Lab 1 (part 1): Getting Started - Digital Input/Output

Preparation

Reading

Lab Exercise Descriptions:

All of the hardware and much of the software developed for Lab 1 parts 1 and 2 will be utilized in *Lab 2: A Microprocessor-Controlled Game*. Since Labs 1 and 2 form an evolutionary sequence, you are encouraged to familiarize yourself with the objectives of both labs before you begin Lab 1.

Lab Manual:

Chapter 1 - Introduction

Chapter 2 - Lab Equipment

Chapter 3 - Programming in C

Chapter 4 - The Silicon Labs C8051F020 and the EVB (Input/Output Ports and Timers)

Chapter 5 - Circuitry Basics and Components

Chapter 8 - Troubleshooting

C language reference concepts:

Data types, declarations, variables, functions, the `while` looping structure, `if-then` statements, bitwise operations

Embedded Control Multimedia Tutorials

Hardware: Circuit Components

Hardware: Logic Probe

Objectives

General

1. Familiarization with the *laboratory computers*, the *SDCC*, and the procedure to develop a C program on the computer and run it on the *SiLabs* C8051 **EV**aluation **B**oard (EVB).

Preparation

1. Complete Worksheet 1 (binary/decimal/hexadecimal counting systems), Worksheet 2 (bitwise and bytewise logic operations), and Worksheet 3 (Port initializations and basic hardware)
2. Complete Homework 1 (software installation) and Homework 2 (initial pseudocode)

3. Develop pseudo-code that describes your program, observing formatting styles discussed in class
4. Complete the Pin-out form, specifying the Port bits required, the *sbit* labels chosen and the initialization settings for the appropriate SFRs.

Hardware

1. Familiarization with the protoboard, the use of the multimeter, and the logic probe.
2. Configuration of a LED on the protoboard to allow them to be switched on /off by one of the C8051's digital output ports.
3. Configuration of a bi-color LED to allow it to be switched on/off by a pair of the C8051's output pins.
4. Configuration of one pushbutton switch and one slide (toggle) switch on the protoboard to allow them to be read by one of the C8051's digital input ports.
5. Configuration of a buzzer to allow it to be turned on/off by one of the C8051's output ports.

Software

1. Introduction to aspects of modular program development in C, including the concepts of *top-down program development*, *scoping of variables*, and *C functions*.
2. Development of a modular C program that will enable the C8051 to acquire digital input from the switches connected to its input ports, and to light LEDs and to turn on buzzer connected to its output ports based on the digital input.

Motivation

The ability of a microprocessor to interface with other digital devices used as sources of *input* (e.g., switches, a keyboard, etc.), destinations for *output* (e.g., relays, motors, LEDs, etc.) or for *input and output* (e.g., another microprocessor) opens up a world of possibilities. Moreover, for a microprocessor to serve as the controller for a host system, it must be able to *acquire input* (digital or analog), and it must be able to *provide output* (digital or analog) in response to those inputs.

The three most common forms of digital ports are *input*, *output*, and *configurable input/output*. While the first two types of ports are dedicated strictly to the type of task implied by their names, I/O configurable ports may have some or all bits programmed via software instructions for either input or output. All of the ports on the C8051 are fully configurable.

In this lab, you will be introduced to digital input and digital output ports. With the attainment of the stated objectives, you will have gained an understanding of how to develop a simple C program for the C8051. You will be able to configure the C8051 to acquire a digital input from an external source and use this value to determine the output on one of its digital output ports.

Since you will apply much of what you develop in this exercise to later exercises, you are encouraged to develop the C code for this exercise so that it is easily extensible to future applications. Remember that one of the keys to software productivity is the ability to re-use software, and the *reusability* of software is a feature that must be designed-in from the start.

Lab Description & Activities

In this lab you will develop the components necessary for a portion of a microprocessor-controlled game. In this lab, you will connect a regular LED, a bi-color LED, 2 pushbuttons, a slide switch, and a buzzer. Additionally, you will create software for the EVB to read inputs from the pushbuttons and slide switch and produce outputs to the LEDs and buzzer.

Hardware

The pushbutton and slide switch will act as simple switches when connected in the manner shown in the circuit schematic. When the pushbutton is pressed or the slide switch is in the "off" position, the circuit is closed causing the voltage to drop across the resistor and a logic LOW 0[V] to be read at the EVB input pin. The pushbutton returns to a normally-open state when released causing a logic HIGH 5[V] to the input pin. The slide switch will have a logic HIGH at the input pin when the switch is in the "A" position because the switch is open and there is no voltage drop across the resistor. When the slide switch is moved to position "B", the input to the port bit is grounded (0[V]), corresponding to a digital logic LOW. These switches will be connected to the C8051's Port 2 and Port 3, as shown in the schematic.

The output signals from the EVB control the Buzzer, the LED, and the Bi-color LED. Similar to the description for input signals, output voltages of 0[V] and 5[V] are used to turn these devices 'on' and 'off'. Recall, a digital LOW 0[V] may be used to turn a circuit component 'on' and a digital HIGH 5[V] may be used to turn a circuit component 'off'. It is necessary to study the schematic when determine the correlation between 'on/off' states and digital voltage levels. It is important not to confuse these circuit voltages with the concepts of TRUE/FALSE used in the program. By studying the schematic, recognize that in order to turn the buzzer on, the output signal at P3.7 needs to be set LOW 0[V]. Similarly, the output at P3.6 must be a digital LOW 0[V] to turn the LED on. A digital HIGH 5[V] on these Port pins will turn the devices off. The control of a Bi-color LED requires 2 output bits. To turn the Bi-color LED OFF two possible output states are possible, both bits can be HIGH or both bits can be LOW. To turn it on with one of its colors, one bit must be HIGH, and the other must be LOW. To turn on the other color, the reverse is necessary. In the initialization routine of the software, the Port 2 and Port 3bits need to be configured for the appropriate input and output states.

Use the C8051 EVB Port Connector diagram on the back cover to determine the pin numbers that correspond to each of the input and output pins shown in *Figure E.1*. Write the pin numbers in the blanks next to each bit representation (e.g., bit 0 of Port 3, P3.0, is pin 38on the EVB connector).

Software

Modify the C program listed following the last page of this lab description so that it controls the state of the LEDs connected to Port 3 by reading the switches connected to Port 2 according to the following criteria (the program is available electronically on LMS under the Laboratory 1 section of Course Material with the name *lab1-1.c*):

1. When the Slide switch is 'off' (input is a HIGH voltage), LED0 is on, all other output devices are off
2. When the Slide switch is 'on' and both Pushbuttons are pushed, the Buzzer is turned on
3. When the Slide switch is 'on'and only Pushbutton 1 is pushed, the BiLED is green
4. When the Slide switch is 'on'and only Pushbutton 2 is pushed, the BiLED is red

Your program should also print the status of the Slide switch and Pushbuttons on the computer terminal screen. For example, if the Slide switch is 'on' and both Pushbuttons are activated, your program should print "Slide switch on, Pushbutton 1 and 2 activated" on the screen.

Remember, when setting an output Port pin in your initialization routine, you will want to do this without affecting the other pins on the same port - bitwise and unary operators such as "&" and "~" are useful for this task.

More information on bitwise and unary operators can be found in *Chapter 3 - Programming in C* of the Lab Manual.

You are required to develop your program in a modular fashion. Your "main" routine should be rather short, simply calling a series of C functions that in turn may call other C functions, etc. By employing C functions, try to avoid duplication of functionally equivalent or even similar chunks of code. Develop your program with comments and use functions that are easy to understand, which is helpful when someone else needs to refer to your code.

Writing Assignment - Lab Notebook

You and your lab partner should be keeping a Lab Notebook (one for each team), which documents the progress of your work in the lab. Be sure to read and follow the guidelines in *Appendix B- Writing Assignment Guidelines*, on page 117 for proper Lab Notebook formatting.

Grading-Preparation and Checkoff

The checkoff procedure for Laboratory 1-1 is available on LMS. Prior to the starting the laboratory you must complete

1) The appropriate Worksheets
2) The Pin-out form
3) The Pseudocode

When you are ready to be checked off, the TAs will be looking at the following items

4) That your project performs all the indicated requirements
5) Appropriately formatted and commented source code
6) Clean and neat hardware, with appropriate use of colors for source and ground connections

Additionally, you will be asked a number of questions. The questions will cover topics such as

7) Identify parts of software operations, understanding the hardware components, understanding the test equipment

The final item that will be included in the Laboratory grade is your

8) Notebook.

The above 8 items each have an individual contribution to your Laboratory grade.

Remember

Do not remove the circuitry built for Lab 1 - it will be used in Lab 2.

Record in your lab notebook the identification number of the protoboard you are using and attach a sticker to the bottom of it with your name and your lab partner's name!

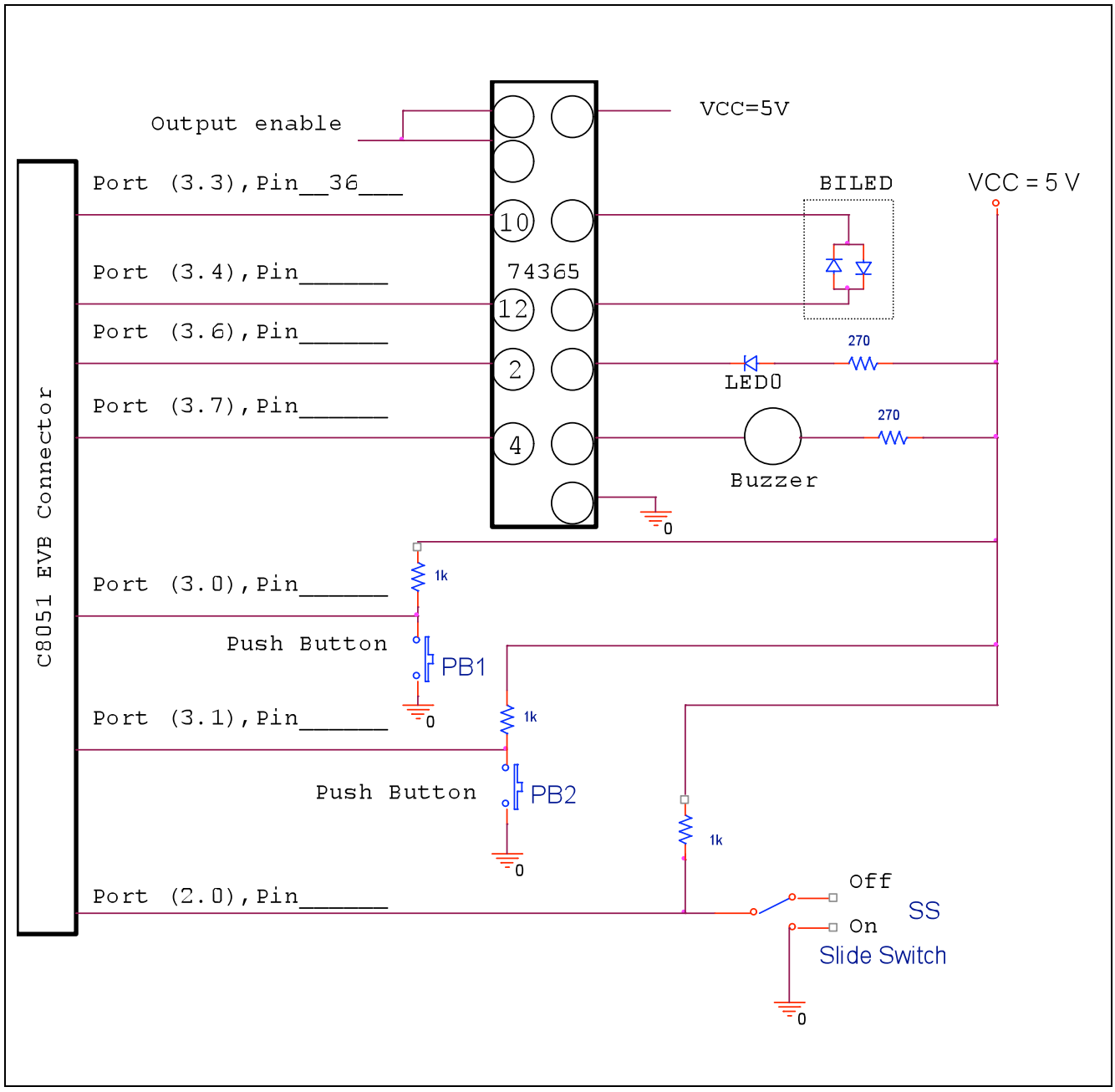

Figure 1-1.1 - Suggested hardware configuration for Lab 1 (part 1)

C Program for Lab 1 (part 1)

```
/*     Names:
       Section:
       Date:
       File name:
       Program description:
*/
/*
  This program is incomplete. Part of the code is provided as an example. You
  need to modify the code, adding code to satisfy the stated requirements. Blank
  lines have also been provided at some locations, indicating an incomplete line.
*/
#include <c8051_SDCC.h> // include files. This file is available online
#include <stdio.h>

//-----------------------------------------------------------------------------
// Function Prototypes
//-----------------------------------------------------------------------------
void Port_Init(void);  // Initialize ports for input and output
int sensor1(void);     // function which checks Pushbutton
int sensor2(void);     // function that checks the Slide switch
void Set_outputs(void);// function to set output bits

//-----------------------------------------------------------------------------
// Global Variables
//-----------------------------------------------------------------------------
sbit at 0xB6 LED0;     // LED0, associated with Port 3 Pin 6
//sbit _____________;   // BILED0, associated with ?????
//sbit _____________;   // BILED1, associated with ?????
//sbit _____________;   // Buzzer, associated with ?????
sbit at 0xA0 SS;       // Slide switch, associated with Port 2 Pin 0
sbit at 0xB0 PB1;      // Push button 1, associated with Port 3, Pin 0
//sbit _____________;   // Push button 2, associated with ?????


//***************
void main(void)
{
       Sys_Init();       // System Initialization
       putchar(' ');     // the quote fonts may not copy correctly into SiLabs IDE
       Port_Init();      // Initialize ports 2 and 3

       while (1)         // infinite loop
              {
                     // main program manages the function calls

                     Set_outputs();
              }
}


//***************
/* Port_Init - Initializes Ports 2 and 3 in the desired modes for input and output */

void Port_Init(void)
{
    // Port 3
//  P3MDOUT ______; // set Port 3 output pins to push-pull mode (fill in the blank)
//  P3MDOUT ______; // set Port 3 input pins to open drain mode (fill in the blank)
//  P3 _________;  // set Port 3 input pins to high impedance state (fill in the blank)
```

```
    // Port 2
    // configure Port 2 as needed
//
//
}

//***************
// Set outputs:
//    The following code is incomplete, lighting an LED depending
//    on the state of a single pushbutton.

void Set_outputs(void)
{

       if (sensor2())         // if Pushbutton activated
              {
                     LED0 = 0;  // Light LED
                     printf("\rSlide switch is off \n");
              }

              else             // if Pushbutton is not pressed
              {
                     LED0 = 1;  // turn off LED
                     printf("\rSlide switch is on \n");
              }

}
//***************
// Sensor - Returns a 0 if Pushbutton 1 not activated
//          or a 1 if Pushbutton 1 is activated.
//          This code reads a single input only, associated with PB0
// Note this code is not used by function yet, you must incorporate it
int sensor1(void)
{

       if (!PB1) return 1;
       else      return 0;
}
//***************
// Sensor - Returns a 0 if Slide switch is 'off'
//          or a 1 if Slide switch  is 'on'
//          This code reads a single input only, associated with SS
int sensor2(void)
{

       if (!SS) return 1;
       else     return 0;
}
```

Recommended Procedure for Laboratory Development

When developing the code and hardware for the laboratories, debugging mistakes typically accounts for the bulk of the work. One classic mistake made by new project designers is to attempt to implement the complete system and then fix errors. However, for a fairly large project, it becomes problematic to identify the sources of error when there are hundreds of lines of code or dozens of hardware components. To avoid this situation, it is much better to develop your project in steps, verifying operation at appropriate junctures. For Laboratory 1, a suggested development cycle has been provided to give you an idea of how to improve your efficiency and avoid time-consuming mistakes. In future labs, some basic recommendations are provided, it will up to you to determine the details of the development process.

1) Complete the hardware aspect of Worksheet 3.
   This procedure is required to prepare for Laboratory 1, and is mentioned since it represents a good example of scaling down the Laboratory. Having finished the worksheet, you will have wired a buffer chip, the slide switch, the LED and the Buzzer. At this point, if things are functioning as directed, you are confident the buffer chip outputs to the LED and Buzzer are functional and no further change is required.
2) Download and compile the code labeled *lab1-1.c* from LMS. Your goal is to verify that no syntax errors exist in the code before performing any edits.
   Key sections of the code are commented out and it is incomplete with regard to completing Laboratory 1. However, the code is sufficient to control the LED with the pushbutton.
3) Fill in the missing pin labels of Figure 1-1.1 for both the EVB connector and the buffer chip.
   Refer to the back of the manual and the explanation of the buffer chip.
4) Edit the initialization section of the code to configure Port 2 and Port 3 for the appropriate I/O operations indicated in Figure 1-1.1.
   Use the tools developed in Worksheet 3 to set bits to 1 or 0 as needed. Make sure that you do not change the other bits. Remember the assignment operations for this procedure, make use of the `&=` and `|=` syntax operations. You will need to complete the existing Port 3 commands and add the Port 2 commands. Also, you will need to remove the comments once you complete the Port 3 lines.
5) Connect the Slide Switch to the appropriate EVB input for SS.
   You will need to remove the wires in Worksheet 3 that connect directly from the Slide Switch to the Buffer chip. Refer to Step 3 to identify the pin number that is associated with this input.
6) Connect the appropriate EVB output pin for LED0 to the input on the buffer chip.
   Again, you need to remove the existing wire that connects to the buffer for LED0.
7) Add print statements to the output portion of the code, indicating the status of the slide switch.
   This requirement is a debugging procedure. It allows the user to test the input aspects of the code independently from the output portion.
8) Download the code to the microcontroller and run the program.
   If the hardware and software are correct, the LED should turn ‘on’ and ‘off’, depending on the Slide switch position. This step verifies the functionality of both the hardware and software while minimizing the amount of code or wire connections that need debugging. It is highly recommended that your initial efforts are as simple as possible so that mistakes are easily corrected.

9) Add the hardware for the two Pushbuttons and functions to determine their states.
10) Refine the output code to print the status of all three input devices (SS, PB0 and PB1).

    Following success with the simpler circuit, continue to develop the laboratory in a stepwise fashion. Adding two input devices and checking their status requires implementation of the circuit components and refinement of the code the controls the output status. The additional 'if' statements set the stage for controlling the outputs.
11) Once again, download the code to the microcontroller and verify operation.
12) Add the buzzer hardware and refine the code that sets the Port pins controlling the output devices.
13) Add the BILED hardware and refine the code that sets the Port pins controlling the output devices.

    Steps 9-13 should be an iterative process based on the procedure you performed in Steps 5-8. Again, it is highly recommended that you do not attempt to complete a massive section of code of hardware without incremental steps. You will find the debugging much easier by take smaller, incremental steps.
14) Prepare for checkoff.