Chapter 13. CCP Module and PWM

This chapter covers the CCP (Capture/Compare/PWM) module of PIC 16F877, however, our main focus is on PWM (Pulse Width Modulation). PWM is a technique of controlling analog circuits using digital output from microcontroller. In other words, PWM a digital encoder of analog signal level. The duty cycle of a square wave generated by microcontroller is modulated to encode a specific analog signal level. The analog signal level is determined by means of a repeating series of on and off pulses of the square wave. The on-time is the time during which the digital value is applied, and the off-time is the period during which that supply is switched off. The duty-cycle is defined as the ratio of on-time and the sum of on-time and off-time, which is the pulse width. Once can control motor speed and light dimmer among other applications using PWM technique.

The focus of this chapter on PWM is how to generate different duty-cycle of different pulse width using the built-in module of CCP in 16F877. We first discuss about CCP module and then concentrate our discussion on the PWM mode of the module.

1. CCP (Capture/Compare/PWM) Module of 16F877

As noted above, CCP module has 3 different modes of operation. Before we proceed, we list necessary file registers for the operations of the CCP module. First we have to know that 16F877 has two CCP modules: CCP1 and CCP2. As noted in the pin diagram, there are two CCP pins: CCP1 and CCP2 pin. When CCP1 module is used for PWM mode, for example, the digital pulse would be generated from the CCP1 pin.

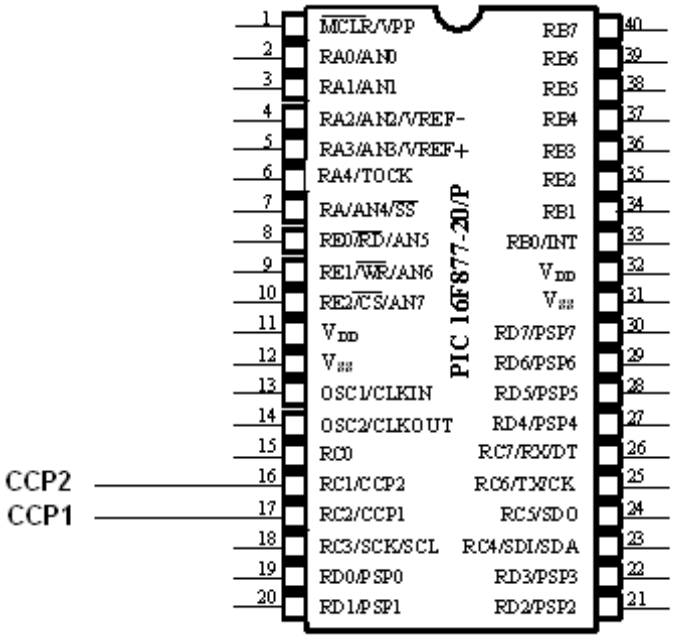


Fig. 85 CCP pins of PIC 16F887

Each CCP module has a 16-bit register which can operate as 16-bit capture register (in Capture Mode), or 16-bit compare register (in Compare Mode), or 16-bit PWM duty cycle register. As

we know there could not be a 16-bit register in 16F877 since the microcontroller is an 8-bit processor. Therefore, two 8-bit register are used to form a 16-bit register:CCPR1H and CCPR1L for upper and lower byte of the 16-bit register for CCP1 module, and CCPR2H and CCPR2L for CCP2 module.

CCP modules are controlled by CCP control registers: CCP1CON for CCP1, and CCP2CON for CCP2. In this chapter, when we refer any one of the modules without specifying either CCP1 or CCP2, then we use 'x' to indicate either one. For example, the generic symbol CCPx indicates either one of two CCP modules. Similarly, CCPxCON can be either CCP1CON or CCP2CON, depending upon with module you use.
The RAM addresses of the CCP register are listed below:

Address	Register Name	Functions
0C	PIR1	Peripheral Interrupt Request 1
0D	PIR2	Peripheral Interrupt Request 2
8C	PIE1	Peripheral Interrupt Enable 1
8D	PIE2	Peripheral Interrupt Enable 2
11	TMR2	Timer2 Module
92	PR2	Timer2 Module Period
12	T2CON	Timer2 Module Control
15	CCPR1L	Lower Byte of CCP1 Register
16	CCPR1H	Upper Byte of CCP2 register
17	CCP1CON	CCP1 Module Control
18	CCPR2L	Lower Byte of CCP2 Register
1C	CCPR2H	Upper Byte of CCP2 Register
1D	CCP2CON	CCP2 Module Control

The details of CCP1CON register is illustrated here. As we can see, CCPM3:CCPM0 decide the three modes in CCP module. Also, DCB1:DCB0 are used only for PWM as the last two LSBs for 10-bit PWM duty cycle.

CCP1CON Register

---	---	DCB1	DCB0	CCPM3	CCPM2	CCPM1	CCPM0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bits 7:6 **Not Used**

Bits 5:4 **DCB1:DCB0** Used only for PWM mode. These two bits are the two LSB bits of 10-bit PWM duty cycle. The upper 8 bits are found in CCPR1L.

Bits 3:0 **CCPM3:CCPM0** CCP1 Mode Selection Bits
0000 CCP off

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, CCP Low in initialization
1001	Compare Mode, CCP High in initialization
1010	Compare Mode, Software Interrupt
1011	Compare Mode, trigger Special Event
11xx	PWM Mode

2. Capture Mode

In Capture mode, CCPRxH:CCPRxL captures the 16-bit value of the TMR1 register when an event occurs on pin CCPx. An event is defined as:
• Every falling edge
• Every rising edge
• Every 4th rising edge
• Every 16th rising edge

An event is selected by control bits CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF of PIRx register, is set. The CCPxIF bit must be cleared in software. If another capture occurs before the value in register CCPRxL and CCPRxH is read, the previous captured value will be lost.

A capture does not reset the 16-bit TMR1 register. This is because Timer1 can also be used as the time base for other operations. The time between two captures can easily be computed as the difference between the value of the second capture that of the first capture. When Timer1 overflows, the TMR1IF bit of PIR1 register will be set, and if enabled an interrupt will occur, allowing the time base to be extended to greater than 16-bits.

When the Capture mode is changed, a capture interrupt may be generated. The user should keep the CCPxIE bit of PIEx register clear to disable these interrupts and should clear the CCPxIF flag bit of PIRx register following any such change in operating mode.

3. Compare Mode

In Compare mode, the 16-bit CCPRxH and CCPRxL register value is constantly compared against the TMR1register pair value. When a match occurs, the CCPx pin is:
• Driven High
• Driven Low
• Remains Unchanged

The action on the pin is based on the value of control bits CCPxM3:CCPxM0 (CCPxCON<3:0>). At the same time, a compare interrupt is also generated.

The user must configure the CCPx pin as an output by clearing the appropriate TRISC bit. Selecting the compare output mode, forces the state of the CCPx pin to the state that is opposite of the match state. So if the Compare mode is selected to force the output pin low on match, then the output will be forced high until the match occurs (or the mode is changed).

4. PWM Mode

In PWM mode, the CCPx pin of 16F877 produces up to a 10-bit resolution PWM output. Since the CCPx pin is multiplexed with the PORTC data latch, the corresponding TRISC bit must be cleared to make the CCPx pin an output. A PWM output has a *time-base* (i.e., pulse period) and a *time* that the output stays high (i.e., duty cycle). The frequency of the PWM is the inverse of the period (1/period). The PWM period is specified by writing to the PR2 register. PWM period in second, T_{pwm}, is determined by the following formula:

$$T_{pwm} = ([PR2]+1) \cdot 4T_{osc} \cdot [TMR2_{ps}] \text{ [sec]},$$

where TMR2_{ps} is the TMR2 pre-scale value, and T_{osc} is the oscillation period.

Let's explain the basis of the equation above. Since one instruction takes four oscillation periods we have the term $4T_{\text{osc}}$. Also, since the PR2 register content will increase by 1 every instruction cycle, we have the term [PR2]+1. Again, the pre-scale further increase the number of actual oscillation cycles to the $4T_{\text{osc}}$ multiplied by the pre-scaler of TMR2, we have the term TMR2_{ps}. All together, we have the PWM period.

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set (exception: if PWM duty cycle = 0%, the CCPx pin will not be set)
- The PWM duty cycle is latched from CCPRxL into CCPRxH

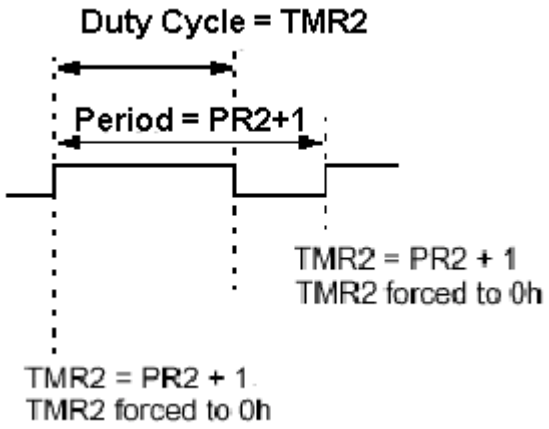
The PWM duty cycle is specified by writing to the CCPRxL register and to the DCxB1:DCxB0 (CCPxCON<5:4>) bits. The CCPRxL contains the eight MSbs and CCPxCON<5:4> contains the two LSbs. This 10-bit value is represented by DCxB9:DCxB0 in the equation below. The PWM duty cycle in second, D_{pwm}, is given by the following formula:

$$D_{pwm} = [DB9:DB0] \cdot T_{osc} \cdot [TMRS_{ps}] \text{ [sec]}$$

where [DB9:DB0] is the 10-bit value which comes from CCPRxL<8:0> for eight MSBs concatenated by CCPxCON<5:4> for two LSbs.

The DCxB9:DCxB0 bits can be written to at any time, but the duty cycle value is not latched into CCPRxH until after a match between PR2 and TMR2 occurs (which is the end of the current period). In PWM mode, CCPRxH is a read-only register. The CCPRxH register and a 2-bit internal latch are used to double buffer the PWM duty cycle. This double buffering is essential for glitch-less PWM operation. When CCPRxH and 2-bit latch match the value of TMR2

concatenated with the internal 2-bit Q clock (or two bits of the TMR2 prescaler), the CCPx pin is cleared. This is the end of the duty cycle.



Let's consider the maximum PWM resolution, R_{pwm}. This is to decide how small a duty cycle can be determined. First consider the period of PWM, T_{pwm}, with relation with the period of the oscillator clock, T_{osc}. We have to know that the period of PWM is determined a multiple of the period of oscillation. In other words, in a PWM period there must be n number of the oscillation period: $T_{pwm} = n \cdot T_{osc}$. By the way, the resolution in bits for a number *n* is determined by the power of 2 of the number *n*. (Remember that, in ADC, the 10-bit resolution has the value in the range of 1 – 1024 or 2^0 - 2^{10}. In other words, the resolution in bits is the power of 2 for the maximum value it can provide.)

Therefore, the above equation can be changed to: $T_{pwm} = 2^{R_{pwm}} \cdot T_{osc}$ where R_{pwm} is the maximum PWM resolution in bits. Therefore, the equation for R_{pwm} can be rewritten as:

$$R_{pwm} = \log_2\left(\frac{T_{pwm}}{T_{osc}}\right) = \frac{\log(T_{pwm}/T_{osc})}{\log 2}.$$

This equation can be further altered in terms of frequencies of PWM and oscillation:

$$R_{pwm} = \frac{\log(f_{osc}/f_{pwm})}{\log 2}.$$

Let's have an example. The following example I got from one of the Microchip Technology's manual of Mid-Range Microcontrollers. Assuming that the desired PWM frequency is 78.125KHz (i.e., PWM period of 12.8μs) with oscillation speed of 20MHz, and TMR2 prescale 1. First thing we have to find is the content of PR2 register. From the PWM period equation, $T_{pwm} = ([PR2]+1) \cdot 4T_{osc} \cdot [TMR2_{ps}]$, we can draw an equation for [PR2], the content for PR2 register: $[PR2] = \frac{T_{pwm}}{4T_{osc} \cdot [TMR2_{ps}]} - 1$.

Since, T_{pwm}=12.8μs, $T_{osc} = 1/20\times10^6 = 0.05\,\mu s$, and $TMR2_{ps}$=1 (since prescale is set to 1),

$$[PR2] = \frac{12.8}{4(0.05)\cdot 1} - 1 = 64 - 1 = 63$$

Let's calculate the maximum resolution we get from PWM duty cycle under the example case. From the maximum resolution equation, $R_{pwm} = \log_2\left(\frac{T_{pwm}}{T_{osc}}\right)$, since T_{pwm}=12.8μs and T_{osc}=0.05μs, we can get $R_{pwm} = \log_2\left(\frac{T_{pwm}}{T_{osc}}\right) = \log_2 \frac{12.8}{0.05} = \log_2 256 = 8$. So the maximum resolution of the PWM duty cycle in the example is 8. What does this mean?

What maximum resolution being 8 means that, simply put, the 10-bit PWM values determined by DCxB9:DCxB0 is less than or equal to 2^8. In other words, even though PWM duty cycle can get you maximum 10-bit resolution, it does not mean that it provides you with 10-bit resolution every time. In the example case, it provides only 8-bit resolution. In analogy with a one-foot ruler which has 12 divisions of 1 inch scale, and 12 divisions inside each inch scale, we can say that the ruler has maximum resolution of 192 (or 7-bit resolution). This means you can dive one foot by 192 equally divided scales. And the resolution is therefore, 0.0052 in. In analogy to PWM, for a given pulse period, to have 8-bit resolution is to have a total number of 256 division scales in the pulse period. In our example, the pulse period is 12.8μs, therefore the each scale is 0.05 μs. In other words, there is no way to reduce the scale to, for example, 12.8/512 =0.025μs, since this scale is close to 1 instruction execution time. Duty cycle is simple the multiple of the minimum scale.

In other words, any value in DCxB9:DCxB0 greater than 255, would be regarded as if the same as 255. In order to achieve higher resolution, the PWM frequency must be decreased. Or, in order to achieve higher PWM frequency, the resolution must be decreased. The following table (also adopted from the Microchip Technology's Mid-Range PIC Family Microcontroller) lists example PWM frequencies and resolutions for f_{osc} = 20 MHz. The TMR2 prescaler and PR2 values are also shown.

PWM frequency	**1.22KHz**	**4.88KHz**	**19.53KHz**	**78.12 KHz**	**156.3KHz**	**208.3KHz**
TMR2 prescaler	16	4	1	1	1	1
[PR2] value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution [bits]	10	10	10	8	7	5.5

Now let's further discuss about the PWM duty cycle formulae with the given example. Assume that we want to have a 50% duty cycle, on for the half the pulse period and off for the rest of the pulse period. From the PWM duty cycle equation, $D_{pwm}[\text{sec}] = [DB9:DB0]\cdot T_{osc}\cdot[TMRS_{ps}]$, (note that the duty cycle is given with [sec] not in[%]), we get the equation for [DB9:DB0]:

$$[DB9:DB0] = \frac{D_{pwm}}{T_{osc}\cdot[TMRS_{ps}]}.$$

Since 50% of the PWM pulse period is (0.5)(12.8μs)=6.4μs, and with T_{osc}=0.05μs and $TMR2_{ps}$=1, we get $[DB9:DB0]=\frac{6.4\times10^{-6}}{0.05\times10^{-6}\cdot1}=128=80\,\text{h}$.
Therefore $[DB9:DB0]=0x80=0010000000$.

In practice, we have to write B'0010000' (or 0x20 as the upper 8 digits)to CCP1L register, and the last two digits 00 to DCxB1:DCxB0 bits (CCPxCON<5:4>). If we follow all these steps, the pin CCP1 would generate +5V pulse with its period 12.8μs with on-period of 6.4 μs and off-period 6.4 μs.

5. PWM Application with 16F877

Configuration Steps for PWM
The following steps configure the CCP module for PWM operation. Before we start the first step, we have to have a clear picture in the pulse period and duty cycle, applying the formula we discussed before.

The first step is to write the PWM period to PR2 register. Let's have a practical example of period of 0.5ms. Since 16F877 clock frequency is 20 MHz, T_{osc}=0.05μs. With the selection of TMR2 prescale value 16, then we have the following PR2 value:

$$[PR2]=\frac{T_{pwm}}{4T_{osc}\cdot[TMR2_{ps}]}-1=\frac{5\times10^{-4}}{4(0.05\times10^{-6})(16)}-1=155.25\approx155=0x9B$$

So we can have the following lines of code for PWM period:

```
; PWM Period Setting
      banksel     PR2
      movlw       0x9b        ;0.5ms Period
      movwf       PR2
```

Second, we establish the PWM duty cycle by writing to the DCxB9:DCxB0 bits. Let's assume that we want to generate 50% duty cycle pulse. Since we have to express the duty of PWM, D_{pwm}, in terms of second, we have the half the period, 0.25ms, as D_{pwm} (50% of the period of 0.5ms): $D_{pwm}=2.5\times10^{-4}$. Remember that the second step is to find the value for [DB9:DB0]. From the equation for [DB9:DB0], we have the value for it:

$$[DB9:DB0]=\frac{D_{pwm}}{T_{osc}\cdot[TMRS_{ps}]}=\frac{2.5\times10^{-4}}{(0.05\times10^{-6})(16)}=312.5\approx313=139\text{h}$$

In binary, [DB9:DB0] should have: 01 0011 1001. Since the highest 8 bits are to be stored in CCPR1L register (the content of CCPR1L is then 0x4E), and the two least significant bits are to be stored in to CCP1CON<5:4> (the bit 5 must be cleared while the bit 4 set for this case), we have the following lines of code for the second step:

```
;PWM duty cycle 50% setting
      movlw       0x4E
```

```
      movwf       CCPR1L
      bcf         CCP1CON, B5       ;B5   EQU   0x05 must be declared above
      bsf         CCP1CON, B4       ;B4   EQU   0x04
```

Third step is to properly allocate the CCP1 pin, which shares with RC2 pin of PORTC. Since the CCP1 pin is an output pin, PORTC<2> must be set as an output by clearing the TRISC<2> bit. In the following lines of code, PWM1 must be declared as 2 for CCP1 pin.

```
;TRISC Setting for output
      banksel     TRISC
      bcf         TRISC, PWM1
```

Four, we establish the TMR2 prescale value and enable Timer2 by T2CON register configuration. Let's look at TMR2 before its control register T2CON in detail for use in the fourth step. TMR2 consists of 3 components: pre-scaler, period register, and post-scaler. The pre-scaler is to decide how often TMR2 counts in terms of the oscillation frequency. There are 3 different scales- 1:1, 1:4, and 1:16. If 1:1 is selected, TMR2 counts at every oscillation cycle (i.e. 0.2μs in 20MHz oscillator). The ratio of 1:4 makes TMR2 count at every 4th oscillation cycle. This means the counting frequency is four timer slower than 1:1 ratio. Then, the ratio of 1:16 slows the counting further to 16 times of the oscillation cycle. The content of TMR2 register is continuously compared with PR2 register. Note that PR2 stores the PWM period information. When they are matched, and internal interrupt is generated to indicate the overflow. The post-scaler is to control the frequency of the interrupt flagging. If ratio of 1:1 is selected for postscale, the interrupt flag is issued every time TMR2 and PR2 are matched. If for example 1:16 is selected, the flag will be issued at every 16th time they are matched.

T2CON register shown in the figure is to control the ratios of pre-scaler and post-scaler. Also, there is a bit (bit2, TMR2ON) which controls the On/Off of the TMR2 module.

T2CON Register

---	TPS3	TPS2	TPS1	TPS0	TMR2ON	T2PS1	T2PS0
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

Bit 7: Unused

Bit 6:3 Timer2 Output Postscale Select Bits
0000 = 1:1
0001 = 1:2
.
.
1111 = 1:16

Bit 2 Timer2 ON bit
1 = Timer2 ON
0 = Timer2 OFF

Bit 1: 0 Timer2 Clock Prescale Select Bits
00 = 1
01 = 4
1x = 16

Since we already selected our pre-scale as 16, we choose the 1:16 ratio by selecting bits 1 and 0 of T2CON as 11 or 10. For post-scale, we choose 1:1 ratio by selecting TPS3:TPS0 as 0000. Finally we turn on TMR2 module by setting the TMR2ON bit. In all, the proper byte data for T2CON is then, 00000111=0x07. We selected 0 for the bit 7 and 1 for the last bit. The fourth step can be realized by the following lines of code:

```
      banksel     T2CON
;TMR2 Prescale selection and TMR2 Turn On
      movlw       0x07
      movwf       T2CON
```

Fifth and the last step in the PWM setting is to select the PWM mode from CCP module by CCP1CON register bit selection. In CCP1CON, the last four bits, CCPM3:CCPM0 are assigned for mode selection, and 11xx bit formation is for PWM mode. So the following simple lines would suffice for the fifth step:

```
;PWM module selection from CCP1CON
;PWM module is selected 11xx of the lower nibble of CCP1CON
      banksel     CCP1CON           ;PWM Turn ON
      bsf   CCP1CON, 0x03     ;11xx<3:0> is PWM mode
      bsf   CCP1CON, 0x02
```

Example Code of PWM

The following code shows a complete PWM program which generates a pulse of 0.5ms period with duty cycle of 50% at CCP1 pin of 16F877. The pulse we can see is a continuous stream of 0.25ms long +5V DC signal followed by 0.25 ms long 0V signal. Readers would appreciate the many line of comments at the upper part of the code in particular.

```
;PWM-1.ASM
;
;This program uses CCP1 module with TMR2 for
;PWM signal generation at ccp1 (RC2) pin
;
; PWM with 0.5 ms period with 50% duty cycle
;   |-----|
;---|     |-----
;
;
        list P = 16F877

STATUS      EQU   0x03
INTCON      EQU   0x0b
TMR2        EQU   0x11
PIE1        EQU   0x8c
```

```
PIR1        EQU   0x0c
PR2         EQU   0x92        ;
CCP1CON     EQU   0x17
CCPR1L      EQU   0x15
T2CON       EQU   0x12
TRISC       EQU   0x87
B5          EQU   0x05
B4          EQU   0x04
PWM1        EQU   0x02
TMR2IF      EQU   0x01

      CBLOCK      0x20              ; RAM AREA for USE at address 20h
            FIFTY
        ENDC                  ;end of ram block
;
;=========================================================
        org       0x0000            ;line 1
        GOTO      START       ;line 2 ($0000)

;=========================================================
;     16F877 Clock Frequency = 20 MHz
;     Tosc = 1/[Clock Frequency]
;     PWM Period = 2000Hz = 0.0005 Sec
;     TMR2 Prescale = 16
;     PR2 = (period / [4*Tosc*Prescale]) - 1
;         = 155
;         = 0x9b
;     PWM Ducty Cycle = 50% of the PWM period
;                 = 0.5 *(0.0005)=0.00025
;       --->[0.00025]/[5*10^(-8)*16]=312.5 ---> 313--->0x139
;     CCPR1L       CCP1CON
;       76|5432|10  76|54|3210
;       01|0011|10    |01|
;     CCPR1L:CCP1CON<5:4>=(PWM Duty Cycle)/(Tosc*Prescale)=139h
;     CCPR1L = 0x4e
;     CCP1CON<5:4>=b'01'
;
      org   0x0005
START
      banksel     PIE1
      clrf        PIE1
      banksel     CCP1CON
      clrf        CCP1CON
      clrf        TMR2
      clrf        INTCON
      clrf        PIR1
      movlw       0x4e        ;50% Duty Cycle
      movwf       FIFTY
; PWM Period Setting
      banksel     PR2
      movlw       0x9b        ;2000Hz Period
      movwf       PR2
      banksel     FIFTY
;PWM Duty Cycle
;50% is selected for 5 seconds
      movf        FIFTY, 0    ;Move 50% Duty Data to W reg.
      movwf       CCPR1L
```

```
      bcf         CCP1CON, B5
      bsf         CCP1CON, B4
;TRISC Setting for output
      banksel     TRISC
      bcf         TRISC, PWM1
      banksel     T2CON
;TMR2 Prescale selection and TMR2 Turn On
      movlw       0x07
      movwf       T2CON
;CCP1CON Module Setting
;
      banksel     CCP1CON           ;PWM Turn ON
      bsf         CCP1CON, 0x03     ;11xx<3:0> is PWM mode
      bsf         CCP1CON, 0x02

      END
```

Next example code is the continuation and extension of the 0.5ms 50% duty PWM signal generation. This code is to generate PWM of 0.5ms period with 50% duty cycle for 5 seconds, 2 second delay, followed by 5 seconds of same PWM but with 20% duty cycle. Readers are encouraged to pay attention to the comments line in the upper part of the code. The first PWM is generated for 5 seconds, the PWM is off for 2 seconds, and then with 20% duty cycle it is ON again for the next 5 seconds before it goes off. Time delay subroutines are those developed with 'time delay without using timer module.'

```
;PWM-2.ASM
;
;This program uses CCP1 module with TMR2 for
;PWM signal generation at ccp1 (RC2) pin
;
; PWM with 2KHz 50% duty cycle for 5 seconds, with 2 sec delay
; followed by 5 seconds of 20% duty cycle
;
        list P = 16F877

STATUS      EQU   0x03
INTCON      EQU   0x0b
TRISD       EQU   0x88
PORTD       EQU   0x08
TMR2        EQU   0x11
PIE1        EQU   0x8c
PIR1        EQU   0x0c
PR2         EQU   0x92        ;
CCP1CON     EQU   0x17
CCPR1L      EQU   0x15
T2CON       EQU   0x12
TRISC       EQU   0x87
B5          EQU   0x05
B4          EQU   0x04
PWM1        EQU   0x02
TMR2IF      EQU   0x01
LED1        EQU   0x01
LED0        EQU   0x00
```

```
      CBLOCK      0x20              ; RAM AREA for USE at address 20h
            FIFTY
            TWENTY
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount1s
            Kount10s
            Kount1m
        ENDC                  ;end of ram block
;
;
;=========================================================
      org         0x0000            ;line 1
      GOTO        START       ;line 2 ($0000)
;=========================================================
;IF
;     16F877 Clock Frequency = 20 MHz
;     Tosc = 1/[Clock Frequency]
;     PWM Period = 2000Hz = 0.0005 Sec
;     TMR2 Prescale = 16
;THEN,
;     PR2 = (period / [4*Tosc*Prescale]) - 1
;         = 155
;         = 0x9b
;-----
;IF
;     PWM Ducty Cycle = 50% of the PWM period
;                 = 0.5 *(0.0005)=0.00025
;       --->[0.00025]/[5*10^(-8)*16]=312.5 ---> 313--->0x139
;     CCPR1L       CCP1CON
;       76|5432|10  76|54|3210
;       01|0011|10    |01|
;THEN
;     CCPR1L:CCP1CON<5:4>=(PWM Duty Cycle)/(Tosc*Prescale)=139h
;     CCPR1L = 0x4e
;     CCP1CON<5:4>=b'01'
;--
;IF
;     PWM Ducty Cycle = 20% of the PWM period
;                 = 0.2 *(0.0005)=0.0001
;       --->[0.0001]/[5*10^(-8)*16]=125 --->0x7D
;     CCPR1L      CCP1CON
;       76|5432|10  76|54|3210
;       00|0111|11    |01|
;THEN
;     CCPR1L:CCP1CON<5:4>=(PWM Duty Cycle)/(Tosc*Prescale)=7dh
;     CCPR1L = 0x1f
;     CCP1CON<5:4>=b'01'
;
;
      org         0x0005
START
      banksel     TRISD
      clrf        TRISD
      banksel     PORTD
```

```
      clrf        PORTD
      banksel     PIE1
      clrf        PIE1
      banksel     CCP1CON
      clrf        CCP1CON
      clrf        TMR2
      clrf        INTCON
      clrf        PIR1
      movlw       0x4e        ;50% Duty Cycle
      movwf       FIFTY
      movlw       0x1f        ;20% Duty Cycle
      movwf       TWENTY
; PWM Period Setting
      banksel     PR2
      movlw       0x9b        ;2000Hz Period
      movwf       PR2
      banksel     FIFTY
;PWM Duty Cycle
;50% is selected for 5 seconds
DUTY
      movf        FIFTY, 0    ;Move 50% Duty Data to W reg.
      banksel     PORTD
      bsf         PORTD, LED1

      movwf       CCPR1L
      bcf         CCP1CON, B5
      bsf         CCP1CON, B4
;TRISC Setting for output
      banksel     TRISC
      bcf         TRISC, PWM1
      banksel     T2CON
;TMR2 Prescale selection and TMR2 Turn On
      movlw       0x07
      movwf       T2CON
;CCP1CON Module Setting
      banksel     CCP1CON           ;PWM Turn ON
      bsf         CCP1CON, 0x03     ;11xx<3:0> is PWM mode
      bsf         CCP1CON, 0x02
      banksel     PORTD
      bsf         PORTD, LED1
      call        delay1s
      call        delay1s
      call        delay1s
      call        delay1s
      call        delay1s           ;Pulse is generated during this period
      banksel     CCP1CON
      bcf         CCP1CON, 0x03           ;PWM off
      bcf         CCP1CON, 0x02
      bcf         CCP1CON, 0x01
      bcf         CCP1CON, 0x00
      banksel     PORTD
      bcf         PORTD, LED1
      call        delay1s
      call        delay1s                 ;PWM is off during this period

      banksel     TWENTY
      movf        TWENTY,0
```

```
      banksel     PORTD             ;20% duty
      bsf         PORTD, LED0
      banksel     CCPR1L
      movwf       CCPR1L
      bcf         CCP1CON, B5
      bsf         CCP1CON, B4
      banksel     CCP1CON           ;PWM Turn ON
      bsf         CCP1CON, 0x03     ;11xx<3:0> is PWM mode
      bsf         CCP1CON, 0x02
      call        delay1s
      call        delay1s
      call        delay1s
      call        delay1s
      call        delay1s                ;PWM with 20% duty generated
      banksel     PORTD
      bcf         PORTD, LED0
      banksel     CCP1CON
      bcf         CCP1CON, 0x03          ;PWM off
      bcf         CCP1CON, 0x02
      bcf         CCP1CON, 0x01
      bcf         CCP1CON, 0x00
      call        delay1s
      call        elay1s                 ;PWM off for 2 seconds
      goto        DUTY                   ;Do it again
;
;=============================================================
;DELAY SUBROUTINES


; 1 instruction cycle for 20MHz clock is 0.2 us
; Therefore 120 uS delay needs 600 instuction cycles
;  600 =199*3 +3 ---->Kount=199=0xC7
;  or  =198*3 +6 ---->Kount=198=0xC6
;  or  =197*3 +9 ---->Kount=197=0xC5

Delay120us
      banksel     Kount120us
      movlw       H'C5'       ;D'197'
      movwf       Kount120us
R120us
      decfsz      Kount120us
      goto        R120us
      return
;
;100us delay needs 500 instruction cycles
;  500 =166*3 +2 ---->Kount=166=0xA6
;  or  =165*3 +5 ---->Kount=165=0xA5
;  or  =164*3 +8 ---->Kount=164=0xA4
Delay100us
      banksel     Kount100us
      movlw       H'A4'
      movwf       Kount100us
R100us
      decfsz      Kount100us
      goto        R100us
      return
```

```
;
;10ms delay
; call 100 times of 100 us delay  (with some time discrepancy)
Delay10ms
      banksel     Kount10ms
      movlw       H'64' ;100
      movwf       Kount10ms
R10ms call        delay100us
      decfsz      Kount10ms
      goto        R10ms
      return
;1 sec delay
;call 100 times of 10ms delay
Delay1s
      banksel     Kount1s
      movlw       H'64'
      movwf       Kount1s
R1s   call        Delay10ms
      decfsz      Kount1s
      goto        R1s
      return
;
;
;10 s delay
;call 10 times of 1 s delay
Delay10s
      banksel     Kount10s
      movlw       H'0A'       ;10
      movwf       Kount10s
R10s  call        Delay1s
      decfsz      Kount10s
      goto        R10s
      return
;
;1 min delay
;call 60 times of 1 sec delay
Delay1m
      banksel     Kount1m
      movlw       H'3C' ;60
      movwf       Kount1m
R1m   call        Delay1s
      decfsz      Kount1m
      goto        R1m
      return
;=====================================================

      END
```