Keyboard Conversion Program For The OSI C1P

Ronald C. Whitaker
Salt Lake City, UT

One of the features advertised by OSI for the C1P and Superboard microcomputers is a lower case character set. While this feature is present, trying to make use of it can be both confusing and frustrating. When the shift lock is depressed, the alphabet is decoded as upper case characters and the upper row of keys produces numbers. The symbols on the upper row of keys are produced when either shift key is depressed. This works well when writing BASIC programs since lower case characters are not recognized by the BASIC interpreter except when used in PRINT statements. When the shift lock is released, the alphabet is decoded as lower case and the upper row of keys is decoded as garbage. The numbers can be obtained by depressing the left shift key. This also changes the alphabet to upper case. Depressing the right shift key produces only garbage. There's probably a reason somewhere for this bizarre decoding pattern, but it certainly seems illogical.

This problem with keyboard decoding didn't make too much difference to me until I tried to write a program to teach touch typing skills to my kids. Then it became very apparent that the keyboard must decode as much like a standard typewriter keyboard as possible. This means that, when the shift lock is depressed, the alphabet must decode as uppercase and the upper row of keys must produce the symbols. With the shift lock released, the alphabet will decode as lower case, and the upper row will produce numbers. Finally, both right and left shift keys must decode the same and produce upper case alphabet characters and symbols from the upper row of keys. To do this, I wrote a short machine language program to change the decoding.

This program makes use of the fact that, when a subroutine call is made to $F000, the subroutine polls the keyboard and the ASCII code of the next key pressed is placed in memory location 531. The keyboard conversion routine then examines this value and, if necessary, converts it to a standard typewriter character and places it back in memory location 531. The converted value is then PEEKed from memory by the BASIC program and POKEd to the screen. Below is a BASIC program which demonstrates the use of this subroutine. It works as follows:

Line 5 specifies the end of RAM available to BASIC

Lines 10–30 POKE the subroutine into the protected area

Line 40 clears the screen

"...it became very apparent that the keyboard must decode as much like a standard typewriter keyboard as possible."

Lines 100–115 set up the beginning of the line to be written

Line 120 loads zero page addresses 11 and 12 (USR vector) with the starting address of the machine language subroutine and calls the subroutine via the USR function.

Line 130 provides a line feed/carriage return function whenever the RETURN key is pressed.

Line 140 provides a true backspace whenever the RUBOUT key is pressed

Line 150 erases the page and starts the program at the top of the screen

It is necessary to protect the area at the top of RAM where the machine language subroutine is stored so that the BASIC program will not write over the top of it. The REMARK lines at the beginning of the program give values to use in lines 5 and 10 when other than 8K systems are used. I intentionally avoided the "free memory" from $0222 to $02FF because of its growing popularity with almost everyone who writes short machine code programs. This popularity has led to an ever increasing number of conflicts between otherwise useful programs.

If I were going to improve the program further I would do two additional things: first, I would write another machine language program to erase the page instead of using the PRINT/scroll erase that I have included here. Second, I would write another machine language program that would scroll the screen upward when the cursor reached the bottom of the page. However, these additions are beyond the scope of the rather simple demonstration program listed here and are left for the reader to implement.

The Keyboard Conversion program as described here suggests additional uses for the OSI C1P microcomputer which would be difficult and impractical with the standard keyboard decoding provided by OSI. Two that readily come to mind are typing tutor programs and character oriented word processors or text editors. The conversion subroutine has been an interesting exercise in machine language programming and use of the USR function. Hopefully, the reader will find it useful and I would be interested to hear of any uses found for it.

KEYBOARD CONVERSION PROGRAM

```
5 POKE 133,179:POKE 134,31
6 REM..FOR 16K SYSTEMS CHANGE LINE 5 TO
  "POKE 133,179:POKE 134,63"
7 REM..FOR 24K SYSTEMS CHANGE LINE 5 TO
  "POKE 133,179:POKE 134,95"
8 REM..FOR 32K SYSTEMS CHANGE LINE 5 TO
  "POKE 133,179:POKE 134,127"
10 FOR X=8116 TO 8192:READ Y:POKE X,Y:NEXT
11 REM..FOR 16K SYSTEMS CHANGE LINE 10 TO
   "FOR X=16308 TO 16384:"
12 REM..FOR 24K SYSTEMS CHANGE LINE 10 TO
   "FOR X=24500 TO 24576:"
13 REM..FOR 32K SYSTEMS CHANGE LINE 10 TO
   "FOR X=32692 TO 32768:"
15 DATA 32,0,253,173,0,223,201,255,240,34,201,253,
   240,50,173,19,2,201
20 DATA 47,208,6,169,63,141,19,2,96,201,59,144,1,
   96,201,48,176,1,96,56
25 DATA 233,16,141,19,2,96,173,19,2,201,92,144,1,
   96,201,45,176,1,96,56
30 DATA 233,32,141,19,2,96,173,19,2,201,107,144,1,
   96,201,81,176,217,96
40 FOR N=1 TO 30:PRINT:NEXT
100 PL=53411:S=0
110 IFPL>54171 THEN 40
115 POKE PL+S,128
120 POKE11,180:POKE12,31:X=USR(X):C=
    PEEK(531)
130 IFC=13THENPOKEPL+S,32:S=0:PL=PL+64:
    GOTO110
140 IFC=127ORC=159THENPOKEPL+S,32:S=S-1
    :POKEPL+S,128:GOTO110
150 IFC=10ORC=42THEN40
160 POKEPL+S,C
170 S=S+1:IFS>23THENS=S-1
180 GOTO110
```

©

NOTICE

Krell Software's College Boards 81/82 Preparation Series is now available on OSI.

See our ad elsewhere in this issue.