CPE 323
MODULE 09: APPENDIX A
MSP430 Universal Clock Subsystem

Aleksandar Milenković

Email: milenka@uah.edu

Web: http://www.ece.uah.edu/~milenka

Overview

This module discusses the MSP430's Universal Clock Subsystem.

Contents

1 Clock Module

So far we wrote a number of MSP430 programs assuming that the processor clock is around 1 µs (i.e., the clock frequency is approximately 1 MHz). The MSP430 family supports several clock modules and a user has a full control over these modules. By changing the content of relevant clock module control registers, one can change the processor clock frequency, as well as the frequency of other clock signals that are used for peripheral devices. In this section, we will discuss the organization of the Unified Clock System (UCS) used in the MSP430F5529 device.

1.1 Unified Clock System (UCS)

MSP430x5xx family of microcontrollers have Unified Clock System (UCS) that provides various clocks for the MSP430 modules. The UCS module includes up to five clock sources and provide three clock signals.

The five clock sources supported by the UCS module are as follows:

- **XT1CLK**: Low-frequency or high-frequency oscillator that can be used either with low-frequency 32768 Hz watch crystals, standard crystals, resonators, or external clock sources in the 4 MHz to 32 MHz range. XT1CLK can be used as a clock reference into the FLL. Some devices only support the low frequency oscillator for XT1CLK.
- **VLOCLK**: Internal very low power, low-frequency oscillator with 10 KHz typical frequency.
- **REFOCLK**: Internal trimmed low-frequency oscillator with 32768 Hz typical frequency, can be used as a clock reference into the FLL.
- **DCOCLK**: Internal digitally controlled oscillator (DCO) that can be stabilized by the FLL.
- **XT2CLK**: Optional high-frequency oscillator that can be used with standard crystals, resonators, or external clock sources in the 4 MHz to 32 MHz range. XT2CLK can be used as a clock reference into the FLL.

Following are the clock signals provided by the USC module:

- **ACLK**: Auxiliary clock. The ACLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. ACLK can be divided by 1, 2, 4, 8, 16, or 32. ACLK/n is ACLK divided by 1, 2, 4, 8, 16, or 32 and is available externally at a pin. ACLK is software selectable by individual peripheral modules.
- **MCLK**: Master (Main) clock. MCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. MCLK can be divided by 1, 2, 4, 8, 16, or 32. MCLK is used by the CPU and system.
- **SMCLK**: Subsystem master clock. SMCLK is software selectable as XT1CLK, REFOCLK, VLOCLK, DCOCLK, DCOCLKDIV, and when available, XT2CLK. DCOCLKDIV is the DCOCLK frequency divided by 1, 2, 4, 8, 16, or 32 within the FLL block. SMCLK can be divided by 1, 2, 4, 8, 16, or 32. SMCLK is software selectable by individual peripheral modules.

On a PUC, the configuration of UCS is as follows:

- XT1 in LF mode is selected as source for XT1CLK which is selected for ACLK.
- DCOCLKDIV is selected for MCLK and SMCLK.
- FLL operation is enabled and XT1CLK is selected as reference clock for FLL.
- If the 32768 Hz crystal is used for XT1CLK, fault control logic sources ACLK with 32768 Hz REFOCLK because XT1 takes some time to stabilize.
- When the crystal start-up is obtained, FLL stabilizes MCLK and SMCLK at 1048576 Hz (2^20 Hz). f_{DCO} = 2097152 Hz.

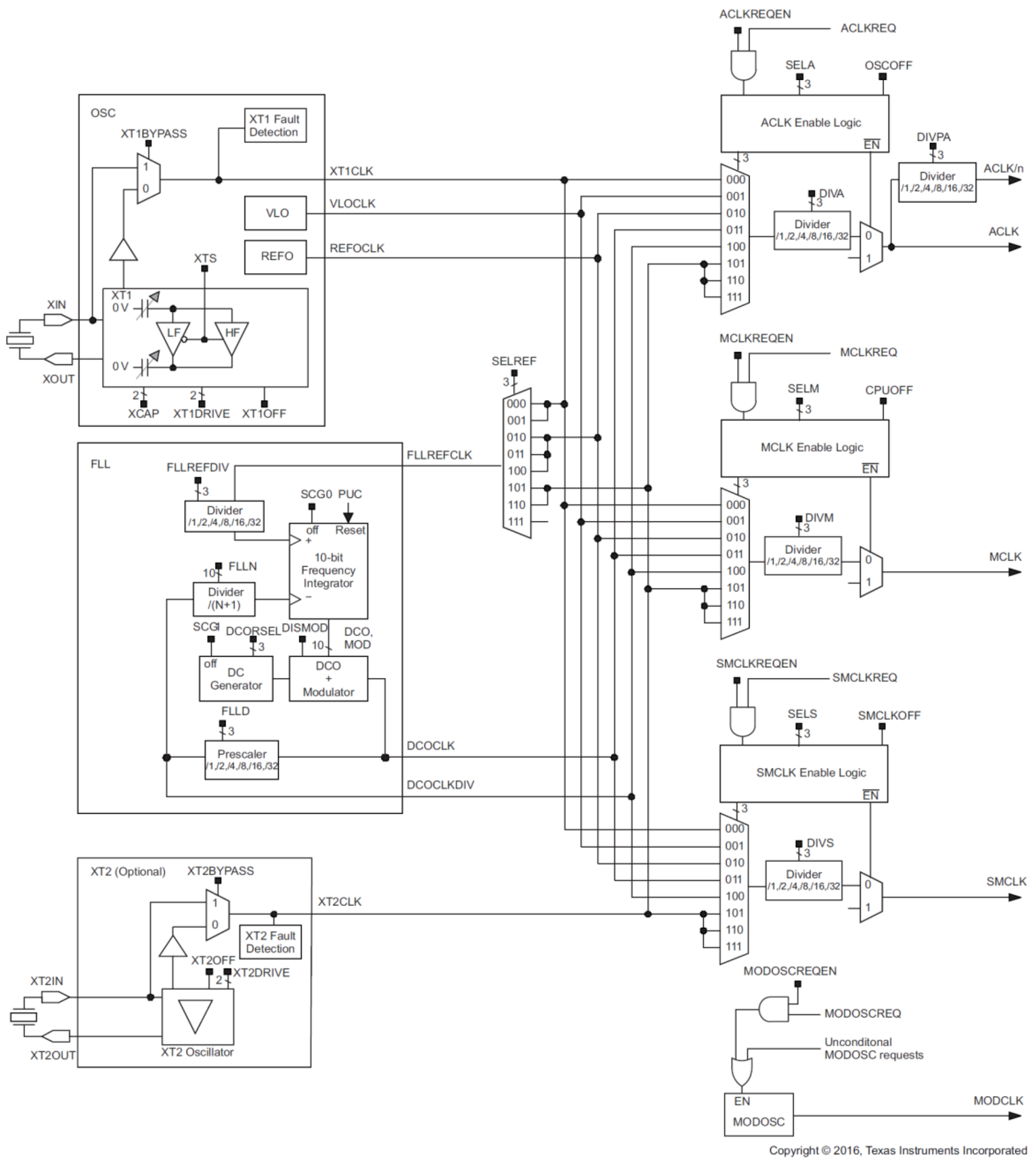

Figure 1. Block diagram of UCS module in MSP430x5xx devices

The operating modes of MSP430 are configured by the status register (R2) control bits: **SCG0**, **SCG1**, **OSCOFF** and **CPUOFF**. The operating modes are as follows:

- Active (SCG=0, SCG1=0, OSCOFF=0, CPUOFF=0): CPU is active and all other modules are active.
- LPM0 (SCG=0, SCG1=0, OSCOFF=0, CPUOFF=1): CPU is off, FLL is on, ACLK is on, Vcore is on.

- LPM1 (SCG=1, SCG1=0, OSCOFF=0, CPUOFF=1): CPU if off, FLL is off, ACLK is on, Vcore is on.
- LPM2 (SCG=0, SCG1=1, OSCOFF=0, CPUOFF=1): CPU if off, FLL is off, ACLK is on, Vcore is on.
- LPM3 (SCG=1, SCG1=1, OSCOFF=0, CPUOFF=1): CPU if off, FLL is off, ACLK is on, Vcore is on.
- LPM4 (SCG=1, SCG1=1, OSCOFF=1, CPUOFF=1): CPU if off, FLL is off, ACLK is off, Vcore is on.

The UCS module can be configured using registers from **UCSCTL0** through **UCSCTL8**.

Digital Controlled Oscillator (DCO):

- The frequency of DCO can be adjusted by software using DCORSEL (in UCSCTL1 register), DCO and MOD bits (in UCSCTL0).
- DCO can also be stabilized using FLL to a multiple of FLL reference clock (i.e. multiple frequency of FLLREFCLK/n). FLL can accept different reference sources selectable by SELREF bits (UCSCTL3). The reference sources can be XT1CLK, REFOCLK or XT2CLK if available.
- The value of *n* can be defined in FLLREFDIV bits in UCSCTL3 register (n = 1, 2, 4, 8, 12 or 16). Default value is *n*=1.
- FLLD bits (in UCSCTL2 register) can be configured for FLL pre-scalar divider value D of 1, 2, 4, 8, 16 or 32. The default of D = 2 and MCLK and SMCLK are sourced from DCOCLKDIV thus providing clock frequency of DCOCLK/2.
- The divider (N+1) and divider D define DCOCLK and DCOCLKDIV frequencies. Value N+1 can be set from FLLN bits (in UCSCTL2 register) where N>0. Setting FLLN to 0 will cause FLLN to be 1 to prevent unintentional write.
- The final frequency of DCOCLK and DCOCLKDIV are as follows

$$f_{DCOCLK} = D \times (N + 1) \times (f_{FLLREFCLK} \div n)$$
$$f_{DCOCLKDIV} = (N + 1) \times (f_{FLLREFCLK} \div n)$$

Figure 2 Formula to compute DCOCLK and DCOCLKDIV frequencies

Frequency Locked Loop (FLL):

- The FLL continuously counts up or down a frequency integrator.
- The count is adjusted +1 with the frequency $f_{FLLREFCLK}$/n (n=1, 2, 4, 8, 12 or 16) or -1 with the frequency f_{DCOCLK}/[D*(N+1)]
- Five of the integrator bits (UCSCTL0bits 12 to 8) set the DCO frequency tap. Thirty-two taps are implemented for the DCO, and each is approximately8% higher than the

previous. The modulator mixes two adjacent DCO frequencies to produce fractional taps. For a given DCO bias range setting, time must be allowed for DCO to settle on the proper tap operation. (n*32)$f_{FLLREFCLK}$ cycles are required between taps requiring worst case of (n*32*32)$f_{FLLREFCLK}$ cycles for DCO to settle.

1.2 Changing Processor Clocks: Examples

The following examples illustrate (Code 1, Code 2, and Code 3) how you can change the processor clock frequency by modifying individual bits in the control registers. Please note that these examples only change the clocks and make them visible on external ports (some digital I/O ports have a special function to pass the clocks to the output, so we can observe them from the outside by connecting to oscilloscope). For learning how internal digitally controlled oscillator works read the corresponding user manual.

```
/*******************************************************************************
 *  File:        Lab6_D6.c
 *  Description: MSP430F5529 Demo - FLL, Runs Internal DCO at 2.45MHz
 *               This program demonstrates setting the internal DCO to run at
 *               2.45MHz.
 *  Clocks:      ACLK = 32768Hz,
 *               MCLK = SMCLK = DCO = (74+1) x ACLK = 2457600Hz
 *
 *                MSP430F5529
 *            -----------------
 *        /|\|              XIN|-
 *         | |                 | 32kHz
 *         --|RST          XOUT|-
 *           |                 |
 *           |             P7.7|--> MCLK = 2.45MHz
 *           |                 |
 *           |             P2.2|--> SMCLK = 2.45MHz
 *           |             P1.0|--> ACLK = 32kHz
 *           |                 |
 *
 *  Author:     Aleksandar Milenkovic, milenkovic@computer.og
 *  Date:       September 2010
 *  Modified:   Prawar Poudel, August 2020
 *******************************************************************************/
#include  <msp430.h>

void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                   // Stop watchdog timer
  P1DIR |= BIT0;                              // ACLK set out to pins
  P1SEL |= BIT0;
  P2DIR |= BIT2;                              // SMCLK set out to pins
  P2SEL |= BIT2;
  P7DIR |= BIT7;                              // MCLK set out to pins
  P7SEL |= BIT7;

  UCSCTL3 = SELREF_2;                         // Set DCO FLL reference = REFO
  UCSCTL4 |= SELA_2;                          // Set ACLK = REFO
```

```c
    UCSCTL0 = 0x0000;                          // Set lowest possible DCOx, MODx

    // Loop until XT1,XT2 & DCO stabilizes - In this case only DCO has to stabilize
    do
    {
      UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);
                                               // Clear XT2,XT1,DCO fault flags
      SFRIFG1 &= ~OFIFG;                       // Clear fault flags
    } while (SFRIFG1&OFIFG);                   // Test oscillator fault flag

    __bis_SR_register(SCG0);                   // Disable the FLL control loop
    UCSCTL1 = DCORSEL_4;                       // Select DCO range for operation
    UCSCTL2 |= 74;                             // Set DCO Multiplier for 2.45MHz
                                               // (N + 1) * FLLRef = Fdco
                                               // (74 + 1) * 32768 = 2.45MHz
    __bic_SR_register(SCG0);                   // Enable the FLL control loop

    // Worst-case settling time for the DCO when the DCO range bits have been
    // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
    // UG for optimization.
    // 32 x 32 x 2.45 MHz / 32,768 Hz = 76600 = MCLK cycles for DCO to settle
    __delay_cycles(76000);

    while(1);
  }
```

Code 1. Changing DCO to Run at 2.45 MHz using UCS Module (Lab6_D6.c)

```c
/*****************************************************************************
 *  File:         Lab6_D7.c
 *  Description: MSP430F5529 Demo - FLL, Runs Internal DCO at 8MHz
 *                This program demonstrates setting the internal DCO to run at
 *                8MHz.
 *  Clocks:       ACLK = 32768Hz,
 *                MCLK = SMCLK = DCO = (121+1) x 2 x ACLK = 7995392Hz
 *
 *                 MSP430F5529
 *             -----------------
 *         /|\|              XIN|-
 *          | |                 | 32kHz
 *          --|RST          XOUT|-
 *            |                 |
 *            |             P7.7|--> MCLK = 8MHz
 *            |                 |
 *            |             P2.2|--> SMCLK = 8MHz
 *            |             P1.0|--> ACLK = 32kHz
 *            |                 |
 *
 *  Author:     Aleksandar Milenkovic, milenkovic@computer.og
 *  Date:       September 2010
 *  Modified:   Prawar Poudel
 *  Date:       August 2020
 ******************************************************************************/

#include  <msp430.h>

void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;            // Stop watchdog timer

   P1DIR |= BIT0;                       // ACLK set out to pins
   P1SEL |= BIT0;
   P2DIR |= BIT2;                       // SMCLK set out to pins
   P2SEL |= BIT2;
   P7DIR |= BIT7;                       // MCLK set out to pins
   P7SEL |= BIT7;

   UCSCTL3 = SELREF_2;                  // Set DCO FLL reference = REFO
   UCSCTL4 |= SELA_2;                   // Set ACLK = REFO
   UCSCTL0 = 0x0000;                    // Set lowest possible DCOx, MODx

   // Loop until XT1,XT2 & DCO stabilizes - In this case only DCO has to stabilize
   do
   {
     UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);
                                             // Clear XT2,XT1,DCO fault flags
     SFRIFG1 &= ~OFIFG;                      // Clear fault flags
   } while (SFRIFG1&OFIFG);                  // Test oscillator fault flag

   __bis_SR_register(SCG0);                  // Disable the FLL control loop
   UCSCTL1 = DCORSEL_5;                      // Select DCO range 8MHz operation
   UCSCTL2 |= 249;                           // Set DCO Multiplier for 8MHz
                                             // (N + 1) * FLLRef = Fdco
                                             // (249 + 1) * 32768 = 8MHz
   __bic_SR_register(SCG0);                  // Enable the FLL control loop

   // Worst-case settling time for the DCO when the DCO range bits have been
   // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
   // UG for optimization.
   // 32 x 32 x 8 MHz / 32,768 Hz = 250000 = MCLK cycles for DCO to settle
   __delay_cycles(250000);
   while(1);                          // Loop in place
}
```

Code 2. Changing DCO to Run at 8 MHz using UCS Module (Lab6_D7.c)

```c
/******************************************************************************
 *  File:         Lab6_D8.c
 *  Description: MSP430F5529 Demo - FLL, Runs Internal DCO at 1MHz
 *                This program demonstrates setting the internal DCO to run at
 *                8MHz when SW1 is pressed.
 *
 *                A LED will keep blinking at 1Hz (0.5s ON and 0.5s OFF) at
 *                clock running at default frequency of 1Mhz. When SW1 is pressed,
 *                the frequency is changed to ~ 8 Mhz. When SW1 is pressed again,
 *                the frequency is restored to 1Mhz.
 *
 *  Clocks:       ACLK = 32768Hz,
 *                MCLK = SMCLK = DCO = (249+1) x ACLK = 8192000Hz
 *
 * Input:        SW1 (P2.1)
 * Output:       LED2 (P4.7)
 *
 *                 MSP430F5529
 *             -----------------
 *         /|\|              XIN|-
 *          | |                 | 32kHz
 *          --|RST          XOUT|-
 *            |                 |
 *      SW1-->|P2.1         P7.7|--> MCLK = 1 or 8MHz
 *     LED2<--|P4.7             |
 *            |             P2.2|--> SMCLK = 1 or 8MHz
 *            |             P1.0|--> ACLK = 32kHz
 *            |                 |
 *
 *  Modified:   Prawar Poudel
 *  Date:       August 2020
 ******************************************************************************/

// mandatory include statement
#include  <msp430.h>

// this function configures the clock sources as follows
// .. use internal REFOCLK for FLL reference clock (UCSCTL3 = SELREF_2)
// .. ACLK is sourced with REFOCLK (UCSCTL4 |= SELA_2)
// .. sets DCO tap to 0 (UCSCTL0 = 0)
// .. sets the modulation bit counter value to 0 (UCSCTL0 = 0)
void configure_clock_sources();
// this function changes the frequency of clock to 8 MHZ
inline void change_clock_freq_8Mhz();
// this function changes the frequency of clock to 8 MHZ
inline void change_clock_freq_1Mhz();


char is8Mhz = 0;

void main(void)
{
   WDTCTL = WDTPW + WDTHOLD;             // Stop watchdog timer

   P1DIR |= BIT0;                        // ACLK set out to pins
   P1SEL |= BIT0;
```

```

    P2DIR |= BIT2;                      // SMCLK set out to pins
    P2SEL |= BIT2;

    P7DIR |= BIT7;                      // MCLK set out to pins
    P7SEL |= BIT7;

    _EINT();                            // enable interrupts
    P2DIR &= ~BIT1;                     // set P2.1 as input (SW1)
    P2REN |= BIT1;                      // enable pull-up resistor
    P2OUT |= BIT1;
    P2IE |= BIT1;                       // enable interrupt at P2.1
    P2IES |= BIT1;                      // enable hi->lo edge for interrupt
    P2IFG &= ~BIT1;                     // clear any errornous interrupt flag

    P4DIR |= BIT7;                      // set P4.7 as output (LED2)

    configure_clock_sources();          // configure the clock sources

    while(1)                            // Loop in place (infinite)
    {
        P4OUT ^= BIT7;                  // toggle LED2
        __delay_cycles(500000);         // arbitrary delay of 500ms
    }
}

// this ISR handles the SW1 key press
#pragma vector = PORT2_VECTOR
__interrupt void PORT2_ISR(void)
{
    // let us clear the flag
    P2IFG &= ~BIT1;

    //debouncing section
    __delay_cycles(25000);

    // if SW1 is not pressed, return
    if((P2IN&BIT1)!=0x00)
        return;


    if(is8Mhz==0)
    {
        // if not at 8Mhz, let us change to 8Mhz
        change_clock_freq_8Mhz();
        is8Mhz = 1;
    }else
    {
        // if already in 8Mhz, let us take back to 1Mhz
        change_clock_freq_1Mhz();
        is8Mhz = 0;
    }
}

// this function changes the frequency of clock to 8 MHZ
```

```
void change_clock_freq_8Mhz()
{
    __bis_SR_register(SCG0);                  // Disable the FLL control loop
    UCSCTL1 = DCORSEL_5;                      // Select DCO range 8MHz operation
    UCSCTL2 = 249;                            // Set DCO Multiplier for 8MHz
                                              // (N + 1) * FLLRef = Fdco
                                              // (249 + 1) * 32768 = 8MHz
    __bic_SR_register(SCG0);                  // Enable the FLL control loop

    // Worst-case settling time for the DCO when the DCO range bits have been
    // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
    // UG for optimization.
    // 32 x 32 x 8 MHz / 32,768 Hz = 250000 = MCLK cycles for DCO to settle
    __delay_cycles(250000);
}

// this function changes the frequency of clock to 1 MHZ
void change_clock_freq_1Mhz()
{
    __bis_SR_register(SCG0);                  // Disable the FLL control loop
    UCSCTL1 = DCORSEL_3;                      // Select DCO range 1MHz operation
    UCSCTL2 = 32;                             // Set DCO Multiplier for 1MHz
                                              // (N + 1) * FLLRef = Fdco
                                              // (32 + 1) * 32768 = 1MHz
    __bic_SR_register(SCG0);                  // Enable the FLL control loop

    // Worst-case settling time for the DCO when the DCO range bits have been
    // changed is n x 32 x 32 x f_MCLK / f_FLL_reference. See UCS chapter in 5xx
    // UG for optimization.
    // 32 x 32 x 1 MHz / 32,768 Hz = 33792 = MCLK cycles for DCO to settle
    __delay_cycles(33792);
}


// this function configures the clock sources as follows
// .. use internal REFOCLK for FLL reference clock (UCSCTL3 = SELREF_2)
// .. ACLK is sourced with REFOCLK (UCSCTL4 |= SELA_2)
// .. sets DCO tap to 0 (UCSCTL0 = 0)
// .. sets the modulation bit counter value to 0 (UCSCTL0 = 0)
void configure_clock_sources()
{
    UCSCTL3 = SELREF_2;                  // Set DCO FLL reference = REFO
    UCSCTL4 |= SELA_2;                   // Set ACLK = REFO
    UCSCTL0 = 0x0000;                    // Set lowest possible DCOx, MODx

    // Loop until XT1,XT2 & DCO stabilizes - In this case only DCO has to stabilize
    do
    {
    UCSCTL7 &= ~(XT2OFFG + XT1LFOFFG + DCOFFG);   // Clear XT2,XT1,DCO fault flags
    SFRIFG1 &= ~OFIFG;                            // Clear fault flags
    } while (SFRIFG1&OFIFG);                      // Test oscillator fault flag
}
```

Code 3. Changing Clock Frequency to Observe Change in LED Blinking