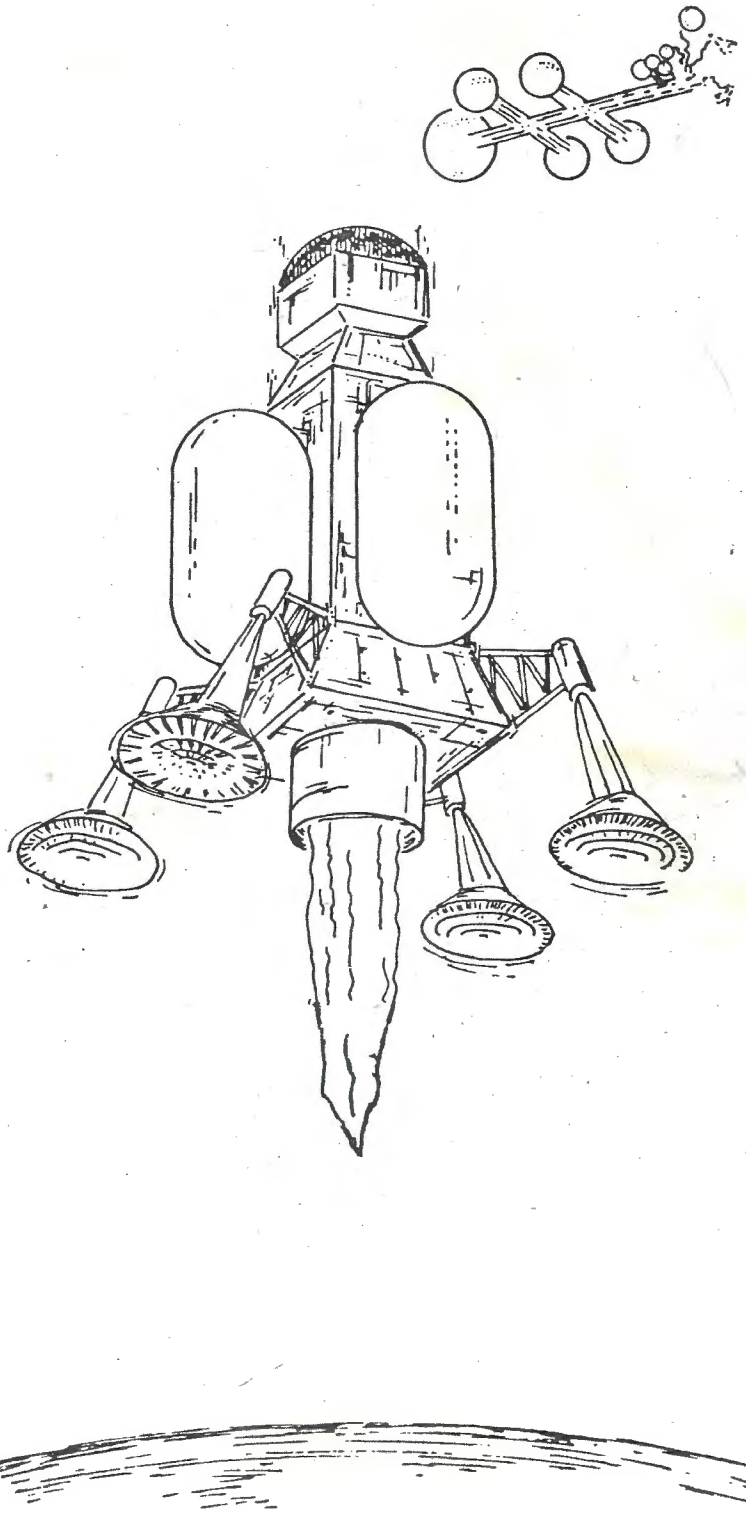
OSI·TEMS

Volume 2 No. 8 October '80



Volume II, Number 8 OSI-tems October 1980

TABLE OF CONTENTS

OSI USER'S GROUP

Meets first Thursday of every month at:

Polk's Hobbies
314 Fifth Ave.
New York, NY 10001

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

Dues: Standard Membership, $10.; Student Membership, $5

OSI-tems Staff

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

This month's editor: Terry Terrance

EDITORIAL

Recently copies of some other OSI newsletters have crossed my desk at Orion Software. All of these publications use phrases like, "the only," "the best," and "serving the needs of all OSI users" to describe themselves. Usually, on closer inspection, one finds that they are certainly not "the only" newsletter offering anything; nor are they, in comparison to OSI-tems, "the best;" and usually the only person that they serve is the one-man writer/editor/publisher (and usually software vendor) who puts them out. It utterly escapes me how the subscribers to these publications can sit back and enjoy them when they have such little substance, and much implicit in-house advertising.

We have in 'OSI-tems' a superior newsletter because we are 'members' and not 'subscribers.' As members I believe that most of us feel inclined to contribute something to the production of this journal. And this is a healthy attitude that should lead to 'OSI-tems' continuing to be a thriving, multifaceted, interesting and independent publication that can strive to 'serve all needs.' Witness the inclusion in this issue of our first article dealing with OS 65U.

If we lapse into being just 'subscribers' then OSI-tems will become like most other OSI newsletters; the organ for the few people who contribute, necessarily reflecting their limited talents and interests (and sometimes their commercial ventures).

So please get articles, programs and tidbits in. This month only 5 items were submitted to me. I considered filling the issue out with fluff and old programs that were written long ago but I decided against it. It should not be the editor's job to fill out the pages at any cost. With all of the talent and energy amoung our membership, it should be the editor's job to pick the best from amoungst the dozens of articles that should be submitted. OSI-tems should be a reflection of the care, concern and pride of the members. Otherwise we'll all become 'subscribers'

Here are the editors for the next two issues of OSI-tems. Send articles to them or leave them at Polk's.

November:

Ugo Re
167 Sprucewood Dr.
Levittown, NY 11756

December:

Dan Schwartz
75 E. 190 St.
Bronx, NY 10468

MISCELLANY

Some OSI newsletters that have come to my attention:

OHIO SCIENTIFIC USER'S INDEPENDENT NEWSLETTER
Published by: Charles Curley
6061 Lime Av. #2
Long Beach, CA 90805

Mr. Curley describes his newsletter as: "The Ohio Scientific User's Independent Newsletter (OSUIN) is edited, published, largely written and almost everything else by Charles Curley ..... Machines covered run from OSI Superboard up to and including C3's..... OS 65D and OS 65U..... BASIC and assembly are currently covered." Subscriptions are $10. for six issues.

This newsletter came to my attention because Mr. Curley is planning an OSI resources issue and inquired of Orion for our product line. The sample issue sent along to me (#5, Aug. 1980) is 8 pages long. It is a typical newsletter (as opposed to OSI-tems which is more like a magazine) with short announcements of software and hardware products, bugs and fixes submitted by readers, short "look what I found"-type of features and appeals from various readers for help. No complete programs or how-to articles were in this issue. OS 65D and OS 65U items predominate almost to the exclusion of items for ROM systems.

A disk-count (ha-ha) subscription program is available whereby bulk subscriptions submitted on an OS 65U disk are accepted at $8. apiece.

My (biased) opinion of this newsletter is that it is the typical one-man publication and somewhat overpriced to boot. However, it did have a few interesting items, and, considering the relative dirth of OSI information sources, any new source is welcome.

BACK TO BASIC
#43 Cross Keys Shopping Center
Florissant, Missouri 63033

Back to Basic is an OSI dealer and computer store which has a users group and newsletter. They seem to have much the same type of set-up as we do. However, I do not have a sample of their publication and they seem to be a little 'tight-mouthed' about giving out information. They suggest that our members contact them directly.

Here's an interesting item in the way of a dial-up service:

Gamemaster
P.O. Box 1483
Evanston, Ill. 60204
24 Hour Hotline 312-236-7522
Dial-Up line 312-726-8260

The October issue of 'Popular Electronics' carried a few paragraphs on the Gamemaster dial-up service. Supposedly, "Gamemaster is a unique communication

system with a number of games, mail, technical data and so on."

What attracted my attention was the free sign-on time that is offered on the first friday of the month from 10 am to 10 pm. This was interesting in that you could try out the system before paying the sign-on fee, which for Gamemaster is $50. with a $2.75/hr. connect charge.

To get more information, I have tried to call their '24 Hour Hotline' number several times a day for the past two weeks with no answer. I did, however, get a modem tone when I called this number on a Saturday; but that doesn't help much when one is expecting to get information from a real person. There was always a modem tone on the dial-up number whenever I called, so apparently the service is real, even if their 24 hour number is not.

So, since most of you will get this before the first friday in October, maybe some enterprising member with a modem set-up might try to connect during the free time and then write something up for the next OSI-tems.

SOURCE UPDATE

Mike Bassman
39-65 52 St
Woodside, NY 11377

This month has seen a lot of new products for OSI machines. Here are some of the newer ones . If you are a new reader, take note: Source Update is a constantly updated list of new hardware and software for the OSI Challenger series. If you know of any hardware or software not mentioned here, please let me know. For a total list of companies, simply gather together all the previous 'Source Updates.'

JDS Software
2334 Antigua Court
Reston, VA 22091

At present selling only a very complete manual for the Superboard II/ Challenger IP. Covers BASIC in detail and has a nice graphics supplement.

Earthship
Box 489
Sussex, NJ 07461

Selling software for the Challenger line. They have a really good lunar lander and other programs yet to be tested.

Grafix
911 Columbia Avenue
North Bergen, NJ 07047

These people have an incredible high resolution color graphics board for the Superboard II/ Challenger IP. Totally seperate video display system, so you could run two monitors. Maximum of 256x192 display points (selectable in several levels up to that point--ed.). Includes RF modulator, has room for extra RAM expansion.

Software Consultants
7053 Rose Trail
Memphis, TN 38134

Disassembled and commented source listing of OS 65D. A real plus for disk users!

Software Alchemists
4303 North Charles St.
Baltimore, MD 21218

These people have a very nice series of games and utilities for the C1P. So far I have seen only one of their games, but it has been excellent. They are advertising a package for BASIC editing. Write for their catalog.

SOFTWARE REVIEW

By Mike Cohen

A/65 Assembler.....From Pegasus software

This is an excellent new assembler for all OSI 8" disk systems. It has several unique features including the ability to link several files together and place object code directly on disk. It has several additional directives, including .PAGE which specifies a heading , .OPT which gives a series of listing options, and .FILE which tells the name of the next file to be loaded and linked to the previous text.

By specifying options LIST and NOLIST, you can selectively list only portions of the source code. You can also print the symbol table, but since it is unsorted, the symbol table is nearly useless. When it is invoked by typeing A/65 in the kernel, you are asked if you want a listing of the source code or the symbol table and if object code should be generated, either to memory or to disk. When you specify a filename and the options it will immediately start assembling. Any options specified in the source file as .OPT directives will override the options specified in the opening dialogue. A/65 assembles much faster than OSI's assembler and can print a listing and generate code in one operation.

A/65 does have some bugs, such as the symbol table, and a few peculiarities that take some getting used to. It has no editor, so you must use OSI's assembler or WP-2 to produce the source code. One annoying feature is that it doesn't print the line numbers in the source code, making it difficult to find errors. Also, it doesn't accept colons, periods, or dollar signs in labels, all of which are allowed in OSI's assembler. If it comes across any of these, it will quit assembling. In spite of these things, it is an excellent program that could be a real life-saver, especially if you have limited memory or extremely large programs to assemble.

(Another) Print At

Salomon Lederman

Several months ago I presented a machine language routine that would print a message at any place on the screen. After speaking to several interested persons I realized that there were some hardships with that routine. So here's version 2 of that routine.

As written, this routine will find T$ and put it at a screen location. The screen address is put into the routine in the following way. Determine the first location on the screen, the place where your message id to begin. Convert the address to HEX. Put the lo part of the address in location $0245 or 581 decimal. The hi part of the address goes into $0249 which is 585 decimal. The hi byte is the one which begins with $D1, $D2, $D3 ...

If you would like to change T$ to some other string, to print another string variable on the string, you do the following: POKE 547,K where K is the ASCII value of a one letter character. For example T has the value 84 so that's what is location 547.

The routine fits into the unused portion of page two. Just set up the user vectors to POKE 11,34: POKE 12,2 and you're all set to go. X=USR(X) is the way to get the routine to go. Happy printing.

PRINT T$ AT

```
0222  A9 54
0224  85 93
0226  A9 80
0228  85 94
022A  20 53 AD
022D  A0 00
022F  B1 95
0231  85 EA
0233  A2 00
0235  E6 95
0237  D0 02
0239  E6 96
023B  B1 95
023D  95 E8
023F  E8
0240  E0 02
0242  D0 F1
0244  A9 C9
0246  85 EB
0248  A9 D1
024A  85 EC
024C  A2 00
024E  B1 E8
0250  91 EB
0252  E8
0253  E4 EA
0255  D0 01
0257  60
0258  E6 E8
025A  D0 02
025C  E6 E9
025E  E6 EB
0260  D0 02
0262  E6 EC
0264  4C 4E 02
```

Extending OSI BASIC-IN-ROM

The OSI BASIC-IN-ROM has one very important subroutine located on page zero of memory which allows the programmer to control its actions. This gives us the chance to add our own subroutines by temporarily jumping from Basic and returning back without Basic knowing we were ever there. The subroutine is used by Basic to read code character by character. To get an idea of the purpose of this subroutine, see the artical (OSI-tems, V.II, No.7) by Michael Bassman called "Understanding Basic, Part I - The routine at ØØBC". Since Basic has to jump to this subroutine to read Basic code, we can make our extension here by changing this RAM routine and adding the following jump:

Location	Machine Langusge	Mnemonics
ØØBC	4C ØØ 1F	JMP $1FØØ
ØØBF	EA EA EA	NOP

Now when Basic jumps to this parsing subroutine, it will make an additional jump to the programmers control. But before we can add our own extension we must replace the code we wrote over so Basic will not miss it. The code is then replaced as follows:

Location	Machine Language	Mnemonics
1FØØ	E6 C3	INC $C3
1FØ2	DØ Ø2	BNE
1FØ4	E6 C4	INC $C4
1FØ6	AØ ØØ	LDY #$ØØ
1FØ8	B1 C3	LDA $C3,Y

Notice that we must also add the LDA instruction so that we know what we are reading. As you can see, this code is somewhat different since we are no longer working in page zero.

Now that we are in control of reading the Basic code we must have something to look for, and what we are looking for is our extended Basic instruction. In other words, if what we have just read (which is now stored in the accumulator) is our extended Basic instruction, then we can branch to our subroutine which will proform that operation. But first we must devise some kind of instruction to read, and that is not as easy as one might think.

In an artical which is in the June 80' issue of Micro magazine (page 25:15) by Edward H. Carlson, a form of extended Basic is presented. As the author explains, he had to devise a character swap so that his instruction would not execute as Basic read the code for the first time. (That is, Basic will read a line of code twice. The first time the code is read is just after the carriage return when the code is entered from the keyboard. This is done to change the " 5Ø 52 49 4E 54 " for "PRINT" to its token, which is

simply " 97 ". The second time is for the execution of the code, either in immediate mode or in the RUN mode. I am not sure that this is all true, but things seem to work that way. I hope maybe someone could give a better explaination sometime.) I found this trick somewhat cumbersome, in fact to much so, and had to take a new approach.

Finally it came to me. If Basic did first read a line of Basic code after the carraige return, it would be impossible to read any Basic tokens. So if we were to look for tokens in the accumulator, there would be none during the input and the extended routine would not execute. This then means that we do not have to preform any character swap because Basic is doing it for us.

Now the "%C" for a machine language screen clear can be changed to "PRINTCLEAR" using two Basic tokens. In the program, I have included two extensions. The first is, of course, the machine language screen clear. The example is as follows:

```
PRINTCLEAR      (in immediate mode); or

10 PRINTCLEAR   (in a program)
```

The second extension is the "PRINT AT" instruction, but instead of an "AT" we are going to use an "ON". The example is as follows:

```
PRINTON addr "Message"          ; or

10 PRINTON addr "Message"
```

The message will start at the named address, which must be in Hex, and will end at the second quote. To write this routine, I had to borrow parts of two routines that were found in the June issue of OSI-tems. (I hope the authors do not mind me using there programs here, but I could not have done it without them.)

I hope that some users out there can make other useful extended routines to add to our Basic in Rom. It would be a nice idea for future issues.

Program listing

Location	Machine language	Mnemonics
1FØØ	E6 C3	INC $C3
1FØ2	DØ Ø2	BNE $1FØ6
1FØ4	E6 C4	INC $C4
1FØ6	AØ ØØ	LDY #$ØØ
1FØ8	B1 C3	LDA $C3,Y
1FØA	C9 97	CMP #$97
1FØC	FØ Ø3	BEQ $1F11
1FØE	4C C5 ØØ	JMP $ØØC5
1F11	E6 C3	INC $C3
1F13	DØ Ø2	BNE $1F17
1F15	E6 C4	INC $C4

Location	Machine Language	Mnemonics
1F17	B1 C3	LDA $C3,Y
1F19	C9 9A	CMP #$9A
1F1B	DØ Ø3	BNE $1F2Ø
1F1D	4C 3Ø 1F	JMP $1F3Ø
1F2Ø	C9 9Ø	CMP #$9Ø
1F22	DØ Ø3	BNE $1F27
1F24	4C 4D 1F	JMP $1F4D
1F27	C6 C3	DEC $C3
1F29	1Ø Ø2	BPL $1F2D
1F2B	C6 C4	DEC $C4
1F2D	4C C2 ØØ	JMP $ØØC2
1F3Ø	A2 ØØ	LDX #$ØØ
1F32	86 FE	STX $FE
1F34	A9 DØ	LDA #$DØ
1F36	85 FF	STA $FF
1F38	AØ ØØ	LDY #$ØØ
1F3A	A9 2Ø	LDA #$2Ø
1F3C	91 FE	STA $FE,Y
1F3E	C8	INY
1F3F	CØ ØØ	CPY #$ØØ
1F41	DØ F9	BNE $1F3C
1F43	E6 FF	INC $FF
1F45	E8	INX
1F46	EØ Ø8	CPX #$Ø8
1F48	DØ F2	BNE $1F3C
1F4A	4C BC ØØ	JMP $ØØBC
1F4D	A2 ØØ	LDX #$ØØ
1F4F	2Ø BC ØØ	JSR $ØØBC
1F52	C9 3A	CMP #$3A
1F54	1Ø Ø5	BPL $1F5B
1F56	29 ØF	AND #$ØF
1F58	4C 5E 1F	JMP $1F5E
1F5B	38	SEC
1F5C	E9 37	SBC #$37
1F5E	9D FC 1F	STA $1FFC,X
1F61	E8	INX
1F62	EØ Ø4	CPX #$Ø4
1F64	DØ E9	BNE $1F4F
1F66	AD FC 1F	LDA $1FFC
1F69	ØA	ASL A
1F6A	ØA	ASL A
1F6B	ØA	ASL A
1F6C	ØA	ASL A
1F6D	18	CLC
1F6E	6D FD 1F	ADC $1FFD
1F71	8D 92 1F	STA $1F92
1F74	AD FE 1F	LDA $1FFE
1F77	ØA	ASL A
1F78	ØA	ASL A
1F79	ØA	ASL A
1F7A	ØA	ASL A

Location	Machine Language	Mnemonics
1F7B	18	CLC
1F7C	6D FF 1F	ADC $1FFF
1F7F	8D 91 1F	STA $1F91
1F82	2Ø BC ØØ	JSR $ØØBC
1F85	A9 ØØ	LDA #$ØØ
1F87	85 CC	STA $CC
1F89	2Ø BC ØØ	JSR $ØØBC
1F8C	C9 22	CMP #$22
1F8E	FØ Ø9	BEQ $1F99
1F9Ø	8D FF FF	STA
1F93	EE 91 1F	INC $1F91
1F96	4C 89 1F	JMP $1F89
1F99	A9 EF	LDA #$EF
1F9B	85 CC	STA $CC
1F9D	4C BC ØØ	JMP $ØØBC

Note: Since I did not use page two, you must save the top of memory for the program. For my 8K system I must answer 7936 to "MEMORY SIZE? " when starting.

Peter Schreiber
1609 Washington Avenue
Seaford, N.Y. 11783

MEMORY LOCATIONS CONTAINING THINGS OF INTEREST IN OS 65 U

000B,C	Address of USR routine
000D	Number of extra nulls to be inserted after the carrige return
000E	Number of characters since last carrige return
000F	Terminal width for auto CRLF
0010	Terminal width for comma spaced columns
0013-5A	Input buffer
005F	String variable being processed flag (?)
0061	?
0064	CTRL O flag (hi bit on= surpress printing)
0065	Sometimes contains $68 (??)
0079,7A	Pointer to initial null of BASIC program workspace
007B,7C	Pointer to beginning of BASIC variable storage space
007D,7E	Pointer to beginning of BASIC array storage space
007F,80	Pointer to end of array space/beginning of free memory
0081,82	Pointer to end of string space/top of free memory
0085,86	Pointer to top of memory allowed to be used by BASIC
0087,88	Current line number
0089,8A	Sometimes next line number (?)
008F,90	DATA pointer
0095,96	This is where ADOB leaves address of variable it found
0097,98	Address of variable to be assigned value by OUTVAR (AFC1)
00AA,AB	Points to pointer of next BASIC line after LIST
00AD,AE	The contents of this pair is printed in decimal by $B962
00AE,AF	This is where INVAR ($AE05) leaves its argument
00D1-D7	Clobbered by OSI disassembler in Extended Monitor; kills BASIC
00E0-E6	Apparently unused page zero space
00E8-FF	Apparently unused (by BASIC) page zero space
00FB	ROM monitor load flag
00FC	ROM monitor contents of current memory location
00FE,FF	Address of current ROM monitor memory location
0130	NMI routine
01C0	IRQ routine (can be overwritten by stack being used by BASIC)
0200	Current screen cursor is at D700+(0200); initialized to (FFE0)
0201	Save character to be printed
0202	Temp storage used by CRT driver
0203	LOAD flag ($80= LOAD from tape)
0205	SAVE flag (0= not SAVE mode)
0206	Time delay for slowing down CRT driver
0207-E	Variable execution block-code for screen scroll-not reusable
0212	CTRL C flag (not 0 = ignore CTRL C) (reset by RUN)
0213-16	Polled keyboard temporary storage and counter
A000-37	BASIC initial work jump table (in token order; add 1 to ea. addr)
A038-65	BASIC non-initial word jumps (real entry addresses)
A084-163	BASIC keywords in ASCII; hi bit set as delimiter; in token order
A164-86	Error messages with null delimiter
BE4E	"Written by" message

This table was complied by Wally Kendall, courtesy of OSIO. Submitted to us by Don Valentine and Mike Cohen.

Possibly some of the addresses are the same in OS 65D and ROM BASIC

INSIDE OS 65D's 'KERNEL'

Terry Terrance
119 S. Highland Ave.
Ossining, NY 10562

This whole article may be superfluous now that a commented, disassembled listing of OS 65D is available; however, here it is anyway.

When a command line is sent off to OS 65D from BASIC, the Extended Monitor, or the Assembler it is loaded into the command buffer located from $2E1E to $2E2F (11806 - 11823 dec.). So the maximum command line is 18 bytes long (including the carriage return).

There is a table located from $2E30 to $2E78 (11824-11896 dec.), disassembly of which shows each entry to be four bytes long. The first two bytes of each entry are the first two characters of the command's directive, such as CA for CALL. The next two bytes are an address, in high byte, lo byte format, pointing to one less than the address where the code for that instruction lies.

I believe that the command processing code lies from $2A84 to $2ABF (10884 to 10943 dec.). What it does is match the first two characters from the command buffer against the first two bytes of the table entries. When a match is found the next two bytes from the table are pushed onto the stack, fourth byte first. Then a RTS is executed which causes a jump to one beyond the address which was just put on the stack.

If all of this is correct, then some exciting possibilities open up for creating 'customized' versions of the operating system. Some of the less frequently used commands could be changed or overlaid to create new commands. Also, by suitably modifying the command decoder it may be possible to make a command line accept two directives (although I doubt this).

The command table looks like this:

ASCII of first two bytes	Address pointed to by next two bytes
HO	$ 2663
D9	2AC0
AS	2ADE
BA	2AE6
CA	2B11
DI	2B29
EM	2B2F
GO	2B46
IN	2B55
IO	2B83
LO	2BA7
ME	2BC6
PU	2BDD
RE	2BFD
XQ	2C22
SA	2C28
SE	2C43