# *Lab 2: A Microprocessor-Controlled Game - Memory*

## Preparation

***Reading***

**Lab Manual:**

*Chapter 2 - Lab Equipment*

*Chapter 3 - Arrays*

*Chapter 4 - The Silicon Labs C8051F020 and the EVB* (sections relating to A/D Conversion)

***C language reference concepts:***

Data types, declarations, variables, variable scope, symbolic constants, functions, modular program development, variable passing, arrays

***Embedded Control Multimedia Tutorials***

*Hardware: Multimeter*

*Basics: Analog/Digital Conversion*

## Objectives

***General***

1. Introduction to analog-to-digital conversion using C8051 and development of a simple interactive game using the C8051 as the controller, and utilizing various switches and LEDs on the protoboard for game I/O.

***Hardware***

1. Familiarization with the use of the multimeter as a voltmeter.
2. Configuration of a potentiometer to provide variable voltage input to the analog to digital input port of the C8051.
3. Keep the circuit from Laboratory 1.2, adding an LED and two Pushbuttons

***Software***

1. The speed at which the game plays will be a function of the digital value read from the *game speed potentiometer* using the 8-bit analog to digital converter.
2. Further refinement of programming skills requiring bit manipulation.
3. Adaptation and re-use of software modules developed in previous lab exercises.
4. Refinement of skills in *top-down* and *modular program development* on a larger program.
5. Add the port initializations for the added hardware components

## Motivation

In the previous labs, you explored the use of *digital* inputs and outputs. However, most of the *"real world"* is *analog*. In this lab, you will explore how the C8051 can convert an *analog* signal into a *digital* value. The skills refined and developed in this lab exercise will be applicable to the next lab sequence—the *Blimp.* Portions of that lab sequence may also require the use of LEDs to indicate the status of various system parameters, the use of switches for input, the use of A/D conversion to determine the game speed, and the use of interrupts to keep track of time.

## Lab Description & Activities

***Description of Game Objective***

This game is called LITEC Memory. The program selects a random sequence of numbers of 0, 1, or 2, which corresponds to lighting 1 of 3 LEDs. Each LED is a different color. There will be a total of 4 push buttons, each corresponding to one color LED. The game starts by playing a random sequence of LED flashes. The goal of the game is for the player to correctly memorize a sequence of LEDs that were lit, and then to press the buttons in that same sequence.

***Description of Game Components***

The components of game consists of 3 LEDs (LED0, LED1, and LED2), Four pushbuttons (PB0, PB1, PB2, and PB3), 1 bi-color LED (BLED), 1 potentiometer, and 1 buzzer. A particular push button should be pressed if the corresponding LED was lit. If an incorrect pushbutton is pressed, the buzzer should sound and the game ends. The fourth push button is used to start a new round. The potentiometer is used to determine the speed of random LED display. The bi-color LED indicates if the program is displaying a sequence or waiting for the player response.

***Description of Game Play:***

At the start of the game (before the start game push button is pressed):

1. The three LEDs should be off
2. Buzzer should be off
3. BILED should be red

The program waits for the start game push button to be pressed. As soon as it is pressed, the game starts with a new round. The game speed is determined immediately by determining the voltage at the common leg of the potentiometer. This voltage is converted to an *on_time*, which is the time that the LEDs are lit before moving on to the next LED in the sequence. The time will be approximately 0.2 [s] < *on_time* < 1.5 [s].

*on_time* = [(A/D conversion result from the potentiometer) * 5+ 200] milliseconds.

The voltage across the potentiometer varies from

0 [V] to 2.4 [V]. A voltage input of 0 [V] from the potentiometer corresponds to 0.2 [s] and a voltage input of 2.4 [V] corresponds to 1.5 [s]. After an LED is lit for *on_time*, all of the LEDs should be turned off for *off_time*.

*off_time* = *on_time*/2

To summarize the process before the game starts, your program should:

1. Read the voltage input from the potentiometer and perform A/D conversion to determine a digital equivalent value
2. Calculate the wait time based on the digital value of the voltage
3. Generate an array of 5 random numbers from 0 to 2 (corresponding to which LED should be lit for the sequence)

After determining the game speed, the game play begins. As in Laboratory 1.2, a random number, 0, 1 or 2 is chosen.

1. If the random number is 0, LED0 is lit
2. If the random number is 1, LED1 is lit
3. If the random number is 2, LED2 is lit

When a game turn begins, the appropriate LED is lit for the *on_time* determined by the potentiometer setting. All LEDs are then off for *off_time*. The program should cycle through all 5 elements of the randomly generated array and light the appropriate LED each time for *on_time*, followed with all the LEDs off for *off_time*. After the sequence is played, the bi-color LED should turn green indicating that the program is ready for input. The player now repeats the pattern by pressing the corresponding push buttons. The program should compare the value of the pressed buttons with the values generated in the original array and print to the console whether the answer was correct or incorrect for each button press. If they all match, flash the 3 LEDs three times. If any value does not match, the buzzer should sound for 1.5 [s].

To summarize the game play process, the following steps should occur when game play starts

1. Bi-color LED is off, wait for the push button to be pressed.
2. A random number array is generated.
3. The appropriate LEDs are lit for *on_time*, following by *off_time*. This is repeated for all values of the random number array.
4. Bi-color LED is on.
5. The code records a correct or incorrect response and prints the result for *each* button that is pressed.
6. If all of the responses were correct, flash the LEDs three times. If any were incorrect, sound the buzzer for 1.5 [s].
7. (Repeat)

### *Game Enhancements*

You are encouraged to add additional functionality to your game for extra credits. You could add any additional hardware components or modify the software program to create a more complex, fun game as you wish. For software enhancement, you could make the game increase the length of the sequence if the user answers correctly from the previous round. For hardware enhancement, you could add an additional LED to introduce more complex patterns.You are encouraged to be creative.

Talk to your grading TA if you are not sure what to do for game enhancement. Up to five points extra credit will be awarded to students who demonstrate these types of enhancements.

## Hardware

*Figure C.3* shows the hardware configuration for this lab. The potentiometer is connected between +5V and Ground, with the middle pin connected to P1.1.

***Remember***

The neatness of wiring is important to debug the hardware.

The placement of LEDs and corresponding push buttons must be considered. Each button should be close to the corresponding LED, and they shouldn't be too closely packed.

## Software

You will need to write a C program to perform as described above. You are required to include code for the A/D conversion. At the end of this assignment a C programs provides an example for performing A/D conversion. The A/D reference voltages must be set to be 2.4 Volts and the A/D conversion gain must be set to be 1.

Following is a summary of A/D conversion SFR implementation.

1. In program initialization:
   - Configure analog input pins - set desired A/D pins in P1MDOUT to "0" and P1 to "1" and P1MDIN to "0".
   - Configure reference - set internal reference by clearing REF0CN pin 3:

     ```
     REF0CN=0x03;
     ```
   - Configure A/D converter gain - set to gain of 1:

     ```
     ADC1CF |=0x01
     ```
   - Enable converter:

     ```
     ADC1CN = 0x80;
     ```

2. Conduct A/D conversion:
   - Set pin to convert with AMUX1SL.

     ```
     AMUX1SL = XX; //XX: 0-7
     ```
   - Clear conversion complete bit

     ```
     ACD1CN &= ~0x20;
     ```
   - Start conversion:

     ```
     ACD1CN |= 0x10;
     ```
   - Wait for conversion complete:

     ```
     while((ADC1CN & 0x20) == 0x00);
     ```

3. Read results:
   - Access results register: *ADresult* = ADC1;

You will configure Timer0 using SYSCLK and 16-bit counting, as in Laboratory 1.2.

## Demonstration and Verification

1. Complete the psuedo-code that describes the program operation
2. Complete the Pin-out form, labelling the Port bits used, *sbit* variables, and initialization values
3. Use the multimeter to demonstrate that the potentiometer is connected correctly.
4. Run your C program and demonstrate that it performs as stated above.
5. Tell a TA how you created the *on_time*. Write a calculation in the lab notebook that estimates the worst case error of the actual time lag compared to the equation in the lab description.
6. Your TA may ask you to explain how sections of the C code or circuitry you developed for this exercise work. To do this, you will need to understand the entire system.
7. Also play the game (one player at a time) among the pair and fill the results of the game in the table (shown below) for three rounds using different values of speed pot setting.
8. Capture the screen of SecureCRT or Hyperterminal showing the output of one of the sample rounds and take a print out. Attach this printout along with your code in the lab notebook.

| | *on_time* | Number of Correct Answers | |
|---|---|---|---|
| | | Player 1 (Name: __________) | Player 2 (Name: ________) |
| Round 1 | | | |
| Round 2 | | | |
| Round 3 | | | |
| Total | | [ ] | [ ] |

## Writing Assignment - Design Report

You and your partner must submit a report for the Microprocessor-Controlled Game, according to the *Embedded Control Design Report format* on page 146. This report should cover the design and development of the entire game system. *Quality* and *completeness* are the criteria for grading your design reports.

Please note that the Design Report is in addition to your Lab Notebook - the Lab Notebook should still be kept up-to-date with regards to the work you did in this lab. Don't forget to refer to the guidelines in- *Writing Assignment Guidelines* on page 142.

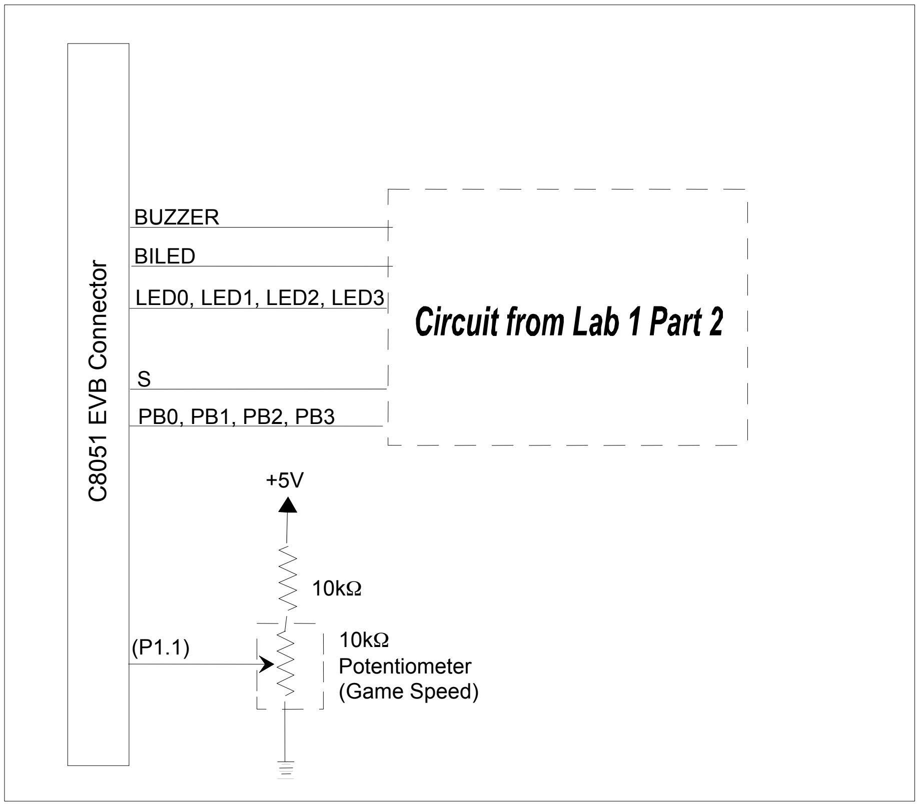

***Figure 2.1 - Suggested hardware configuration for Lab 2***

## *Sample C Program for Lab 2*

```
/* This program demonstrates how to perform an A/D Conversion */
main()
{
    unsigned char result;

    Sys_Init();                     // Initialize the C8051 board
    Port_Init();                    // Configure P1.0 for analog input
    ADC_Init();                     // Initialize A/D conversion

    while (1)
    {
        result = read_AD_input(0);  // Read the A/D value on P1.0
    }
}

void Port_Init(void)
{
    P1MDIN &= ~0x01;                // Set P1.0 for analog input
    P1MDOUT &= ~0x01;               // Set P1.0 to open drain
    P1 |= 0x01;                     // Send logic 1 to input pin P1.0
}

void ADC_Init(void)
{
    REF0CN = 0x03;                  // Set Vref to use internal reference voltage (2.4 V)
    ADC1CN = 0x80;                  // Enable A/D converter (ADC1)
    ADC1CF |= 0x01;                 // Set A/D converter gain to 1
}

unsigned char read_AD_input(unsigned char n)
{
    AMX1SL = n;                     // Set P1.n as the analog input for ADC1
    ADC1CN = ADC1CN & ~0x20;        // Clear the "Conversion Completed" flag
    ADC1CN = ADC1CN | 0x10;         // Initiate A/D conversion

    while ((ADC1CN & 0x20) == 0x00);// Wait for conversion to complete

    return ADC1;                    // Return digital value in ADC1 register
}
```