University of Puerto Rico at Mayagüez
Department of Electrical and Computer Engineering
Microprocessor Interfacing Laboratory

Experiment 4: Low-power Modes, LED Display Techniques & Keypads

---

Objectives

- Understand low power computing in your microcontroller
- Learn how to manipulate LED-based displays
- Learn how to use keypads

Duration

- 2 Hours in laboratory and extra time for homework.

Materials

- Your Microcontroller IDE application
- Your Microcontroller development board
- Dual seven-segment displays
- Twelve-key keypad
- Two push-button switches
- Assorted discrete resistors

Introduction

Power consumption in embedded systems is an important design factor that affects a wide range of aspects from battery life in portable applications to issues such as reliability, cost, size, and environmental impact. Therefore, using MCUs with low-power modes and learning how to and when to activate and use those modes becomes of outmost importance in the design of embedded systems.

The activation and use of low power modes in a microcontroller unit involves minimization of the individual current consumption of its internal peripherals, minimization of the CPU activity, and optimization of the code running during active periods. Depending on the particular MCU, the activation of a low power mode can involve: disabling the CPU by turning it off or sending it to a standby mode, reducing the CPU clock frequency, changing the clock source, etc. The system can exit any of these modes by an external interrupt.

In addition to learn about power management methods, this experiment will also discuss efficient techniques to manage multi digit displays and switch matrices. The subject of multi-digit displays will be discussed through the usage of dynamic visualization techniques in seven-segment LED displays. These techniques allow managing multiple digits with a single set of data lines, helping to reduce the number of I/O ports needed. The handling of switch matrices will be addressed with matrix scanning techniques that also allow to reduce the number of I/O lines required to serve large numbers of switches.

Procedure

Part I: Low Power Mode Operation

1. Assemble the up/down scroll system developed in homework 1 to scroll messages in an LCD display using two keys. Load the software version of your code that reads the switches by polling. Make the MCU board is only connected to the host computer through the programming adapter, with no other external power supply connected to it for the download process.
2. Remove the programming adapter and all connections to the host computer. Your program will safely stay in the MCU FLASH memory.
3. Connect you development system with an external power supply so that an ammeter can be connected to measure the MCU current consumption. Connect the LCD to the power supply such that its current does not pass through the MCU ammeter. Make sure the meter is in current mode and leads are connected ammeter input. Figure 1 below shows a diagram illustrating the connection.
4. Power-up the board and LCD and measure the MCU current. Icc = _____
5. Scroll the list up/down a few times to see the average load current measured by the ammeter.
6. Turn-off the power supply, remove the cables from the board and re-attach the programming adapter. Edit your code to serve the keys by interrupts. This time, modify the main to make the MCU enter low power mode right after the system set-up is complete and the interrupts have been enabled.

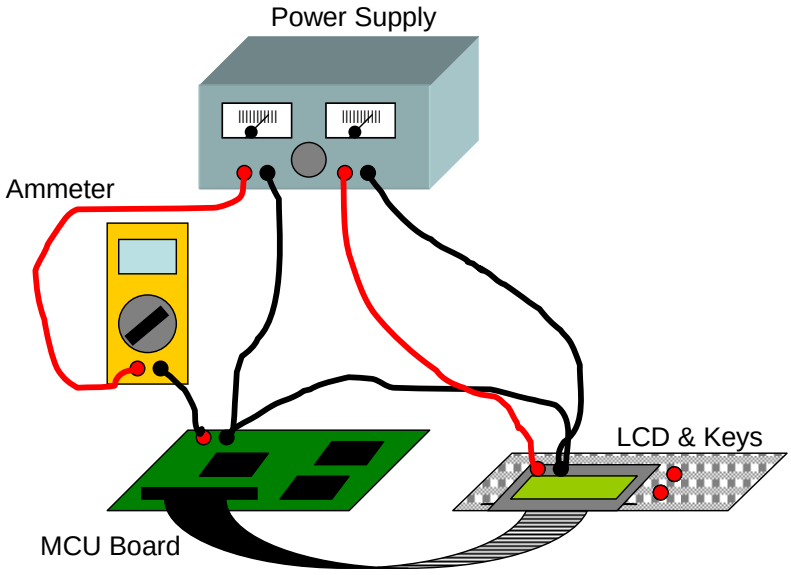

Figure 1: Connection diagram for part I.

7. Download the code to the system and still with the programming adapter connected (no connection to external power supply yet!), debug your code to make sure it works. Once you have a working version, remove the programming adapter and re-connect the external power supply as indicated in Figure 1. Operate the system now in a similar fashion as in
8. Measure again the MCU current consumption using the ammeter. Icc = _____
9. Did you notice any change in the supply current with respect to that measured in step 4? Comment on the obtained measurements.

Exercise 1:

Assuming the MCU is powered with a 1200mAH battery in both cases, how many hours would the battery last in each case. Assume the nominal voltage would be maintained up to the point that 25% of the battery remains.

Part II: Using 7-segment Displays

1. Connect a 7-segment display as illustrated in Figure 2 and calculate the value of the series resistor to set the segment current to not overload your MCU I/O pin output current and satisfy the requirements of the 7-segment. Also calculate the transistor base resistor to ensure it would saturate with the maximum 7-segment current with an overdrive factor of 1.
   This is a typical form of connecting a 7-segment display. The data signal contains the 7-seg code for the different digits and the common cathode (anode) line controls when the display is on or off.
2. Make a table to decode the digits from 0-F into 7-segment codes. For example, since you are using a common cathode 7-segment display, a logic high value is needed on the data line to power-on any segment. The control line needs to turn the driving transistor on to get the segment lit. Figure 3 shows the segment labels used in Table 1 to obtain the digit codes. The first two entries are already decoded as an example. Complete the rest of the table up to the code needed for digit F.

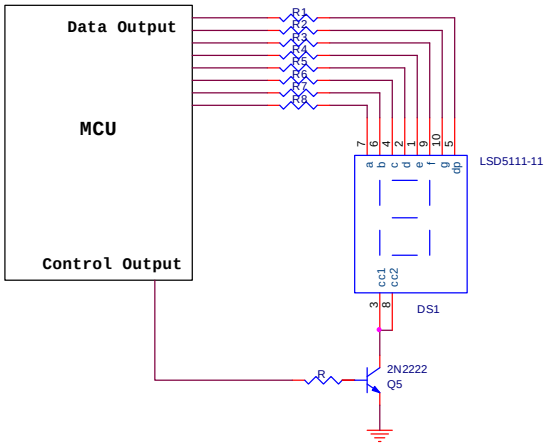

Figure 2, schematic for 7-seg

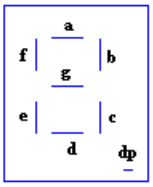

Figure 3, segments names

#	dp	g	f	e	d	c	b	a	7-seg
0	0	0	1	1	1	1	1	1	2Fh
1	0	0	0	0	0	1	1	0	06h
2									
3									
4									
5									
6									
7									
8									
9									
A									
B									
C									
D									
E									
F									

Table 1, Codes for 7-segment display of digits from 0 through F

3. Make a look-up table with this data and write a program that sends the appropriate code to the 7-segment port to display the digits between 0 and 9. The pseudocode below outlines the program to be written.

Lookup table, Pseudocode

```
;-------------------------------------------------------------------------------
; Progam Start
; INIT RESET VECTOR
; INIT STACK POINT, WDT
;-------------------------------------------------------------------------------
Lookup = 2Fh, 06h,…..
Number = 5
While TRUE
  Port = @lookup + number
  Pin_control = 1
  Endwhile
;-------------------------------------------------------------------------------
```

Exercise 2:

Make a 0-F counter with the MCU timer to automatically increase the count on the display every second.

Part III: Dynamic Display Using Two 7-segment Displays

1. Add a second 7-seg display and driver to your circuit as illustrated in Figure 4. With these two displays, let's apply a dynamic visualization technique that consist on the following steps:

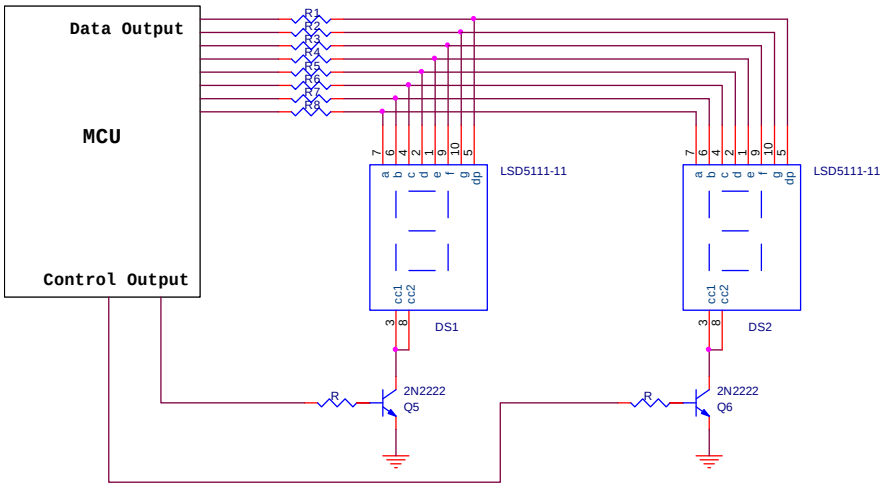

Figure 4, schematic for two 7-seg displays

a. Turn off the control signal for both 7-seg
b. Send the data to appear in the first 7-seg
c. Turn on the first 7-seg control signal
d. Delay loop
e. Turn off the control signal for both 7-seg
f. Send the data to appear in the second 7-seg
g. Turn on the second 7-seg control signal
h. Delay loop
i. Back to step a

2. Display a fixed number on the two 7-seg displays (ex. 55) and add a switch to increase the delay loop duration with each push. See the next pseudo code as reference. Use an ISR for the switch.

Main loop, Pseudocode

```
;-------------------------------------------------------------------------
; INIT RESET VECTOR, STACK POINT, WDT
;-------------------------------------------------------------------------
Lookup = 2Fh, 06h,…..
Number = 55h;
While TRUE
  Num_temp = number >> 4              ;get the most significant nibble
  Pin_control_1 = Pin_control_2 = 0   ;turn off both displays
  Port = @lookup + Num_temp           ;obtain 7-seg code for upper digit
  Pin_control_1 = 1                   ;turn on display

  Call delay_loop                     ;delay loop

  Num_temp = number & 0fh             ;get the least significant nibble
  Pin_control_1 = 0                   ;turn off display
  Port = @lookup + Num_temp           ;obtain 7-seg code for lower digit
  Pin_control_2 = 1                   ;turn on display
  Call delay_loop                     ;delay loop
  Endwhile
```

Delay loop subrutine, Pseudocode

```
;-------------------------------------------------------------------------
Count_Temp = delay_count  ;delay_count increase in ISR

While Count_Temp > 0
Count_Temp = Count_Temp – 1;    ;delay loop
EndWhile
Return
```

3. Run this program an increment the count delay using the switch until the numbers on the display appears to be static.

Exercise 3:

Make a 00-FF counter in dynamic mode. Replace the software delay loop by a refresh routine using a timer. Adjust the timer for a refresh rate of 60Hz.

Part IV: Using a Keypad

1. A key matrix can be scanned using a sequential sampling method. This method consists in setting a scan code with a “logic high” to the line of the row (R*x*) we want to scan and “logic low” the rest of the lines. All columns are read at once (return code). If a logic 1 is detected in a particular column line (C*y*) it means that key in the position (R*x*, C*y*) is depressed. These steps are sequentially repeated for each column until the entire keypad is scanned. If several keystrokes are expected, the scanning algorithm is executed in a loop until all keystrokes are received. This sequence assumes pull-down resistors are connected in the return lines.
2. Connect the keypad to your MCU as illustrated in Figure 5. Guided by the flowchart in Figure 6, scan the keypad and show in one display the pulsed key. This flowchart illustrates how to scan the keypad using a periodic timer interrupt to set the scanning frequency. It assumes the timer was programmed with the proper values. For this application, a scan period of 10ms is appropriate.

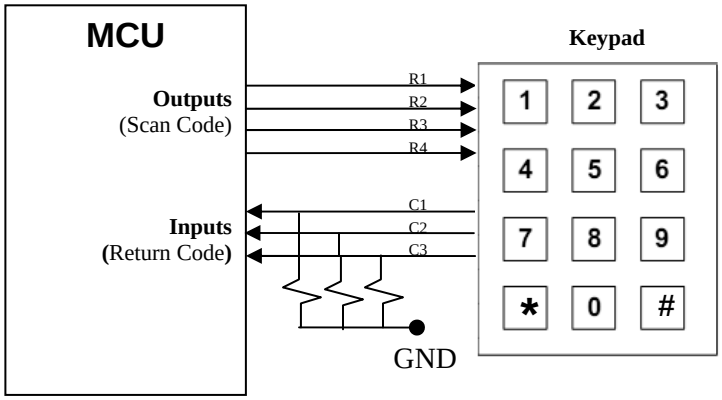

Figure 5: Keypad connection diagram.

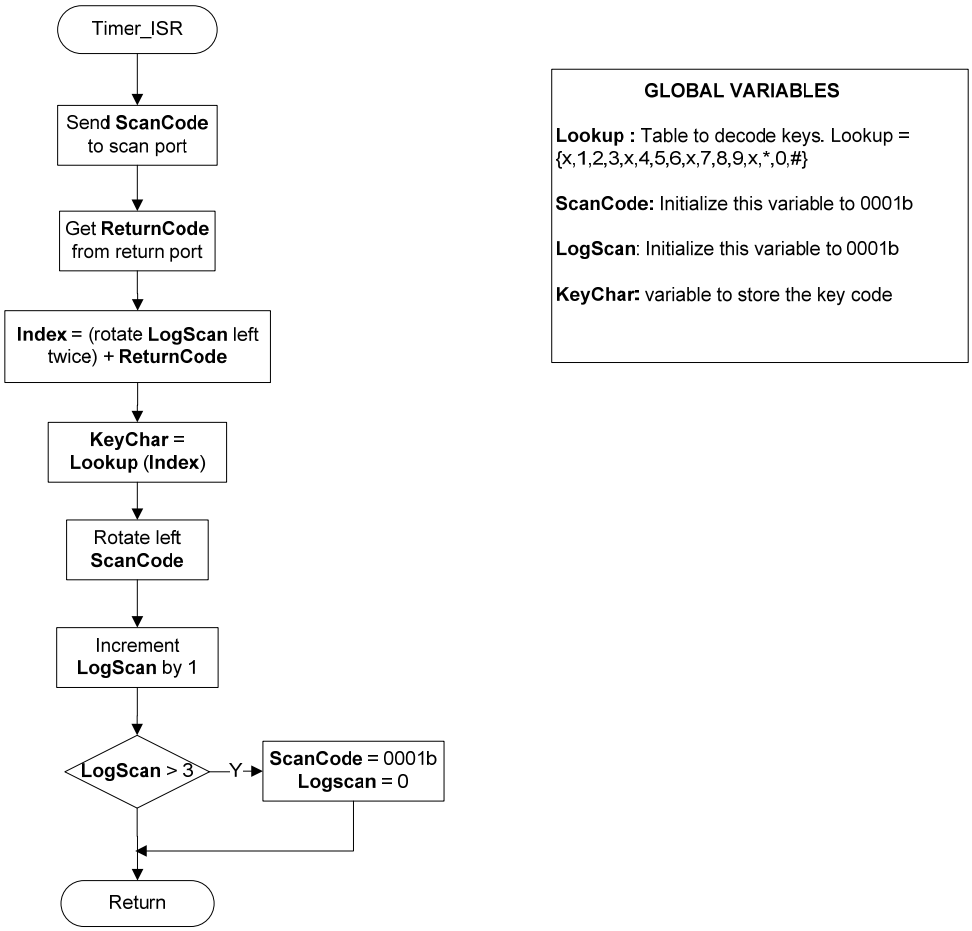

Figure 6, Read Keypad flowchart

Exercise 4:

Modify your two-digit program to show on the 7-segments the value of the last two depressed keys. Assign code ‘a’ to the * key and code ‘b’ to the # key.

Homework

Make a decimal add/Subtract calculator for two numbers in stack (RPN) mode. Where ‘*’ = ADD, ‘#’ = SUBTRACT, and add two switches for Clear and ENTER.