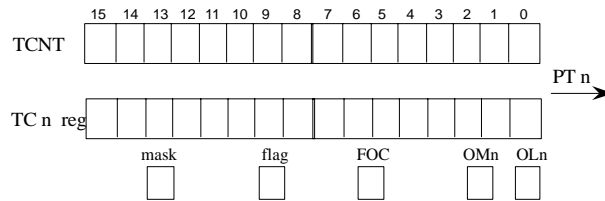


6. Timing Generation and Measurements

Chapter 6 objectives are to use:

- Input capture to generate edge based interrupts;
- **Input capture to measure period;**
- Input capture to measure pulse width;
- Output compare to create periodic interrupts;
- **Output compare to generate square waves;**
- Both input capture and output compare measure frequency, phase, long periods.

Output compare



When does an output compare happen

TCNT equals TCn

What action occurs

sets a flag **CnF**

optional (**CnI**): requests an interrupt

optional (**OMn OLn**): changes output pin **PTn**

Acknowledge interrupt

write a one to the **CnF** flag in **TFLG1**

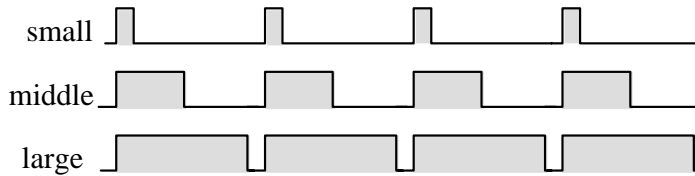
TIOS	7	6	5	4	3	2	1	0	\$0040
	IOS7	IOS6	IOS5	IOS4	IOS3	IOS2	IOS1	IOS0	
TSCR1	7	6	5	4	3	2	1	0	\$0046
	TEN	TSWAI	TSBCK	TFFCA	0	0	0	0	
TSCR2	TOI	0	0	0	0	PR2	PR1	PR0	\$004D
TIE	7	6	5	4	3	2	1	0	\$004C
	C7I	C6I	C5I	C4I	C3I	C2I	C1I	C0I	
TFLG1	C7F	C6F	C5F	C4F	C3F	C2F	C1F	C0F	\$004E
TFLG2	TOF	0	0	0	0	0	0	0	\$004F

	7	6	5	4	3	2	1	0	
TCTL1	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4	\$0048
TCTL2	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0	\$0049

OMn	OLn	Effect of when TCn=TCNT
0	0	does not affect PTn
0	1	Toggle PTn
1	0	Clear PTn=0
1	1	Set PTn=1

Table 6.12. The action caused by 6812 output compare event.

6.2.5. Pulse Width Modulation



High + Low will always equal 10000

Duty cycle is High/10000

This implementation can not generate waves close to 0 or 100% duty cycle. If **T** is the maximum number of cycles to process the output compare interrupt, then both **High** and **Low** must be greater than **T**.

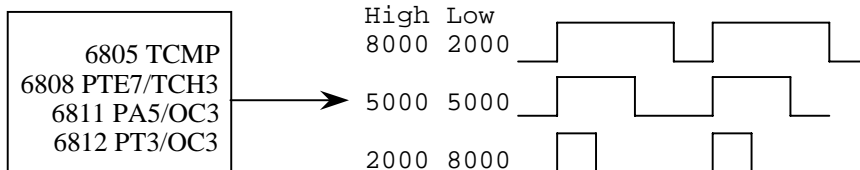


Figure 6.24. Pulse width modulation using output compare.

```
// MC9S12C32
unsigned short Period; // in usec
unsigned short High; // Num of usec High
unsigned short Low; // Num of usec Low
// Period is High+Low Cycles
interrupt 8 void TC0handler(void){
    TFLG1 = 0x01; // ack C0F
    if(PTT&0x01){ // PT0 is now high
        TC0 = TC0+High; // 1 for High usec
    }
    else{ // PT0 is now low
        TC0 = TC0+Low; // 0 for Low usec
    }
}
```

```

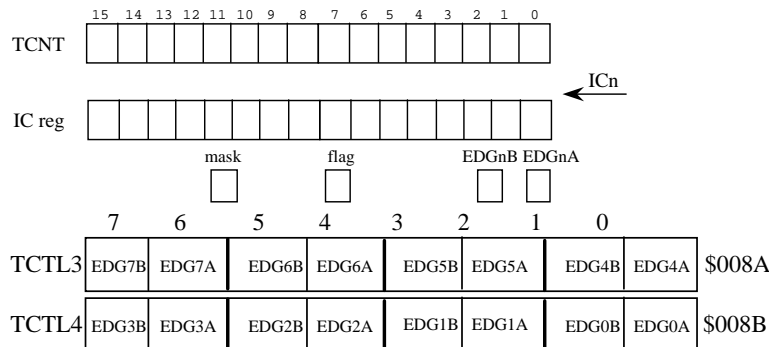
}
void PWM_Init(unsigned short period){
    asm sei // make atomic
    Period = period;
    High = Low = period/2;
    TIOS|=0x01; // enable channel 0
    DDRT|=0x01; // PT0 is output
    TSCR1=0x80; // enable
    TSCR2=0x02; // 1us clock
    TIE |=0x01; // Arm output compare 0
    TFLG1=0x01; // Initially clear C0F
    TCTL2 = (TCTL2&0xFC)|0x01 ; // toggle
    TC0 = TCNT+50; // first right away
    asm cli
}
// duty cycle varies from 50 to Period-50
void PWM_Duty(unsigned short duty){
    if(duty<50) return; // too small
    if(duty>(Period-50)) return; // too big
    asm sei // make atomic
    High = duty;
    Low = Period-High;
    asm cli
}

void main(void){
    PWM_Init(10000); // 10ms
    PWM_Duty(8000); // 80%
    while(1);
}

```

****Run Metrowerks PWM project*****

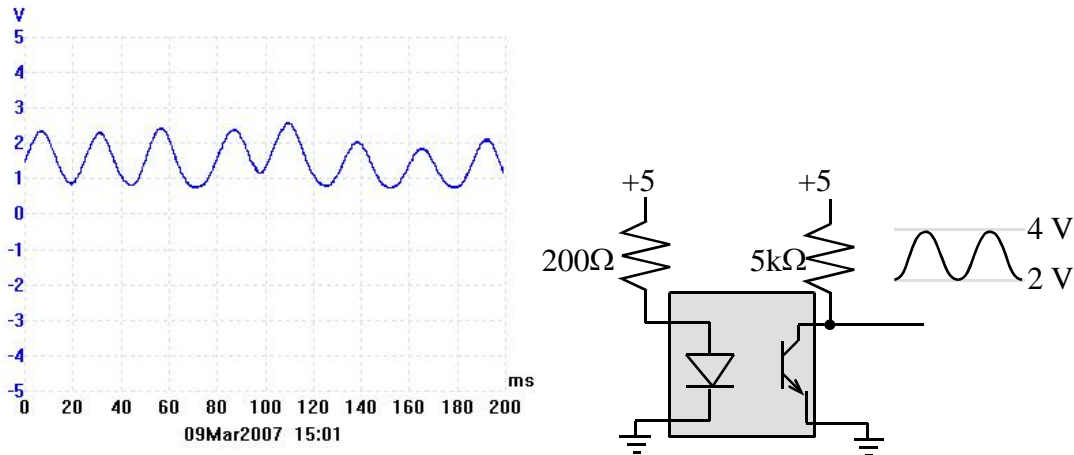
16 bit Period Measurement with a 1us resolution



EDGnB	EDGnA	active edge
0	0	none
0	1	capture on rising

1	0	capture on falling
1	1	capture on both rising and falling

Table 6.6. Two control bits define the active edge used for input capture.



Tachometer data collected with a 15-slot disk powered at +9V. IR sensor interface on left

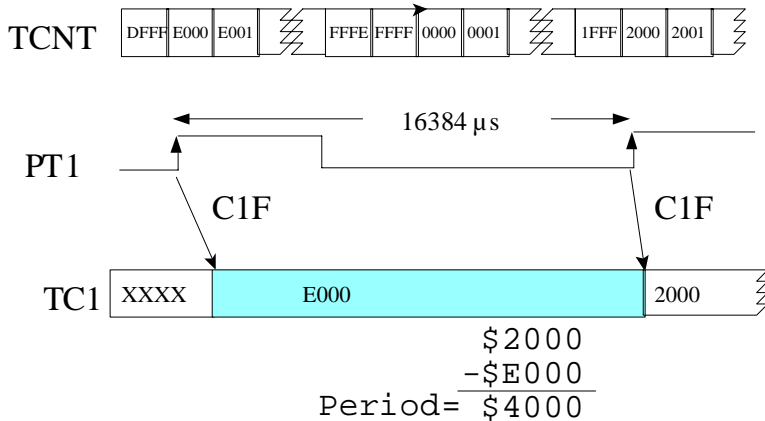


Figure 6.8. Timing example showing counter rollover during 16 bit period measurement.

```
// PT1/IC1 input = external signal
// rising edge to rising edge
// resolution = 1us
// Range = 36 μs to 65 ms,
// no overflow checking
// IC1 interrupt each period,
void Period_Init(void){
    asm sei          // make atomic
    TIOS &= ~0x02;  // PT1 input capture
    DDRT &= ~0x02;  // PT1 is input
    TSCR1 = 0x80;   // enable TCNT
    TSCR2= 0x02;    // 1us clock
```

```
TCTL4 = (TCTL4&0xF3)|0x04; // rising
TFLG1 = 0x02; // Clear C1F
TIE |= 0x02; // Arm IC1
asm cli
}
unsigned short Period; // units of 1us
interrupt 9 void TC1handler(void){
unsigned short static First;
    Period = TC1-First;
    First = TC1; // Setup for next
    TFLG1 = 0x02; // ack by clearing C1F
}
```

Program 6.6. C language period measurement.

****Run with TExaS ****