```
//light sensor: p6.1-out. in -p1.1, make sure ADCSREF0
#include <msp430.h>

unsigned int light;

int main(void)
{
   WDTCTL = WDTPW | WDTHOLD;                    // Stop WDT

   // Configure GPIO
   P6DIR |= BIT1;                           // Set P1.0/LED to output direction
   P6OUT &= ~BIT1;                          // P1.0 LED off

   // Configure ADC A1 pin
   P1SEL0 |= BIT4;
   P1SEL1 |= BIT4;

   // Disable the GPIO power-on default high-impedance mode to activate
   // previously configured port settings
   PM5CTL0 &= ~LOCKLPM5;

   // Configure ADC12
   ADCCTL0 |= ADCSHT_8 | ADCON;                 // ADCON, S&H=16 ADC clks
   ADCCTL1 |= ADCSHP;                       // ADCCLK = MODOSC; sampling timer
   ADCCTL2 &= ~ADCRES;                      // clear ADCRES in ADCCTL
   ADCCTL2 |= ADCRES_2;                     // 12-bit conversion results
   //ADCMCTL0 |= ADCINCH_12;                    // A1 ADC input select; Vref=AVCC
   ADCIE |= ADCIE0;                         // Enable ADC conv complete interrupt
   ADCMCTL0 |= ADCSREF_1 | ADCINCH_4;
```

```
  while(1)
  {
    ADCCTL0 |= ADCENC | ADCSC;                  // Sampling and conversion start
    __bis_SR_register(LPM0_bits | GIE);         // LPM0, ADC_ISR will force exit
    __no_operation();                           // For debug only
    if (light < 200)
      P6OUT |= BIT1;                            // Set P1.0 LED on
    else
      P6OUT &= ~BIT1; // Clear P1.0 LED off
    __delay_cycles(5000);
  }
}

// ADC interrupt service routine
#if defined(__TI_COMPILER_VERSION__) || defined(__IAR_SYSTEMS_ICC__)
#pragma vector=ADC_VECTOR
__interrupt void ADC_ISR(void)
#elif defined(__GNUC__)
void __attribute__ ((interrupt(ADC_VECTOR))) ADC_ISR (void)
#else
#error Compiler not supported!
#endif
{
  switch(__even_in_range(ADCIV,ADCIV_ADCIFG))
  {
    case ADCIV_NONE:
      break;
    case ADCIV_ADCOVIFG:
```

```
            break;
        case ADCIV_ADCTOVIFG:
            break;
        case ADCIV_ADCHIIFG:
            break;
        case ADCIV_ADCLOIFG:
            break;
        case ADCIV_ADCINIFG:
            break;
        case ADCIV_ADCIFG:
            light = ADCMEM0;
            __bic_SR_register_on_exit(LPM0_bits);           // Clear CPUOFF bit from LPM0
            break;
        default:
            break;
    }
}
```