Lab 2: A Microprocessor-Controlled Game – Mastermind

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, potentiometers (pot), buzzers, 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, but modify it to contain a total of:
   1 pushbutton, 6 slide switches, 2 LEDs (Amber & Green), 3 BiLEDs, 1 pot, 1 buzzer, and whatever is required for pull-up resistors, current-limiting resistors and buffers.

Software

1. The reaction period will be a function of the digital value read from the *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.
6. NOTE: all times in this lab may be rounded to the nearest clock overflow integer value.

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 *Car*. 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 read a potentiometer voltage to set an initial timing value, and the use of interrupts to keep track of time.

Lab Description & Activities

Description of Game Objective

This game is called *LITEC Mastermind*. Details for the curious about the original game toy and its many variations can be found on the web via a simple search on the keyword "Mastermind game". This LITEC version of the program will select a random pattern to display on 3 BiLEDs; either Red, Green, or Off (for example Off – Green – Red). At the beginning the program generates 3 random numbers from 0 to 2 that correspond to the color of the 3 BiLEDs (0 is Off, 1 is Red, 2 is Green). The goal of the game is to guess this pattern correctly with the fewest tries and in the shortest amount of time. Low score wins. The length of the time window for guesses is determined by reading a "timer" pot before the game starts. The buzzer will provide feedback by indicating a wrong response (none of the colors guessed are correct with a 500ms burst) or indicate a completely correct guess to end the turn (a series of 5 very short bursts). Two players take turns and the "player" LEDs (Amber & Green) are used to indicate whose turn it is. Correctly guessing the pattern by a player ends their turn. Speed is also important as the number of points awarded for a guess start at 1 for a quick response and increment linearly to a maximum of 6 points if the interval exceeds the length of the time window set by the timer pot. The goal of the game is for the player to accumulate the fewest points possible.

Description of Game Components

The display unit consists of a row of 3 BiLEDs. Another row of 6 slide switches below the BiLEDs, 2 switches corresponding to each BiLED, will be used to input the player's guess. One slide switch in the ON position indicates the choice of Green and in the OFF position indicated the choice of Red. The 2nd slide switch in the OFF position indicates the choice of Off, regardless of the 1st switch's position. Once a player adjusts both slide switches for each of the 3 BiLEDs, one "guess" pushbutton will be used by the played to confirm his/her choice. Based on the guess, the total number of correct color and position guesses will be displayed on the SecureCRT terminal along with text recording the guessed pattern (for example "R G 0, 3 1" indicates their pattern guess and that 3 colors are correct but only one is in the correct position). The table below shows a possible sequence of guesses and responses.

Before the start of the game players should adjust the "speed" potentiometer to set the length of time the games uses to determine the points for a guess. The length will vary from 15 to 60 seconds, as the pot is turned from its minimum to maximum position clockwise. With a 15 second period any guess entered under 3 seconds is worth 1 point, under 6 seconds for 2 points, … under 15 seconds for 5 points. Any guesses that take more than 15 seconds have a maximum point value of 6. If the pot is set for the maximum period of 60 seconds, all the delays are adjusted linearly: under 12 seconds for 1 point, under 24 seconds for 2 points, … anything over 60 seconds for 6 points.

Actual pattern	Player guess	Correct Color Guesses	Correct Position Guesses
G-R-0	R-G-0	3	1
	R-0-G	3	0
	G-0-R	3	1
	0-G-R	3	0
	G-R-0	3	3
R-0-R	R-R-R	2	2
	R-R-0	3	1
	0-R-R	3	1
	R-0-R	3	3
0-0-G	R-G-0	2	0
	G-G-0	2	0
	0-G-0	3	1

Table 2.1 – LITEC Mastermind Display Examples

As the player adjusts the slide switches before entering the final choice the corresponding BiLED will display the corresponding color immediately. While a player is having a turn either the Amber or Green LED will be on to indicate which player is active. At the end of a turn when the player has correctly guessed the pattern, the buzzer has sounded 5 short blasts, and the total points for that turn are displayed on the terminal, the 3 BiLEDs will turn off and the player LED will switch. The program will pause until the "guess" pushbutton is pressed to indicate the next player is ready to play. After each player has guessed their patterns, the total scores for both players are displayed and the winner declared, either Amber or Green.

The indicators of the game are the buzzer and the BiLEDs. The buzzer sounds for 500ms when a guess has no correct colors. After each turn the score should be displayed on SecureCRT.

The choice inputs are 6 slide switches. Note that this requires adding hardware to the circuit from Lab 1-2 and moving the BiLEDs and switches to make the game easier to play.

The "speed" potentiometer is used to determine the reaction period window. It is read after the pushbutton is pressed to start the game.

Description of Game Play:

At the start of the game (before the pushbutton switch has been pressed):

1. The Green and Amber LEDs should be off
2. Buzzer should be off
3. BiLEDs should be off
4. The score should be cleared
5. Output an appropriate message to SecureCRT about the game and setting the "speed" pot.

The program waits for the pushbutton to be pressed (with debouncing). At that point the initial reaction window period is determined by reading the voltage at the wiper leg of the game speed potentiometer. This voltage is converted to a period T_{MAX}, which is used to calculate the time period. Depending on the voltage input, T_{MAX} will be in the approximate range 15,000ms - 60,000ms.

You can use the following expression to determine T_{MAX}. You will need to find appropriate values for *n* and *m* that satisfy the time range. The voltage across the potentiometer varies from 0V to 2.4V. Therefore a potentiometer voltage input of 0V corresponds to T_{MAX} = ~15,000ms and a voltage input of 2.4V corresponds to ~60,000ms. (Make sure T_{MAX} is declared as an unsigned int.)

T_{MAX} = [(A/D conversion result from the potentiometer) × n + m] milliseconds.

The program must keep track of each player's points. It will be easier to keep track of both players' scores using arrays. Many of the turn programming steps can be reused with simply a change of the array index.

Summarizing the process, after the game starts, your program should:

1. BiLEDs are off, Amber & Green LEDs are off, output a message, clear scores, and wait for the pushbutton to be pressed.

Game play

2. Read the game speed pot voltage and calculate T_{MAX}. Set $T = T_{MAX}$ and output value (ms).
3. Repeat the following steps each player.
4. Light Amber player LED.

Player turn

5. Generate 3 random values from 0 to 2 for BiLED pattern.
6. Repeat steps 7 to 15 until a match is found.
7. Start timed window for *T* ms and wait for pushbuttons to be pressed. Stop the clock if T_{MAX} is exceeded to avoid timing errors due to overflow.
8. While waiting for pushbutton read 6 slide switches and activate corresponding BiLEDs.
9. When the "guess" pushbutton is pressed capture the numerical value of the 3 pairs of slide switches (0, 1, or 2 for each pair).
10. If T_{MAX} was reached set points to 6, otherwise set points to 5T(ms)/T_{MAX}(ms) + 1 (integer divide and ignore fraction)
11. Compare the slide switches pair numerical values (0, 1, or 2) to the random color code values and display on SecureCRT the current 3-character guess as either 0, R, or G for each color or the digits 0, 1 or 2. Also display the number of correct colors, the number of correct positions and the total points scored so far.
12. Add points to score.
13. If no color matches are found sound the buzzer (5000ms).
14. If no complete match is found repeat back to step 6.
15. If a perfect match is found display score for the turn & sound 5 very short buzzer blasts.
16. Change player LED to Green, and after waiting for the pushbutton, repeat back to step 5.

Game end

17. If second turn is done then calculate total score and declare a winner.

18. Wait for the pushbutton to be pressed before starting a new game (step 1).

Game Enhancements

You are encouraged to add additional functionality to your game for a few points of extra credit. You could add any additional hardware components or modify the software program to create a more complex, fun game as you wish. For software, you could display an enhanced scoreboard or have players play 2 rounds for a total winning score. You are encouraged to be creative. Talk to your grading TA if you are not sure what to do for game enhancement. Up to four points extra credit will be awarded to students who demonstrate these types of enhancements. You must write up a description of your enhancement in your lab notebook and have the check-off TA sign to verify it worked. Then you must also include a section in your game report so that your grading TA can add in the extra credit to your report grade. **Do not attempt any enhancements if you are already behind schedule.**

Hardware

Figure 2.1 shows an overview of the hardware configuration for this lab. The 1k potentiometer connects to +5V through a 1k resistor and is input into P1.0.

Remember

Neatness of wiring is important for debugging the hardware and credit on check-off. The placement of LEDs and corresponding switches must be considered. The switches, BiLEDs and game pot should be logically laid out, and they shouldn't be too closely packed together.

A useful tool for debugging may be a "cheat" mode (recompile with an extra printf() statement) that prints the random pattern to allow you to verify the guess evaluations. This statement can be commented out and a recompile will give you back the normal play mode.

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;

An example of code using a `break;` statement

```
   ...
   while(tcount < Tmax)    //start reaction window timing
   {
      if(PB0 == 0)  //PB0 pressed
      {
            PB0flag = 1;
            break;
      }

      if(PB1 == 0)
      {
            ...
      }

      if( ... )
   }                       //end of while

// Switch Debouncing

   if(PB0flag == 1)
   {
      while(PB0 == 0);     //wait for PB release
      if((tcount < Tmax) && (randact == 0))score[i]+=1;     //add point for correct act
   }
   else if(...)
...
```

Demonstration and Verification

1. Complete the pseudo-code that describes the program operation
2. Complete the Pin-out form, labeling the Port bits used, *sbit* variables, and initialization values
3. Use the multimeter to demonstrate that the potentiometers are connected correctly.
4. Run your C program and demonstrate that it performs as stated above.
5. Tell a TA how you created T_{MAX}. Write a calculation in the notebook and report that estimates the worst-case error of the actual time compared to the equation in the lab description. Ex., for a T_{MAX} = 15,000ms, given your Timer0 integer rounding on the SYSCLK period, what is the actual time period.
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. Capture the screen of SecureCRT or HyperTerminal showing the output of one of the sample games (one possible example shown below) and take a print out. Attach this printout along with your code in the lab notebook.

SecureCRT Output Example

```
Set the speed post and press the pushbutton to begin LITEC Mastermind.
Starting Period: 25 seconds

Amber Player Turn

120     3    1    1 pt
102     3    0    4 pt
201     3    1    10 pt
021     3    0    16 pt
219     3    3    22 pt   (MATCH!)

Amber Points = 22

Green Player Turn

111     2    2    1 pt
110     3    1    5 pt
011     3    1    11 pt
101     3    3    17 pt   (MATCH!)

Green Points = 17

Amber Points = 22, Green Points = 17,
Winner is Green!
```

There is a handout on the top LMS page, Terminal_output_Guidelines {using printf()}, that explains output options to SecureCRT for overwriting lines (using \r without \n) and backing up the cursor. Following those guidelines will permit you to output to the terminal lines similar to those in the above example.

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 of the lab manual. There is also a game report rubric on LMS with Lab 2 (GradingGameReport_C8051-student). This report should cover the design, development, and operation 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 153 of the manual.

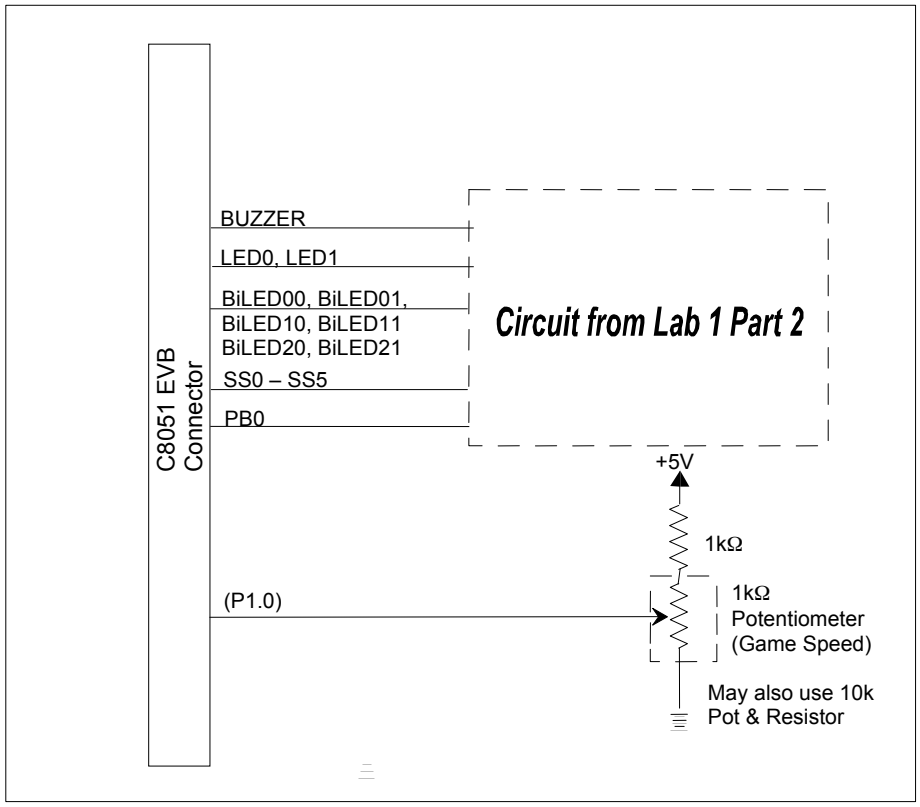

Figure 2.1 - Suggested hardware configuration for Lab 2
NOTE: Based on Lab 1, replace 10k pot with 1k pot and 10k resistor with 1k resistor to +5V and add two BiLEDs as well as regular LEDs. Several other components may be added or removed.

Figure 2.2 – Optional larger potentiometer used for the Game Speed Input.

Sample ADC 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
    putchar(' ');                   // Required for output to terminal
    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
}
```