Lab 2: A Microprocessor-Controlled Game – Tennis

Preparation

Reading

Lab Manual:

Chapter 2 - Lab Equipment

Chapter 4 - The Silicon Labs C8051F020 and the EVB (sections relating to Timer 0 and 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 and Logic Probe

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 LEDs, Pushbuttons and more as specified. Some rearrangement of the positions of the LEDs and Pushbuttons may be required.

Software

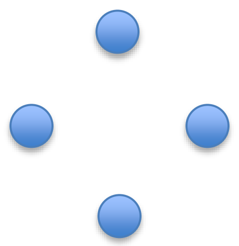
1. The speed at which a new random value is generated or other operations may be a function of the digital conversion read from the *game speed potentiometer*.
2. Further refinement of programming skills requiring bit manipulation.
3. Adaptation and re-use of software modules developed in previous lab 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 following labs—the *Smart Car & Gondola*. Portions of those labs 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 or other parameters, and the use of interrupts to keep track of time.

Lab Description & Activities

Teams with only 2 members may skip Mode 1 (Pattern Mode) and use preset values for blink rate and step period.



Description of Game Objective

There are three modes to the game, corresponding to the four LEDs shown above. (See the hardware description for determining how to select the modes.)

Mode 1 is 'Pattern Mode'. The LEDs will turn on and then off (blink) in a set of sequences,

Sequence 1: Top, right, bottom, left, top, left, bottom, right

Sequence 2: Top, bottom, left, right, left, bottom, top

Sequence 3: Left, right, left, right

The duration of each blink will depend on the Analog input varying from 0.25s to 1s as the input voltage varies from 0V to 2.4V. The input should be checked at the start of each sequence. There should be no delay between blinks, ie. in sequence 1, as soon as the top LED turns off, the right LED should turn on.

Mode 2 is 'Ping Pong' mode. The left player's LED will turn on for a period of time determined by the analog input, followed by the right player's LED turning on for the same period of time. During each players LED on time, that player will push their pushbutton as many times as possible. There will be a total of three rounds, after which the player with the most pushes wins the game. However, if a player pushes a button when it is not their turn, they automatically lose.

The duration of each player's turn (on-time) will depend on the Analog input varying from 1s to 5s as the input voltage varies from 0V to 2.4V. A player may change the duration at any time, so each round of the three rounds may have a different on-time. The Analog input should be checked at the start of each round, and the on-time set appropriately. There should be no delay between players turns.

Mode 3 is ‘Tennis’ mode. There will be four pushbuttons, one associated with each LED. The LEDs will start off blinking in a clockwise direction (top, right, bottom, left, top…). However, the starting LED is determined by a random number (0 to 3). Player one will push a button corresponding to an on LED and ‘reverse’ the direction of the LEDs to a counter-clockwise direction (top, left, bottom, right, top,….). Again, the starting LED is determined by a random number (0 to 3). Player two will then push a button corresponding to an on LED, to set the blink pattern back to a clockwise direction. If a player pushes a wrong button (one where the LED is not lit) or the LED has turned off during the push, that push does not count and the player continues their turn until they push and release a button while the LED is lit. The game should consist of five rounds. Scoring should keep track of the number of LED blinks for each player. The player with the least number of blinks wins.

The duration of each player’s turn (on-time) will depend on the Analog input varying from 0.25s to 1s as the input voltage varies from 0V to 2.4V. The input should be checked at the start of the game and be the same for all five rounds.

Description of Game Components

The components of the game consists of

1) 4 Push buttons
   a. The push buttons should be used as indicated for each mode.
   b. The ‘top’ pushbutton should used to start each game (mode), based on the slide switches. When a game has finished, the game should wait until the top pushbutton is pushed and then check the slide switches. The appropriate mode should 5s later.
2) 2 Slide switches
   a. If both slide switches are in the off position, none of the modes are selected and all LEDs should be off.
   b. If slide switch 1 is on and slide switch 2 is off, mode 1 is selected
   c. If slide switch 1 is off and slide switch 2 is on, mode 2 is selected
   d. If slide switch 1 is on and slide switch 2 is on, mode 3 is selected
3) 4 LEDs, used as indicated for each mode
4) 1 BiLED, used to indicate which player won the game in mode 2 and 3, green for player 1 and red for player 2.
5) 1 potentiometer for the analog input
6) Additional components as needed (resistors, chips, etc.)

Mode 1

```
Turn off all LEDs, and BiLED
Determine the number of counts corresponding to the on-time
Blink Sequence 1
Blink Sequence 2
Blink Sequence 3
Exit the mode
```

Mode 2 (1st player creates a sequence, 2nd player plays the game with that sequence)

```
Turn off all LEDs
Repeat 3 times
    Determine the number of counts corresponding to the on-time
    Turn on left LED for the on-time
        Count left button presses
        Check to see if the right button is pressed (right player loses)
    Make sure the left button is not still pressed when the LED on-time is finished
    Turn on right LED for the on-time
        Count right button presses
        Check to see if the right button is pressed (left player loses)
    Make sure the right button is not still pressed when the LED on-time is finished
Display score and determine winning player
Set the BiLED
```

Mode 3 (Random value lights LEDs, player must match pattern by adjusting the potentiometer)

```
Turn off all LEDs
Determine the number of counts corresponding to the on-time
Repeat 5 times
    Set a clockwise blink pattern
        Determine starting LED
        Repeat
            Blink the appropriate LED
            Check if button is pushed and released during on-time
        If button was pressed correctly,
            Add total number of LED blinks to the score
            Exit clockwise blink pattern
    Set a counter-clockwise blink pattern
        Determine starting LED
        Repeat
            Blink the appropriate LED
            Check if button is pushed and released during on-time
        If button was pressed correctly,
            Add total number of LED blinks to the score
            Exit clockwise blink pattern
Display score and determine winning player
Set the BiLED
```

Possible List of Tasks and Needed Functions (not necessarily a complete list)

You should look at this list and make team assignment to distribute the load equally.

- Function to read the potentiometer and set the on-time for each LED.
- Function to turn on the appropriate LED and turn off all other LEDs
- Function(s) to set the blink pattern (Mode 1)
- Function to count the number of presses during the LED on-time and make sure the presses only occur while the LED is on for the appropriate player (Mode 2)
- Function(s) to set the blink rotation (Mode 3)
- Function to set the starting LED (Mode 3)
- Function to check if a push button was pressed and released during an LED on-time (Mode 3)
- Function to set the BiLED based on the winning player (Modes 2 and 3)
- Functions to initialize all the ports used in the game.
- Task to wire up **and verify** the protoboard hardware. A test function (similar to the hardware test code for Lab 1-1) verify the operation of all inputs and outputs.

NOTE: All pushbuttons must be debounced for proper game operation. Timer 0 should be in 16-bit mode with a SYSCLK trigger (as in Lab 1-2).

.

Hardware

Figure 1 shows the hardware configuration for this lab. The potentiometer is connected in series with a 10kΩ resistor between +5V and Ground, with the middle pin connected to P1.1. All other port pin connections are up to the team. NOTE: no buzzer is needed in the game.

Remember

The neatness of wiring is important to debug the hardware.

The placement of LEDs and corresponding pushbuttons must be considered. Each button should be close to the corresponding LED where appropriate, and they shouldn't be too closely packed. For example, LEDs used as a bar graph in some games should be in a straight line.

A/D Conversion

You will need to write a C program to perform Analog-to-Digital conversion on one (or more) of the port pins. You can find an example of the code in the manual, in the appropriate section. While you are encouraged to use that code as a template, it is important that you understand the details in both the initialization function and the function that returns the A/D value.

Timing

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

Parallel Development

It is **not** necessary to wait until all software is complete before the code can be run and debugged. You are strongly encouraged to verify both the hardware and software incrementally by writing modular code to accomplish specific tasks, ie. read the analog input, set the LEDs, etc. (as summarized previously).

Demonstration and Verification

1. Use the multimeter to demonstrate that the potentiometer is connected correctly.
2. Run your C program and demonstrate that it performs as stated above.
3. Tell a TA how you created required time delays. Be prepared to show your calculations.
4. 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.

8. Submit your C code file to LMS.