TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

Module 8

Lab 8: Interfacing Input and Output

Lab 8: Interfacing Input and Output

8.0 Objectives

The purpose of this lab is to learn how to interface a switch and an LED to a microcontroller using TI's Launchpad development board.

1. You will first build the circuits on a breadboard and perform explicit measurements in order to verify they are operational and to improve your understanding of how they work.
2. You will then design your own **Window intruder detector alarm system** using this knowledge.

Good to Know: The switches interfaced in the lab will become bump detectors on the robot. The robot will use LED output as a debugging tool for you to visualize what the software is doing.

8.1 Getting Started

8.1.1 Software Starter Projects

Look at these example projects: **InputOutput** (input/output of switches and LEDs on the LaunchPad), **GPIO** (simple output to four pins), **Switch** (software driver of an input switch), and **Lab08_Switches_LED** (starter project for this lab)

8.1.2 Student Resources (in datasheets directory-Links)

B3F-1052.pdf Switch Datasheet
HLMP-4700.pdf LED Datasheet
CarbonFilmResistor.pdf resistor data sheet
MSP432P4xx Technical Reference Manual (SLAU356)
Meet the MSP432 LaunchPad (SLAU596*)*
MSP432 LaunchPad User's Guide (SLAU597*)*
MSP432P401R Datasheet, msp432p401m.pdf (SLAS826)

8.1.3 Reading Materials

Chapter 8, "Embedded Systems: Introduction to Robotics"

8.1.4 Components needed for this lab:

All the components needed for this lab are included in the TI-RSLK MAX Kit (TIRSLK-EVM). For this lab you will need to remove the MSP432-LaunchPad carefully from your robot and gather all the other components.

Quantity	Description	Manufacturer	Mfg P/N
1	MSP-EXP432P401R LaunchPad	TI	MSP-EXP432P401R
1	Red 2mA 5mm diffused LED	Avago	HLMP-4700
1	Carbon 1/6W, 5%, 470Ω	Yageo	CFR-12JB-470R
3	B3F tactile push button switches	Omron	B3F-1052
1	solderless breadboard	Pololu	4000

8.1.5 Lab equipment needed

Voltmeter
Oscilloscope (one channel at least 10 kHz sampling)
Logic Analyzer (4 channels at least 10 kHz sampling

Lab 8: Interfacing Input and Output

8.2 System Design Requirements

In this lab you will design, develop and test a window intruder detector alarm system. As shown in the block diagram of Figure 8.1, our simple window intruder detector alarm system has three inputs and one output. The inputs are three switches are implemented with positive logic, see Figure 8.2. The first switch input is called **Activate**, which serves as the arm/disarm control. There are two window sensors, called **Window1** and **Window2**. When **Activate** is pressed or true, the security system is activated. When **Activate** is not pressed or false, the system is deactivated, meaning the alarm will be OFF regardless of the state of the window sensors. The window is in a secure position when the window sensor is pressed or true. It is unsafe if either window sensor is not pressed. The output is a LED called **Alarm,** which is implemented in positive logic. You will flash the LED at 5 Hz (on for 100ms, off for 100ms) to signify the unsafe condition. In other words, the LEDs should blink rapidly when an intruder is detected by the sensors **Window1** or **Window2**. You will connect these switches and LED to your breadboard and interface them to your MSP432 LaunchPad development board based on the truth table shown in Table 8.1.

Note: If you are using the TI-RSLK MAX kit (TIRSLK-EVM) you will have the modified TI'S MSP432 LaunchPad development board with headers soldered onto J5 and with the 5V jumper disconnected. This can be used in all labs.

Activate switch	Window1 sensor	Window2 sensor	Alarm (LED)
OFF	X	X	OFF
ON	Not Pressed	Not Pressed	Flash at 5 Hz
ON	Not Pressed	Pressed	Flash at 5 Hz
ON	Pressed	Not Pressed	Flash at 5 Hz
ON	Pressed	Pressed	OFF

Table 8.1. Truth Table for the Window Intruder detector alarm system.

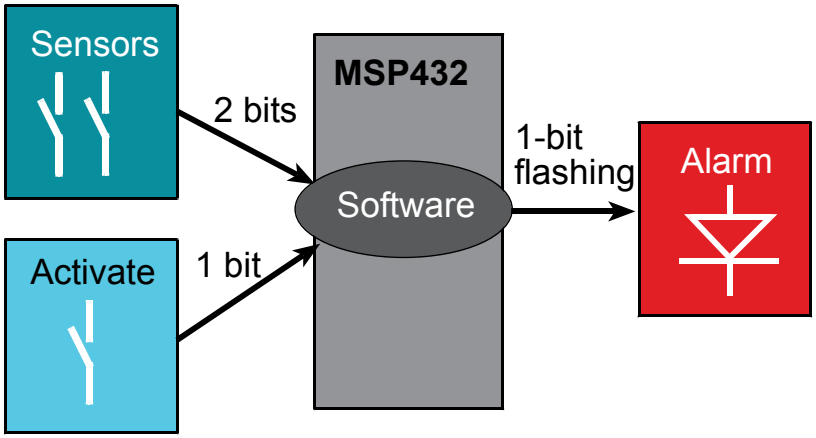

Figure 8.1. Window Intruder Detector Alarm System.

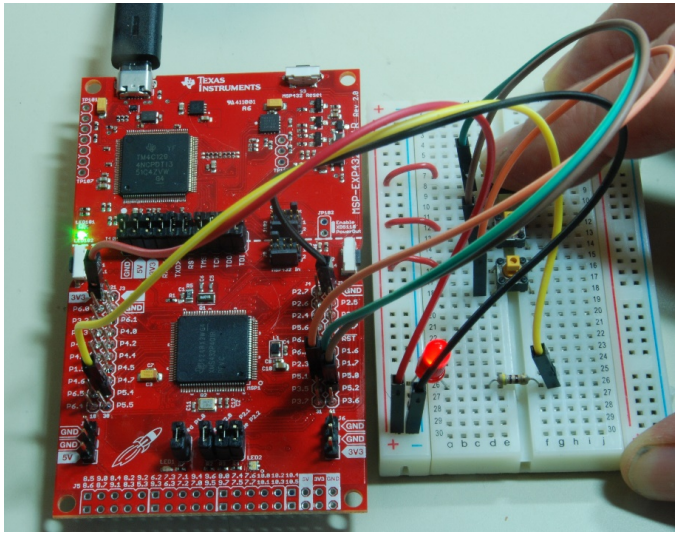

Figure 8.2.MSP432 LaunchPad (non-modified) and external circuits

Lab 8: Interfacing Input and Output

8.3 Experiment set-up

8.3.1 Switch interface

You will eventually need three switches. However, you will begin by interfacing one switch to the TI LaunchPad development board. Figure 8.3 shows a possible way to connect the switch to the microcontroller. You can use an internal or external pull-down resistor to make the voltage on the pin zero when the switch is not pressed. . We recommend you remove the LaunchPad from the robot.

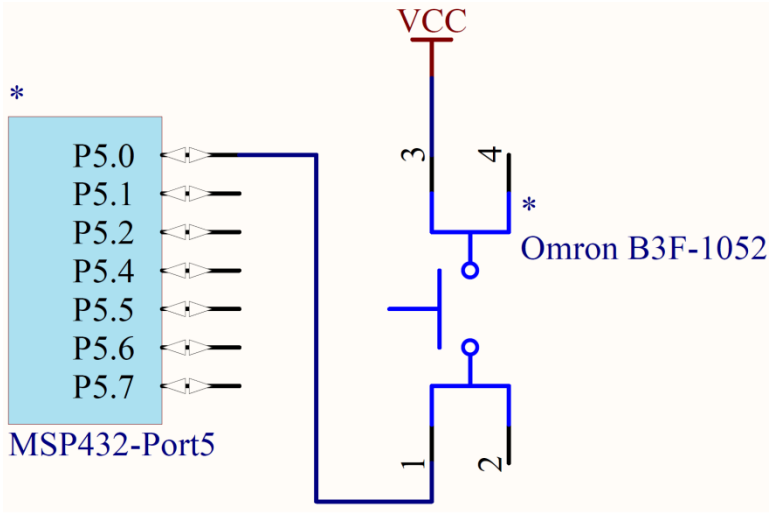

Figure 8.3 . Interface connecting the switch to P5.0 using internal pulldown. (CircuitMaker).

In order to standardize timing throughout the class, we will activate the external crystal and run at 48 MHz.

Warning: *Limit the current into and out of port pins to be less than 6 mA. One very bad way to build the switch interface is to place one side of the switch to +3.3V and the other side to ground, causing a 3.3V to ground short whenever the switch is pressed.*

Hint: Sample code to interface one switch with internal pull ups using P5.0

```
uint8_t sensor;
int Program8_1(void){
  Clock_Init48MHz();  // makes bus clock 48 MHz
  P5->SEL0 &= ~0x01;  // configure P5.0 GPIO
  P5->SEL1 &= ~0x01;
  P5->DIR &= ~0x01;   // make P5.0 in
  P5->REN |= 0x01;    // enable pull resistor on P5.0
  P5->OUT &= ~0x01;   // P5.0 pull-down
  while(1){
    sensor = P5->IN&0x01; // read switch
  }
}
```

While **Program8_1** is running, use a voltmeter to measure the voltage on the pin when the switch is not pressed and when the switch is pressed. You should get 0 V when the switch is not pressed, and you should get 3.3 V when the switch is pressed. Compare the voltage on the pin to the value in the software after the input is read. Use the debugger to observe the global variable **sensor**. You will eventually expand the system to have the three inputs, and the software will read the status of the three switches. We purposely gave this example with one switch, knowing you will need to modify the hardware and software to input from three switches. You will find a **Program8_2** in the project, which is similar to 8_1, but activates the **TExaS** logic analyzer.

Lab 8: Interfacing Input and Output

8.3.2 LED interface

Look up the desired (V_f, I_f) operating point of your LED used for this lab using its datasheet. Assuming the microcontroller pin is 3.3V calculate the resistor needed to obtain that operating point. Choose a standard resistor value near that value (e.g., 100, 220, 270, 330, 470, 680, 820, or 1000 Ω). Decide which pin you will use for the LED output, and draw the interface circuit for your LED, similar to Figure 8.4. You will have to redraw Figure 8.4 moving the LED to a pin not used by the switches.

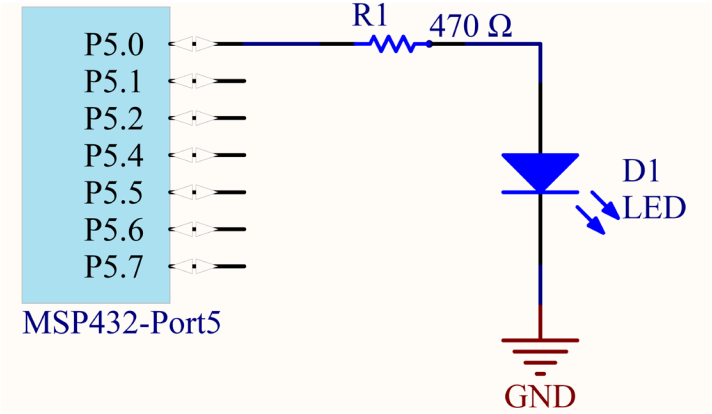

Figure 8.4. Example interface connecting the LED output to P5.4. (CircuitMaker)

Perform LED measurements with four resistance values such as the 220, 470, 680, 1000 ohms provided in the kit. When the software outputs a high (assume 3.3 V output), estimate the current through and the voltage across the LED given the circuit you have built. However, when measuring the actual LED current and voltage, you will need to **single step**, because if you run, the pin will oscillate around a million times per second, and the LED will look dim.

Modify **Program8_3** so the microcontroller makes the appropriate pin an output. For example, if you connect the LED to P5.4, you will have to edit it so Port5 bit 4 is an output, and so the main loop oscillates bit 4. We purposely wrote **Program8_3** using the same pin as we used for input in the previous example, knowing you would need to modify this program to output to the specific port pin to which you connected your LED. The program should simply toggle the LED on and off.

Notice that the LED operations in **Program8_3** are written as a **software driver**, which is a set of functions that facilitate LED operation. Furthermore, see how the LED functions form an abstraction, separating what it does (LED initialization, turn on, turn off, toggle) from how it works (P5 pin 0).

Run your modified **Program8_3** and single step it until the LED is on. Measure the voltage across the resistor and the voltage across the LED. Single step the software until the LED is off and measures the two voltages again. Use Ohm's Law on the resistor to calculate the current through the resistor. The resistor current will also equal the LED current. Compare the actual (V_f, I_f) operating point of the LED with the expected values calculated during design. Measure the voltage on the microcontroller pin when the software is stopped with the output of the microcontroller is high. Compare this measured voltage to the expected value of 3.3 V.

Hint: Use this program to test the LED interface

```
void LED_Init(void){
  P5->SEL0 &= ~0x01;  // configure P5.0 GPIO
  P5->SEL1 &= ~0x01;
  P5->DIR |= 0x01;    // make P5.0 output
}
void LED_On(void){
  P5->OUT |= 0x01;  // turn on
}
void LED_Off(void){
  P5->OUT &= ~0x01; // turn off
}
void LED_Toggle(void){
  P5->OUT ^= 0x01;  // change
}
int Program8_3(void){
  Clock_Init48MHz();  // makes bus clock 48 MHz
  LED_Init();         // activate output for LED
  while(1){
    LED_On();
    LED_Off();
  }
```

Lab 8: Interfacing Input and Output

8.3.3 LED toggling

Next, you will develop and test software that flashes the LED 5 times/sec. Basically, you need to write the Wait1ms() function. In the next lab, we will use the SysTick timer for delays. However in this lab, you could use two *nested for loops*. Software loops are very inaccurate for time delays. So, in this lab, the delay may be any value such that the LED flashes anywhere from 4 to 6 times per second. Use a stopwatch, logic analyzer, or oscilloscope to verify the LED flashes at the desired rate.

Figure 8.5 shows Program 8.4 running with the Logic Analyzer active. To activate the logic analyzer to visualize Port 5, execute

```
TExaS_Init(LOGICANALYZER_P5);
```

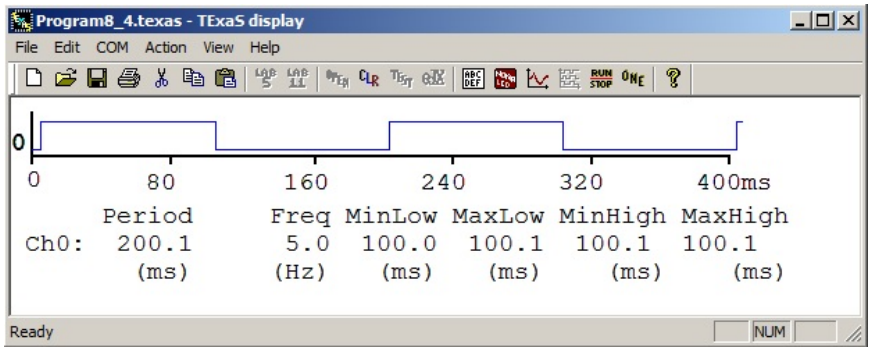

Figure 8.5. TExaS Logic Analyzer running Program 8.4 showing P5.0 toggles at 5 Hz.

```
Hint: Use this program to test the LED flashing
int Program8_4(void){
  Clock_Init48MHz();   // makes bus clock 48 MHz
  TExaS_Init(LOGICANALYZER_P5);
  LED_Init();          // activate output for LED
  while(1){
    LED_Toggle();
    Clock_Delay1ms(100);  // approximately 100 ms
  }
}
```

Lab 8: Interfacing Input and Output

8.4 Window Detector Alarm System

8.4.1 Hardware implementation:

We will use four pins on the MSP432 to implement the alarm system. It is recommended that you not use pins that already have hardware connected to them. Choose the four pins you wish to use and draw a circuit diagram of your hardware similar to Figure 8.6.

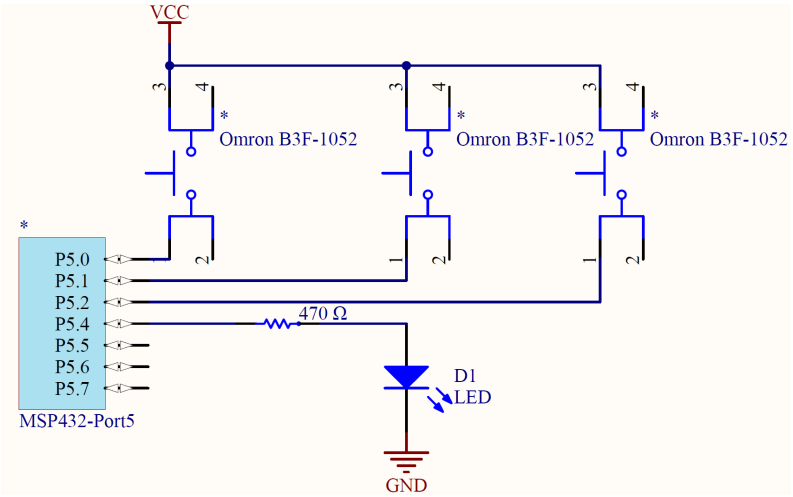

Figure 8.6. Hardware schematic interfacing input and output pins.

8.4.2 Software Implementation

The basic approach to this system is described in this pseudocode, and drawn as a flowchart in Figure 8.7.

1. Make the LED pin an output and make the three switch pins inputs.
2. The system starts with the LED off.
3. Wait about 100 ms.
4. Look at the three switches; if **Activate** is pressed and one or both **Window1** and **Window2** are not pressed, then toggle the LED once else turn the LED off.
5. Steps 3 – 5 are repeated over and over.

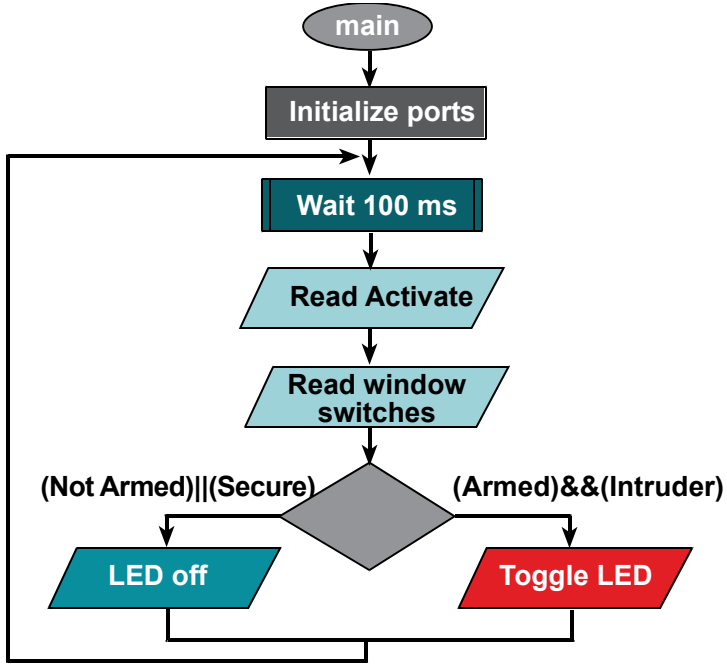

Figure 8.7. Possible software algorithm for the system drawn as a flowchart.

The structure of this lab had you build and test each subsystem separately. In this section, we combine the switch and LED interfaces to implement the Window intruder detector alarm system. To test the overall system you should first single step your software to verify it operates as intended for each of the eight cases listed in Table 8.1. Use the debugger to observe the three inputs and one output simultaneously. First, use the step over debugger command to verify the proper functional behavior

Then, you can start the program and test the system running full speed. You could use a scope or logic analyzer to verify the LED flashes at 5 Hz, when Activate is pressed and one or both Window1 and Window2 are not pressed.

Lab 8: Interfacing Input and Output

8.5 Troubleshooting

Can't program LaunchPad:

- Check the cables, jumpers on the LaunchPad development board.
- Check the Windows driver to see if the board is recognized by the operating system.
- Try another LaunchPad on this computer. Try this LaunchPad on another computer

LED does not turn on:

- Check the polarity of the LED.
- Repeat measurements done in section 8.3.2. The resistor in series with the LED should be somewhere between 220 and 2000 ohms.
- If there is 2 to 3V across the LED and the LED is dark, then it is broken (open circuit) or backwards.

Switches don't work:

- Many switches have 4 pins, and you may be confusing across which of the pins the switch is connected.
- Use an ohmmeter measure the resistance across the switch for the pressed and not pressed conditions.
- The debugger allows you to visualize the port registers; so it is a good idea to use the debugger to check if your initialization properly configured the direction and pulldown mode.

8.6 Things to think about

These questions are meant to test your understanding of the concepts in this lab.

- How would you make the LED brighter?
- What would happen if you plugged the LED in backward?
- What would happen if you reversed the LED and resistor? i.e., connect microcontroller output to the + side of the LED, and connect the - side of the LED to one end of the resistor, connect the other end of the resistor to ground. Would it still work? Why?
- The switch will bounce on/off/on for about 1 – 2 ms each time you push it. Similarly, the switch will bounce off/on/off when released. This lab actually debounces the switch. What operation in the main program causes the software to ignore the bouncing of the switch that occurs when touched and released? Debouncing means the software responds to the switch touch event only once, and not multiple times as the switch bounces.

Lab 8: Interfacing Input and Output

8.7 Additional challenges

In this section, we list additional activities you could do to further explore the concepts of this module. You could extend the security system or propose something completely different. For example,

- Add a second LED to indicate if the systems is activated
- Add a green LED to indicate all is well
- Implement this lab as a finite state machine
- Add more switches and implement a digital door lock (user hits the keys in a certain order, and the lock is simulated by the LED)
- Implement a demand pacemaker. The user pushes a switch to simulate atrial sensor, and the ventricular pacing is simulated by an LED.
- Modify the following Challenge function so the system removes switch bouncing and properly counts the number of times the switch is pressed.

```
int Challenge(void){ uint32_t Count=0;
  Clock_Init48MHz();  // makes bus clock 48 MHz
  Switch_Init();      // activate input from switch
  while(1){
    while(Switch_Input()==0){}; // wait for touch
    Count++;  // number of times switch is touched
    while(Switch_Input()!=0){}; // wait for release
  }
}
```

8.8 Which modules are next?

In this section, we list future modules that build on the concepts learned in this module.

Module 9) Use SysTick to implement time delay and dim an LED
Module 10) Add bump sensors to robot using switches
Module 13) Use periodic interrupts to run tasks in the background
Module 14) Use interrupt triggered to recognize a switch has been pressed

8.9 Things you should have learned

In this section, we review the important concepts you should have learned in this module are how to:

- Use an ohmmeter, voltmeter and logic analyzer
- Draw a circuit using a program like CircuitMaker
- Build circuits using a breadboard
- Program the direction register
- Perform input/output using a digital port, writing the functions as a software driver
- Interface a positive logic switch with pulldown
- Interface a positive logic low-current LED
- Create software delays using for-loops
- Toggle an LED using software delays

ti.com/rslk

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265