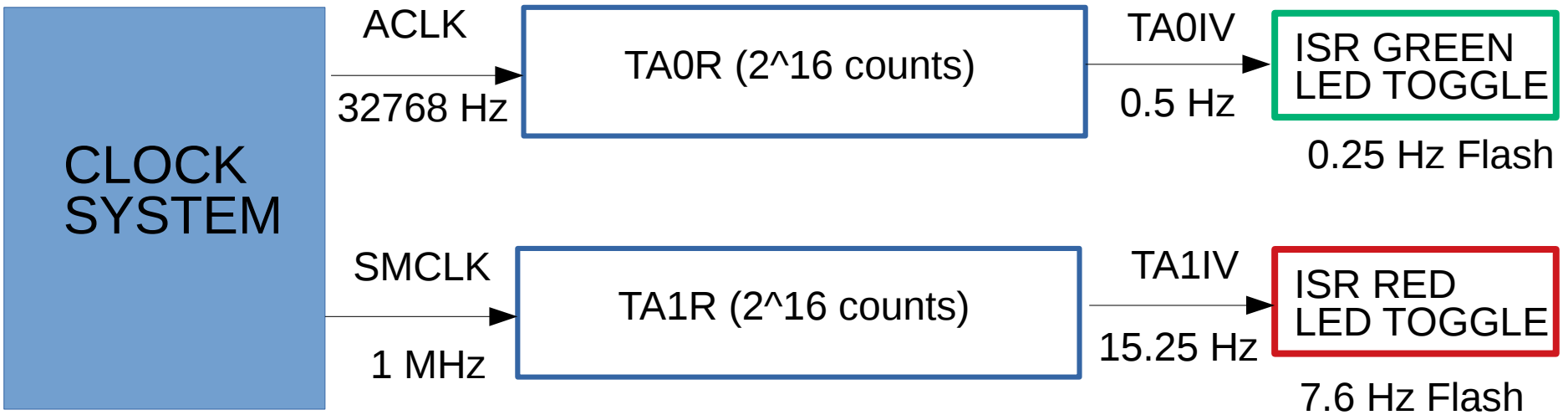
Clock System Assignment Project

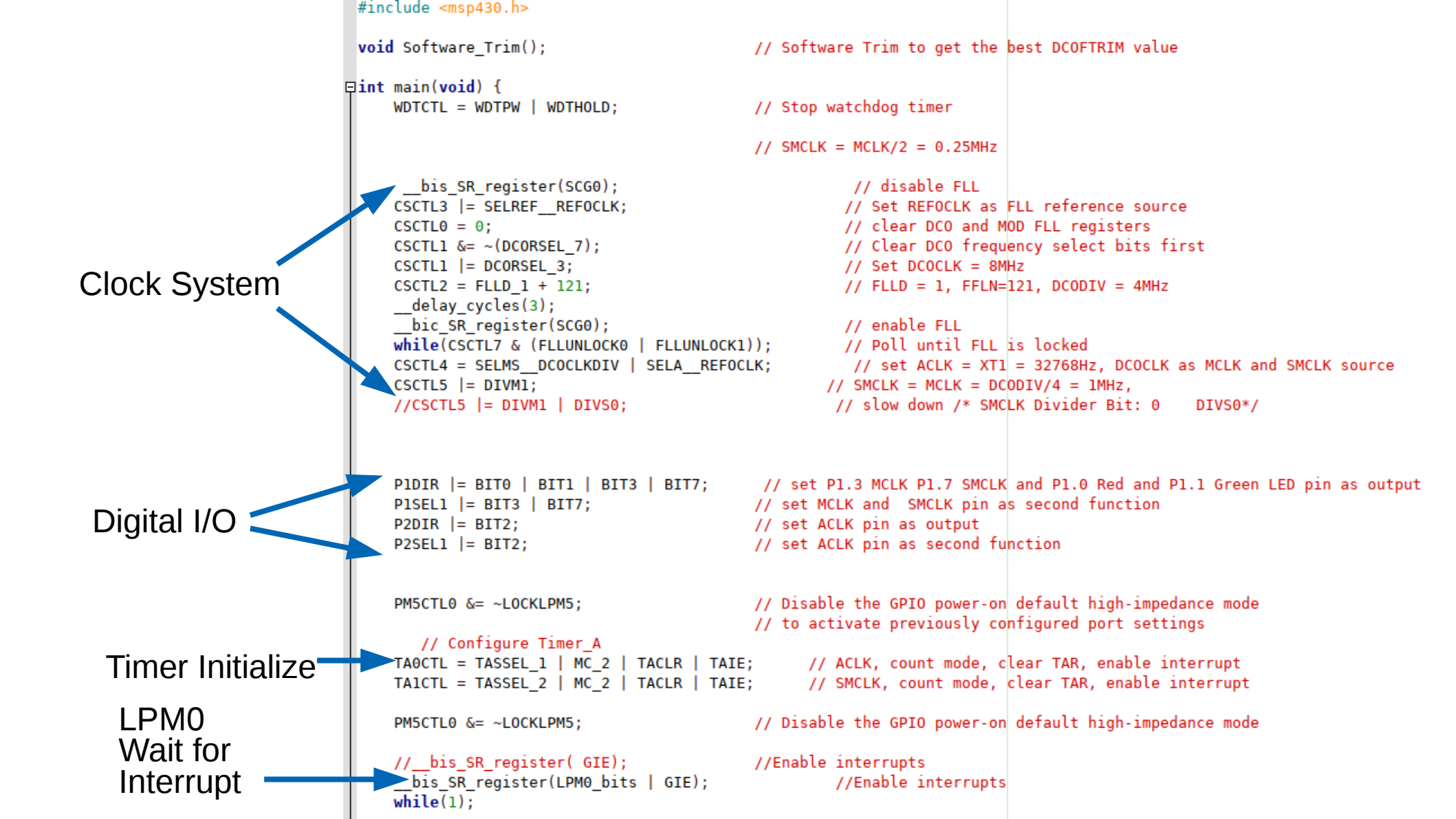
ASSIGNMENT M5 BLOCK DIAGRAM



Setup

Loop

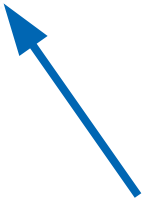
```c
#include <msp430.h>

void Software_Trim();                       // Software Trim to get the best DCOFTRIM value

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

                                            // SMCLK = MCLK/2 = 0.25MHz


     __bis_SR_register(SCG0);                           // disable FLL
    CSCTL3 |= SELREF__REFOCLK;                         // Set REFOCLK as FLL reference source
    CSCTL0 = 0;                                        // clear DCO and MOD FLL registers
    CSCTL1 &= ~(DCORSEL_7);                            // Clear DCO frequency select bits first
    CSCTL1 |= DCORSEL_3;                               // Set DCOCLK = 8MHz
    CSCTL2 = FLLD_1 + 121;                             // FLLD = 1, FFLN=121, DCODIV = 4MHz
    __delay_cycles(3);
    __bic_SR_register(SCG0);                           // enable FLL
    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));         // Poll until FLL is locked
    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK;          // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
    CSCTL5 |= DIVM1;                                  // SMCLK = MCLK = DCODIV/4 = 1MHz,
    //CSCTL5 |= DIVM1 | DIVS0;                         // slow down /* SMCLK Divider Bit: 0    DIVS0*/



    P1DIR |= BIT0 | BIT1 | BIT3 | BIT7;       // set P1.3 MCLK P1.7 SMCLK and P1.0 Red and P1.1 Green LED pin as output
    P1SEL1 |= BIT3 | BIT7;                    // set MCLK and  SMCLK pin as second function
    P2DIR |= BIT2;                            // set ACLK pin as output
    P2SEL1 |= BIT2;                           // set ACLK pin as second function


    PM5CTL0 &= ~LOCKLPM5;                     // Disable the GPIO power-on default high-impedance mode
                                              // to activate previously configured port settings
       // Configure Timer_A
    TA0CTL = TASSEL_1 | MC_2 | TACLR | TAIE;      // ACLK, count mode, clear TAR, enable interrupt
    TA1CTL = TASSEL_2 | MC_2 | TACLR | TAIE;      // SMCLK, count mode, clear TAR, enable interrupt

    PM5CTL0 &= ~LOCKLPM5;                     // Disable the GPIO power-on default high-impedance mode

    //__bis_SR_register( GIE);                //Enable interrupts
    __bis_SR_register(LPM0_bits | GIE);             //Enable interrupts
    while(1);
```




```c
#include <msp430.h>

void Software_Trim();                       // Software Trim to get the best DCOFTRIM value

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

                                            // SMCLK = MCLK/2 = 0.25MHz


    __bis_SR_register(SCG0);                            // disable FLL
    CSCTL3 |= SELREF__REFOCLK;                          // Set REFOCLK as FLL reference source
    CSCTL0 = 0;                                         // clear DCO and MOD FLL registers
    CSCTL1 &= ~(DCORSEL_7);                             // Clear DCO frequency select bits first
    CSCTL1 |= DCORSEL_3;                                // Set DCOCLK = 8MHz
    CSCTL2 = FLLD_1 + 121;                              // FLLD = 1, FFLN=121, DCODIV = 4MHz
    __delay_cycles(3);
    __bic_SR_register(SCG0);                            // enable FLL
    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));          // Poll until FLL is locked
    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK;           // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
    CSCTL5 |= DIVM1;                                   // SMCLK = MCLK = DCODIV/4 = 1MHz,
    //CSCTL5 |= DIVM1 | DIVS0;                          // slow down /* SMCLK Divider Bit: 0    DIVS0*/



    P1DIR |= BIT0 | BIT1 | BIT3 | BIT7;      // set P1.3 MCLK P1.7 SMCLK and P1.0 Red and P1.1 Green LED pin as output
    P1SEL1 |= BIT3 | BIT7;                  // set MCLK and  SMCLK pin as second function
    P2DIR |= BIT2;                          // set ACLK pin as output
    P2SEL1 |= BIT2;                         // set ACLK pin as second function


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

       // Configure Timer_A
    TA0CTL = TASSEL_1 | MC_2 | TACLR | TAIE;      // ACLK, count mode, clear TAR, enable interrupt
    TA1CTL = TASSEL_2 | MC_2 | TACLR | TAIE;      // SMCLK, count mode, clear TAR, enable interrupt

    PM5CTL0 &= ~LOCKLPM5;                   // Disable the GPIO power-on default high-impedance mode

    //__bis_SR_register( GIE);              //Enable interrupts
    __bis_SR_register(LPM0_bits | GIE);             //Enable interrupts
    while(1);
```

Timer0_A3 Interrupt Service Routine

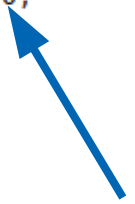
```
// Timer0_A3 Interrupt Vector (TAIV) handler  (ACLK/(2^16))*2
#pragma vector=TIMER0_A1_VECTOR
__interrupt void TIMER0_A1_ISR(void)
{
    switch(TA0IV)
    {
        case TA0IV_NONE:
            break;                          // No interrupt
        case TA0IV_TACCR1:
            break;                          // CCR1 not used
        case TA0IV_TACCR2:
            break;                          // CCR2 not used
        case TA0IV_TAIFG:
            P1OUT ^= BIT1;                  // overflow  GREEN= 4 sec  0.25 Hz
            break;
        default:
            break;
    }
}
```



Toggle GREEN LED

Timer1_A3 Interrupt Service Routine

```
// Timer1_A3 Interrupt Vector (TAIV) handler   (SMCLK/(2^16))*2
#pragma vector=TIMER1_A1_VECTOR
__interrupt void TIMER1_A1_ISR(void)
{
    switch(TA1IV)
    {
        case TA1IV_NONE:
            break;                              // No interrupt
        case TA1IV_TACCR1:
            break;                              // CCR1 not used
        case TA1IV_TACCR2:
            break;                              // CCR2 not used
        case TA1IV_TAIFG:
            P1OUT ^= BIT0;                      // overflow  RED= 131Msec 7.63Hz
            break;
        default:
            break;
    }
}
```



Toggle RED LED

FR2xx_4xx CS | Clock System

- Four independent clock sources
  - Low Frequency
    - XT1 32768 Hz crystal
    - VLO 10 kHz
  - High Frequency
    - DCO Specific ranges
    - MODCLK Internal 5MHz
- DCO
  - Default = 1MHz
  - FLL with REFO or XT1 reference
- ACLK = Only XT1 or REFO
- SMCLK and MCLK have same source selection
  - Though, SMCLK can be further divided
  - SMCLK can be active even if MCLK is off for LPM

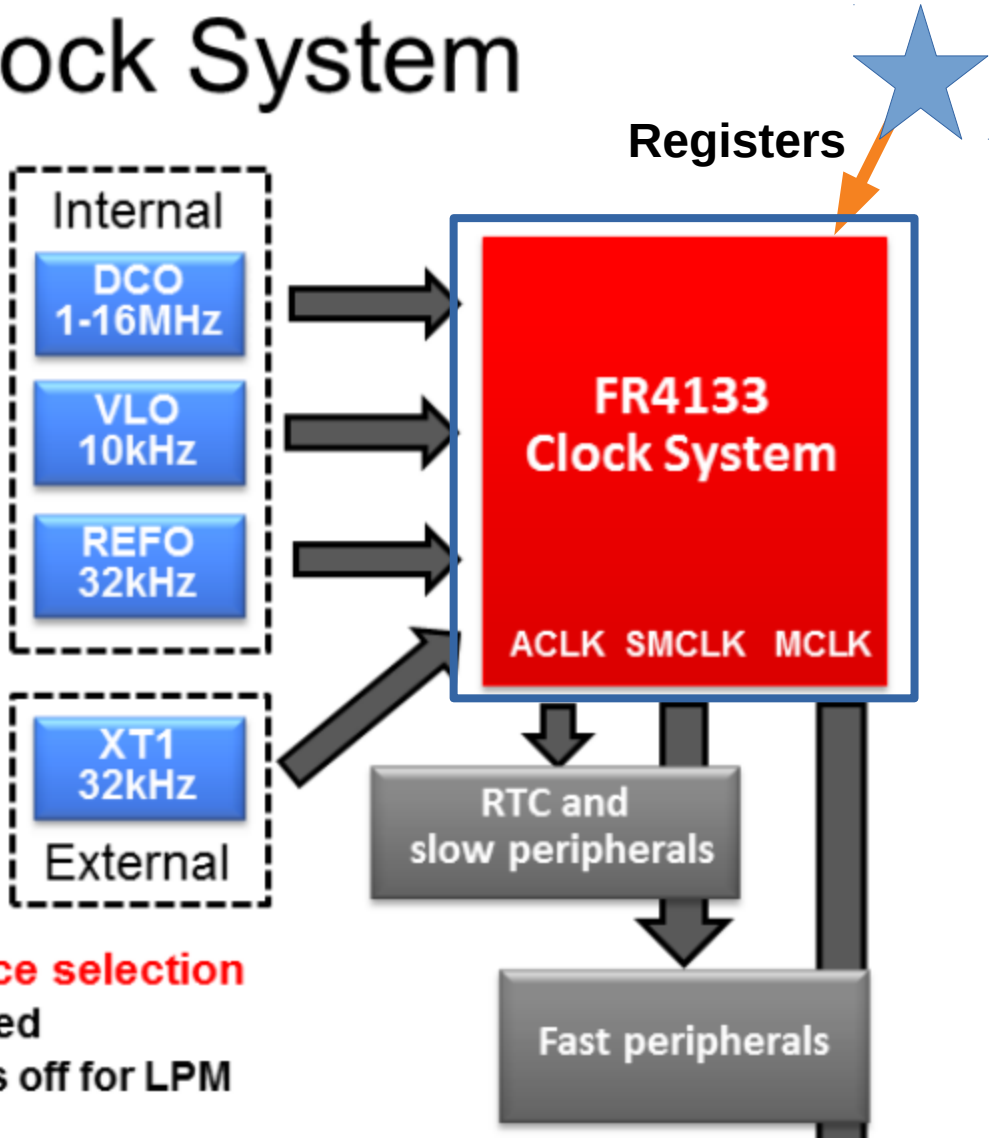


Table 3-2. CS Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	CSCTL0	Clock System Control Register 0	Read/write	Word	0000h	Section 3.3.1
02h	CSCTL1	Clock System Control Register 1	Read/write	Word	0033h	Section 3.3.2
04h	CSCTL2	Clock System Control Register 2	Read/write	Word	101Fh	Section 3.3.3
06h	CSCTL3	Clock System Control Register 3	Read/write	Word	0000h	Section 3.3.4
08h	CSCTL4	Clock System Control Register 4	Read/write	Word	0100h	Section 3.3.5
0Ah	CSCTL5	Clock System Control Register 5	Read/write	Word	1000h	Section 3.3.6
0Ch	CSCTL6	Clock System Control Register 6	Read/write	Word	08C1h	Section 3.3.7
0Eh	CSCTL7	Clock System Control Register 7	Read/write	Word	0740h	Section 3.3.8
10h	CSCTL8	Clock System Control Register 8	Read/write	Word	0007h	Section 3.3.9

Functional Block Diagram -Clock System

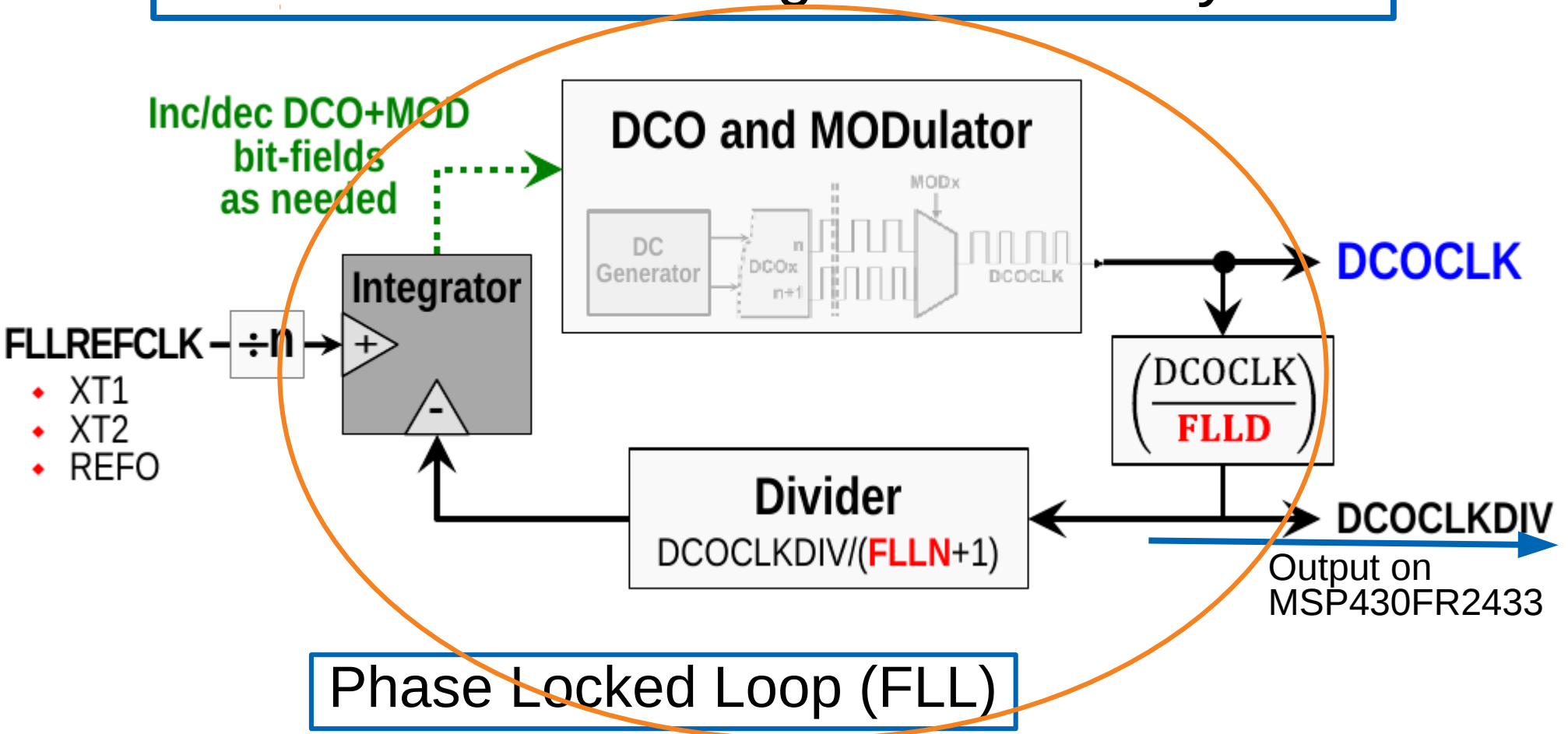
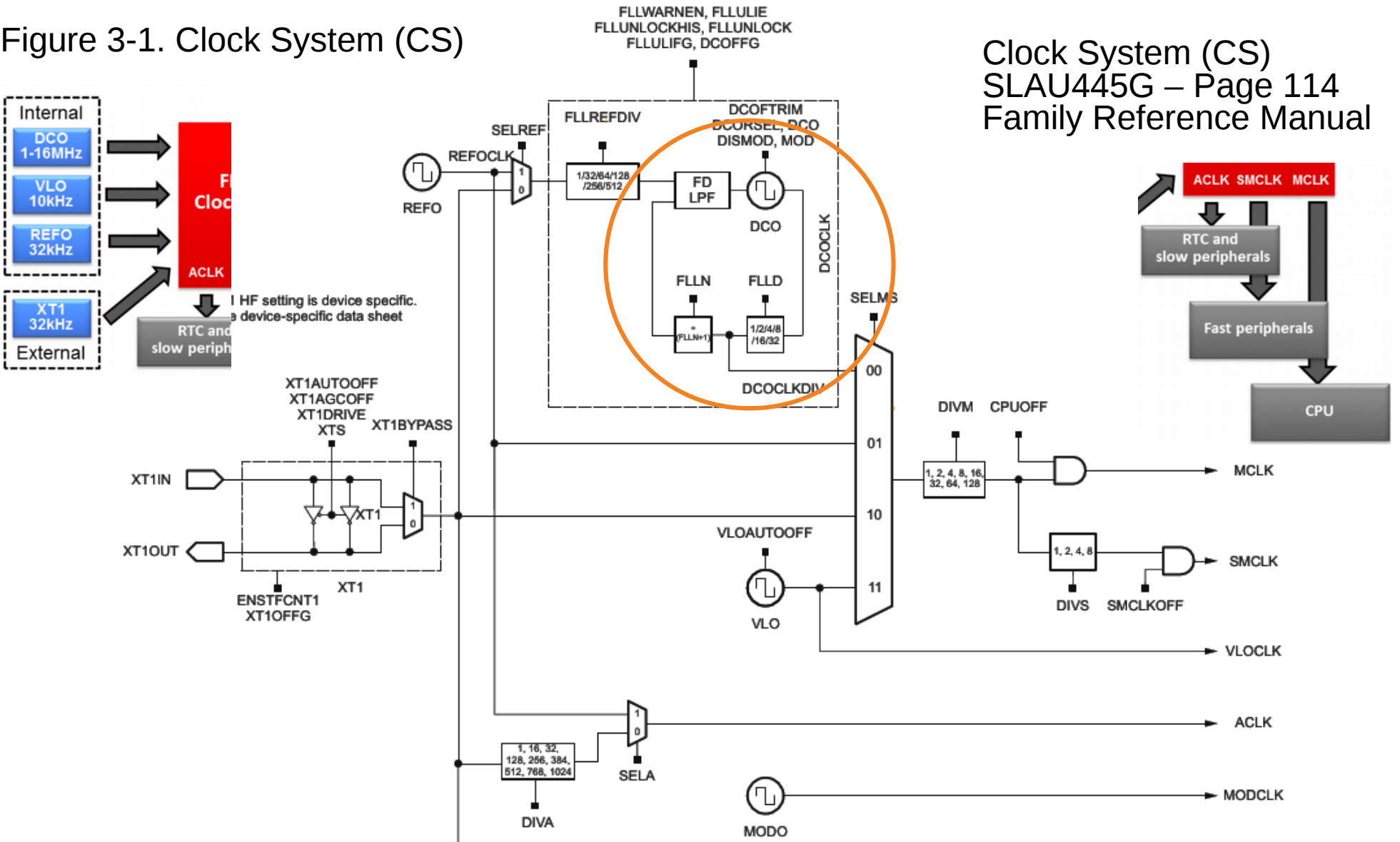
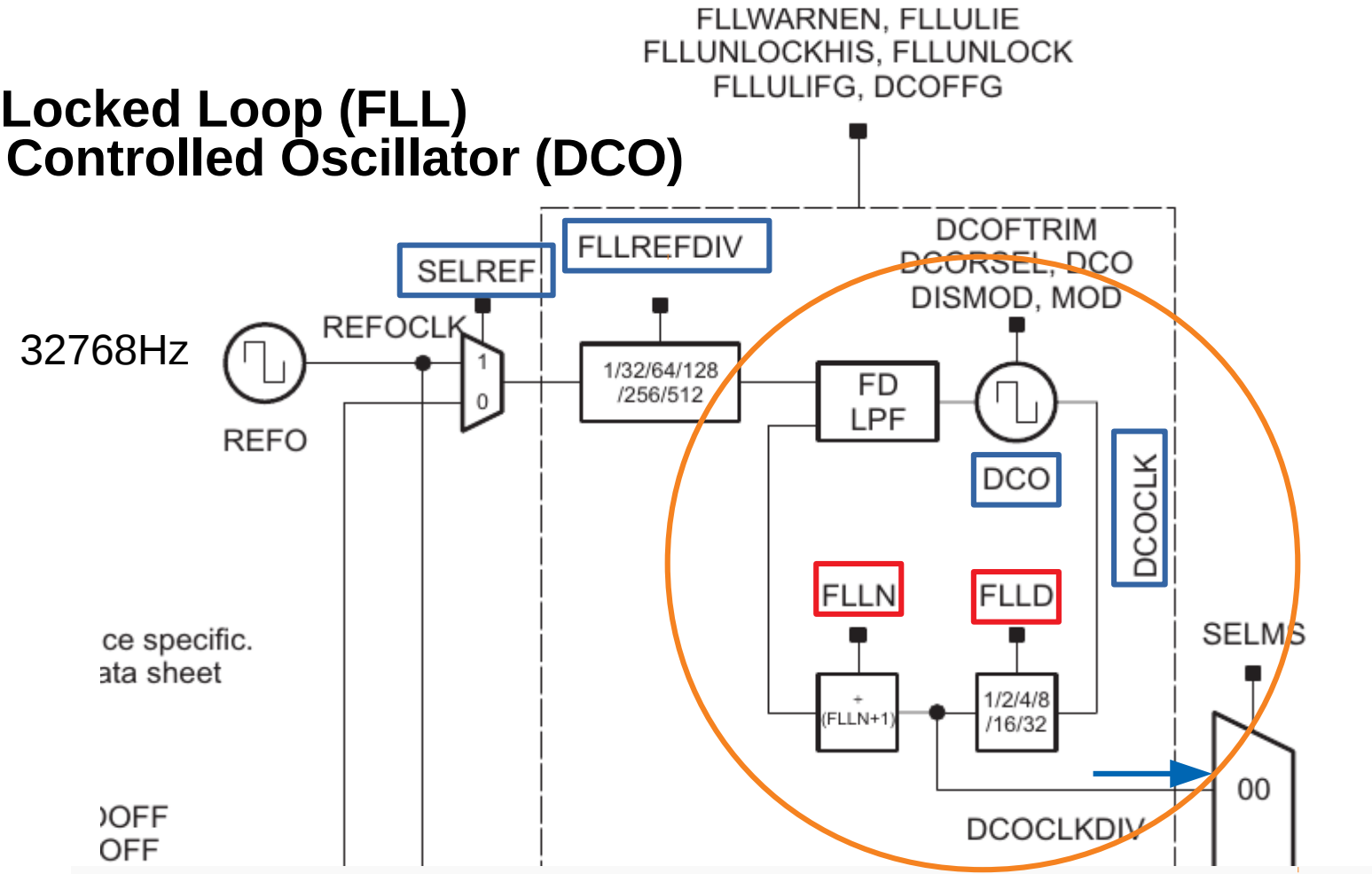


Figure 3-1. Clock System (CS)

Clock System (CS)
SLAU445G – Page 114
Family Reference Manual



Phase Locked Loop (FLL)
Digital Controlled Oscillator (DCO)



$$DCOCLK = (FLLREFCLK/n) * FLLD * (FLLN + 1)$$

where: n = FLLREFDIV

CS Output Section

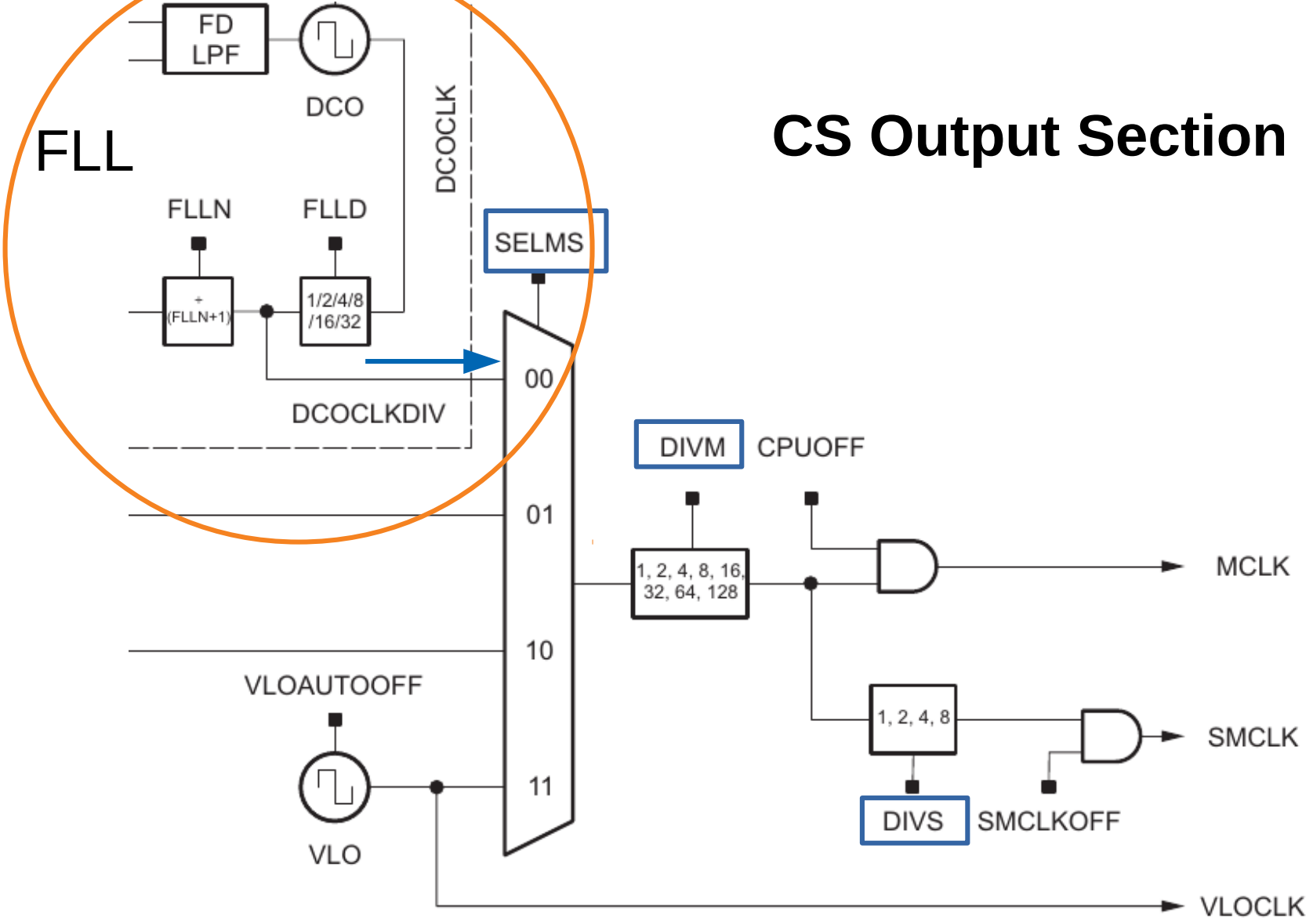
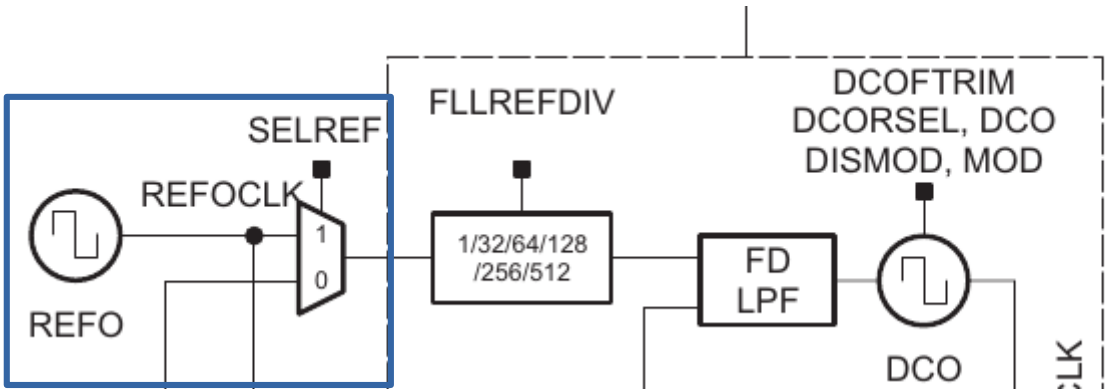


Table 3-2. CS Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	CSCTL0	Clock System Control Register 0	Read/write	Word	0000h	Section 3.3.1
02h	CSCTL1	Clock System Control Register 1	Read/write	Word	0033h	Section 3.3.2
04h	CSCTL2	Clock System Control Register 2	Read/write	Word	101Fh	Section 3.3.3
06h	CSCTL3	Clock System Control Register 3	Read/write	Word	0000h	Section 3.3.4
08h	CSCTL4	Clock System Control Register 4	Read/write	Word	0100h	Section 3.3.5
0Ah	CSCTL5	Clock System Control Register 5	Read/write	Word	1000h	Section 3.3.6
0Ch	CSCTL6	Clock System Control Register 6	Read/write	Word	08C1h	Section 3.3.7
0Eh	CSCTL7	Clock System Control Register 7	Read/write	Word	0740h	Section 3.3.8
10h	CSCTL8	Clock System Control Register 8	Read/write	Word	0007h	Section 3.3.9

```
35)     __bis_SR_register(SCG0);                    // disable FLL
36)    CSCTL3 |= SELREF__REFOCLK;                   // Set REFOCLK as FLL reference source
37)    CSCTL0 = 0;                                  // clear DCO and MOD FLL registers
38)    CSCTL1 &= ~(DCORSEL_7);                      // Clear DCO frequency select bits first
39)    CSCTL1 |= DCORSEL_3;                         // Set DCOCLK = 8MHz
40)    CSCTL2 = FLLD_1 + 121;                       // FLLD = 1, FFLN=121, DCODIV = 4MHz
41)    __delay_cycles(3);
42)    __bic_SR_register(SCG0);                     // enable FLL
43)    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked
44)    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK; // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
45)    CSCTL5 |= DIVM1;                             //  SMCLK = MCLK/2 = DCOCLKDIV/4 = 1MHz
46)    //CSCTL5 |= DIVM1 | DIVS0;                   // slow down /* SMCLK Divider Bit: 0    DIVS0*/
```

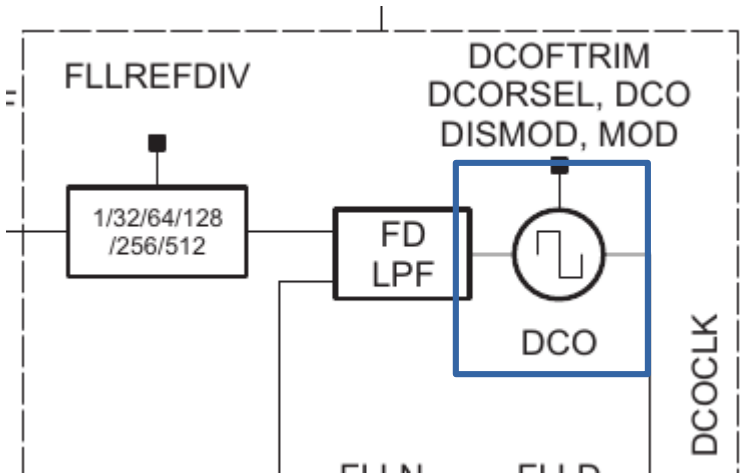
32768Hz




```
36)     CSCTL3 |= SELREF__REFOCLK;
```

CSCTL3	15	14	13	12	11	10	09	08	07	06	SREF	03	02	01	00

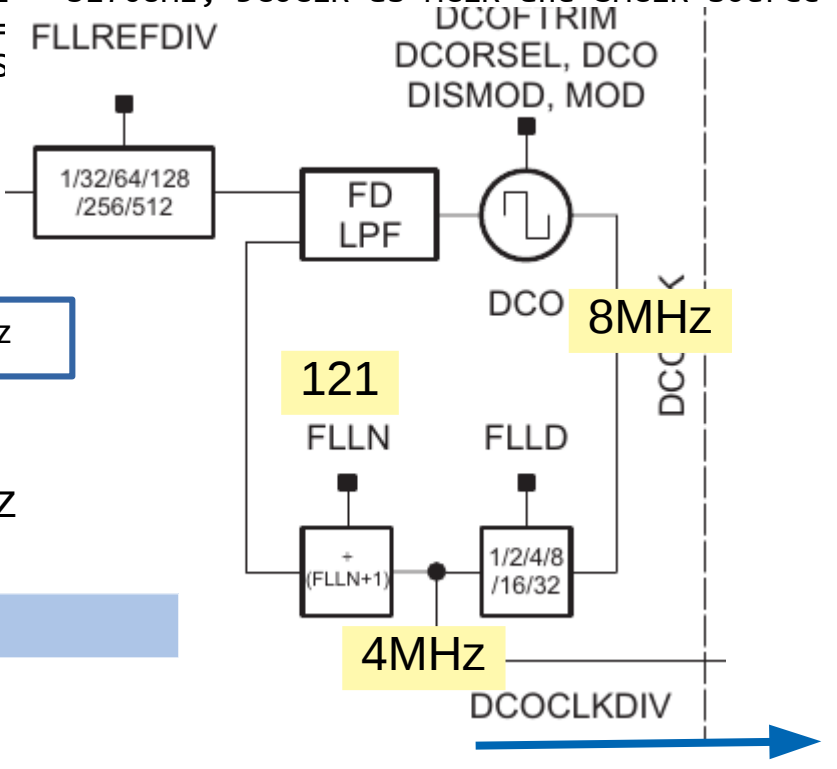
```
35)     __bis_SR_register(SCG0);                  // disable FLL
36)    CSCTL3 |= SELREF__REFOCLK;                 // Set REFOCLK as FLL reference source
37)    CSCTL0 = 0;                                // clear DCO and MOD FLL registers
38)    CSCTL1 &= ~(DCORSEL_7);                    // Clear DCO frequency select bits first
39)    CSCTL1 |= DCORSEL_3;                       // Set DCOCLK = 8MHz
40)    CSCTL2 = FLLD_1 + 121;                     // FLLD = 1, FFLN=121, DCODIV = 4MHz
41)    __delay_cycles(3);
42)    __bic_SR_register(SCG0);                   // enable FLL
43)    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked
44)    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK; // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
45)    CSCTL5 |= DIVM1;                           //  SMCLK = MCLK/2 = DCOCLKDIV/4 = 1MHz
46)    //CSCTL5 |= DIVM1 | DIVS0;                 // slow down /* SMCLK Divider Bit: 0    DIVS0*/
```



39)CSCTL1 |= DCORSEL_3; Set DCOCLK = 8MHz

CSCTL1	15	14	13	12	11	10	09	08	07	06	05	04	DCORSEL	00

```
35)     __bis_SR_register(SCG0);                  // disable FLL
36)    CSCTL3 |= SELREF__REFOCLK;                 // Set REFOCLK as FLL reference source
37)    CSCTL0 = 0;                                // clear DCO and MOD FLL registers
38)    CSCTL1 &= ~(DCORSEL_7);                    // Clear DCO frequency select bits first
39)    CSCTL1 |= DCORSEL_3;                       // Set DCOCLK = 8MHz
40)    CSCTL2 = FLLD_1 + 121;                     // FLLD = 1, FFLN=121, DCODIV = 4MHz
41)    __delay_cycles(3);
42)    __bic_SR_register(SCG0);                   // enable FLL
43)    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked
44)    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK; // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
45)    CSCTL5 |= DIVM1;                           // SMCLK = MCLK =
46)    //CSCTL5 |= DIVM1 | DIVS0;                 // slow down /* S
```



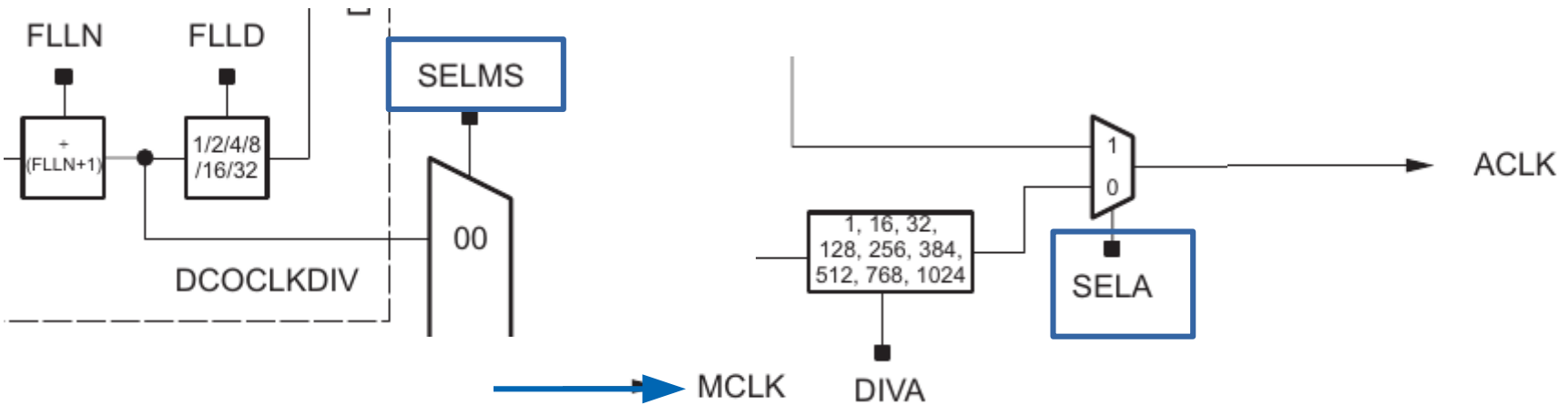

```
40)CSCTL2 = FLLD_1 + 121; // FLLD = Bit 1, FFLN=121, DCODIV = 4MHz
```

FFLN = 121+1 = 4MHz/32768Hz

CSCTL2	15	FFLD	11	10	FFLN

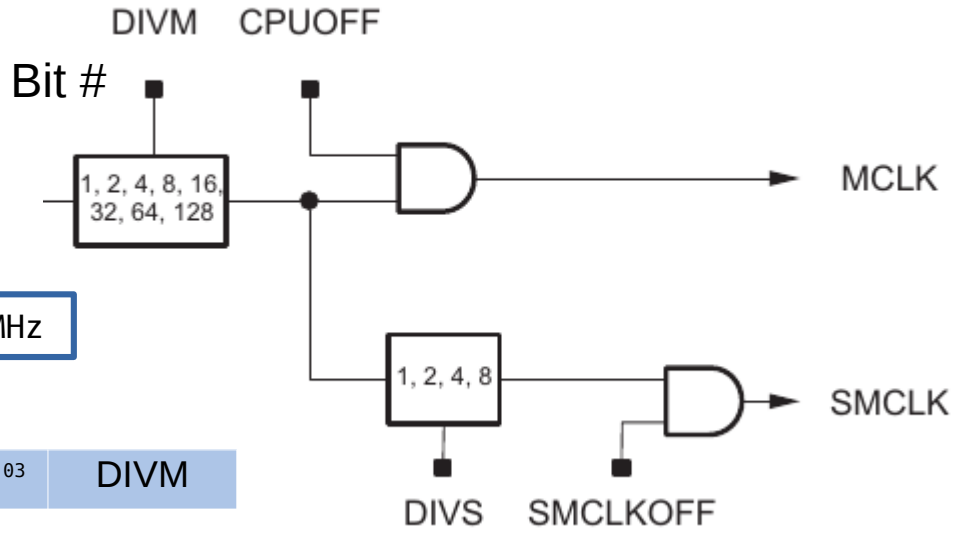
```
35)     __bis_SR_register(SCG0);                          // disable FLL
36)    CSCTL3 |= SELREF__REFOCLK;                         // Set REFOCLK as FLL reference source
37)    CSCTL0 = 0;                                        // clear DCO and MOD FLL registers
38)    CSCTL1 &= ~(DCORSEL_7);                            // Clear DCO frequency select bits first
39)    CSCTL1 |= DCORSEL_3;                               // Set DCOCLK = 8MHz
40)    CSCTL2 = FLLD_1 + 121;                             // FLLD = 1, FFLN=121, DCODIV = 4MHz
41)    __delay_cycles(3);
42)    __bic_SR_register(SCG0);                           // enable FLL
43)    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));        // Poll until FLL is locked
44)    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK;         // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
45)    CSCTL5 |= DIVM1;                                   // SMCLK = MCLK = DCODIV/4 = 1MHz
46)    //CSCTL5 |= DIVM1 | DIVS0;                         // slow down /* SMCLK Divider Bit: 0    DIVS0*/
```

```
44)CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK; // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
```



CSCTL4	15	14	13	12	11	10	09	SA	07	06	05	04	03	SELMS

```
35)     __bis_SR_register(SCG0);                        // disable FLL
36)    CSCTL3 |= SELREF__REFOCLK;                       // Set REFOCLK as FLL reference source
37)    CSCTL0 = 0;                                      // clear DCO and MOD FLL registers
38)    CSCTL1 &= ~(DCORSEL_7);                          // Clear DCO frequency select bits first
39)    CSCTL1 |= DCORSEL_3;                             // Set DCOCLK = 8MHz
40)    CSCTL2 = FLLD_1 + 121;                           // FLLD = 1, FFLN=121, DCODIV = 4MHz
41)    __delay_cycles(3);
42)    __bic_SR_register(SCG0);                         // enable FLL
43)    while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));      // Poll until FLL is locked
44)    CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK;       // set ACLK = XT1 = 32768Hz, DCOCLK as MCLK and SMCLK source
45)    CSCTL5 |= DIVM1;                                 //  SMCLK = MCLK = DCODIV/4 = 1MHz
46)    //CSCTL5 |= DIVM1 | DIVS0;                       // slow down /* SMCLK Divider Bit: 0    DIVS0*/
```




```
45)CSCTL5 |= DIVM1;  //  SMCLK = MCLK = DCODIV/4 = 1MHz
```

CSCTL5	15	14	13	12	11	10	09	08	07	06	DIVS	03	DIVM

Rest of ‘Setup’ part of the Program

```
50      P1DIR |= BIT0 | BIT1 | BIT3 | BIT7;      // set P1.3 MCLK P1.7 SMCLK and P1.0 Red and P1.1 Green LED pin as output
51      P1SEL1 |= BIT3 | BIT7;                  // set MCLK and  SMCLK pin as second function
52      P2DIR |= BIT2;                          // set ACLK pin as output
53      P2SEL1 |= BIT2;                         // set ACLK pin as second function
54
55
56      PM5CTL0 &= ~LOCKLPM5;                   // Disable the GPIO power-on default high-impedance mode
57                                              // to activate previously configured port settings
58         // Configure Timer_A
59      TA0CTL = TASSEL_1 | MC_2 | TACLR | TAIE;      // ACLK, count mode, clear TAR, enable interrupt
60      TA1CTL = TASSEL_2 | MC_2 | TACLR | TAIE;      // SMCLK, count mode, clear TAR, enable interrupt
61
62      PM5CTL0 &= ~LOCKLPM5;                   // Disable the GPIO power-on default high-impedance mode
63
64      //__bis_SR_register( GIE);              //Enable interrupts
65      __bis_SR_register(LPM0_bits | GIE);             //Enable interrupts
66      while(1);
```

```
72  // Timer0_A3 Interrupt Vector (TAIV) handler  (ACLK/(2^16))*2
73  #pragma vector=TIMER0_A1_VECTOR
74  __interrupt void TIMER0_A1_ISR(void)
75  {
76      switch(TA0IV)
77      {
78          case TA0IV_NONE:
79              break;                          // No interrupt
80          case TA0IV_TACCR1:
81              break;                          // CCR1 not used
82          case TA0IV_TACCR2:
83              break;                          // CCR2 not used
84          case TA0IV_TAIFG:
85              P1OUT ^= BIT1;                  // overflow  GREEN= 4 sec  0.25 Hz
86              break;
87          default:
88              break;
89      }
90  }
```

```
93  // Timer1_A3 Interrupt Vector (TAIV) handler   (SMCLK/(2^16))*2
94  #pragma vector=TIMER1_A1_VECTOR
95  __interrupt void TIMER1_A1_ISR(void)
96  {
97      switch(TA1IV)
98      {
99          case TA1IV_NONE:
100             break;                              // No interrupt
101         case TA1IV_TACCR1:
102             break;                              // CCR1 not used
103         case TA1IV_TACCR2:
104             break;                              // CCR2 not used
105         case TA1IV_TAIFG:
106             P1OUT ^= BIT0;                      // overflow  RED= 131Msec 7.63Hz
107             break;
108         default:
109             break;
110     }
111 }
112
```

Now, go run the code!