Modifying SBASIC's OUTCH statement.

By Karl Lunt

I've about given up writing code in 68hc11 assembly language, having switched over to SBasic. SB handles most of the tasks I need to do, including formatted output, interrupts, serial I/O, and some fairly powerful control structures.

But now and then I'm forced to return to assembly language, to solve some of the few off-the-wall problems that SB just can't handle. As a case in point, consider formatted output to a device other than the SCI.

Hooking an LCD or other character-oriented output device to you robot can be way cool. Combined with SB's formatted PRINT statements, you can have your machine display some serious debug info while you run it through its paces. But SB wants to talk to the SCI by default, and most of the time you will have hooked your LCD to either a set of output lines or to the SPI.

Still, if you are willing to write a little assembly language, you can connect SB's PRINT statement to your LCD, and get formatted output on your new device. The following example, taken from my electronic stopwatch program, shows how to do it. Note that the code below only replaces _outch. Setting up the LCD requires additional routines, which I wrote in SB in this example. Consult the file STOPWTCH.BAS on my website for the full set of code.

The trick lies in writing a custom version of the _outch routine. This is a library routine that is called by SB anytime your source file uses a PRINT or OUTCH statement. The default library routine, shipped in the SB distribution file, talks to the SCI. Any invocation of PRINT or OUTCH automatically generates a JSR to the label _outch. The output .asc file also contains an INCLUDE statement for the asmhc11 assembler, which causes the assembler to load the outch11.lib file. Refer to the listing below, which shows the default outch11.lib file.

Changing PRINT or OUTCH so they use your selected output device rather than the SCI requires two steps. First, you must write your own assembly language version of _outch. Note that your _outch replacement must adhere exactly to the rules set down by the original _outch. This means that your routine must expect the character to output in the A register, and your routine MUST preserve the contents of the Y, X and S registers.

The second step involves disconnecting SBasic from its internal _outch library file, forcing it to use your _outch routine instead. This task is quite easy, given SB's design. Simply include the statement:

```
asmfunc _outch
```

in the body of your program, prior to using the first PRINT or OUTCH statement. Declaring _outch with an ASMFUNC statement warns SB not to include the outch11.lib file, and instead to rely on finding a routine by that name somewhere in your source file.

Using these two simple steps causes SB to redirect all formatted output from PRINT statements to your custom output device. Now you can display variable or memory contents in hex or decimal, right on your output device, using nothing more than a PRINT statement.

You can even write an _outch routine that looks at the ASCII characters generated by your SB program and converts them into actions to perform on various output devices. For example, you might write an _outch routine that accepts the letter A as a command to move a servo to a certain position, and the letter B to return that servo to its home position. Similarly, the letter X might turn a camera on, while Y might turn it off.

Or you could make an _outch routine that actually sets up packets for transmission to a network. The code in _outch would handle all of the low-level work, such as packet framing and CRC generation. When the _outch routine sees an outgoing CR or LF character, it would finish up the packet and send it out the network connection.

Now you can have all kinds of fancy output messages on your special output device, with no changes to your main SBasic code. So what are you waiting for?

Keep on keeping on...

Karl

```
*
*  outch11.lib      send a character to the output device (68hc11)
*
```

```
*  This routine sends a single character, passed in the A reg, to the
*  active output device, normally the SCI.
*
*  This is the low-level SBasic output routine, used by other library
*  functions such as _prtstrc.  You can change this routine, if you
*  wish, to support other output devices.  Make sure, however, that
*  you retain _outch as the subroutine's name.
*
*  This routine preserves all registers.
*

_outch
        pshx                            save X
        ldx     #$1000                  point X at I/O regs
        brclr   $2e,x,#$80 *            loop until ready to send
        staa    $2f,x                   send char
        pulx                            restore X
        rts
```

Here is the _outch routine from my STOPWTCH.BAS program; source is available on my website stopwtch.bas. This program assumes the LCD is hooked up in 4-wire mode, with PB7-PB4 holding the data nybble, PB3 tied to the LCD's E signal and PB2 tied to the LCD's RS signal.asm

```
PORTB   equ     $1004
_outch
        pshb
        ldab    PORTB
        andb    #$07
        orab    #$07
        stab    PORTB                   rs=1, e=0, msb=0
        orab    #$0f
        stab    PORTB                   rs=1, e=1, msb=0
        tab
        andb    #$f0
        orab    #$0c
        stab    PORTB                   rs=1, e=1, data
        andb    #$f4
        stab    PORTB                   rs=1, e=0, data

        ldab    PORTB
        andb    #$07
        orab    #$07
        stab    PORTB                   rs=1, e=0, msb=0
        orab    #$0f
        stab    PORTB                   rs=1, e=1, msb=0
        tab
        lslb
        lslb
        lslb
        lslb
        andb    #$f0
        orab    #$0c
        stab    PORTB                   rs=1, e=1, data
        andb    #$f4
        stab    PORTB                   rs=1, e=0, data
        ldd             #$100
out1
        subd    #1
        bne             out1
        pulb
        rts
endasm
```