Experiment 2: 7-Segment Display and Interrupt Subroutine

13.11.2023
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.

For example, the capital letter A could be drawn on the display using A,B,C,E,F and G bars. Thus, to write the letter A we should set the bits of P1OUT according to the following schemes.

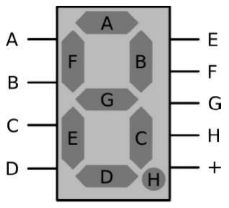

Figure 1: 7-segment display

Value	**H**	**G**	**F**	**E**	**D**	**C**	**B**	**A**
A	0	0	1	1	0	1	1	1

Table 1: Some inputs for 7-segment display

In our design kit, we have 4 7-segment displays as shown in Figure 2. Here, let's examine the wire connections carefully.

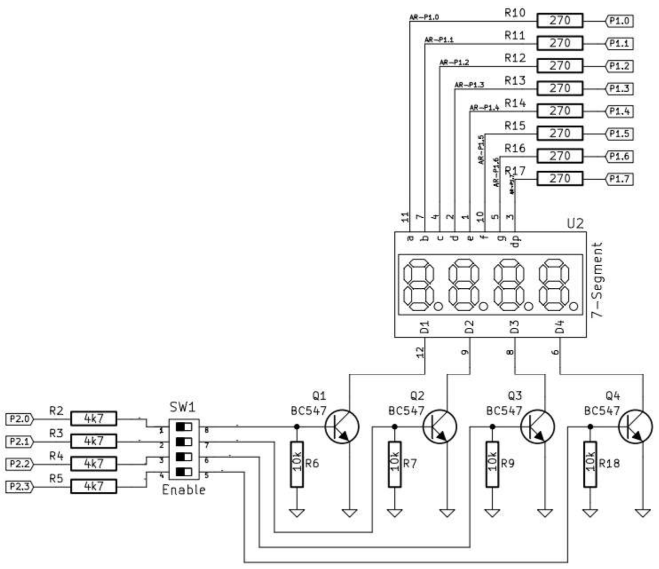

Figure 2: Port connections of 7-segment displays

Looking at the scheme we can summarize the following outcomes:

- To write the value on a display, we should push the bits into P1OUT. Bits 0-6 of P1OUT are directly connected to the display.

- Among the four displays, we should select the one to use via P2OUT. Bits 0-3 are connected to the enable inputs of the LEDs.
- Enabling only one LED at a time, it is impossible to show a four digits number on the kit. *Or is it?* Luckily, human visual cortex is not very complex. If we enable each LED respectively, and do not do waiting operations between each LED, we can fool the mortals! An example showcase is shown in Figure 3.

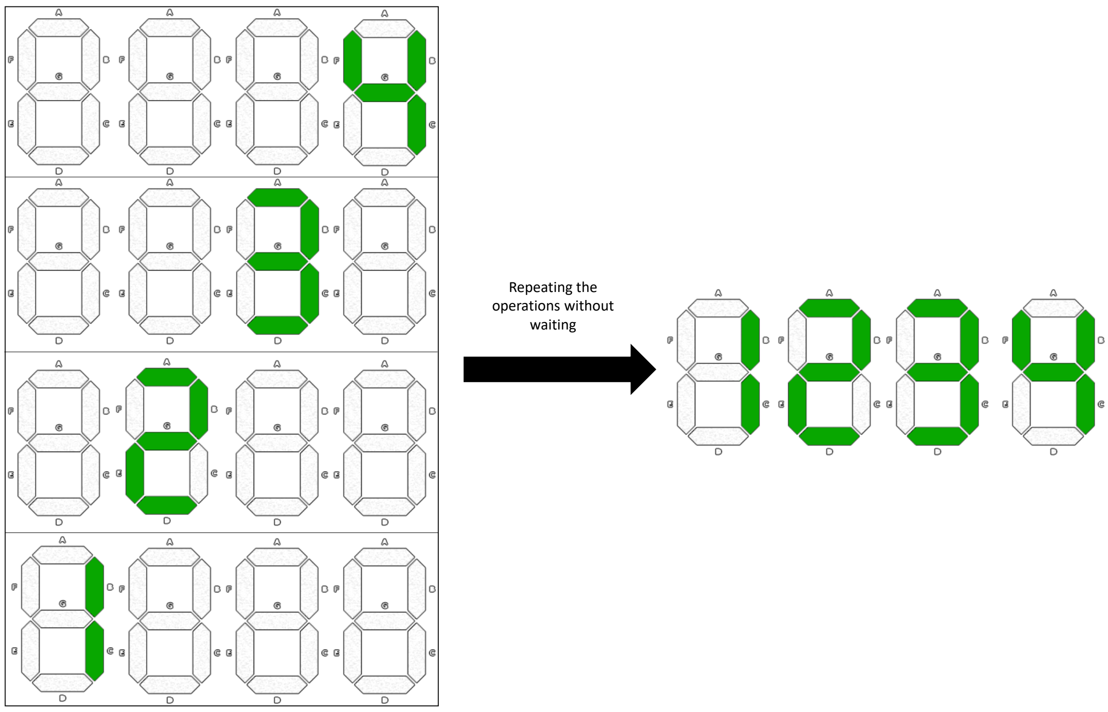

Figure 3: A method to show a four digit number.

The 7-segment display could be used to show the digits and many letters. A detailed list is given in Figure 4.

2 Part 1: (30 points) Counter Program

In the first part of the experiment, you will write a counter program that counts between 0-9 repeatedly. A full period should last for broadly 10 seconds. You can use the following Delay subroutine to ensure that showing a selected number takes approximately one second.

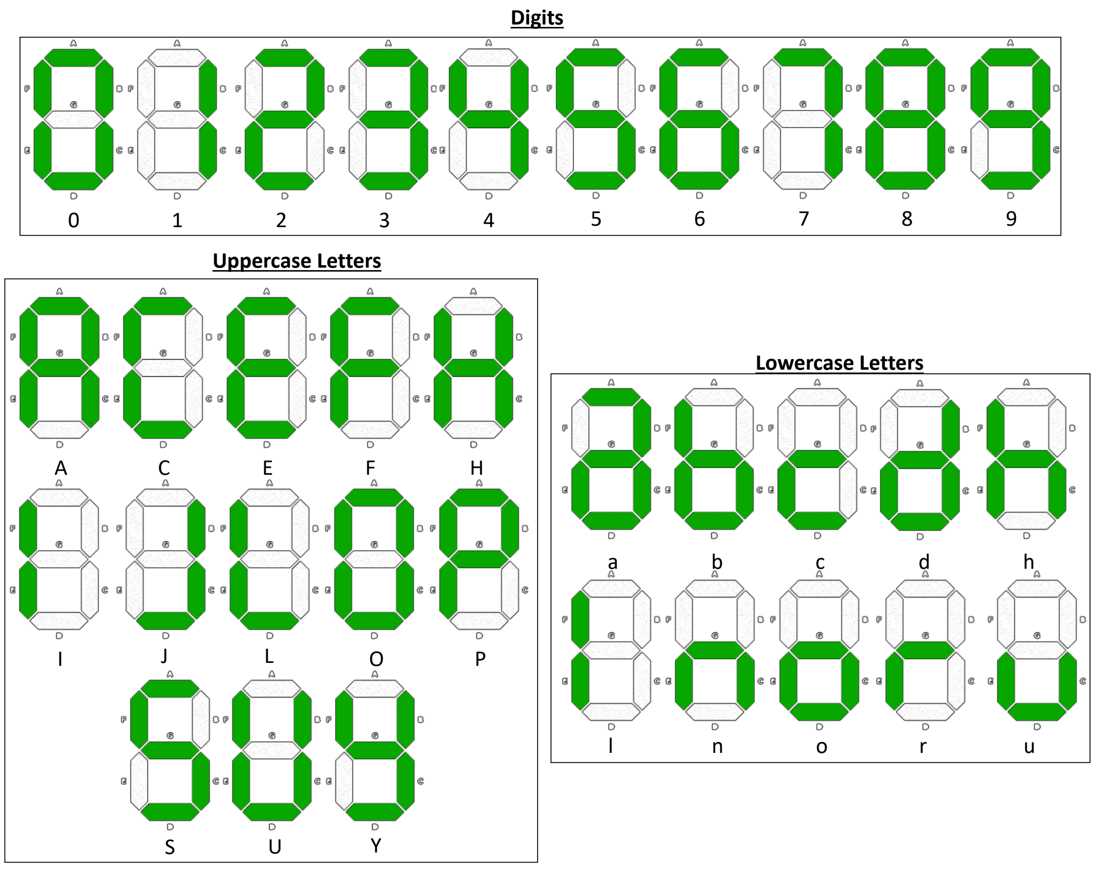

Figure 4: Some characters to show in the 7-segment display

```
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
```

You can define a byte array in memory and select the inputs from there. The byte array could be defined after the main code. Using the addressing "*#array*", we can reach this area of the memory.

```
     .data
arr  .byte 00111111b, 00000110b, 01011011b, 01001111b, 01100110b, 01101101b,
      01111100b, 00000111b, 01111111b, 01100111b
```

3 Part 2: (20 points) ACHILLEUS

To honor the brave warrior, change the code you've written so that it shows the letters A-C-H-I-L-L-E-U-S repeatedly.

4 Part 3: (30 points) Interrupts - I

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

- A-C-H-I-L-L-E-U-S
- 1-2-3-4-5-6-7-8-9
- 1-A-2-C-3-H-4-I-5-L-6-L-7-E-8-U-9-S

You can define the first two arrays in memory. Hovewer, or the third mode **you should use the other two arrays.**

You **should** 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
```

5 Part 4: (20 points) Interrupts - II

Change the code so that it only displays the letters for "0-1-2-3-4-5-6-7-8-9". Then, change the interrupt subroutine so that,

- **Button presses 1,2,3,4 times:** It pauses/unpauses the display
- **The 5th button press:** Resets the process, starts again from the first letter.