Unit 1

Module: Clocks and Timers: Sources & Controls -2

Introduction

Timers are often thought of as the heartbeat of an embedded system. Whether you need a periodic wake-up call, a one-time delay, or need a means to verify that the system is running without failure, Timers are the solution.

MSP430's TIMER_A module. Not only does it provide rudimentary counting/timing features, but provides sophisticated capture and compare features that allow a variety of complex waveforms – or interrupts – to be generated. In fact, this timer can even generate PWM (pulse width modulation) signals.

Timers of MSP430

MSP430 series chips have two types of Timers known as Timer_A and Timer_B. Both of them are 16 bit timers with several capture compare channels and selectable clock sources. Timer_B is slightly complex than Timer_A and offers more extensive interrupt capabilities and capture compare channels.

MSP430Gxxx series chips that ship with Launchpad development board usually have two Timer_A modules named as Timer0_A3 and Timer1_A3.Here A3 indicates the number of Capture Compare registers (here 3 registers) available in each timer.Timer A is a 16 bit Timer/Counter with 3 capture compare registers.

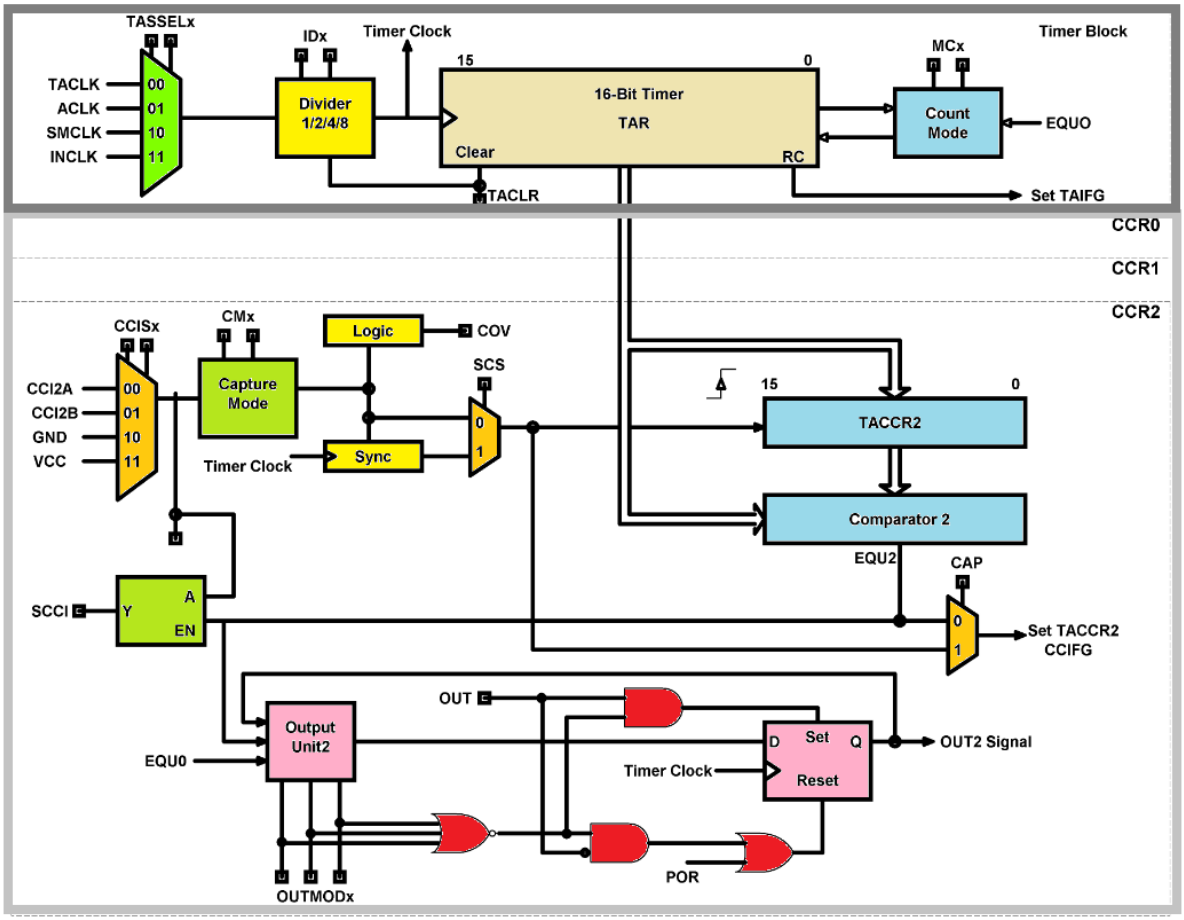
MSP430 Timer_A Introduction

Timer_A is a 16-bit timer/counter with three capture/compare registers. Timer_A can support multiple capture/compares, PWM outputs, and interval timing. Timer_A also has extensive interrupt capabilities. Interrupts may be generated from the counter on overflow conditions and from each of the capture/compare registers.

Timer_A features include:

- Asynchronous 16-bit timer/counter with four operating modes
- Selectable and configurable clock source
- Two or three configurable capture/compare registers
- Configurable outputs with PWM capability
- Asynchronous input and output latching
- Interrupt vector register for fast decoding of all Timer_A interrupts

Simplified Block Diagram of Timer_A



The Timer can be clocked either by SMCLK, ACLK,TACLK or INCLK.

SMCLK is an internal clock generally taken from DCO (Digitally Controlled Oscillator) and is in the Megahertz range. The DCO clock is temperature dependent and may change depending upon temperature.

ACLK is an internal clock taken either from VLO or an external 32KHz clock crystal connected to the MSP430.ACLK when taken from an external clock crystal is usually stable and do not change with temperature. The ACLK frequency is usually in Kilo hertz range (32KHz).

TACLK and INCLK are external clocks that can be connected to MSP430 through external pins.

TASSELx bits are used to select the required clock to the 16 bit Timer register and IDx bits are used to select clock divider which are present in the TACTL register.

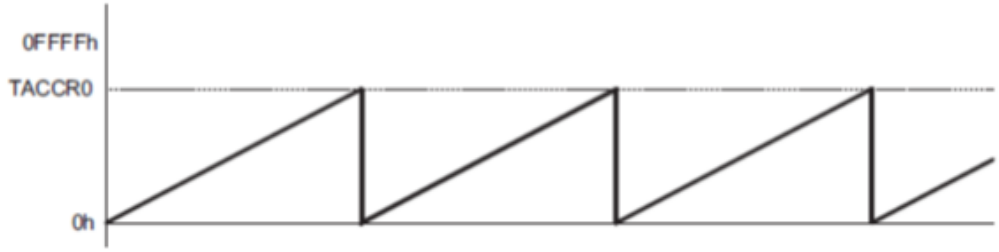
The 16 bit Timer A register(TAR) starts counting on the positive edge of the clock. TAR can be cleared by setting the TACLR bit to 1.

Timer A Operating Modes

Timer_A has 4 operating modes selected by the MCx bits in TACTL register.

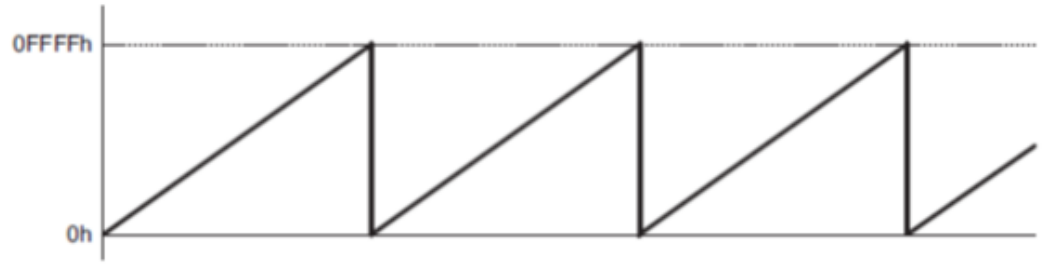
- MCx = 00 – Timer is halted and Timer stops counting
- MCx = 01 – Timer is started in the Up Mode
- MCx = 10 – Timer is started in Continuous Mode
- MCx = 11 – Timer is started in Up/Down Mode

Up Mode



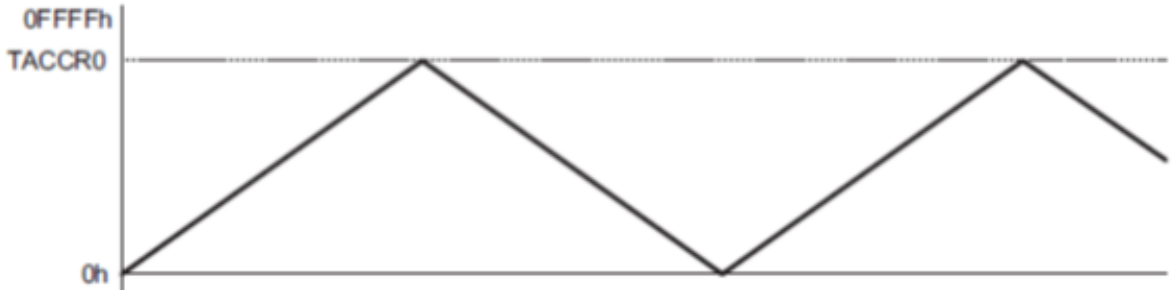
User has to load count in TA0CCR0 register. The timer register TAR starts incrementing from 0 after each clock cycle and compares its value continuously with value of TACCR0 register. When the TAR register value equals TACCR0, the TAR register rolls back to 0 and the timer restarts counting from zero. The TAIFG interrupt flag is set when the TAR count is roll back to zero. This process is repeated continuously.

Continuous Mode



The timer register TAR starts increments from 0 after each clock cycle and compares its value continuously with FFFF H. When the TAR register value equals to FFFF H, the TAR register roll back to value 0 and the timer restarts counting from zero. The TAIFG interrupt flag is set when the TAR count is roll back to zero. This process is repeated continuously. When TACCR0 is modified while running, the timer counts up to the new period and if the new period is less than the current count value.

Up/Down Mode



User has to load count in TA0CCR0 register. The timer repeatedly counts up to the value of compare register TACCR0 and back down to zero. The period is twice the value in TACCR0. The TACCR0 CCIFG interrupt flag and the TAIFG interrupt flag are set only once during a period, separated by 1/2 the timer period. The TACCR0 CCIFG interrupt flag is set when the timer counts from TACCR0−1 to TACCR0, and TAIFG is set when the timer completes counting down from 0001h to 0000h

Timer_A Registers

Register	Short Form
Timer_A control	TACTL
Timer_A counter	TAR
Timer_A capture/compare control 0	TACCTL0
Timer_A capture/compare 0	TACCR0
Timer_A capture/compare control 1	TACCTL1
Timer_A capture/compare 1	TACCR1
Timer_A capture/compare control 2	TACCTL2
Timer_A capture/compare 2	TACCR2
Timer_A interrupt vector	TAIV

16-Bit Timer Counter Register (TAR)

The 16-bit timer/counter register, TAR, increments or decrements (depending on mode of operation) with each rising edge of the clock signal. The timer can generate an interrupt when it overflows. TAR may be cleared by setting the TACLR bit. Setting TACLR also clears the clock divider and count direction for up/down mode.

Clock Source Select and Divider

The timer clock can be sourced from ACLK, SMCLK, or externally via TACLK or INCLK and it is selected with the TASSELx bits. The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the IDx bits.

TACTL Register

15	14	13	12	11	10	9	8
Unused						TASSELx	
7	6	5	4	3	2	1	0
IDx		MCx		Unused	TACLR	TAIE	TAIFG

TASSELx (bits 9 and 8) Timer_A clock source select

Clock Source Select: The timer clock can be sourced from ACLK, SMCLK, or externally via TACLK or INCLK and it is selected with the TASSELx bits.

- 00 TACLK
- 01 ACLK
- 10 SMCLK
- 11 INCLK

TACTL Register – IDx (bits 7 and 6)

The selected clock source may be passed directly to the timer or divided by 2, 4, or 8, using the IDx bits.

IDx (Bits 7-6): Input divider. These bits select the divider for the input clock.

- 00 /1
- 01 /2
- 10 /4
- 11 /8

TACTL Register – MCx (bits 5 and 4)

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 clock 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

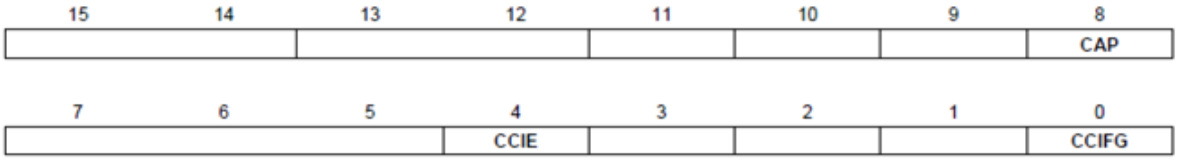
1 Interrupt enabled

TAIFG (Bit 0)Timer_A interrupt flag

0 No interrupt pending

1 Interrupt pending

TACCTLx, Capture/Compare Control Register



Important bits in compare mode to use timer to generate delay are:

CAP Bit 8 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

Timer A Interrupts

Two interrupt vectors are associated with the 16-bit Timer_A module one for TACCR0 and the other for rest of the channels as well as the Timer overflow (TAIFG):

TACCR0 interrupt vector for TACCR0 CCIFG

TAIV interrupt vector for all other CCIFG flags and TAIFG

In capture mode any CCIFG flag is set when a timer value is captured in the associated TACCRx register.

The register TACCR0 has a separate vector location which has a higher priority compared to the other timer interrupts.

Rest of the capture compare channels (CCR1 and CCR2) along with the Timer overflow (TAIFG) are given separate vector location. Since all of them are combined together the interrupt vector register TAIV is used to determine which flag requested an interrupt.

In compare mode, any CCIFG flag is set if TAR counts to the associated TACCRx value. Software may also set or clear any CCIFG flag. All CCIFG flags request an interrupt when their corresponding CCIE bit and the GIE bit are set.

Example Code to to blink an LED connected to P1.0 of MSP430G2553 (Launchpad board) whenever TAR overflows

```
#include "msp430g2553.h"
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD; // Stop the WDT

  // Port Configuration
  P1DIR |= BIT0; // P1.0 output
  P1OUT &= ~BIT0; // P1.0 = 0

  // Timer 0 Configuration
  TACTL |= MC_0; // Stop Timer0_A3
  TACTL  = TASSEL_1 + ID_2 + TAIE;// ACLK,ACLK/4,enable TAR interrupt
  TACTL |= MC_2; // Start Timer0 in Continuous Mode

  _BIS_SR(LPM0_bits + GIE); // Sleep in LPM0 with interrupts enabled
}

#pragma vector = TIMER0_A1_VECTOR //Timer0,TAIFG interrupt vector
__interrupt void TimerA(void)
{
  switch(TAIV)
  {
    case 0x002: // CCR1 not used
              break;
    case 0x004: //CCR2 not used
              break;
    case 0x00A: P1OUT ^= BIT0; //TAR overflow interrupt
              break;
  }
}
```

Example:

In this configuring all the capture compare registers(CCR) as well as the timer overflow interrupt. We enable the TAIFG interrupt in TACTL first and then load all the CCR registers with the count value.After that CCR interrupts for the respective registers are enabled and counter is started in continuous mode.

```
// Timer 0 Configuration
TACTL = MC_0;// Stop the timer
TACTL = TASSEL_2 + ID_3 + TAIE;// Timer Clock = SMCLK,1MHz/8 = 125KHz, enable TAR interrupt

CCR0 = 10000; // Count in CCR0 register
CCR1 = 30000; // Count in CCR1 register
CCR2 = 40000; // Count in CCR2 register

CCTL0 |= CCIE; // Enable CCR0 interrupt
CCTL1 |= CCIE; // Enable CCR1 interrupt
CCTL2 |= CCIE; // Enable CCR2 interrupt

TACTL |= MC_2; // Start the timer in continuous mode

_BIS_SR(LPM0_bits + GIE); // Sleep in LPM0 with interrupts enabled
```

When the Timer register counts and reaches the value 10000, CCR0 interrupt is triggered and the program control goes to the CCR0 interrupt routine of Timer 0. You can then lit up an LED or do something.

```
#pragma vector = TIMER0_A0_VECTOR //CCR0 interrupt vector

__interrupt void CCR0_Interrupt(void)

{

    P1OUT ^= BIT0; //toggle an LED

}
```

When TAR reaches the count = 30000 the CCR1 interrupt is triggered, CCR2 interrupt is triggered at 40000 and finally the timer overflow (TAIFG) is triggered as TAR rolls over.