ESET 269 – Embedded Systems Development in C

Digital I/O (Input/Output)

Dr. Garth V. Crosby

Registers of Digital I/O

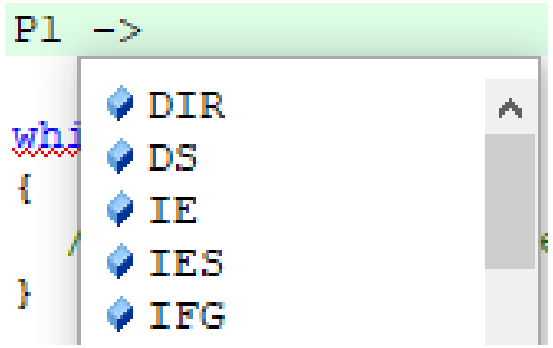
- Registers control the function and behavior of the port pins.
- Need to set these registers through program
  - Struct is already defined and declared through startup.c file
- Registers
  - Port select (specifies if pin is digital I/O or other peripheral)
  - Port direction (is pin an input or output)
  - Port out (digital output value of port)
  - Port in (digital input value of port)
  - Port pull up/pull down resistor (enable a pull up or pull down resistor)

Register Variables in Keil

- Ports are set up as struct pointers P1, P2, P3, etc. The members of the struct are the registers
  - Port select (SEL0 & SEL1)
  - Port direction (DIR)
  - Port out (OUT)
  - Port in (IN)
  - Port pull up/pull down resistor (DREN)

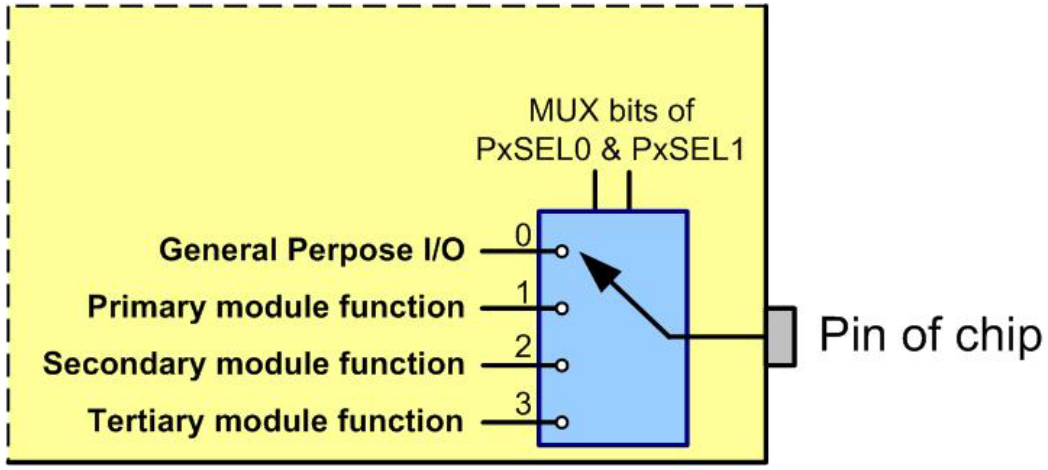
```
P1 ->DIR
```

- The value of the member is an 8-bit number
  - 1 bit for each pin on the port



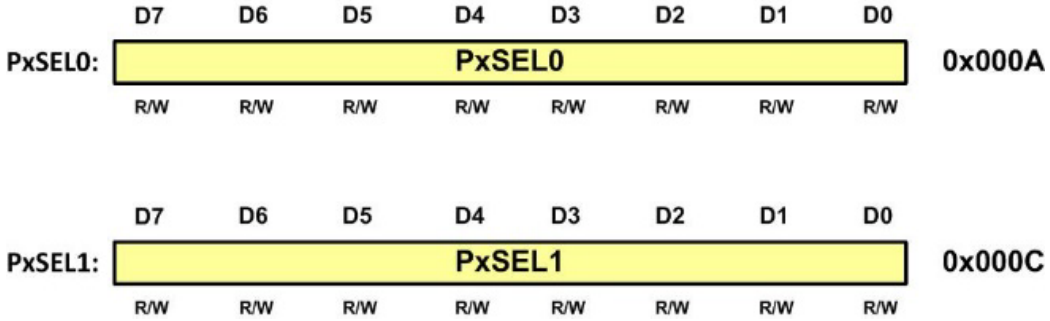
Enabling Digital I/O

- To use a pin as digital, its select registers (SEL0 and SEL1) are set to 0



PxSEL1	PxSEL0	Meaning
0	0	Alternative 0 (Default Simple I/O)
0	1	Alternative 1 (UART, SPI), 12C, …)
1	0	Alternative 2 (Timers, …)
1	1	Alternative 3 (ADC, Comparator, …)

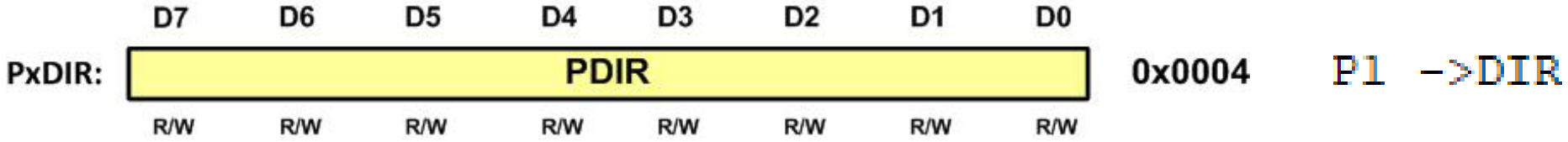
Px denotes a Port
D0 – D7 denote a Pin




```
P1 ->SEL0
P1 ->SEL1
```

Setting Pin Input or Output

- After setting the SEL0 and SEL1 modes, the direction of the pins have to be specified with DIR register
  - 1 sets the pin to Output
  - 0 sets the pin to Input



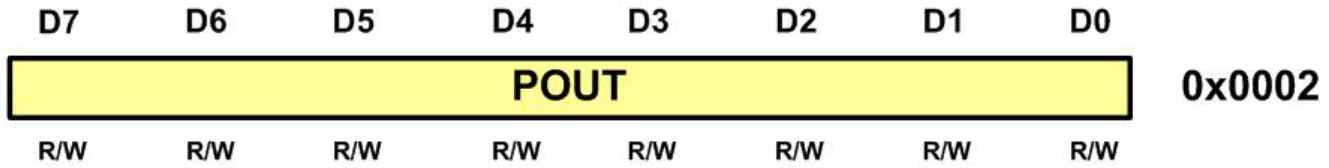
- How would pins 5, 3, 2, and 0 be set as digital outputs of port 1?

Need to write a value to P1DIR:
Set the pin to 1 for output.
Setting bits 5,3,2, and 0: 101101
101101→2D
Write 0x2D or Write 0b101101

Setting an Output Value

- Once a pin is specified as an output, it can generate a HIGH or LOW via the OUT register
  - 1 – generate a HIGH
  - 0 – generate a LOW

P1 ->OUT



- Assuming pin 6 on port 1 is a digital output. How would a digital HIGH be generated?

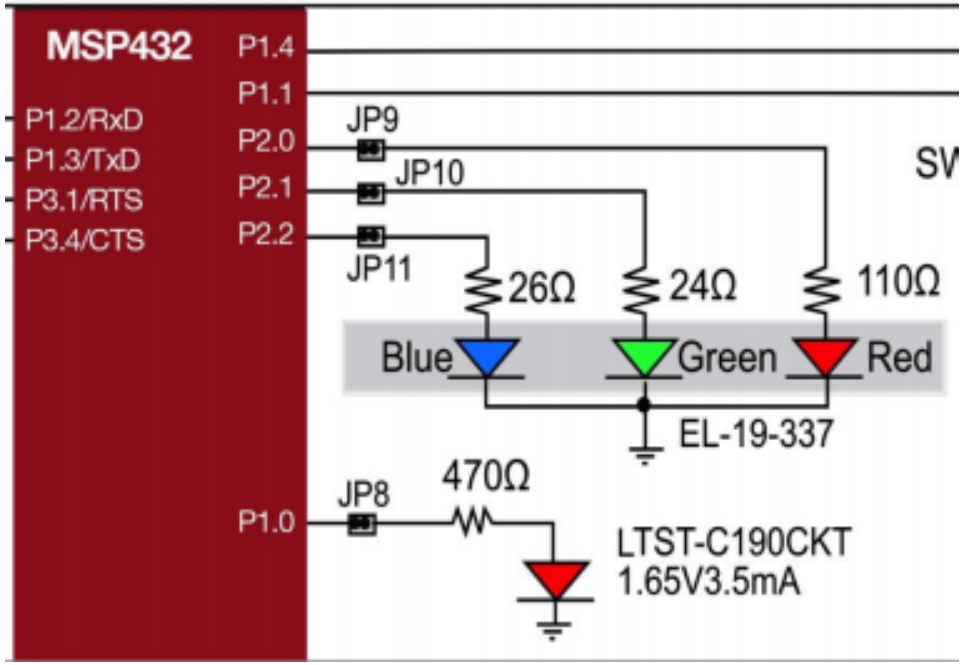
Answer: Set P1.5

Summary of Digital Output

- Set the SEL0 and SEL1 registers for the pins you want to use a digital I/O
- Set the DIR register to 1 on each port for the pins you want to be outputs
- Set the OUT register to 1 or 0 to set the digital outputs HIGH or LOW

Exercise

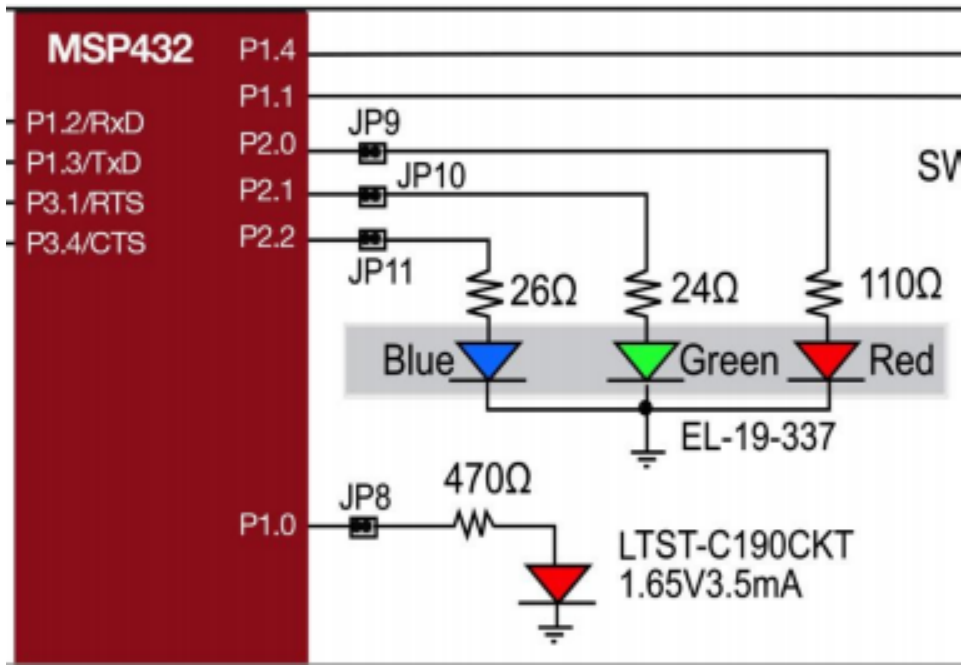
- Write code to turn on the green LED on port 2 pin 1.



To toggle the green LED of the LaunchPad board, the following steps must be followed.

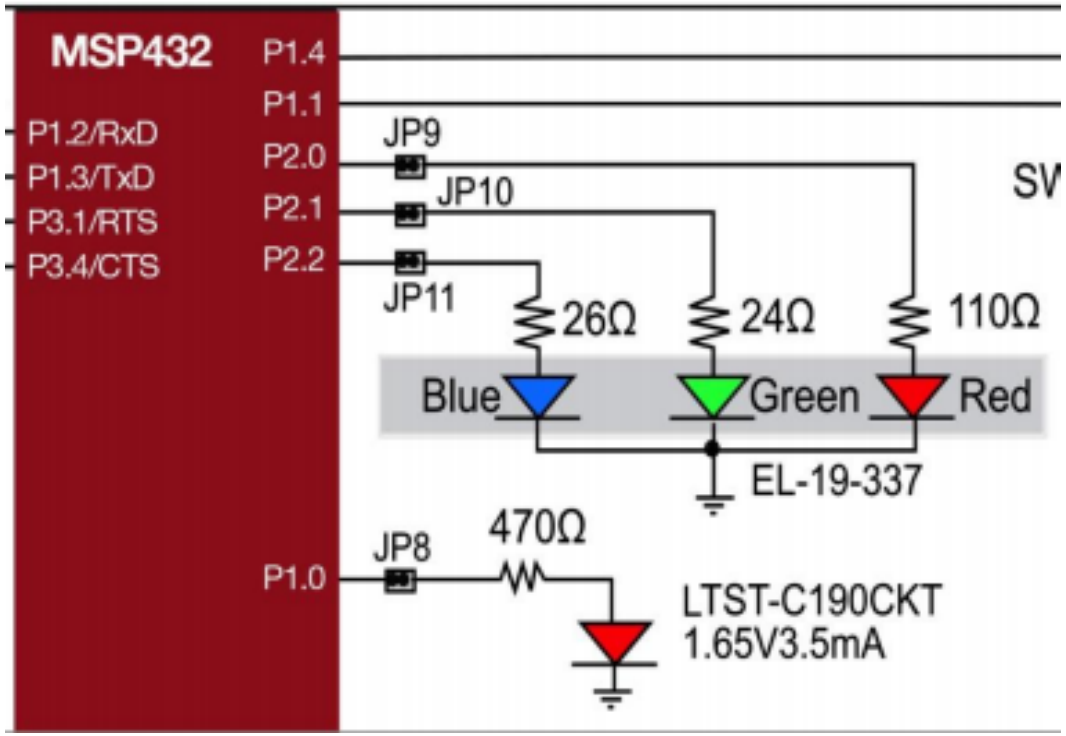
1) Configure P2.1 (P2SEL1:P2SEL0 Register) to select simple GPIO function for P2.1,
2) set the Direction register bit 1 of P2DIR as output,
3) write HIGH to bit 1 of P2OUT register to turn on the green LED,
4) call a delay function,
5) write LOW to bit 1 of P2OUT register to turn off the green LED,
6) call a delay function,
7) Repeat steps 3 to 7.

```
#include "msp.h"

void delayMs(int n);

int main(void) {
    P2->SEL1 &= ~2;          /* configure P2.1 as simple I/O */
    P2->SEL0 &= ~2;
    P2->DIR |= 2;            /* P2.1 set as output pin */

    while (1) {
        P2->OUT |= 2;        /* turn on P2.1 green LED */
        delayMs(500);
        P2->OUT &= ~2;       /* turn off P2.1 green LED */
        delayMs(500);
    }
}

/* delay milliseconds when system clock is at 3 MHz */
void delayMs(int n) {
    int i, j;

    for (j = 0; j < n; j++)
        for (i = 250; i > 0; i--);      /* Delay 1 ms */
}
```

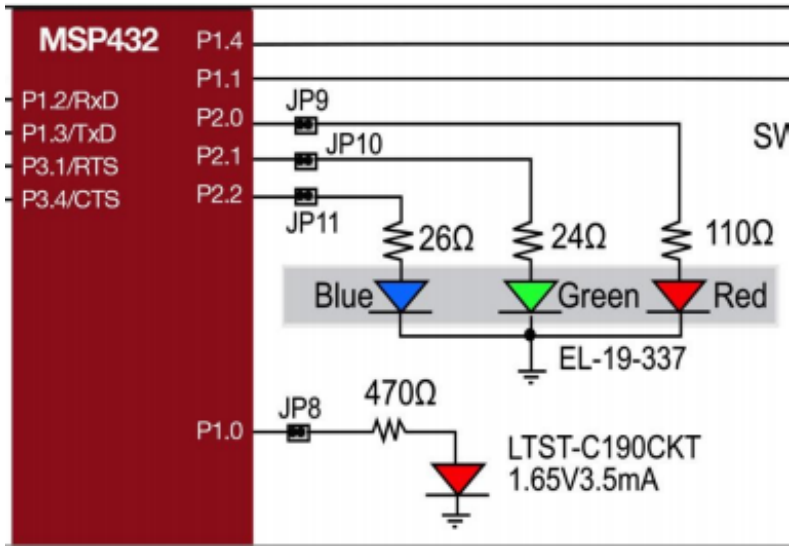


Exercise

- Write code to turn on the RED, GREEN and BLUE which are connected to on port 2.



```c
#include "msp.h"

void delayMs(int n);

int main(void) {
    P2->SEL1 &= ~7;          /* configure P2.2-P2.0 as simple I/O */
    P2->SEL0 &= ~7;

    P2->DIR |= 7;            /* P2.2-2.0 set as output */
    P2->OUT |= 7;            /* turn all three LEDs on */

    while (1) {
        P2->OUT ^= 7;        /* toggle P2.2-P2.0 all three LEDs */
        delayMs(500);
    }
}

/* delay milliseconds when system clock is at 3 MHz */
void delayMs(int n) {
    int i, j;

    for (j = 0; j < n; j++)
        for (i = 250; i > 0; i--);      /* Delay */
}
```



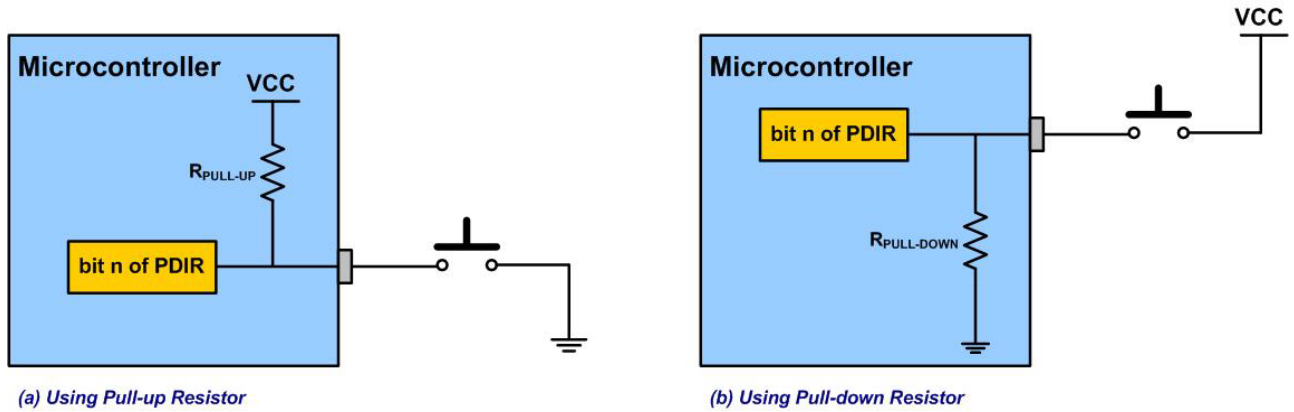
Digital Input

- Read a digital voltage on a pin
  - HIGH or LOW (3.3 V or GND)
- Pin is configured for digital input by setting the DIR register for the pin to 0
  - Pull up and pull down resistors specified with REN and OUT registers
- The value of the digital pin is taken from the input (IN) register for a given pin
  - Use bitwise operations to read specific pin or pins

Setting Pin For Digital Input

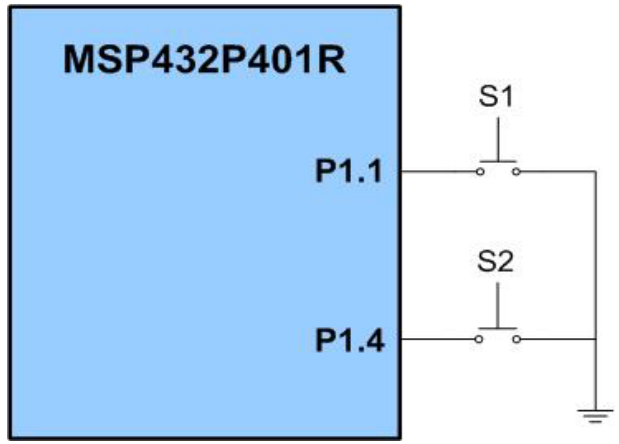
- DIR register is set to 0 for a pin to be digital input
- The resistor enable register (REN) is used to enable the internal resistor of a given input pin
  - 1 - use resistor, 0 is do not use resistor
- The internal resistor of an input pin must be set as pull up or pull down through the OUT register
  - 1 – Pull up, 0 – Pull down

```
P1->DIR &=~0x02; //set pin 1 on port 1 to input
P1 ->REN |=0x02; //enable internal resistor for pin 1
P1->OUT |=0x02; //set internal resistor to pull up
```



Example Setup

- Assume SW1 is located on pin 1.1 and SW2 is located on pin 1.4.
- Buttons on Launchpad are set up for active LOW
  - Ties input pin to GND when pressed
- Find the contents of the P1DIR, P1REN and P1OUT registers so that these pins will function as input pins to sense the switch positions.



Solution

- We need to clear bit 4 and bit 1 in P1DIR to make P1.1 and P1.4 pins as input. And to activate the internal pull resistor, the P1REN register needs to be 0b00010010 and the P1OUT needs to be 0b00010010 to enable the pull-up option.

```
P1->DIR &=~0x12; //set pin 1 and 4 to input
P1 ->REN |=0x12; //enable internal resistor for pin 1 and pin 4
P1->OUT |=0x12; //set internal resistor to pull up for pin 1 and 4
```

Reading a Pin Value

- The input register (IN) contains the input value on a pin
- Need to apply a mask to extract the bit or bits of interest

```
int x;
x = P1->IN & 0x01
if(x == 1)
{
  //do something
}
```

```
if((P1->IN & 0x01) !=0)
{
  //do something
}
```

- How would the above code change to read the value of pin 4 on port 1?

Using Output & Input on Same Port

- Pins can be mixed and matched for input and output on same port
- Use bitwise operations to set and read registers appropriately

```
P1 ->DIR |= 0x01; //set pin 0 to output
P1 ->DIR &=~0x02; //set pin 1 to input
P1 ->REN |= 0x02; //enable resistor for pin 1
P1 ->OUT |= 0x02; //use pull up on pin 1
P1 ->OUT |= 0x01; //turn pin 0 HIGH
P1 ->OUT &= ~0x01; //turn pin 0 LOW
```

* Pin 0 and 1 share the same OUT and DIR register. Be careful the code doesn't accidently modify a value it shouldn't

Example

```
int main()
{
  //set pin 7 and 4 to digital
  P3->SEL0 &=~0x90;
  P3->SEL1 &=~0x90;

  P3->DIR |=0x80; //set pin 7 output
  P3->DIR &=~0x10; //set pin 4 to input
  P3->REN |=0x10; //use internal resistor pin 4
  P3->OUT &=~0x10; //use pull down configuration

  while(1)
  {
    if((P3->IN & 0x10)==0x10) //if pin 4 is HIGH
    {
      P3->OUT |=0x80; //set pin 7 HIGH
    }
    else
    {
      P3->OUT &=~0x80; //set pin 7 LOW
    }
  }
}
```