```c
/*
--------------------------------------------------------------
2.996 / 6.971: Biomedical Devices Design Laboratory
Lab Example Software - Digital I/O

This is a quick demo of how to set up and use the digital I/O
on the MSP430F2274. It outputs an 8-bit binary counter on the
eight PORT1 pins. The counter changes on P2.0 rising edge
interrupts, with P2.1 setting the direction (increment or
decrement). This sort of setup could be used to count
quadrature encoder pulses.

SC - 9/30/2007
--------------------------------------------------------------
*/

#include "msp430x22x4.h"

// declare an 8-bit counter variable
volatile unsigned char counter = 0;
// Note: Use 'volatile' type to prevent optimization since
// this variable can be modified by an interrupt routine.
// See lecture notes.

// special syntax for interrupt service routines
#pragma vector=PORT2_VECTOR     // declared in "msp430x22x4.h"
__interrupt void P2_ISR(void)
{
  if((P2IFG & BIT0) != 0)       // if interrupt flag on P2.0
  {
    if((P2IN & BIT1) != 0)      // if P2.1 is high
    { counter++; }              //   increment counter
    else                        // if P2.1 is low
    { counter--; }              //   decrement counter
    P2IFG &= ~BIT0;             // clear P2.0 interrupt flag
  }
}
// Note: Some interrupt flags are automatically cleared, but
// most need to be reset in software.

void main(void)
{
  // stop watchdog timer
  WDTCTL = WDTPW | WDTHOLD;

  // special function to enable all interrupts
  // peripheral interrupts must still be enabled individually
  _EINT();

  // set P1.0-1.7 as outputs (all others default to inputs)
  P1DIR = 0xFF;
  // initialize all P1 outputs low
  P1OUT = 0x00;
  // Note: Most registers clear to 0x00 on a reset, but some do not.
  // P1OUT is one that doesn't, so it is necessary to clear it here.
  // See User's Guide, 8-7.

  // enable interrupt on P2.0, rising edge
  P2IE = BIT0;
  P2IES &= ~BIT0; // for falling edge, use P2IES = BIT0;
  // Note: P2IES is another register that may not clear on reset.


  while(1) // loop forever
  {
    // make P1 into a parallel output binary counter
    // could drive a row of LEDs, for example
    P1OUT = counter;
  }
}
```