University of Texas at El Paso

Electrical and Computer Engineering Department

EE 3176 – Laboratory for Microprocessors I

Spring 2022

LAB 02

GPIO Port Parallel I/O

Goals:

- Write a C program to produce binary outputs and read input binary signals through general-purpose input/output (GPIO) ports.
- Write GPIO port functions to produce outputs and read inputs by polling.

Pre Lab Questions:

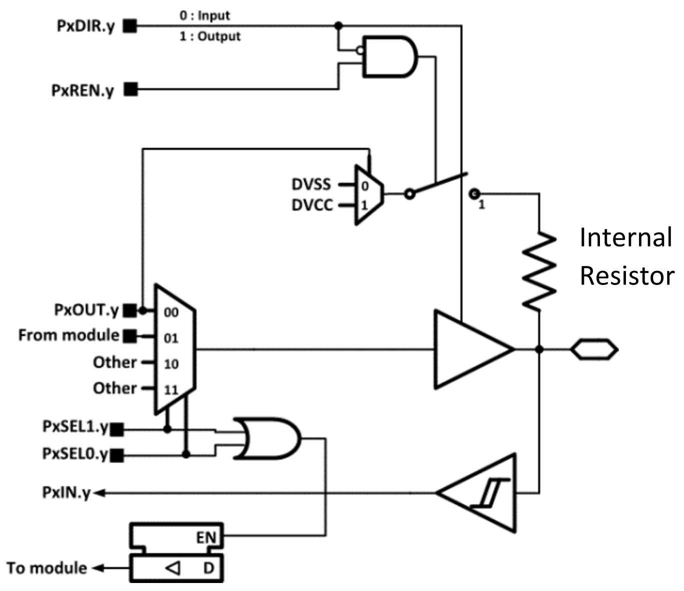
- How many GPIO ports are available in the MSP432 microcontroller you are using?
- How many bits per port does it have?
- How many bits per port are accessible through Launchpad pins?
- How many input and/or output variables can you read and/or write through an MSP432 port that has all bits accessible at Launchpad pins?
- What is the purpose of internal resistors associated with GPIO port bits and when do you configure them for pull-up or pull-down? What are the advantages and disadvantages of using internal resistors in an embedded design.

Port I/0

Lab Guide

You will work with MSP432 general-purpose input/output (GPIO) ports in this lab. Each port is mapped in the address space of the processor, accessible as a memory location, and referred to as a register. Port functions are configured and accessed through a set of port-dedicated registers. By default, GPIO port bits are configured as digital inputs. Since ports are byte-addressable, specific port bits must be handled by numerical masks. For instance, Port X bits, where X is some character used to name a specific port, are numbered 0 through 7 and the least significant bit is bit zero. Thus, Port X bits are PX.0, PX.1, PX.2, and so on. Each GPIO port has a set of dedicated registers as shown for Port 1 in the table below.

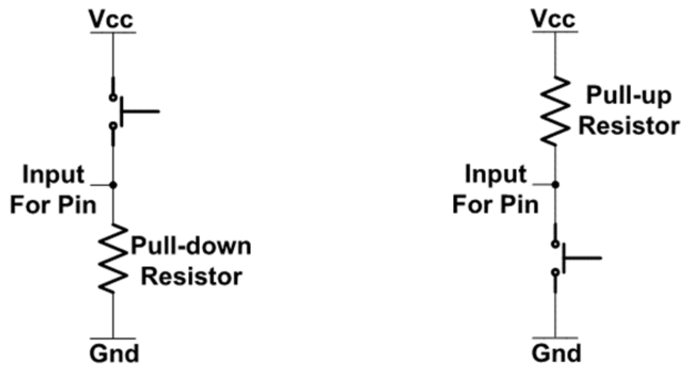
Register use in C	Function	Bit values
P1->DIR	To set input/output pin direction	0 = Input (default), 1 = Output
P1->IN	To read input data from port pins	0 = Low voltage, 1 = High voltage
P1->OUT	To write output data to port pins	0 = Low voltage, 1 = High voltage
P1->REN	Internal resistor control	0 = Disconnect, 1 = Connect. If connected, P1->OUT bits set resistor as pull-up (=1) or pull-down (=0)
P1->SEL0	Pin peripheral selection 0	Connect port bits to peripherals. Default: SEL0 = SEL1 = 0 for all bits.
P1->SEL1	Pin peripheral selection 1	



GPIO port pin hardware

Resistor Configurations

When connected using PX->REN, internal resistors can be configured as either Pulldown or Pullup resistors as shown below.



Active high input Active low input

Pin setups to connect an external button.

Reading from and Writing to Port Registers

C	EXAMPLE	NOTES
|=	`P1->OUT |= 0b00001111;      // 0x0F`	Sets the least significant nibble.
&= ~(…)	`P1->OUT &= ~(0b11110000); // 0xF0`	Clears the most significant nibble.
=	`P1->OUT = 0b01000001;      // 0x41`	Overwrite P1 output. BIT6=BIT0=1
^=	`P1->OUT ^= 0b01000001;     // 0x41`	Toggle bits 0 and 6 in P1 output.
–	`if( !(P1->IN – BIT3) ){ … }`	Is P1 input == BIT3, 0b00001000
&	`if( (P1->IN & BIT3) == BIT3){ … }`	Is BIT3 **set** in P1 input.

You can also read a **Port Register** using a C assignment. For instance, int val = P1->IN; will read the value of signals connected to Port 1 pins. Notice that C code above indicates that each port is a structure with a member assigned to each port register (OUT, IN, DIR, etc.) and you use a pointer to access a port (P1, P2, P3, etc.).

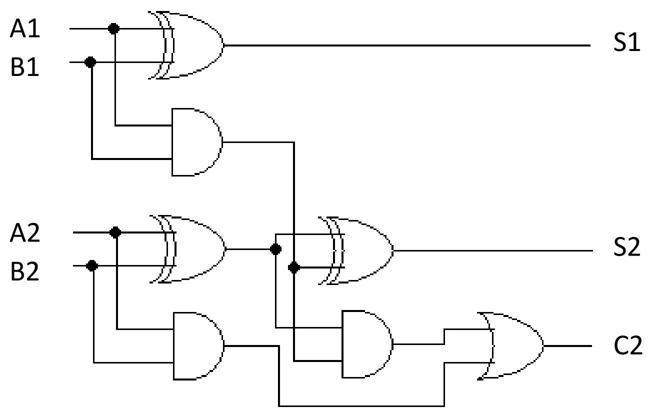
Notes:

- The microcontroller takes any input voltage below 0.8V as a Logical 0 and any input voltage above 2.4V as a Logical 1. Inputs between 0.8V and 2.4V will produce unpredictable results when used as digital inputs.

- Make sure you configure internal resistors or connect external ones according to inputs to be produced by the sensors and devices in your circuits. For instance, a button or a switch may be used to input either a 1 or a 0 when closed.
- The internal resistor can ground or set to 1 a pin to avoid having floating inputs.
- The pullup resistor configuration is best for LaunchPad onboard buttons.

System Design

Use the MSP432 Launchpad to design and implement a two-bit adder as shown below.



INPUTS				OUTPUTS					
				WHEN CO = L			WHEN CO = H		
A1	B1	A2	B2	S1	S2	C2	S1	S2	C2
0	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	1	0
0	1	0	0	1	0	0	0	1	0
1	1	0	0	0	1	0	1	1	0
0	0	1	0	0	1	0	1	1	0
1	0	1	0	1	1	0	0	0	1
0	1	1	0	1	1	0	0	0	1
1	1	1	0	0	0	1	1	0	1
0	0	0	1	0	1	0	1	1	0
1	0	0	1	1	1	0	0	0	1
0	1	0	1	1	1	0	0	0	1
1	1	0	1	0	0	1	1	0	1
0	0	1	1	0	0	1	1	0	1
1	0	1	1	1	0	1	0	1	1
0	1	1	1	1	0	1	0	1	1
1	1	1	1	0	1	1	1	1	1

Your design must produce the output values in the truth table for each set of inputs in either manual mode or automated mode. In manual mode, a set of user inputs must produce the outputs given in the truth table. In the automated mode, all the valid input combinations must be internally produced to drive the outputs. Your design must show both the input and the output values using LEDs with a short visualization delay when appropriate. When all truth table entries have been displayed, the system must turn on an automated pass done sound.

Show your system in operation to your lab T.A. and answer any questions the T.A. may have for you.

Port I/0

Frequently Asked Questions

What size resistors should I use with my LEDs?

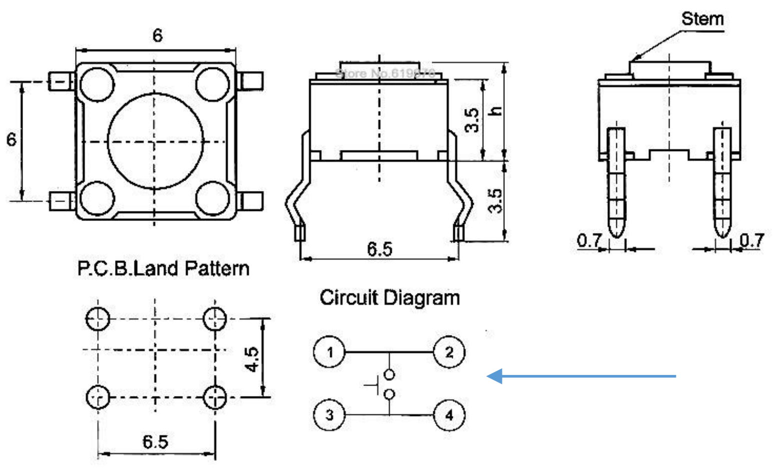
It depends on the LED, but most LEDs will turn on with a 10mA current. Thus, your resistor must be calculated for the turn-on current given your highest logical voltage value. Connecting an LED directly to a port output, i.e., without a current-limiting resistor, will burn the port pin output circuit.

Can I modify P1->IN to simulate a button being pushed?

No. **P1->IN** is read-only. Attempting to write to this register will have no effect. You will have to physically change the state of the pins. Alternatively, you can temporally change the condition of the desired code (whatever code you are expecting the button/pin to affect) to always be true or false in the meantime.

How do I connect my switch!?

Checkout the diagram below if you are having trouble connecting your switch:



6
6
3.5
h
3.5
6.5
Stem
0.7
0.7
P.C.B.Land Pattern
4.5
6.5
Circuit Diagram
1
2
3
4