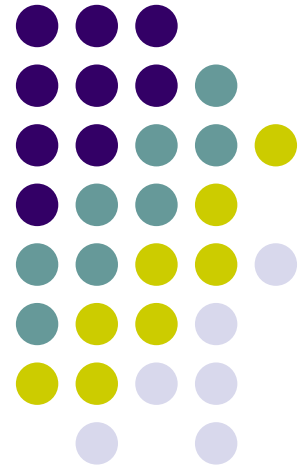


MSP430 Clock System and Timer

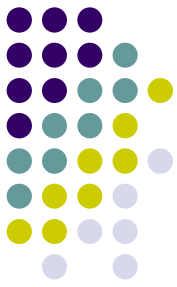
TA: Yin Wang

CSU610 SWARM, Spring 2007

CCIS, Northeastern University



Outline



- MSP430 basic clock module
- MSP430 Timer A
- Timer A examples

MSP430 Basic Clock Module

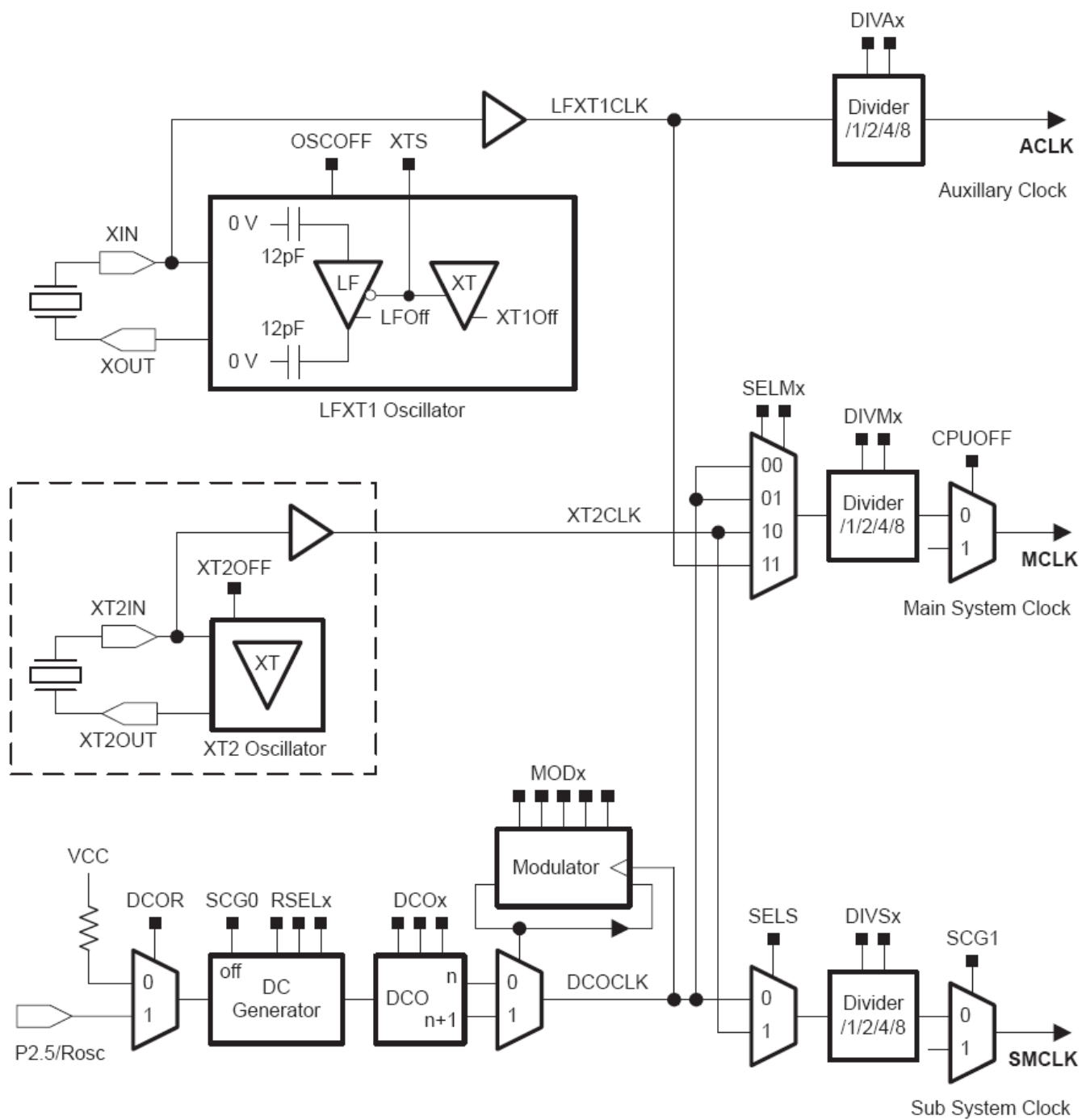


- Clock Sources:
 - **LFXT1CLK** : Low-frequency/high-frequency oscillator
 - **XT2CLK** : Optional high-frequency oscillator
 - **DCOCLK** : Internal digitally controlled oscillator (DCO)
- Tmote Sky Configuration:
 - **LFXT1CLK** : 32.768KHz crystal
 - **XT2CLK** : N/A
 - **DCOCLK** : Built-in DCO with configurable range from <100KHz to 4MHz

MSP430 Basic Clock Module



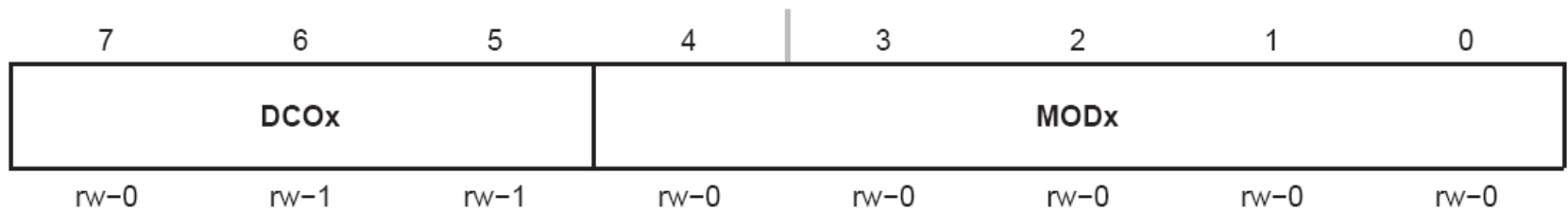
- Clock Signals:
 - **ACLK**: Auxiliary clock. The signal is sourced from LFXT1CLK with a divider of 1, 2, 4, or 8. (The calibration program for the serial link sets the divider to 4, but after the calibration it can be changed to any other values.) ACLK can be used as the clock signal for Timer A and Timer B.
 - **MCLK**: Master clock. The signal can be sourced from LFXT1CLK, XT2CLK (if available), or DCOCLK with a divider of 1, 2, 4, or 8. MCLK is used by the CPU and system.
 - **SMCLK**: Sub-main clock. The signal is sourced from either XT2CLK (if available), or DCOCLK with a divider of 1, 2, 4, or 8. SMCLK can be used as the clock signal for Timer A and Timer B.



Clock System Registers



DCOCTL, DCO Control Register



DCOx	Bits 7-5	DCO frequency select. These bits select which of the eight discrete DCO frequencies of the RSELx setting is selected.
MODx	Bits 4-0	Modulator selection. These bits define how often the $f_{\text{DCO}+1}$ frequency is used within a period of 32 DCOCLK cycles. During the remaining clock cycles (32-MOD) the f_{DCO} frequency is used. Not useable when DCOx=7.

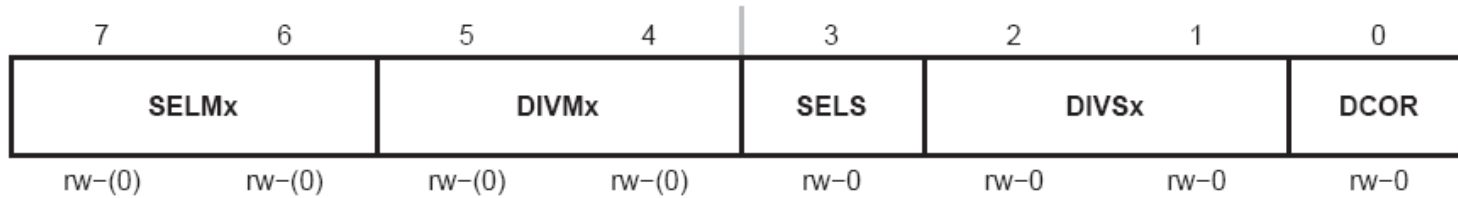


BCSCTL1, Basic Clock System Control Register 1



XT2OFF	Bit 7	XT2 off. This bit turns off the XT2 oscillator 0 XT2 is on 1 XT2 is off if it is not used for MCLK or SMCLK.
XTS	Bit 6	LFXT1 mode select. 0 Low frequency mode 1 High frequency mode
DIVAx	Bits 5-4	Divider for ACLK 00 /1 01 /2 10 /4 11 /8
XT5V	Bit 3	Unused. XT5V should always be reset.
RSELx	Bits 2-0	Resistor Select. The internal resistor is selected in eight different steps. The value of the resistor defines the nominal frequency. The lowest nominal frequency is selected by setting RSELx=0.

BCSCTL2, Basic Clock System Control Register 2

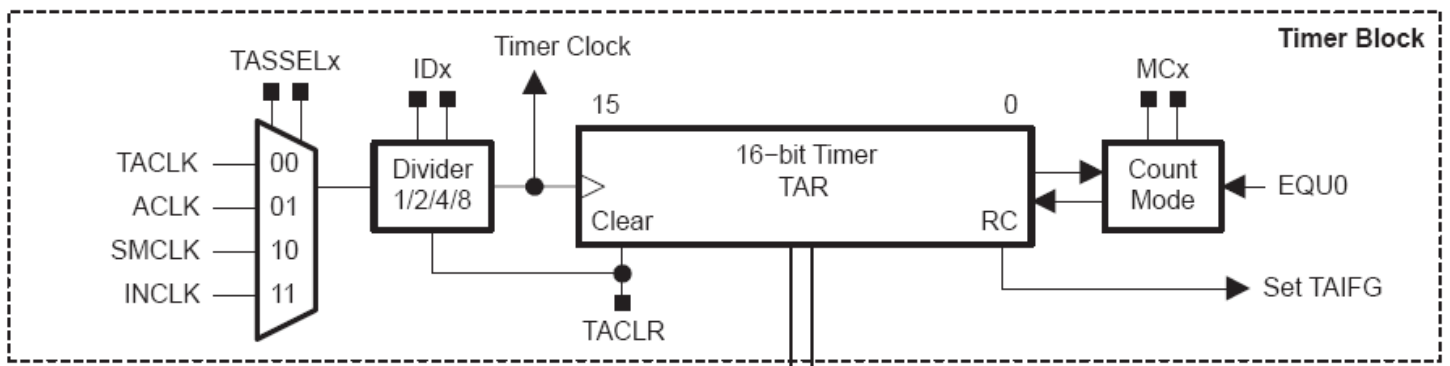


SELMx	Bits 7-6	Select MCLK. These bits select the MCLK source. 00 DCOCLK 01 DCOCLK 10 XT2CLK when XT2 oscillator present on-chip. LFXT1CLK when XT2 oscillator not present on-chip. 11 LFXT1CLK
DIVMx	BitS 5-4	Divider for MCLK 00 /1 01 /2 10 /4 11 /8
SELS	Bit 3	Select SMCLK. This bit selects the SMCLK source. 0 DCOCLK 1 XT2CLK when XT2 oscillator present on-chip. LFXT1CLK when XT2 oscillator not present on-chip.
DIVSx	BitS 2-1	Divider for SMCLK 00 /1 01 /2 10 /4 11 /8
DCOR	Bit 0	DCO resistor select 0 Internal resistor 1 External resistor

MSP430 Timer_A



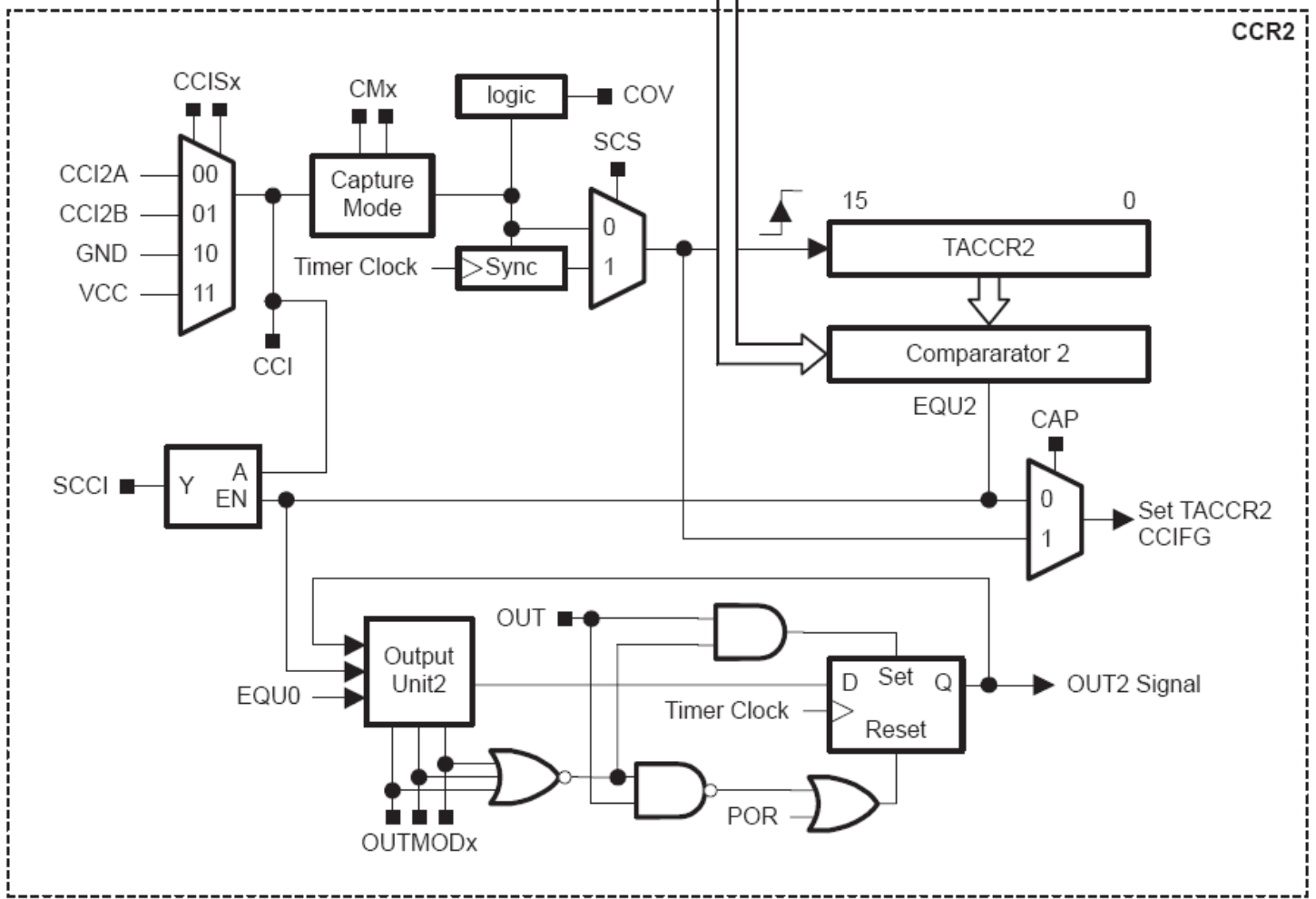
- A 16-bit counter
- 4 modes of operation – Stop, Up, Continuous, Up/Down
- 3 capture/compare registers (CCR_x)
- 2 interrupt vectors – TACCR0 and TAIV



CCR0

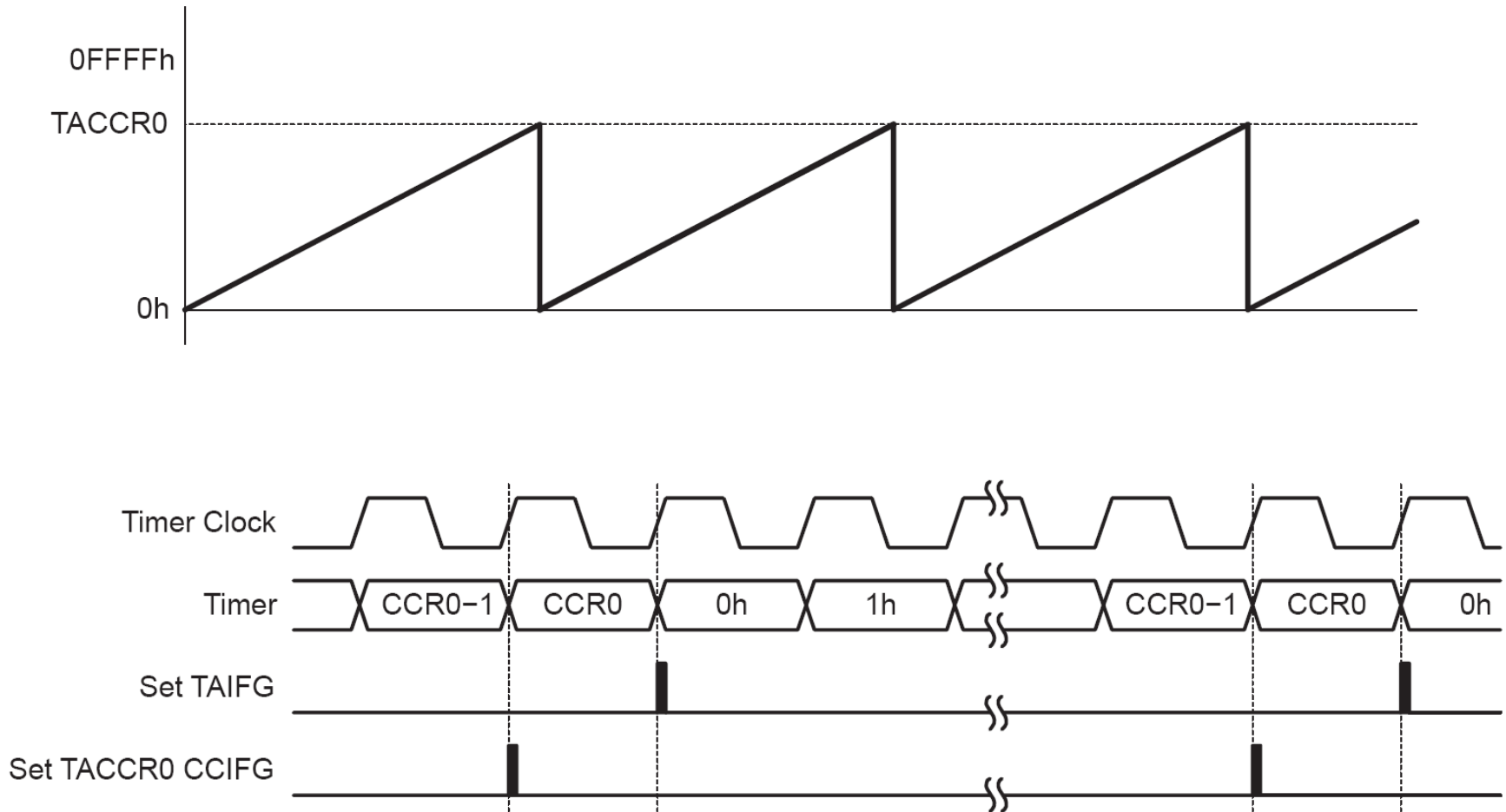
CCR1

CCR2

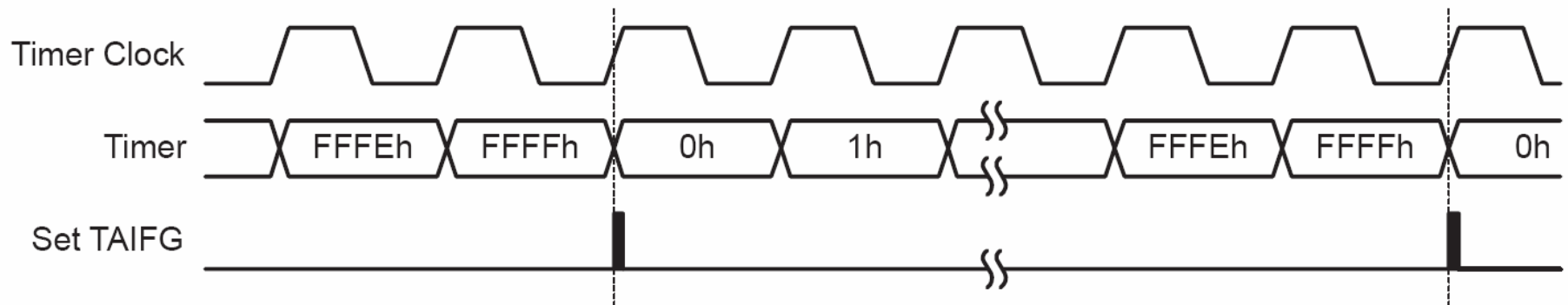
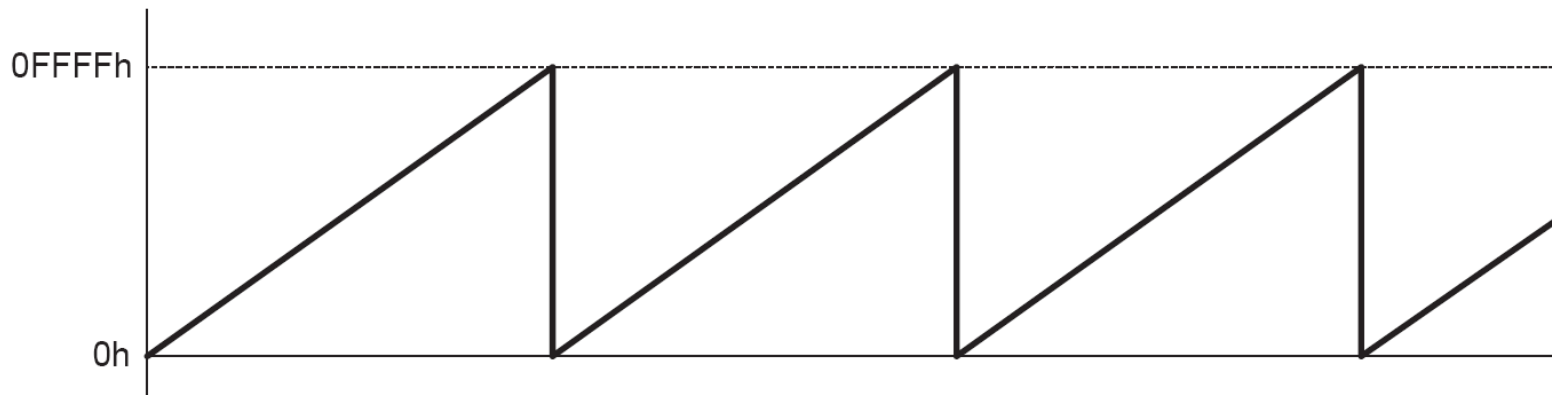




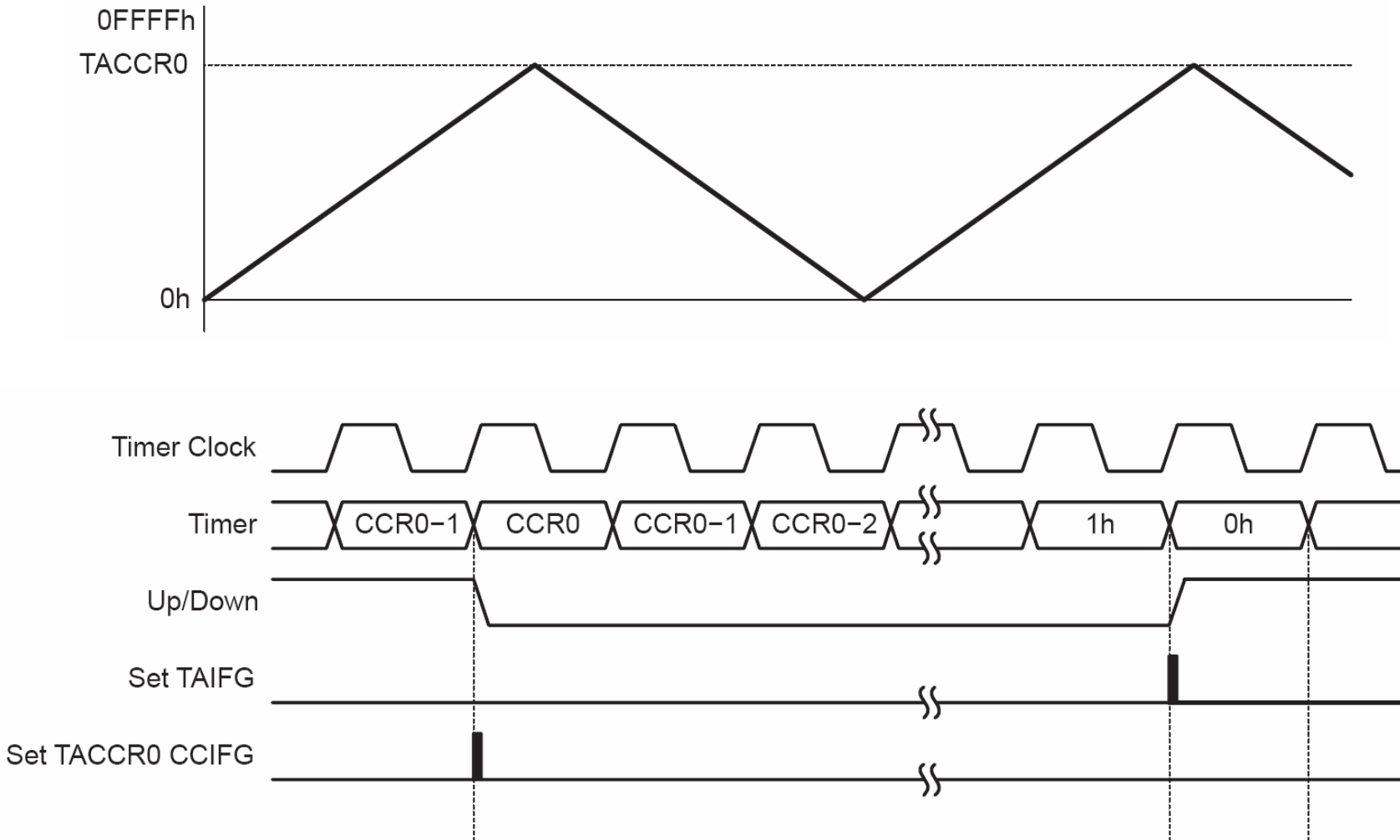
Modes of Operation: Up Mode



Modes of Operation: Continuous Mode



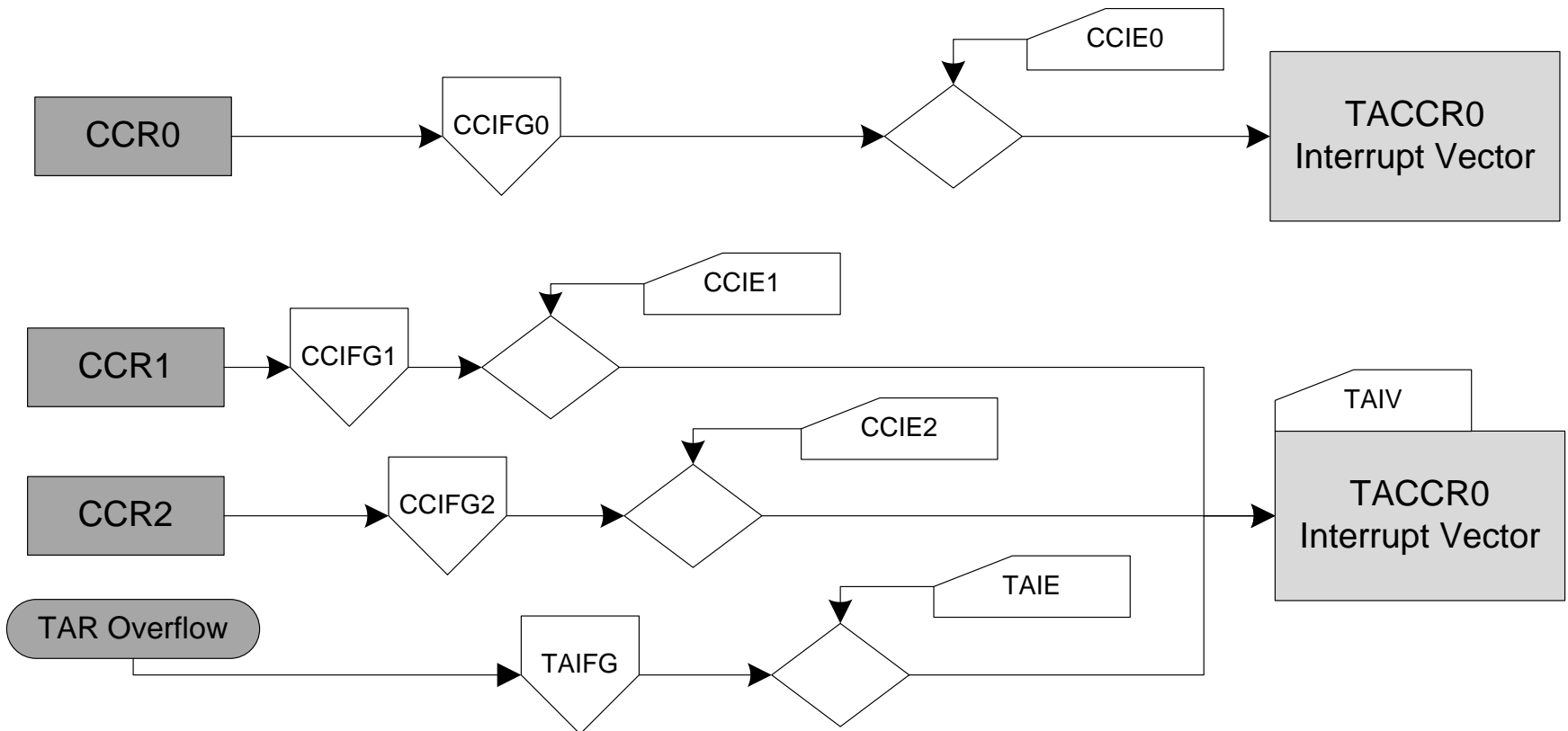
Modes of Operation: Up/Down Mode





Timer_A Interrupt Vectors

- TACCR0 interrupt vector for CCIFG of CCR0
- TAIV interrupt vector for TAIFG and CCIFGs of CCR1, CCR2





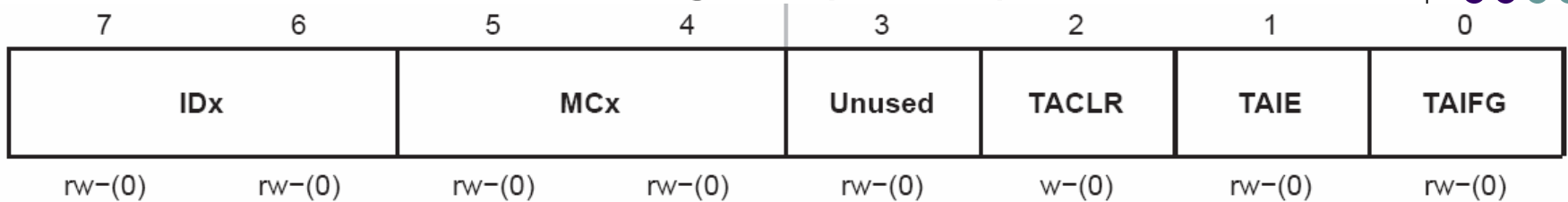
Timer_A Registers

- **TACTL, Timer_A Control Register (PART 1)**



Unused	Bits 15-10	Unused
TASSELx	Bits 9-8	Timer_A clock source select
		00 TACLK
		01 ACLK
		10 SMCLK
		11 INCLK

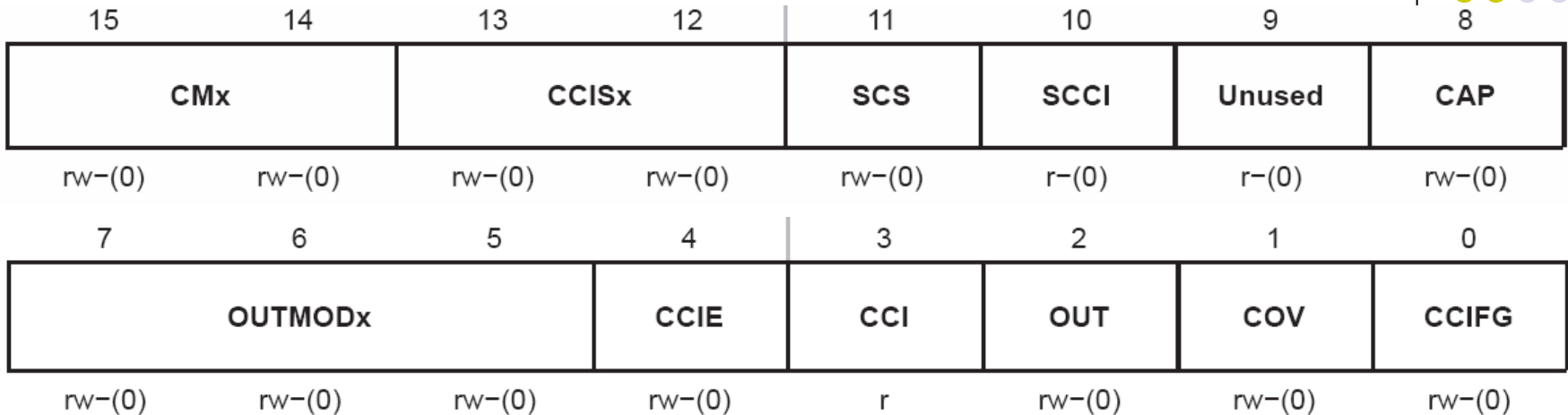
● TACTL, Timer_A Control Register (PART 2)



IDx	Bits 7-6	Input divider. These bits select the divider for the input clock. 00 /1 01 /2 10 /4 11 /8
MCx	Bits 5-4	Mode control. Setting MCx = 00h when Timer_A is not in use conserves power. 00 Stop mode: the timer is halted 01 Up mode: the timer counts up to TACCR0 10 Continuous mode: the timer counts up to 0FFFFh 11 Up/down mode: the timer counts up to TACCR0 then down to 0000h
Unused	Bit 3	Unused
TACLR	Bit 2	Timer_A clear. Setting this bit resets TAR, the TACLK divider, and the count direction. The TACLR bit is automatically reset and is always read as zero.
TAIE	Bit 1	Timer_A interrupt enable. This bit enables the TAIFG interrupt request. 0 Interrupt disabled 1 Interrupt enabled
TAIFG	Bit 0	Timer_A interrupt flag 0 No interrupt pending 1 Interrupt pending



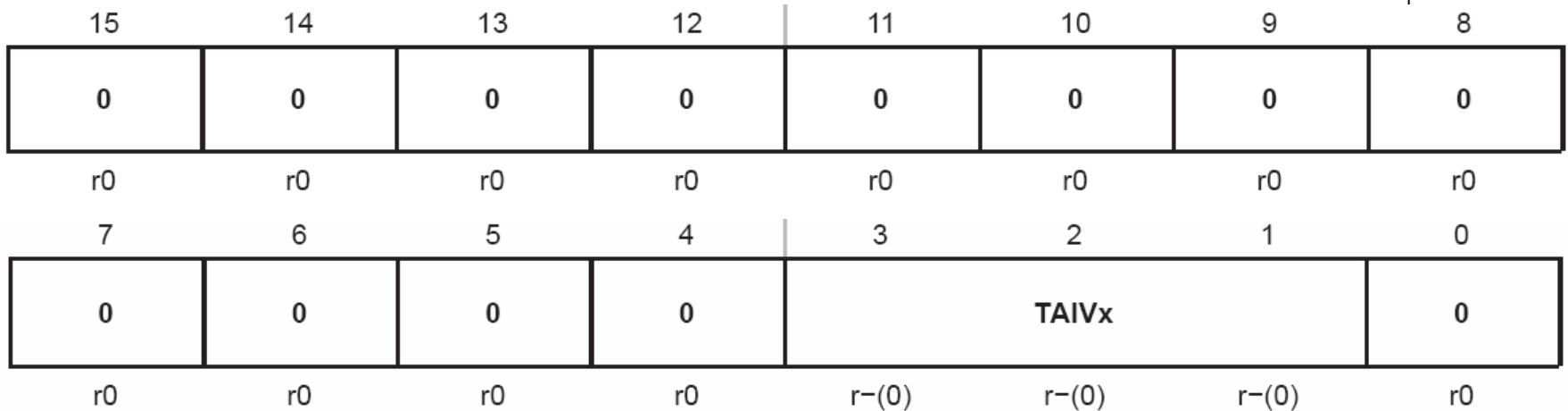
● TACCTLx, Capture/Compare Control Register



- CAP** Bit 8 Capture mode
 0 Compare mode
 1 Capture mode
- CCIE** Bit 4 Capture/compare interrupt enable. This bit enables the interrupt request of the corresponding CCIFG flag.
 0 Interrupt disabled
 1 Interrupt enabled
- CCIFG** Bit 0 Capture/compare interrupt flag
 0 No interrupt pending
 1 Interrupt pending



- TAIV, Timer_A Interrupt Vector Register**



TAIV Contents	Interrupt Source	Interrupt Flag	Interrupt Priority
00h	No interrupt pending	-	
02h	Capture/compare 1	TACCR1 CCIFG	Highest
04h	Capture/compare 2	TACCR2 CCIFG	
06h	Reserved	-	
08h	Reserved	-	
0Ah	Timer overflow	TAIFG	
0Ch	Reserved	-	
0Eh	Reserved	-	Lowest



Example 1

Continuous Mode

Output pin P6.0 with toggle rate = $32768/(2*50) = 328\text{Hz}$

```
#include "include/include.h"
#include "include/hardware.h"

void main ( void )
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
    P6DIR |= 0x01;                       // P6.0 output
    CCTL0 = CCIE;                         // CCR0 interrupt enabled
    CCR0 = 50;
    TACTL = TASSEL_1 + MC_2;             // ACLK, contmode
    eint();                               // Enable the global interrupt
    LPM0;                                 // Enter low power mode
}

// Timer_A TACCR0 interrupt vector handler
interrupt (TIMERA0_VECTOR) TimerA_procedure( void ){
    P6OUT ^= 0x01;                       // Toggle P6.0
    CCR0 += 50;                           // Add offset to CCR0
}

// OR
_BIS_SR(LPM0_bits + GIE);
```



Example 2

Up Mode

Output pin P6.0 with toggle rate = $32768/(2*50) = 328\text{Hz}$

```
#include "include/include.h"
#include "include/hardware.h"

void main ( void )
{
    WDTCTL = WDTPW + WDTHOLD;    // Stop WDT
    P6DIR |= 0x01;              // P6.0 output
    CCTLO = CCIE;               // CCR0 interrupt enabled
    CCR0 = 50-1; ?
    TACTL = TASSEL_1 + MC_1;    // ACLK, upmode
    _BIS_SR(LPM0_bits + GIE);   // Enable the global interrupt and enter LPM0
}

// Timer_A TACCR0 interrupt vector handler
interrupt (TIMERA0_VECTOR) TimerA_procedure ( void ){
    P6OUT ^= 0x01;             // Toggle P6.0
}
```



Example 3

Continuous Mode

Output pin P6.0 with toggle rate = $32768/(2*5)$ = 3277Hz

Output pin P6.1 with toggle rate = $32768/(2*50)$ = 328Hz

Output pin P6.2 with toggle rate = $32768/(2*500)$ = 32.79Hz

Output pin P6.3 with toggle rate = $32768/(2*65536)$ = 0.25Hz

```
#include "include/include.h"
#include "include/hardware.h"
void main ( void )
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop WDT
    P6DIR |= 0x0F;                       // P6.0, P6.1, P6.2 and P6.3 output
    CCTL0 = CCIE;                         // CCR0 interrupt enabled
    CCTL1 = CCIE;                         // CCR0 interrupt enabled
    CCTL2 = CCIE;                         // CCR0 interrupt enabled
    CCR0 = 0;
    CCR1 = 0;
    CCR2 = 0;
    TACTL = TASSEL_1 + MC_2+ TAIE;       // ACLK, contmode, TAIE enabled
    _BIS_SR(LPM0_bits + GIE);           // Enable the global interrupt and enter LPM0
}
```



Example 3, continued

```
// Timer_A TACCR0 interrupt vector handler
interrupt (TIMERA0_VECTOR) TimerA0_procedure ( void ){
    P6OUT ^= 0x01;           // Toggle P6.0
    CCR0 += 5;              // Add offset to CCR0
}

// Timer_A TAIV interrupt vector handler
interrupt (TIMERA1_VECTOR) TimerA1_procedure ( void ){
    switch( TAIV ) ?
    {
        case 2: P6OUT ^= 0x02;           // Toggle P6.1
                CCR1 += 50;            // Add offset to CCR1
                break;

        case 4: P6OUT ^= 0x04;           // Toggle P6.2
                CCR2 += 500;           // Add offset to CCR2
                break;

        case 10: P6OUT ^= 0x08;          // Toggle P6.3
                break;
    }
}
```