Introduction to Embedded System Design

Seven Segment Displays with MSP430, Low Power Modes in MSP430

Dhananjay V. Gadre
Associate Professor
ECE Division
Netaji Subhas University of Technology, New Delhi

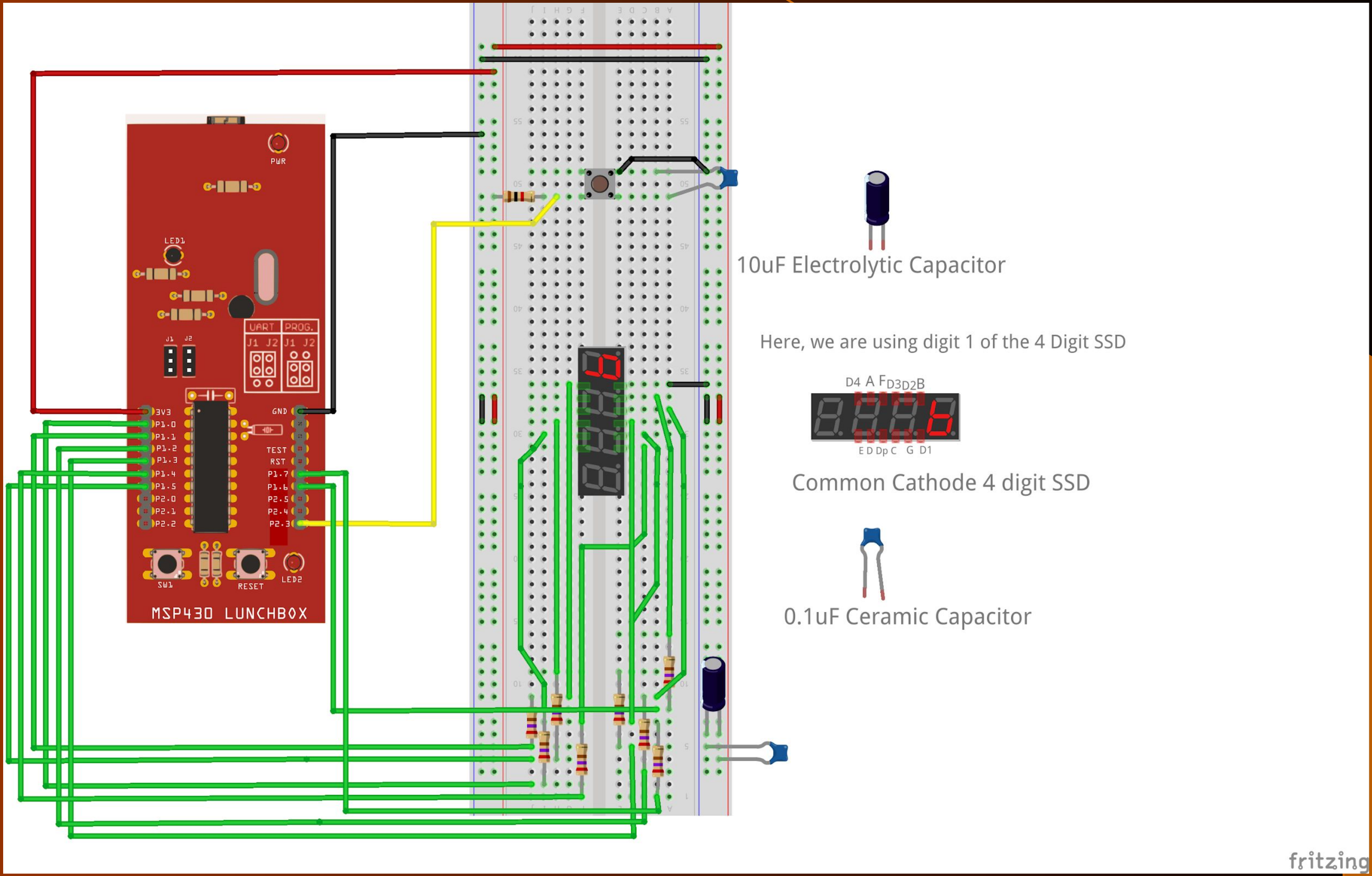
Badri Subudhi
Assistant Professor
Electrical Engineering Department
Indian Institute of Technology, Jammu

Interfacing SSD with MSP430

HelloSSD:
This example code is used to display a hexadecimal (0 to F) up counter on a single digit SSD(Common Cathode). The count increases on every press of an external switch.

Interfacing SSD with MSP430

Fritzing Diagram



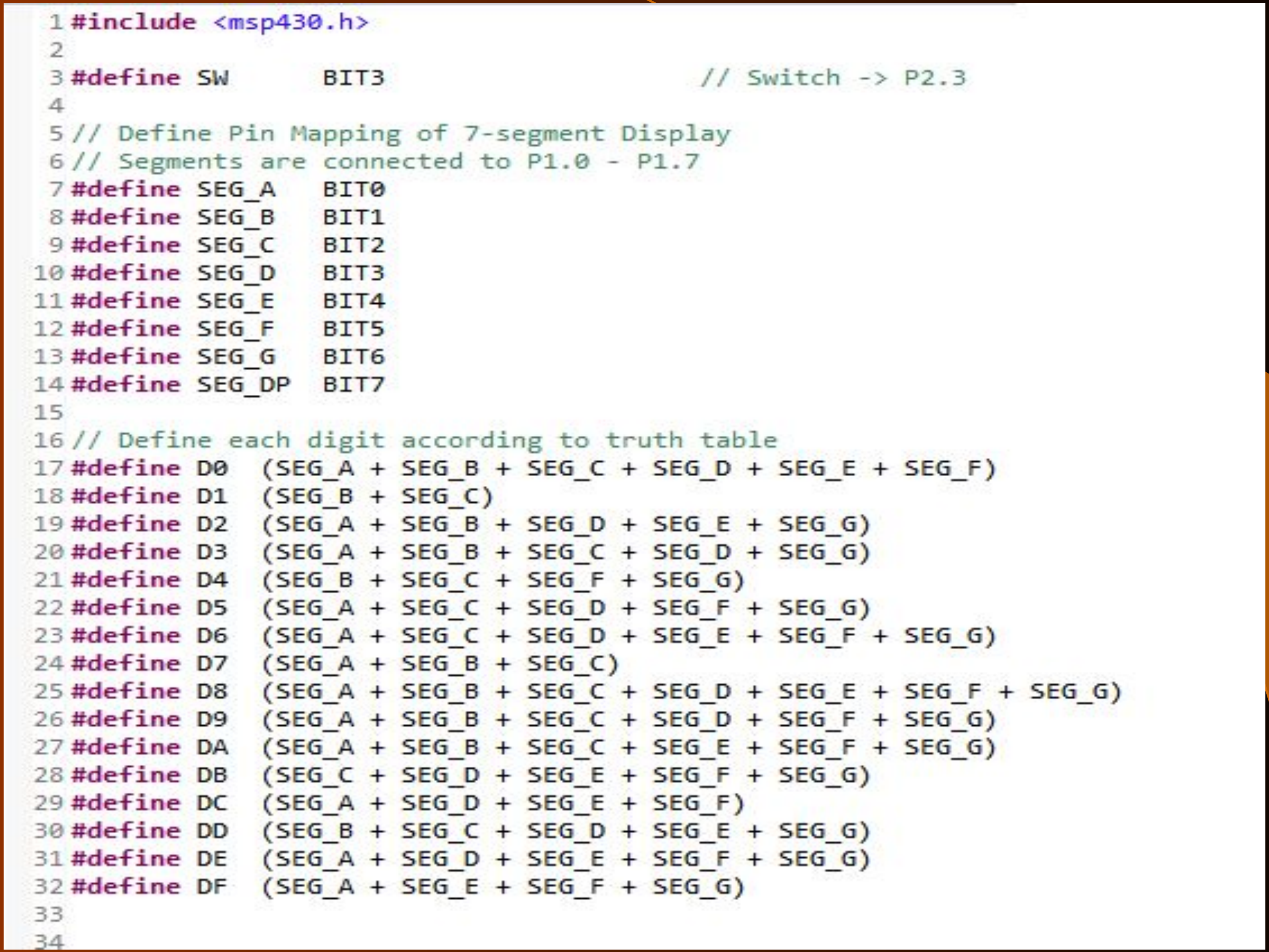
10uF Electrolytic Capacitor

Here, we are using digit 1 of the 4 Digit SSD

Common Cathode 4 digit SSD

0.1uF Ceramic Capacitor

Code Example: HelloSSD

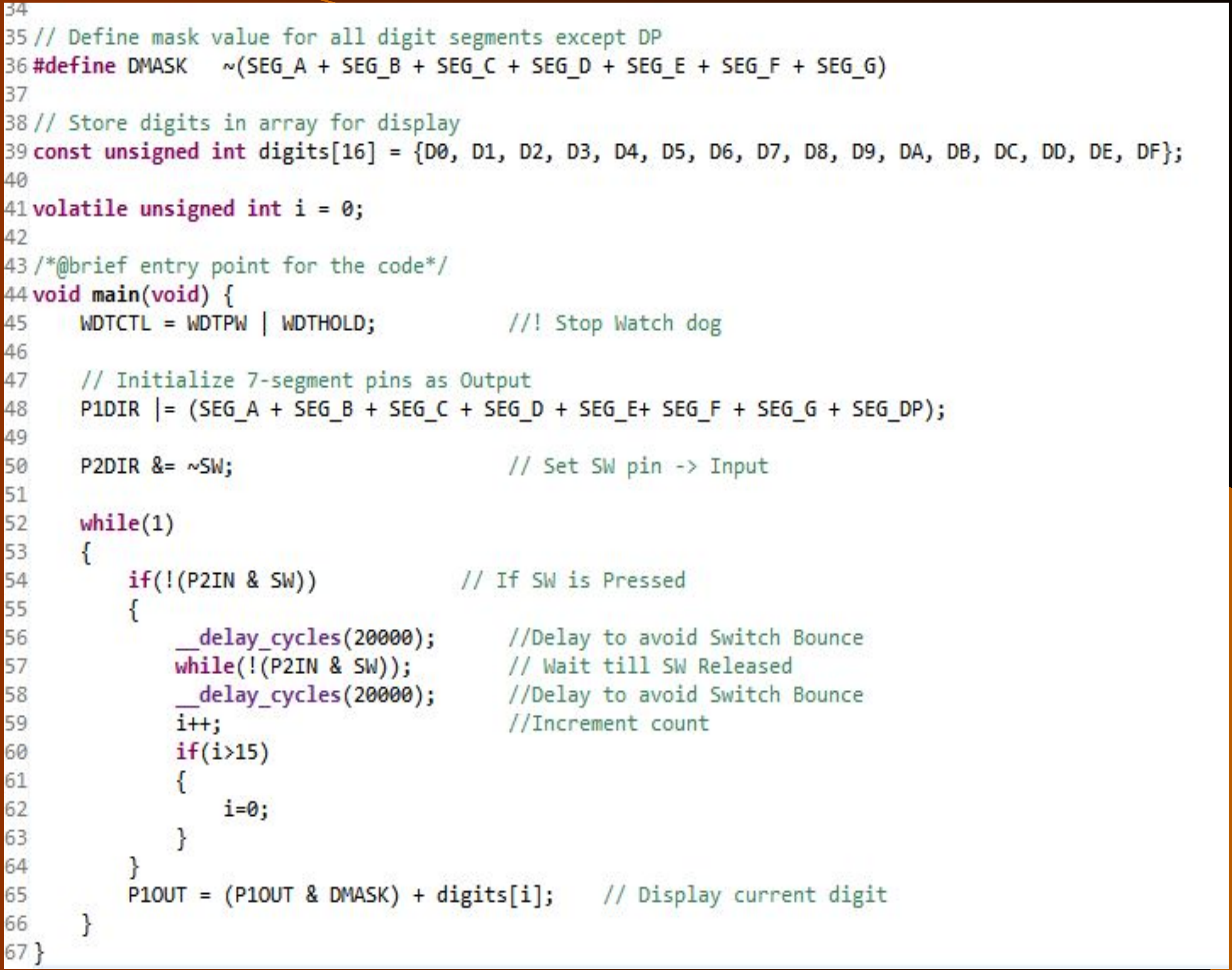


```c
#include <msp430.h>

#define SW      BIT3                    // Switch -> P2.3

// Define Pin Mapping of 7-segment Display
// Segments are connected to P1.0 - P1.7
#define SEG_A   BIT0
#define SEG_B   BIT1
#define SEG_C   BIT2
#define SEG_D   BIT3
#define SEG_E   BIT4
#define SEG_F   BIT5
#define SEG_G   BIT6
#define SEG_DP  BIT7

// Define each digit according to truth table
#define D0  (SEG_A + SEG_B + SEG_C + SEG_D + SEG_E + SEG_F)
#define D1  (SEG_B + SEG_C)
#define D2  (SEG_A + SEG_B + SEG_D + SEG_E + SEG_G)
#define D3  (SEG_A + SEG_B + SEG_C + SEG_D + SEG_G)
#define D4  (SEG_B + SEG_C + SEG_F + SEG_G)
#define D5  (SEG_A + SEG_C + SEG_D + SEG_F + SEG_G)
#define D6  (SEG_A + SEG_C + SEG_D + SEG_E + SEG_F + SEG_G)
#define D7  (SEG_A + SEG_B + SEG_C)
#define D8  (SEG_A + SEG_B + SEG_C + SEG_D + SEG_E + SEG_F + SEG_G)
#define D9  (SEG_A + SEG_B + SEG_C + SEG_D + SEG_F + SEG_G)
#define DA  (SEG_A + SEG_B + SEG_C + SEG_E + SEG_F + SEG_G)
#define DB  (SEG_C + SEG_D + SEG_E + SEG_F + SEG_G)
#define DC  (SEG_A + SEG_D + SEG_E + SEG_F)
#define DD  (SEG_B + SEG_C + SEG_D + SEG_E + SEG_G)
#define DE  (SEG_A + SEG_D + SEG_E + SEG_F + SEG_G)
#define DF  (SEG_A + SEG_E + SEG_F + SEG_G)
```



```c

// Define mask value for all digit segments except DP
#define DMASK   ~(SEG_A + SEG_B + SEG_C + SEG_D + SEG_E + SEG_F + SEG_G)

// Store digits in array for display
const unsigned int digits[16] = {D0, D1, D2, D3, D4, D5, D6, D7, D8, D9, DA, DB, DC, DD, DE, DF};

volatile unsigned int i = 0;

/*@brief entry point for the code*/
void main(void) {
    WDTCTL = WDTPW | WDTHOLD;           //! Stop Watch dog

    // Initialize 7-segment pins as Output
    P1DIR |= (SEG_A + SEG_B + SEG_C + SEG_D + SEG_E+ SEG_F + SEG_G + SEG_DP);

    P2DIR &= ~SW;                       // Set SW pin -> Input

    while(1)
    {
        if(!(P2IN & SW))            // If SW is Pressed
        {
            __delay_cycles(20000);      //Delay to avoid Switch Bounce
            while(!(P2IN & SW));        // Wait till SW Released
            __delay_cycles(20000);      //Delay to avoid Switch Bounce
            i++;                        //Increment count
            if(i>15)
            {
                i=0;
            }
        }
        P1OUT = (P1OUT & DMASK) + digits[i];    // Display current digit
    }
}
```

Low Power Modes in MSP430

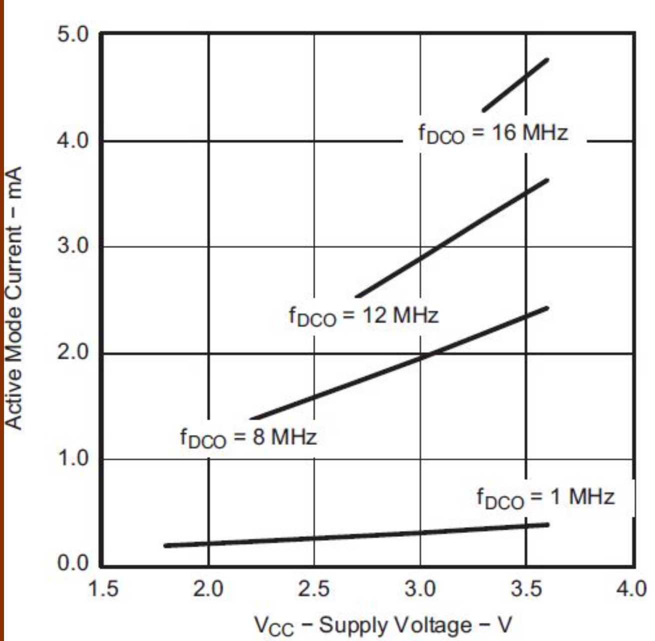
The MSP430 was designed primarily for low power applications and this is reflected in a range of low-power modes of operation.

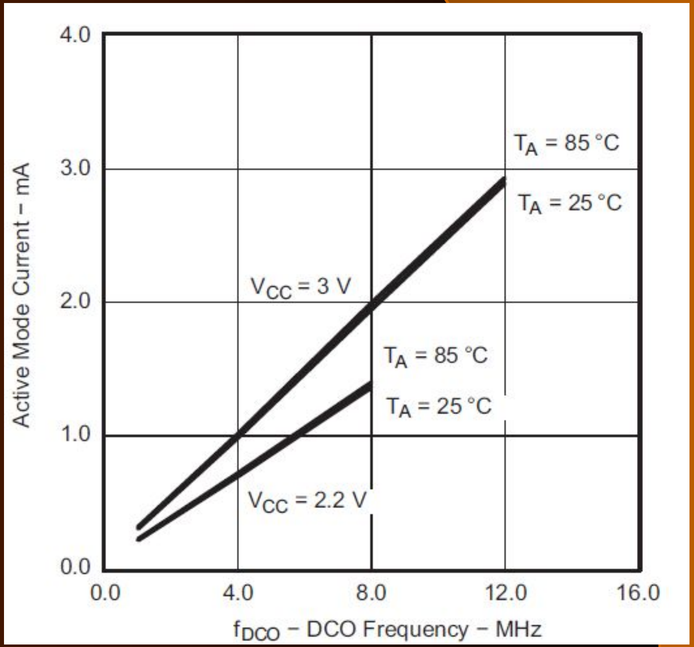
OPERATING MODES:

- Active Mode
- LPM0
- LPM1
- LPM2
- LPM3
- LPM4

Active Mode

- All clocks are active.
- SUPPLY CURRENT(at 1MHz):
  - 230uA(at 2.2V)
  - 330uA(at 3.0V)





LPM0

- CPU is disabled.
- ACLK and SMCLK remain active, MCLK is disabled.
- Supply current(at 1 MHz) : 56uA(at 2.2V)

LPM1

- CPU is disabled.
- ACLK and SMCLK remain active, MCLK is disabled.
- DCO and DC generator is disabled if DCO is not used for SMCLK

LPM2

- CPU is disabled.
- MCLK and SMCLK are disabled.
- DCO's DC generator remains enabled.
- ACLK remains active.
- Supply Current : 22uA(at 2.2V)

LPM3

- CPU is disabled.
- MCLK and SMCLK are disabled.
- DCO's DC generator is disabled.
- ACLK remains active.
- Supply Current : 0.5uA(at 2.2V)

LPM4

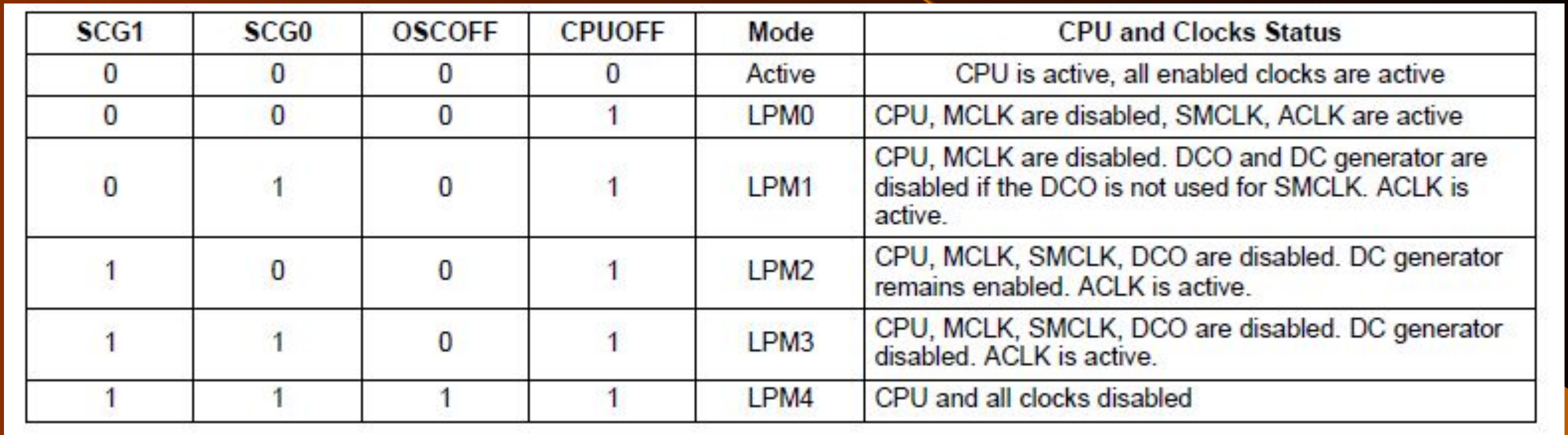
- CPU is disabled.
- ACLK is disabled.
- MCLK and SMCLK are disabled.
- DCO's DC generator is disabled.
- Crystal oscillator is stopped.
- Supply Current : 0.1uA(at 2.2V)

Controlling Low Power Modes

- LPM operations are controlled through 4 bits of the Status Register: SCG0, SCG1, CPUOFF and OSCOFF.

- All these bits are cleared in active mode and particular combinations are set for each LPM.

Controlling Low Power Modes



SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled, SMCLK, ACLK are active
0	1	0	1	LPM1	CPU, MCLK are disabled. DCO and DC generator are disabled if the DCO is not used for SMCLK. ACLK is active.
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO are disabled. DC generator remains enabled. ACLK is active.
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO are disabled. DC generator disabled. ACLK is active.
1	1	1	1	LPM4	CPU and all clocks disabled

- Two ways to put MSP430 into LPM Mode 4, for example, are:
  - __low_power_mode_4();   //This also enables the GIE bit.
  - __bis_SR_register(LPM4_bits + GIE);   //+GIE is used to enable interrupts.

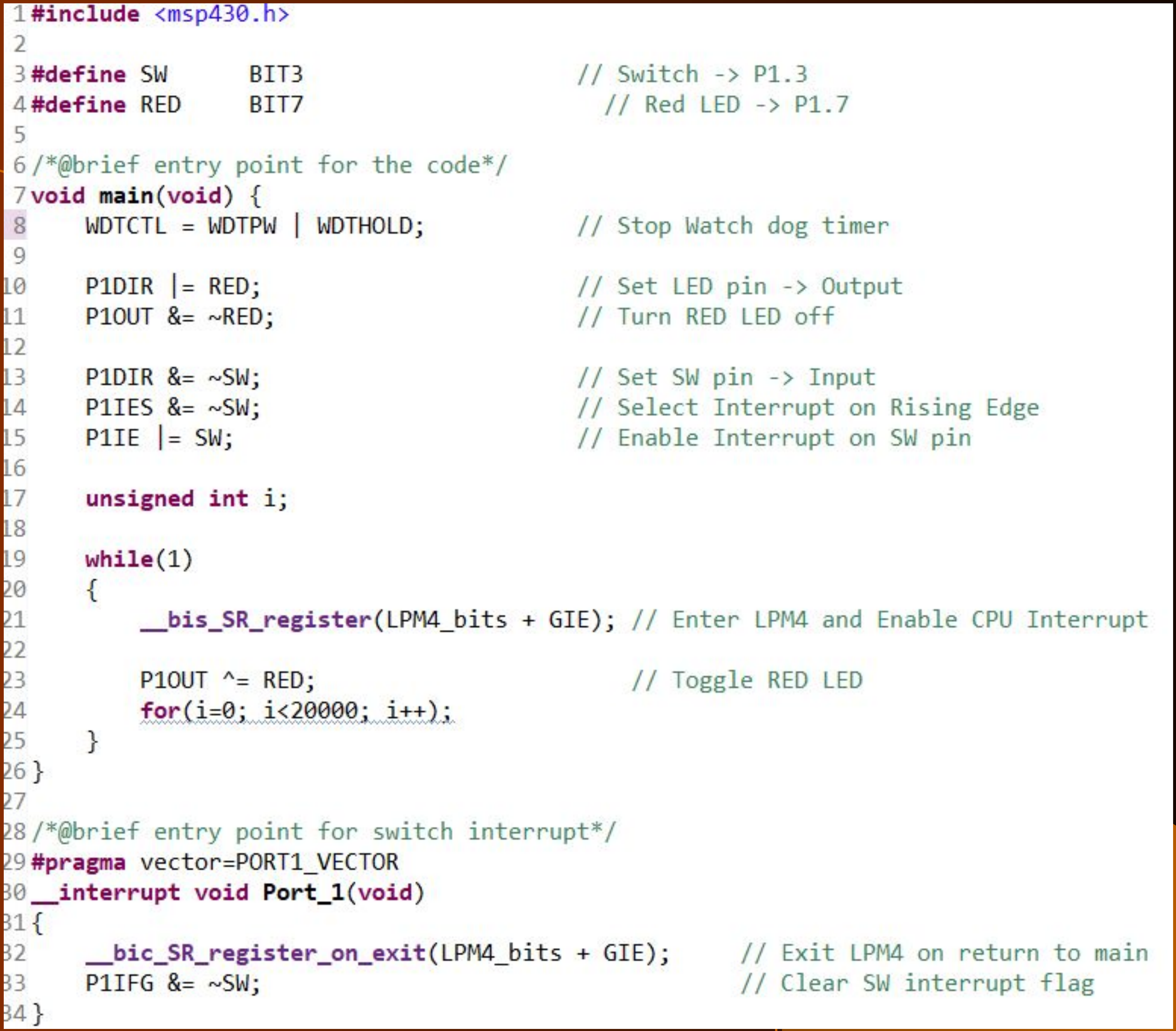
Interrupts and LPM

- MSP430 is designed to stay in a Low Power Mode for most of the time.
- The main function **generally** configures the peripherals, enables interrupts, puts the MSP430 into LPM, and plays no further role.
- MSP430 is woken up only when an interrupt comes.
- An LPM is suspended whenever an enabled interrupt is serviced.
- On entering the ISR, the SR is stored on stack and the CPUOFF, SCG1, OSCOFF bits are automatically reset.
- On exiting the ISR, the original SR is popped from the stack and the previous operating mode is restored, so that the LPM is resumed after the ISR.

Ultra Low Power Mode Advisor

- ULP Mode Advisor is integrated into CCS.
- It checks your code against a thorough checklist to achieve the lowest power possible and provides detailed notifications and remarks

Code Example: HelloLPM



```c
#include <msp430.h>

#define SW      BIT3                    // Switch -> P1.3
#define RED     BIT7                      // Red LED -> P1.7

/*@brief entry point for the code*/
void main(void) {
    WDTCTL = WDTPW | WDTHOLD;           // Stop Watch dog timer

    P1DIR |= RED;                       // Set LED pin -> Output
    P1OUT &= ~RED;                      // Turn RED LED off

    P1DIR &= ~SW;                       // Set SW pin -> Input
    P1IES &= ~SW;                       // Select Interrupt on Rising Edge
    P1IE |= SW;                         // Enable Interrupt on SW pin

    unsigned int i;

    while(1)
    {
        __bis_SR_register(LPM4_bits + GIE); // Enter LPM4 and Enable CPU Interrupt

        P1OUT ^= RED;                       // Toggle RED LED
        for(i=0; i<20000; i++);
    }
}

/*@brief entry point for switch interrupt*/
#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    __bic_SR_register_on_exit(LPM4_bits + GIE);     // Exit LPM4 on return to main
    P1IFG &= ~SW;                                   // Clear SW interrupt flag
}
```

Thank you!