Timer_A and Timer_B Introduction (1/6)

- Timer A and B are two general-purpose 16-bit counter/event timers;
- There are slight differences between the two timers;
- Features common to both timers include:
  - **Asynchronous 16-bit timer/counter with four operating modes:**
    - Timer_A length: 16 bits;
    - Timer_B length: programmable: 8, 10, 12, or 16 bits.
    - Timer/counter register, TAR (Timer_A) or TBR (Timer_B) -from now on described as TxR- 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;
    - Wide interrupt interval range: 1/MCLK to 32 seconds.

Timer_A and Timer_B Introduction (2/6)

- **Choice of selectable and configurable clock source:**
  - ACLK;
  - SMCLK;
  - External - via TACLK or INCLK (TASSELx bits);
  - The selected clock source may additionally be divided by 2, 4, or 8 (IDx bits configuration).

- **Configurable capture/compare registers:**
  - Timer_A has 3 or 5 capture/compare registers;
  - Timer_B has 3 or 7 capture/compare registers;
  - Timer_B capture/compare registers can be grouped.

Timer_A and Timer_B Introduction (3/6)

- **Configurable outputs and internal connections to several other modules:**
  - Faster response;
  - No cycles are wasted while the Interrupt Service Routine (ISR) loads/executes;
  - Avoids CPU wakeup;
  - Saves power.

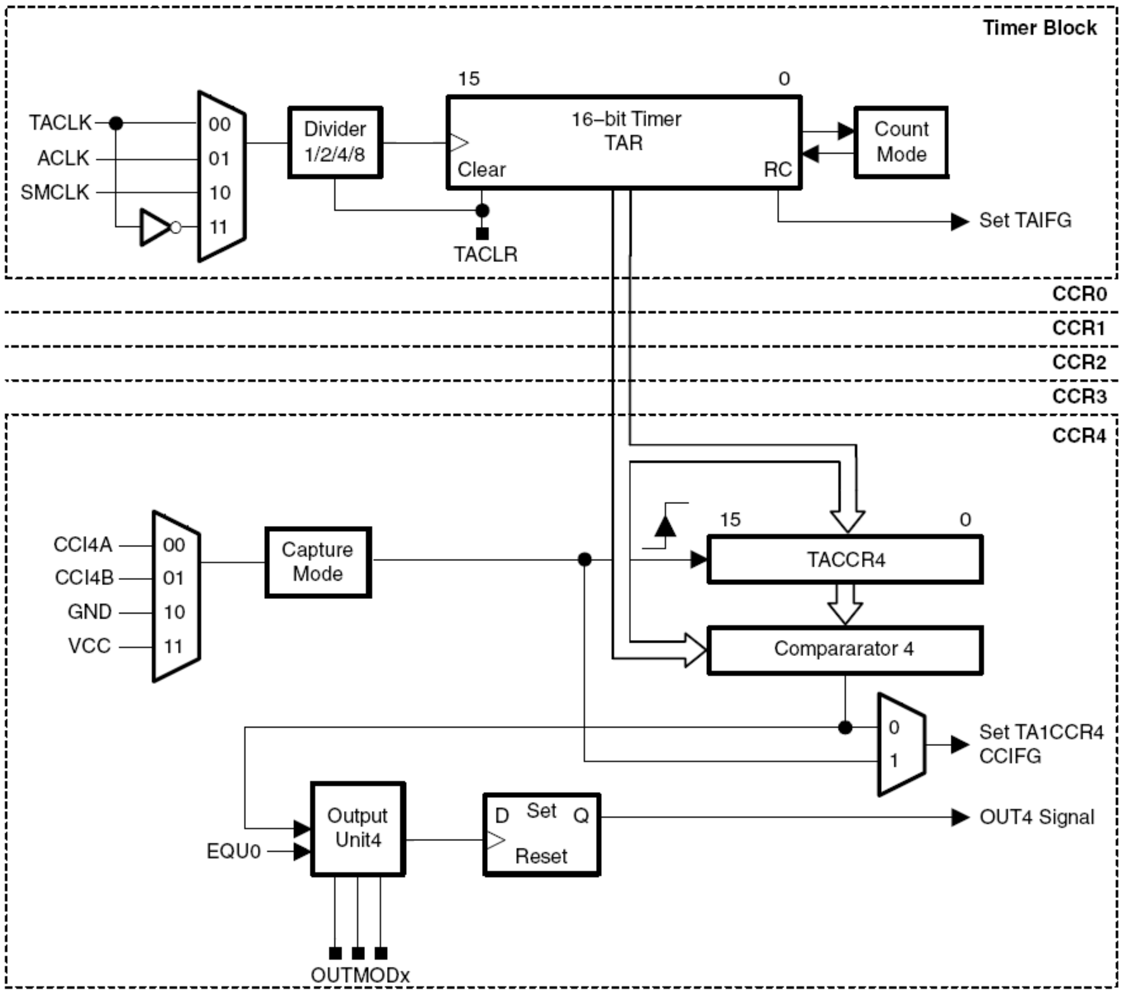
  - **Outputs capability: Pulse Width Modulation (PWM);**

Timer_A and Timer_B Introduction (4/6)

- **Asynchronous input and output latching:**
  - Timer_A Capture/Compare (Cap/Com) registers are not buffered, being updated immediately when written to;
  - Timer_B Cap/Com registers are double-buffered with synchronized loading
- **Interrupt vector register for fast decoding of all Timer_A and Timer_B interrupts:**
  - TACCR0 (or TBCCR0) interrupt vector for TACCR0 (or TBCCCR0) CCIFG;
  - TAIV (or TBIV) interrupt vector for the remaining CCIFG flags and TAIFG (or TBIFG).

Timer_A and Timer_B Introduction (5/6)

– **Block diagram (Timer_A):**



>> Contents

Timer_A and Timer_B Introduction (6/6)

– **Timers have four modes of operation:**

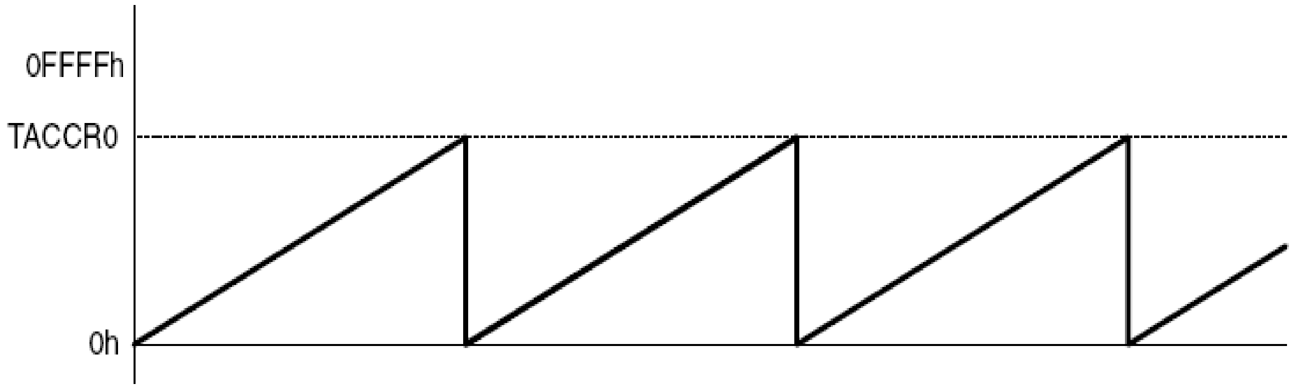
- MCx bits (Timer_A or Timer_B Control Register)

MCx	Mode	Description
0 0	**Stop**	**The timer is halted**
0 1	**Up**	**Up counting mode (from 0x0000 to the value in the TACCR0 or TBCCR0 register)**
1 0	**Continuous**	**Continuous counting mode (from 0x0000 to 0xFFFF)**
1 1	**Up/down**	**Up/down counting mode (from 0x0000 to the value in the TACCR0 or TBCCR0 register and back down to zero)**

Timer_A and Timer_B operating modes (1/3)

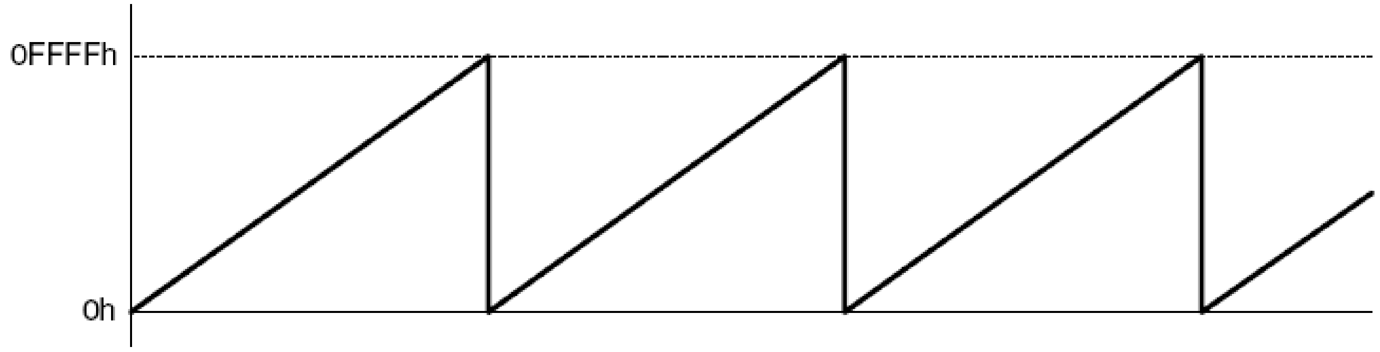
- Up mode:
  - TxR counts up till it reaches the value in the TxCCR0 register;
  - TxR->TxCCR0: TACCR0 interrupt flag, CCIFG, is set;
  - TxR=TxCCR0: EQU0 = 1 (restarts counting in TxR);
  - TxCCR0->0: TxIFG interrupt flag is set:
    - Interrupt period:

      t_{INT} = 1/[f_{CLK}/Prescaler/(TxCCR0+1)].



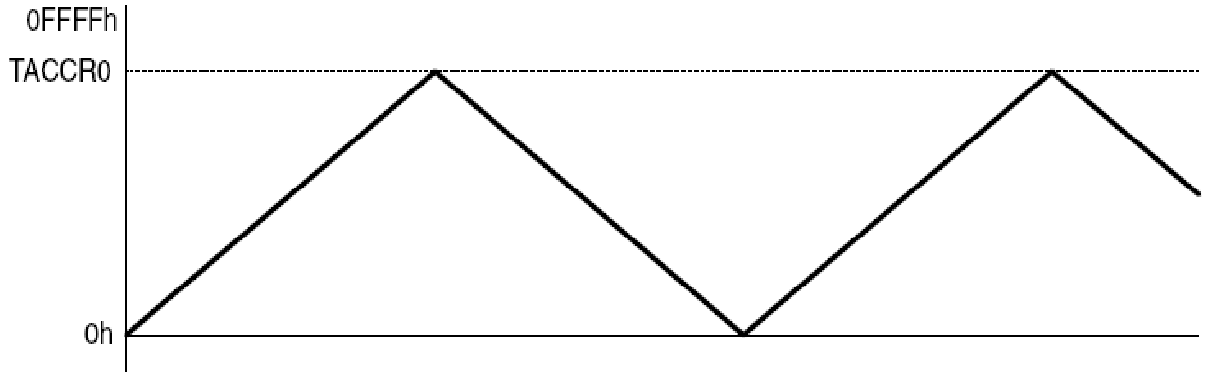
Timer_A and Timer_B operating modes (2/3)

- Continuous mode:
  - TxR counts up till it reaches 0xFFFF (65536 counts);
  - TxR=0xFFFF: TxR counting from zero (next clock pulse);
  - 0xFFFF->0: TxIFG interrupt flag is set:
    - Interrupt period: t_{INT} = 1/[f_{CLK}/Prescaler/65536];
  - (Correct only for TAR; for TBR 4 different end values. See User's Guide for additional details).



Timer_A and Timer_B operating modes (3/3)

- Up/down mode:
  - TxR counts up till it reaches the value in the TxCCR0 register;
  - TxCCR0-1 -> TxCCR0: Interrupt flag, CCIFG, is set;
  - TxR=TxCCR0: Counting is inverted;
  - 0x0001->0x0000: Interrupt flag TxIFG is set:
    - Interrupt period: t_{INT} = 1/[f$_{\mathrm{CLK}}$/Prescaler/(TxCCR0$\times$2];



>> Contents

Timer_A and Timer_B reset

- The timers can be reset by the following actions:
  - Writing 0 in the TxR register;
  - Writing 0 in the TxCCR0 register, provided that the timer is not in continuous mode;
  - Setting the TxCLR bit in the Timer Control Register (TxCTL).

>> Contents

- TACTL, Timer_A Control Register

15					10	9	8
Unused						TASSEL1	TASSEL0
7	6	5	4	3	2	1	0
ID1	ID0	MC1	MC0	Unused	TACLR	TAIE	TAIFG

Bit		Description
9-8	**TASSELx**	**Timer_A clock source: TASSEL1 TASSEL0 = 0 0 ⇒ TACLK** **TASSEL1 TASSEL0 = 0 1 ⇒ ACLK** **TASSEL1 TASSEL0 = 1 0 ⇒ SMCLK** **TASSEL1 TASSEL0 = 1 1 ⇒ INCLK**
7-6	**IDx**	**Clock signal divider: ID1 ID0 = 0 0 ⇒ / 1** **ID1 ID0 = 0 1 ⇒ / 2** **ID1 ID0 = 1 0 ⇒ / 4** **ID1 ID0 = 1 1 ⇒ / 8**
5-4	**MCx**	**Clock timer operating mode: MC1 MC0 = 0 0 ⇒ Stop mode** **MC1 MC0 = 0 1 ⇒ Up mode** **MC1 MC0 = 1 0 ⇒ Continuous mode** **MC1 MC0 = 1 1 ⇒ Up/down mode**
2	**TACLR**	**Timer_A clear when TACLR = 1**
1	**TAIE**	**Timer_A interrupt enable when TAIE = 1**
0	**TAIFG**	**Timer_A interrupt pending when TAIFG = 1**

- Timer_A (and Timer_B) contain independent capture and compare blocks, TACCRx (or TBCCRx);
- These blocks may be used to capture timer register contents, as they are at the time of an event, or to generate an event when the timer register contents correspond to the capture/compare register contents, e.g. to generate time intervals;
- The setting of capture/compare is selected by the mode bit CAP in the individual Capture/Compare Control registers, TACCTLx (or TBCCTLx)

- Capture mode:
  - Used to measure the period of time events with minimal CPU intervention.
    - **Procedure:**
    - Set the CAP bit to select capture mode;
    - Set the SCS bit to synchronize the capture with the next timer clock (recommended to avoid race conditions);
    - The input signal is sampled by the CCIxA (or CCIxB) input, selected by the CCISx bits in the Capture/Compare Control Register, TACCTLx (or TBCCTLx);

- The capture edge of the input signal (rising, falling, or both) is selected by the CMx bits;
- When a valid edge is detected on the selected input line, the value in the Timer register is latched into the TACCRx (or TBCCRx) register, providing a time mark for the event;
- The interrupt flag CCIFG is set;
- The bit COV (=1) controls an overflow event when a second capture is performed, before the value from the first capture is read.

- Compare mode:
  - Used for pulse generation or generation of interrupts at specific time intervals (PWM output signals).

  - **Procedure:**
    - Reset the CAP bit to select compare mode;
    - TxR counts up to the value programmed in the TxCCRx register;
    - When the timer value is equal to the value in the TxCCRx register, an interrupt is generated:
      - Interrupt flag CCIFG is set;
      - Internal signal EQUx = 1 (where x is the number of the CCR channel).

  - EQUx affects the output compare signal OUTx according to the output mode (defined by the OUTMODx bits in the TxCCTL;
  - The input signal CCI is latched into SCCI.

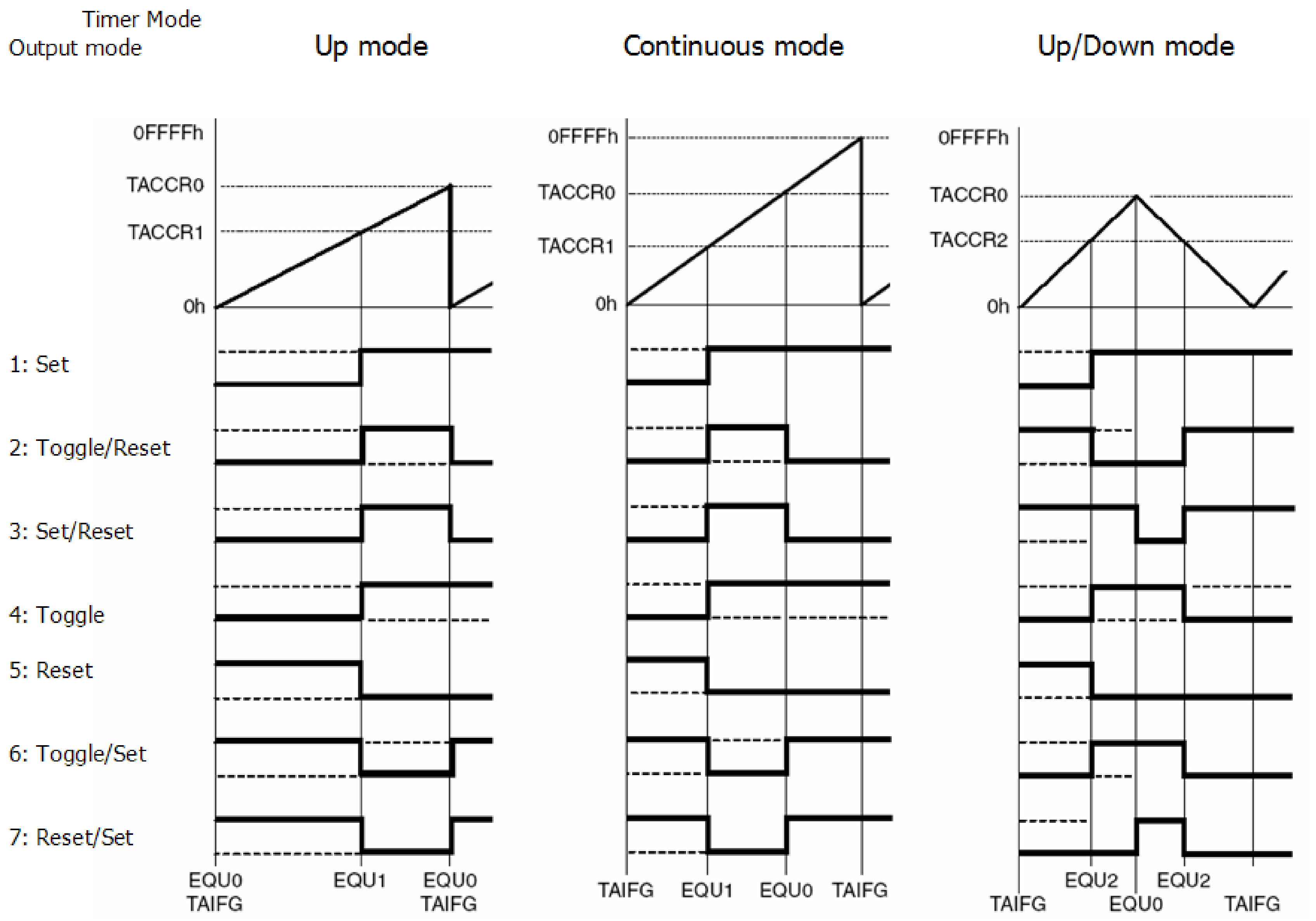
- Output operating modes uses:
  - Modes 2, 3, 6 and 7: PWM output signals;
  - Mode 3: active PWM signal at low state;
  - Mode 7: active PWM signal at high state;
  - Modes 2 and 6: complementary PWM signals;
  - Modes 1 and 5: single event generation;
  - Mode 4: signal with 1/2 frequency of the timer signal.

>> Contents

- Output operating modes (OUTMODx bits):

OUTMODx	Mode	Description
0 0 0	**Output**	**The output signal OUTx is defined by the bit OUTx**
0 0 1	**Set**	**OUTx = 1 ⇒ timer = TxCCRx** **OUTx = 0 ⇒ timer = 0 or until another output mode is selected and affects the output**
0 1 0	**Toggle/Reset**	**OUTx = toggle ⇒ timer = TxCCRx** **OUTx = 0 ⇒ timer = TxCCR0**
0 1 1	**Set/Reset**	**OUTx = 1 ⇒ timer = TxCCRx** **OUTx = 0 ⇒ timer = TxCCR0**
1 0 0	**Toggle**	**OUTx = toggle ⇒ timer = TxCCRx** **The output period is double the timer period**
1 0 1	**Reset**	**OUTx = 0 ⇒ timer = TxCCRx** **OUTx = 1 ⇒ another output mode is selected and affects the output**
1 1 0	**Toggle/Set**	**OUTx = toggle ⇒ timer = TxCCRx** **OUTx = 1 ⇒ timer = TxCCR0**
1 1 1	**Reset/Set**	**OUTx = 0 ⇒ timer = TxCCRx** **OUTx = 1 ⇒ timer = TxCCR0**

- Output examples:



>> Contents

- TACCTLx, Timer_A Cap/Com Control Register

15	14	13	12	11	10	9	8
CM1	**CM0**	**CCIS1**	**CCIS0**	**SCS**	**SCCI**	**Unused**	**CAP**

Bit		Description			
15-14	**CMx**	**Capture mode:**	**CM1 CM0 = 0 0**	⇒	**No capture**
			CM1 CM0 = 0 1	⇒	**Capture on rising edge**
			CM1 CM0 = 1 0	⇒	**Capture on falling edge**
			CM1 CM0 = 1 1	⇒	**Capture on both edges**
13-12	**CCISx**	**Capture/compare input select:**	**CCIS1 CCIS0 = 0 0**	⇒	**CCIxA**
			CCIS1 CCIS0 = 0 1	⇒	**CCIxB**
			CCIS1 CCIS0 = 1 0	⇒	**GND**
			CCIS1 CCIS0 = 1 1	⇒	**V_{cc}**
11	**SCS**	**Synchronize capture input signal with timer clock:**			
			SCS = 0	⇒	**Asynchronous capture**
			SCS = 1	⇒	**Synchronous capture**
10	**SCCI**	**Synchronized capture/compare input**			
8	**CAP**	**Mode:**	**Capture mode**	⇒	**CAP = 1**
			Compare mode	⇒	**CAP = 0**

• TACCTLx, Timer_A Cap/Com Control Register

7	6	5	4	3	2	1	0
OUTMOD2	**OUTMOD1**	**OUTMOD 0**	**CCIE**	**CCI**	**OUT**	**COV**	**CCIFG**

Bit		Description	
7-5	**OUTMODx**	**Output mode:**	**OUTMOD2 OUTMOD1 OUTMOD0 = 0 0 0 ⇒ bit OUT** **OUTMOD2 OUTMOD1 OUTMOD0 = 0 0 1 ⇒ Set** **OUTMOD2 OUTMOD1 OUTMOD0 = 0 1 0 ⇒ Toggle/Reset** **OUTMOD2 OUTMOD1 OUTMOD0 = 0 1 1 ⇒ Set / Reset** **OUTMOD2 OUTMOD1 OUTMOD0 = 1 0 0 ⇒ Toggle** **OUTMOD2 OUTMOD1 OUTMOD0 = 1 0 1 ⇒ Reset** **OUTMOD2 OUTMOD1 OUTMOD0 = 1 1 0 ⇒ Toggle / Set** **OUTMOD2 OUTMOD1 OUTMOD0 = 1 1 1 ⇒ Reset / Set**
4	**CCIE**	**Capture/compare interrupt enable when CCIE = 1.**	
3	**CCI**	**Capture/compare input**	
2	**OUT**	**Output state**	
1	**COV**	**Capture overflow when COV = 1**	
0	**CCIFG**	**Capture/compare interrupt flag CCIFG = 1 when interrupt pending**	

Timer_A and Timer_B Interrupts (1/3)

- Interrupt characteristics:
  - **Capture mode:**
    - Any CCIFG flag is set when a timer value is captured in the associated TxCCRx register.
  - **Compare mode:**
    - Any CCIFG flag is set if TxR counts up to the TxCCRx value.
    - Software may also set or clear a CCIFG flag;
    - All CCIFG flags request an interrupt when their corresponding CCIE bit and GIE bit are set.

Timer_A and Timer_B Interrupts (2/3)

- Interrupt vectors associated with Timer_A:
  - **TACCR0 interrupt vector for TACCR0 CCIFG:**
    - TACCR0 CCIFG flag has the highest priority Timer_A interrupt;
    - The TACCR0 CCIFG flag is automatically reset when the TACCR0 interrupt request is serviced.

Timer_A and Timer_B Interrupts (3/3)

- **TAIV interrupt vector for TACCR1 CCIFG to TACCR4 CCIFG and TAIFG:**
  - Flags are given priority and combined to source a single interrupt vector (decreasing priority);
  - TAIV determines which flag requests the interrupt;
  - Disabling interrupts do not affect the value in TAIV;
  - Any access (read/write) of TAIV automatically resets the highest pending interrupt flag;
  - If another interrupt flag is set, another interrupt is immediately generated after servicing the initial interrupt.

>> Contents

Timer_B special features (1/3)

- Programmable length of the TBR register (equivalent to TAR in Timer_A) to be 8, 10, 12, or 16 bits:
  - Configurable through selection of the CNTLx bits in TBCTL (equivalent to TACTL in Timer_A);
  - The maximum count value, TBR(maximum), for the selectable lengths is 0FFh, 03FFh, 0FFFh, and 0FFFFh, respectively;
- Three or seven capture/compare blocks TBCCRx;

>> Contents

Timer_B special features (2/3)

- Double-buffered compare latches with synchronized loading:
  - In Timer_A, the signal generation in compare mode may cause noise during compare period updates because the TACRRx value is used directly to compare with the timer value;
  - To avoid this condition, the compare latches TBCLx, buffered by TBCCRx, holds the data for the comparison to the timer value in compare mode;
  - The CLLDx bits at the TBCCTLx register configure the timing of the transfer from TBCCRx to TBCLx.

Timer_B special features (3/3)

- Grouping channels capability:
  - Multiple compare latches may be grouped together for simultaneous updates of the TBCLGRPx bits;
  - Two conditions are required:
    - All TBCCRx registers must be updated;
    - The load event controlled by the CLLDx bits must occur.
- All outputs can be put into a high-impedance state:
  - TBOUTH = 1 puts Timer_B outputs into a high-impedance state, allowing higher security and lower delay time responding to failures.
- The SCCI bit function is not implemented.