Coded Data For OSI1P

Charles Stewart

I have had my OSI1P for several months now and have a number of word game programs including a version of Hangman which I utilize for my children's spelling words. The OSI has a nice feature of listing on the screen the program as it loads from cassette. This is fine for checking proper program loading and recorder levels etc., but at times a method of hiding information would be useful particularly in word games where you don't want the player to have access to the word list.

In search of a solution to this problem I have written a program which hides the information in DATA statements in ASCII code and writes DATA statements so that a file of words can be generated and inputed to the host program such as Hangman. The program to convert raw data to ASCII follows:

```
3 REMSET G$ TO DATA DIM VAR AS LARGE
AS MEMORY PERMITTS
5 G$ = "DATA":X = 1:DIMA$(50),X(50),Y(50):
7 INPUT"DATA LINE TO START";DA
8 INPUT"INCREMENT BY";IN
10 REM TO END TYPE '*' TO QUESTION
'WORD TO HIDE'
20 INPUT"WORD TO HIDE";A$(X):IFA$(X) =
"*"THEN50
40 X =X + 1:GOTO20
50 PRINT"SAVE CODED WORDS":INPUT
"RECORDER READY";B$:SAVE
85 FORX = 1TO10:PRINT:NEXT:X = 1
95 IFA$(X) = "*"THEN140
97 PRINTDA;G$;:FORW = 1TOLEN(A$(X)):
H$ = MID$(A$(X),W,1)
112 PRINTASC(H$);:NEXT
115 PRINT:DA = DA + IN:X = X + 1:GOTO95
140 PRINTDA + IN;G$;"-1":POKE517,0
150 STOP
```

Listing No. 1

How it works:

Line 5 G$ is set to 'DATA', var X set to 1, DIM var to the number of words you want to hide, I used 50 in my example. Line 7 sets the starting point of the generated data statements and should reflect free line numbers preferably at the end of your host program, line 8 is the increment value. Line 10 ends the input portion when a '*' is inputed to the question• "Word to hide" and moves to the output section in line 50. Line 40 increments x by 1 and starts the loop over again.

Lines 50-85 place the computer in the save mode, reset X to 1 and check for the end flag. Line 97 prints the line number selected in line 7 and prints G$ and the coded information, i.e. W is set to the length of the coded word. H$ is set to the letter in A$ for each increment of the for next loop. Line 112 prints the two letter ASCII code and returns for the next letter in A$. Line 115 increments the DATA statement by the number selected in line 8, X is incremented by 1 and loops to line 95 where end flag is checked. When the last word is coded the end flag is set and line 140 is executed giving an end of DATA flag for the decoding program.

The resulting data may be stored on tape and inputed to your word game or any program you may wish to hide data in. A file of ASCII data may be set up allowing children's spelling words, etc. to be inputed to the host program. The host program must also contain the following decoder program.

```
300 DIMX(50):RESTORE:X = 1:DIMA$(50):DIMJ
(50),R$(50),Y(50)
305 REM READ CODED WORDS CHECK FOR
END FLAG
310 READA$(X):IFA$(X) = "-1"THEN430
320 REM GET ASCII CODED DATA
330 FORJ = 1TOLEN(A$(X))STEP4:B$ = MID$(A$
(X),J,1) + MID$(A$(X),J + 1,1)
350 REM CONVERT DATA TO RAW ASCII
360 R = VAL(B$):R$(J) = CHR$(R):NEXTJ
400 REM ADD $ TOGETHER TO RETEVE
WORD
410 A$(X) = R$(1) + R$(5) + R$(9) + R$(13) + R$(17)
 + R$(21) + (21) + R$(25) + R$(29) + R$(33)
411 A$(X) = A$(X) + R$(37) + R$(41) + R$(45)
413 FORY51TO50STEP4:R$(Y) = " ":NEXT
415 REM PRINT WORD LIST
417 PRINTA$(X)
420 X = X + 1:GOTO310
430 STOP
```

Listing No. 2

How it works:

Line 300 DIM VAR to the maximum number of inputs required by the host program operation, restores the data pointer and sets X to 1. The coded data is read in line 310 and checked for end of data flag. Line 330 retrieves the ASCII code for each letter, i.e. sets B$ to the two character code representing one letter (89 = the letter 'Y') for the length of the data line. Line 300 converts the number code to the letter and line 410 retrieves the hidden word. Line 413 erases R$, (utilized in 410) in preparation for the next word. Line 417 prints the word list, shown here for example only. Line 420 increments X and returns to the beginning of the loop.

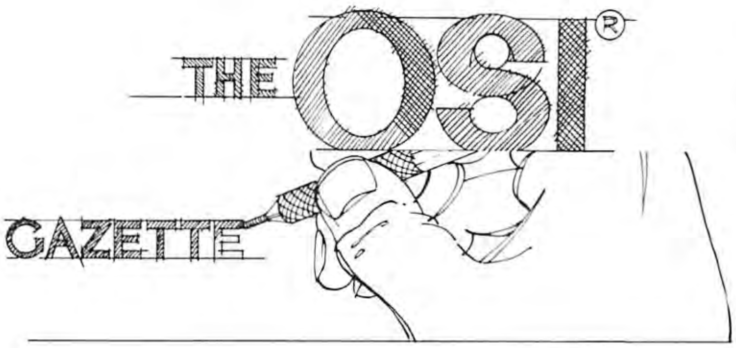
When the program is run, A$(x) is set to the hidden words, there are various methods utilized in games to randomly select the word used but with this method you can just increment X by 1 each time the user wants to play again thus eleminating the possibility of the selection of the same word twice.

```
1000 DATA 67 79 77 80 85 84 69 3 2 73 73
1010 DATA 80 46 79 46 32 66 79 8 8 32 53 52 48 54
1020 DATA 71 82 69 69 78 83 66 7 9 82 79
1030 DATA 78 46 67 46 32 50 55 5 2 48 51
1050 DATA-1
```

Figure 1: Example of coded data statements generated by listing No. 1

```
OK
RUN300
COMPUTE II
P.O. BOX 5406
GREENSBORO
N.C. 27403
BREAK IN 430
OK
```

Figure 2. Decoding of data above by program listing 2 ©



O.S.I. Graphics Character Set

W. Blaine Garland

We have had our OSI Superboard II (with 8K memory) for six months now, and with its help, my sons and I are continuously discovering something new in the world of 6502 Single Board Computing.

Aside from the terrific low cost, one of the main reasons we chose the Superboard II was its extensive graphics capabilities. However, in attempting to demonstrate to friends all the possibilities of the graphics character set, we were severely limited by leafing through "The Challenger Character Graphics Reference Manual". The table lists all 256 characters. We then tried in vain to explain to our friends how they are called to video display.

Recently, we developed a short, BASIC Demonstration Program which calls each character in the CG-4 ROM to the screen consecutively. The characters are identified by the numeric variable Y, displayed with the typical POKE statement in line 110 and incremented on the screen with the FOR, NEXT loop at lines 80 and 170. The remainder of the program is essentially "window dressing".

Running the Demo Program lists each memory location in decimal and each character in the CG-4 ROM Character generator. It also indicates the two "spaces" in the set (locations 32 and 96) and the beginning and ending of the upper and lower case alphabet, numerals and punctuation which is the standard ASCII character set (locations 33 and 123 respectively). You can change the speed for incrementing characters on the screen by revising the FOR, NEXT timing loop in line 150. With the loop set at 1 to 500, the 256 character set can be displayed in about three minutes. This must approach the maximum attention span of any non-computer addict! You may also want to revise the program by deleting the ASCII characters and displaying only the 165 graphics and gaming symbols or vice-versa.

Incidentally, in developing the program, we had first tried calling the characters to the screen using the PRINT CHR$ (X) function. For some reason locations 10 and 13 would only print a blank space on the screen. We would appreciate another Challenger User's insight into the error of our ways.

To many readers, this program may seem simplistic. But to those of us who possess little experience and even less OSI documentation, it represents yet one more major step toward mastering the Superboard II and 6502 singleboard computing through "experimentation"!

So the next time a friend wants to know more about your micro-computer's graphics capabilities, demonstrate to them - **GRAPHICALLY**!

Program Listing
OSI Graphics Characters

```
10 FOR X = 1 TO 29 : PRINT : NEXT
20 FOR X = 54119 TO 54215 : POKE X, 32 : NEXT
30 PRINT " OSI GRAPHICS CHARACTERS" : PRINT
40 PRINT " IN CG-4 CHARACTER" : PRINT
50 PRINT " GENERATOR ROM" : PRINT: PRINT
60 FOR T = 1 TO 500 : NEXT
70 I=0
80 FOR Y = 1TO 255
90 I = I + 1
100 PRINT TAB(2) I TAB(7) CHR$(45)
110 POKE 54095, Y
120 IF Y = 32 THEN PRINT TAB (12) "(SPACE)": PRINT:
    PRINT TAB (12) "ASCII BEGINS"
140 IF Y = 96 THEN PRINT TAB (12) "(SPACE)"
150 IF Y= 123 THEN PRINT TAB (12) "ASCII ENDS"
160 FOR T = 1 TO 500: NEXT
170 PRINT
180 NEXT Y
190 END
```

©

Atari Joysticks on the OSI C1P

Charles L. Stanford

One of the great advantages of the Ohio Scientific Challenger 1P and Superboard II computers is the easy yet effective graphics programming. The game symbols in the Character Generator ROM, plus the relatively simple 'POKE' programming of the screen refresh memory, opens many possibilities for games and other graphics simulations. The biggest disadvantage is the need to play through the keyboard in order to move the characters around the screen in a Gunfight, Tank, or Spacewar game.

This article will provide both construction and programming details plus a short keyboard input tutorial for interfacing the Atari joysticks to the C1P. In addition, almost every detail is identical for the C2, C4, etc. This joystick was chosen for several reasons, but primarily because Sears, Roebuck carries them at most retail stores for $9.99 each. They are also quite reliable and provide an easily interfaced digital output rather than the analog signal of most other such devices. The article will also include a generalized program for using the joystick with your own games, as well as the program patches (modifications) needed to convert the Space Invader game included on OSI's sample cassette SCX-102 to joystick play.

The Polled Keyboard

First, a little background on the OSI 600 board's polled keyboard. Unlike most other computer input keyboards, it does not convert a key actuation directly into the appropriate ASCII code in hardware. Rather, an input-output port at the address $DF00 (#57088), connected to an X-Y matrix keyboard, allows key closures to be detected and translated in software. (Note: In this article, binary numbers will be prefixed '%', Hex numbers '$', and decimal numbers '#'.) Each row of keys can be set 'On' or 'Off' by poking a binary number to the port address. For example, the instruction POKE 57099,127 actually sets the port input latches to the binary number %0111 1111. Thus rows 0 through 6 (starting with 0 on the right) are held at '1' and row 7 is at '0'. The computer program, whether in the ROM monitor or during a game in BASIC, then watches the port. If the key '7' is pressed, the program will see a %1111 1101 at the port. Key '1' would provide %0111 1111. More than one key being pressed simultaneously can also be detected. Pressing '1' and '3' would return the code %0101 1111. Each of these binary numbers would, of course, be translated automatically by the basic assembler into their equivalents in Decimal. Keys '1' and '3' being pressed at once would thus return a #95. The serious programmer should either develop a facility in converting numbers from binary to hex and decimal at will, or keep a conversion chart handy. Refer to your Graphics Reference Manual, Figure 3-2, for more details on the 600 Board polled keyboard's physical layout and electrical connections.

While you are programming the computer in BASIC, or if a running program asks for an INPUT, the monitor in ROM scans the keyboard constantly until a key closure is detected. Then the row and column are compared, and the appropriate ASCII code is returned to BASIC and to program storage. You can do the same thing in BASIC during a game by following the instructions in the Graphics Reference Manual for POKE'ing and PEEK'ing the keyboard as described above. But even better, you can connect a joystick directly to the keyboard matrix and simulate keyboard input through the motion of a control lever or pushbutton switch.

Keyboard Input Access Connections

Jack J4, located on the left front of the 600 Board, includes connections to seven columns and to rows 1, 6, and 7. This gives access to twenty keys, which is more than sufficient for even the most complicated game setup. This jack takes a 12-pin standard Molex connector, which can then be connected through a cable to any other common multi-pin socket, or directly to the game device. The C1P owner will probably prefer a more sophisticated socket arrangement attached to the outer case, while the Superboard II requires only the Molex jack.

A connector series which has become increasingly popular in recent years, generally known as the 'D', 'DB', or RS-232 types, has been chosen by Atari for their various interface plugs and sockets. These connectors come in various configurations, and can have 9, 15, 25, 37, or 50 pins. Atari uses the DB9S on the Joysticks, and the DB9P on the computer case. This connector series is designed, however, so that a DB25P will accept two of the DB9S's. Thus, I decided to use this connector for this project to allow easy wiring and permit future keyboard connections. These plugs are readily available from mail order houses such as Jade, Jameco, et al. Please note that 'plug' and 'socket' are the opposite you might expect; the plug mounts on the panel, and the socket is affixed to the cable end.

It is relatively easy to mount a DB25P above the keyboard. I chose a location on the upper vertical surface (above and to the right of the RUBOUT key), rather than on the rear panel. There is a cutout on the rear panel for an RS-232 connector, but I expect to use it for that purpose at some future time. To install the connector, completely dismantle the case, removing the board and carefully set it aside.

OSI

SOFTWARE FOR OSI

OSI

A JOURNAL FOR OSI USERS!!

The Aardvark Journal is a bimonthly tutorial for OSI users. It features programs customized for OSI and has run articles like these:

1) Using String Variables.

2) High Speed Basic On An OSI.

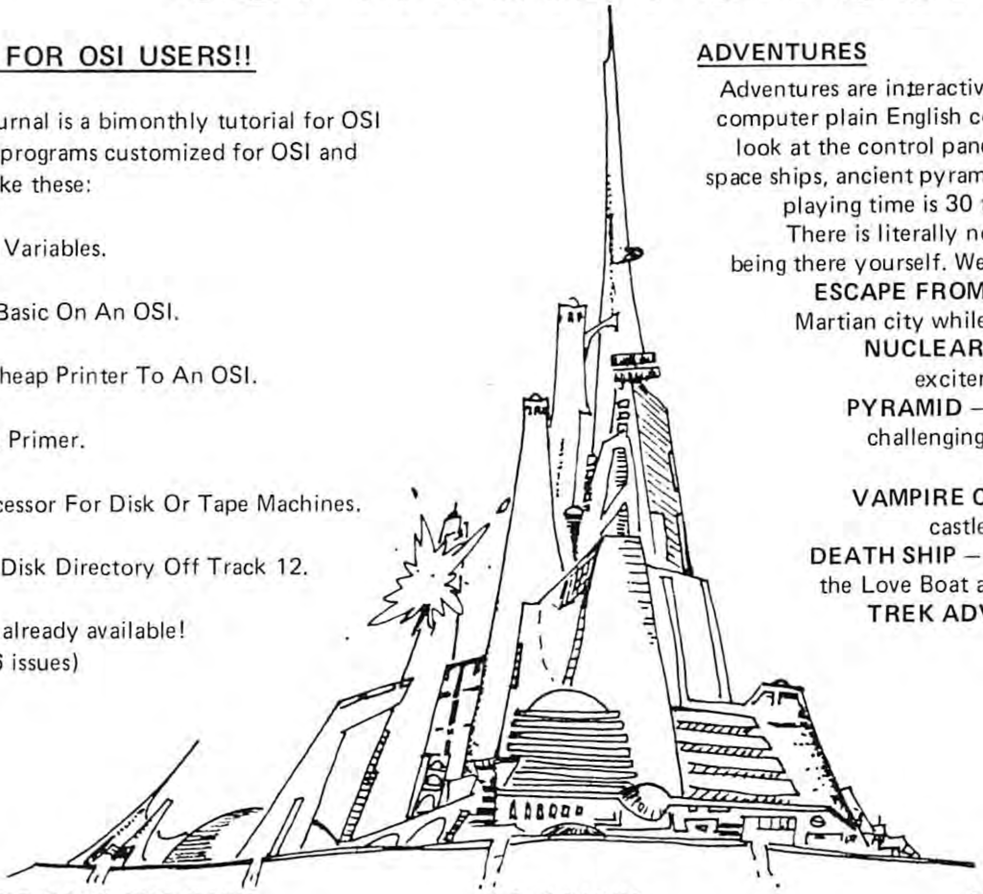
3) Hooking a Cheap Printer To An OSI.

4) An OSI Disk Primer.

5) A Word Processor For Disk Or Tape Machines.

6) Moving The Disk Directory Off Track 12.

Four back issues already available!
$9.00 per year (6 issues)



ADVENTURES

Adventures are interactive fantasies where you give the computer plain English commands (i.e. take the sword, look at the control panel.) as you explore alien cities, space ships, ancient pyramids and sunken subs. Average playing time is 30 to 40 hours in several sessions. There is literally nothing else like them – except being there yourself. We have six adventures available.

ESCAPE FROM MARS – Explore an ancient Martian city while you prepare for your escape.

NUCLEAR SUBMARINE – Fast moving excitement at the bottom of the sea.

PYRAMID – Our most advanced and most challenging adventure. Takes place in our own special ancient pyramid.

VAMPIRE CASTLE – A day in old Drac's castle. But it's getting dark outside.

DEATH SHIP – It's a cruise ship – but it ain't the Love Boat and survival is far from certain.

TREK ADVENTURE – Takes place on a familiar starship. Almost as good as being there.

$14.95 each

NEW SUPPORT ROMS FOR BASIC IN ROM MACHINES

C1S – for the C1P only, this ROM adds full screen edit functions (insert, delete, change characters in a basic line.), Software selectable scroll windows, two instant screen clears (scroll window only and full screen.), software choice of OSI or standard keyboard format, Bell support, 600 Baud cassette support, and a few other features. It plugs in in place of the OSI ROM. NOTE: this ROM also supports video conversions for 24, 32, 48, or 64 characters per line. All that and it sells for a mesly $39.95.

C1E/C2E for C1/C2/C4/C8 Basic in ROM machines.
This ROM adds full screen editing, software selectable scroll windows, keyboard correction (software selectable), and contains both an extended machine code monitor and a fix for the string handling bug in OSI Basic!! It has breakpoint utilities, machine code load and save, block memory move and hex dump utilities. A must for the machine code programmer replaces OSI support ROM. Specify system! $59.95

STRING BUG FIX (replaces basic ROM chip number 3)
All this chip does is to replace the third basic ROM and correct the errors that were put into the ROM mask. $19.95

DATA SHEETS

OS65D LISTING
Commented with source code, 83 pages. $24.95

THE (REAL) FIRST BOOK OF OSI
65 packed pages on how OSI basic works. Our best selling data sheet. $15.95

OSI BASIC IN ROM
Ed Carlson's book of how to program in basic. Now available from Aardvark. $8.95

P.C. BOARDS

MEMORY BOARDS!! – for the C1P. – and they contain parallel ports!
Aardvarks new memory board supports 8K of 2114's and has provision for a PIA to give a parallel ports! It sells as a bare board for $29.95. When assembled, the board plugs into the expansion connector on the 600 board. Available now!

REAL SOUND FOR THE C1P – and it's cheap! This bare board uses the TI sound chip to give real arcade type sound. The board goes together in a couple of hours with about $20.00 in parts. Bare board, plans, and sample program – $15.95

ARCADE AND VIDEO GAMES

ALIEN INVADERS with machine code moves – for fast action. This is our best invaders yet. The disk version is so fast that we had to add selectable speeds to make it playable.
Tape - $10.95 – Disk - $12.95

TIME TREK (8K) – real time Startrek action. See your torpedoes move across the screen! Real graphics – no more scrolling displays. $9.95

STARFIGHTER – a real time space war where you face cruisers, battleships and fighters using a variety of weapons. Your screen contains working instrumentation and a real time display of the alien ships. $6.95 in black and white - $7.95 in color and sound.

SEAWOLFE – this one looks like it just stepped out of the arcades. It features multiple torpedoes, several target ships, floating mines and real time time-to-go and score displays. – $6.95 in black and white $7.95 in color and sound.

SCREEN EDITORS

These programs all allow the editing of basic lines. All assume that you are using the standard OSI video display and polled keyboard.

C1P CURSOR CONTROL – A program that uses no RAM normally available to the system. (We hid it in unused space on page 2). It provides real backspace, insert, delete and replace functions and an optional instant screen clear.
$11.95

C2/4 CURSOR. This one uses 366 BYTES of RAM to provide a full screen editor. Edit and change lines on any part of the screen. (Basic in ROM systems only.)

FOR DISK SYSTEMS – (65D, polled keyboard and standard video only.)

SUPERDISK. Contains a basic text editor with functions similar to the above programs and also contains a renumberer, variable table maker, search and new BEXEC* programs. The BEXEC* provides a directory, create, delete, and change utilities on one track and is worth having by itself. – $24.95 on 5" disk - $26.95 on 8".

DISK UTILITIES

SUPER COPY – Single Disk Copier
This copy program makes multiple copies, copies track zero, and copies all the tracks that your memory can hold at one time – up to 12 tracks at a pass. It's almost as fast as dual disk copying. – $15.95

DISK CATALOGER
This utility reads the directory of your disks and makes up an alphabetic list off ~~all~~ your programs and what disks they are on. $14.95

MACHINE CODE RENUMBERER
(C2/4-MF only)
Renumbers all or part of a program at machine code speeds. – $15.95

VISA

This is only a partial listing of what we have to offer. We now offer over 100 programs, data sheets, ROMS, and boards for OSI systems. Our $1.00 catalog lists it all and contains free program listings and programming hints to boot.

master charge

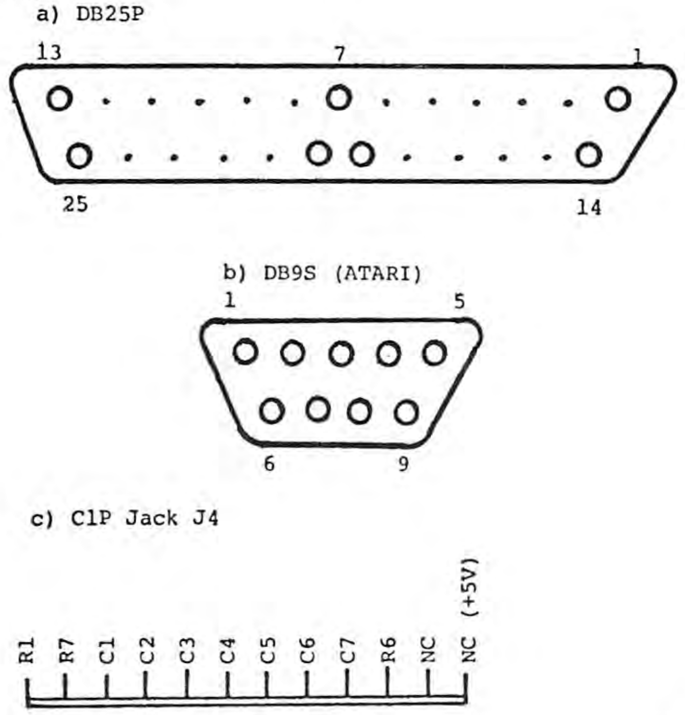
OSI

Aardvark Technical Services • 1690 Bolton • Walled Lake, MI 48088
(313) 669-3110 or (313) 624-6316

OSI

Mark the outline of the connector, and make a smooth, even cutout. I drilled several holes around the inside of the pattern, connected them with a side-cutter bit in an electric drill, and smoothed out the edges with a fine file. Be sure to rest the case on a soft surface such as an old towel, and center-punch all holes to avoid slipping and scarring the case with the drill bit. Remove all metal chips thoroughly to avoid shorting the PC board or power supply. Insert the connector, and drill the two mounting holes.

Figure 1



	DB25P		DB9S (L & R)		
	Pin	Connection	Pin	Color	Connection
	1	C4	1L	White	Up
	2	C5	2L	Blue	Down
	3	C6	3L	Green	Left
U	4	C7	4L	Brown	Right
P	5	NC (+5V)	5L	NC	-
P	6	NC (C1)			
E	7	NC (C2)			
R	8	NC (C3)			
	9	C4	1R	White	Up
R	10	C5	2R	Blue	Down
O	11	C6	3R	Green	Left
W	12	C7	4R	Brown	Right
	13	NC (+5V)	5R	NC	-
	14	C3	6L	Orange	PB
L	15	NC (R1)	7L	NC	-
O	16	R7	8L	Black	Common
W	17	NC (gnd)	9L	NC	-
E	18	NC			
R	19	NC			
	20	NC			
R	21	NC			
O	22	C3	6R	Orange	PB
W	23	NC (R1)	7R	NC	-
	24	R6	8R	Black	Common
	25	NC (gnd)	9R	NC	-

Next, connect color-coded wires of sufficient length to each of the connector pins you plan to use. Figure 1 shows the basic connections needed for both the DB25 and keyboard jack J4. You can use either a ten or twelve pin molex connector at J4, but be sure to match-mark it so that the orientation is always correct. You will note that the pins marked 'NC' on Fig. 1 have another connection indicated in parentheses. To save work later, I selected a variety of useful signals and made the connections to the DB25. At some future time, a ten-key input or other useful device can be easily plugged in, using a DB15 of DB25 socket. I ''borrowed the +5 volts from a pad near J4 in case my future peripheral needs power, but put a 100 ohm ½ watt resistor in series to avoid damage in case of a short circuit.

I use surplus ribbon cable as a cheap source for stranded color-coded wire. If you are adept at soldering, and have four hands, leave the ribbon cable intact. Otherwise, carefully separate the individual wires, solder them into proper place, and wrap the bundle every three inches. Double-check each connection before reassembling your computer.

Superboard II owners can just cut the DB9's off the end of the joystick cables and solder the wires directly to the molex connector. Figure 1 shows Atari's color coding, but it's best to check it, as production standards can change.

Readers who have seen articles on connecting Atari joysticks to computers such as the PET, with user ports, may try to combine the 'shoot' function with 'up' and 'down' as described. It won't work! There is already a diode in each keyboard row line, and the combined voltage drop across two diodes in series exceeds the threshold (trigger) voltage of 74LS integrated circuits. The method does work well with ports, and allows both joysticks to be connected to the eight data lines of one port address.

Testing the Joystick

It's pretty easy to run an elementary test of the completed circuit, as can be seen from the decoding of Table 1. If you Cold-start your C1P, the Up, Down, Left, Right, and Shoot functions will write the figures in Row 7 to the screen. The diagonal motions, and combinations of motion plus shoot, will give no screen indication, as they are the equivalent of multiple simultaneous key presses. Key in the following program to test all modes:

```
  5 FOR S = 0 TO 30 : PRINT : NEXT
 10 DIM G(16) : POKE 530,1
 20 POKE 57088,127
 25 REM-RT JYSTK POKE 57088,191
 30 FOR X = 0 TO 16 : READ G(X) : NEXT
 40 DATA 83,0,0,0,0,0,19,17,18,0,21,23,22,0,20,16,79
 50 Y = PEEK(57088)
 60 Z=Y OR 247:IF Z=247 THEN X=0:GOTO 80
 70 X=(Y/16) + 1
 80 IF G(X) = 0 THEN PRINT "ILLEGAL INPUT":
    GOTO 100
 90 POKE 54134,G(X)
100 FOR T = 0 to 100 : NEXT
110 GOTO 50
```

Table 1 - Output codes for nine-position joystick:

Position Output Codes

		OUTPUT CODES					
JOYSTICK	EQUIVALENT	SHOOT BUTTON OPEN			SHOOT BUTTON CLOSED		
POSITION	KEYS DOWN	BINARY	HEX	DECIMAL	BINARY	HEX	DECIMAL
	4 -	1110 1111	EF	239	1110 0111	E7	231
	3 :	1101 1111	DF	223	1101 0111	D7	215
Center	None	1111 1111	FF	255	1111 0111	F7	247
	2 0	1011 1111	BF	191	1011 0111	B7	183
	1 9	0111 1111	7F	127	0111 0111	77	119
	2,3 0,:	1001 1111	9F	159	1001 0111	97	151
	2,4 0,-	1010 1111	AF	175	1010 0111	A7	167
	1,4 9,-	0110 1111	6F	111	0110 0111	67	103
	1,3 9,:	0101 1111	5F	95	0101 0111	57	87

First, check the program for errors by using the equivalent key inputs from Table 1. Then, plug the joystick in and try all nine quadrants. Test both positions of the socket; note that it may be necessary to shave some Atari connectors with a file or sharp knife to fit the plugs. If any errors appear, the schematic and the output chart should be compared with the result of the test program to ascertain the reason and the proper corrective action required.

Game programming for the Joystick

It helps to have some knowledge of Boolean Algebra for the programming, but you can probably muddle through it as I did at first. The Boolean AND and OR operators in Basic can be very handy for masking unwanted inputs. In Boolean Algebra, 0 OR 0 =; 0 OR 1 = 1; OR 1 = 1. This operation is handled bit-by-bit, with no carry as occurs in binary addition. So %1110 0111 OR %1111 0111 OR %1111 0111 © %1111 0111. Thus an input which calls for both a move and a shoot can be masked so the computer sees only the shoot. The first binary number shown above results when the joystick is in the UP position *and* the Shoot button is pressed. If you want a program to stop moving and shoot whenever the button is pressed, just mask all but Column 3 by OR'ing with %1111 0111. This is done in BASIC by the following sequence:

```
200 POKE 530,I:POKE 57088,127
210 Y = PEEK(57088)
220 Z = Y OR 247
230 IF Z = 247 THEN 400
240 GOTO 500
250 REM - LINE 400 IS A SHOOT ROUTINE
260 REM - LINE 500 IS A MOVE ROUTINE
```

Thus, line 220 makes all but column 3 by OR'ing the input with %1111 0111 (#247), and line 230 checks to see if that bit is 0 which would mean that the key at row 7, column three is pressed. In this case, of course, it would mean that the shoot button is pressed.

A generalized subroutine in BASIC which would allow a single-square object to be moved around the screen by a game program is as follows:

```
42000 DIM G(16):FOR X = 1 TO 16:READ
G(X):NEXT :S = 53743
42010 KEY = 57088:POKE KEY, 127
42020 P = PEEK(KEY)
42030 PP = P OR 247
42040 IF PP = 247 THEN 400:REM-SHOOT
42050 X = (240 AND P)/16 + 1
42055 REM-LINE 42050 CONVERTS P TO NUMBERS
1 THRU 16
42060 E = G(X)
42070 GOTO 500 : REM-MOVE
42075 REM-LINE 42080 IS SCREEN MOVE OFFSETS
42080 DATA 0,0,0,0,0,33,-31,1,0,31,-33,-1,0,32,-32,0
```

A program which would use the above subroutine to move the object could be as follows:

```
500 REM-MOVE ROUTINE
510 S0 = 53248
520 SS = S + E:IF SS < S0 OR SS > S0 +
1024 THEN 42010
525 REM-LINE 520 KEEPS OBJECT IN SCREEN
MEMORY AREA
530 POKE SS,161:POKE S,32
540 S = SS
550 GOTO 42010
```

Line 530 POKE's the object to the new location SS, then blanks the old location S. You can vary the speed of movement by inserting a time delay such as "545 FOR T = 0 TO 99:NEXT".

A typical shoot routine could be as follows:

```
400 REM-SHOOT ROUTINE
410 FOR X = 1 TO 16:POKE S + G(X),188:NEXT X
420 FOR T = 0 TO 999:NEXT T
430 FOR X = 1 TO 16:POKE SS + G(X),32:NEXT X
440 Z = Z + 1:IF Z = 5 THEN POKE 530,0:STOP
450 GOTO 42010
```

To put all three sequences together, replace lines 200 through 260, above, with:

```
200 REM-MOVE AND SHOOT DEMO PROGRAM
210 RESTORE:Z = 0:POKE 530,1:GOTO 42000
```

Thus, combining lines 200 through 42080 provides a program which will allow the user to move a block around the screen by either key or joystick input, as well as simulate that block being destroyed by an explosion.

A careful examination of the OSI demo program mentioned earlier reveals a very similar action, except that more than one character is involved. Of course, Space Invader has several other routines such as move and shoot-back at random, scoring, etc.

Program Listing I shows the necessary modifications to the cassette program for the C1P for joystick conversion. It would have been nice to list the complete program; however, copyright laws forbid such publication without permission of the author.

When you run the program, you may be pleasantly surprised to find that not only have you added four more directions of movement (the diagonals), but play is speeded up by a factor of about 1.5.

```
2      REM - ATARI JOYSTICK 9/80
6      DIM G(16): FOR X=1 TO 16:
       READ G(X): NEXT
10     Z1=1
70     POKE KEY,127: P=PEEK(KEY): PA=P OR 247:
       IF PA=247 THEN 1100
71     X=(240 AND P)/16+1
72     DELETE
73     DELETE
74     DELETE
80     DELETE
1000   IF X=16 THEN 1050
1005   DELETE
1010   IF FND(H+G(X)-1)=0 OR FND(H+G(X)+1)=24
       THEN 50
1015   IF H+G(X)>54268 OR H+G(X)<53349 THEN 50
1020   H=H+G(X): I=H+1: J=H-1
1040   POKE I-G(X),V: POKE J-G(X),W
1047   E=X: GOTO 50
1050   E=0: GOTO 50
1110   GOTO 50
10005 DATA 0,0,0,0,0,33,-31,1,0,31,-33,-1,0,
       32,-32,0
20021 PRINT:PRINT:PRINT:PRINTTAB(6);"SPACEWARS":
       PRINTTAB(6);"----------"
20022 PRINT
20045 PRINT"USE KEYS AS FOLLOWS:"
20050 PRINT:PRINT TAB(8);"4
20052 PRINT TAB(8);"1 4
20055 PRINTTAB(8);"1
20057 PRINTTAB(8);"1 3
20060 PRINTTAB(8);"3
20062 PRINTTAB(8);"2 3
20065 PRINTTAB(8);"2
20067 PRINTTAB(8);"2 4
20070 PRINT"  STOP  NONE
20072 PRINT"  SHOOT  5
20075 ZZ=53800
20077 FOR X=0 TO 7
20080 POKE ZZ+X*32,X+16
20082 NEXT
20085 PRINT"CAREFUL, HE SHOOTS BACK!
20087 INPUT C$
```

©

Missing Listing
from Compute II, #3
Fast Graphics
by Charles Stanford

```
Listing 3: BASIC Program
  5 REM- CHOO CHOO COLLISION
 10 REM- FAST GRAPHICS DEMO
 15 GOSUB 100
 20 D=99
 25 A=59:B=29:C=29:POKE609,210
 30 GOSUB 50
 35 A=156:B=123:C=11:POKE609,209
 40 GOSUB 50
 45 GOTO 200
 50 REM- SCREEN WRITE SUBROUTINE
 55 FOR X=0 TO C
 60 POKE 11,34:POKE 254,96:POKE 608,A
 65 A=A-1:B=B+1
 70 X=USR(X)
 75 POKE 11,56:POKE 254,157:POKE 669,B
 80 X=USR(X)
 85 FOR T=0 TO D:NEXT T
 90 NEXT X
 95 RETURN
100 REM- MACHINE GRAPHICS WRITE TO RAM
    SUBROUTINE
110 RESTORE
115 POKE 11,34: POKE 12,2: POKE 254,96: POKE 255,2
120 FOR P=0 TO 61: READ M: POKE 546+P,M:NEXT P
130 DATA 160,0,169,32,153,0,211,153,0,210,153,
    0,209,153,0
135 DATA 208,200,208,241,234,234,234,160
    0,177,254,141,86,2,200
140 DATA 177,254,141,87,2,200,177,254,
    170,200,224,254,240,236,224,255
145 DATA 240,8,177,254,200,157,68,209,
    208,236,96,234,234,234,234,234
148 REM- GRAPHICS FIGURE TABLE
149 FOR P=0 TO 60: READ M: POKE 608 +P,
    M: NEXT P
160 DATA 155,209,1,2,3,167,4,157,5,161,8,167
165 DATA 32,165,33,161,34,161,35,161,37,155,38,
    176,39,161,40,161
170 DATA 64,166,65,161,67,161,68,161,69,128,70,
    161,71,161,72,161
175 DATA 96,176,97,224,98,225,99,226,102,226,
    104,226,255
180 DATA 131,209,0,165,3,161,4,156,5,165,7,2
182 DATA 32,161,33,161,34,178,35,155,36,161,37,
    161,38,161,39,161,40,167
184 DATA 64,161,65,161,66,161,67,128,68,161,69,
    161,70,161,71,161,72,168
186 DATA 96,226,98,226,101,226,102,224,103,225,
    104,178
187 DATA 255,0,0,0,0,0,0,0,0,0
199 RETURN
200 REM- EXPLOSION
210 GOSUB 300
220 Z=53711
230 FOR X=1 TO 6
240 FOR Y=1 TO 8
250 POKE Z+X*X(Y),42
260 NEXT Y
270 NEXT X
280 FOR T=0 TO 2500: NEXT
290 END
300 REM- EXPLOSION DATA
310 X(1)=-33: X(2)=-32: X(3)=-31
320 X(4)=-1: X(5)=1
330 X(6)=31: X(7)=32: X(8)=33
399 RETURN
```

©

SOFTWARE FOR OSI

Video Games 1 ....$15
Three games. Head-On is like the popular arcade game. Tank Battle is a Tank Game for two to four. Trap! is an enhanced blockade style game.

Video Games 2...$15
Three games. Grelin Hunt is an arcade-style game for one to three. Gunfight is a duel of mobile artillary. Indy is a race game for one or two.

Adventure: Marooned In Space...........................$12
An adventure that runs in 8K! Save your ship and yourself from destruction.

Dungeon Chase $10
A real-time video game where you explore a twenty level dungeon.

Board Games 1 ....$15
Two games. Mini-gomoku is a machine language version of five stones gomoku. Cubic is a 3-D tic-tac-toe game. Both with graphics.

Super- Biorhythms $15
A sophisticated biorhythm program with many unique features.

C1 Shorthand..$12
Use only two keys to enter any one of the BASIC commands or keywords. Saves much typing when entering programs. Written in machine language.

Disassembler $12
Use this to look at the ROMs in your machine to see what makes BASIC tick. Reconstruct the assembler source code of machine language programs to understand how they work. Our disassembler outputs unique suffixes which identify the addressing mode being used, no other program has this!

Cassette Data Base Manager $15
Here's a DBM that will work on non-disk systems. Most of the features of disk based DBMs are here. Does NOT use DATA statements as similar programs do.

For all BASIC-in-ROM systems. Selected programs available on disk. Color and sound on video games.

Send for free catalog listing many more programs.

ORION SOFTWARE ASSOCIATES

147 Main Street
Ossining, NY 10562

Breakout for OSI1P

Charles Stewart

```
 6 POKE605,0:FORK=611TO625:POKEK,32:NEXT
K:POKEK,255
 10 POKE515,0:CLEAR
 12 FORX=1TO10:READI(X):NEXT
 15 DATA30,31,32,33,34,-30,-31,-32,-33,-
34
 20 REM SET UP QUICK CLEAR
 30 POKE11,34:POKE12,2:POKE574,96
 40 FORX=0TO27:Y=PEEK(65036+X):POKE546+X
,Y:NEXT
 41 X=USR(X):INPUT"INSTRUCTIONS";A$:IFAS
C(A$)=89THEN8600
 47 X=USR(X):PRINT"HIT ESCAPE TO START"
 48 IFPEEK(57088)=254THENR8=RND(1):GOTO4
8
 50 X=USR(X):TP=53445:BOT=54149
 51 INPUT"NAME PLEASE";A$:PRINT:PRINT
 52 INPUT"DIFFICULTY LEVEL";DI
 53 DI=DI*INT(10*RND(1)):X=USR(X)
 54 CH=0
 60 FORSC=TP-32TOTP-8:POKESC,96:NEXT
 65 FORSC=TPTOTP+217:POKESC,159:NEXT
 70 FORSC=53437TO54173STEP32:POKESC,143:
NEXT
 80 FORSC=53412TO54148STEP32:POKESC,136:
NEXT
  100 BA$="BALL":K=611
  102 IFPEEK(K)=255THEN110
  105 X1=PEEK(K):POKE54160+K-611,X1:K=K+1
:GOTO102
  110 BALL=54151:FORLE=1TOLEN(BA$):POKEBA
+LE,ASC(MID$(BA$,LE,1)):NEXT
  120 KEY=57088:POKE56900,1
  121 S$="SCORE=":S=53390
  122 FORW=1TOLEN(S$):POKES+W,ASC(MID$(S$
,W,1)):NEXT
  125 FORSC=54115TO54115+32:POKESC,131:NE
XT
  130 SYM=54060
  131 VI=53965+INT(RND(1)*10)
  132 B=1
  133 X=I(8)
  134 D$=STR$(B):D=54155
  135 POKESYM,155
  136 FORY=2TOLEN(D$):POKED+Y,ASC(MID$(D$
,Y,1)):NEXT:POKED+Y,32
  139 U$=STR$(CH+B*5):FORE=2TOLEN(U$):POK
EE+S+7,ASC(MID$(U$,E,1)):NEXT
  140 POKEKE,254:ST=255-PEEK(KE)
  150 IFST=5THENSYM=SYM-1:POKESYM+1,32
  155 IFST=3THENSYM=SYM+1:POKESYM-1,32
  160 IFPEEK(VI+1)=143THENX=-33:POKEVI,96
:VI=VI+X:POKEVI,226
  170 IFPEEK(VI-1)=136THENX=-30:GOTO300
  190 IFPEEK(VI+32)=155THEN2030
  210 IFPEEK(VI-32)=159THEN3000
  220 IFPEEK(VI+32)=131THENGOSUB6000:B=B+
1:GOTO133
  295 IFPEEK(VI-32)=96THEN8000
  300 POKEVI,32:VI=VI+X:POKEVI,226
  305 FORT=1TODI:NEXT:GOTO135
  2030 G=INT(RND(1)*10):IFG<7ORG>10THEN20
30
  2035 X=I(G):GOTO300
```

All About OSI BASIC-IN-ROM

Reference Manual

computeII.: *"...any of several sections of this very well presented manual are worth the purchase price"*

Aardvark Journal: *"It is the book you were hoping was packed with your computer at the factory"*

PEEK(65): *"in goes far enough...to hold the interest of advanced programers as well as novices."*

Complete, concise, accurate, detailed. USR(X). Bugs. Tapes: BASIC, autoload and homemade. Source code and variable tables above $0300. Memory maps: $00,01, 02,A000-BFFF. Line-by-line description of MONITOR in $FE,FF.

$8.95 from your dealer
or postpaid from me.
Edward H. Carlson
3872 Raleigh Dr.
Okemos, MI 48864

```
 3000 X=32:CH=CH+1:POKEVI-32,32:GOTO300
 6000 FORR9=169TO32STEP-1:POKEVI,R9:NEXT
R9
 6010 VI=53965+INT(RND(1)*10):RETURN
 8000 CH=CH+(B*5):X=USR(X):PRINT"IT TOOK
 YOU"CH"TO BREAKOUT ";A$
 8002 IFR2=0THENPOKE605,CH:GOTO9000
 8003 PRINT"THE LOW SCORE IS";PEEK(605):
PRINT:PRINT
 8004 IFCH<PEEK(605)THEN9000
 8005 FORH=1TO10:PRINT:NEXT
 8010 PRINT"CARE TO TRY AGAIN"
 8015 R2=R2+1
 8020 INPUTA1$:IFLEFT$(A1$,1)="Y"THEN50
 8500 END
 8600 X=USR(X):PRINT"THE OBJECT IS TO BR
EAK-"
 8610 PRINT"OUT..LOW SCORE WINS!!":PRINT
"SHIFTS CONTROL PADDLE
 8620 PRINT:PRINT"EACH BALL COUNTS 5 POI
NTS":PRINT
 8630 INPUT"READY TO START";A$:GOTO47
 9000 POKE605,CH
 9010 FORK=611TO611+LEN(A$)-1:POKEK,ASC(
MID$(A$,K-610,1)):NEXTK
 9020 POKEK,32:POKEK+1,32
 9030 A$=STR$(CH):FORG=2TOLEN(A$):POKEK+
G,ASC(MID$(A$,G,1)):NEXTG
 9040 POKEK+G,255
 9050 PRINT:PRINT:GOTO8010
OK
```

©