DESIGN NOTE #043

AUTHOR: KELLY SMALL
KEYWORDS: STRINGS, STACK POINTER

A String Display Routine

Introduction

A lot of embedded projects use routines to send data to displays or UARTs for serial communication. Here is an approach for sending null terminated strings from Flashrom to a UART or a display routine, using assembly code.

Overview

There are several ways to send a string of data to a display (or UART). A real brute force approach would be something like the following:

```
        ldi     r16,’H’
        rcall   DisplayChar
        ldi     r16,’e’
        rcall   DisplayChar
        ldi     r16,’l’
        rcall   DisplayChar
        ldi     r16,’l’
        rcall   DisplayChar
        ldi     r16,’o’
        rcall   DisplayChar
```

Not very code efficient or easy to read. A better approach would be to store the whole string in one location, point to it, and call a routine to send it to the display.

```
        ldi     zl,low(Message1)        ;point the Z register to a null
        ldi     zh,hi(Message1)         ;terminated string in memory
        rcall   DisplayString           ;display the string

        {somewhere else in your code}
Message1:
        .db     "Hello",0
```

This approach is a lot cleaner and you can tell at a glance what the string is that’s going to be displayed. The only thing I don’t like about this approach is your message strings could be several pages away from the code that references them, forcing you to search through the code to find the message string, or the location in the code where that message is referenced.

The final approach used here is to call the display routine, with the string defined in memory, right after the call as follows:

```
rcall   DisplayRom
.db     "Hello",0
```

Now you can see the string that will displayed, right at the code location that will display it. Of course at first glance one would wonder how you can define a string right in the middle of your executable code, and how does the subroutine know where the string is stored without setting up a pointer before the subroutine call is made. The answer involves some clever use of the Stack Pointer.

Once the call to the subroutine is made, the contents of the Stack Pointer points to the next location after the call instruction, which is the beginning of the string we want to display. By popping the stack into a register pair, we now have a pointer to the string. By using the Z-register, you can make use of the LPM (or ELPM) instruction to fetch the string one character at a time. As you loop through the routine fetching characters, the Z-register increments to point to each character until the end of the string is reached. At this point the Z-register actually points to the next instruction after the string, not the Stack Pointer, so the Z-register is pushed back on the stack so the RET instruction will return to the first instruction after the string.

The following examples assume you have a routine called DisplayChar that will take the contents of R16 and send it to your display hardware.

```
;*****************************************************************
;       DisplayRom - Send a null terminated message to a        *
;                    display routine.  This version optimized   *
;                    for the ATmega128                          *
;                    Uses R16                                   *
;*****************************************************************
DisplayRom:
        pop     zh
        pop     zl              ;move Stack Pointer to Z-register
        lsl     zl              ;shift Z-register over for
        rol     zh              ;LPM operations

DR1:    elpm    r16,z+          ;get a character from rom
        cpi     r16,0           ;test for end of string
        breq    Rdone           ;jump when end of string
        rcall   DisplayChar     ;send the data
        rjmp    DR1
Rdone:
        lsr     zh              ;restore the Stack by pointing
        ror     zl              ;Z just past the rom-based string
        push    zl              ;then push it on the Stack so
        push    zh              ;the return operation places it
        ret                     ;in the Program Counter
```

```
;*****************************************************************
;       DisplayRom - Send a null terminated message to a        *
;                    display routine.  This version works with  *
;                    micros that don't support enhanced LPM     *
;                    instructions.                              *
;                    Uses R0 and R16                            *
;*****************************************************************
DisplayRom:
        pop     zh
        pop     zl              ;move Stack Pointer to Z-register
        lsl     zl              ;shift Z-register over for
        rol     zh              ;LPM operations

DR1:    lpm                     ;byte character from rom
        adiw    zl,1            ;inc Z-register
        mov     r16,r0
        cpi     r16,0           ;test for end of string
        breq    Rdone           ;jmp when end of string
        rcall   DisplayChar     ;display data
        rjmp    DR1
Rdone:  lsr     zh              ;restore the Stack by pointing
        ror     zl              ;Z just past the rom-based string
        push    zl              ;then push it on the Stack so
        push    zh              ;the return operation places it
        ret                     ;in the Program Counter
```