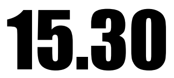
Laboratory #5
Kitchen Timer

Objectives:

- Implement interrupt functionality for the PIC32 timer module(s).
- Implement a kitchen timer, using the PIC32 GPIO pins and internal timer(s). Use the keypad and the four digit 7-segment displays for user I/O functions.

Files provided on Blackboard:

- Datasheets for the four digit 7-segment display and the 2-to-4 decoder
- lab5-ktimer.X.production.hex lab5-ktimer-audio.X.production.hex (executables that show the expected behavior of the kitchen timer, for the basic and the extra credit option)



Hardware configuration:

The 12-key (4x3) keypad from the last lab and the four-digit 7-segment display module that is already installed on the I/O shield will be used together for this week's lab assignment. In the previous lab assignment you have already implemented a device driver for the keypad module, which you will be using for the kitchen timer application. Refer to the lecture and Lab#4 documents for information about the hardware configuration of the keypad. The configuration for the four digit 7-segment display unit is provided below. The four digit seven-segment displays are configured to operate in a common-anode mode. Refer to the datasheet available on Blackboard for more information about the hardware details of the display unit.

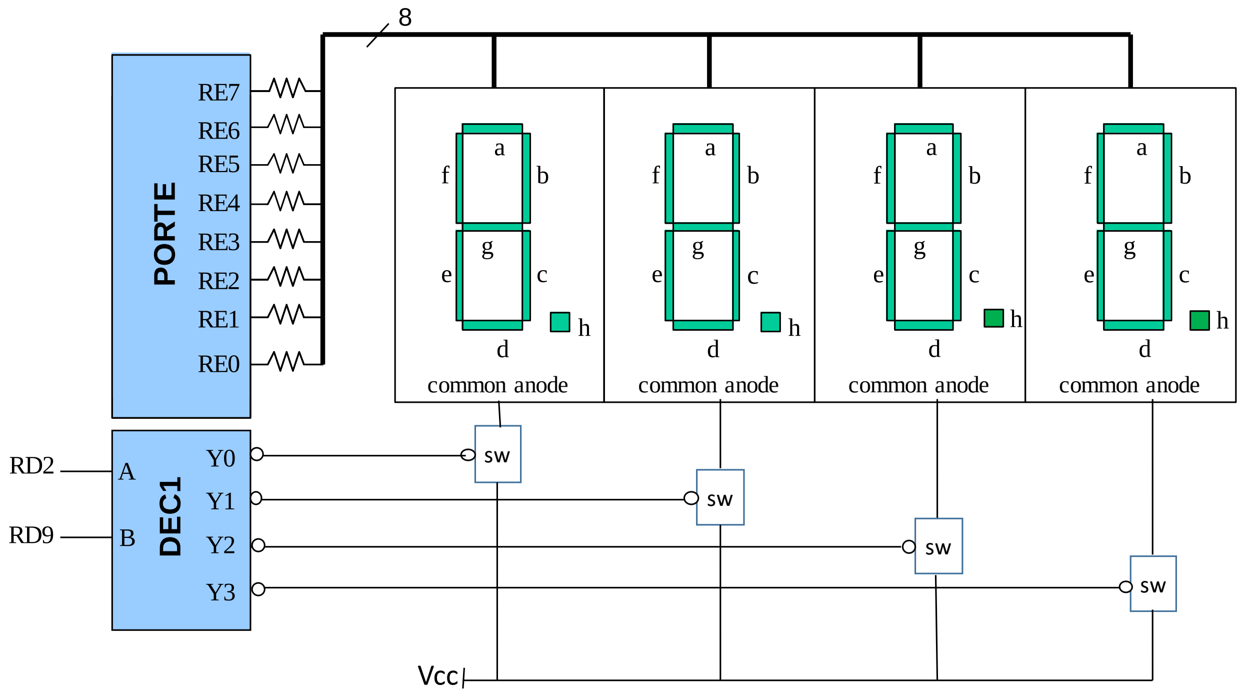


Figure 1: Interfacing the four 7-segment display units to the PIC32 GPIO pins

Since the data inputs for the four 7-segment displays are all connected to a common PORTE (RE7:0), a time de-multiplexing scheme needs to be employed using the display enable signals to select one display unit at a time for outputting data. The *common-anode* terminals of the four 7-segment display units (in the order most to least significant positions) are controlled by four MOSFET switches which are in turn controlled by the four active low outputs (Y0, Y1, Y2, and Y3) of decoder 1 (DEC1). The two select input signals for this decoder are driven by two pins of PORTD, namely RD2 and RD9. The relationship between the decoder inputs and outputs is given as follows:

Decoder DEC1 (SN74LVC1G139 2-to-4 Line Decoder)					
Inputs		Outputs			
B	A	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

From the decoder table, you can observe that only one of the four 7-segment display units can be enabled at any time. Therefore, to display a four digit number (such as **15.30**), you need to drive the corresponding display units in a sequential manner, by enabling each display and writing the data to the common PORTE for a short period of time (about 1 millisecond). The data you write to PORTE corresponds to the BCD-to-7-segment mapping (in active-low format) of the decimal number you want to display. This process needs to be repeated periodically to refresh the display and generate a steady output.

The decimal point in the middle of the four digit number can be turned on by using the most significant bit of the second display from the left.

Task:

The kitchen timer will have two modes of operation: the SET mode and the RUN mode.

When your program initially starts it always enters into the SET mode. While in the SET mode, your program waits for the user to enter time by using the keypad input. Each key-press on the keypad will "shift-in" a new digit on the right hand side of the four digit 7-segment display in a manner similar to a "queue" data structure. The old digits on the display will be shifted to the left as new digits entered from the keypad are introduced on the right side (in the least significant digit position).

Once you set up the time for the kitchen timer using the inputs from the keypad you can then press the "#" key to switch to the RUN mode of operation. During the RUN mode your program counts down by one second per second until the count reaches zero. The user should be able to switch back and forth between the SET mode and the RUN mode at any time, by pressing the same "#" key for mode control.

Any user attempts to enter invalid inputs, such as those that are not from the decimal digits ('0' to '9') or the '#' character could be quietly ignored by your application.

As part of satisfying the functional requirements of this project you are also required to keep the following points into account:

1) Use the keypad device driver your implemented in Lab#4 to access the user inputs from the keypad. Since we are not displaying the keypad inputs on the serial terminal, you need to remove the testing code you used in the previous lab assignment, for printing the keypad input characters using UART interface.

2) The time display on the four 7-segment LED display units should always appear steady no matter what mode of operation you are in. For this to work correctly you need to keep the display refreshed continuously at periodic intervals of time. This naturally calls for the use of a periodic timer interrupt. Therefore, you are required to have at least one timer interrupt to handle this functionality.

Finally, you need to implement an alarm feature for your kitchen timer. Thus, when the count reaches zero while the timer is in RUN mode, you need to activate the alarm by toggling the zero display (**00:00**) on-and-off once per second. The alarm feature should be turned off when the user returns to the SET mode.

Software Design: Since this lab assignment is relatively more involved than the previous ones, you are expected to develop a software design with the use of flowcharts to describe the algorithm you follow in implementing the major components of the program. You are expected to use modular approach in your program design, and at a minimum you need to provide the flowchart design for the display driver function and the main function that manages the overall operation of the program with its different modes.

Power consumption and its impact: One of the most important criteria in embedded systems design is power consumption. In your report explain what design techniques could be applied to minimize power consumption of the embedded system you have designed in this lab assignment. Also discuss the economic, environmental, and other implications of power consumption.

Extra credit opportunity: For up to 5% extra credit opportunity incorporate an audio alarm using the onboard speaker that can be controlled using the OC4 pin of the PIC32 (based on its output compare timer function).

Check-off and report:

Demonstrate your working program to the lab instructor and submit, via Blackboard, a lab report that follows the format and structure guidelines given in "Laboratory and Project Report Guidelines" available on Blackboard. Your report needs to include: Title page, Table of Contents, Objectives, Hardware, Software Design, Program Source Code, your answer for the question on power consumption and its impact, and finally Conclusions. It is recommended that you use flowcharts to present your structured high level program design for the major functional modules of your software.

Appendix:

Here is code template for the seven segment display driver which initializes a hardware timer module for implementing a timer interrupt. A small section of the header file is also shown.

```
// seven_seg.c
//this is an incomplete program code for the seven segment device driver code

#include "seven_seg.h"          //assuming you named your header file this way

//Example -- initialize Timer1 to trigger interrupts every 1ms
void init_Timer1(void) {
   TMR1 = 0x0000;               //clear timer register
   PR1 = xxxx;                  //set the period

   //init interrupts
   mT1SetIntPriority(5);        //group priority set to 5
   mT1ClearIntFlag();
   // configure for multi-vectored mode and enable system interrupt
   INTEnableSystemMultiVectoredInt();
   mT1IntEnable(1);             //enable T1 interrupts
   T1CON = xxxx;                //enable timer, set prescaler to 1:8

}



//Timer1 interrupt handler
//it is used to refresh the 7-segment displays every 1ms
void __ISR(_TIMER_1_VECTOR, IPL5SOFT) T1ISR(void)
{



    mT1ClearIntFlag();          //clear interrupt flag before exiting
}
```

```
//seven_seg.h
//this is incomplete template shows the include section for the seven_seg header file

#include <p32xxxx.h>

//suppress plib warnings
#define _SUPPRESS_PLIB_WARNING
#define _DISABLE_OPENADC10_CONFIGPORT_WARNING

#include <plib.h>
```