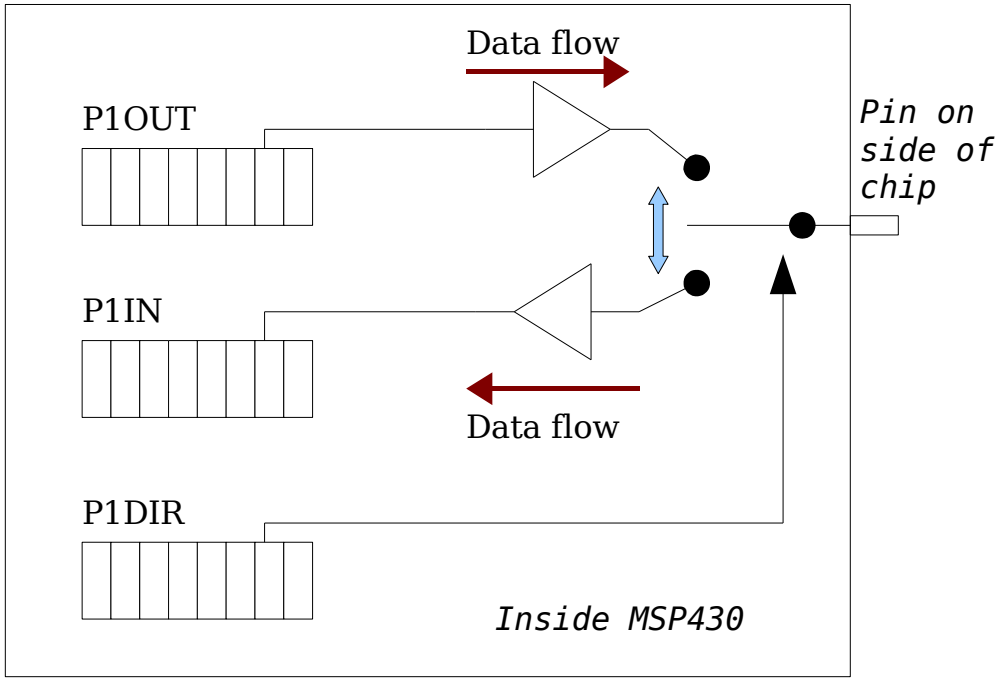
Simple Input/Output using the MSP430

(note: these notes are written with the MSP430F149 in mind)

The MSP430 uses memory mapped I/O. From a programmer's perspective, the effect of this is that writing to particular memory locations changes the logic state of pins (outputs) on the side of the chip. Reads from particular memory locations return the logic states of particular pins (inputs).

Up to 6 ports are supported : P1 to P6 depending on the model of the MSP430 being used. All ports “map to” 8 bit memory locations. Each port is associated with (at least) three memory locations: PnIN, PnOUT and PnDIR(ection) where 'n' represents the port number. Consider Port1 for a moment.
P1IN is mapped to memory address 20 (hex)
P1OUT is mapped to memory address 21 (hex)
P1DIR is mapped to memory address 22 (hex).

The Data direction register determines whether a particular bit in a port is an input or an output. Figure 1 shows how bit 1 of port 1 may be configured as an input or an output by setting or clearing bit 1 of the Direction register. If a DIRection register bit is a “1” then the corresponding port data bit is an output. If a DIRection register bit is a “0” then the corresponding port data bit is an input.



Now lets say we want to use Port 1 to monitor the status of a push button and to control a Light Emitting Diode when the button is pressed. The circuit could be represents as shown in Figure 2 below:

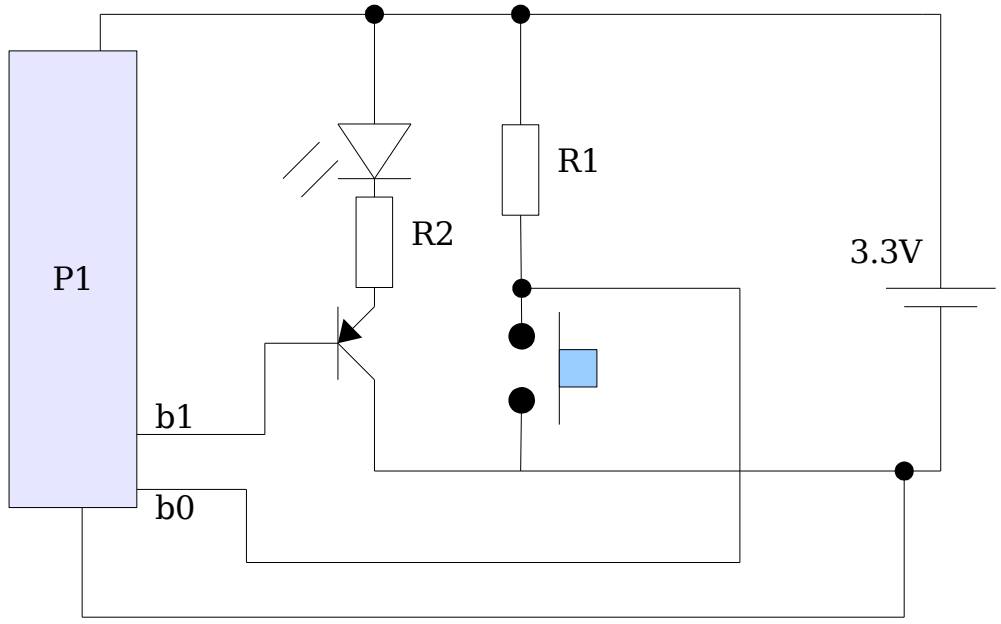


Figure 2

To turn on the LED, bit1 of PORT1 should be set to 0. To turn off the LED, bit1 of PORT1 should be set to 1.

The program continually monitors P1IN checking to see whether its least significant bit is a 0 or 1. This bit will be zero if the button is pressed, and if so, the LED is turned on.

We could program this in C as follows:

```
typedef unsigned char byte;
#define P1IN  *((byte *)0x20)
#define P1OUT *((byte *)0x21)
#define P1DIR *((byte *)0x22)
#define P4IN  *((byte *) 0x1c )
#define P4OUT *((byte *) 0x1d )
#define P4DIR *((byte *) 0x1e )
// Watchdog timer definitions (see later)
#define WDTCL *((short *)0x120)
#define WDTHOLD 0x0080
#define WDTPW 0x5a00
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD; // Stop watchdog timer (see later)
  P1DIR |= 0x02;            // Set P2.1 is an output
  while( 1==1 )
  {
    if( (P1IN & 0x01) != 0) {
      // SWITCH IS OPEN SO TURN OFF LED
      P1OUT = P1OUT | 2;
    } else {
      // SWITCH IS OPEN SO TURN ON LED
      P1OUT = P1OUT & 0xfd;
    }
  }
}
```

Ports 1 to 6 (where available) are memory mapped as follows (note the “0x” in front implies HEX):

Port	PnIN	PnOUT	PnDIR	PnSEL
P1	0x20	0x21	0x22	0x26
P2	0x28	0x29	0x2a	0x2e
P3	0x18	0x19	0x1a	0x1b
P4	0x1c	0x1d	0x1e	0x1f
P5	0x30	0x31	0x32	0x33
P6	0x34	0x35	0x36	0x37