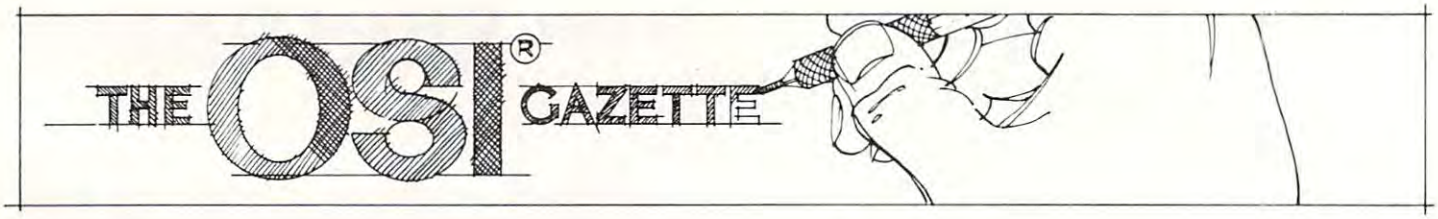


Through The Fill-The-Buffer Routine

With Gun And Camera

Kerry Lourash
Decatur, Illinois

This is an effort to shed some light on the Fill-the-Buffer routine (FTB) of OSI BASIC-in-ROM. Subroutines with FFXX addresses are for the C1P, but should be about the same for the C2P. Let me warn you - all numbers in this article are hexidecimal, unless stated otherwise! I will appreciate any corrections or additions readers may have.

What Is It?

The buffer mentioned is a section of zero-page memory (locations 13-5A). When you type in a line of BASIC or the tape recorder loads your favorite program the computer stores one BASIC line at a time in the buffer. Since the buffer is only 72 (decimal) bytes long, no BASIC line can be longer than 72 (dec.) characters. By the way, when you type a 4-digit line number, you have only 68 (dec.) characters left in the line. The FTB takes input from the keyboard or ACIA (Asynchronous Communication Interface Adapter), depending on the status of the SAVE and LOAD flags. After the line is stored in the buffer, other routines tokenize the line and store it in the BASIC workspace.

What Does It Do?

This is what the FTB does:

1. Filters input so no graphics or control characters except "BEL" (end of line) and NULL (zero) gets into the buffer.
2. Checks the "CTRL 0" (output) flag (loc. 64) to see if characters should be output to TV and ACIA.
3. Counts the number of characters input and gives an automatic carriage return/line feed (CR/LF) if the line length stored in loc. 0F is exceeded.
4. Outputs ten NULLS after a CR, and an additional number of NULLS equal to that stored in loc. 0D after a LF.
5. Implements control characters such as carriage return (0D), line feed (0A), "BEL" (07), backspace (5F), and line delete (@,40).
6. Puts a NULL in the buffer at the end of a line to mark the end of line for following routines. Sets the X and Y registers to the start of the buffer(-1).

Preparing For Our Journey

Machine language routines are murder to decipher, and the FTB is no exception. The code is compact in order to stuff BASIC into 8K of ROM, and uses nested subroutines extensively. In my chart, I've put the subs immediately after the point where they are called, instead of in numerical order. Also, subs are indented and bracketed, so the addresses at the far left are the main routine and the subs are at the right, in brackets. The format is somewhat like the outlines we did in school. I've tried to make the routine understandable to both machine language and BASIC oriented readers. The ML addresses have been kept so any part of the routine can be pinpointed and disassembled for additional info; BASIC readers can consider the addresses as line numbers. Most assembly language has been replaced by explanations of what is happening. I have used only a few mnemonics and have given their BASIC equivalents in the heading of the chart.

Into The Jungle

Now we're thru the preliminaries, on with the safari! Look for line A357 on the chart; this is our starting point. First, the X register is zeroed. The x-reg. counts characters as they are input into the buffer. Through a series of JSR's(JSR = GOSUB) and JMP(GOTO) thru RAM, we come to the input sub at FFBA. For those who have the Aardvark cursor program, this is where it steps in and does its stuff. Locations 218 and 219 are changed so that BASIC jumps to the Aardvark program instead of FFBA.

The Input Trek

The input sub looks at loc. 203, the LOAD flag. If the MSB (Most Significant Bit) of 203 is zero, the sub goes to FD00, the keyboard scan sub, which waits for an input from the keyboard, decodes it, puts it in the A register, and returns (RTS) to A389. On the other hand, if the MSB of loc. 203 is 1, the sub checks the LSB (Least Significant Bit) of F000, the ACIA's status register, and waits 'til it is zero, which means the ACIA has a byte ready in F001. This byte is stored in the A-reg. and the routine returns to A389, just like the keyboard routine does. Oh yes, one thing I forgot to mention: before F000 is

OSI

SOFTWARE FOR OSI

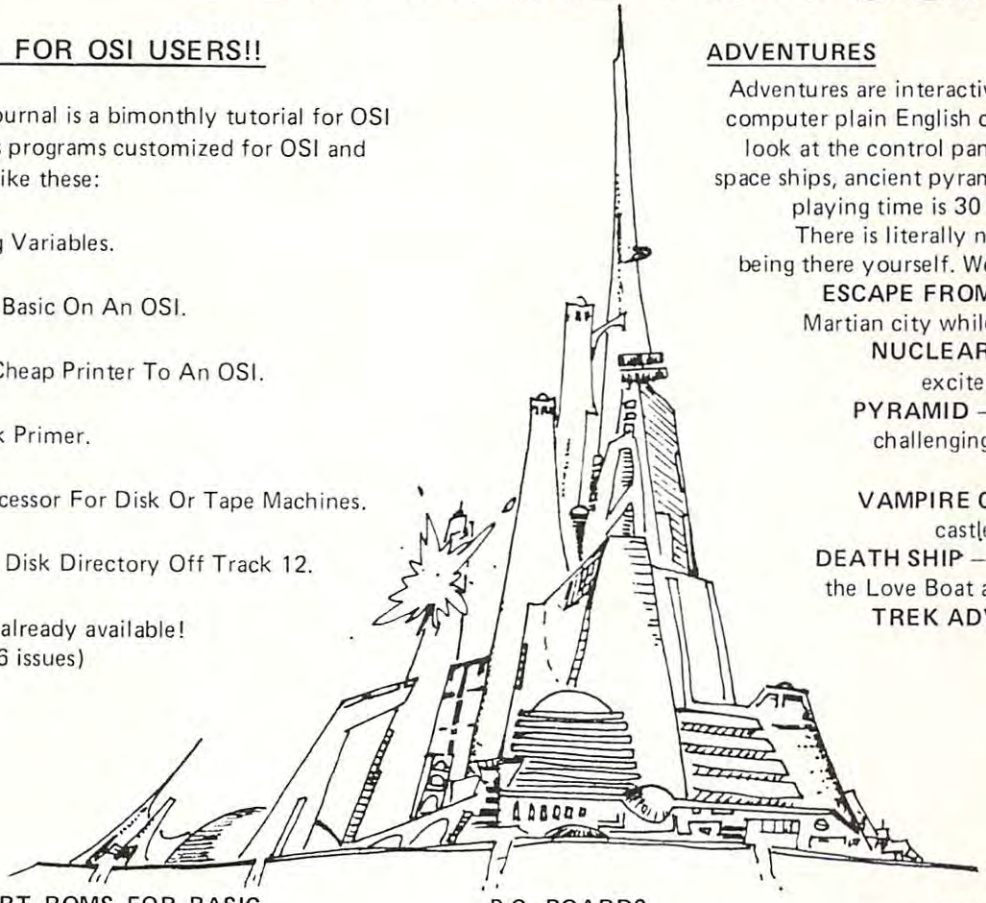
OSI

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.

First year 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.

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 chose 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. Replaces video swap tape on C1P model 2. All that and it sells for a measly $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 an extended machine code monitor. 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. Requires installation of additional chip when installed in a C2 or C4. C1 installation requires only a jumper move. Specify system $59.95.

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

MAXIPROSS (WORD PROCESSOR) – 65D polled keyboard only - has global and line edit, right and left margin justification, imbedded margin commands, choice of single, double or triple spacing, file access capabilities and all the features of a major word processor – and it's only $39.95.

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!

PROM BURNER FOR THE C1P – Burns single supply 2716's. Bare board – $24.95.

MOTHER BOARD – Expand your expansion connector from one to five connectors or use it to adapt our C1P boards to your C4/8P. - $14.95.

ARCADE AND VIDEO GAMES

GALAXIA one of the fastest and finest arcade games ever written for the OSI, this one features rows of evasive, hardhitting, dogfighting aliens thirsty for your blood. For those who loved (and tired of) Alien Invaders. – P.S. The price is a giveaway. SPECIFY SYSTEM!
Cassette $9.95 – 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

INTERCEPTOR C1P ONLY! An all machine code program as fast and smooth as the arcades. You use your interceptor to protect your cities from hordes of enemy invaders. A pair of automatic cannons help out, but the action speeds up with each wave of incoming ships. The fastest and most exciting C1P game yet.
C1P Cassette $19.95

MINOS – A game with amazing 3D graphics. You see a maze from the top, the screen blanks, and then you are in the maze at ground level, finding your way through on foot. Realistic enough to cause claustrophobia. – $12.95

SINGLE STEPPER / MONITOR

This is probably the finest debugging tool for machine code ever offered for OSI systems. Its' trace function allows you to single step through a machine code program while it continuously displays the A, X, Y and status registers and the program and stack pointers. You can change any of the registers or pointers or any memory location at any time under program control. It takes well under 1k and can be relocated anywhere in free memory. It is a fine tool for all systems – and the best news of all is the extremely low price we put on it. – Tape $19.95 – Disk $24.95

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".

AARDVARK IS NOW AN OSI DEALER!

Now you can buy from people who can support your machine.

– THIS MONTH'S SPECIALS –

Superboard II	$279
C1P Model II	429
C4P	749
8K 610 board for C1P	269
Epson MX-80 printer with RS232 installed	595

. . . and we'll include a free Text Editor Tape with each machine!

True 32X32 Video Mod Plans for C1P
(4 Chips $3.00 Crystal Required)
$7.95

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

OSI

checked, the keyboard is checked to see if the space bar has been hit. If so, the LOAD flag is turned off and we JMP to FD00 and then RTS to A389.

Now we have a byte, but we're not done processing it yet. At A389-A396 there is a section of code that tells the CPU to do nothing for a few microseconds. I'm not sure whether this is a time delay or just a spot where some code was deleted and the gap not closed up. Anyone know? After this lull, the MSB of the input byte is set to zero so we don't get any graphics characters and if the char. is a CTRL 0(0F) the output flag (loc. 64) is toggled. That means the output flag is changed to FF (all 1's) if it is zero, and vice versa. Finally, the input processing is completed and we RTS to the main routine at A35C.

Character Runs The Gauntlet

At A35C the character is tested to see if it is a "BEL". If it is, the X-register is checked to see if the buffer is full (more than 71 dec.). If there is room in the buffer, "BEL" is stored in the buffer and sent to the output sub A8E5 (more on this sub later). At A381 we are sent back to A359 to get another character. If the buffer is full, the "BEL" is output to the TV (or ACIA, if doing a SAVE) by A8E5, but "BEL" is not stored in the buffer. Now we are back at A359.

Let's temporarily bypass the test for carriage return (A360) and look at A364. This test blocks out control and graphics characters and sends us back to A359. That's why there's no way to stick a graphics char. directly into a line, even in a PRINT statement, without a CHR$ command. Look in your graphics manual and see what characters are legal (20-7D).

At A36C we test for @, the line erase character. We branch to A351 and JSR to A8E5 (outputs the @ character). Then a JSR to A86C, which sends a CR and a LF to A8E5, sending the cursor to "home". Now an RTS to A357 to zero the buffer counter, and we are back at A359, ready to start filling the buffer again. A370 tests for "SHIFT 0". Oddly enough, the ASCII of "SHIFT 0" happens to be 5F, which is also the cursor character. This time we branch back to A34B. A JSR to A8E5 outputs a cursor character. A34E decrements the buffer counter (X), and if we haven't erased backward beyond the start of the buffer, A34F sends us to ol' A359. If we have erased too far, a JSR to A86C homes the cursor, A357 zeroes the buffer counter, and we start filling the buffer at A359.

At A376 the buffer counter is checked. If the buffer is full, the input char. is changed to "BEL" (A37C) and output (A8E5) to tell you you're wasting your time. Nothing is stored in the buffer and we branch to A359 for another journey thru the FTB. Finally at A378, the character, if it has passed all the tests, is stored in the buffer. The contents of the buffer counter (X) are added to the number 13 (start of the buffer) and the character is stored at the resulting address. A37A increments the buffer counter, counter and A37E JSR's to A8E5, which prints the character.

The A8E5 Routine

Now for an explanation of the A8E5 sub. If the MSB of the output flag (loc. 64) is a 1, we RTS with no output to TV or ACIA.

If the MSB is zero, we check to see if the ASCII of the char. is less than 20 (BEL, CR, LF). If so, we skip the line length check and branch to A8FA. At A8FA we JSR to FFEE, which JMPs to the address found in 021A and 021B. This address is normally FF69, but you could cook up your own routine and put its starting address in 021A and 021B. From FF69, we JSR to BF2D, the video output sub, which I will explain in another article. To make a long story short, a "BEL" will be displayed as a graphics character, a CR will cause the cursor to be moved to the start of the line, and a LF will scroll the screen and "home" the cursor.

Now we RTS from the video sub and check the status of the SAVE flag (205). If 205 contains a zero, we RTS to A901. If the SAVE flag is non-zero the ACIA status register is monitored until its second bit is zero and then the character is sent to the ACIA (loc. F001). If the character is a CR then 10 (dec.) NULLs are also sent to the ACIA (this gives the computer time to process the line and scroll the screen when the program is LOADed from tape) and then we RTS to A901. A901 RTS's to A381 which brings us back to A359.

Back at A8EA, we assumed the input character would be less than 20. Let's see what happens if it's greater than 20. At A8EE addresses 0E and 0F are compared. 0E is the counter for the number of characters since the last CR. 0F contains the user-selectable line length (remember the "terminal width?" message at cold start?).

Don't confuse this line length with the maximum line length for the video stored at FFE1 or the cursor position counter at loc. 0200. If 0E and 0F are equal then there is a JSR to A86C. At A86C a CR and anLF are fed to the A8E5 sub for an automatic LF/CR. At A87A an additional number of NULLs equal to the number stored in loc. 0D are output. If 0E and 0F aren't equal there is a branch to A8F7 and 0E is incremented before the JSR to FFEE. The character is output to the TV and, if the SAVE flag is on, to the ACIA. Finally, we return to A359.

Last Leg of Our Journey!

Have patience, our journey is almost at an end. We skipped over the CR test at A360, now let's go through that one. If the input is a CR, a branch is made to A86C which puts a NULL at the end of the line in the buffer, marking the end of the line. This done, we are at A86C, which starts the auto CR/LF and the extra NULLs from loc. 0D. When we reach the end of the sub at A88A we RTS not to the FTB but to the Tokenize-the-Buffer routine, which is another story.

I highly recommend both Carlson's *OSI Basic In ROM* and William's and Dorner's *First Book of OSI*. The information in their books was invaluable in writing this article. I would like to hear from other people interested in Basic-in-ROM.

Fill-The-Buffer Routine (A357)

```
          JSR – GOSUB
          RTS – RETURN
 BRANCH,JMP – GOTO
          INC – ADD 1 (TO)
          DEC – SUBTRACT 1 (FROM)
        /02180/ – CONTENTS OF (LOC. 0218)
         CHAR – ASCII CHARACTER
          MSB – MOST SIGNIFICANT BIT
          LSB LEAST SIGNIFICANT BIT
ALL NUMBERS IN HEX:
A34B JSR   A8E5
     A8E5  (SEE A8E5 BELOW)
     .....
     A901  RTS
A34E DEC   X-REG. (BUFFER COUNTER)
A34F IF X  GREATER THAN ZERO THEN A359
A351 JSR   A8E5
     A8E5  (SEE BELOW)
     .....
     A901  RTS
A354 JSR   A86C
     A86C  (SEE BELOW)
     .....
     A88A  RTS
A357 ZERO X-REGISTER (BUFFER COUNTER = 0)
A359 JSR   A386
     A386  JSR FFEB
           FFEB JMP /218,219/ (NORMALLY FFBA)
           FFBA IF LOAD FLAG OFF, BRANCH TO FFD8
           FFBF IF SPACE BAR HIT, BRANCH TO FFD5
           FFCB IF ACIA NOT READY, BRANCH TO FFBF
           FFDI LOAD CHAR FROM ACIA AND RTS
           FFD5 TURN OFF LOAD FLAG
           FFD8 JMP TO FDOO (KEYBOARD SCAN SUB)
           FDOO (RETURNS WITH CHAR. IN A-REGISTER)
           FDCE RTS
     A389  TIME DELAY?
     A396
     A397  MASK MSB OF CHAR. TO ZERO
     A399  IF CHAR, IS "CNTRL 0" TOGGLE OUTPUT
           FLAG (0064)
     A3A5  RTS
A35C  IF CHAR. IS "BEL" (END OF LINE), BRANCH TO A376
A360  IF CHAR. IS CARRIAGE RETURN,  BRANCH TO A866
A866  PUT A NULL AT END OF LINE IN THE BUFFER (THIS SUB ALSO SETS X REGISTER & Y-REGISTER TO POINT
      AT BUFFER
      FOR GET-CHAR. SUB)
A86C  (SEE BELOW)
A88A  RTS  GO TO TOKENIZE BUFFER ROUTINE-THE END.
A364  IF CHAR. IS LESS THAN 20 OR GREATER THAN 7D THEN A359
A36C  IF CHAR. IS @ (ERASE LINE) THEN A351
A370  IF CHAR. IS 5F (BACKSPACE, SHIFT 0) THEN A34B
A376  IF LINE LENGTH IS GREATER THAN 71(DEC.) THEN A37C
A378  STORE CHAR. IN BUFFER
A37A  INC X-REG. (BUFFER COUNTER) AND GOTO A37E
A37C  CHANGE A-REG. (CHAR. INPUT) TO "BEL"
A37E JSR   A8E5
     A8E5  IF OUTPUT FLAG(0064) IS ON, RTS (NO OUTPUT)
     A8EA  IF CHAR. IS LESS THAN 20(BEL, CR, LF)
           BRANCH TO A8F9
           CHARS ALLOWED PER LINE, JSR A86C
           A86C PUT CR IN A-REG. (TO BE OUTPUT)
           A86E PUT CR IN ACCRESS 0E
           A870 JSR A8E5
                A8E5
                .....
                A901 RTS
           A873  PUT LF IN A-REG.
           A875  JSR A8E5
                 A8E5
                 .....
                 A901 RTS
           A87A  OUTPUT NO. OF NULLS IN ADDRESS OD
           A886  ZERO ADDRESS OE (NO. OF CHARS. SINCE CR)
           A88A  RTS
A8F7 INC   0E
A8FA JSR   FFEE
     FF69  JSR BF2D
           BF2D VIDEO OUTPUT ROUTINE
           .... (THIS WILL BE EXPLAINED
           NEXT INSTALLMENT.)
           BF72 RTS
     FF6D  IF SAVE FLAG /0205/ IS OFF, RTS
     FF73  JSR FCB1
           FCB1 IF STATUS REG.(f000) OF ACIA
           NOT READY, THEN FCB1
           FCBA WRITE CHAR. TO ACIA (F001)
           FCBD RTS
     FF76  IF CHAR WAS NOT A CR, RTS
     FF7D  WRITE 10(DEC.) NULLS TO ACIA
     FF8A  RTS
A901 RTS
A381  BRANCH TO A359
```