Module 8

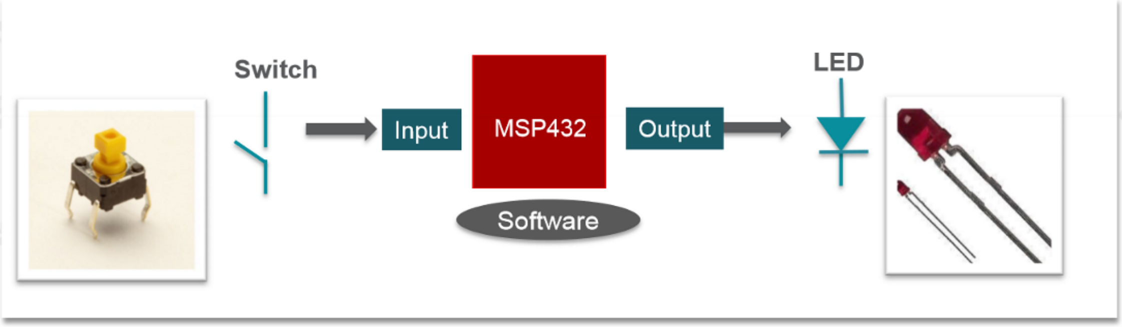
Lecture : Interfacing input and output - Switches

Interfacing input devices using Switches

Relevance: Bumper Switches
- Translate the robot hitting an object into software and handle that issue

You will learn in this module

- Fundamentals of switches
- How to interface switches TI's Launchpad Development board
- Software driver (set of functions to create an abstract module)
- Motivation for lab



Switch

Input

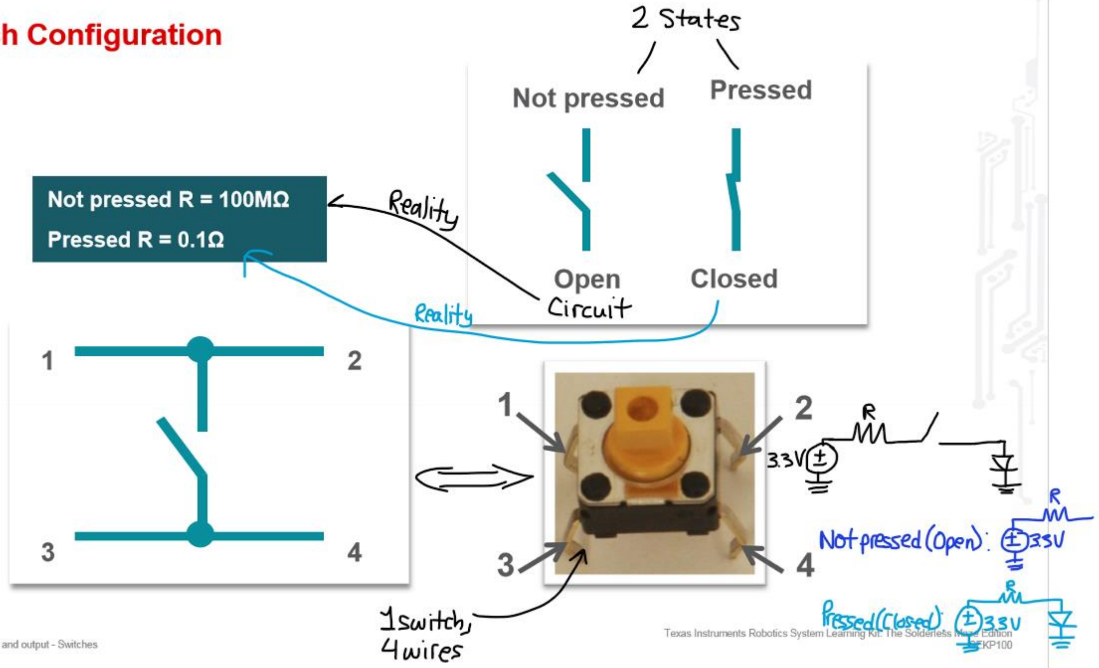
MSP432

Output

LED

Software

Switch Configuration



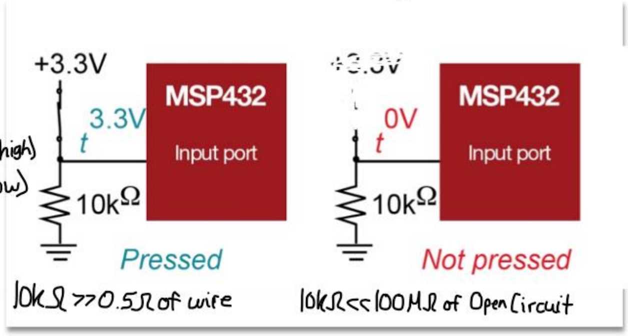
Not pressed R = 100MΩ

Pressed R = 0.1Ω

Positive Logic Switch Interface(2 ways to use it)

Pull-Down Resistor ⟺ Positive Logic

Pressed → 3.3V (Logic high)
Not Pressed → 0V (Logic Low)



10kΩ >> 0.5Ω of wire

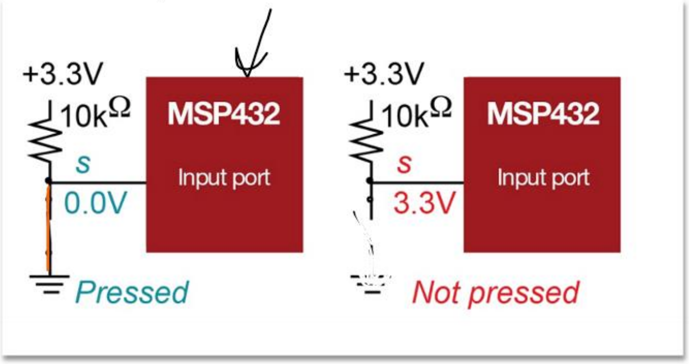
10kΩ << 100MΩ of Open Circuit

Positive Logic *t*
- pressed, 3.3V, true
- not pressed, 0V, false

Negative Logic Switch Interface

Pull-Up Resistor ⟺ Negative Logic

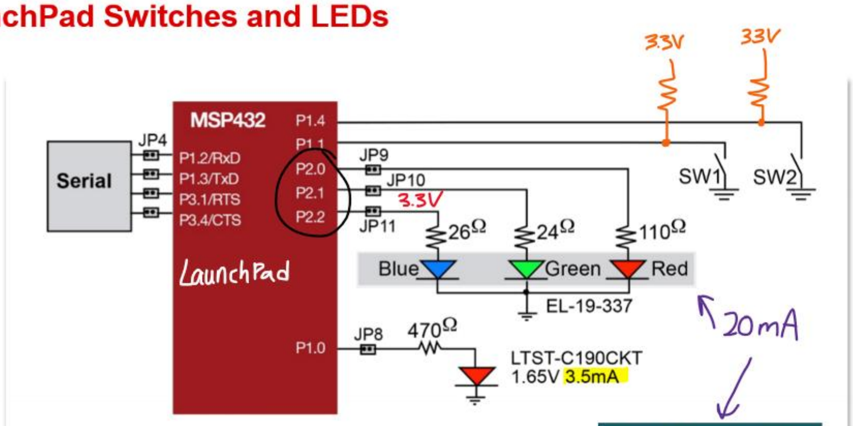
Pressed → 0V
Not Pressed → 3.3V



Negative Logic *s*
- pressed, 0V, true
- not pressed, 3.3V, false

No advantages between
(+) vs (-) logic

LaunchPad Switches and LEDs



```
P2->DS = 0x07;
```

The Switches on the LaunchPad

- Negative logic
- Require internal pull-up

The LEDs are positive logic

Software Driver (inputs)

↳ functions that let you use a device (here the switches)

All here works for Positive and Negative Logic

Initialization (executed once at beginning)

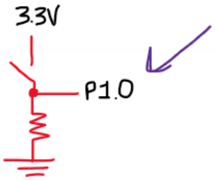
1. Set *DIR* to 0 for input
2. Enable pullup on inputs

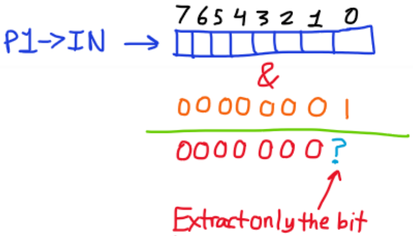
Input from switches

1. Read from data input port
2. Mask (select) desired bits

Mask

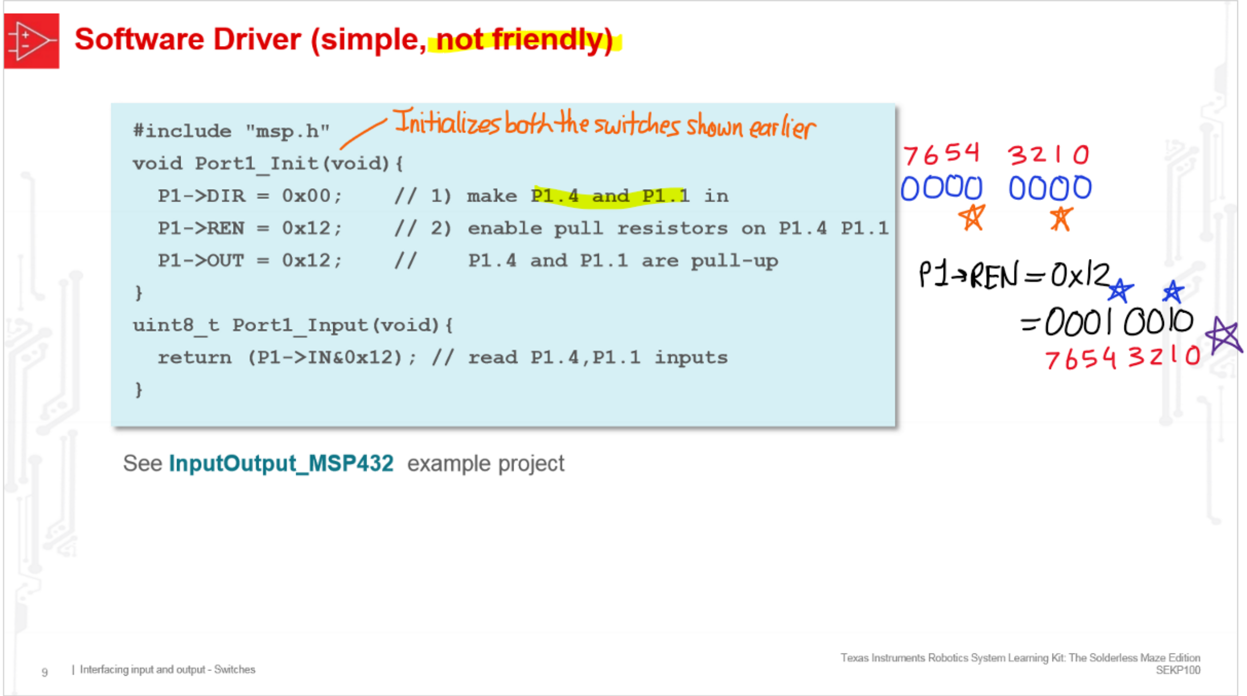
```
all = P1->IN;
in = all&0x01;
```





Extract only the bit that we want

For Positive Logic
?=1 if pin is pressed
?=0 if pin is not pressed



Software Driver (simple, not friendly)

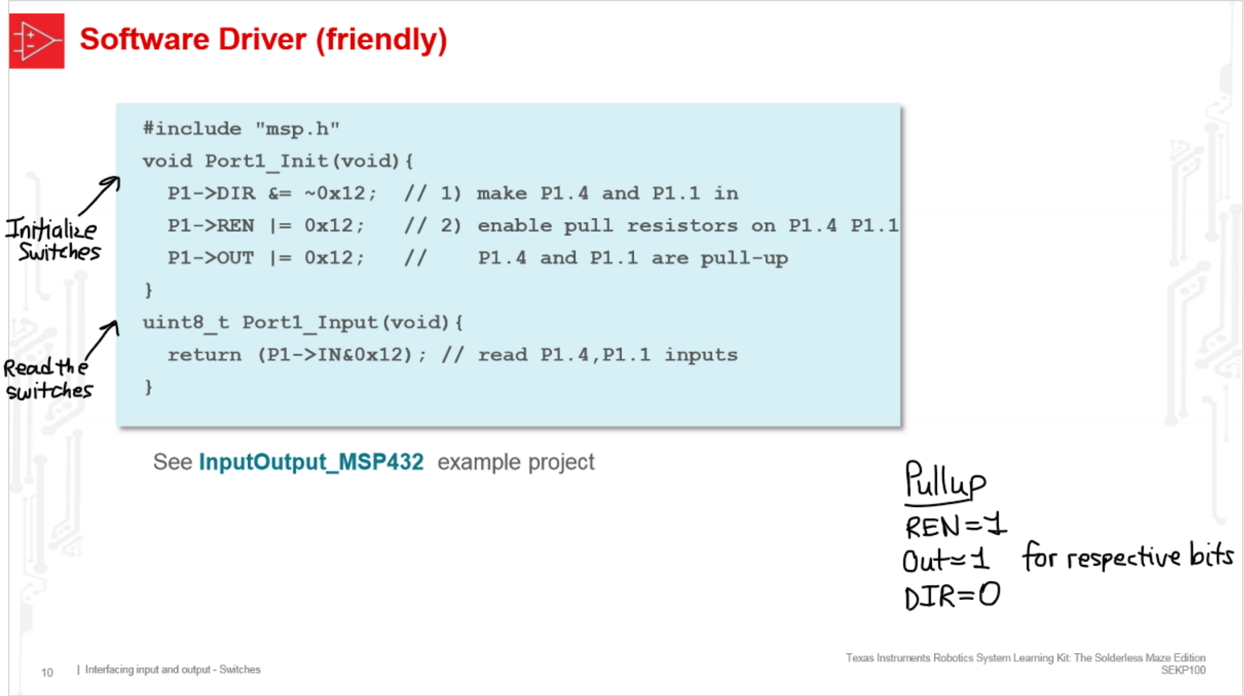
```
#include "msp.h"
void Port1_Init(void){
  P1->DIR = 0x00;    // 1) make P1.4 and P1.1 in
  P1->REN = 0x12;    // 2) enable pull resistors on P1.4 P1.1
  P1->OUT = 0x12;    //    P1.4 and P1.1 are pull-up
}
uint8_t Port1_Input(void){
  return (P1->IN&0x12); // read P1.4,P1.1 inputs
}
```

Initializes both the switches shown earlier

7654 3210
0000 0000

P1→REN = 0x12
= 0001 0010
7654 3210

See **InputOutput_MSP432** example project



Software Driver (friendly)

```
#include "msp.h"
void Port1_Init(void){
  P1->DIR &= ~0x12;  // 1) make P1.4 and P1.1 in
  P1->REN |= 0x12;   // 2) enable pull resistors on P1.4 P1.1
  P1->OUT |= 0x12;   //    P1.4 and P1.1 are pull-up
}
uint8_t Port1_Input(void){
  return (P1->IN&0x12); // read P1.4,P1.1 inputs
}
```

Initialize Switches

Read the switches

See **InputOutput_MSP432** example project

Pullup
REN=1
Out=1 for respective bits
DIR=0

Application

Switches provide

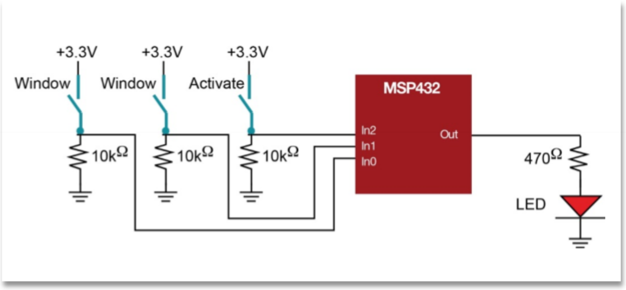
1. Feedback to robot as bump sensors to determine if there is an obstruction
2. Control/command inputs to robot (e.g., start/stop)

Voltage = Current x Resistance

Summary

- Positive and negative logic
- Ohm's Law for resistors
- Switch interface with pullup or pulldown
- LaunchPad switches and LEDs
- Software driver
  - Initialization
  - Input/Output functions

V = I * R


+3.3V
+3.3V
+3.3V
Window
Window
Activate
10kΩ
10kΩ
10kΩ
MSP432
In2
In1
In0
Out
470Ω
LED