Experiment 6: Playing with Time

(16-20).12.2024
Res. Asst. Altay Ünal
unal21@itu.edu.tr

OK Lewis, close that gap as much as you can. It's hammer time.

Peter Bonnington

1 Introduction

In this experiment, timer, one of the most important features of the microprocessors, is explored.

2 Part 1

In this part, you should write an infinite loop as your main program code in order to lit different digits of 7-segment display panel simultaneously. You should be able to accomplish following output given in Figure 1 as the result of the first part.

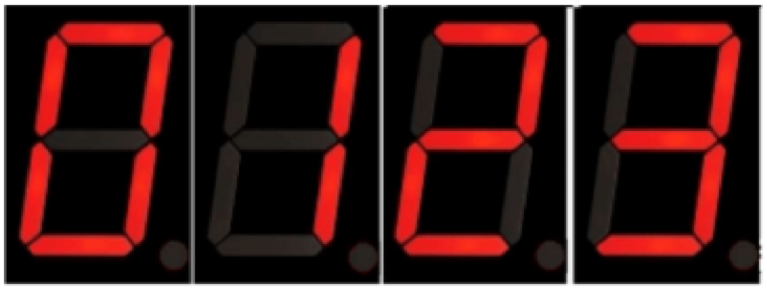

Figure 1: Sample output.

Hint: If a light flashes in a high frequency, we can not see flashes but only constant light.

3 Part 2

In this part, you are expected to design a chronometer with some features using the buttons. Additional features to your chronometer is given below.

- Reset - When reset button is pressed, time should be reset back to 0.
- Stop - When this button is pressed, the counter should stop counting.
- Start - When this button is pressed, the counter should start counting.
- Save Best Time - When the stop and start buttons are pressed at the same time, if the current time is the longest duration, then it must be saved.

There are three additional code blocks that you should include your code in order to build a chronometer. The list of these code blocks are given below.

- Timer Interrupt Subroutine
- Interrupt Subroutine
- BCD Conversion Subroutine

At first define seconds and centiseconds variables in data section as following. You may change the initial value as needed.

```
          .data
seconds         .byte 00h
centiseconds    .byte 00h
```

Timer and Timer Interrupt Subroutine

MSP430 family micro-controllers contain two 16-bit timers which could be utilized independently. Basic building blocks of the time are represented in Figure 2. Basic steps of the configuration and operation of the timer will be explained briefly. For further information, you should read **"Timer-A"** chapter in **"MSP430 User Guide"**.

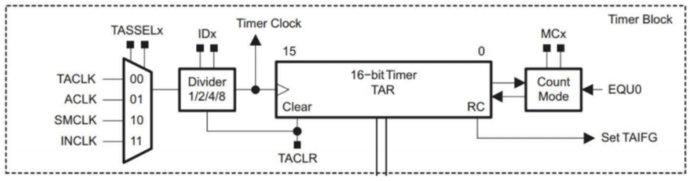

Figure 2: Timer circuit.

Timer could use four different signals as counting input. In this experiment you should use SMCLK signal as counting input. SMCLK is a square wave with 1048576 Hz frequency. Timer contains two operation modes (capture and compare) four different timer (counting) modes (Stop, Up, Continues, UpAndDown). We are going to use compare mode with up counting mode in this experiments. The list of the registers you should set are given below. Please check **MSP430 User Guide** at pages 369-372.

Register:

- TimerA Control (TA0CTL): Configuration of the timer
- TimerA Compare Capture (TA0CCR0): Holds the data to compare
- TimerA Comp. Cap. Control (TA0CCTL0): Configuration of Comp/Cap mechanism

Basically you should set the registers with the information given to you at the top. All the important information are given at the top. As you can remember from the previous lab session, microcomputer understands interrupts with interrupt flags and these flags should be cleared before returning from the interrupt. In timer this flag is located in one of the registers on top. Be sure to configure the timer correctly to generate interrupts with 10msec-period.

Hint: If you could not decide the value of a bit, it is probably 0.

To declare the timer interrupt please add the following line in the interrupt vector section of your code. Timer interrupts are handled as an ordinary interrupt from this point on.

```
        .sect "int09"
        .short TISR
```

Interrupt Subroutine

Remember your work in the previous experiment about the interrupts and implement your features accordingly.

BCD Conversion Subroutine

You should divide *centiseconds*, *seconds* values into BCD-digits in order to print them through 7-segment displays. We are holding the time information in variables of centiseconds and seconds but these values can not be written to 7 segment display directly. Convert the time to BCD-digits and use the following array to print the time into the 7-segment display.

```
array .byte 00111111b, 00000110b, 01011011b, 01001111b, 01100110b,
    01101101b, 01111101b, 00000111b, 01111111b, 01101111b
lastElement
```