(ln): PRINT"#";l: PRINT: GOSUB 8011: PRINT
7140 INPUT "Field Number "; y$: PRINT
7150 f2 = VAL(y$): IF f2<k1 OR f2>nf THEN 7140
7160 l$(ln,k1) = l$(ln,k1) + MID$(STR$(f2), k2)
7170 IF l=lf(ln) THEN 7200
7180 l$(ln,k1) = l$(ln,k1) + ","
7190 PRINT "Enter the character that will be used to separate"
7191 INPUT "this field from the next one when it is printed ";y$
7192 l$(ln,k2) = l$(ln,k2) + LEFT$(y$+" ", k1)
7200 PRINT: NEXT l
7250 NEXT ln
7260 PRINT!(28): ma = k0
7270 FOR ln = k1 TO nl: PRINT ln;" ";: IF lf(ln)=k0 THEN PRINT:GOTO 7400
7280 xp=k1: x1=k1: t$="": ll=k0: IF lf(ln)=k0 THEN PRINT: GOTO 7370
7290 x=k0
7300 c = ASC( MID$(l$(ln,k1), xp, k1))
7310 xp = xp+k1: IF c = ASC(",") THEN 7340
7320 x = x*kt + VAL(CHR$(c)): IF xp <= LEN(l$(ln,k1)) THEN 7300
7340 t$ = t$ + n$(x): ll = ll + fl(x)-k1: IF xp > LEN(l$(ln,k1)) THEN 7360
7350 t$ = t$ + MID$(l$(ln,k2), x1, k1): x1 = x1+k1: ll = ll+k1: GOTO 7290
7360 PRINT t$
7370 IF ll > ma THEN ma = ll
7400 NEXT ln: PRINT
7410 INPUT "Is this alright "; y$
7420 PRINT: IF LEFT$(y$+" " ,k1) <> "y" THEN RUN
7425 PRINT "The largest line will be"; ma; "characters wide"
7430 PRINT :PRINT "For each label to be printed across, please"
7440 PRINT "enter the tab setting to be used": PRINT
7450 x1=k0: FOR k = k1 TO ac: PRINT "Label #"; k
7460 INPUT "Tab Setting "; y$: PRINT: xp = VAL(y$)
7470 IF xp <= x1 THEN PRINT"TOO FAR LEFT": PRINT: GOTO 7460
7480 ta(k) = xp: x1 = ta(k)+ma: NEXT k: PRINT
7490 PRINT "These settings require a page width of"; x1
7500 PRINT: INPUT "Is this alright "; y$
7510 PRINT: IF LEFT$(y$+" ", k1) <> "y" THEN 7423
7520 PRINT "How many lines should be skipped after each"
7530 INPUT "label has been printed (ie. between labels) "; y$
7540 PRINT :sk = VAL(y$): IF sk <> k0 THEN 7570
7550 INPUT "Are you sure you want 0 lines skipped "; y$
7560 PRINT: IF LEFT$(y$+" ",k1) <> "y" THEN 7520
7570 PRINT "There are"; tn; "records in "; f$: PRINT
7580 INPUT "Enter the RECORD NUMBER you wish to start with "; y$
```

OSI-CALC: SPREADSHEET PROGRAM

OSI-CALC has been a smash hit here at PEEK[65]. Written entirely in BASIC by Paul Chidley of TOSIE, the program gives you a 26 column by 36 row spreadsheet with many features. Don't let the fact that it's written in BASIC fool you. It's VERY FAST.

Each cell can contain text (left or right justified) or numeric data (in floating point or dollar format) or a formula which computes its results based on the contents of the other cells. Formulas can perform addition, subtraction, multiplication or division using cell contents and/or numeric constants. Spreadsheets can be stored on disk, and the program does very nice printing too.

OSI-CALC requires 48K of memory and OS-65D V3.3. Specify video or serial system and mini-floppy or 8" disks. Price $10.00 plus $3.70 shipping ($13.70 total).

U-Word: A Preview

by Richard L. Trethewey

U-Word is the newest word processor for serial systems running under OS-65U. The program's main claim to fame is that it runs on virtually any 65U-compatible system including Level 3, Portland boards, and (reportedly) Denver Boards. In fact, that's why Softouch wrote it in the first place. They needed a word processor that would run on a Portland board for a customer.

U-Word displays the text on the screen exactly as it will appear on paper. The text is stored in memory, so document size is limited. The current version has a buffer that is approximately 14K long. That's enough to store 3 or 4 pages of single-spaced text. The actual figure varies depending on the number of lines and the length of each one.

The display is very clean, with the top four lines occupied by the title, the current INSERT/TYPEOVER mode setting, and a ruler which shows the character positions, TAB and margin

continued on page 10

continued from page 9

settings. The top display also includes a prompt to enter "<ESC>?" to get help. Entering that command brings up a menu screen that shows all of the commands that are available. Entering one of the commands brings up another screen which describes that command

Moving the cursor is accomplished using the TAB, HOME, and arrow keys or by using one of the two-keystroke commands. The program can be configured to use just about any terminal. If your terminal supports it, you can program your function keys to replicate the command sequences, or the program will allow you to alter the command sequences to suit your tastes (although the version I used didn't support this feature).

In reviewing the program, I entered about two screens worth of text. The program performed smoothly here, quickly handling word wrapping when necessary. To test the program's editing abilities, I tried moving the cursor with the arrow keys and typing over and inserting text at random rates and intervals. I am not fond of the way the DELETE key is handled. If the cursor is at the end of the line, the program backspaces and then deletes. But if it's in the middle of text, the character under the cursor is deleted. Nitpicking, perhaps, but I didn't feel comfortable with this. The program also tended to lose keystrokes when inserting text near the start, although that is understandable to a certain extent.

Searching for words was very fast. The program also allows you to immediately jump to the start or end of the text, or page forward and backward one page at a time. In fact, all cursor movement was quick, smooth, and easy.

Saving or loading text to or from disk is easy, you just issue the disk command, enter the name of the file you want to use and then press "Y". I would have liked to have had the ability to select the disk drive here too so as to be able to keep multiple copies of documents, but that feature isn't available.

Softouch is releasing what they call version 1.0 of U-Word, planning to add features as demand for the product dictates. Future versions will support block operations, search and replace, and merging with OS-DMS data files. Softouch considers this release to be about 60% of what the final product will be, and have priced it accordingly. As each new version is released, current owners will be able to upgrade for the cost of the number of steps they're buying, to a final estimated price of $395.00 (from its current $237).

Overall, U-Word performed well. The on-line help screens made it easy to use even though I didn't have any documentation to guide me. The program's compatibility and flexibility are sure to make it popular, especially with those who have been locked out of word processing by incompatible hardware. Best of all, at $237.00, it's priced right.

U WORD

Finally! An OS-65U based word processor that's fast, easy to use, user configurable, actively supported, and available now!

SOFTOUCH, Inc. announces the release of "U-WORD" for Ohio Scientific Challenger and 200 series computers. This powerful word processor works on single user, timeshared, and multiprocessed ["Portland board"] computers.

- Very fast editing [machine code]
- Printer selectable at time of printout
- On-line help screens
- Menu-driven configuration

"U-WORD" Version 1.0 is available NOW for $237⁰⁰

SOFTOUCH INC.

3120 Far Hills Avenue • Dayton, Ohio 45429
• [513] 299-3473

8 More K for the 610 Board

by Scott Larson

Just because the C1P/Superboard was designed almost 10 years ago, doesn't mean that it can't take advantage of many of the new chips that are becoming available. This is a simple example of how you can improve your old computer with new technology.

A fully populated 610 board has 48 2114 static memory chips to give the C1P atotal of 32K, but still leaves 8K of memory space unused. This design has other flaws as newer static memory chips have been created that have much more memory to a chip, use less power, and best of all, are much cheaper. The · 64 chip (seen in ads as HM6264LP-15) has 8K by 8 bits in a single 28 pin package, which is the equivalent of 16 2114's and a 74LS138 in one chip. The 610 board providesall the neccesary signals for the chip and is very easily connected to its 40 pin socket to fill the last 8K space to give the C1P a total of 40K. This chip has dropped in price over 100% in the past year, and is available from JDR Microdevices, Jameco and other major companies for only $4 or $5. The only other materials needed are a 28 pin DIN socket for the chip, a 40 pin DIN plug for J2 on the 610 board, and a 74LS08 for a small amount of decoding.

Only one small modification is needed to the 610 board. When an 8K block is addressed on the 610 board, a gate in U8 (74LS20) provides the DD (data direction) line. Since all the inputs of this gate are used, we have to disconnect one of the inputs to the gate (U8 pin 13), connect one of the inputs (U18 pin 12), and the chip select for the new 8K (U18 pin 11) to an AND gate. The output is then connected to the gate (U8 pin 13). The rest of the connections to the 8K chip are shown in this table:

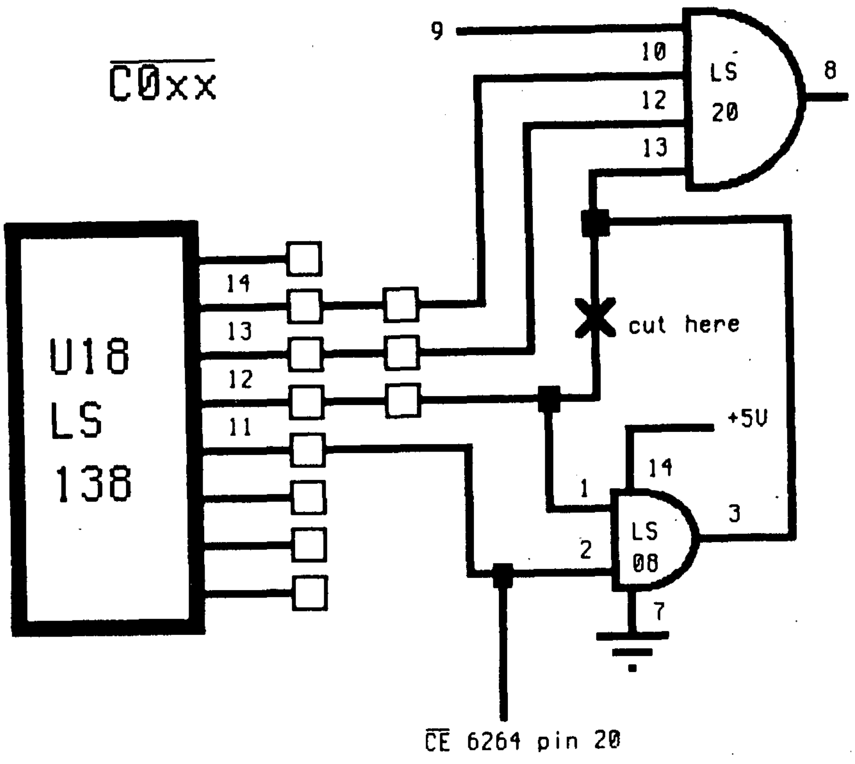


Figure 1

Function	610 board	6264
A0	J2-pin 1	pin 10
A1	J2-pin 2	pin 9
A2	J2-pin 3	pin 8
A3	J2-pin 4	pin 7
A4	J2-pin 5	pin 6
A5	J2-pin 6	pin 5
A6	J2-pin 7	pin 4
A7	J2-pin 8	pin 3
A8	J2-pin 33	pin 25
A9	J2-pin 34	pin 24
A10	J2-pin 35	pin 21
A11	J2-pin 36	pin 23
A12	J2-pin 37	pin 2
D0	J2-pin 10	pin 11
D1	J2-pin 11	pin 12
D2	J2-pin 12	pin 13
D3	J2-pin 13	pin 15
D4	J2-pin 28	pin 16
D5	J2-pin 29	pin 17
D6	J2-pin 30	pin 18
D7	J2-pin 31	pin 19
CE NOT	U18-pin 11	pin 20
R/W	U5-pin 9	pin 27
R/W NOT	U5-pin 8	pin 22
+5		pin 26
+5		pin 28
GND		pin 14

The 6264 uses so little power (and dissipates no measurable heat), it can take power directly from the 600 or 610 board. After making these onnections, I had only one small problem. One of the data lines couldn't function with the added load (not surprising considering that there are 14 2114 chips connected to it). So I simply connected the data input of the chip to the 8T28 buffer input and let the 8T28's on the 600 board handle it. I have low power 2114's on my 610 board, but some might not so I have included a table of the data inputs of all the data lines in case some have more trouble than I did. If your data comes back with bits missing, simply disconnect the appropriate line from J2, and connect it to the input pin in the table below;

Line	Input
D0	U13-pin 2
D1	U13-pin 12
D2	U13-pin 5
D3	U13-pin 9
D4	U14-pin 2
D5	U14-pin 12
D6	U14-pin 5
D7	U14-pin 9

Support Your Local OSI Dealer or Vendor

```
7590 PRINT: sr = VAL(y$): IF sr<k1 OR sr>tn THEN 7580
7600 INPUT "Enter the RECORD NUMBER you wish to end with "; y$
7610 PRINT: er = VAL(y$): IF er<sr OR er>tn THEN 7600
7620 ns = k0: INPUT "Did you want to do any sorts "; y$
7630 PRINT: IF LEFT$(y$+" ",k1) <> "y" THEN 7730
7640 INPUT "How many sorts did you want to do "; y$
7650 PRINT: ns=VAL(y$): IF ns=k0 THEN 7620
7660 DIM ss$(ns), sf(ns): FOR x = k1 TO ns
7670 PRINT "Sort #"; x: PRINT: GOSUB 8011
7680 PRINT: INPUT "Search in which FIELD NUMBER "; y$
7690 PRINT: sf(x)=VAL(y$): IF sf(x)<k1 OR sf(x)>nf THEN 7670
7700 PRINT "Search for what STRING in "; n$(sf(x));: INPUT" "; y$
7710 PRINT: ss$(x) = y$: l = LEN(ss$(x)): IF l => fl(sf(x)) THEN 7700
7720 NEXT x: PRINT
7730 INPUT "Enter the DEVICE NUMBER for this printing "; y$
7740 PRINT: dv = VAL(y$) OR PEEK(8993): REM Always echo to console!
7750 r6 = sr: x1 = k1
7760 GOSUB 300: IF a$(k1) = "^P" THEN 7840
7761 IF ns = k0 THEN GOSUB 7911: GOTO  7830
7770 go$="pass": FOR k = k1 TO ns: l1=LEN(ss$(k)): l2=LEN(a$(sf(k)))
7780 FOR j = k1 TO l2
7790 IF MID$(a$(sf(k)), j, l1) = ss$(k) THEN 7810
7800 NEXT j: go$="fail": GOTO 7820
7810 j = l2: NEXT j
7820 NEXT k: IF go$="pass" THEN GOSUB 7911
7830 IF x1 > ac THEN GOSUB 7880: x1 = k1
7840 IF r6 <> er THEN r6 = r6 + k1: GOTO 7760
7850 IF x1 <> k1 THEN GOSUB 7880
7860 INPUT "Press <RETURN> to continue "; y$: RUN
7870 REM- Print Labels
7880 FOR l = k1 TO nl: FOR k = k1 TO ac: PRINT#dv,TAB(ta(k));p$(l,k);
7890 p$(l,k) = "": NEXT k: PRINT#dv: NEXT l
7900 FOR l = k1 TO sk: PRINT#dv: NEXT l: RETURN
7910 :
7911 FOR ln = k1 TO nl: IF lf(ln) = k0 THEN 7919
7912 xp = k1: xq = k1
7913 x = k0
7914 c = ASC(MID$(l$(ln,k1), xp, k1))
7915 xp = xp + k1: IF c = ASC(",") THEN 7917
7916 x = x*kt + VAL(CHR$(c)): IF xp < LEN(l$(ln,k1)) THEN 7914
7917 p$(ln,x1) = p$(ln,x1) + a$(x): IF xp > LEN(l$(ln,k1)) THEN 7919
7918 p$(ln,x1) = p$(ln,x1) + MID$(l$(ln,k2) ,xq, k1): xq=xq+k1: GOTO 7913
7919 NEXT ln: x1 = x1 + k1: RETURN
7998 END
7999 :
8000 REM- Display Fields
8010 PRINT !(28);"File: "; f$: PRINT
8011 PRINT " #"; TAB(k4); "Field Name"; TAB(32); "Length": PRINT
8020 FOR k = k1 TO nf: PRINT k; TAB(k4); n$(k); TAB(34); fl(k)-k1
8030 NEXT k: RETURN
8040 :
11100 s = k1:REM- Gather Directory
11101 FOR k = k0 TO ht: ut(k) = k0: f$(k)="": NEXT k
11105 DISK!"ca 2e79=" + dt$ + RIGHT$(STR$(s),k1)
11110 FOR i = di TO di + pg-k1 STEP k8: IF PEEK(i) = e THEN 11150
11120 st = FNa(PEEK(i+k6)): et = FNa(PEEK(i+k7))
11130 FOR j = k0 TO k5: f$(st) = f$(st) + CHR$(PEEK(i+j)): NEXT j
11140 f$(st) = f$(st) + CHR$(st) + CHR$(et)
11146 FOR k = st TO et: ut(k) = k1: NEXT k
11150 NEXT i: IF s=k1 THEN s=k2: GOTO 11105
11160 RETURN
11170 :
13000 REM- Open a DMS-65D Master File on Device 6
13010 TRAP 58000: GOSUB 50000
13020 INPUT "File Name "; f$: PRINT: IF LEN(f$)>k5 THEN 13020
13030 IF LEN(f$)<k5 THEN f$ = f$ + " ": GOTO 13030
13040 f$ = f$ + "0": DISK open, k6, f$: TRAP 0
13050 st(k6) = FNa(PEEK(9002)): et(k6) = FNa(PEEK(9003))
13080 i6=k9: GOSUB 210: INPUT#k6,eodf
13090 i6=20: GOSUB 210: INPUT#k6,bodf
13100 i6=31: GOSUB 210: INPUT#k6,rl
13110 i6=42: GOSUB 210: INPUT#k6,nr
13120 i6=53: GOSUB 210: nf=k0
13130 INPUT#k6,y$ ,k: nf = nf+k1
13140 i6 = (PEEK(9132) + PEEK(9133)*pg) - bs(k6)
13150 i6 = i6 +( FNa( PEEK(9004) ) - FNa( PEEK(9002)) )* ts
13160 IF i6 < bodf THEN 13130
13170 IF PEEK(9004) = PEEK(9002) THEN 13190
13180 DISK!"ca 3a7e=" + RIGHT$(STR$(FNa(PEEK(9002))),k2) + ",1"
13190 i6 = 53: GOSUB 210: DIM n$(nf),fl(nf),i6(nf),a$(nf):i=k0
13200 FOR k= k1 TO nf: INPUT#k6,n$(k),fl(k): i6(k)=i: i=i+fl(k): NEXT k
13210 tn = INT((eodf-bodf)/rl): RETURN
13220 :
20000 REM- Report Writer
20010 PRINT "DMS-65D Report Writer": PRINT: GOSUB 13000
20020 INPUT "Enter the TITLE for this report "; ti$
20030 PRINT: GOSUB 8011
20040 INPUT "How many FIELDS will be printed on each line "; y$
20050 PRINT: ac=VAL(y$): IF ac<k1 OR ac>nf THEN 20040
```

SAM the (S)elf (A)ware (M)icrocomputer

by Richard E Reed

In the misty winter dawn shining through the tawdry ghetto windows of his lab the scientist (obviously mad) stoops over the tangle of wires and boards on the operating table. His fingers can be seen hovering above the petrified caterpillars, clicking toggle switches and closing momentary contacts. In the early morning stillness his feverish voice echoes,... pontificating like GOD as the customary lights (albeit tiny) flash off and on in sync with his actions.

"Clear his experience banks!!!" (clear the memory...)
"Start his character records" (Set some counters to zero...)
"Initialize the life forces!!!" (Turn it on, you dummy...)

There is a flurry of activity among the lights. The haggard and unkempt form arises in triumph.

"He works! He's working! Ha ha ha ha ha! He's working!!!"
"SAM is born!" (to nobody) "Look at him,--he works!!!"

Nine days before I had locked myself in the run-down room on motel row in Fresno, armed with an OSI bare board, a couple of bread boards from Radio Shack, some LEDs, switches, a bag of ICs, and the intention to create SAM. I added some junk food and drinks, and for a week and a half I lost track of time while I attempted to put my ideas in silicon and electric fields.

I confess! I was an OSI addict from the word go. I bought one of their first bare boards and ICs to populate it. I had already acquired my 6502 from MOS Technology as soon as they announced it for an unbelievable $25.00. I learned to program in machine language while waiting for the boards to arrive, and decided to whet my skills with an ambitious project.

I am also an AI (A)rtificial (I)ntelligence freak (not the AI that is being palmed of as such today, where

```
20060 DIM ln(ac), ta(ac)
20070 ma = k0: FOR x = k1 TO ac: PRINT "Current TAB is"; ma
20080 PRINT: GOSUB 8011: PRINT
20090 PRINT "Enter the FIELD NUMBER for position"; x; " ";
20100 INPUT y$: PRINT: l=VAL(y$): IF l<k1 OR l>nf THEN 20080
20110 PRINT "Enter the TAB SETTING for position"; x; " ";
20120 INPUT y$: PRINT: ta(x) = VAL(y$): IF ta(x)=k0 THEN 20135
20130 IF ta(x) <= ma THEN PRINT "TOO FAR LEFT": GOTO 20110
20135 ln(x) = l: t = fl(ln(x)): IF t > LEN(n$(l)) THEN 20140
20136 t = LEN(n$(l))
20140 ma = ta(x) + t: NEXT x: PRINT: pw = ma
20141 PRINT "Page Width is currently "; pw
20142 INPUT "Did you want to change this "; y$
20143 PRINT: IF LEFT$(y$+" ",k1) <> "y" THEN 20150
20144 INPUT "Enter new PAGE WIDTH "; y$
20145 PRINT: x = VAL(y$)
20146 pw = x
20150 PRINT f$; " contains"; tn; "records.": PRINT
20160 INPUT "What RECORD NUMBER you wish to start with "; y$
20170 PRINT: sr=VAL(y$): IF sr<k1 OR sr>tn THEN 20160
20180 INPUT "What RECORD NUMBER you wish to end with "; y$
20190 PRINT: er=VAL(y$): IF er<sr OR er>tn THEN 20180
20200 ns=k0: INPUT "Did you want to do any sorts "; y$
20210 PRINT: IF LEFT$(y$+" ",k1) <> "y" THEN 20310
20220 INPUT "How many sorts did you want to make "; y$
20230 PRINT: ns = VAL(y$): DIM ss$(ns), sf(ns)
20240 FOR x = k1 TO ns
20250 PRINT "Sort #"; x: PRINT: GOSUB 8011: PRINT
20260 INPUT "Sort in which FIELD NUMBER "; y$
20270 PRINT: sf(x)=VAL(y$): IF sf(x)<k1 OR sf(x)>nf THEN 20260
20280 INPUT "What STRING did you want to search for "; y$
20290 PRINT: ss$(x)=y$: l=LEN(ss$(x)): IF l=>fl(sf(x)) THEN 20280
20300 NEXT x: PRINT
20310 ti = INT( (ma-LEN(ti$)) /k2): REM- Title centering
20320 INPUT "Which DEVICE NUMBER is this to be printed on "; y$
20330 dv=VAL(y$) OR PEEK(8993)
20340 pn = k1: GOSUB 20450
20350 FOR r6 = sr TO er: GOSUB 300: IF a$(k1)="^P" THEN 20420
20360 IF ns = k0 THEN 20410
20370 FOR x = k1 TO ns: l1 = LEN(ss$(x)): l2=LEN(a$(sf(x)))
20380 FOR j = k1 TO l2: IF MID$(a$(sf(x)), j, l1) = ss$(x) THEN 20400
20390 NEXT j: x=ns: NEXT x: GOTO 20420
20400 j = l2: NEXT j: NEXT x
20410 GOSUB 20520
20420 NEXT r6
20430 INPUT "Press <RETURN> to continue "; y$: RUN
20440 :
20450 PRINT#dv,CHR$(12); TAB(pw-k8);"Page";pn: PRINT#dv
20460 PRINT#dv,TAB(ti); ti$: PRINT#dv
20470 FOR l = k1 TO pw: PRINT#dv,"-";: NEXT l: PRINT#dv
20480 FOR l = k1 TO ac: PRINT#dv, TAB(ta(l)); n$(ln(l));: NEXT l
20490 PRINT#dv: FOR l = k1 TO pw: PRINT#dv,"-";: NEXT l: PRINT#dv
20500 lc = k8: pn = pn+k1: RETURN
20510 :
20520 FOR l = k1 TO ac: PRINT#dv, TAB(ta(l)); a$(ln(l));: NEXT l
20530 PRINT#dv: lc = lc+k1: IF lc>60 THEN GOSUB 20450
20540 RETURN
20550 :
50000 INPUT "Drive (A/B/C/D) "; y$: y$ = LEFT$(y$+" ",k1)
50010 PRINT: c = ASC(y$): IF c>az THEN c = c - tt
50020 IF c<aa OR c>ASC("D") THE N50000
50030 DISK!"se " + CHR$(c): RETURN
50040 :
58000 REM- Show File Not Found
58010 PRINT: PRINT "FILE: ";f$;" NOT FOUND": PRINT
58020 :
58999 REM- Abort!
59000 GOSUB 60000: RUN
59010 :
60000 FOR k = k1 TO 3000: NEXT k: RETURN
```

a few language and logical functions are being emulated). We're talking the Frankensteinian variety,...trying to create viable beings. Now that my own real computer was available I set out in earnest to devise (in 4K total RAM) a being with the following properties:

1. He must start with no knowledge.
2. He must behave randomly at first.
3. He must learn to survive.
4. He must develop habits.
5. He must have virtue and vice.
6. He must behave unpredictably.
7. He must develop differently each time he is run.
8. He must be able to forget.
9. The program must fit 256 bytes.
10. He must run in a 4K machine.
11. He must be expandable.
12.He must install in a robot

SAM fulfilled every specification. As originally conceived, his senses and his coin-flip decisions came from a free-running counter I installed at $4000 Hex. The highly erratic TTL gated clock ran at about 33MHz. SAM operated in a world of 256 environmental situations (one byte's worth), and could act on each of these in 8 different ways. His actions affected a "good" and a "bad" counter which tracked his progress.

A look at the random number counter gave him an environmental situation. He then went to a 2K block of memory and examined 8 bytes to see if he had dealt with that environment before. (Each page of the 2K block epresented a response, whlethe ndividualbyte represented an environment.) In that location two things were stored: the value of the environment- reaction, and the number of times it was used. After his examination SAM knew whether any reactions had been tried, and (if so) which one had the highest value

If no reactions were tried, SAM went to the random number generator to get one. By masking off the high order 5 bits he selected one of 8 behaviors. Each behavior consisted of performing an EXclusive-OR between the environment and one of 8 arbitrarily chosen bytes. The result was divided into two 4-bit nybbles. The upper 4 bits gave a value of 0 to 15 which could be added to the good counter. The lower nyble was added to the bad counter. The two were combined to yield a value from 0 to 31. This was stored in the low order 5 bits of the appropriate memory location. Each time this particular reaction was used 32 was added to the byte (up to a maximum of 7 uses). This incremented the upper 3 bits.

If the current environment had been acted upon before, SAM checked to see if it had been used a total of 7 times. If so, he repeated it automatically. If not he went to the random number generator and flipped a 32-sided "coin". If the result of that flip was greater than the reaction value, SAM went to the random selection routine described above. If it was smaller, he repeated the old

```
5 REM <<<<<<<< SAM by Richard Reed >>>>>>>>
10 GOTO 10000
100 IF N2/N9% = INT(N2/N9%) THEN Z = -Z
110 Z = RND(Z): Z% = INT(Z*X%): RETURN
200 J% = A%(G%,F%) AND N1%: L% = A%(G%,F%) AND N2%: H% = L%: RETURN
400 IF A%(R,S) < N2% THEN A%(R,S) = A%(R,S) AND N1%
410 S = S + 1: IF S > N4% THEN S=N%: R = R + 1
415 IF R > 7 THEN R = N%
420 I FS / 3 = INT(S/3) THEN 400
430 RETURN
500 IF A%(G%,F%) < N2% THEN A%(G%,F%) = A%(G%,F%) + 32
501 RETURN
600 IF P% < 2 THEN RETURN
610 RESTORE: FOR K = N% TO N3%+1: READ K$: PRINT#P%,TAB(B%*K);K$;:NEXT:PRINT#P%
620 RETURN
700 B%(G%,F%) = B%(G%,F%) + 1: RETURN
5000 X%=O%: GOSUB D: F%=Z%: H%=N%: J%=N%: FOR I = N% TO N3%: A%=A%(I,F%) AND N1%
5010 IF A%>J% THEN H%=1: J%=A%: G%=I: L% = A%(I,F%) AND N2%
5020 NEXT I
5030 X% = N1%: GOSUB D: REM IF L% = N3% THEN 5100
5040 IF H%>N% AND ((J%*1.452-8)>=Z% OR L%=N2%) AND N2/50<>INT(N2/50) THEN 5100
5050 X%=8: GOSUB D: G%=Z%: J% = A%(G%,F%) AND N1%: L% = A%(G%,F%) AND N2%: H%=L%
5060 IF H% > N% THEN 5030
5100 T% = F% AND NOT C%(G%): K% = T%
5110 L% = INT(K%/16): M = K% - L% * 16: N = 17 - L% + M: IF N>30 THEN N = N1%
5120 IF SR = 1 THEN RETURN
5130 AT% = A%(G%,F%) AND 31: IF AT% = N% THEN A%(G%,F%) = A%(G%,F%) + N
5135 GOSUB E1: GOSUB 500: DY = INT(A%(G%,F%) / 32)
5140 IF LC% > 80 THEN PRINT#P%, CHR$(12): GOSUB 600: LC% = 1
5150 P = P + L%: Q = Q + M: PRINT#P%, N2, Q, P, N, M, F%, G%, B%(G%,F%), DY
5160 N2 = N2 + 1: LC% = LC% + 1: GOSUB 400: IF Q < TV THEN 5000
5165 DIM T%(32): DIM T1%(32): PRINT#P%, CHR$(12)
5180 SR=1: FOR W= N% TO N4%: FOR I= N% TO N3%: T1=(A%(I,W)AND N2%)/32: F%=W:G%=I
5185 GOSUB 5100: T2 = N: T3 = B% * I
5187 T%(N) = T%(N) + 1: T1%(N) = T1%(N) + B%(I,W)
5190 PRINT#P%, TAB(T3); T1; T2; TAB(T3+7);B%(I,W);: NEXT I:PRINT#P%,TAB(115);W
5195 NEXT W: PRINT#P%, CHR$(12)
5200 FOR I= 1 TO 130: N$ = N$+"*" :N1$ = N1$+"%": NEXT I
5220 FOR I = N% TO N1%: IF T1%(I)  >AD% THEN AD% = T1%(I)
5225 IF T%(I) > AD% THEN AD% = T%(I)
5227 NEXT I
5230 IF AD% < 120 THEN 5300
5235 DI = INT(AD% / 120) + 1
5240 FOR I= N%TON1%: T1%(I) = INT(T1%(I)/DI): T%(I) = INT(T%(I)/DI): NEXT I
5300 FOR I = N% TO N1%: PRINT #P%,I; TAB(5); LEFT$(N$,T%(I))
5310 PRINT #P%, TAB(5); LEFT$(N1$,T1%(I)): NEXT I: POKE 2720, 14: END
6000 DATA MOVE, POSITIVE, NEGATIVE, VALUE,+ VAL, ENVIR, REACT, USE, SIEVE
10000 O%=256: DIM A%(8,O%), B%(8,O%): M%=1: N2=M%: D=100: E=200: E1=700
10010 Z%=23456:B%=14:N%=0:N0%=1:N1%=31:N2%=224:N3%=7:N4%=255:N9%=1000
10020 INPUT "Output Device"; P%: INPUT "Maximum value"; TV
10026 FOR I= 0 TO 7: C%(I)=INT(I* 36.428715): PRINT#P%, TAB(7*I); C%(I);: NEXT I
10027 GOTO 10040
10030 FOR I= 0 TO 7: X%=O%: GOSUB D: C%(I)=Z%: PRINT#P%,TAB(7*I);Z%;: NEXT I
10040 PRINT #P%: PRINT #P%
10060 GOSUB 600: POKE 2720, 8: GOTO5000
```

reaction and added 32 to the use counter if it was not at the maximum.

Once every action SAM would examine a memory location (stepped through from the beginning of the memory block to the end, then around again) to see if its combination had been used 7 times. If not SAM xeroed that location, forgetting he had ever done anything with that environment/reaction. This constituted the complete SAM. In a typical run of 40,000 moves his "good" counter had advanced about 3.3 times higher than the "bad" (500000/150000), and his "habits" were fairly well established; ie: 93% of his actions were on fixed memory.

SAM's design has proven to be very flexible. A new environment can be added with the addition of 2K of RAM and a little overhead program to switch between environments. He can have 62 environments in a 64K RAM The complexity of the environment or the reactions can be altered virtually by changing the memory allocated to them

Installation within a robot merely involves substituting sensory input for the random generator, and output to motor devices instead of the numeric reactions. In a simple scenario SAM could receive energy for "watering" plants. When his watering can got low, he could receive "points" for fetching more water. When his power was low he could switch back to getting points for watering plants If he were turned loose with either the water in his bucket and power in his battery, and these parameters were provided in his environment, he would soon teach himself his chores, and manage to keep the plants watered and his power up.

While SAM was originally written in machine language, and his current version is propriatary information I am not free to share, I have written a BASIC version which operates almost identically with the original SAM but which offers greater ease of user modification, greater simplicity of understanding, and more extensive reporting of what has happened. That listing is included for anyone who may wish to experiment with SAM themselves.

Lines 10000 to 10060 define variables and set up the output device, the total count SAM is expected to attain, and 8 random reactions. Lines 100 and 110 are the random number generator. N% is the current event count, and every 1000 moves it selects a new random seed. X% contains the maximum value to be returned. Z% is the new random number.

200 gets the reaction value and the use counter in J% and L% respectively H%is a flag to show prior usage.

The subroutine at 400 to 430 operates the memory "sieve". Line 400 has been modified to clear only the use count rather than the whole byte for reporting purposes. This fact has not been used to modify the operation of SAM. 410 cycles the lower address byte, and when necessary, the page number. 420 resets the page number if the cycle has gone full circle. Line 415 was added to speed up the sieve a little. Lines 500 and 510 increment the use counter if it is less than maximum.

The main program begins in line 5000, which picks up an "environment" from the random number generator. Subroutine calls and line calls are named variables because of the increase in speed of execution. Subroutine D is the random generator. X% is the size limiter, Z% is the returned number, and F% is the environment variable.

5010 to 5030 look at all possible reactions to see if any were used and if there value is greater than the last one found. In 5032 we get a 31-sided coin flip, and if the reaction has the maximum usage, we skip testing the coin. Line 5040 sets some other skip tests; the test involving J%*1.452-8 expands the acceptance/rejection criteria by always excluding very bad reactions and always using very good ones; the test using N2/50 selects an unused reaction every 50 moves so that SAM can learn new habits from time to time. Lines 5050 and 5060 get new reactions and test them if the coin flip warrants it.

Lines 5100 through 5160 perform the mathematical calculations involved in performing a reaction, updating the memory, printing the results, and getting another environment. One line that needs comment is 5125. It is a conditional return that makes use of the lines just above it for a subroutine call from the tabulated reports printed after line 5160.

AD$

UCSD Pascal/Fortran 8" for polled keyboard $50, 502 board $30, D&N proto board $20, bare 80-column video board $25, Sams C2-C3 manual $20. Ron Battle, 1011 Yale NE, Albequerque, NM 87106

FOR SALE: 12 fully populated 520 boards. Each provides 16K of static RAM. Not tested. $50.00 plus shipping. Contact PEEK(65)

FORTH $24.95. Utilities available also. Free catalog. Aurora Software, 37 South Mitchell, Arlington Heights, IL 60005

Have you got something to sell? Why not take out a classified ad in PEEK? Ads cost 35 cents per word, not including "price" words. Copy is due 30 days before the cover month.

ATTENTION: DEALERS!

PEEK(65) needs new subscribers and you need new customers, and together we can make it happen with our own Co-op advertising program. This program pays dealers for signing up new subscribers with free ad space in PEEK(65). Just five paid subscriptions will earn a 1/9th page advertising credit in PEEK(65).

Most dealers sell their own software with the systems they install. By advertising in PEEK, you vastly expand the potential market for your products. And how many sales have you lost because you couldn't find the application your customer wanted? Dealer ads can be our own Yellow Pages. Readers and customers win too by increasing the number of uses for their equipment.

Call or write today for details and your free promotional materials. Making a PEEK(65) subscription a part of every sale is painless and profitable. This time, "Co-op" pays you.

Watch
This
Space
Grow!

PEEK [65]

PO Box 586
Pacifica, CA 94044
415-359-5708

Bulk Rate
U.S. Postage
PAID
Pacifica, CA
Permit #92
Zip Code 94044

DELIVER TO:

Gerald M. Van Horn
640 SW Addison Ave.
Junction City, OR
97448

GOODIES for OSI Users!

PEEK (65)
The Unofficial OSI Users Journal

( )	**C1P Sams Photo-Facts Manual.** Complete schematics, scope waveforms and board photos. All you need to be a C1P or SII Wizard, just	$7.95	$
( )	**C4P Sams Photo-Facts Manual.** Includes pinouts, photos, schematics for the 502, 505, 527, 540 and 542 boards. A bargain at	$15.00	$
( )	**C2/C3 Sams Photo-Facts Manual.** The facts you need to repair the larger OSI computers. Fat with useful information, but just	$30.00	$
( )	**OSI's Small Systems Journals.** The complete set, July 1977 through April 1978, bound and reproduced by PEEK (65). Full set only	$15.00	$
( )	**Terminal Extensions Package** - lets you program like the mini-users do, with direct cursor positioning, mnemonics and a number formatting function much more powerful than a mere "print using." Requires 65U.	$50.00	$
( )	**RESEQ** - BASIC program resequencer plus much more. Global changes, tables of bad references, **GOSUBs** & GOTOs, variables by line number, resequences parts of programs or entire programs, handles line 50000 trap. Best debug tool I've seen. MACHINE LANGUAGE - VERY FAST! Requires 65U. Manual & samples only, $5.00 Everything for	$50.00	$
( )	**Sanders Machine Language Sort/Merge** for 0S-65U. Complete disk sort and merge, documentation shows you how to call from any BASIC program on any disk and return it or any other BASIC program on any disk, floppy or hard. Most versatile disk sort yet. Will run under LEVEL I, II, or III. It should cost more but Sanders says, "...sell it for just..."	$89.00	$
( )	**KYUTIL** - The ultimate OS-DMS keyfile utility package. This implementation of Sander's SORT/MERGE creates, loads and sorts multiple-field, conditionally loaded keyfiles. KYUTIL will load and sort a keyfile of over 15000 ZIP codes in under three hours. Never sort another Master File.	$100.00	$
( )	**Assembler Editor & Extended Monitor Reference Manual** (C1P, C4P & C8P)	$6.95	$
( )	**65V Primer.** Introduces machine language programming.	$4.95	$
( )	**C1P, C1P MF, C4P, C4P DF, C4P MF, C8P DF Introductory Manuals** ($5.95 each, please specify)	$5.95	$
( )	**Basic Reference Manual** — (ROM, 65D and 65U)	$5.95	$
( )	**C1P, C4P, C8P Users Manuals** — ($7.95 each, please specify)	$7.95	$
( )	**How to program Microcomputers.** The C-3 Series	$7.95	$
( )	**Professional Computers Set Up & Operations Manual** — C2-OEM/C2-D/C3-OEM/C3-D/C3-A/C3-B/C3-C/C3-C'	$8.95	$

TOTAL	$
CA Residents add 6% Sales Tax	$
C.O.D. orders add $1.90	$
Postage & Handling	$ 3.70
TOTAL DUE	$

POSTAGE MAY VARY FOR OVERSEAS

Name ____________________
Street ____________________
City ____________ State ________ Zip ________