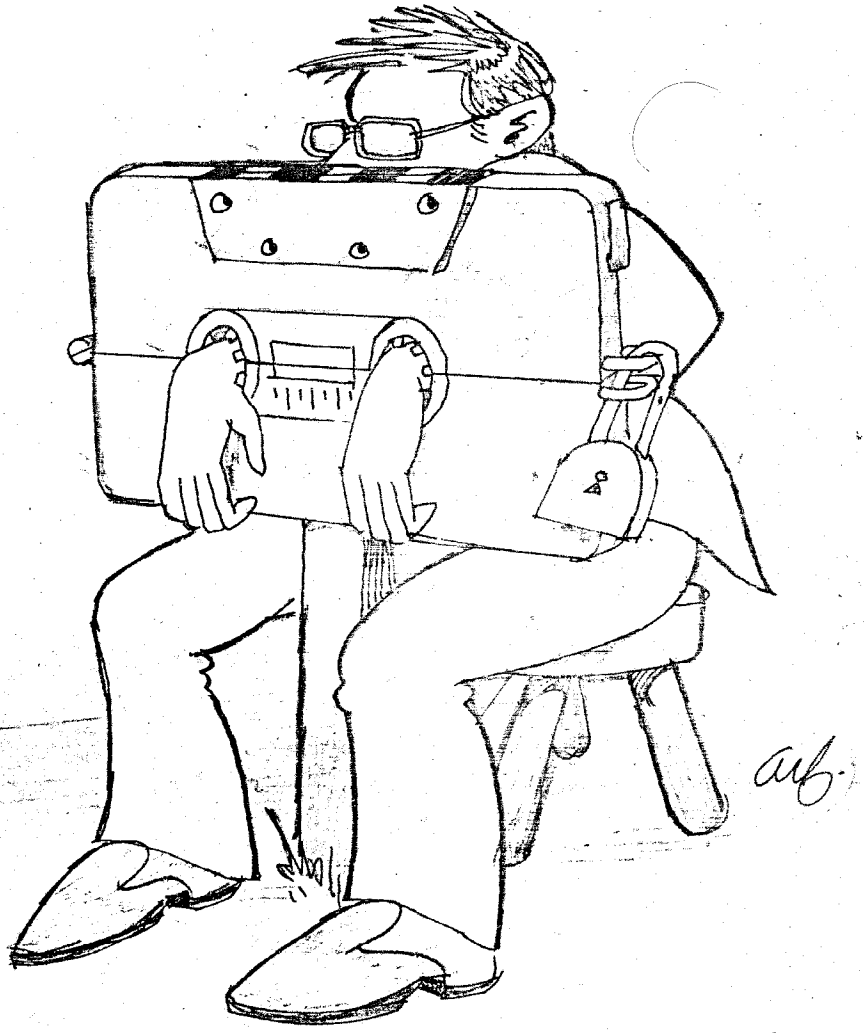
OSI-TEMS



the Challenger Journal

OSI-tems, Volume II, Number 8. September 4, 1980

OSI-tems, Volume II, Number 7. September 4, 1980

TABLE OF CONTENTS

OSI Users Group

Meets the first Thursday of every month at
Polk's Aristocraft
314 Fifth Ave.
New York, NY 10001
212-279-9034

David Gillet	President
Warren Modell	Vice President
Daniel Schwartz	Treasurer
Thomas Cheng	Secretary

Dues: $10 Standard Membership, $5 Student Membership

OSI-tems Staff

Larry Thaler
Thomas Cheng
Salomon Lederman
Michael Bassman
Terry Terrance
Mike Cohen

This month's Editor: Mike Cohen

EDITORIAL

This month's issue of OSI-tem's was done entirely on a C1P, and it wouldn't have been possible without WP6502, an excellent word processor available from Dwo Quong Fok Lok Sow for $75. Parts of the newsletter were actually printed with my BASE 2 printer.

Most of the programs in this month's OSI-tems can easily be adapted to run on ALL OSI systems, so this is no longer only a C1 newsletter!

CORRECTIONS TO OUR LAST ISSUE
Ugo V. Re
Levittown, N.Y.

In issue #7 page 19, lines 450 & 470 of the C1P changes to the Single Disk Drive program, should be as follows:

```
450 NUM=VAL(CHR$(PEEK(54119))
470 PAG$(PAG+1)=CHR$(PEEK(54122-PAG*32))
```

WHERE TO SEND ARTICLES FOR OSI-TEMS

Here are the editors for the next three issues of OSI-tems. Send articles to them or leave it at Polk's, but by all means send articles, no matter how trivial it may seem to you. What may not seem like much to you may be a big help to other club members.

OCTOBER:
Terry Terrance
Cedar Lane, Apt A-16
Ossining, NY 10562
914-762-3683

NOVEMBER:
Ugo Re
167 Sprucewood Drive
Levittown, NY 11756

DECEMBER:
Dan Schwartz
75 E. 190 st
Bronx, NY 10468

A QUICK AND DIRTY WAY TO SAVE MACHINE LANGUAGE PROGRAMS ON TAPE

by Dan Schwartz

There are a number of different formats in which a machine language program can be put on tape.

For example, there is the "auto load" format, which has the advantage that there is software in ROM that can read these tapes back.

Then, there is the "checksum" format, used by OSI for the extended monitor and the assembler and by Aardvark for their chess program. It's advantage is that it is possible to detect a bad load while one of these tapes is loading, due to the checksum recorded on the tape. The disadvantage is that you must first load in the checksum loader before you can load these tapes. OSI and Aardvark have the checksum loader followed immediately on their tapes by the actual program, but you can be sure that it took a lot of work to produce these tapes. Neither of the above two formats can be recorded on a tape without loading appropriate software routines to generate them.

Those of you who have seen the "Gomoku" tape, distributed by Orion Software, know that I used a third method to record this tape: rather than an ASCII-encoded format which gives the object code in the form of hexadecimal digits, the actual 8-bit bytes of object code are directly recorded on the tape, one after another. This raw binary data is preceded on the tape by a few lines of immediate-mode BASIC commands, which setup the loop parameters for inputting the data from tape to memory, and initiate the execution of this loop. While it is true that producing this tape, complete with the BASIC loader, entailed a reasonable amount of work, just saving the raw binary data on tape is extremely easy. It is the method I use when I am developing a program and want to save the current, unfinished version on tape so I can continue work on it some other time. I use it because it requires no program at all to generate the tape, really. You can do it with one line of immediate-mode BASIC. To save the contents of memory locations XX through YY on tape, just type in:

FOR X=XX TO YY: I=PEEK(X):WAIT 61440,2:POKE 61441,I:NEXT

(C2 & C4 users- use 64512 and 64513 instead of 61440 and 61441). Type that in, but don't hit carriage return yet. Rewind your tape to the beginning and put the tape recorder in record. When the leader has passed over the heads, hit <RETURN>. That's all there is to it - when you get the OK, the tape is done.

.... 04

What have you actually recorded on the tape? Just a raw memory dump, with no indication of how many bytes are there, or where they go in memory. You have to remember it - that's why I called this the "quick and dirty" method. But for your own use, for temporary storage over a period of days, rather than months, you can't beat it.

Now, how are you going to get that data back into memory when you come back tomorrow? You have to have a way of synchronizing the execution of a loop by your computer, which will get data from the tape and POKE it into memory, with the actual presence of that data at the playback head of your tape recorder. For that synchronization, you use another WAIT statement. The WAIT statement in the routine to save the data checked bit 1 in the 6850 status register, which is the "transmit data register empty" or TDRE signal. That just means that this bit goes to a logical "1" when the 6850 has finished transmitting a character to the tape. This bit is at a logical "0" only while a data byte is actually in the process of being sent out, one bit at a time, to the cassette circuitry.

Bit 0 of this same register is, the "receive data register full" or RDRF signal. It is a logical one when a byte has been received by the chip, and is reset to zero when the data byte is actually read from the data register. Well, almost always it is. There is one exception, and that is whatwe use to synchronize the tape. When the data is received by the serial port, it comes in LSB first, MSB last. As soon as the MSB is in, it is transferred to the "output data register". With Motorola's version of the 6850 at least, if you do a read of the data register right at this point, the status bit will not ber reset, at least if the MSB in the byte being read is a "1".

I'm not quite sure why this is, and I suspect it might not work with second-source versions of the 6850, but since all of OSI's cassette machines contain Motorola 6850's, that shouldn't be a problem. Anyway, to synchronize the inputting FOR-NEXT loop with the data on the tape and then read the tape into memory, you use the following line of BASIC:

```
WAIT 61441,128:FOR X=XX TO YY: WAIT 61440,1: I=PEEK(61441):
POKEX,I: NEXT
```

(again, of course, use 64512 and 64513 for machines other than the C1P).

.... 05

That's what's on the GOMOKU tape, but it will work just as well if you type it in from the keyboard. Just make sure that, when you hit the return key, your tape recorder is past the physical leader on the tape and is into the 5 seconds or so of "leader tone" which must precede the actual data. The initial WAIT statement will make the computer wait until the first byte of the object code has come in. Just about any op-code you might use to begin a program with is going to have bit 7 set, so the computer will immediately pass to executing the body of the loop, which checks the input status bit, gets a data byte when it's ready, pokes it into place, and continues. Because bit 7 was set on the first object code byte, the status bit will still be set the first time it is checked within the loop, so you won't miss the first byte. In the event that the first byte of recorded data does not have bit 7 set, the computer is going to keep waiting until it gets a byte that does have bit 7 set, and that is the byte that will go into location XX. You could change the initial WAIT statement to look for some other bit, but that introduces another problem, in that reading this byte (which the WAIT statement does) will reset the status bit, and location XX will then get the second byte on the tape, rather than the first, because the first WAIT statement within the loop is going to wait for the status bit to go high again, which it will only when the second byte on the tape is received.

Since most programs do start with an op-code above hex 80, though, you shouldn't run into this problem. If you do, it might be best to put a dummy byte with bit 7 set into memory at the location just before the first one you really want to save, and to save and load back the whole block of memory, including this dummy byte, which you can simply ignore once the tape is loaded. You can't use the status bit for the initial synchronization, because it is going to be set from having received noise from the leader tape.

BLOCK LETTERS

By Mike Cohen

Here's a convenient, but slow subroutine which will allow letters made up of graphics characters to be printed anywhere on the screen. Enter it with a GOSUB 5000. The string you want printed should be in X$ and T should point to the upper left corner of the first letter. It will run with no changes on all systems.

```
1 T=53481:IFPEEK(65506)THENT=53447
2 FORM=0TO32:PRINT:NEXT
3 T=T-3:H=T
10 X$="ABCDEFGH":GOSUB5000:T=H+5*L:H=T
20 X$="IJKLMNOP":GOSUB5000:T=H+5*L:H=T
30 X$="QRSTUVWX":GOSUB5000:T=H+5*L:H=T
40 X$="YZ123456":GOSUB5000:T=H+5*L:H=T
50 X$="7 8 9 0":GOSUB5000
999 END
1000 POKET,189:POKET+1,190
1020 POKET+L,136:POKET+L+1,143:POKET+2*L,210:POKET+2*L+1,207
1030 POKET+3*L,136:POKET+3*L+1,143:RETURN
1099 REM 'B'
1100 GOSUB1400:POKET+1,190:POKET+L+1,189
1110 POKET+2*L+1,190:POKET+3*L+1,189:RETURN
1199 REM 'C'
1200 GOSUB2400:FORM=1TO2:POKET+M*L+1,32:NEXTM:RETURN
1299 REM 'D'
1300 POKET,210:POKET+1,190:FORM=1TO2:POKET+M*L,136:NEXTM
1310 POKET+3*L,209:POKET+3*L+1,189:FORM=1TO2:POKET+M*L+1,143:NEXTM:
1320 RETURN
1399 REM 'E'
1400 GOSUB1500:POKET+3*L,209:POKET+3*L+1,128:RETURN
1499 REM 'F'
1500 POKET,210:POKET+1,135:POKET+L,209:POKET+2*L,210
1510 POKET+3*L,136:RETURN
1599 REM 'G'
1600 GOSUB2400:POKET+L+1,32:POKET+2*L+1,207:RETURN
1699 REM 'H'
1700 POKET,136:POKET+1,143:POKET+L,209:POKET+L+1,208
1710 POKET+2*L,210:POKET+2*L+1,207:POKET+3*L,136:POKET+3*L+1,143
1720 RETURN
1799 REM 'I'
1800 FORM=0TO3:POKET+M*L,143:POKET+M*L+1,136:NEXTM:RETURN
1899 REM 'J'
1900 FORM=0TO2:POKET+M*L+1,143:NEXTM:POKET+3*L+1,208
1910 POKET+3*L,209:POKET+2*L,136:RETURN
1999 REM 'K'
2000 FORM=0TO3:POKET+M*L-1,143:NEXTM:POKET+1,202:POKET+L,189
2010 POKET+2*L,190:POKET+3*L+1,200:RETURN
2099 REM 'L'
2100 FORM=0TO2:POKET+M*L,136:NEXT:POKET+3*L+1,128:POKET+3*L,209:RETURN
2199 REM 'M'
2200 FORM=0TO3:POKET+M*L-1,143:POKET+M*L+1,143:NEXTM
```

.... 07

```
2210 POKET,207:POKET+1,207:POKET+L,143:RETURN
2299 REM 'N'
2300 FORM=0TO3:POKET+M*L-1,143:POKET+M*L+2,136:NEXTM
2310 POKET,200:POKET+L,199:POKET+2*L+1,200:POKET+3*L+1,199:RETURN
2399 REM 'O'
2400 GOSUB3000:POKET,210:POKET+1,207:RETURN
2499 REM 'P'
2500 POKET,210:POKET+1,207:POKET+L,209:POKET+L+1,208
2510 FORM=2TO3:POKET+M*L,136:NEXT:RETURN
2599 REM 'Q'
2600 GOSUB2400:POKET+3*L+1,188:RETURN
2699 REM 'R'
2700 GOSUB2500:POKET+2*L+1,200:POKET+3*L+1,199:RETURN
2799 REM 'S'
2800 POKET,210:POKET+1,135:POKET+3*L+1,208:POKET+3*L,128
2810 POKET+L,209:POKET+L+1,128:POKET+2*L+1,143:RETURN
2899 REM 'T'
2900 GOSUB1800:POKET,207:POKET+1,210:RETURN
2999 REM 'U'
3000 POKET,136:POKET+1,143:POKET+3*L,209:POKET+3*L+1,208
3010 FORM=1TO2:POKET+M*L,136:POKET+M*L+1,143:NEXTM:RETURN
3099 REM 'V'
3100 GOSUB3000:POKET+3*L,190:POKET+3*L+1,189:RETURN
3199 REM 'W'
3200 FORM=0TO3:POKET+M*L-1,143:POKET+M*L+1,143:NEXTM
3210 POKET+3*L,208:POKET+3*L+1,208:RETURN
3299 REM 'X'
3300 POKET,200:POKET+1,201:POKET+L,199:POKET+L+1,202
3310 POKET+2*L,201:POKET+2*L+1,200:POKET+3*L,202:POKET+3*L+1,199:RETURN
3399 REM 'Y'
3400 GOSUB1800:POKET,200:POKET+L,199:POKET+1,201:POKET+L+1,202:RETURN
3499 REM 'Z'
3500 POKET,135:POKET+1,207:POKET+3*L,209:POKET+3*L+1,128
3510 POKET+L+1,189:POKET+2*L,189:RETURN
3599 REM '1'
3600 FORM=0TO3:POKET+M*L+1,136:NEXT:POKET,201:POKET+3*L,128
3610 POKET+3*L+1,209:RETURN
3699 REM '2'
3700 GOSUB3500:POKET,189:POKET+1,190:RETURN
3799 REM '3'
3800 POKET,189:POKET+1,190:POKET+L+1,208:POKET+2*L+1,207
3810 POKET+3*L,190:POKET+3*L+1,189:RETURN
3899 REM '4'
3900 FORM=0TO3:POKET+M*L+1,136:NEXTM:POKET+L+1,209
3910 POKET,136:POKET+L,209:RETURN
3999 REM '5'
4000 GOSUB2800:POKET+L+1,32:POKET+2*L+1,190:POKET+3*L+1,189:RETURN
4099 REM '6'
4100 GOSUB2800:POKET+2*L,136:POKET+3*L,209:RETURN
4199 REM '7'
4200 POKET,135:POKET+1,207:POKET+L+1,201:POKET+2*L+1,202
4210 POKET+3*L,201:RETURN
4299 REM '8'
4300 GOSUB4100:POKET+1,207:POKET+L+1,208:RETURN
4399 REM '9'
```

```
4400 GOSUB2800:POKET+1,207:POKET+L+1,208:RETURN
4499 REM '0'
4500 GOTO2400
4599 REM ' '
4600 RETURN
5000 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890":
5010 L=32:IFPEEK(65506)THENL=64
5020 FORX=1TOLEN(X$):Y$=MID$(X$,X,1):GOSUB5100:NEXTX:RETURN
5100 T=T+3
5110 FORI=1TOLEN(A$):IFY$=MID$(A$,I,1)THEN5130
5120 NEXTI
5130 ON I GOTO 1000,1100,1200,1300,1400,1500,1600,1700,1800,1900
5140 ON I-10 GOTO 2000,2100,2200,2300,2400,2500,2600,2700,2800,2900
5150 ON I-20  GOTO 3000,3100,3200,3300,3400,3500,3600,3700,3800,3900
5160 ON I-30 GOTO 4000,4100,4200,4300,4400,4500,4600
5170 RETURN
```

SORTING ARRAYS

By Dan Schwartz

An awful lot has been written about how to sort arrays. A recent issue of MICRO for example, had an article with three different ways of doing it, each of which took about a page of the magazine for a BASIC listing of the sort routine.

I don't know about you, but it seems to me that there is an easier way to do it. What I present here is a sort routine in 4 lines of BASIC. I haven't seen quite this procedure described in any of the literature, so I don't know if it has a name or not, but it works just fine.

Here, then, is a short demo program using this routine.

```
10 REM SORT DEMO
20 REM DAN SCHWARTZ 75 E 190 ST, BRONX NY 10468
30 REM FIRST SET UP AN UNSORTED ARRAY
40 DIM A(20):FOR X=0 TO 20:A(X)=RND(X):NEXT
50 REM NOW SORT IT
60 FOR N=1 TO 20:A=A(N)
70 FOR M=0 TO N-1:IFA(M)<A THEN NEXT M
80 FOR P=N TO M+1 STEP -1:A(P)=A(P-1):NEXTP
90 A(M)=A:NEXT N
100 REM NOW PRINT THE SORTED ARRAY
110 FOR X=0 TO 20: PRINT X,A(X):NEXT X
```

NEWS FROM OSI

Mike Bassman
39-65 52 st.
Woodside NY 11377

The most exciting thing this month is the C1P series 2. Many people have asked what the main differences are between the old C1P and the series 2. I will enumerate the important ones. First of all, the video display has been upgraded. Instead of the old 24X24 display, the series 2 has a 48X12 display which is software selectable to 24X24. In the 48x12 mode, horizontal lines are not contiguous, so all graphics must be done in the 24x24 mode. This assures software compatibility with the old C1P. A switch selectable printer/modem RS 232C serial port is now standard. A new color and sound board, officially the 630 board, will plug into the 40 pin expander socket of the series 2. This board will not be directly compatible with the old C1P. In effect, the C1P series 2 is a put together C4P.

Another interesting new product is the C4P DF. The DF stands for double floppy, as opposed to the old C4P MF (mini-floppy). The DF also means 8 inch drives as opposed to 5 1/4 drives. The C4P MF will still be produced. The C4P DF is virtually a C8P.

What is happening here is that the entire OSI line is moving up a notch, so to speak. The only low end machine left is the Superboard 2. The list prices on the new equipment is as follows:

C1P series 2 - $479 with 8K
C4P DF - $2799

Ohio Scientific is going to open a series of retail stores within Montgomery-Ward department stores. Emphasis will be put on the C1P series 2 for personal applications, and the C4P DF for small business.

Hayden Books, in cooperation with OSI will be producing SARGON, one of the best chess programs available, in an OSI version. A Hayden representative said that they intend to put out an entire line of software for the OSI in a matter of 3-4 months.

Prices for OSI equipment will be going up very shortly. The new pricing is supposed to be effective August 1st, but since the new price list has not arrived, equipment is still being sold at the old prices. The new price list is supposed to have prices on individual parts, so hopefully, OSI hobbyists can now buy things previously unavailable. Superboard owners could then get the official cases and power supplies, etc. The new prices are supposed to be compensated for by new features, such as having some of the connectors active on the back of the C4P cassette system.

Visicalc, the $150 program that sells $2000 Apples, will shortly be available on disk for OSI systems under the name 'Planner'. On the 5 1/4" version, it will be part of the MDMS system (mini data base management system).It will be acctually improved over the orisinal in that on-screen data can be stored on disk. Retail price will be around $100.

The neatest piece of software I have seen for the C4P MF and the C8P DF is Plot Basic. It is an addition to the regular disk basic that has 9 different graphics commands. Some of the features allow vertical and horizontal lines, plotting points at up to 128x64 resolution, setting an origin point, putting any character of any color anywhere on the screen, setting foreground and background color, defining shapes, putting them on disk, moving them non-destructively against the background. There are many other features, as well.

SOFTWARE REVIEWS

By Mike Cohen

Rated 0* to 7*

.....

ASTEROIDS from Dare Data & Design 5*

This one is surprisingly good. Nice action & good use of graphics, but a little too slow in the beginning. As you play it, though, it starts to speed up, and eventually you will run into an alien that keeps on firing at you until you destroy it.

.....

OSI HEXDOS From 6502 Program Exchange 6 1/2 *

A new disk operating system for the C1 which uses BASIC in ROM. Supports named files, which are created automatically when a program is saved. The entire operating system fits on track zero and includes an I/O distributer similar to OS65D and drivers for a real-time clock and tone generator on the 610 board. Has mid-line editing and several special keyboard functions. Disks can be easily formatted & copied with one drive.

.....

OSI FORTH From Technical Products co. 4 1/2 *

A true FORTH compiler based on the APPLE FORTH from PROGRAMMA. Any desired features which are lacking can be easily added and saved on disk. The compiled definitions are incredibly compact. In fact they recommend 16k as the minimum memory size for a disk system. FORTH code runs almost as fast as machine language. However, FORTH is a difficult language to learn, and their documentation doesn't help much.

At the last meeting, we discussed the PK80 PEAK READING VM which is sold by

Cook Laboratories
375 Ely Ave.
Norwalk, Ct. 06854

It retails for $49.50 complete with test tape and is well worth the money. However, for those of you with a tighter budget and a well stocked " Parts Box" there are alternatives...

First of all, check with the public library for a copy of the February 1980 issue of Popular Electronics - Specifically pages 85-88. There you will find an article written by Emory Cook about the PK80.

As I mentioned you have some choices;

I) You can build from a kit $25.90 (not including test tape- more about that later)

2) Purchase the circuit board for $2.50 & populate it yourself from your "Junque Box"

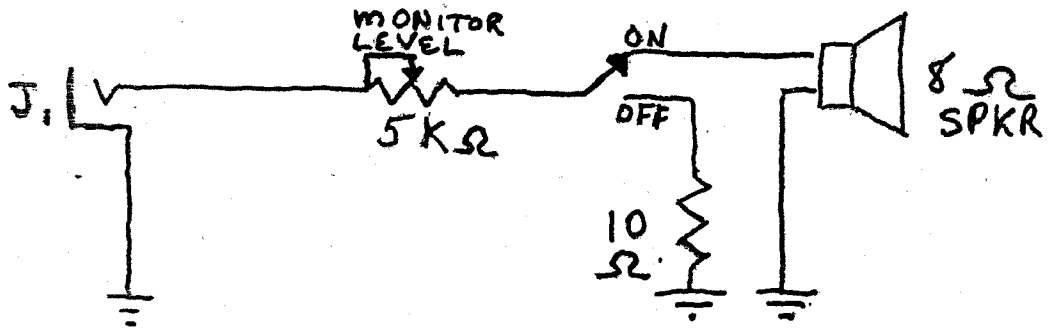
3) Use the foil patterns in the article to make your own P.C. Board and procede as above.

I did #3. I purchase a surplus meter movement from Poly Pak, a case and miniature plugs from Radio Shack. Total investment five dollars. The rest of the parts I had lying around.

I should point out that I did not purchase the test tape from Cook Labs. Instead, I made an assumption that turned out valid for me.

I assumed that OSI programs tapes were made on a Professional Recorder with correct Azimuth & used these for calibration. This worked out fine for me.

I also made a minor circuit addition (see below): I added a built-in speaker so as to listen to the data stream while a program is being loaded (my meter is always "in line") without having to remove the plug.



To sum it up :

I) Its easy to build: Either as a kit or from scratch.

2) Biggest saving if you Build from a well stocked parts box or if you frequent the surplus electronics shop in the city (total cost: $I0.00).

3) The Azimuth calibration tape will cut into your savings:I get by without it.

4) Any way you cut it, the meter is handy to have around-it makes reliable loads consistently possible.

I should also add that since building the PK80 I have designed & constructed my own version of the meter. Mine is a passive circuit (requiring NO power) which does away with the OP-AMPS. More about that on another occasion.

MILAGROS MONTALVO

Milagros Montalvo

Software Reviews

Mike Bassman
39-65 52 st.
Woodside, N.Y. 11377

This month has shown some surprising things. First of all, a lot of new software for Ohio Scientific computers, especially the Challenger 1P, has become available. Secondly, in general, the quality of said software is improving. Thirdly, more documentation can now be bought. There are disassembies of OS-65D, more manuals for the Superboard, explanations of Rom Basic, and more. Onwards to the reviews.

CHAT-the intelligent terminal. Charles Shartsis.

This program will turn your 4K Challenger 1P or Superboard into an intelligent communications device. It has all the options of a standard terminal: selection of parity, number of data bits, number of stop bits, and other transmission information. What makes it special are these features:

- Ability to get data from tape and transmit it.
-Ability to take incoming data and store it in memory for later permanent storage on tape.
- Can transmit a Break code.
- Lets the user redefine any or all of the keys on the keyboard, and save these changes on tape.
- Automatic backup facility.
- Totally menu driven.
- Extremely extensive and complete documentation.

List price for this program is $24.95. Available from:

Charles A. Shartsis
9308 Cherry Hill Rd. #812
College Park, MD 20740

....software reviews....continued

Space Invaders. Ohio Scientific.

Surprise! Following the line of trash previously put out by OSI is probably the best video game available. Done all in machine language, the game is smooth, fast, exciting, and bug-free. Resembles the arcade game exactly in play, and uses very clever graphics. Fifteen levels of play, and keeps a high score for each. Altogether an excellent program, certainly worth getting if you can afford it. List price is $29. Versions available for the C1, C2, and C4.

Hide and Seek. Ohio Scientific.

Oh well. Another version of the 'Black Box' game. You can always use the tape as a blank. List price is $6.

Air-Sea-Battle. Aardvark Technical Services.

A game in which you are a boat, and can fire at planes flying overhead or drop a depth charge for the sub below. Not spectacular, but not bad, either.

Awari. Aardvark Technical Services.

A very nice computerized version of the board game. In this game, the computer plays against you on any of 4 skill levels. A very interesting game, and habit forming. Worthwhile.

Orbital Fighter. Aardvark Technical Services.

In this game, you command the Enterprise and fight off the horde of aliens that has invaded your monitor. An extremely simple game, but it will keep you glued to your TV for hours.

Worms and Gobbler. Aardvark Technical Services.

An extremely simple minded game that would not keep the attention of the most retarded earthworm for more than 30 seconds. Qualifies for the rip-off of the year award.

XEROX MODE OSI

Thomas Cheng
26 Madison Street, Apt 4I
New York, NY 10038

For all of you tape users, here is a handy little tape utility. A copy utility which will take in anything(!) from a tape, the print it out again (rather, it will output it directly to the tape port). Play the tape as closely to the program listing as possible, then run the program. Starting from the first non-null character, the program will take in characters and poke them directly into memory. Meanwhile (back at the ranch...) these characters will be echoed to the screen. When the input is finished, press the space bar until the program prompts you (since the space bar is scanned after a character is taken in, rewind, cue, etc. until a character is detected and the spacebar scanned).

Line 100 sets the top byte of memory that will be used. If you have a C2 or C4, the locations holding the top byte of user RAM may be different, so either change the locations or set the top byte of RAM manually. (editors note: the locations are the same, since all OSI machines use the same ROM). Line 115 will cut out the initial nulls. Line 120 gets the program from the port, and lines 220-225 will output it again.

If you want to use this with a C4, change 61440 to 64512, and 61441 to 64513 (this is the location of the ACIA).

Admittedly, this program eats up a lot of memory. However if anyone cares to tackle the job of rewriting this in machine language with the feature of compressing machine language programs, I will gladly take a copy. (Machine code loading from tape takes 2 bytes for each byte of object code, plus added overhead for the checksum and byte locations, etc.)

Now, I can finally make backups for my machine language programs!

```
100 TM=(PEEK(133)+PEEK(134)*256)-1:TD=TM:FL=61440:PO=FL+1
110 POKE530,1:POKE57088,253:POKE11,0:POKE12,253
115 WAITFL,1:IFPEEK(PO)=0THEN115
120 WAITFL,1:CH=PEEK(PO):POKETM,CH:TM=TM-1:PRINTCHR$(CH);
190 IFPEEK(57088)<>239THEN120
200 PRINT:PRINT:PRINT"HIT C FOR A COPY":PRINT:PRINT:PRINT:PRINT
210 K=USR(K):IFPEEK(531)<>67THEN210
220 FORK=TDTOTMSTEP-1:CH=PEEK(K):WAITFL,1:POKEPO,CH
225 PRINTCHR$(CH);:NEXT
230 PRINT"HIT C FOR ANOTHER COPY"
240 PRINT"ANYTHING ELSE TO QUIT"
```