Brownout Reset (BOR)

At power-up, the brownout circuitry holds device in reset until V_{cc} is above hysteresis point

Startup from BOR:

- RST/NMI pin is configured as reset
- I/O pins are configured as inputs
- Clocks are configured
- Peripherals and CPU registers are initialized (see user guide)
- Status register (SR) is reset
- Watchdog timer powers up active in watchdog mode
- Program counter (PC) is loaded with reset vector location (0xFFFE)

  If reset vector is blank (0FFFFh), the device enters LPM4

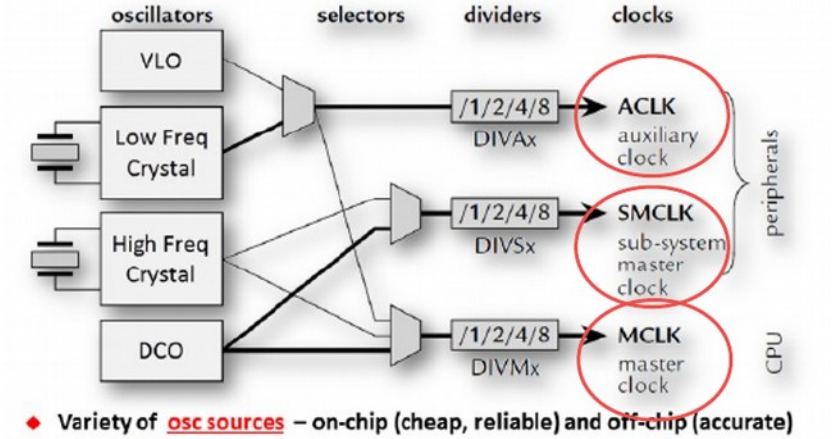
Three Levels of Reset

- BOR is most comprehensive, followed by:
  - **POR** = Power-On Reset
  - **PUC** = Power-Up Clear

Table 3-7. CSCTL4 Register Description

Bit	Field	Type	Reset	Description
15-9	Reserved	R	0h	Reserved. Always reads as 0.
8	SELA	RW	1h	Selects the ACLK source. 0b = XT1CLK with divider (must be no more than 40 kHz) 1b = REFO (internal 32-kHz clock source)
7-3	Reserved	R	0h	Reserved. Always reads as 0.
2-0	SELMS	RW	0h	Selects the MCLK and SMCLK source. 000b = DCOCLKDIV 001b = REFOCLK 010b = XT1CLK 011b = VLOCLK 1xxb = Reserved for future use

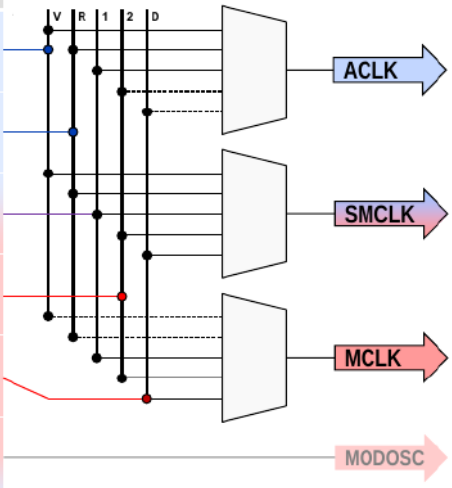
MSP430 – Lot's of Options



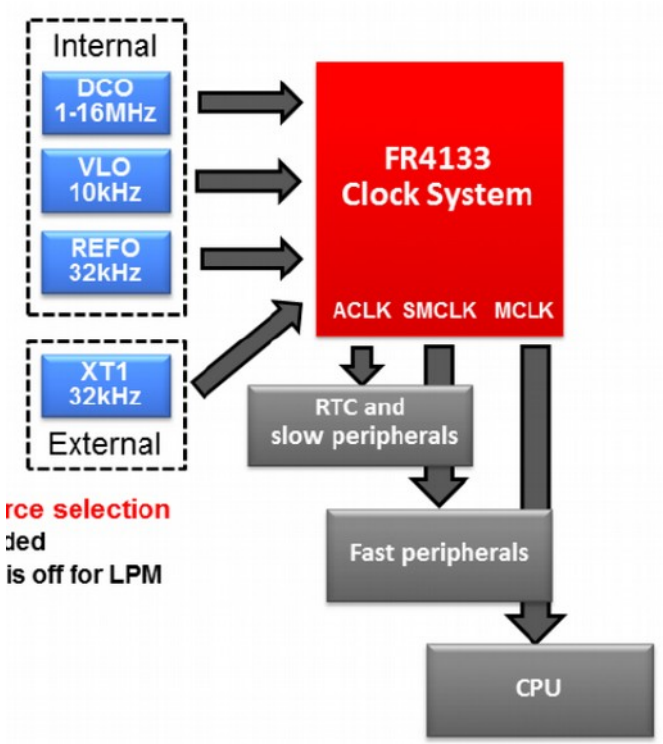
- Variety of osc sources – on-chip (cheap, reliable) and off-chip (accurate)
- Rich selection of oscillator sources routed to internal clocks
- Many clock dividers enhance the available clock frequencies
- All MSP430 devices provide at least 3 internal clocks – provides flexibility in tuning system's power vs performance

MSP430FR2433 Clock Sources

		Frequency
RC	VLO	~10 KHz
	REFO	32768 Hz
	XT1	• LF: < 50 KHz • HF: 4-Max MHz
	XT2 not available	
	DCO	100 KHz to CPU Max
Async	MODOSC	• 5 MHz • 5 MHz / 128



*Note: This is a general description, please refer to datasheet/UsersGuide for complete details regarding your device



- DCO setup is hybrid of FR5xx DCO and F5xx DCO + FLL
- Specific frequency ranges
  - Ranges centered on 1, 2, 4, 8, 12, 16MHz
  - Selected with DCORSEL bits
- Uses FLL with reference frequency to tune within frequency range
- 512 DCO steps within these smaller ranges = smaller steps
  - Allows very accurate DCO + FLL even with just REFO – no crystal (+/-2% over temperature)
  - Even more accurate with crystal (+/-0.5% over temperature)
  - Much less jitter because steps are smaller
- FLL allows compensation for temperature drift

```
          MSP430FR2433
        ---------------
    /|\|               |
     | |               |
     --|RST            |
       |          P1.0 |---> LED
       |          P1.3 |---> MCLK  = 1MHz
       |          P1.7 |---> SMCLK = 1MHz
       |          P2.2 |---> ACLK  = 32768Hz
```

```
// details on default values are in SLASE59B datasheet - page 54

//Use the GPIO pins to output the clock signals - Observe with Oscilloscope
P1DIR |= BIT0 | BIT3 | BIT7;            // set LED MCLK and SMCLK pins as outputs
P1SEL1 |= BIT3 | BIT7;                 // set MCLK and  SMCLK pins as second function
                                       //  (P1SEL0 is 0 from POR) so connect them to MCLK, SMCLK
P2DIR |= BIT2;                         // set ACLK pin as output
P2SEL1 |= BIT2;                        // set ACLK pin as second function
```

```
// set MCLK, SMCLK to 8 MHz using DCO as source
__bis_SR_register(SCG0);               // disable FLL
CSCTL3 |= SELREF__REFOCLK;             // Set REFO as FLL reference source
CSCTL0 = 0;                            // clear DCO and MOD registers
CSCTL1 &= ~(DCORSEL_7);                // Clear DCO frequency select bits first
CSCTL1 |= DCORSEL_3;                   // Set DCO = 8MHz
CSCTL2 = FLLD_0 + 243;                 // DCODIV = 8MHz
__delay_cycles(3);
__bic_SR_register(SCG0);               // enable FLL
while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1)); // Poll until FLL is locked

CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK; // set default REFO(~32768Hz) as ACLK source, ACLK = 32768Hz
                                       // default DCODIV as MCLK and SMCLK source
```

```
// Configure one FRAM waitstate as required by the device datasheet for MCLK
// operation beyond 8MHz _before_ configuring the clock system.
FRCTL0 = FRCTLPW | NWAITS_1;

__bis_SR_register(SCG0);                       // disable FLL
CSCTL3 |= SELREF__REFOCLK;                     // Set REFO as FLL reference source
CSCTL0 = 0;                                    // clear DCO and MOD registers
CSCTL1 &= ~(DCORSEL_7);                        // Clear DCO frequency select bits first
CSCTL1 |= DCORSEL_5;                           // Set DCO = 16MHz
CSCTL2 = FLLD_0 + 487;                         // DCOCLKDIV = 16MHz
__delay_cycles(3);
__bic_SR_register(SCG0);                       // enable FLL
while(CSCTL7 & (FLLUNLOCK0 | FLLUNLOCK1));     // FLL locked

CSCTL4 = SELMS__DCOCLKDIV | SELA__REFOCLK;     // set default REFO(~32768Hz) as ACLK source, ACLK = 32768Hz
                                               // default DCOCLKDIV as MCLK and SMCLK source
```

ASSIGNMENT M5 BLOCK DIAGRAM

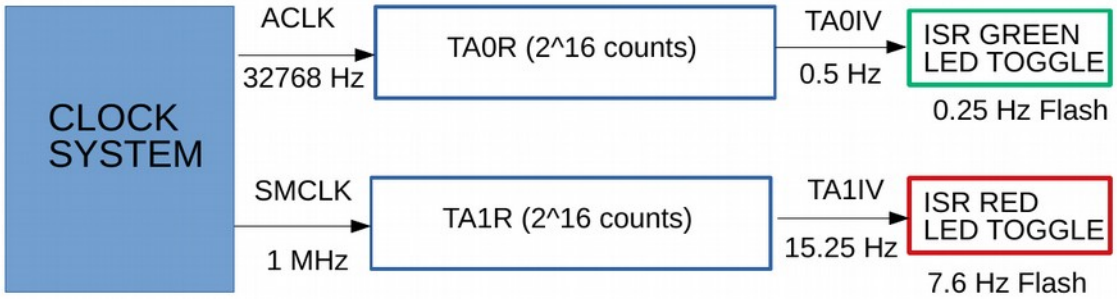
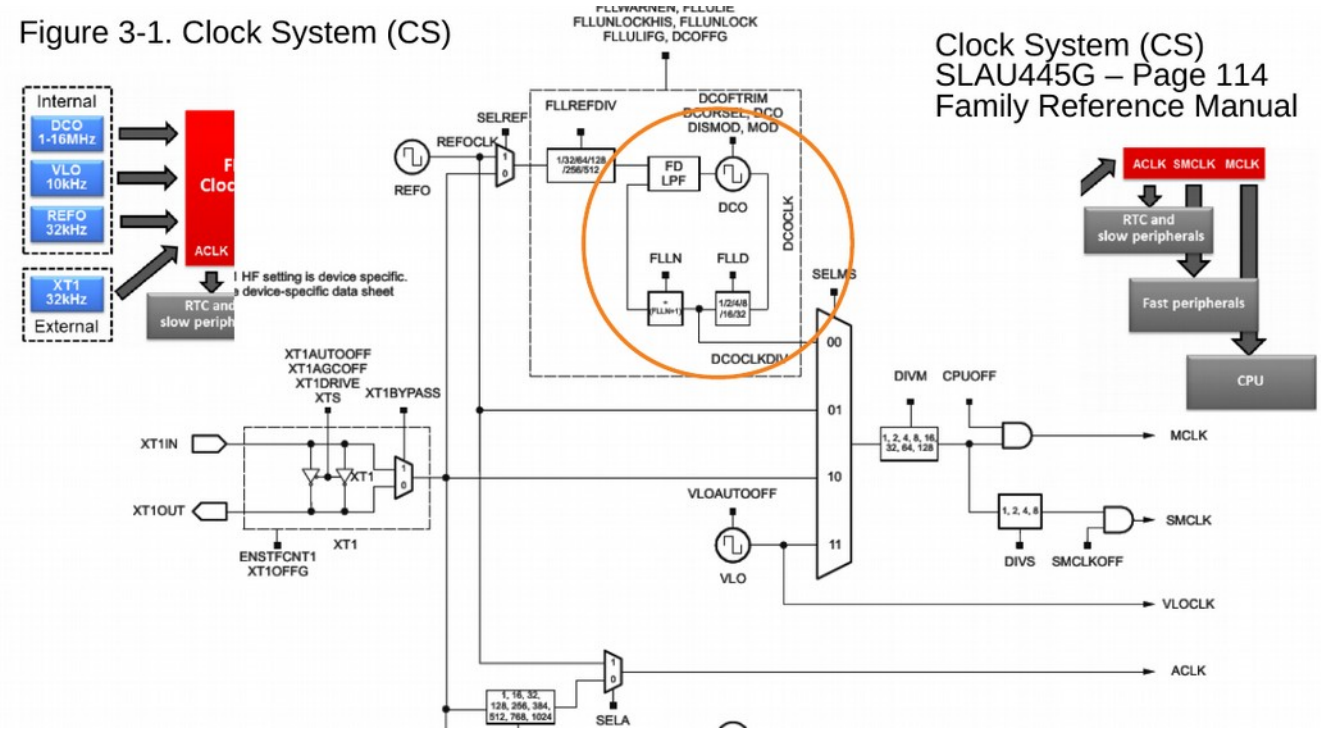
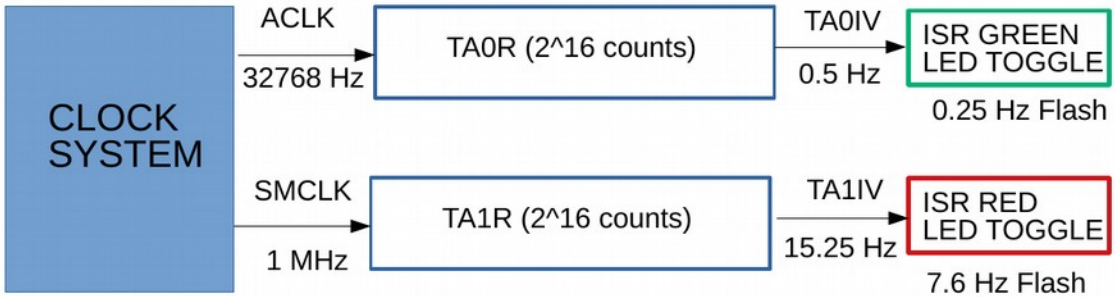


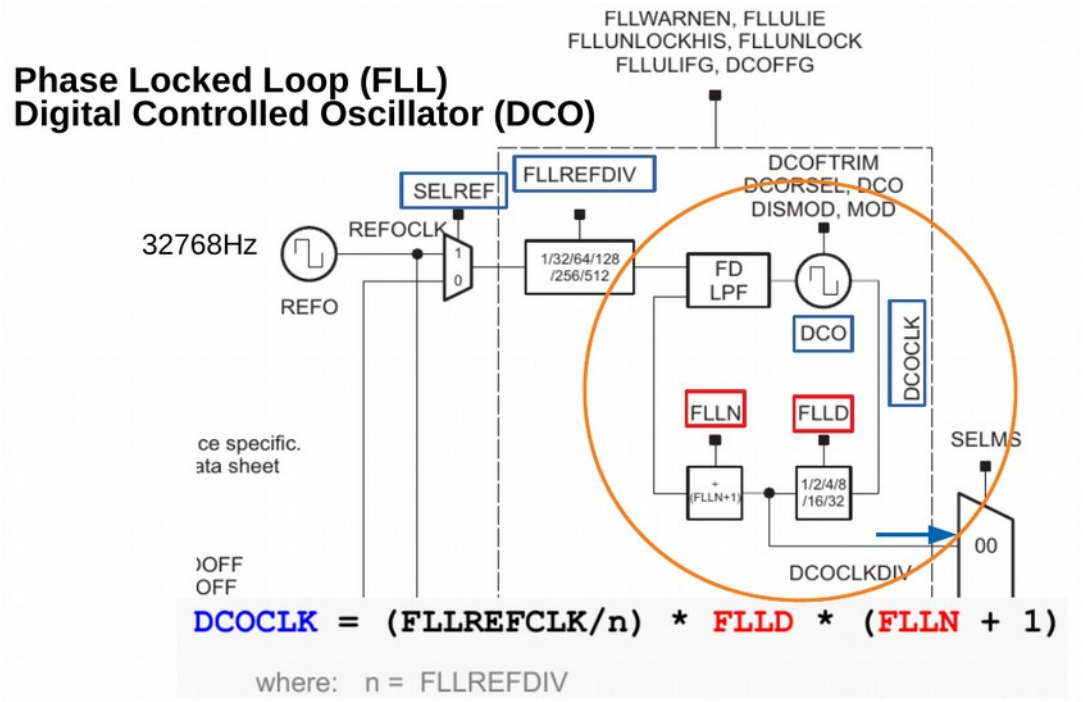
Figure 3-1. Clock System (CS)

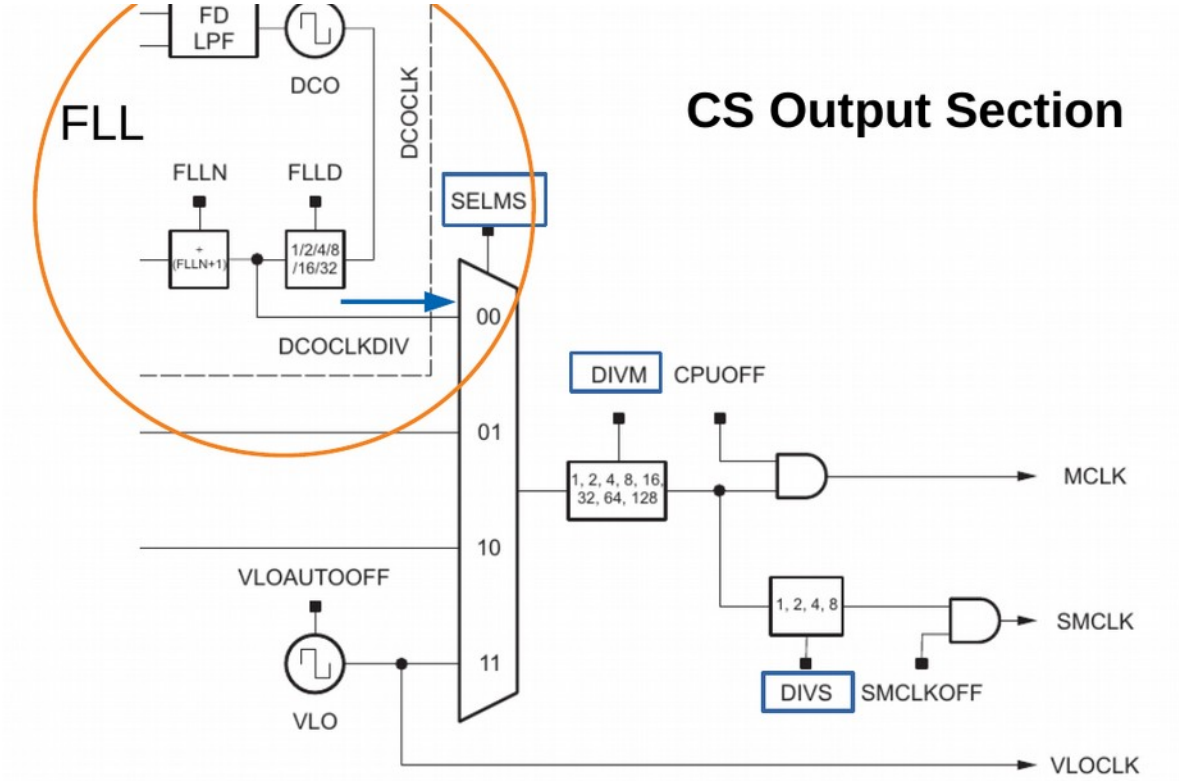
Clock System (CS)
SLAU445G – Page 114
Family Reference Manual



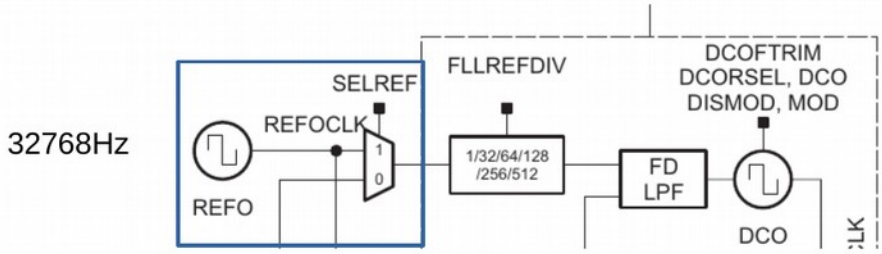
ASSIGNMENT M5 BLOCK DIAGRAM

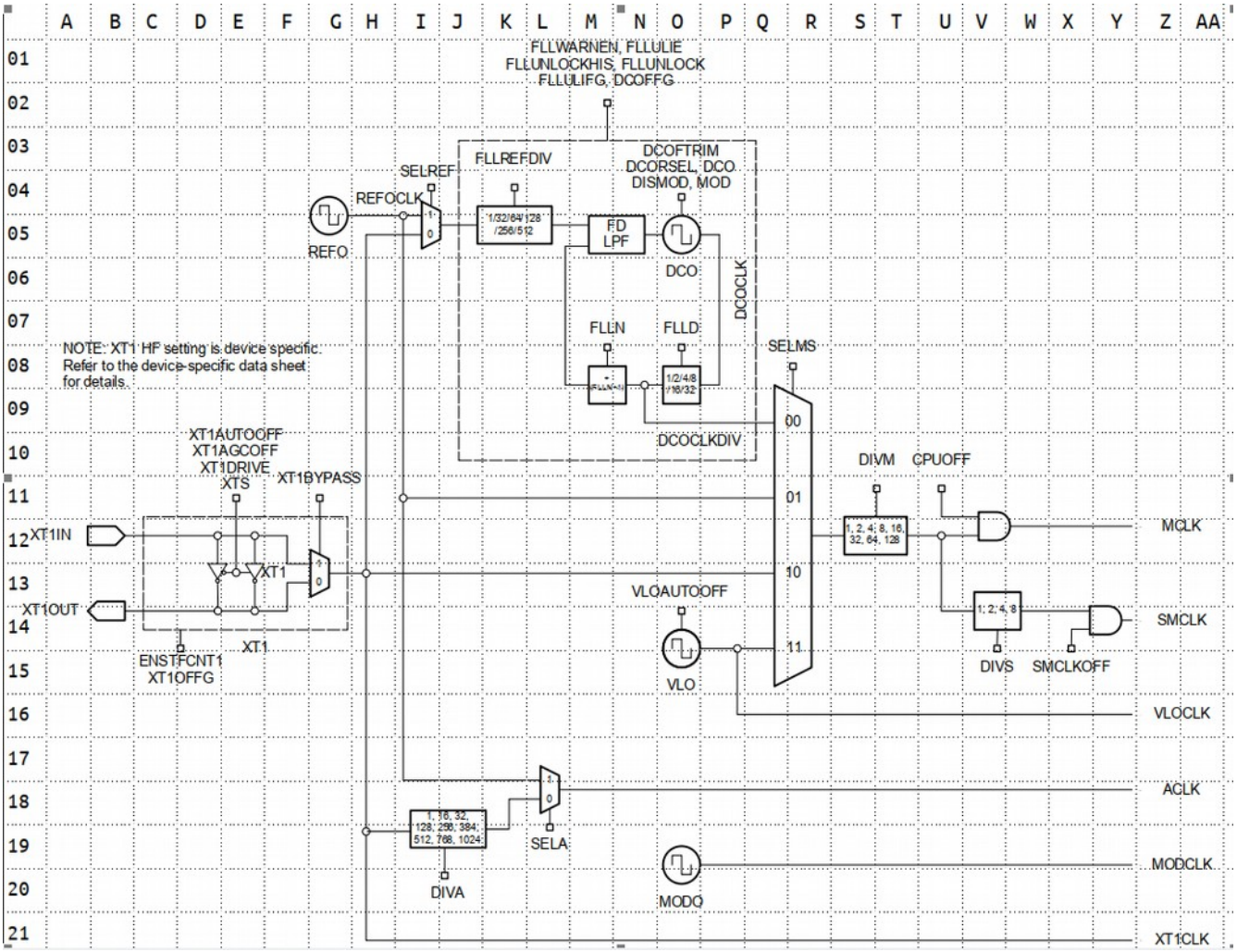






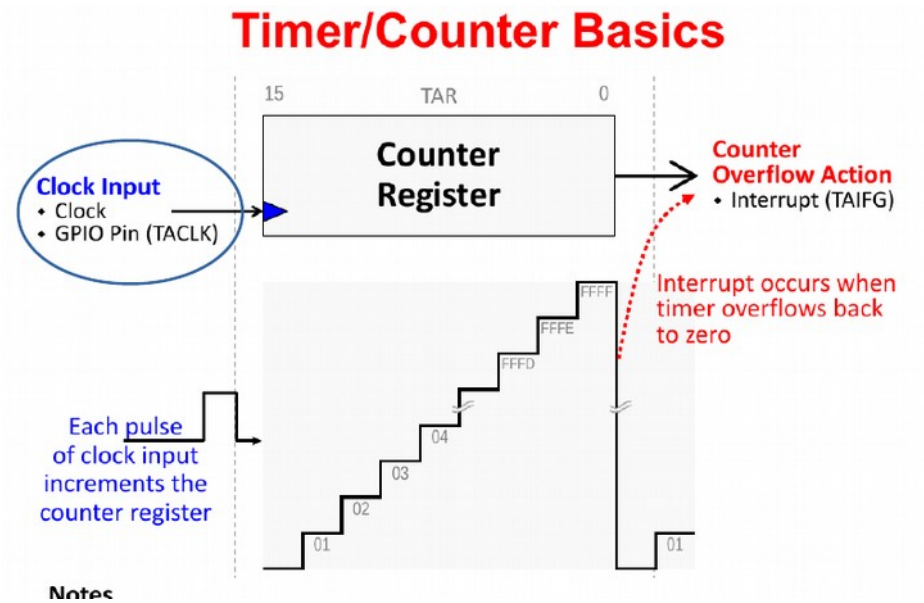

```
35)     __bis_SR_register(SCG0);                // disable FLL
36)    CSCTL3 |= SELREF__REFOCLK;               // Set REFOCLK as FLL reference source
37)    CSCTL0 = 0;                              // clear DCO and MOD FLL registers
38)    CSCTL1 &= ~(DCORSEL_7);                  // Clear DCO frequency 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*/
```

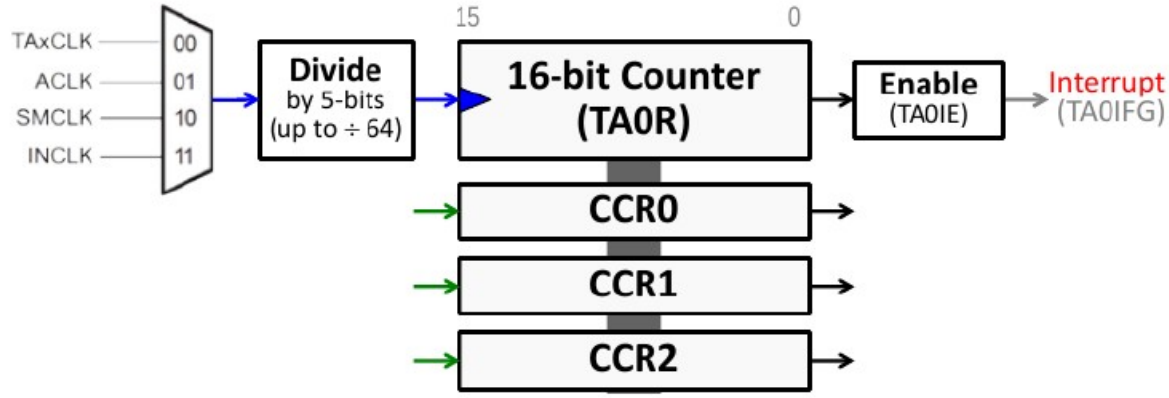



FLLWARNEN, FLLULIE
FLLUNLOCKHIS, FLLUNLOCK
FLLULIFG, DCOFFG
SELREF
FLLREFDIV
DCOFTRIM
DCORSEL, DCO
DISMOD, MOD
REFOCLK
REFO
1/32/64/128
/256/512
FD
LPF
DCO
DCOCLK
FLLN
FLLD
NOTE: XT1 HF setting is device specific.
Refer to the device-specific data sheet
for details.
SELMS
00
DCOCLKDIV
XT1AUTOOFF
XT1AGCOFF
XT1DRIVE
XTS
XT1BYPASS
DIVM
CPUOFF
01
XT1IN
XT1
1, 2, 4, 8, 16,
32, 64, 128
MCLK
10
XT1OUT
VLOAUTOOFF
1, 2, 4, 8
SMCLK
XT1
11
ENSTFCNT1
XT1OFFG
VLO
DIVS
SMCLKOFF
VLOCLK
ACLK
1, 16, 32,
128, 256, 384,
512, 768, 1024
SELA
DIVA
MODO
MODCLK
XT1CLK

- Use interrupts to control program flow
- Maximize the time in LPM3
- Replace software with peripherals
- Configure unused pins properly
- Power manage external devices
- Efficient code makes a difference

  Every unnecessary instruction executed is a portion of the battery that's wasted and gone forever





Nomenclature Timer0_A3

Remember:

- Timer0 means it's the first instance of Timer_A on the device.
- _A3 means that it's a Timer_A device and has 3 capture/compare registers (CCR's)
- The clock input, in this example, can be driven by a TACLK signal/pin, ACLK, SMCLK or another internal clock called INCLK.
- The clock input can be further divided down by a 5-bit scalar.
- The TA0IE interrupt enable can be used to allow (or prevent) an interrupt (TA0IFG) from reaching the CPU whenever the counter (TA0R) rolls over.

12.2.3.2 Continuous Mode

In the Continuous mode, the timer repeatedly counts up to 0FFFFh and restarts from zero as shown in Figure 12-4. The capture/compare register TAxCCR0 works the same way as the other capture/compare registers.

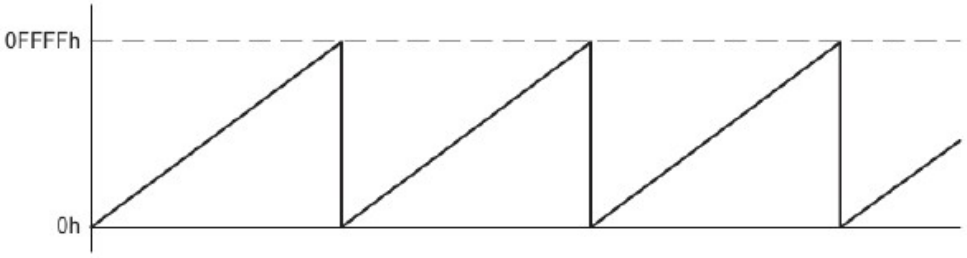
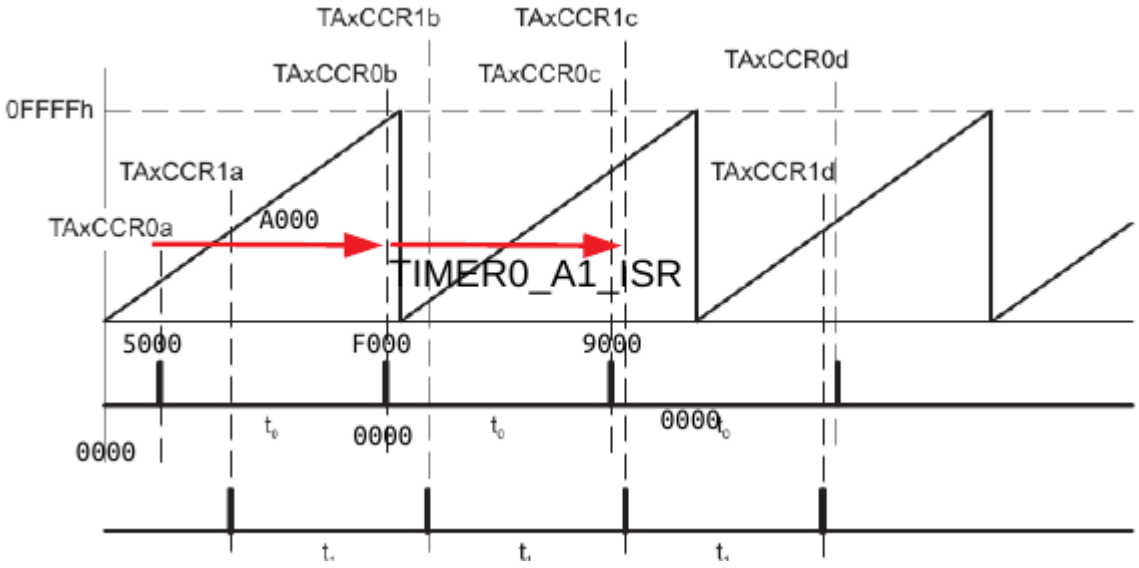


Figure 12-4. Continuous Mode

```
81      // Timer1_A3 setup
82      TA1CCTL0 = CCIE;                        // TACCR0 interrupt enabled
83      TA1CCR0 = 50000;
84      TA1CTL = TASSEL_1 | MC_2;               // ACLK, continuous mode
85
86      __bis_SR_register(LPM3_bits | GIE);     // Enter LPM3 w/ interrupt
87  }
```



The Up mode is used if the timer period must be different from 0FFFFh counts. The timer repeatedly counts up to the value of compare register TAxCCR0, which defines the period (see Figure 12-2). The number of timer counts in the period is TAxCCR0 + 1. When the timer value equals TAxCCR0, the timer restarts counting from zero. If Up mode is selected when the timer value is greater than TAxCCR0, the timer immediately restarts counting from zero.

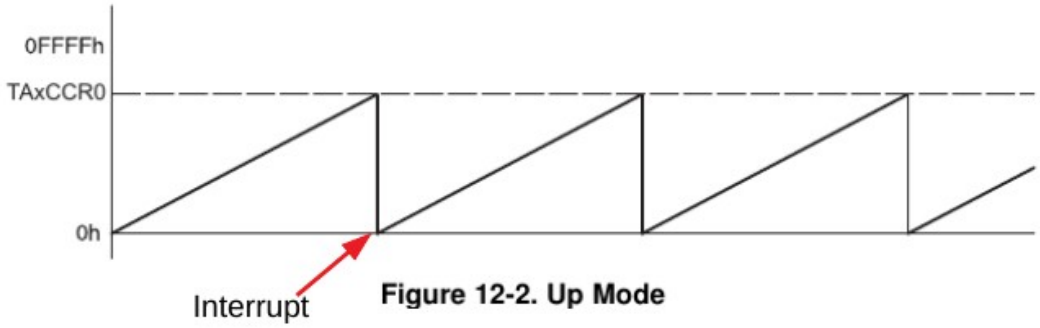
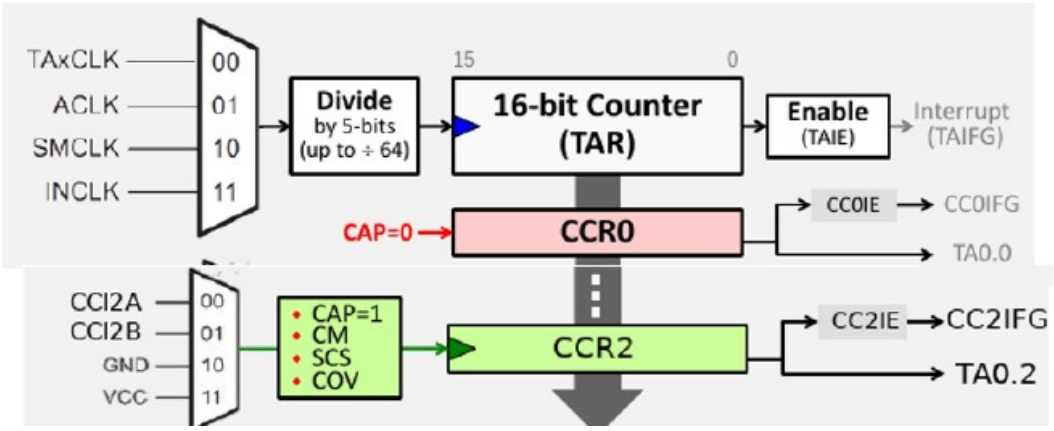


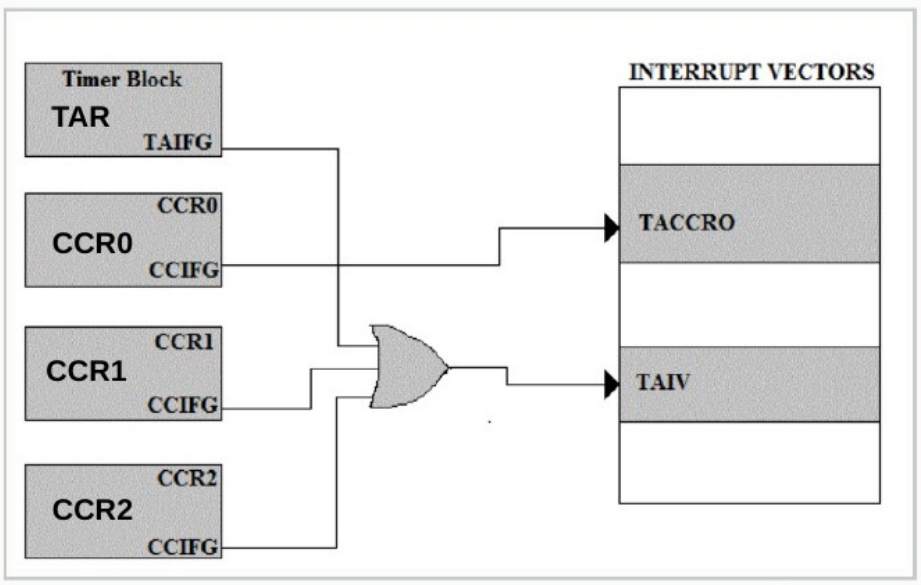
Figure 12-2. Up Mode

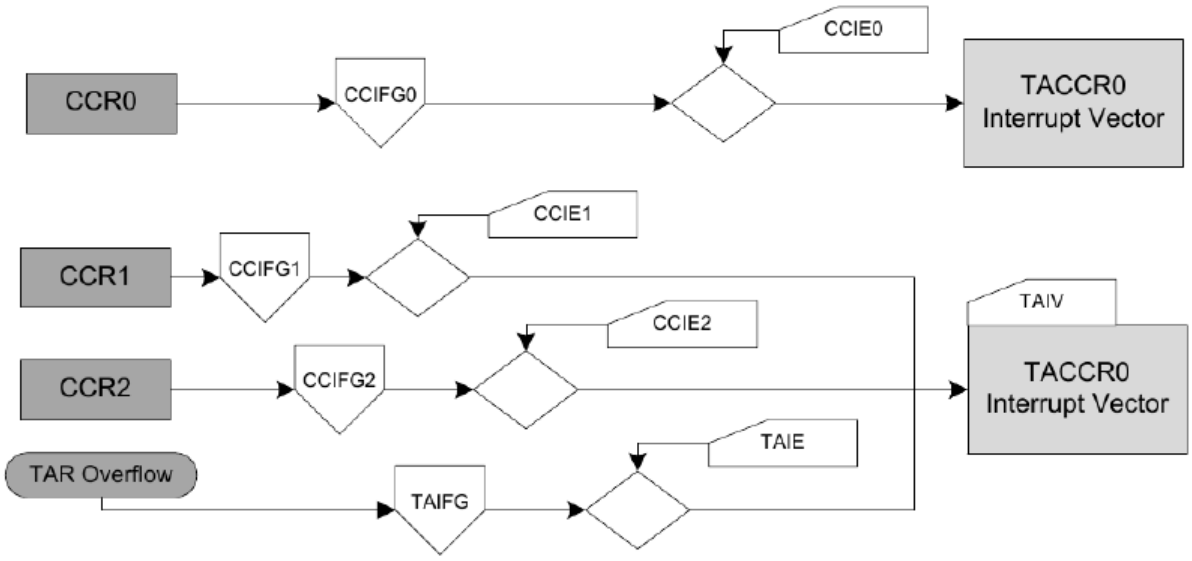


CAP=0 (Capture off)

- Compare mode on
- If CCR = TAR (named EQU0):
  - Interrupt occurs (if enabled)
  - OUT is set/reset/toggled
- OUT can be:
  - Connected to pin (TA0.0)
  - Routed to peripherals
  - OUT bit can be polled
- Many OUT signal options (up to 7 options)

- **Capture or Compare (CAP)**
  CAP=1 for capture
- **Which Edge (CM)**
  Rising, Falling, or Both
- **Sync'd to Clock (SCS)**
  Is capture sync or async?
- **Capture Overflow (COV)**
  Did you miss a capture?






```
37
38      TA0CCTL0 |= CCIE;                           // TACCR0 interrupt enabled
39      TA0CCR0 = 50000;
40      TA0CTL |= TASSEL_2 | MC_2;          // SMCLK, continuous mode
41
42      __bis_SR_register(LPM0_bits | GIE);         // Enter LPM3 w/ interrupts
43      __no_operation();                           // For debug
44  }
45
46  // Timer A0 interrupt service routine for CCR0
47  #pragma vector = TIMER0_A0_VECTOR
48  __interrupt void Timer_A (void)
49
50  {
51      P1OUT ^= BIT0;
52      TA0CCR0 += 50000;                           // Add Offset to TACCR0
53  }
```

Continuous Mode – no overflow interrupt, CCR0 compare

```
39
40      TA0CCTL0 |= CCIE;                           // TACCR0 interrupt enabled
41      TA0CCR0 = 50000;
42      TA0CTL = TASSEL_2 | MC_1;               // SMCLK, UP mode
43
44      __bis_SR_register(LPM0_bits | GIE);         // Enter LPM0 w/ interrupt
45      __no_operation();                           // For debug
46  }
47
48  // Timer A0 interrupt service routine
49  #pragma vector = TIMER0_A0_VECTOR
50  __interrupt void Timer_A (void)
51  {
52      P1OUT ^= BIT0;
53  }
54
```

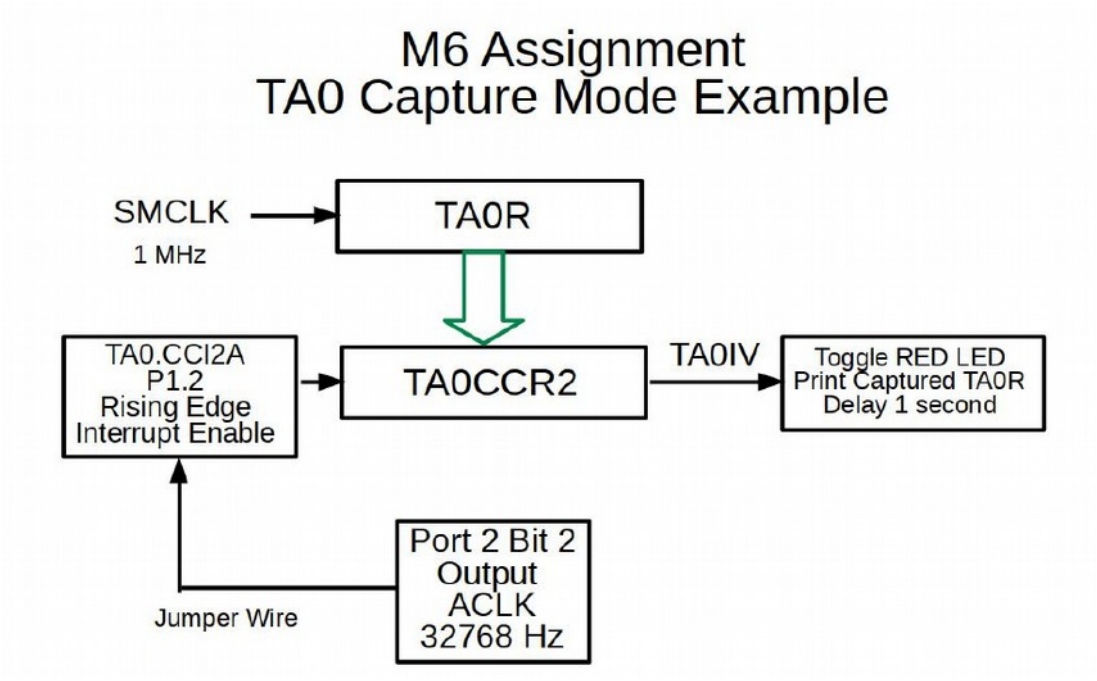
Count Up Mode – CCR0

```
41      // Configure Timer_A
42      TA0CTL = TASSEL_1 | MC_2 | TACLR | TAIE;      // ACLK, continuous mode, clear TAR, enable interrupt
43      // ACLK = 32768 - P1.0 (RED LED) = 32768/(2^16) = 0.5Hz toggle = 2 seconds on, then 2 seconds off
44
45      __bis_SR_register(LPM3_bits | GIE);           // Enter LPM3, enable interrupts
46      __no_operation();                             // For debugger
47  }
48
49  // Timer0_A3 Interrupt Vector (TAIV) handler
50  #pragma vector = TIMER0_A1_VECTOR
51  __interrupt void TIMER0_A1_ISR(void)
52  {
53      switch(TA0IV)
54      {
55          case TA0IV_NONE:
56              break;                                // No interrupt
57          case TA0IV_TACCR1:
58              break;                                // CCR1 not used
59          case TA0IV_TACCR2:
60              break;                                // CCR2 not used
61          case TA0IV_TAIFG:
62              P1OUT ^= BIT0;                        // overflow
63              break;
64          default:
65              break;
66      }
67  }
```

Continuous mode, overflow interrupt

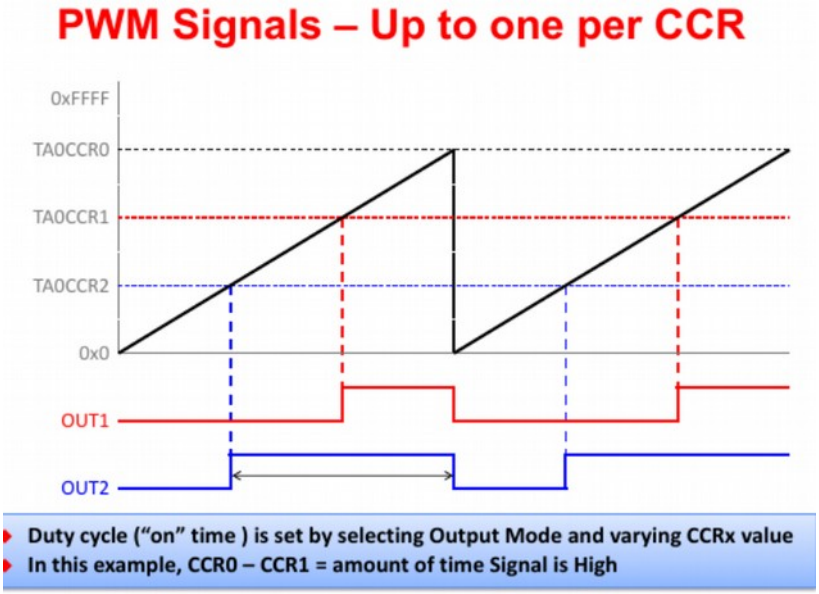
```
// msp430fr243x_ta0_08A.c

// Control Timer_A CCRx Registers
    TA0CCTL0 = CCIE; // CCR0 ISR toggle P1.3
    TA0CCTL1 = OUTMOD_4 + CCIE; // Output TA0.1 (CCR1) to P1.1
    TA0CCTL2 = OUTMOD_4 + CCIE; // Output TA0.2 (CCR2) to P1.2
// Control Timer_A  Register    // ACLK = 32768
    TA0CTL = TASSEL_1 | MC_2 | TACLR | TAIE;      // ACLK, continuous mode, clear
TAR, enable interrupt

    __bis_SR_register(LPM3_bits | GIE);           // Enter LPM3, enable interrupts
    __no_operation();                             // For debugger
}

// Timer0_A0 CCR0 Iinterrupt Vector
#pragma vector = TIMER0_A0_VECTOR
__interrupt void TIMER0_A0_ISR(void)
{
    P1OUT ^= BIT3;  // toggle the output pin
    TA0CCR0 += 8;  //  CCR0= 32768/(2*8) = 2048Hz
}

// Timer0_A3 Interrupt Vector (TAIV) handler
#pragma vector = TIMER0_A1_VECTOR
__interrupt void TIMER0_A1_ISR(void)
{
    switch(TA0IV)
    {
        case TA0IV_NONE:
            break;                               // No interrupt
        case TA0IV_TACCR1:
            TA0CCR1 += 16;                       // CCR1 =32768/(2*16) = 1024Hz
            break;
        case TA0IV_TACCR2:
            TA0CCR2 += 100;                      // CCR1 =32768/(2*100) = 163.84Hz
            break;
        case TA0IV_TAIFG:
            P1OUT ^= BIT0;                       // TAR overflow = 32768/(2*(2^16))
= 0.25Hz
            break;
        default:
            break;
    }
```

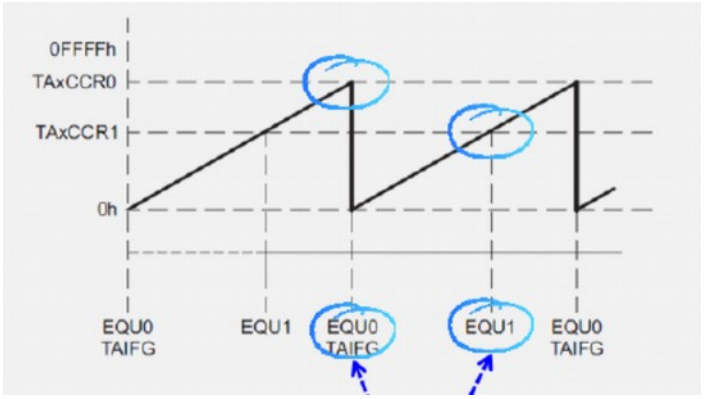
M6 Assignment
TA0 Capture Mode Example

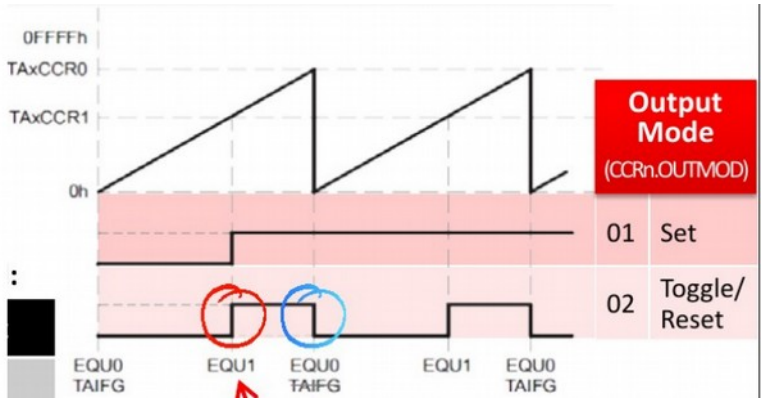


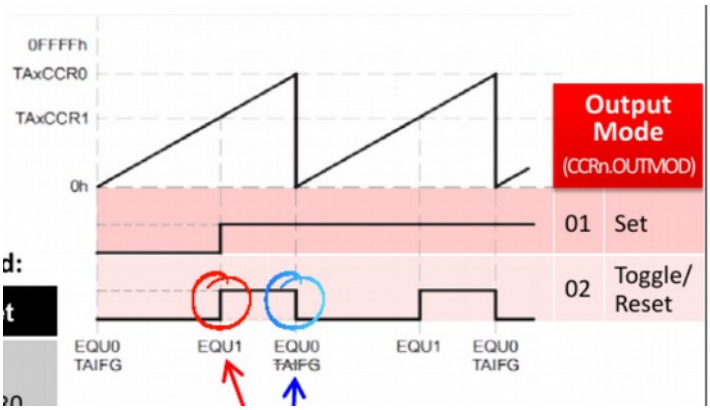

```
73      // Timer0_A3 Setup: Capture each ACLK rising edge
74      TA0CCTL2 |= CM_1 | CCIS_0 | CCIE | CAP | SCS;
75                                                  // Capture rising edge,
76                                                  // Use CCI2A=ACLK,
77                                                  // Synchronous capture,
78                                                  // Enable capture mode,
79                                                  // Enable capture interrupt
80
81      TA0CTL |= TASSEL_2 | MC_2 | TACLR;          // Use SMCLK as TA0 clock source, clear TA0R
82                                                  // Start timer in continuous mode
83
84      __bis_SR_register(LPM0_bits | GIE);
85
86  }
```

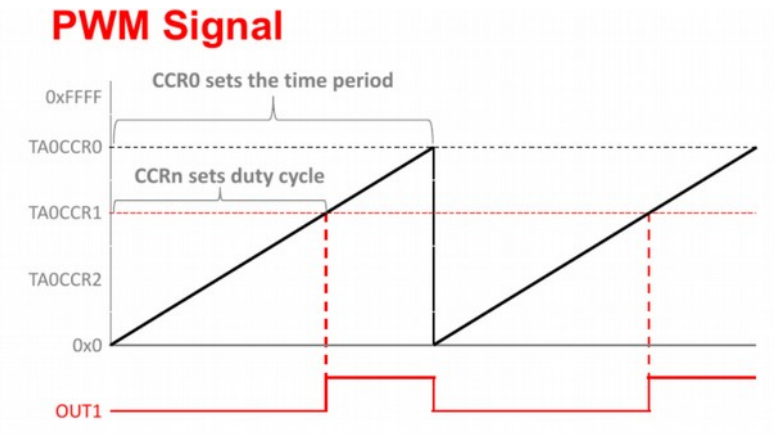
```
88 // Timer0_A3 CC1-2, TA Interrupt Handler
89 #pragma vector = TIMER0_A1_VECTOR
90 __interrupt void TIMER0_A1_ISR(void)
91 {
92   static int LastCapture;
93
94     switch(TA0IV)
95     {
96         case TA0IV_NONE:
97             break;                                  // No interrupt
98         case TA0IV_TACCR1:
99             break;                                  // CCR1 not used
100        case TA0IV_TACCR2:
101
102                     P1OUT ^= 0x01;                  // Toggle P1.0 (LED)
103                     printf("TA0CCR2 %d\n", TA0CCR2);
104                      TA0CTL |= TACLR;
105                     __delay_cycles(1000000);          //slow blink 16960 remainder
106            break;                                   // CCR2 not used
107        case TA0IV_TAIFG:
108            break;                                   // overflow
109        default:
110            break;
111    }
112 }
```


PWM Signals – Up to one per CCR
0xFFFF
TA0CCR0
TA0CCR1
TA0CCR2
0x0
OUT1
OUT2
Duty cycle ("on" time) is set by selecting Output Mode and varying CCRx value
In this example, CCR0 – CCR1 = amount of time Signal is High


0FFFFh
TAxCCR0
TAxCCR1
0h
EQU0
TAIFG
EQU1
EQU0
TAIFG
EQU1
EQU0
TAIFG


0FFFFh
TAxCCR0
TAxCCR1
0h
Output Mode
(CCRn.OUTMOD)
01
Set
02
Toggle/ Reset
EQU0
TAIFG
EQU1
EQU0
TAIFG
EQU1
EQU0
TAIFG

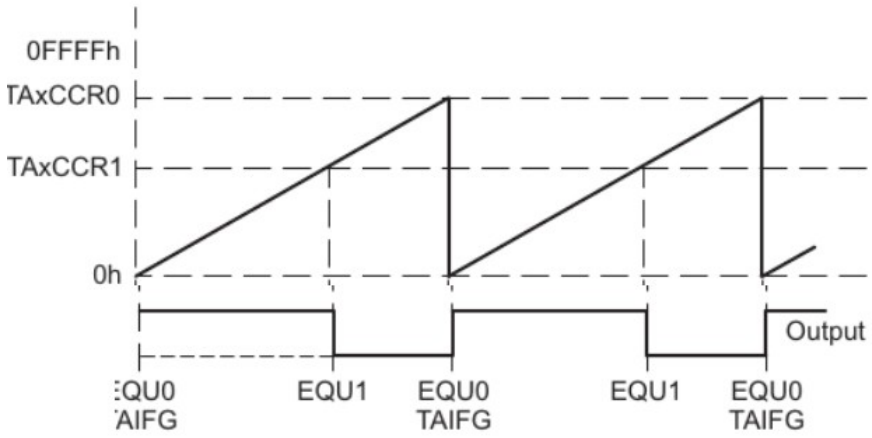





```
81      TA0CCR0 = 128;                          // PWM Period/2
82      TA0CCTL1 = OUTMOD_6;                    // TACCR1 toggle/set
83      TA0CCR1 = 32;                           // TACCR1 PWM duty cycle
84      TA0CCTL2 = OUTMOD_6;                    // TACCR2 toggle/set
85      TA0CCR2 = 96;                           // TACCR2 PWM duty cycle
86      TA0CTL = TASSEL_1 | MC_3;               // ACLK, up-down mode
87
88      __bis_SR_register(LPM3_bits);           // Enter LPM3
89
```

ACLK – LPM3

```
39      TA0CCR0 = 1000-1;                           // PWM Period
40      TA0CCTL1 = OUTMOD_7;                        // CCR1 reset/set
41      TA0CCR1 = 750;                              // CCR1 PWM duty cycle
42      TA0CCTL2 = OUTMOD_7;                        // CCR2 reset/set
43      TA0CCR2 = 250;                              // CCR2 PWM duty cycle
44      TA0CTL = TASSEL__SMCLK | MC__UP | TACLR;    // SMCLK, up mode, clear TAR
45
46      __bis_SR_register(LPM0_bits);               // Enter LPM0
47      __no_operation();                           // For debugger
48  }
```



P1.0/UCB0STE/TA0CLK/A0/V
P1.1/UCB0CLK/TA0.1/A1
P1.2/UCB0SIMO/UCB0SDA/TA
P1.3/UCB0SOMI/UCB0SCL/MC
P1.4/UCA0TXD/UCA0SIMO/TA
P1.5/UCA0RXD/UCA0SOMI/TA
P1.6/UCA0CLK/TA1CLK/TDI/
P1.7/UCA0STE/SMCLK/TDO/A