Experiment 5: 7-Segment Display and Interrupt Subroutine

06.11.2019
Res. Asst. Yusuf Huseyin Sahin
sahinyu@itu.edu.tr

*R*age-Goddess, sing the rage of Peleus' son Achilles,
murderous, doomed, that cost the Achaeans countless losses,
hurling down to the House of Death so many sturdy souls,
great fighters' souls, but made their bodies carrion,
feasts for the dogs and birds,
and the will of Zeus was moving toward its end.
Begin, Muse, when the two first broke and clashed,
Agamemnon lord of men and brilliant Achilles.

Homer, Iliad

*T*ORTOISE: (..) The reason we are here is to have a footrace.
ACHILLES: A footrace? How outrageous! Me, the fleetest of foot of all mortals, versus you, the ploddingest of the plodders! There can be no point to such a race.

Douglas R. Hofstadter, Gödel, Escher, Bach: An Eternal Golden Braid

1 Introduction and Preliminary

In this experiment we will obtain experience on 7-Segment display and interrupt subroutines. It is strongly suggested to read MSP430 User Guide - Chapter 8. Also, you **should** take a look at GPIO registers like P1SEL, P1SEL2, P1IES and P1IFG which are not used until this week, but will be useful for interrupts.

A 7-segment display contains eight LEDs which are controlled by different bits of an 8-bit input. An example scheme to control the display is given in Figure 1. It can be used to show the digits 0-9 and some letters: A, C, E, F, H, I, L, O, P, S and U. Before starting to experiment, it is suggested to fill Table 1 according to the connections given in Figure 1.

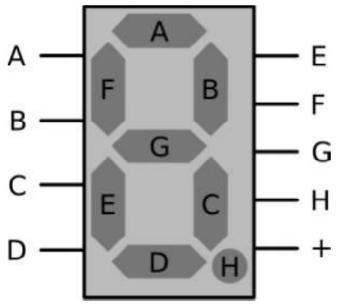

Figure 1: 7-segment display

Value	**H**	**G**	**F**	**E**	**D**	**C**	**B**	**A**
0	0	0	1	1	1	1	1	1
1						1	1	
2		1		1	1		1	1
3		1			1	1	1	1
4		1	1			1	1	
5		1	1		1	1		1
6		1	1	1	1	1		1
7						1	1	1
8		1	1	1	1	1	1	1
9		1	1		1	1	1	1
A		1	1	1		1	1	1
C			1		1	1		1
E		1	1	1	1			1
F		1	1	1				1
H		1	1	1		1	1	
I						1	1	
L			1	1	1			
O		1		1	1	1		
P		1	1		1		1	1
S		1	1		1	1		1
U			1	1	1	1	1	

Table 1: Some inputs for 7-segment display

2 Part 1: Counter Program

In the first part of the experiment you will write a counter program which counts between 0-9 repeatedly. A full period should last for 10 seconds. First, examine the port scheme of 7-segment display box given in Figure 2. Here, as you can see the display takes some bits from GPIO Port 2 as "selector". This bits are used to determine which of four 7-segment display will show the value (If multiple displays are selected, these displays will show the same value since they obtain the same input.). GPIO Port 1 is used to determine which LEDs of the selected display will be turned on.

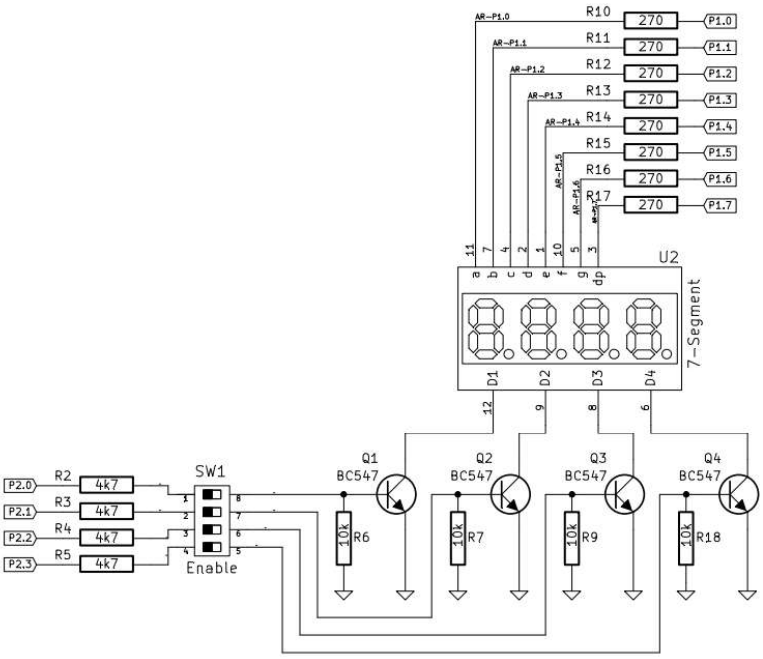

Figure 2: Port connections of 7-segment displays

You can use the following Delay subroutine. This given Delay subroutine finishes in approximately one second.You can define a byte array in memory and select the inputs from there.

```
Delay       mov.w #0A h ,R14 ;Delay to R14
L2          mov.w #07A00h ,R15
L1          dec.w R15 ; Decrement R15
            jnz L1
            dec.w R14
            jnz L2
            ret
```

3 Part 2: Interrupts

In this part, you will write an interrupt subroutine which is called to interchange the values shown in the selected display. There are two modes:

- Writing "Achilles" letter by letter (A,C,H,I...)
- Showing how many times the word "Achilles" is completed until that time. You can consider the last decimal digit (0-9).

You can define a variable in memory to remember the current mode. The subroutine should be used to change this variable.

For interrupts, we will use 6th bit from Port 2. The following code is useful to initialize the interrupt.

```
;these 6 lines of code are standard things to do
;in order to enable a pin of a port to generate interrupts.
init_INT    bis.b #040h, &P2IE ; enable interrupt at P2.6
            and.b #0BFh, &P2SEL ; set 0 P2SEL .6,
            and.b #0BFh, &P2SEL2 ; set 0 P2SEL2 .6 to enable interrupt at
   P2.6

            bis.b #040h, &P2IES ; high -to -low interrupt mode
            clr &P2IFG ; clear the flag
            eint ; enable interrupts
```

You need to store the interrupt vector of your subroutine. To do this, you can add a new vector under "Interrupt Vectors" definition at the end of template code. In the template, there is only one interrupt vector for resetting. Adding a new vector, we can obtain the following code.

```
;-------------------------------------------------------------------
; Interrupt Vectors
;-------------------------------------------------------------------
            .sect    ".reset"                ; MSP430 RESET Vector
            .short   RESET
            .sect ".int03" ; Port Interrupt Vector for P2. This information
                           ;is in fact obtained from msp430g2553.h header
                           ;file where it is defined. For ex. int02
                           ;is defined for port 1.
            .short ISR ;service routine for the interrupt in Port 2.
```

Lastly you should write your interrupt subroutine addressed by "ISR". You need to disable interrupts at the beginning of the subroutine and enable them when leaving to avoid multiple interrupts at a time. An example code is given below.

```
ISR     dint; Disable interrupts
        ;
        ; YOUR CODE
        ;
        clr &P2IFG; Clear the flag
        eint; Enable interrupts
        reti; Return from interrupt
```