EE4390 Microprocessors

Lessons 19 - 22

Standard Timer Module (TIM)

Standard Timer Module (TIM)

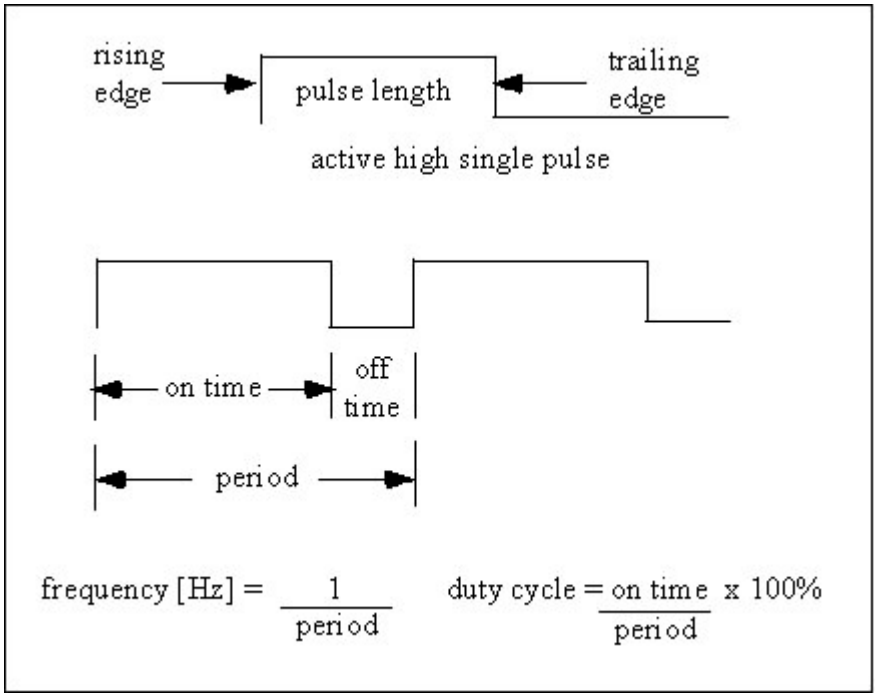
- Fundamental Timing Concepts
- The Standard Timer Module
- Input Capture - measure signal parameters
  - system description
  - register description
  - programming
- Output Compare - generate precision signals
  - system description
  - register description
  - programming

Standard Timer Module (TIM)
Fundamental Timing Concepts

- TIM is a 16-bit binary stop watch
  - contains 8 independent input/output channels
  - all channels use the same “stopwatch” -- the free running counter
  - input -- input capture -- capture key parameters of input signals
    - pulse length, frequency, duty cycle
  - output -- output compare -- generate precision digital signals
    - pulses, square waves, pulse width modulated signals
  - pulse accumulator -- count events

Standard Timer Module (TIM)
Fundamental Timing Concepts (cont)

Signal characteristics

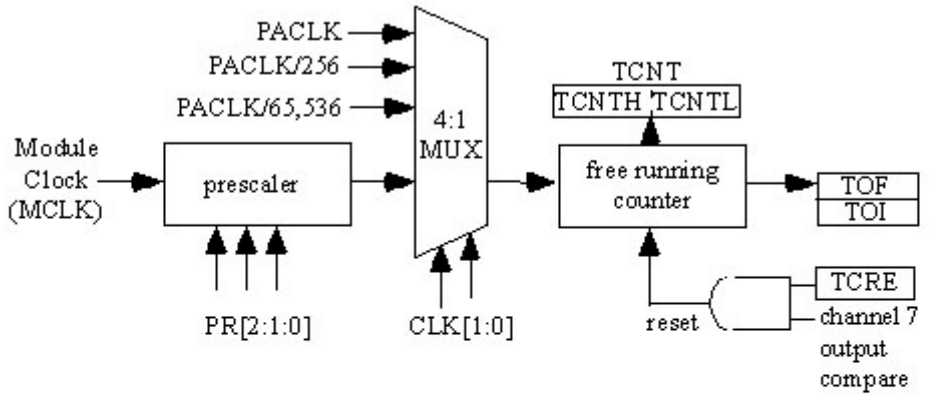


Standard Timer Module
Fundamental Timing (cont)

Free running counter

- Timer Enable (TEN)
- MCLK
- prescalar
- frc - TCNT
- timer overflow (TOF)
- timer overflow interrupt (TOI)
- frc examples

pg 252



Register: Timer System Control Register (TSCR) Address: $0086

7	6	5	4	3	2	1	0
TEN	TSWAI	TSBCK	TFFCA	0	0	0	0
Reset: 0	0	0	0	0	0	0	0

Register: Timer Interrupt Flag 2 (TFLG2) Address: $008F

7	6	5	4	3	2	1	0
TOF	0	0	0	0	0	0	0
Reset: 0	0	0	0	0	0	0	0

Register: Timer Interrupt Mask Register 2 (TMSK2) Address: $008D

7	6	5	4	3	2	1	0
TOI	0	TPU	TDRB	TCRE	PR2	PR1	PR0
Reset: 0	0	1	1	0	0	0	0

Register: Pulse Accumulator Control Register (PACTL) Address: $00A0

7	6	5	4	3	2	1	0
0	PAEN	PAMOD	PEDGE	CLK1	CLK0	PAOVI	PAI
Reset: 0	0	0	0	0	0	0	0

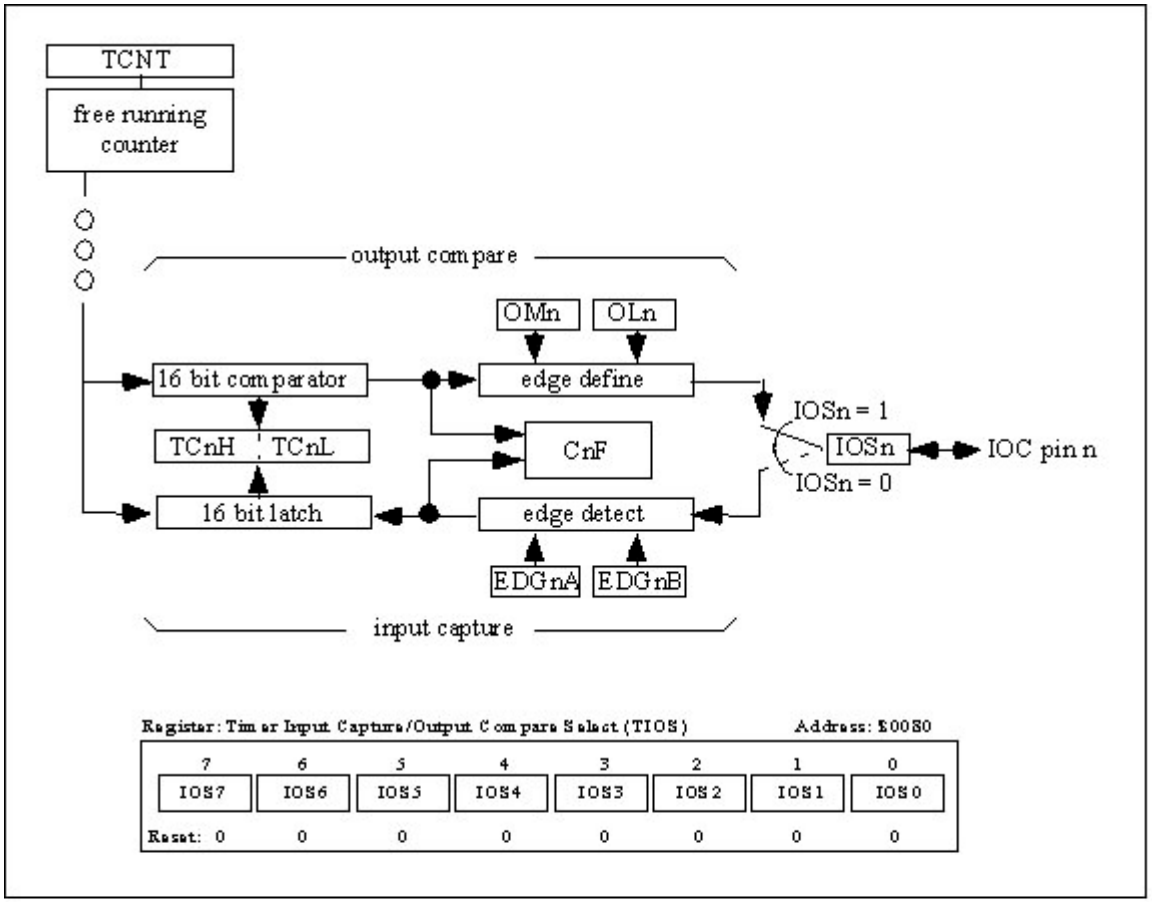
Register: Timer Counter Register High (TCNTH) Address: $0084

7	6	5	4	3	2	1	0
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reset: 0	0	0	0	0	0	0	0

Register: Timer Counter Register Low (TCNTL) Address: $0085

7	6	5	4	3	2	1	0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Reset: 0	0	0	0	0	0	0	0

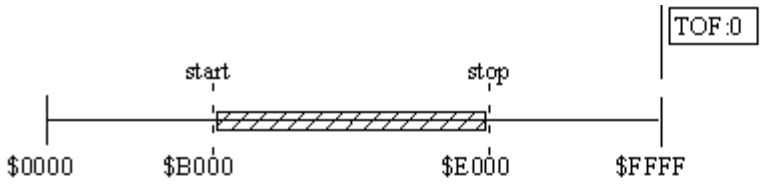
Standard Timer Module (TIM)
Input Capture/Output Compare Channel



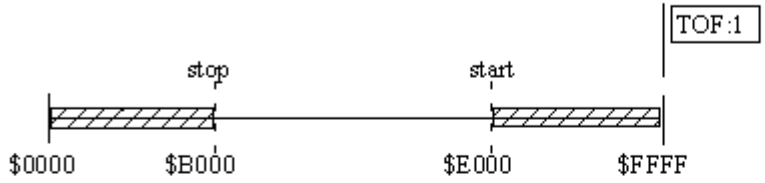
Register: Timer Input Capture/Output Compare Select (TIOS) Address: $0080

7	6	5	4	3	2	1	0
IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0
Reset: 0	0	0	0	0	0	0	0

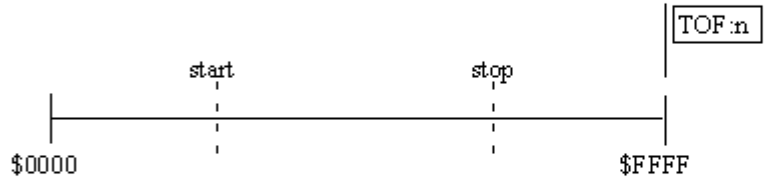
Calculating Elapsed Time



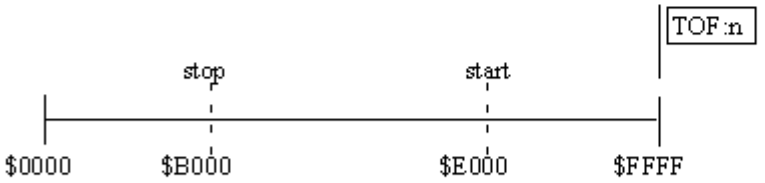
a) **Case 1:** Stop time count > Start time count, no timer overflows



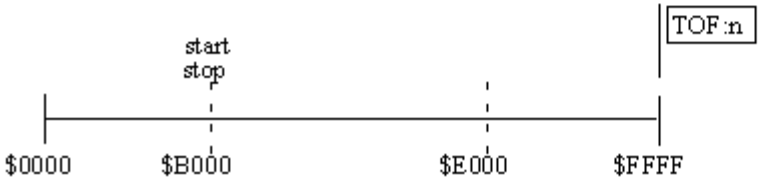
b) **Case 2:** Start time count > Stop time count, one timer overflow



c) **Case 3:** Stop time count > Start time count, n timer overflows



d) **Case 4:** Start time count > Stop time count, n timer overflows



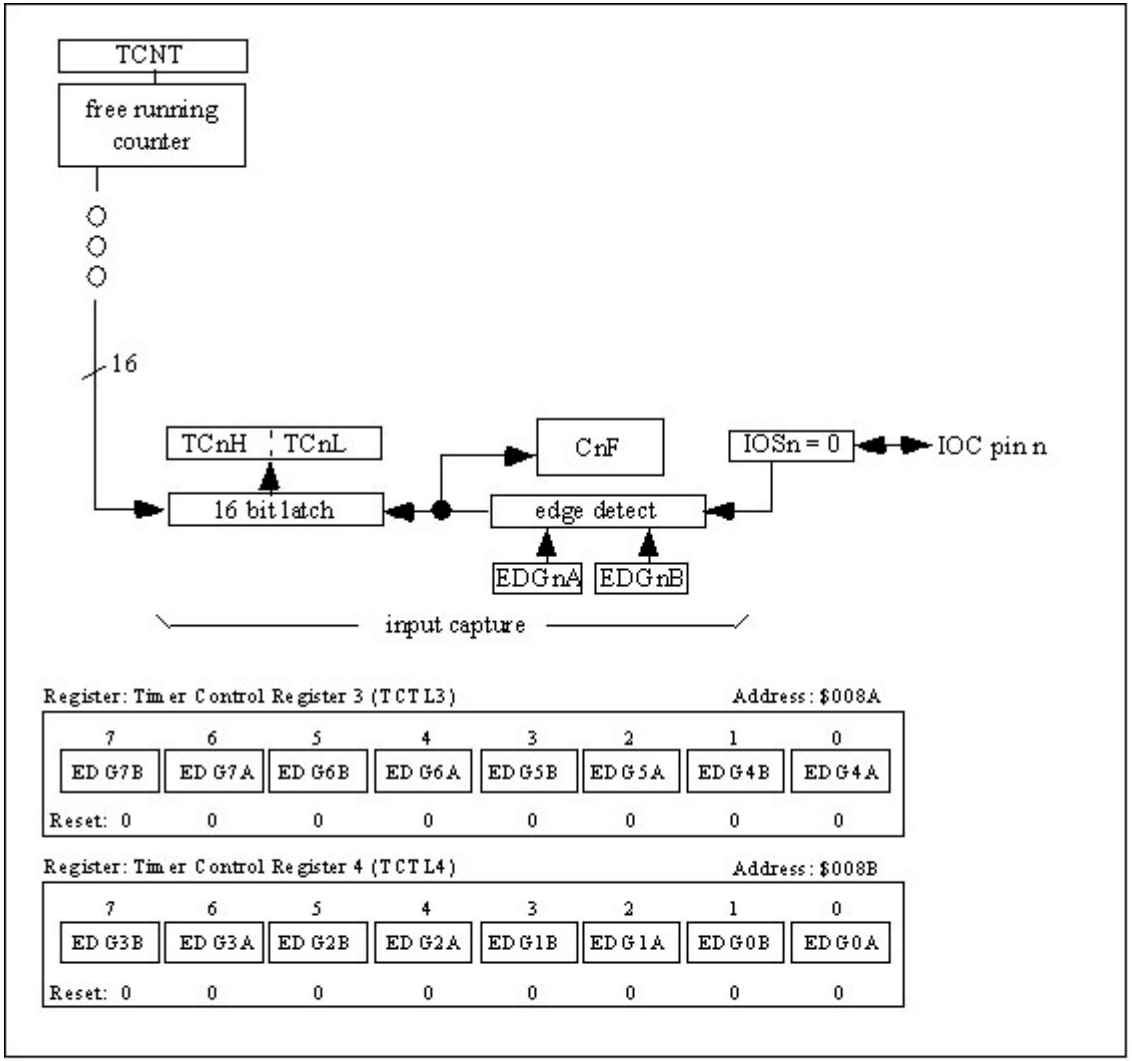
e) **Case 5:** Start time count = Stop time count, n timer overflows

Standard Timer Module (TIM)
Input Capture - measure signal parameters

- Input Capture - measure signal parameters
  - system description
  - register description
  - programming
- General concept
  - capture value of free running counter when user-specified event occurs
    - rising edge, falling edge, any edge
  - calculate elapsed time between key events
  - EX] pulse length, pg 257-258

EDGnB EDGnA	Configuration
00	Capture disabled
01	Rising Edge
10	Falling Edge
11	Any Edge

Standard Timer Module (TIM)
Input Capture - measure signal parameters



Register: Timer Control Register 3 (TCTL3) Address: $008A

7	6	5	4	3	2	1	0
EDG7B	EDG7A	EDG6B	EDG6A	EDG5B	EDG5A	EDG4B	EDG4A
Reset: 0	0	0	0	0	0	0	0

Register: Timer Control Register 4 (TCTL4) Address: $008B

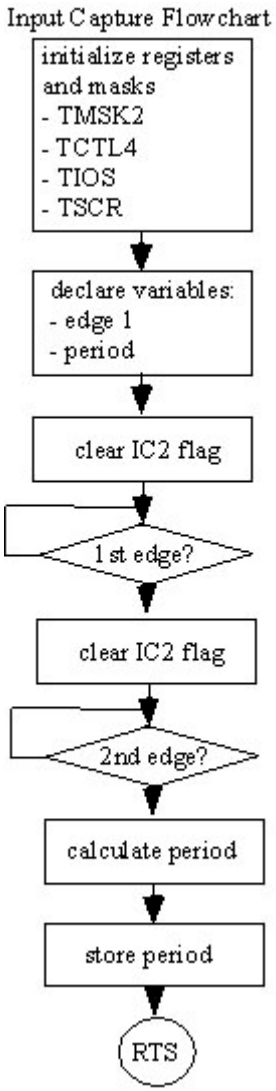
7	6	5	4	3	2	1	0
EDG3B	EDG3A	EDG2B	EDG2A	EDG1B	EDG1A	EDG0B	EDG0A
Reset: 0	0	0	0	0	0	0	0

Programming - Input Capture

EX] Measure the period of a periodic signal connected to input capture channel 2

- Assume MCLK is 8 MHz
- Set prescaler to divide by 4
- Clock frequency to frc is therefore 2 MHz
- Period: 0.5 μs

Programming - Input Capture

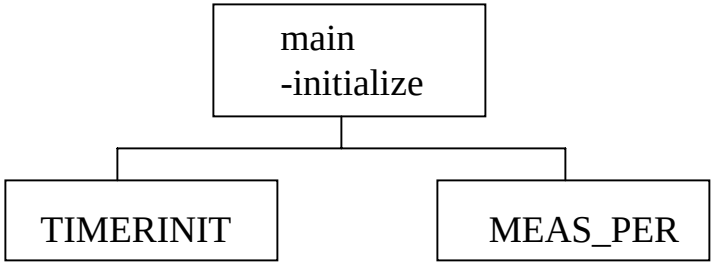


Programming - Input Capture

```
;MAIN PROGRAM:  This program measures the period of a signal connected to
;input capture channel 2 by measuring the count difference between two
;consecutive rising edges.

REG_BASE = $0000        ;addr of register base
TMSK1    = $8C          ;declare offset of reg from base
TMSK2    = $8D          ;declare offset of reg from base
TCTL4    = $8B          ;declare offset of reg from base
TIOS     = $80          ;declare offset of reg from base
TC2H     = $94          ;declare offset of reg from base
TSCR     = $86          ;declare offset of reg from base
TFLG1    = $8E          ;declare offset of reg from base
TCNT     = $84          ;declare offset of reg from base
TMSK2_IN = $02          ;disable TOI, prescale=4
TCTL4_IN = $10          ;config IC2 for rising edge
TIOS_IN  = $00          ;select ch2 for IC
TSCR_IN  = $80          ;enable timer, normal flag clr
CLR_CH2  = $04          ;mask to clr ch 2 flag
```

Programming - Input Capture




```
          .area test_1(abs)
          .ORG  $7000    ;User RAM at $7000
          edge_1 FDB     ;reserve word for variable
          period FDB     ;reserve word for variable


_main::
          BSR TIMERINIT  ;timer initialization subr
          BSR MEAS_PER   ;measure period
DONE:     BSA DONE       ;Branch to self
```

Programming - Input Capture

```
;-----------------------------------------------
;Subroutine TIMERINIT: Initialize timer for IC2;
;-----------------------------------------------

TIMERINIT: CLR       TMSK1          ;disable interrupts
           LDX       #REG_BASE      ;load register base to X
           LDAA      #TMSK2_IN      ;load TMSK2 using index addr
           STAA      TMSK2,X        ;disable ovf, prescale=4
           LDAA      #TCTL4_IN      ;conf IC2 for rising edge
           STAA      TCTL4,X        ;
           LDAA      #TIOS_IN       ;select ch 2 for IC
           STAA      TIOS,X         ;
           LDAA      #TSCR_IN       ;conf IC2 for rising edge
           STAA      TSCR_IN,X      ; enable timer, standard flag clr
           RTS                      ;Return from subroutine
```

Programming - Input Capture

```
;---------------------------------------
;Subroutine MEAS_PER: measures period
;between two input capture events.
;---------------------------------------

MEAS_PER: LDAA      #CLR_CH2              ;Clear IC2 flag
          STAA      TFLG1,X
WTFLG:    BRCLR     TFLG1,X,#$04,WTFLG    ;wait for first edge
          LDD       TCNT,X                ;load value from TCNT reg
          STD       EDGE_1                ;store rising edge 1 count
          LDAA      # CLR_CH2             ;Clear IC2 flag
          STAA      TFLG1,X
WTFLG1:   BRCLR     TFLG1,X,#$04,WTFLG1   ;wait for second edge
          LDD       TCNT,X                ;load value from TCNT
          SUBD      EDGE_1                ;calculate period
          STD       PERIOD                ;store period
          RTS
```

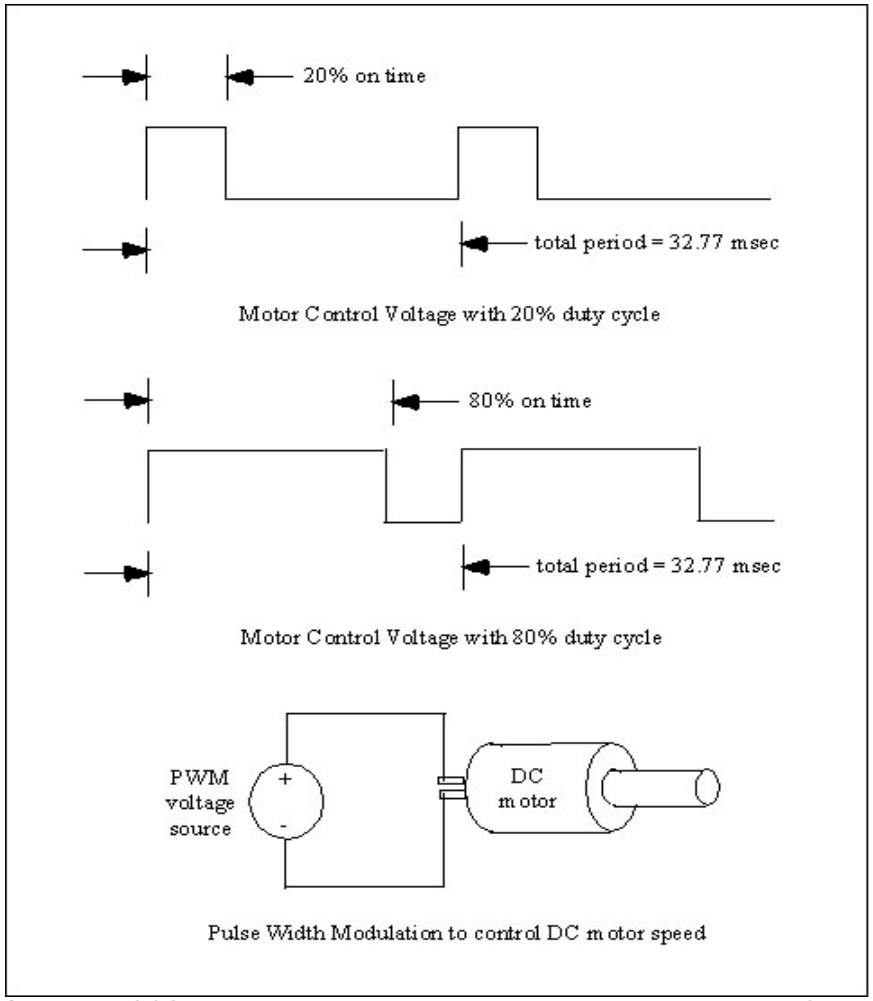
Output Compare - generate precision signals

- generate precision signals based on key events
  - active low or high pulse
  - repetitive signal of

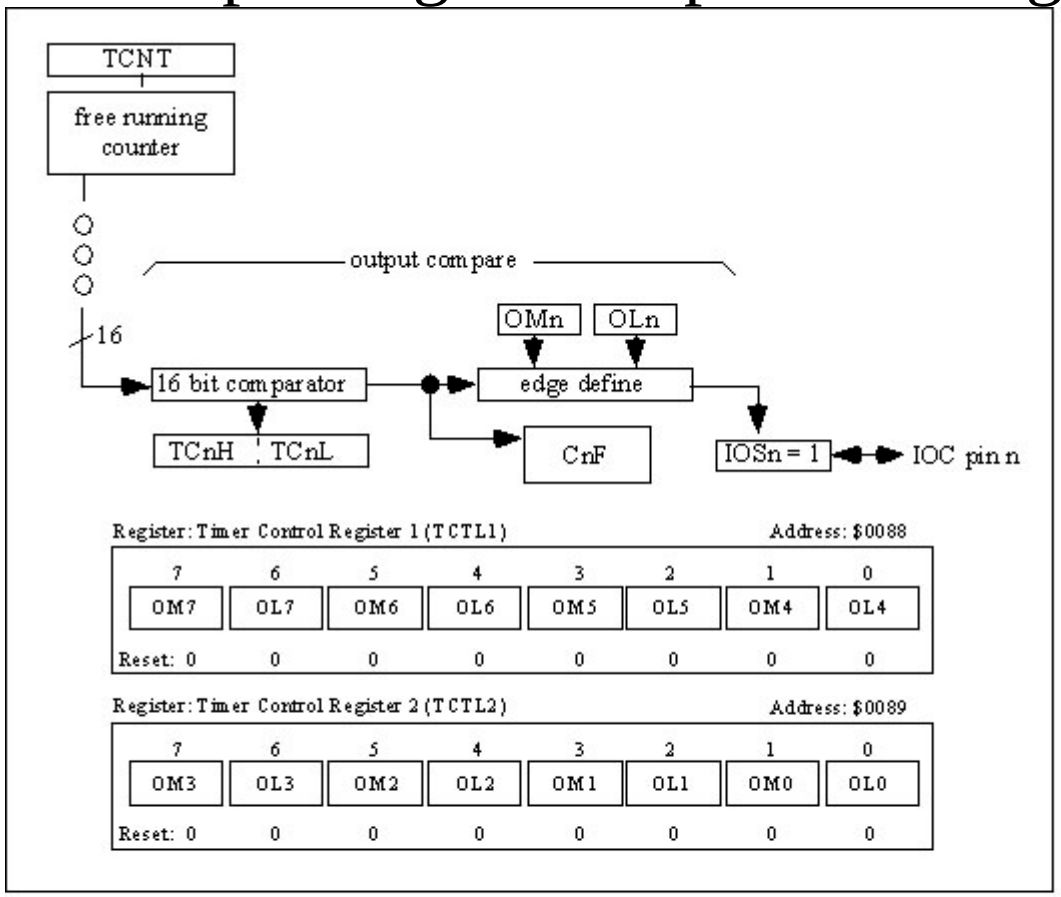
  desired frequency and duty cycle
  - pulse width modulation

- key events

OMn OLn	Configuration
00	Timer Disconnected
01	Toggle OCn output line
10	OCn output line to 0
11	OCn output line to 1



Standard Timer Module (TIM)
Output Compare - generate precision signals



Register: Timer Control Register 1 (TCTL1) Address: $0088

7	6	5	4	3	2	1	0
OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
Reset: 0	0	0	0	0	0	0	0

Register: Timer Control Register 2 (TCTL2) Address: $0089

7	6	5	4	3	2	1	0
OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset: 0	0	0	0	0	0	0	0

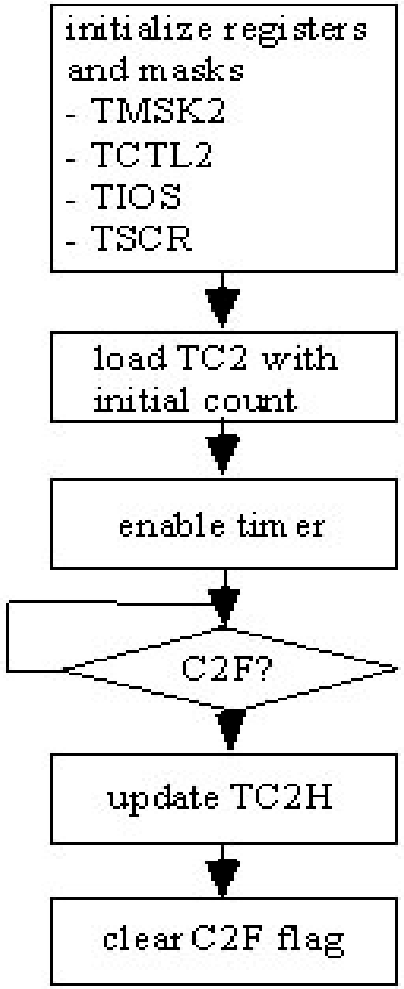
Programming - Output Compare

EX] Generate a 1000 Hz square wave with a 50% duty cycle on output compare channel 2 (OC2).

- Assume MCLK is 8 MHz
- Prescaler divide by 4
- frc clock frequency is 2 MHz, period 0.5 μs
- Therefore:
  - 1000 Hz period is 2000 pulses
  - High time: 1000 pulses ($3E8)
  - Low time: 1000 pulses ($3E8)

Programming - Output Compare

Output Compare Flowchart



Programming - Output Compare

```
;---------------------------------------
;MAIN PROGRAM: This program generates a
;1000 Hz square wave with a 50% duty cycle
;on output compare channel 2.
;---------------------------------------

TMSK1      =  $008C       ;declare locations of registers
TMSK2      =  $008D       ;declare locations of registers
TCTL2      =  $0089       ;declare locations of registers
TIOS       =  $0080       ;declare locations of registers
TC2H       =  $0094       ;declare locations of registers
TSCR       =  $0086       ;declare locations of registers
TFLG1      =  $008E       ;declare locations of registers
TMSK2_IN   =  $02         ;disable TOI, prescale=4
TCTL2_IN   =  $10         ;initialize OC2 toggle
TIOS_IN    =  $04         ;select ch2 for OC
TSCR_IN    =  $80         ;enable timer, normal flag clr
```

Programming - Output Compare

```
          .area test_1(abs)
         .ORG     $7000        ;User RAM at $7000

-main::
         BSR      TIMERINIT    ;timer initialization subr
DONE:    BSR      SQ_WAVE      ;square wave gen subr
         BRA      DONE         ;Branch to self

;-----------------------------------------------
;Subroutine TIMERINIT: Initialize timer for OC2;
;-----------------------------------------------

TIMERINIT: CLR     TMSK1              ;disable interrupts
           MOVB    #TMSK2_IN,TMSK2    ;disable ovf, prescale=4
           MOVB    #TCTL2_IN,TCTL2    ;OC2 toggle on compare
           MOVB    #TIOS_IN,TIOS      ;select ch 2 for OC
           MOVW    #$03E8,TC2H        ;load TC2 with initial comp
           MOVB    #TSCR_IN,TSCR      ;enable timer, standard flag clr4
           RTS                        ;Return from subroutine
```

Programming - Output Compare

```
;----------
: CLEARFLG
;----------
;Clear C2F flag by reading TFLG1 when C2F set and then writing 1 to C2F

CLEARFLG: LDAA    TFLG1      ;To clear OC2 flag, read flag first
          ORAA    #$04       ;then write ``1'' to it
          STAA    TFLG1
          RTS

;---------------------------------------
;Subroutine SQWAVE:                     ;
;---------------------------------------

SQ_WAVE: BRCLR    TFLG1,#$04,SQ_WAVE ;Poll for C2F Flag
         LDD      TC2H               ;load value from TC2 reg
         ADDD     #$03E8             ;add hex value high count
         STD      TC2H               ;set up next transition time
         BRA      CLEARFLG           ;generate repetitive signal
         RTS
```

Programming - Output Compare

```
/* File Name: outcomp.c
 * File Created: 04/20/02
 * File Modified:
 * Author(s): Abbie Wells
 */

/* This file will generate a precision square wave.  It will have a
 * frequency of 519 Hz and a duty cycle of 77%.  It will use output
 * compare channel two (OC2) to generate the signal.
 */

 #include <hc12.h>

 void timer_init(void);
 void half_cycle(unsigned int time);
```

Programming - Output Compare

```
void main(void){
  unsigned int high_time = 2968; // number of clock cycles for signal to
                                 // be high
  unsigned int low_time = 886;   // number of clock cycles for signal to
                                // be low

   timer_init();                // initialize timer and registers
   half_cycle(low_time);

   while(1)
   {
      half_cycle(high_time);         // generate high portion of signal
      half_cycle(low_time);          // generate low portion of signal
   }
 }
```

Programming - Output Compare

```
/* Function: timer_init This function will initialize the timer to use
   output compare channel two (OC2) and toggle. */

 void timer_init(void){
   CLKCTL = 0x02;                          // Set MCLK to 2 MHz
                                           // Sets CPU clock ($0047)
   TMSK1 = 0x00;                           // Disable interrupts
   TMSK2 = 0x00;
   TIOS = 0x04;                            // Set Ch 2 to output comp
   TSCR = 0x80;                            // Enable the timer
   TCTL2 = 0x10;
   TFLG1 = 0x04;                           // Clear all flags
   TC2 = TCNT;
 }
```

Programming - Output Compare

```
/* Function: half_cycle This function will generate either the high or
   low portion of the output compare wave.  It needs to have the
   desired time for the portion of the wave passed to it.  */

 void half_cycle(unsigned int time){
   TC2 += time;                    // Update timer register
   while((TFLG1 && 0x04)==0)
   {
      ;
   }
   TFLG1 = 0x04;                  // Clear flag
 }
```