Computer interfacing (CI)

7. The CCP Unit

7.1 CCP Units

A PIC18 device may have several CCP modules. PIC18F45K22 has **5 CCP** Modules.

- CCP stands for **Capture**, **Compare**, and Pulse Width Modulation (**PWM**).
- Each CCP unit can be configured to perform one of these functions (one at a time).
- CCP functions require the use of one Timer resource; Timers 1, 3 or 5 for Capture and Compare, Timers 2, 4 or 6 for PWM generation.

- The operation of every CCP module is controlled by the registers (x={1..5}) :
  * CCPxCON register (to select the mode),
  * CCPTMRS0 and CCPTMRS1 registers (to select the related operating timer)
  * and a 16-bits data register CCPRx (CCPRxH and CCPRxL).

- A device pin (CCPx pin) can be associated to CCP module operation (with appropriate TRIS configuration).

7.1 CCP Units

CCPxCON Register:

Bits 3-0 allow the selection of the desired CCP function.

REGISTER 14-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB<1:0>		CCPxM<3:0>			
bit 7							bit 0

bit 7-6 **Unused**

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle Least Significant bits

Capture mode:
Unused

Compare mode:
Unused

PWM mode:
These bits are the two LSbs of the PWM duty cycle. The eight MSbs are found in CCPRxL.

bit 3-0 **CCPxM<3:0>:** ECCPx Mode Select bits

0000 = Capture/Compare/PWM off (resets the module)
0001 = Reserved
0010 = Compare mode: toggle output on match
0011 = Reserved

0100 = Capture mode: every falling edge
0101 = Capture mode: every rising edge
0110 = Capture mode: every 4th rising edge
0111 = Capture mode: every 16th rising edge

1000 = Compare mode: set output on compare match (CCPx pin is set, CCPxIF is set)
1001 = Compare mode: clear output on compare match (CCPx pin is cleared, CCPxIF is set)
1010 = Compare mode: generate software interrupt on compare match (CCPx pin is unaffected, CCPxIF is set)
1011 = Compare mode: Special Event Trigger (CCPx pin is unaffected, CCPxIF is set)
TimerX (selected by CxTSEL bits) is reset
ADON is set, starting A/D conversion if A/D module is enabled(1)
11xx =: PWM mode

Note 1: This feature is available on CCP5 only.

7.1 CCP Units

CCPTMRS0/1 Registers.

Allow the selection of a Timer for the desired CCP function.

REGISTER 14-3: CCPTMRS0: PWM TIMER SELECTION CONTROL REGISTER 0

R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
C3TSEL<1:0>		—	C2TSEL<1:0>		—	C1TSEL<1:0>	
bit 7							bit 0

bit 7-6 **C3TSEL<1:0>:** CCP3 Timer Selection bits
00 = CCP3 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP3 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP3 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 5 **Unused**

bit 4-3 **C2TSEL<1:0>:** CCP2 Timer Selection bits
00 = CCP2 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP2 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP2 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

bit 2 **Unused**

bit 1-0 **C1TSEL<1:0>:** CCP1 Timer Selection bits
00 = CCP1 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP1 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP1 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

REGISTER 14-4: CCPTMRS1: PWM TIMER SELECTION CONTROL REGISTER 1

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	C5TSEL<1:0>		C4TSEL<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-4 **Unimplemented:** Read as '0'

bit 3-2 **C5TSEL<1:0>:** CCP5 Timer Selection bits
00 = CCP5 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP5 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP5 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

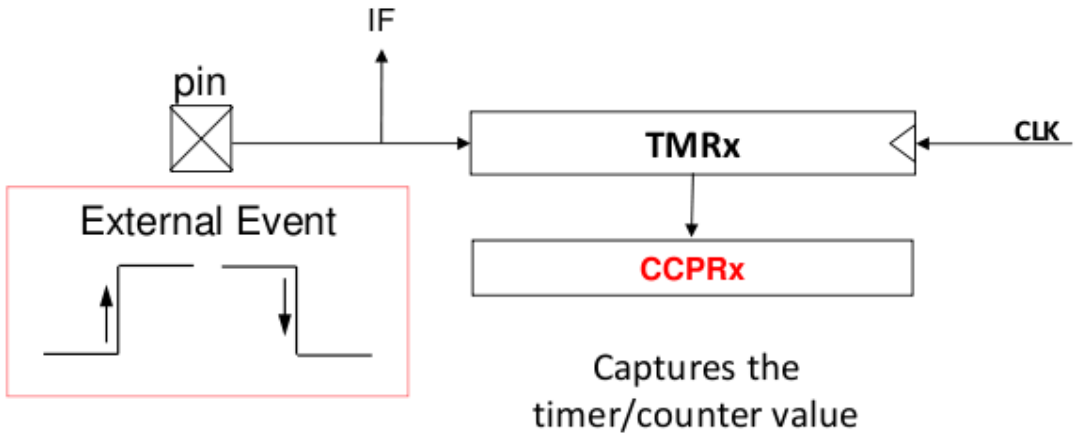
bit 1-0 **C4TSEL<1:0>:** CCP4 Timer Selection bits
00 = CCP4 – Capture/Compare modes use Timer1, PWM modes use Timer2
01 = CCP4 – Capture/Compare modes use Timer3, PWM modes use Timer4
10 = CCP4 – Capture/Compare modes use Timer5, PWM modes use Timer6
11 = Reserved

7.2 Capture function

In capture operation, the CCP module copy the contents of a 16 bit timer into a capture register on a signal edge.

- Main idea: capture event arrival time.
- An event is represented by a signal edge on a pin.

- The PIC18 allow the capture of:
  - every falling edge
  - every rising edge
  - every 4th rising edge
  - every 16th rising edge



Gives a reference on the instant in which the event takes place
(Application: measure the period of a given signal)

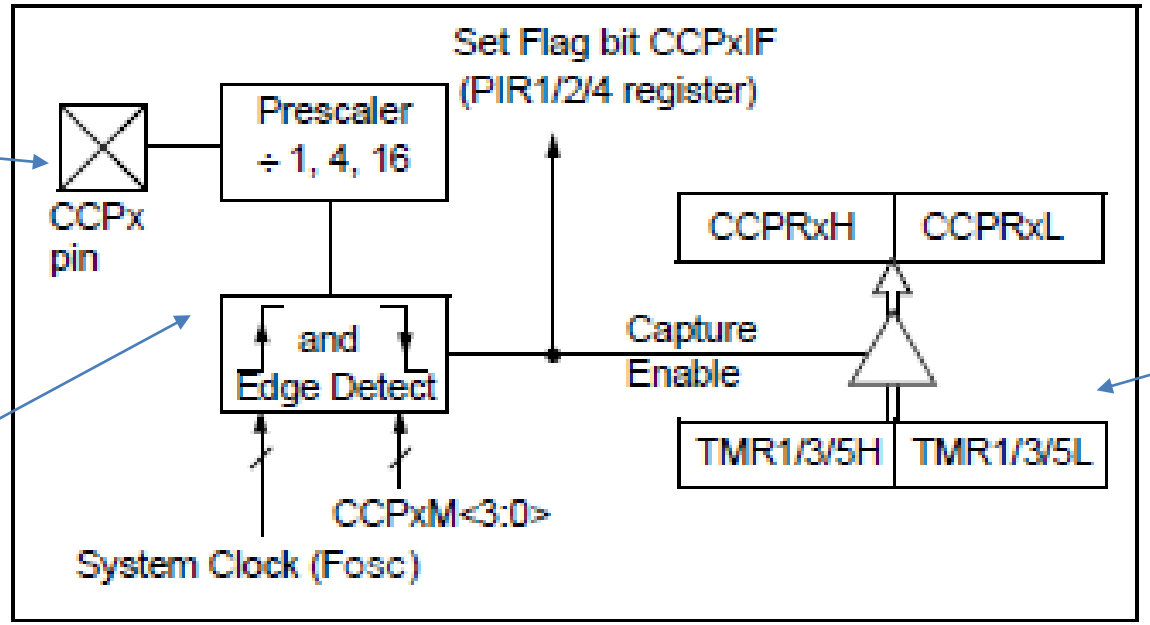
7.2 Capture function

PIC 18 Capture unit diagram.

If desired, CCPxIF must be configured

Pin must be configured as input.

CCPxCON selects edge and dividers



TimerX must be active and configured (clock source, prescaler…)

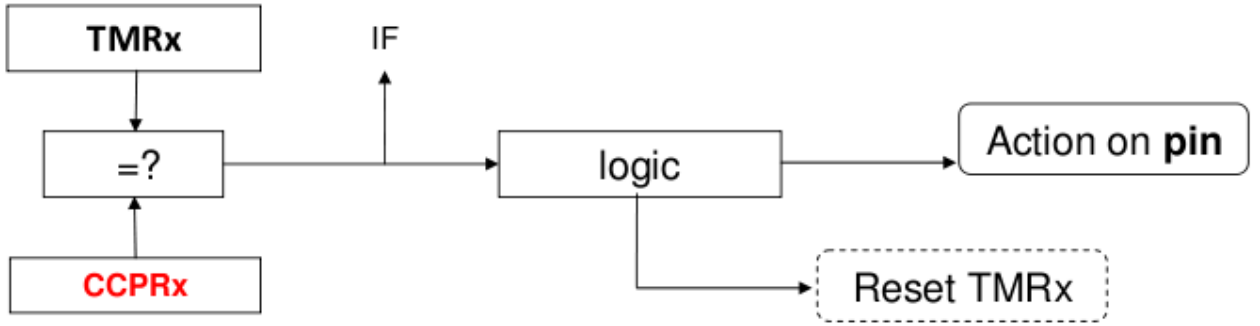
Remember: the five units can operate independently.

7.3 Compare function

In compare operation, the CCP module compares the contents of a CCPR register with that of a Timer (1, 3 or 5) in every clock cycle. When these two registers are equal, the associated pin may be pulled to **high**, or **low**, or **toggled**.

- Action on pin is selected at CCPxCON register.

- Toggle mode is used to create a periodic signal.
- High or Low pin action can be used to set a delayed shot.
- Software interrupt mode (1010) does not modify the pin and can be used to create a delay function.



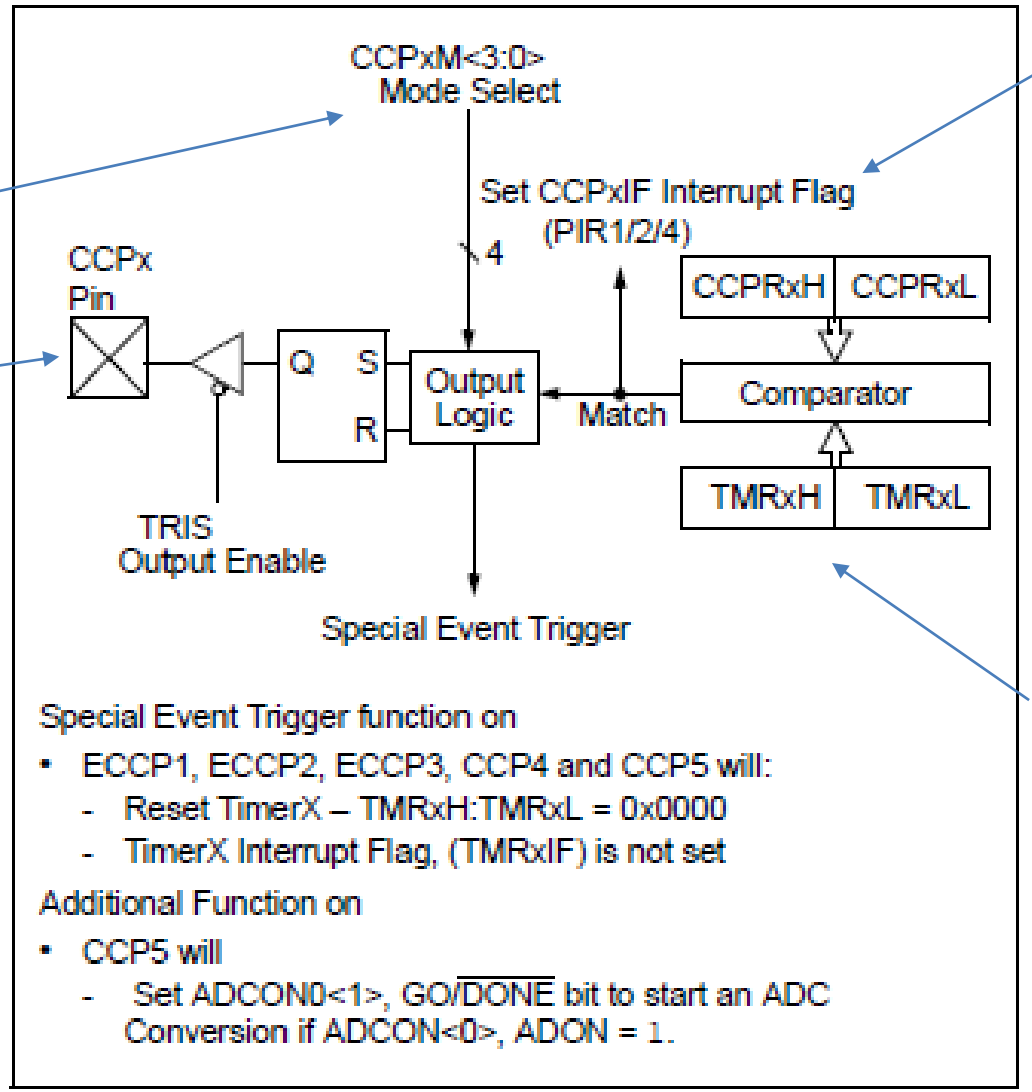
Generates equally spaced time intervals
(Application: Delays, Pulse Trains)

7.3 Compare function

PIC 18 Compare unit diagram.

CCPxCON selects mode

Pin must be configured as output.



If desired, CCPxIF must be configured

TimerX must be active and configured (clock source, prescaler…)

- Special Event Trigger mode (1011) does not modify the pin, resets the associated Timer and additionally can start an AD conversion (CCP5 only).

7.4 PWM function

In PWM operation, the CCP module generates continuously a periodic waveform, where the percentage of '1' (on) and '0' (off) can be configured.

Definitions:

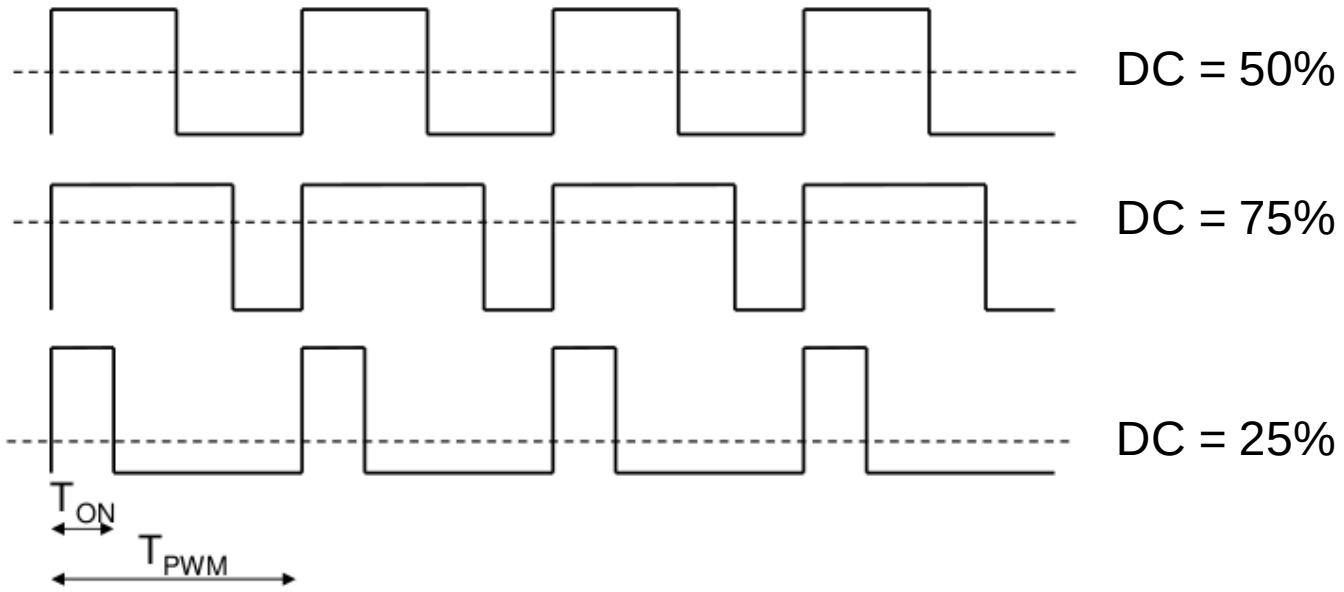
Period = T_{PWM}

Frequency, $F_{PWM} = 1 / T_{PWM}$

Duty Cycle (DC) = $(T_{ON} / T_{PWM}) \times 100$

Period and DC are the two main parameters of a PWM signal.

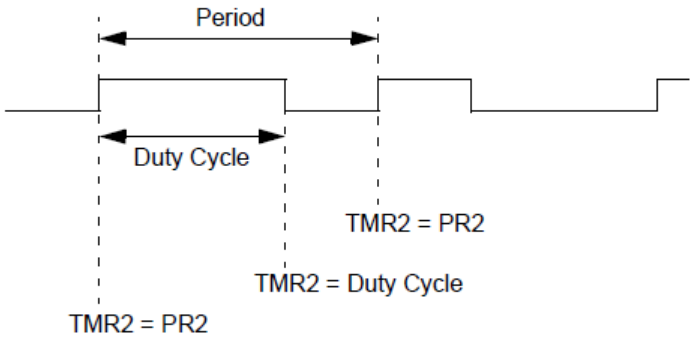
Examples:



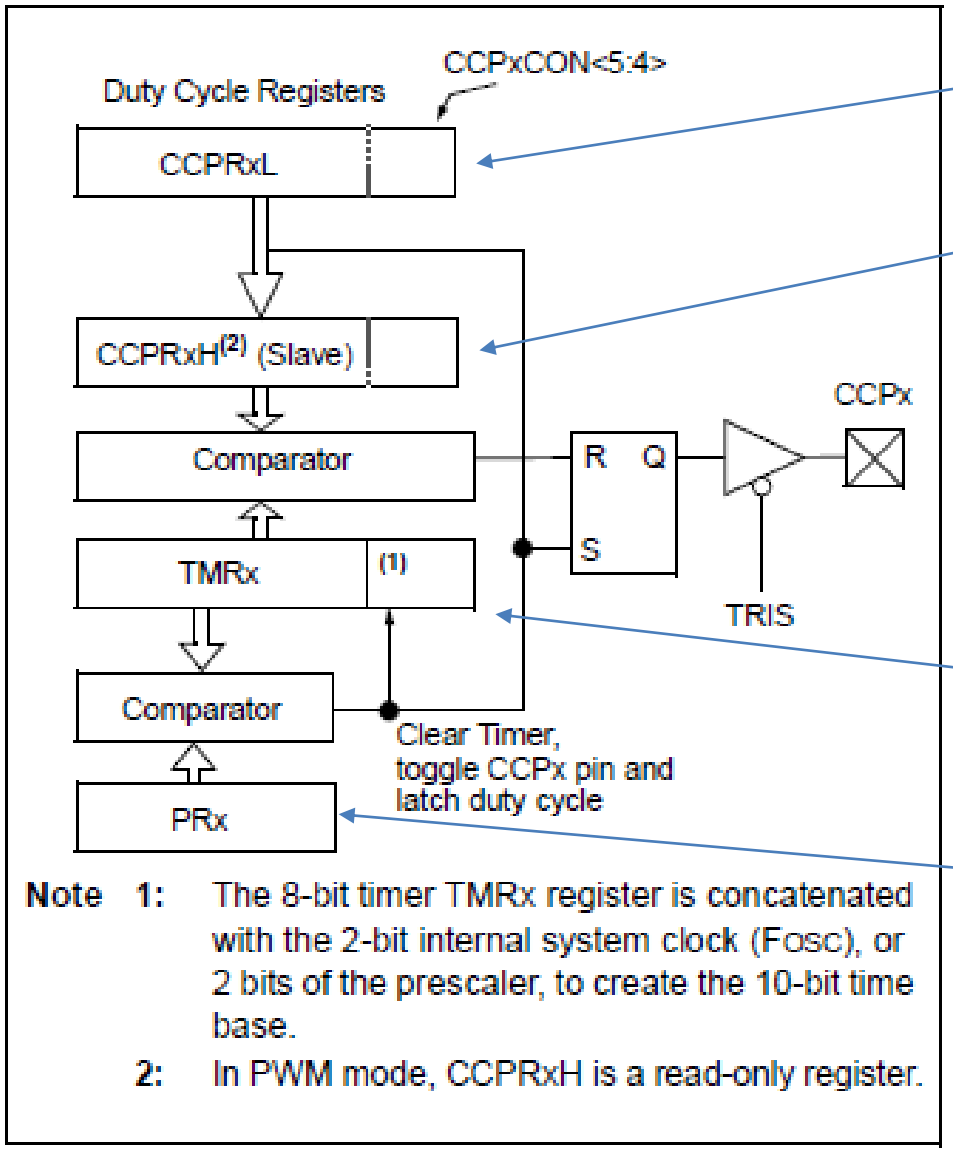
7.4 PWM function

PIC 18 PWM unit diagram.

- The Set/Reset Flip-Flop will drive the CCPx pin.
- There is a 'Set' at the beginning of every period and a 'Reset' when DC is reached.



Example with TMR2 and PRs



Duty Cycle has 10 bit resolution

Duty Cycle can be refreshed only at the end of the period.

Internally, TMRx works with 10 bit resolution

Period can only be defined by 8 bits (the MSB of 10).

7.4 PWM function

PIC 18 PWM programming recipe (provided in the Datasheet):

1. Disable the CCPx pin output driver by setting the associated TRIS bit.
2. Select the 8-bit TimerX resource, (Timer2, Timer4 or Timer6) to be used for PWM generation by setting the CxTSEL<1:0> bits in the CCPTMRSx register.
3. Load the PRx register for the selected TimerX with the PWM period valu
4. Configure the CCP module for the PWM mode by loading the CCPxCON register with appropriate values.
5. Load the CCPRxL register and the DCxB<1:0> bits of the CCPxCON register, with the PWM duty cycle value.
6. Configure and start the 8-bit TimerX resource:
   - Clear the TMRxIF interrupt flag bit of the PIR2 or PIR4 register.
   - Configure the TxCKPS bits of the TxCON register with the Timer prescale value.
   - Enable the Timer by setting the TMRxON bit of the TxCON register.
7. Enable PWM output pin:
   - Wait until the Timer overflows and the TMRxIF bit of the PIR2 or PIR4 register is set.
   - Enable the CCPx pin output driver by clearing the associated TRIS bit.

7.4 PWM function

PIC 18 PWM operation formulas (provided in the Datasheet):

EQUATION 14-1: PWM PERIOD

$$PWM\ Period = [(PRx) + 1] \bullet 4 \bullet Tosc \bullet (TMRx\ Prescale\ Value)$$

Note 1: Tosc = 1/Fosc

EQUATION 14-2: PULSE WIDTH

$$Pulse\ Width = (CCPRxL{:}CCPxCON\langle 5{:}4\rangle) \bullet Tosc \bullet (TMRx\ Prescale\ Value)$$

EQUATION 14-3: DUTY CYCLE RATIO

$$Duty\ Cycle\ Ratio = \frac{(CCPRxL{:}CCPxCON\langle 5{:}4\rangle)}{4(PRx + 1)}$$

EQUATION 14-4: PWM RESOLUTION

$$Resolution = \frac{\log[4(PRx + 1)]}{\log(2)}\ bits$$

TABLE 14-7: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 32 MHz)

PWM Frequency	1.95 kHz	7.81 kHz	31.25 kHz	125 kHz	250 kHz	333.3 kHz
Timer Prescale (1, 4, 16)	16	4	1	1	1	1
PRx Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 14-8: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 20 MHz)

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale (1, 4, 16)	16	4	1	1	1	1
PRx Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

TABLE 14-9: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (FOSC = 8 MHz)

PWM Frequency	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale (1, 4, 16)	16	4	1	1	1	1
PRx Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

7.5 CCP use examples

(1) PWM for a light dimmer

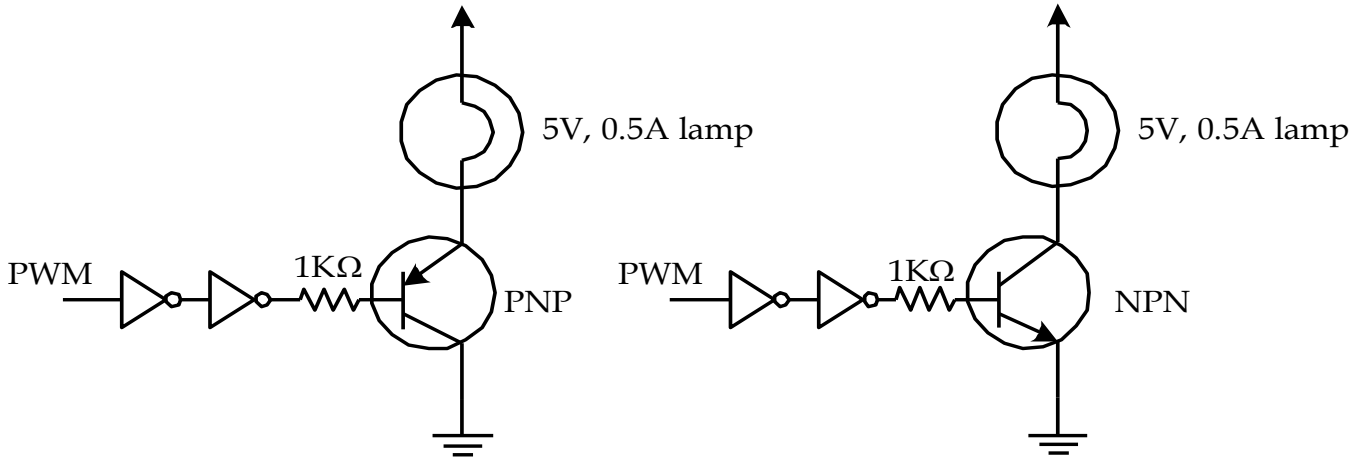


Figure 8.25 Using PWM to control the brightness of a light bulb

DC will be related with the brightness of the bulb.

PWM frequency must be high enough to avoid flickering.

The transistor allow the pass of enough current to light the bulb.

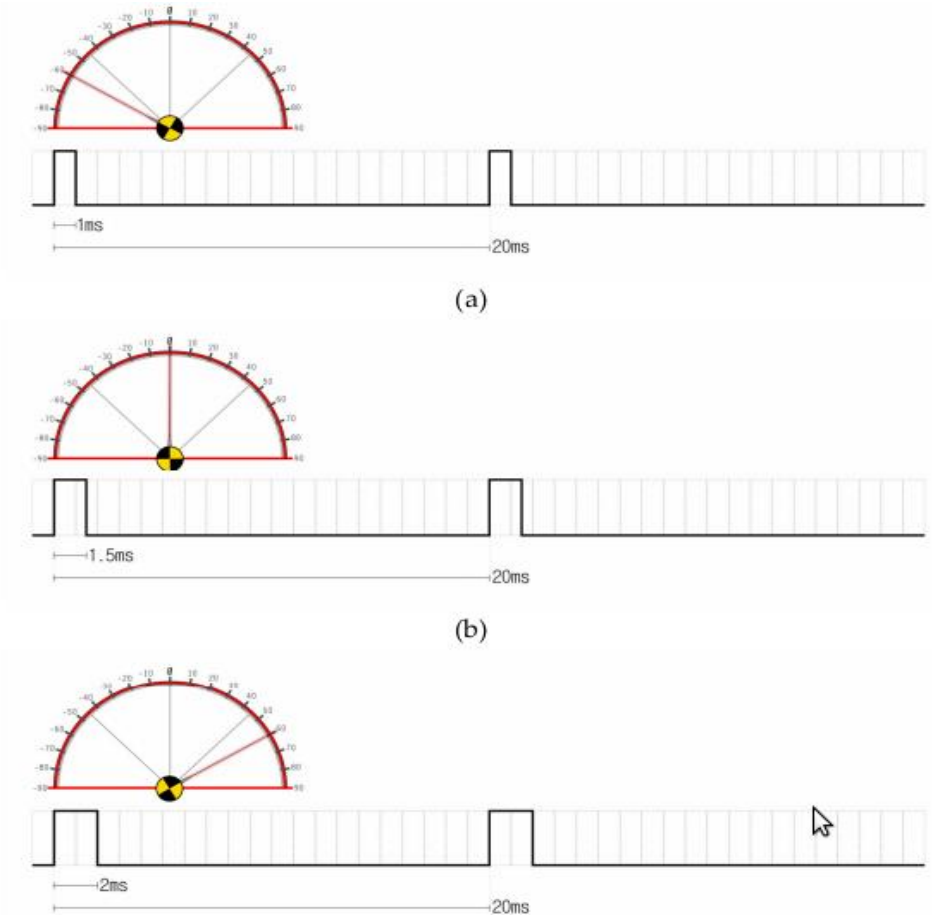
7.5 CCP use examples

(2) PWM for a servomotor

Servomotors allow us to control the angle of a device (e.g. air conditioner' split flap).

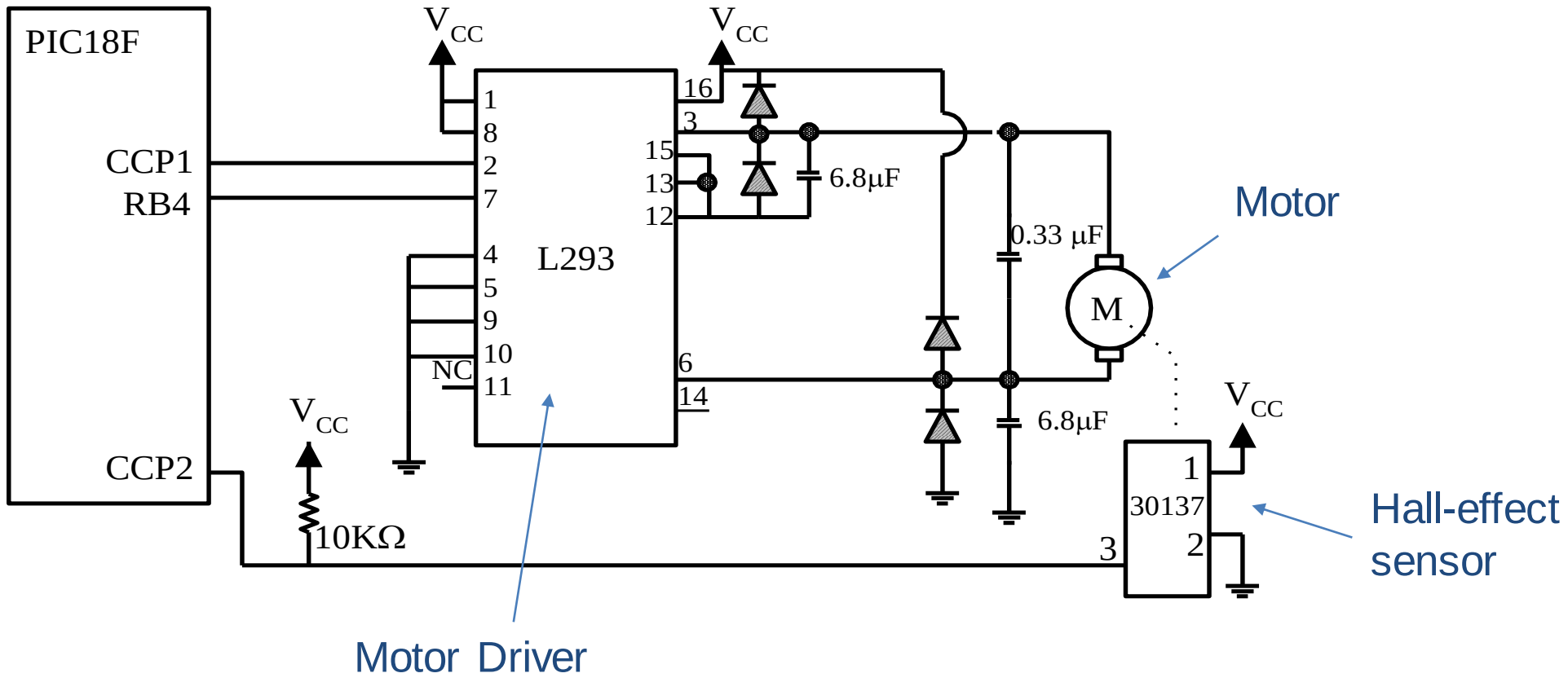
Rotation angle is proportional to the DC provided.

Servomotor's datasheet specifies the PWM frequency.



7.5 CCP use examples

(3.1) PWM and Capture for motor speed control system.



7.5 CCP use examples

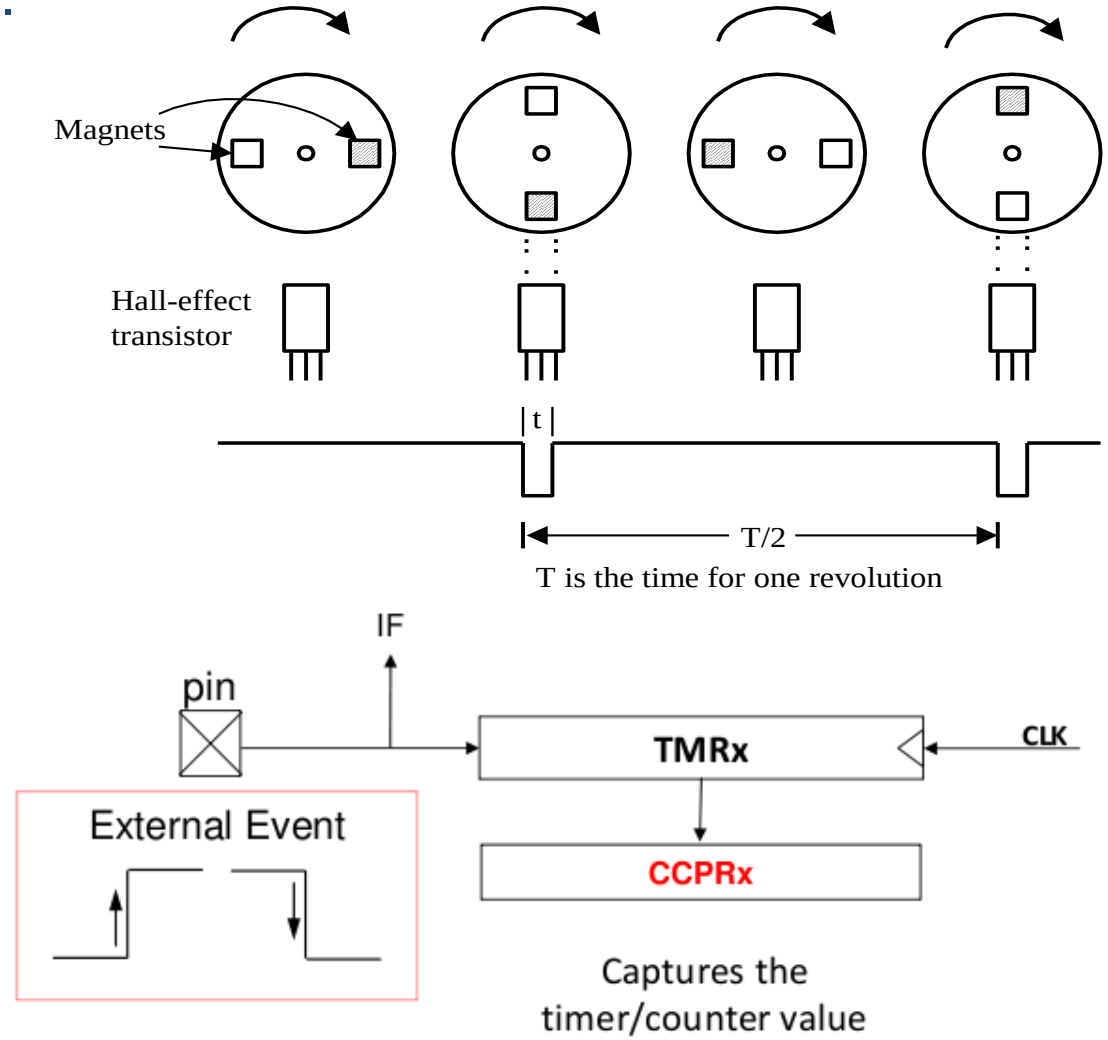
(3.2) PWM and Capture for motor control system.

Hall-effect sensors generate a pulse when a magnet is near.

Having two magnets in the wheel, will generate two pulses per revolution.

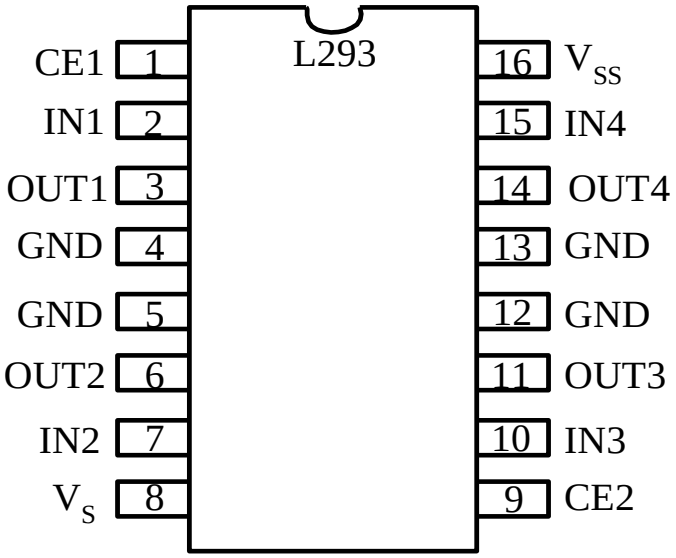
CCP's Capture unit can sense the pulses, allowing us to calculate the speed:

```
if (CCPRxIF ==1) && (CCPRxIE == 1)
{
      Half_Time = CCPRx:
      TMRx=0;
      Speed = Radius*PI / Half_Time;
}
```

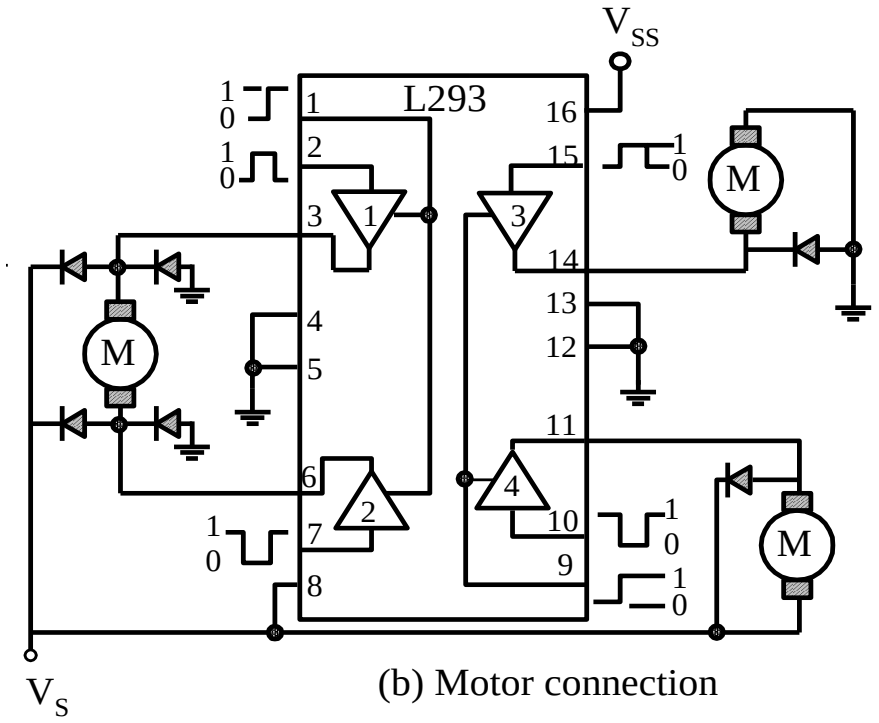


7.5 CCP use examples

(3.3) PWM and Capture for motor speed control system.



(a) Pin Assignment



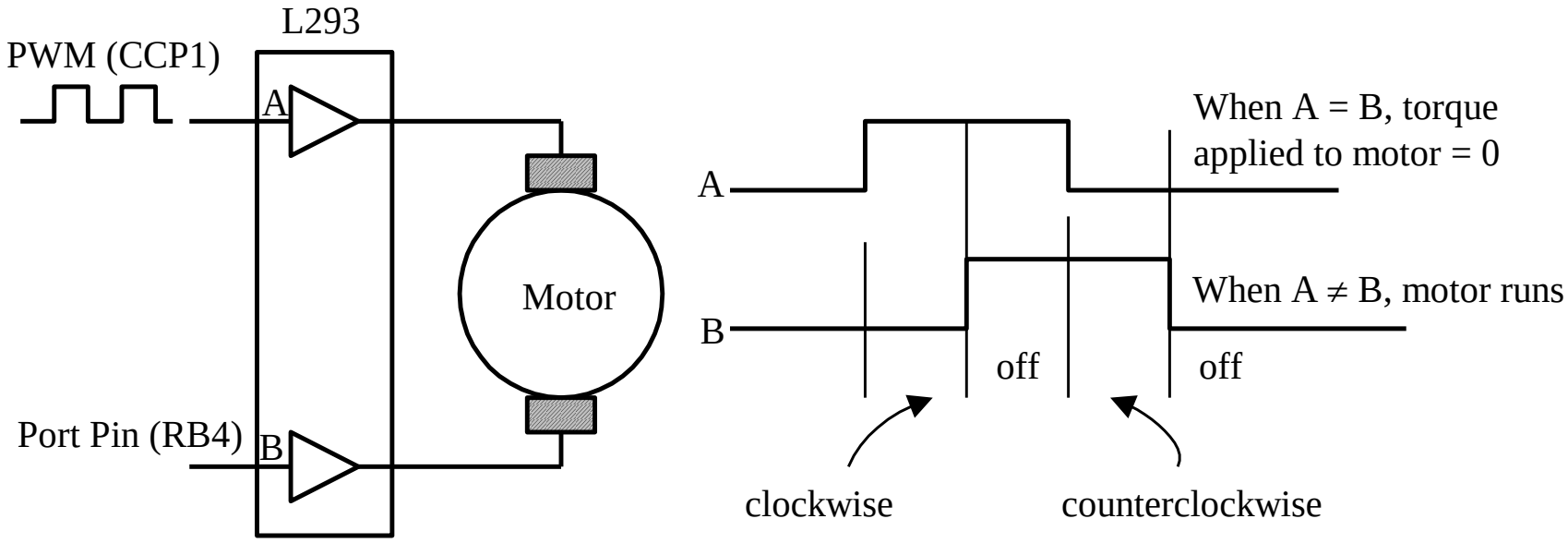
(b) Motor connection

L293 is a power device that allows the connection of up to 4 motors:

4 unidirectional motors (as connected in figure b, right), or 2 bidirectional motors (as connected in figure b, left).

7.5 CCP use examples

(3.4) PWM and Capture for motor speed control system.



Combination of two pins: CCP's PWM output for speed (CCP1), and RB4 for direction, allows us to drive the motor as desired.

Data obtained by the Hall-sensor (3.1 Speed), can be used to adjust PWM DC and reach the desired speed.

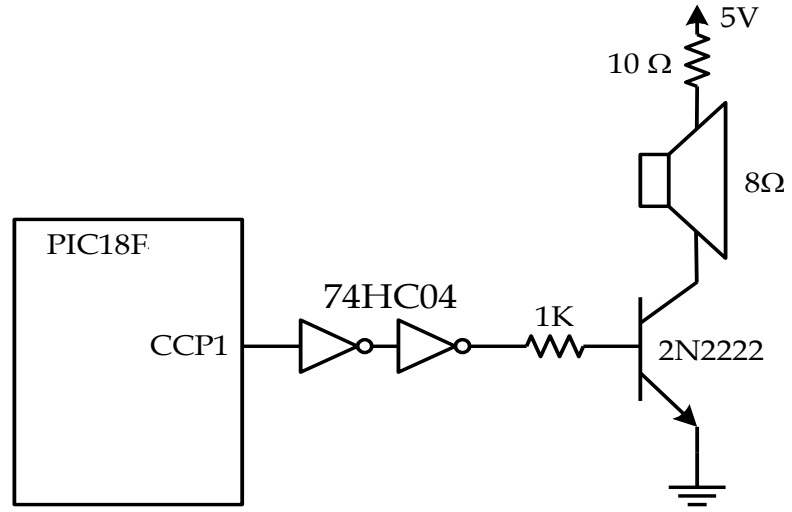
7.5 CCP use examples

(4.1) CCP's Compare unit to generate melodies.

Compare unit, in toggle mode, generates periodic symmetric waveforms.

Note	Octave 0	Octave 1	Octave 2	Octave 3	**Octave 4**	Octave 5	Octave 6	Octave 7
C	16.351	32.703	65.406	130.813	**261.626**	523.251	1046.502	2093.005
C#,Db	17.324	34.646	69.296	138.591	**277.183**	554.365	1100.731	2217.461
D	18.354	36.708	73.416	146.832	**293.665**	587.330	1174.659	2349.318
D#,Eb	19.445	38.891	77.782	155.563	**311.127**	622.254	1244.508	2489.016
E	20.061	41.203	82.407	164.814	**329.626**	659.255	1318.510	2367.021
F	21.827	43.654	87.307	174.614	**349.228**	698.456	1396.913	2637.021
F#,Gb	23.124	46.249	92.449	184.997	**369.994**	739.989	1474.978	2959.955
G	24.499	48.999	97.999	195.998	**391.995**	783.991	1567.982	3135.964
G#,Ab	25.956	51.913	103.826	207.652	**415.305**	830.609	1661.219	3322.438
A	**27.500**	**55.000**	**110.000**	**220.000**	**440.000**	**880.000**	**1760.000**	**3520.000**
A#,Bb	29.135	58.270	116.541	233.082	**466.164**	932.326	1664.655	3729.310
B	30.868	61.735	123.471	246.942	**493.883**	987.767	1975.533	3951.066

Table showing frequencies (Hz) per note. Focus on the 4th octave.



Amplification circuit to drive a speaker from a microcontroller pin.

7.5 CCP use examples

(4.2) CCP's Compare unit to generate melodies.

Example code, for a PIC18 with 4MHz clock.

```
#include  <xc.h>
#define   base      3125      // counter count to create 0.1 s delay
#define   NOTES     38        // total notes in the song to be played
#define   C4   0x777          // semi period
#define   F4   0x598          // for each note frequency
#define   G4   0x4FC          // Remember: toggle mode uses two interrupts per cycle
#define   A4   0x470
#define   B4   0x3F4
#define   C5   0x3BC
#define   D5   0x353
#define   F5   0x2CC

// Melody definition as an array of notes and an array of durations.

const unsigned int per_arr[38]={ C4,A4,G4,A4,F4,C4,C4,C5,B4,C5,A4,A4,F4,D5,D5,D5,C5,A4,C5,C5,B4,A4,B4,C5,
                                 A4,F4,D5,F5,D5,C5,A4,C5, C5,B4,A4,B4,C5,A4};
const unsigned char wait[38] = { 3,5,3,3,5,3,3,5,3,3,5,3,3,5,3,3,5,3,3,3,2,2,3,3,6,3,5,3,3,5,3,3,3,2,2,3,3,6};
```

7.5 CCP use examples

(4.3) CCP's Compare unit to generate melodies.

Example code, for a PIC18 with 4MHz clock (continued).

```
unsigned int half_cycle;                    // Global variable main->RSI

void  delay (unsigned char d)
{                                           // parameter d will be multiplied by 0,1 seconds equivalent
      T0CON = 0x84;                         // set prescaler to Timer0 to 32
      TMR0 = 0xFFFF - d*base;               // set TMR0 to this value so it overflows in d* 0,1 second */
      INTCONbits.TMR0IF = 0;
      while (! INTCONbits.TMR0IF);
}
void interrupt high_ISR(void)               // Every time is called, frequency is actualized (if needed)
{
      if (PIE1bits.CCP1IE && PIR1bits.CCP1IF) {
            PIR1bits.CCP1IF = 0;
            CCPR1 += half_cycle;
      }
}
```

7.5 CCP use examples

(4.4) CCP's Compare unit to generate melodies.

Example code, for a PIC18 with 4MHz clock (continued).

```
void interrupt low_priority LOW_ISR(void)
{
}

void ConfigSystem (void)
{
      TRISCbits.TRISC2 = 0;   /* configure CCP1 pin for output */
      T3CON = 0x81;   /* enables Timer3 in 16-bit mode, Timer1  for CCP1 time base */
      T1CON = 0x81;   /* enable Timer1 in 16-bit mode */
      CCP1CON = 0x02;   /* CCP1 compare mode, pin toggle on match */
      IPR1bits.CCP1IP = 1;   /* set CCP1 interrupt to high priority */
      PIR1bits.CCP1IF = 0;   /* clear CCP1IF flag */
      PIE1bits.CCP1IE = 1;   /* enable CCP1 interrupt */
      INTCON |= 0xC0;            /* enable high priority interrupt */
}
```

7.5 CCP use examples

(4.5) CCP's Compare unit to generate melodies.

Example code, for a PIC18 with 4MHz clock (continued).

```
void main (void)
{
        int i, j;
        ConfigSystem();
        for (j = 0; j < 3; j++) {                       // Play song several times
                i = 0;
                half_cycle = per_arr [0];
                CCPR1 = TMR1 + half_cycle;
                while (i < NOTES) {
                        half_cycle = per_arr [i];       // get the cycle count for half period of the note
                        delay ( wait [ i++] );          // stay for the duration of the note
                }
                INTCON &= 0x3F;                         // disable interrupt
                delay (11);                             // Pause before replay song
                INTCON |= 0xC0;                         // re-enable interrupt
        }
}
```