1300 Henley Court
Pullman, WA 99163
509.334.6306
www.store.digilentinc.com

Lab 1b: Microprocessor I/O

Revised March 9, 2017
This manual applies to Unit 1, Lab 1b

1 Objectives

1. How to develop a plan for a software-based microprocessor design.
2. Understand the management of basic microprocessor digital inputs and outputs.
3. Write an application that utilizes the Basys MX3 processor platform to perform basic calculator functions and display the results on a 4-digit 7-segment LED display.
4. Become familiar with the circuits included on the Basys MX3 trainer board.

2 Basic Knowledge

1. Fundamentals of digital logic.
2. How to interpret a schematic diagram and electric circuits.
3. How to write a computer program using the C language.
4. How to launch a Microchip MPLAB X project.
5. How to configure the PIC32 I/O pins for digital input and output (See Lab 1a).

3 Equipment List

3.1 Hardware

1. Basys MX3 trainer board
2. Standard USB A to micro-B cable
3. Workstation computer running Windows 10 or higher, MAC OS, or Linux

In addition, we suggest the following instruments:

4. Digilent Analog Discovery 2

3.2 Software

The following programs must be installed on your development workstation:

1. Microchip MPLAB X® v3.35 or higher
2. XC32 Cross Compiler
3. PLIB Peripheral Library

4 Project Takeaways

1. Know how to generate a microprocessor development project using MPLAB X.
2. How to solve an embedded design problem using engineering methodologies.
3. How to perform a basic timing problem.

5 Fundamental Concepts

This lab builds on the lessons learned in Lab 1a to solve a more complex static I/O problem. The basic two software elements of all embedded designs, namely initialization and continuous execution of a process loop, are used once again. A simple delay loop is introduced to generate a stable and persistent output to allow humans to read a visual display.

6 Problem Statement

The objective of this lab is to program the PIC32 processor to implement the functionality of a simple integer calculator that displays the result of X • Y where the operator, •, will be add, subtract, multiply, or divide. The most recently computed result is continuously displayed on the 7-segment LEDs such that if an operation push button is held depressed and the X or Y switches change position, the new value is immediately seen on the LED display.

7 Background Information

7.1 Control Registers

There are two methods to individually set or clear writable registers, such as the TRIS and LAT registers. One is to write a sequence of instructions know as read-modify-write (RMW) operations, as was done in Lab 1A. The second method is to use bit-modifying instructions.

Using read-modify-write sequences introduces opportunities for errors when the programs are using interrupts and high capacitive pin loads.[1] In addition to the register assignments, such as *LATA = 23;*, that result in all 16 bits of the register being set at one time, there are modifier instructions that provide bit control. The modifiers that provide atomic bit manipulation are SET, CLR, and INV. Atomic instructions are those that cannot be divided by interrupts, avoiding the errors potentially introduced by RMW operations.

For the SET modifier, the bit positions that are set to a one in the instruction will set the register bits high, while not altering the bit in the unspecified positions. For example, the instruction *LATASET = 0x8020;* will result in bits 5 and 15 being set to a one. Bits 0 through 4 and 6 through 14 will not be changed by this instruction. The *LATACLR* instruction will set only the specified bits to zero. Likewise, the LATAINV instruction will invert only the specified bits.

7.2 Using the Slide Switches

[1] Read Modify Write Problem, https://download.mikroe.com/documents/compilers/mikroc/pic/help/rmw.htm

The disparate assignments of processor I/O pins connected to the eight slide switch inputs makes it imperative to use bit reading instructions. The instruction "*s0 = PORTFbits.RF3;*" is one way of sensing the state for slide switch SW0. To generate a 4-bit binary encoded for the value X as a 4-bit integer value, the following C programming instruction can be used:

$$X = \text{PORTFbits.RF3} + (\text{PORTFbits.RF5} \ll 1) + (\text{PORTFbits.RF4} \ll 2) + (PORTDbits.RD15 \ll 3);$$

Similarly, an expression can be created that will generate a value for another variable, Y, using SW4 through SW7.

7.3 Using the Push Button's Inputs

As with the slide switches, the push buttons must be individually set as PIC32 inputs. BTNR, BTNL, and BTNU are analog pins, and hence require disabling the analog input. Remember that PIC32 I/O pins that also function as analog inputs must have the appropriate ANSELx bit set low. The state of the BTND push buttons can be read using the following C statement:

$$BTND = \text{PORTAbits.RA15};$$

Figure A. shows that these five processor signals all have alternate uses. BTNR, BTND, and BTNC can be used as outputs for controlling servo motors and triggering the Analog Discovery 2. The 10K ohm resistors in series with the push buttons allow the buttons to be pressed with no effect on the PIC32 when these pins are configured as outputs.

BTNL and BTNU are also used for MPLAB debugging and processor programming that renders them unavailable for push button inputs while the MPLAB X debugger is running. The PIC32 must be programmed for standalone operation to allow using BTNL and BTNU for inputs to the PIC32.

7.4 Using a 7-Segment Display

As previously discussed for the slide switch and push button inputs, the 7-segment LED controls are scattered over multiple ports and pins. Again, Table E.1 of Unit 1 must be used to identify those pins that need to have the analog functionality disabled. Each segment should be set on or off using the LATxSET = BIT_xx and LATxCLR = BIT_xx instructions.

The particular multiple digit 7-segment LED display used in this application is a common anode device. The cathodes for the seven LED segment and the decimal point (DP) are active low, meaning the output pin must be set low to activate the LED segment. In addition, one of the four anodes (AN0 through AN3) must also be set low to allow the LEDs set by CA through CG to be turned on for the selected digit. For example, to display the number "5" on LED display number two, we would use the instructions shown in Listing 7.1.

Listing 7.1. Bit-banging Instructions to Display the Number "5" on the Third Digit

```
LATGCLR = BIT_12;   // Segment A on
LATASET = BIT_14;   // Segment B off
LATDCLR = BIT_6;    // Segment C on
LATGCLR = BIT_13;   // Segment D on
LATGSET = BIT_15;   // Segment E off
LATDCLR = BIT_7;    // Segment F on
LATDCLR = BIT_13;   // Segment G on
LATGSET = BIT_14;   // Segment DP off
LATBSET = BIT_12;   // Display 0 off
LATBSET = BIT_13;   // Display 1 off
LATACLR = BIT_9;    // Display 2 on
```

```
LATASET = BIT_10;   // Display 3 off
```

Note: BIT_xx instructions used above are defined in port.h.

Variations of the code in Listing 7.1 must be used to light different numbers on each of the four digits. The last four instructions select the display digit.

7.5 Persistence: Lighting the 7-Segment Display

The display in Fig. 8.2 appears to have LED segments in three digits simultaneously displayed. As just discussed, the cathodes of all four of the 7-segment LEDs are connected in parallel and the particular digit is selected by asserting the common anode pin high. In order to make the 4-digit display appear that all four LED digits are simultaneously on, one should make sure each digit is displayed for a minimum of 1 millisecond. Listing 7.2 provides a simple software delay. Since this delay is dependent on the speed of execution, the “ms_cnt” will need to be changed depending on the speed of the core frequency. For the example shown in Listing 7.2, the ms_cnt value of 1000 is set to generate a one ms delay when the core frequency is set to 80 MHz.

Listing 7.2. Software Time Delay

```
void sw_delay(int ms)
{
unsigned int ms_cnt;        // 1 ms counter
    while(ms--)             // ms loop counter
    {
        ms_cnt = 10000;     // Reset ms counter
        while(--ms_cnt);    // wait for 1 ms
    }
}
```

8 Lab 1b

8.1 Development Plan

Follow the system design plan presented in Unit 1 by completing the actions described in steps 1 through 5 of Table 8.1. After the planning activity has been completed, proceed to the design phase by setting up your development environment. Note that a test plan is included in the list of activities so that any test hardware and or software is included in the overall design.

Table 8.1. Software based system work flow.

Step	Action	Phase of project test to be completed	Test Metrics – How is test to performed	Results
1	Problem statement	Initial action	Peer review	0 = Not adequate 10 = very precise
2	Software Plan	Prior to initiating coding effort	Control Flow Diagram Collaboration Diagram	0 = Vague – not complete 10 = Complete
3	Testing Plan	Prior to initiating coding effort	Identify instrumentation methods	0 = Vague – not complete 10 = Complete

4	Resource list – hardware and software	After understanding of work to be completed	Peer review	0 = Not adequate 10 = Complete
5	I/O Table	Prior to initiating coding effort	Peer review	0 = Vague – not complete 10 = Complete
6	Code Development	Start of software development	Compiles with no errors	Pass / fail
7	Functionality Testing	After code development	Operates according to specification	Pass / fail
8	Documentation	After Testing	Sufficient information to allow work to be independently verified.	0 = Vague – not complete 10 = Complete

8.2 Project Requirements

Using the five push buttons and eight slide switches on-board the Basys MX3, program the PIC32 processor to implement the functionality of a simple integer calculator that displays the result of X • Y where the operator, •, will be add, subtract, multiply, or divide. The button and switch operation assignments are given below. The most recently computed result is continuously displayed on the 7-segment LEDs such that if an operation push button is held depressed and the X or Y switches change position, the new value is immediately seen on the LED display. Only one push button is allowed to be pressed at a time. If multiple push buttons are simultaneously pressed, the result will not change.

Table 8.2. Button operation.

Push Button	Mathematical Operation
BTND	ADD: Result = X + Y
BTNC	SUBTRACT: Result = X – Y
BTNR	MULTIPLY: Result = X * Y
BTNU (1)	DIVIDE: Result = X / Y
BTNL (1)	CLEAR RESULT: Result = 0

Note 1: BTNU and BTNL can only be implemented when MPLAB is not in debug mode (see hardware.h in Unit 1, Appendix C).

The eight slide switches shown in Figure 8.1 are divided into two groups of four switches to generate values for X and Y by setting the switches in a binary fashion. The X and Y values are determined by equations 1 and 2.

$$X = (SW0 \cdot 2^0) + (SW1 \cdot 2^1) + (SW2 \cdot 2^2) + (SW3 \cdot 2^3) \qquad \text{Eq. 1}$$

$$Y = (SW4 \cdot 2^0) + (SW5 \cdot 2^1) + (SW6 \cdot 2^2) + (SW7 \cdot 2^3) \qquad \text{Eq. 2}$$

Where SW0 through SW7 has a value of one if in the on (up) position, or zero if in the off (down) position.

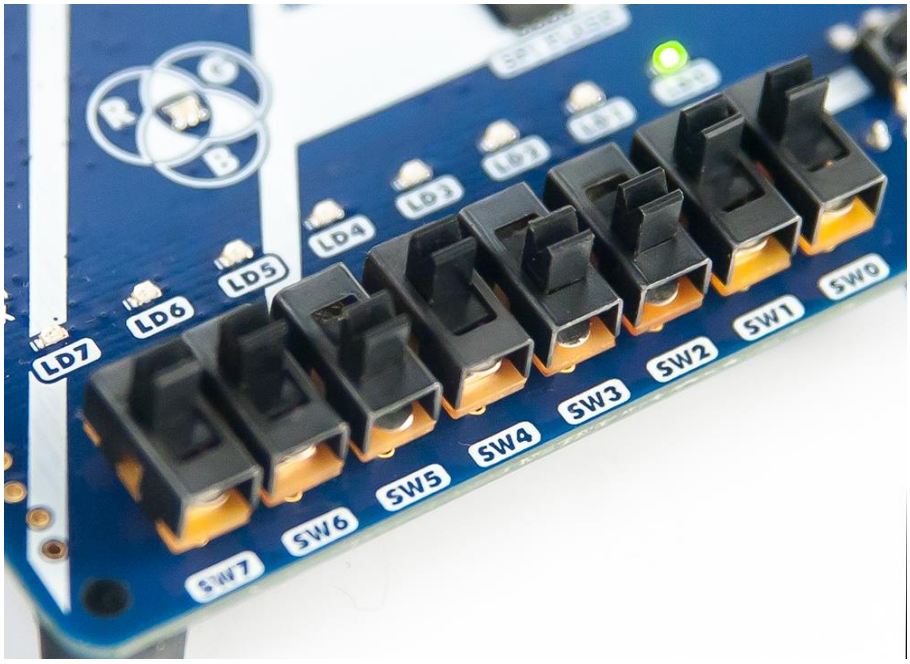

*Figure **8**.1. Partitioning of switches assigned to variables X and Y.*

The mathematical operation on variables X and Y produce a new result that is displayed on the 4-digit 7-segment display, as shown in

Figure 8.1, whenever one of the push buttons is pressed. The operations associated with the push buttons are shown in the green text in

Figure 8.2. (Note: BTNL and BTNU are not functional when MPLAB X is running in debug mode.)

*Figure 8.1**.** 4-digit 7- segment display.*

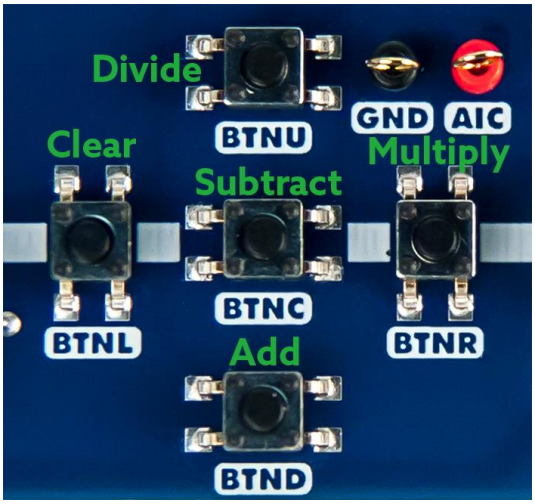

*Figure 8.2**.** Operator assignments to push buttons.*

8.3 Design Phase

1. Generate a work flow table as shown in Table 8.1.
2. Write a problem statement that lists the requirements.
3. List the resources that you will be using in this design.
4. Sketch a flow diagram for this program that identifies what is to be done in the initialization section and what will be done in the infinite loop.

5. Write a test plan that you will use to validate your software design (see above).
6. From the schematic diagram for the Basys MX3 trainer board:
    a. Determine and record the port and bit numbers assigned to switch outputs for SW0 through SW7.
    b. Determine and record the port and bit number assigned to the five push buttons.
    c. Determine and record the eight port and bit number assigned to the cathodes CA-CG and DP of the 7-segment LED module.
    d. Determine and record the four port and bit number assigned to the anodes AN0-AN3 of the 7-segment LED module.
    e. Determine which pins can be set as analog input or digital I/O.

8.4 Construction Phase

1. Create a new project for the Basys MX3 trainer board. Name this project Lab1b.
2. Copy the “config_bits.h” file to the project folder.
3. Add the “config_bits.h” file to the project list of Header Files.
4. Create a new “mainfile” to the project. Name this file “main.c”.
5. Add “config_bits.h” to the top of main.c.
6. Create a function called main by writing “int main(void) { … }
7. Within the initialization section of the main function:
    a. Declare local variables “x” and “y” initialized to a value of zero in the “main” function.
    b. Set the PIC32 I/O pins connected to the switch and push buttons as inputs.
    c. Set the PIC32 I/O pins connected to the 4-digit 7-segment module as outputs.
8. Write and test a function that reads the eight slide switches and returns values “x” and “y.”
9. Write and test a function that determines if a push button is pressed and performs the required mathematical operation. This function returns the result as a signed integer.
10. Write and test a function that displays the result on the 7-segment display with the required persistence.
11. Integrate the functions developed in steps 13-15 into a continuous loop in the “main” function.

8.5 Testing

1. Test the Add operation using the minimum and maximum values for X and Y.
2. Test the Subtract operation using the minimum and maximum values for X and Y.
3. Test the Multiply operation using the minimum and maximum values for X and Y.
4. Test the Divide operation using the minimum and maximum values for X and Y.

9 Questions

1. What is displayed on the 7-segment LED for the operation X / Y when Y is set to zero?
2. What does the 7-segment display show if all anode outputs (AN0 through AN3) are pulled low at the same time?
3. Using schematic diagram of the Basys MX3 and Figs. 31-1 presented in Reference 2 in section 10, when Vdd is 3.3V:
    a. Estimate the output voltage when RA0 pin is set high.
    b. Estimate the current output when RA0 pin is set high assuming that the forward diode voltage drop is 0.7V.
4. If RB8 is programmed to be a digital output, what affect will pushing BTNR have on the output signal?
5. The signal BTNC is connected to PIC32 Port F pin 0 and programmed to serve as a digital input or output.

   a. Consider the case when pin RF0 is programmed to be an input and the signal TRIG_1 is a digital output signal from an external device that is either 0 or 3.3V. What effect will pressing the BTNC button have on the processor RF0 pin when TRIG_1 is low and when it is high?
   b. Consider the case when pin RF0 is programmed to be an output and the signal TRIG_1 is a digital input signal to an external device. What effect will pressing the BTNC button have on the TRIG_1 signal when the RF0 output is low and when it is high?
6. Measure the loop execution time by toggling Pmod connector JA pin 1 at the start of the loop execution using the instruction LATCINC = 0x04. Connect a logic analyzer or oscilloscope probe to JA pin 1 and the ground or common connector to JA pin 5. The loop time is one half the period of the square wave generated at JA pin 1.
   a. What is this time?
   b. Is this time expected?

10 References

1. Best coding practices, https://en.wikipedia.org/wiki/Best_coding_practices
2. PIC32MX330/350/370/430/450/470 Family Data Sheet

Appendix A: Schematic Drawings

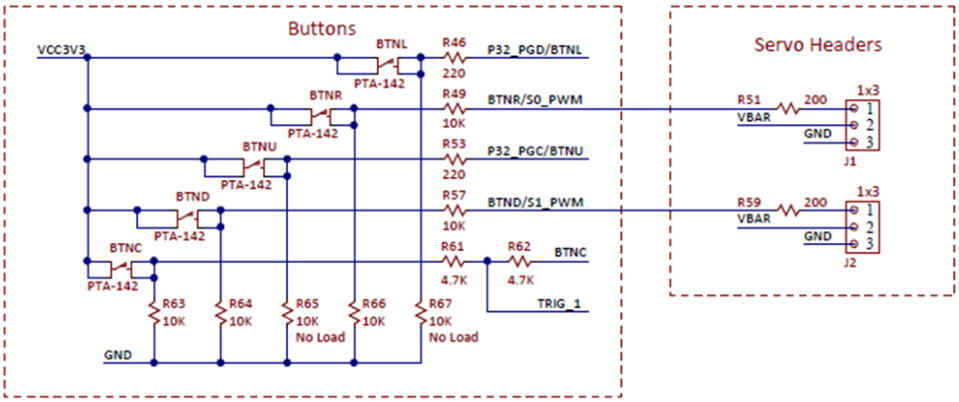

Figure A.1 Push button schematic diagram.

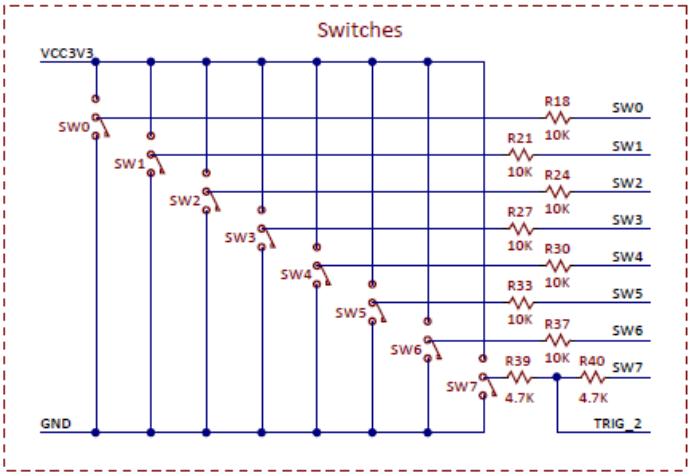

Figure A.2 Schematic diagram of Basys MX3 slide switches.

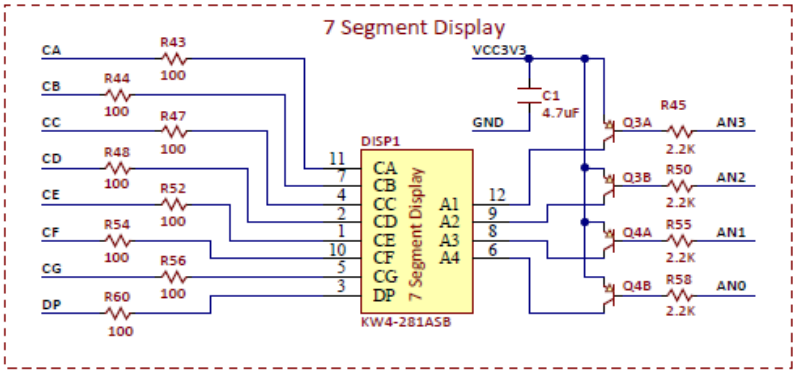

Figure A.3. 4-digit 7-segment display schematic.