TABLE 2

INPUT/OUTPUT

BIT NUMBER	INPUT FLAG
0	SERIAL INPUT (ACIA)
1	POLLED KEYBOARD
2	CASSETTE INPUT ON 430 BOARD
3	NULL (0) INPUT
4	MEMORY INPUT
5	DISK BUFFER #1 INPUT
6	DISK BUFFER #2 INPUT
7	SERIAL INPUTS FROM 550 BOARD

BIT NUMBER	OUTPUT FLAG
0	SERIAL OUTPUT (ACIA)
2	VIDEO MONITOR
3	LINE PRINTER
4	MEMORY OUTPUT
5	DISK BUFFER #1 OUTPUT
6	DISK BUFFER #2 OUTPUT
7	SERIAL OUTPUTS FROM 550 BOARD ©

Next time: Subroutine descriptions...

OSI C1P Fast Screen Clears Revisited

Charles L. Stanford

Since writing the article on Screen Clear Routines for the OSI C1P for Compute II, Issue 1, I've been particularly sensitive to variations on machine language programming methods which could be used to improve the use of the computer. Several publications have been of considerable help, especially Compute and Compute II, Micro, the Aardvark and Progressive Computing Catalogs, and of course Edward Carlson's fine book on OSI BASIC. Mr. Carlson recently published an article which has led, indirectly, to a way of tapping into the Monitor and BASIC routines which input from the keyboard and write to the screen, ACIA, etc. Certainly, these techniques are well known to the more advanced C1P owners. Unfortunately, these people, with few exceptions, aren't writing for publication. So most information is being passed (slowly) by word of mouth or by club newsletters.

There are at least four points at which you can "break into" routines which are actively treating inputs or outputs. These are the Subroutines at $00BC and $0207, and the Jump vectors at $0218 and $021A. I'm sure there are more there for the finding. For this article, the Input vector at $0218 will be used.

Normally, this location holds a Jump Indirect to the routines starting at $FFBA in the monitor ROM which input a character from the keyboard or cassette. But it's no trick to poke a new address into this location, then do a little modifying of the routine. In this case, as shown in the listings, we are changing the vector from $FFBA to $00D8. This is near the end of zero page, which is not used by BASIC. Note, however, that it is used by the Monitor, so a Break to the Monitor followed by a Warm Start will require that the vector be reset and that the program be reentered.

The program is short and simple in operation. Essentially, it Goes sub to FFBA, which inputs a character. Next, the character is tested, and if it is a $7F, the RUBOUT key code, one of the more efficient machine language screen clear routines is effected. If it is any other character, this is skipped, and the program goes on about its business.

Note also that line 2010 in Listing II also POKEs the vector into location $0B, the USR vector. Thus, you will have both a single key screen clear by pressing the rubout and a programmable one by calling X = USR(X).

LIST 1

```
00D8 20 BA FF   JSR $FFBA     GET A CHARACTER
00DB C9 7F      CMP #$7F      IS IT A RUBOUT?
00DD D0 15      BNE $00F4     IF NO SKIP TO END
00DF 48         PHP           SAVE THE CHAR
00E0 A0 00      LDY #$00
00E2 A9 20      LDA #$20$     BLANK CHAR
00E4 99 00 D3   STA-Y         STORE BLANK AT 256
00E7 99 00 D2   STA-Y         LOCATIONS IN FOUR
00EA 99 00 D1   STA-Y         PAGES OF VIDEO RAM
00ED 99 00 D0   STA-Y
00F0 C8         INY           NEXT ADDRESS
00F1 D0 F1      BNE $00E4     PAGE DONE?
00F3 68         PLA           RETRIEVE CHAR
00F4 60         RTS           EXIT SUBROUTINE
```

List 2

```
47000 REM-ONE KEY SCREEN CLEAR
47010 POKE 11, 223:POKE 12, 0:POKE 536, 216:POKE
    537, 0
47020 FOR M = 216 TO 244:READ D:POKE M, D:NEXT
47030 DATA 32, 186, 255, 201, 127, 208, 21, 72, 160, 0
47040 DATA 169, 32, 153, 0, 211, 153, 0, 210, 153, 0, 209
47050 DATA 153, 0, 208, 200, 208, 241, 104, 96 ©
```