Esercitazioni PIC Midrange

Inserto_f : PWM e PWM+.

Il livello delle periferiche via via integrate nei microcontroller corrisponde alla necessità di disporre di elementi in grado di svolgere autonomamente funzioni anche molto complesse, in modo da lasciare libera l'unità centrale per lo svolgimento di altre operazioni.

Tra queste periferiche complesse, una è presente in vari Midrange; si tratta del modulo chiamato **CCP/PWM** che è acronimo di ***Capture Compare PWM*** nella versione base, più semplice e **ECC/PWM+** in quella enhanced, più ricca di opzioni.

A seconda del chip possiamo avere uno o l'altro. Per i modelli che stiamo usando in queste esercitazioni, la tabella seguente riporta la situazione:

PIC	CCP/PWM
12F629/675	no
12F683	**si**
16F684	**enhanced**
16F688	no
16F690	**enhanced**

La versione base dispone di una sola uscita PWM, mentre quella enhanced può gestire 4 uscite.

Iniziamo a considerare la versione base, integrata in 12F683.

CCP/PWM permette di eseguire diverse funzioni:

- nel modo ***Capture*** consente di rilevare la temporizzazione di un evento, ad esempio il periodo
- nel modo ***Compare*** consente di triggerare un evento in funzione di una certa quantità di tempo programmata
- nel modo ***PWM*** consente di generare una o più uscite a frequenza e duty cycle programmabili

Qui ci interessiamo specificamente della funzione ***PWM***.

PWM.

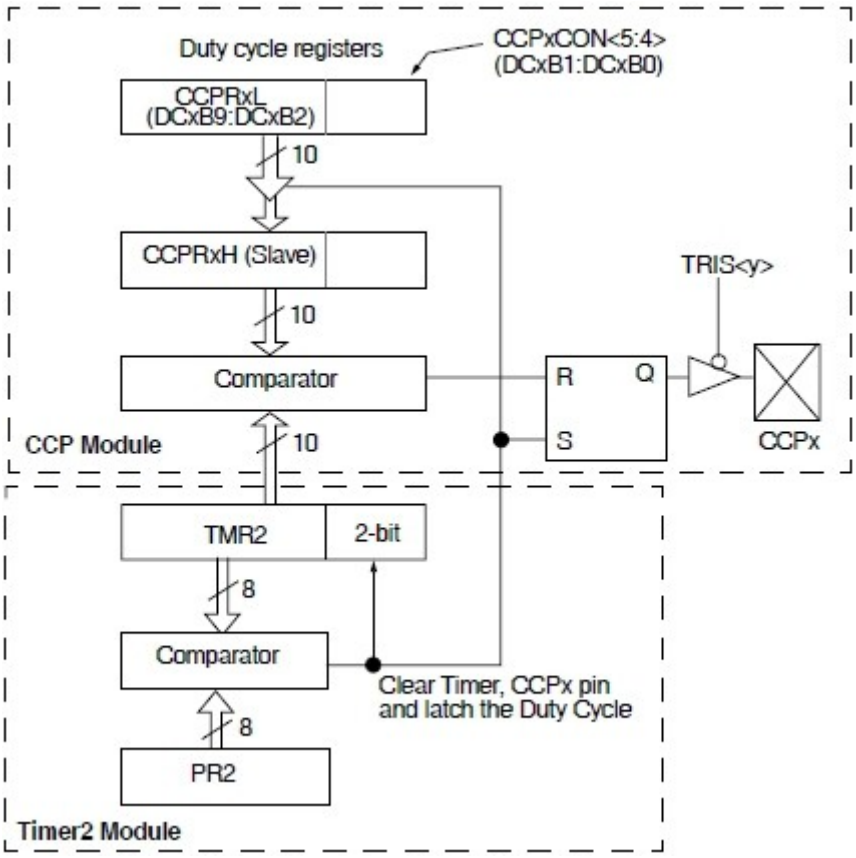
E' noto che un segnale ***PWM*** (***Pulse Width Modulation*** - modulazione a variazione della larghezza d'impulso) è una onda quadra con un **periodo fisso** (l'inverso della frequenza del segnale), ma con **rapporto tra i tempi on/off (duty cycle) variabile**.

Questo vuol dire che un duty cycle del 50% corrisponde ad un tempo uguale di on e di off, ovvero ad una onda quadra simmetrica; un duty cycle del 10% indica il tempo di on (quello di off occuperà il rimanete 90%) e viceversa.
Se il segnale in uscita è sempre a livello alto possiamo dire che si tratta di un duty cycle del 100%, mentre se è sempre a livello basso si tratta di duty cycle dello 0%.

La variazione del duty cycle corrisponde ad una diversa trasmissione di energia: allo 0% il carico collegato all'uscita non riceverà energia, mentre al 100% riceverà energia in continuazione; tutti i valori percentuali intermedi trasferiranno energia proporzionale.

Questo concetto è impiegato, ad esempio, per la variazione di luminosità di un LED o di velocità di un motore.

Lo schema di principio è il seguente:



L'elemento di temporizzazione per la generazione del segnale **PWM** è costituito dal **Timer2** che abbiamo visto nell'esercitazione precedente.

Il comparatore è a 10 bit e questo permette una risoluzione analoga del **PWM**.

Dal lato di **Timer2**, i 10 bit sono formati dagli 8 bit contenuti in PR2 e da quelli del prescaler: dato che **TMR2** è a 8bit, il numero completo saràa 10bit.

Da notare che :

il postscaler non interviene nella temporizzazione del PWM.

Dal punto di vista pratico, va programmato a 1:1 in modo da ottenere, dove necessario, l'attivazione del flag **TMR2IF** al momento della comparazione avvenuta con successo.

Dal lato del modulo **CCP/PWM**, i 10 bit necessari sono forniti chiamati **DCxB9:0**. I più alti (**DCxB9:2**) sono scritti nel registro **CCPRxL**, che è mappato in memoria ed è accessibile in lettura e scrittura. I due bit rimanenti sono forniti dai bit 5:4 di **CCPxCON**, che è pure mappato in memoria e accessibile in scrittura e lettura.

Il valore indicato da questi bit è il numero di impulsi necessari per pareggiare il **TMR2** (+2bit)

La comparazione, però, non avviene su questi registri, ma attraverso il **CCPRxH**, il cui contenuto è la copia di **CCPRxL +** i due bit 5:4 di **CCPxCON** e contiene l'intero blocco di dieci bit **DCxB9:0**.

Questo registro, in modo PWM, non è direttamente accessibile in scrittura, ma può solo essere letto.

La sua funzione è quella di costituire un **doppio buffer che evita glitch** nel cambio di duty cycle. Infatti, i bit **DCxB9:0** possono essere scritti in ogni momento, ma sono trasferiti automaticamente in **CCPRxH** solo dopo una avvenuta comparazione, in modo tale da non permettere una modifica dei bit da comparare durante l'esecuzione di un periodo PWM.

I bit **DC1B1:0** costituiscono i due bit meno significativi della risoluzione a 10 bit del PWM.

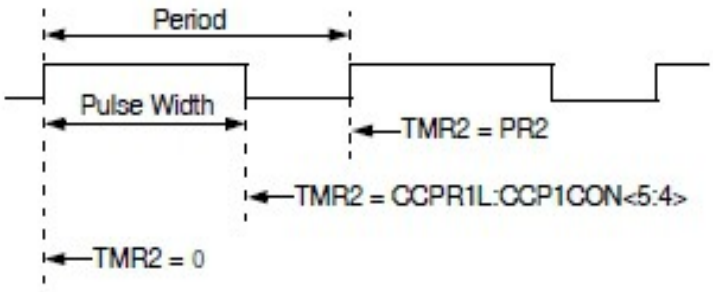
NOTA: la documentazione Microchip indica i registri e i bit con un generico **x** .
Questo è dovuto al fatto a cui si è accennato all'inizio, ovvero all'esistenza di moduli **PWM** che hanno una o più uscite. Registri e bit hanno le stesse funzioni per una o più uscite, cosa indicata dalla **x** alla quale si sostituirà il numero dell'uscita considerata.
Quindi, in presenza di una sola uscita, si potrà parlare di **CCP1**, **CCPR1L**, **CCPR1H**, **DC1B9:0**.

L'uscita del segnale è diretta ad un pin **CCPx,** nel nostro caso **CCP1**, che assume questa funzione se un modo PWM è abilitato.

Osserviamo, dallo schema di principio sopra riportato, che l'uscita dipende dal bit del registro TRIS relativo. Ovvero:

- **per avere il segnale non basta programmare il modulo CCP/PWM, ma occorre anche selezionare il pin CCP1 come uscita.**

Da un punto di vista grafico, la situazione è rappresentabile così:



La comparazione tra il contenuto di **`TMR2`+2bit** e **`CCPR1H`** origina l'uscita PWM.

In sintesi possiamo dire che:

- Il **periodo** è proporzionale a **`PR2`**, ovvero al tempo generato da Timer2
- la **larghezza dell'impulso** (***Pulse Width***) dipende da **`CCPR1L`** e dai bit **`CCP1CON5:4`**, ovvero dal risultato della comparazione a 10 bit.

Normalmente intendiamo che l'impulso generato è a livello alto, anche se la programmazionedel modulo consente di invertire i livelli.

`CCP1CON` è il registro specifico di controllo del modulo **CCP**:

REGISTER 11-1: CCP1CON: CCP1 CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

I bit **`CCP1M3:0`** riguardano il funzionamento complessivo del modulo:

CCP1M3:0	Funzione
0000	Modulo CCP disabilitato
0001 - 0011	riservati
0100	Modo ***Capture***, ogni fronte di discesa
0101	Modo ***Capture***, ogni fronte di salita
0110	Modo ***Capture***, ogni 4 fronti di salita
0111	Modo ***Capture***, ogni 16 fronti di salita
1000	Modo ***Compare***: set uscita alla comparazione
1001	Modo ***Compare***: clear uscita alla comparazione
1010	Modo ***Compare***: interrupt senza uscita
1011	Modo ***Compare***: trigger per evento speciale
110x	**PWM uscita attiva alta**
111x	**PWM uscita attiva bassa**

Le ultime due combinazioni sono quelle che interessano la funzione PWM; sono la stessa cosa, ma con il livello dell'uscita invertito.

 Per default il modulo CCP/PWM è disabilitato. Per utilizzarlo, dobbiamo abilitarne una modalità.

Per utilizzarlo, dobbiamo abilitarne una delle modalità.
Con il modulo disabilitato, l'uscita **CCP1** è pure disabilitata e al pin possono essere applicate la altre funzioni condivise.

Il periodo del PWM è dato da:

$$periodo\ PWM = (PR2+1)*4*Tosc*(TMR2\ prescaler\ value)$$

dove:

- ***PR2*** è il valore caricato nel registro
- ***Tosc*** è il periodo della frequenza del clock che alimenta Timer2, ovvero ***1/ (Fosc/4)***
- ***TMR2 prescaler value*** è il valore dei due bit meno significativi del registro **T2CON**

La frequenza del PWM è definita come l'inverso del periodo ***1/periodoPWM***, ovvero:

$$frequenza\ PWM = Fosc / (4* (PR2+1) * prescaler)$$

Come abbiamo accennato prima, il postscale del Timer2 non viene interessato dalla gestione del PWM.

Quando **TMR2** è uguale a **PR2**, all'impulso di clock successivo si generano questi tre eventi:

- **TMR2** è azzerato
- Il pin **CCP1** è settato (se il duty cycle è uguale allo 0%, il pin **CCP1** non è settato)
- Il duty cycle è latchato da **CCPR1L** a **CCPR1H**

Il duty cycle del PWM è specificato scrivendo un valore di 10 bit in due registri: **CCPR1L** per i bit da 9 a 2 e i bit **DC1B<1:0>** bit del registro **CCP1CON**.

La seguente equazione è utilizzata per calcolare il duty cycle:

$$PWM\ duty\ cycle = (DCxB9:DCxB0) \bullet Tosc \bullet (TMR2\ prescaler)$$

da cui:

$$DCxB9:DCxB0 = 4* (PR2+1) * PWM\ duty\ cycle$$

 **Da notare che se il duty cycle è maggiore del periodo, il pin CCP1 non viene azzerato: questo permette un duty cycle del 100%.**

Il rapporto del duty cycle è dato da:

$$Duty\ Cycle\ Ratio = (DCxB9:DCxB0) / 4 * (PR2+1)$$

Non è possibile la massima risoluzione a 10bit per qualsiasi frequenza di PWM; essa è limitata da

risoluzione = log (4(PR2+1))) / log(2) in bit

Notiamo che :

- la risoluzione è dipendente dal valore caricato in **PR2**, quindi dal periodo del PWM.
- il Duty Cycle dipende dal valore dei bit ***DCxB9:0***, ma anche dal prescaler di Timer2 e dalla ***Fosc.***
- il periodo del PWM dipende dalla frequenza generata da Timer2 (prescaler, **PR2** e ***Fosc***).

La seguente tabelle riporta alcune possibilità per una ***Fosc=20MHz***

Frequenza PWM	1.22kHz	4.88kHz	19.53kHz	78.12kHz	153.6kHz	208.3kHz
Prescaler	16	4	1	1	1	1
PR2	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Risoluzione	10bit	10bit	10bit	8bit	7bit	5.5bit

Se usiamo una ***Fosc=8MHz***:

Frequenza PWM	1.22kHz	4.88kHz	19.53kHz	78.12kHz	153.6kHz	208.3kHz
Prescaler	16	4	1	1	1	1
PR2	0x65	0x65	0x65	0x19	0x0C	0x09
Risoluzione	10bit	10bit	8bit	6bit	5bit	5bit

Se usiamo una ***Fosc=4MHz***:

Frequenza PWM	1.22kHz	4.88kHz	19.53kHz	78.12kHz	153.6kHz	208.3kHz
Prescaler	4	1	1	1	-	-
PR2	0xCC	0xCC	0x32	0x12	-	-
Risoluzione	10bit	9bit	7bit	5bit	-	-

- **la massima frequenza ottenibile per il PWM è proporzionale alla *Fosc***
- **per aumentare la risoluzione, la frequenza del PWM deve essere ridotta a parità di clock oppure aumentata la *Fosc*.**

In effetti, ricordiamo, che si tratta della ***Fosc/4*** che alimenta il Timer2.

Alcuni esempi di calcolo:
Vogliamo ottenere un PWM con frequenza 78.125kHz. Il clock è 20MHz. Usiamo un prescaler di 1

1/78125kHz = (PR2+1) * 4 *1/20MHz * 1
12.8us = (PR2+1) * 4 * 50ns * 1
PR2 = 63

In queste condizioni, la massima risoluzione sarà:

$$1/78125kHz = 2^{PWMresolution} * 1/20MHz * 1$$
$$256 = 2^{PWMresolution}$$
$$log(256) = PWMresolution * log(2)$$
$$PWMresolution = 8bit$$

Questo vuol dire che sarà necessario che il valore dei bit **DCxB9:0** sia compreso tra 0 e 255 (FFh). Con valori maggiori si otterrà un duty cycle del 100%, ovvero il pin **CCP1** sempre a livello 1. Ovviamente non è necessario impegnarsi in questi calcoli, dato che sul web sono presenti numerose applet per il calcolo dei parametri. Alcuni link attivi alla data di compilazione di queste pagine:

- http://eng-serve.com/pic/pic_pwm.html
- http://www.micro-examples.com/public/microex-navig/doc/097-pwm-calculator.html
- http://micro.alleypress.org/

All'atto pratico, la sequenza di inizializzazione del modulo PWM è la seguente:

1. calcolare i parametri necessari per ottenere il PWM voluto con una applet opportuna o con le formule sopra esposte
2. stabilire il periodo scrivendo **PR2**
3. configurare **CCP1CON** per la modalità PWM
4. stabilire il duty cycle scrivendo **DcxB9:0**
5. configurare il Timer2: - cancellate **TMR2IF**
   - stabilire il valore voluto per **TMR2**
   - avviare il Timer2
6. attendere l'overflow (**TMR2IF** settato)
7. configurare il pin **CCP1** come uscita agendo sul relativo bit del TRIS

In istruzioni:

```
; carica PR2
        banksel PR2
        movlw   PR2num
        movwf   PR2

; setup CCP/PWM
        banksel CCP1CON
;               b'00000000'
;                 --00---   DC1B1:0
;                 ----1100  PWM attivo alto
        movlw   b'00001100'
        movwf   CCP1CON

; set Timer2 per pre 1:4
        banksel T2CON
;               b'00000000'
;                 -0000---  postscaler 1:1
;                 -----1--  accendi timer
;                 ------01  prescaler 1:4
        movlw   b'01001101'
        movwf   T2CON
```

```
; Aggiorna il duty cycle per il prossimo periodo
        movlw   pwmduty
        banksel PR2
        movwf   PR2

; interrupt per Timer2
        banksel PIR1
        bcf     PIR1, TMR2IF    ; cancella flag
        banksel PIE1
        bsf     PIE1, TMR2IE
        banksel PIR1

PWM_period_match:
        btfss   PIR1, TMR2IF
         goto   PWM_period_match
        bcf   PIR1, TMR2IF      ; cancella flag

; attiva uscita CCP1
        banksel TRISIO
        bcf     TRISIO, GP2
```

NOTE:

1. **Il modo PWM non dispone di un interrupt specifico, ma si può fare leva su quello del Timer2, se necessario.**
2. **Non è possibile generare PWM in modalità sleep** in quanto il clock primario, che alimenta Timer2, è bloccato.

In generale possiamo aggiungere che:

1. non è sempre possibile ottenere esattamente la frequenza o il duty cycle desiderati, ma ci si deve accontentare di approssimazioni, a meno di cambiare la ***Fosc*** del clock principale

2. in particolare, il valore del duty cycle dovrà essere minore del periodo (valore caricato in **PR2**); in caso contrario **CCP1** resterà settato (duty 100%).

3. la frequenza del segnale PWM deve essere in funzione delle costanti di tempo del sistema: basteranno centinaia di Hz per una lampada ad incandescenza o un LED, ma potrebbero essere necessari valori particolari per altre applicazioni. Ad esempio, dove sono presenti circuiti magnetici e parti in movimento occorreranno frequenze più alte per evitare risonanze e sfruttare meglio le inerzie meccaniche della rotazione di un motore.

4. per contro, frequenze molto alte possono essere fonte di disturbo elettromagnetico o anche di rendimento ridotto negli attuatori di potenza, con aumento del calore prodotto.

5. L'uso di applet per il calcolo dei parametri per ottenere un certo PWM è opportuno, in quanto evita errori e consente una panoramica rapida delle possibili altrenative.

6. Nella scelta del valore della frequenza va considerato che è può essere necessario evitare valori che ricadano nell'ambito dell'udibile nel caso si abbia a che fare con carichi che producono vibrazioni. Attenzione anche alla presenza di animali che hanno una banda audio più ampia e sono sensibili a suoni tra i 20 e 40kHz.

Può essere opportuno uno studio del migliore mezzo per evitare meccanicamente la trasmissione di vibrazioni all'esterno (ammortizzatori, isolanti, materiali elastici, miglioramento costruttivo, ecc.

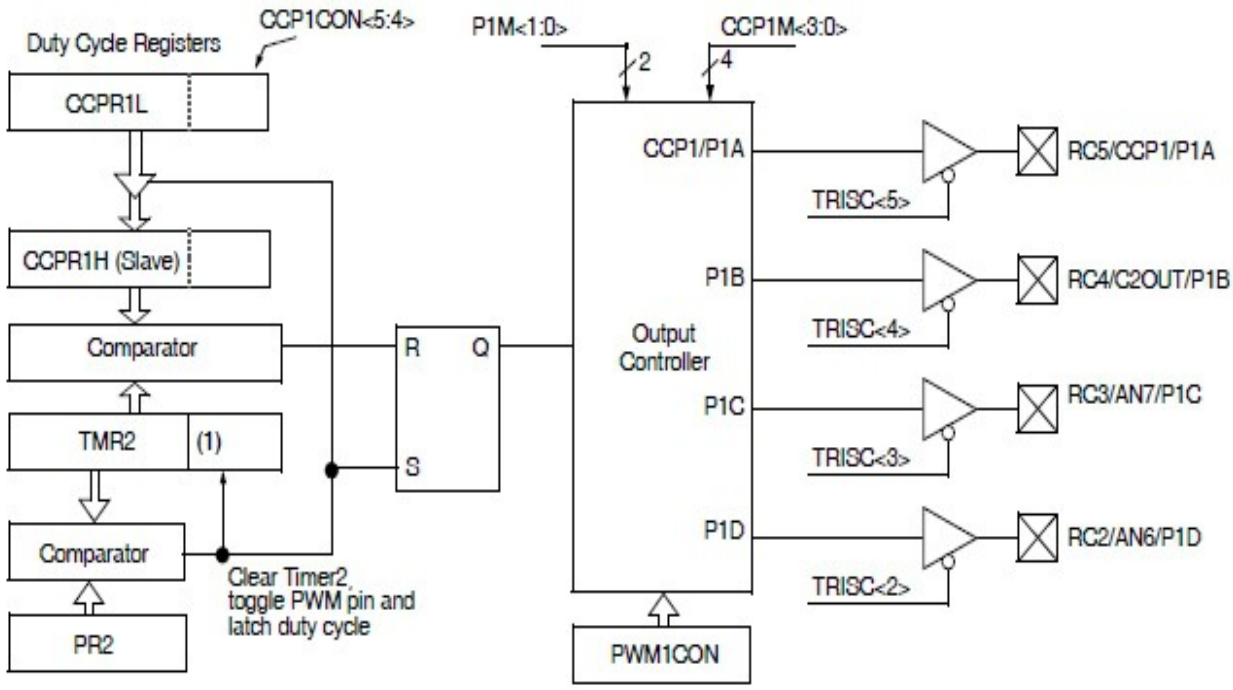
PWM+.

I Midrange sono stati territorio di sperimentazione da parte di Microchip che ha realizzato versioni avanzate delle periferiche integrate.

Anche per il modulo CCP/PWM, c'è una versione enhanced denominata **ECCP/PWM+** e la troviamo installata in chip come **16F685/690/684** ed altri**.**

La periferica è un miglioramento della versione basa; permette di eseguire tutte le funzioni viste prima e dispone di maggiori possibilità.
Si tratta di un modulo di una certa complessità che ha lo scopo di automatizzare quanto più possibile la gestione di carichi di potenza comandati da driver esterni, come solenoidi e motori.

La struttura base del modulo **PWM+** è questa:



La sezione di generazione del clock è quella già vista, basata sul Timer2.

La differenza evidente consiste nella disponibilità di **4 pin di uscita**, gestiti da un ***Output Controller***. Queste uscite sono progettate per il comando di full-bridge e half-bridge (buffer di potenza per motori in corrente continua), oltre che dell'uscita singola della versione non enhanced del modulo.

Il modulo PWM avanzato in grado di generare le seguenti modalità di uscita di cinque PWM:

- **PWM singolo** (con modalità PWM **steering**)
- **Half-bridge PWM**
- **Full-bridge PWM, Modalità Avanti**
- **Full-bridge PWM, Reverse Mode**

La modalità a **PWM singolo** è identica a quanto visto nell'esercitazione precedente.
L'opzione di **steering** permette di impostare uno qualsiasi dei pin come segnale modulato e lo

stesso segnale PWM può essere disponibile simultaneamente su più pin.

Nella modalità **Half-bridge** due pin sono impiegati contemporaneamente come uscite per pilotare driver push-pull. Vengono usati il P1A e il P1B: l'uscita modulata del segnale PWM è sul P1A, mentre il P1B è complementare e sincronizzato, con la possibilità di introdurre tempi morti (dead band) alle commutazioni.

Le modalità **Full-bridge** impiegano tutti e quattro i pin di uscita, dove vengono modulati due segnali PWM.

Se facciamo riferimento al diagramma di principio precedente, è importante notare che:

1. **Tutte le uscite dipendono da un unico timer, quindi la frequenza del PWM sarà la stessa per tutte.**
2. **I latch di queste uscite sono nell'*Output Controller* e non dipendono dai latch dei port.**
3. Il collegamento tra l'uscita PWM e il pin corrispondente dipende, invece, dai TRIS del port: **per ottenere l'uscita occorre portare a 0 il bit relativo nel registro di direzione.**

Il periodo, il duty cycle e la risoluzione sono controllati dai registri comuni alla versione base, dove il timer di riferimento è sempre **Timer2.**

Per informazione, su molti dispositivi più recenti dei Midrange, i progettisti hanno integrato diversi timer (Timer2/4/6) da scegliere per la periferica PWM, che è multipla; questo consente di avere uscite PWM a frequenze diverse. Inoltre, per alcuni di questi chip, è possibile applicare il *Peripheral Pin Select* che permette di assegnare la funzione di uscita ad altri pin.

I registri comuni alla versione base sono:

- registro **`PR2`**
- registro **`T2CON`**
- registro **`CCPR1L`**
- registro **`CCP1CON`**

Il modulo enhanced ha anche caratteristiche aggiuntive come Auto-spegnimento, Auto-restart, dead band che richiedono altri registri:

- registro **`CCP1AS`**
- registro **`PSTR1CON`**
- registro **`PWM1CON`**.

Vediamo in dettaglio i vari registri di controllo.

Quello principale è sempre **`CCP1CON`**:

CCP1CON – ENHANCED CCP OPERATION REGISTER(1) (ADDRESS: 17h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

I bit **DCB1:0** hanno la funzione di bit 1:0 del PWM a 10 bit, come nel modulo base..

I bit **CCP1M3:0** riguardano le modalità di funzionamento del modulo:

CCP1M3:0	Funzione
0000	Modulo CCP disabilitato
0001	riservato
0010	Modo Compare con toggle out
0011	riservato
0100	Modo Capture, ogni fronte di discesa
0101	Modo Capture, ogni fronte di salita
0110	Modo Capture, ogni 4 fronti di salita
0111	Modo Capture, ogni 16 fronti di salita
1000	Modo Compare: set uscita alla comparazione
1001	Modo Compare: clear uscita alla comparazione
1010	Modo Compare: interrupt senza uscita
1011	Modo Compare: trigger per evento speciale
1100	**PWM P1A/1C attivi alto, P1B/1D attivi alto**
1101	**PWM P1A/1C attivi alto, P1B/1D attivi basso**
1110	**PWM P1A/1C attivi basso, P1B/1D attivi alto**
1111	**PWM P1A/1C attivi basso, P1B/1D attivi basso**

Osserviamo che alcune delle combinazioni di bit, riservate nel modulo base, vanno ora a definire una funzione ulteriore del modulo in modo Compare, mentre diventano 4 le modalità PWM.
In sostanza, ECCP/PWM+ riguarda principalmente un ampliamento delle funzioni PWM.

Da notare che le modalità definiscono anche la polarità delle uscite.

Cancellare il registro **CCP1CON** disabilita il modulo e rende inattive le uscite PWM al loro livello di default: il controllo non dipenderà più dall'*Output Controller*, ma dal latch del PORT relativo e la direzione dal TRIS e si potranno impostare le funzioni alternative dei pin.

Però, 4 modalità PWM non esauriscono tutte le possibilità prima elencate, come ad esempio il funzionamento a uscita singola.

Se confrontiamo il registro **CCP1CON** della versione base con quello della versione Enhanced, vediamo che i **bit 7 e 6**, non implementati nella prima, diventano **PM1:0** per l'Enhanced.

Questi due bit si devono aggiungere ai **CCP1M3:0** per la definizione completa della modalità PWM.

I due bit hanno questa funzione:

- per modi non PWM (ovvero **CCP1CON<3:2>=00,01,10**): **P1A** è l'ingresso Capture/Compare. **P1B/C/D** hanno le altre funzioni possibili sui pin
- per modi PWM (ovvero **CCP1CON<3:2>=11**), determinano la situazione delle uscite, secondo la seguente tabella:

PM1:0	Funzione
00	**Uscita singola**: **P1A** (= **CCP1**) modulato, altri con le funzioni alternative
01	**Full-bridge forward**: **P1D** modulato; **P1A** attivo; **P1B, P1C** inattivi
10	**Half-bridge**: **P1A/P1B** modulati con dead band. **P1C/P1D** altre funzioni
11	**Full-bridge reverse**. **P1B** modulato, **P1C** attivo, P1A/P1D inattivi

La situazione dei bit di **CCP1CON** e le funzioni dei pin per il PWM può essere riassunta, quindi, nella seguente tabella:

Modo	CCP1CON	RC5	RC4	RC3	RC2
Singolo PWM	**00xx11xx**	**CCP1**	RC4/C2OUT	RC3/AN7	RC2/AN6
Dual PWM	**10xx11xx**	**P1A**	**P1B**	RC3/AN7	RC2/AN6
Quad PWM	**x1xx11xx**	**P1A**	**P1B**	**P1C**	**P1D**

Ricordiamo che:

Impostando il modo PWM:

- **l'uscita C2OUT deve essere disabilitata, come pure le funzioni analogiche degli altri pin.**
- **I pin usati per il PWM devono essere settati come uscite nel relativo TRIS.**

La sezione di temporizzazione è del tutto identica a quella vista nell'esercitazione precedente: ***Timer 2*** e i registri **PR2** e **CCPR1L** sono gli elementi essenziali e vanno considerati nello stesso modo.
Per quanto riguarda le formule di calcolo vale quanto detto per il modulo PWM base.

Identicamente al modulo base, il periodo del PWM è dato da:

$$periodo\ PWM = [(PR2)+1]*4*Tosc*(TMR2\ prescale\ value)$$

La seguente equazione è utilizzata per calcolare il duty cycle:

$$PWM\ duty\ cycle = (DCxB9:DCxB0) \bullet Tosc \bullet (TMR2\ prescaler)$$

I bit **DCxB9:DCxB0** possono essere scritti in qualsiasi momento, ma il duty cycle valore non è agganciato in **CCPRxH** fino a che la comparazione tra **PR2** e **TMR2** si verificata, ovvero alla fine del corrente periodo.

Il rapporto del duty cycle è dato da:

Duty Cycle Ratio = (DCxB9:DCxB0) / 4(PR2+1)

Non è possibile la massima risoluzione a 10bit per qualsiasi frequenza di PWM; essa è limitata da

log (fosc/Fpwm) / log(2) in bit

Quando si impiega una delle modalità PWM, solitamente all'esterno del microcontroller sono collegati driver di potenza per comandare il carico. L'hardware esterno deve utilizzare il corretto pull-up o pull-down sui pin di uscita.

I bit **CCP1M<1:0>** del registro **CCP1CON** permettono all'utente di scegliere se i segnali di uscita sono attivi a livello alto o basso per ogni coppia di pin di uscita PWM (**P1A/P1C** e **P1B/P1D**).

Le polarità di uscita devono essere selezionate **prima** che i pin di uscita siano attivati.

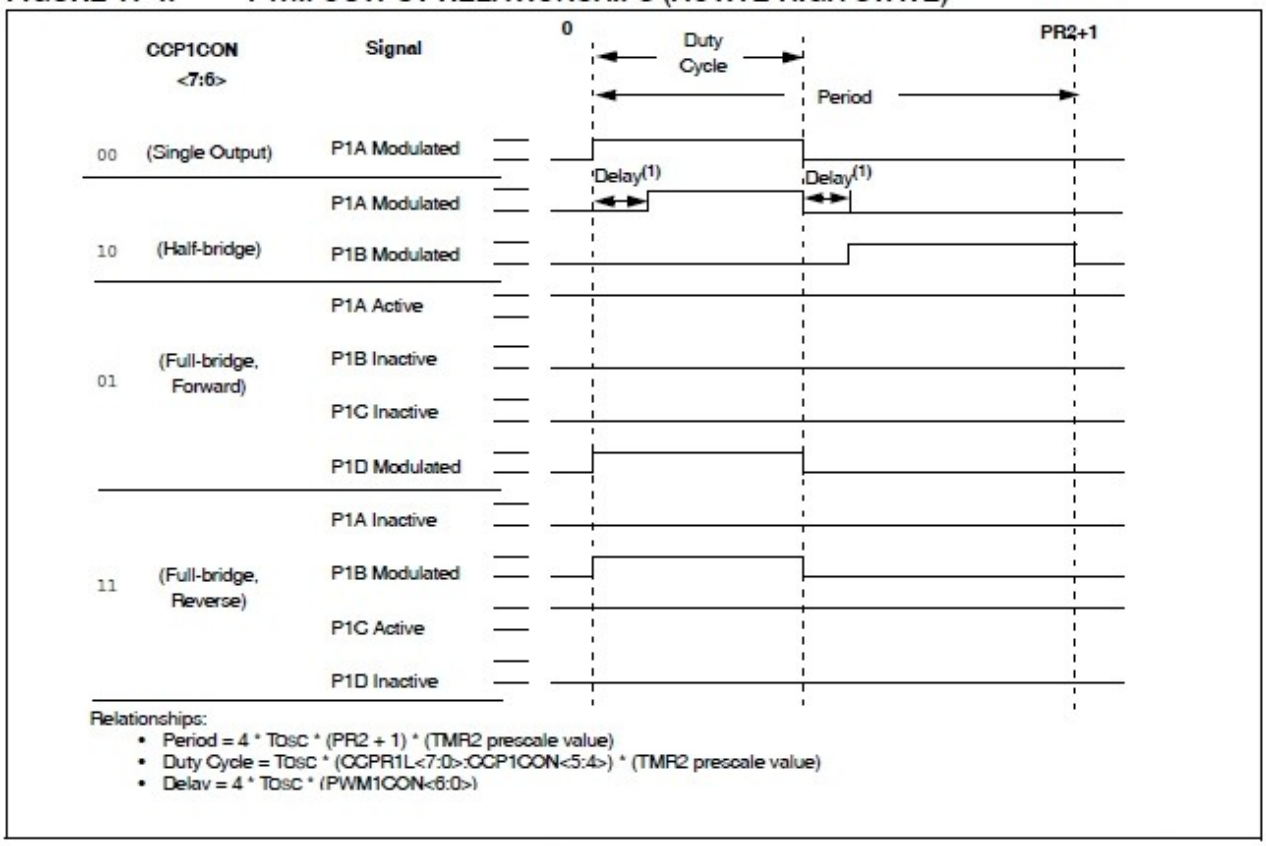
Modificare la configurazione delle polarità mentre i driver di uscita pin PWM sono abilitati è sconsigliato in quanto può danneggiare i circuiti esterni dell' applicazione.

Le modalità PWM devono essere attivate nel modo desiderato e sia completato un ciclo completo PWM prima di abilitare le uscite.
Il completamento di un ciclo PWM può essere determinata monitorando il bit di overflow TMR2IF che verrà portato a 1 all'inizio di un nuovo periodo PWM.

PWM+ singolo.

Le relazioni tra i segnali in uscita nelle varie modalità sono elencati nel diagramma seguente:

FIGURE 11-4: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)



L'uscita singola ricalca esattamente quanto visto nell'esercitazione precedente: un solo pin **`P1A`** viene azionato; gli atri **`P1x`** sono disponibili come I/O digitali o per le altre funzioni che supportano.

`CCP1CON` deve essere programmato con **`00yy11xx`** dove **`yy`** sono i bit **`DC1B1:0`**della risoluzione a 10bit.

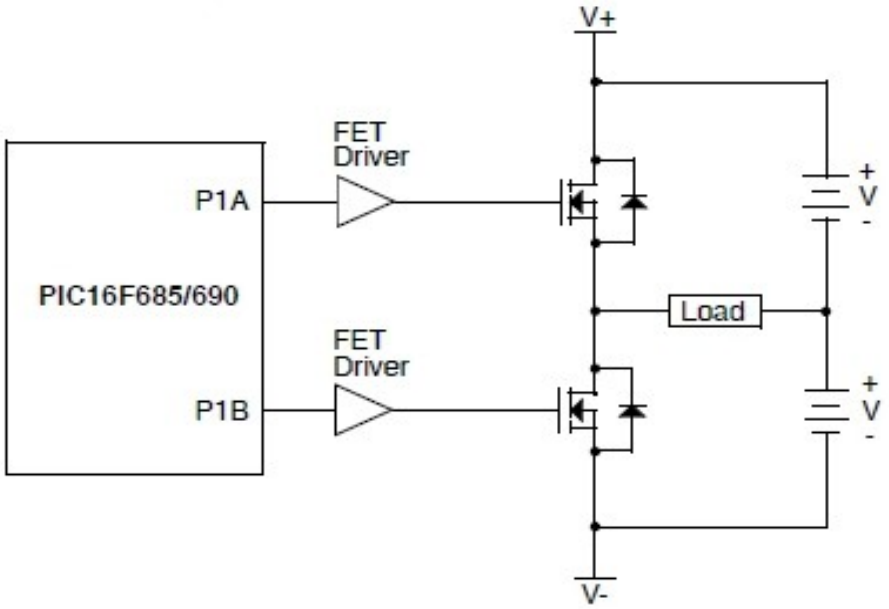
Alla modalità a singola uscita si potrà sovrapporre la funzione di steering, che sarà trattata più avanti.

Half-bridge.

Nella modalità **Half-bridge**, solo due due pin sono usati come uscite: l'uscita PWM viene emessa sul pin **RC5/CCP1/P1A**, mentre l'uscita PWM **complementare** è sul pin **RC4/C2OUT/P1B**.

CCP1CON deve essere programmato con **10yy11xx** dove **yy** sono i bit **DC1B1:0**della risoluzione a 10bit e **xx** identificano la polarità delle uscite.

Tipicamente un **half-bridge** è integrato con un una uscita di potenza del genere **push-pull** che permette di far circolare la corrente nel carico nei due sensi:



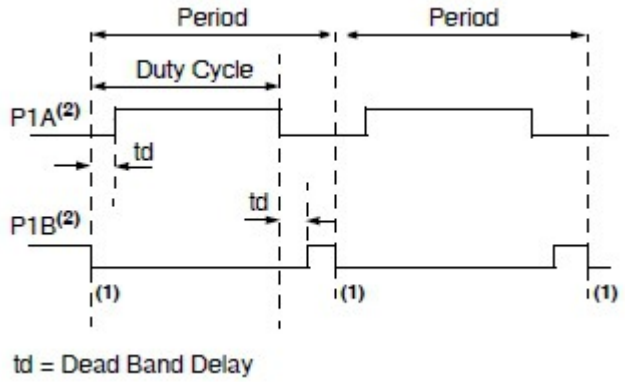
Due elementi di potenza, in questo caso presentato come MOSFET, comandano la corrente nel carico. L'alimentazione richiesta è duale.

Osserviamo che per il comando dei MOSFET **sono necessari appositi gate drivers**, a meno di utilizzare più costosi MOSFET con Logic Gate, pilotabili direttamente dalle uscite del microcontroller.

Se applichiamo direttamente il segnale PWM in uscita da **P1A** e **P1B** avremo che, nel periodo, andrà in conduzione il MOSFET superiore, facendo scorrere la corrente in un verso, quindi andrà in conduzione quello inferiore, invertendo il senso della corrente.

Se entrambi i MOSFET di un half-bridge sono scambiati nella conduzione (uno acceso e l'altro spento che si invertono) nello stesso istante, è possibile che quello che viene spento resti in conduzione per breve periodo di tempo, mentre l'altro viene acceso: durante questo breve intervallo in cui entrambi sono in conduzione, una corrente molto elevata può fluire in entrambi i MOSFET, cortocircuitando l'alimentazione del ponte e danneggiando i semiconduttori.

Le forme d'onda sono queste:



Sono da notare i ritardi ***td*** che possono essere programmati allo scambio tra due uscite.

Questi ritardi sono necessari per evitare l'evento prima citato della conduzione contemporanea, anche per breve perioro, dei due MOSFET. Il ritardo è inserito durante la commutazione in modo tale che uno dei driver sia normalmente ritardato per consentire all'altro di passare completamente in off.

Questo ritardo (***dead band***) è programmabile e viene inserito nel passaggio dal segnale non attivo allo stato attivo, come indicato nel diagramma precedente; l'opzione è gestita con un registro apposito:

PWM1CON – ENHANCED PWM CONFIGURATION REGISTER(1) (ADDRESS: 1Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
bit 7							bit 0

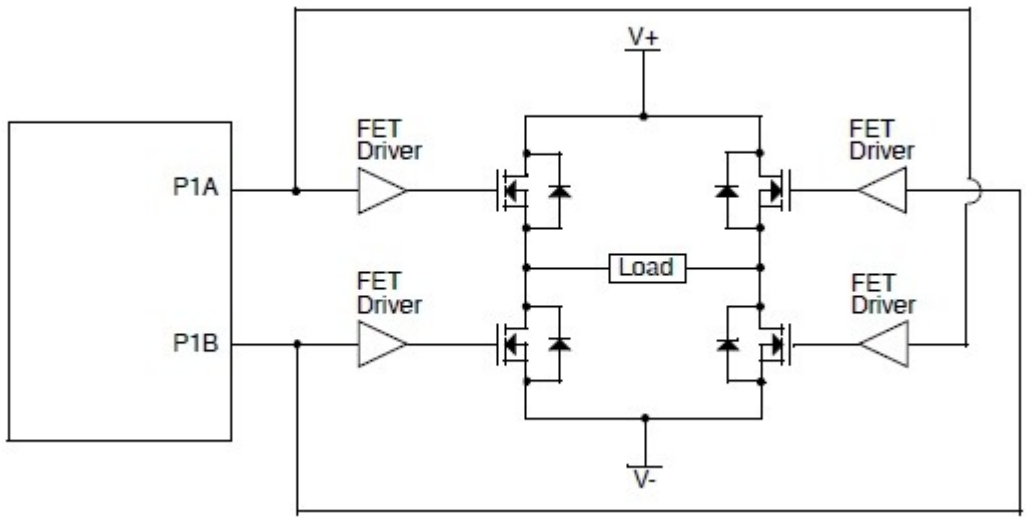
I primi 7 bit, **`PDC6:0`** determinano il ritardo in cicli di istruzione (***Fosc/4***).

Il numero dei cicli introdotti equivale al valore assegnato a questi bit, ovvero da 0000000 a 1111111, cioè tra 0 e 127 decimale. Ad esempio, a 4MHz di clock, il ritardo introdotto sarà programmabile tra 0 e 127us.

Se il valore è maggiore del ciclo di lavoro, l'uscita corrispondente rimane inattiva durante l'intero ciclo.

Il bit 8, **`PRSEN`** ha la funzione di abilitare un ***auto shutdown / auto restart*** che permette di spegnere o riavviare automaticamente il PWM a seguito di un “evento speciale”, azione che vediamo più avanti.

Può non essere desiderabile avere una alimentazione duale. Si potrà, allora, usare un doppio half-bridge, che consente di avere una alimentazione singola.



Vengono portati in conduzione prima i MOSFET comandati da **`P1A`** e poi quelli comandati da **`P1B`**, alternando la polarità applicata al carico con un trasferimento di energia proporzionale al duty cycle.

Si possono realizzare driver di potenza anche con transistor BJT o con circuiti integrati appositi.

Il carico potrebbe anche essere un motore, ma la situazione è abbastanza inefficiente: se il duty cycle è il 50%, la condizione attiva sarà uguale per entrambe le uscite e il carico riceverà corrente in un senso e in quello opposto in ugual misura. Se applichiamo la cosa ad un motore cc, esso sarà sollecitato in uguale misura a ruotare in un senso e in quello opposto.

Se la frequenza del PWM è bassa, si mostrerà un evidente pendolamento o vibrazioni che possono danneggiare il motore; è richiesta una frequenza abbastanza elevata da far si che la combinazione dell'inerzia meccanica e dell'induttanza degli avvolgimenti faccia si che il rotore sia fermo.

Se applichiamo un duty cycle diverso, si avrà la preponderanza del tempo in cui ai capi del motore appare una tensione rispetto a quello in cui appare la tensione opposta: con il 20% di duty cycle, avremo il MOSFET superiore in conduzione per il 20% del periodo e quello inferiore per l'80%. Ne risulta che la rotazione sarà forzata in una direzione. Con un duty dell'80% avremo l'opposto e la rotazione sarà invertita. Quanto maggiore è il tempo per cui è applicata una direzione della corrente, tanto maggiore sarà la velocità.

Con un duty del 100% sarà in conduzione sempre e solo il MOSFET superiore e con lo 0% solo quello inferiore.

Per riassumere:

Duty cycle	Movimento
0%	Reverse, massima velocità
>0% ... <50%	Reverse, velocità in riduzione
50%	Fermo
>50% ... <100%	Forward, velocità in crescita
100%	Forward, massima velocità

Un controllo del genere prende il nome di ***Locked Anti-phase PWM – LAP***).

E' evidente che, per un motore, si tratta di una soluzione poco efficiente, mentre ha applicazioni con altri tipi di carico.

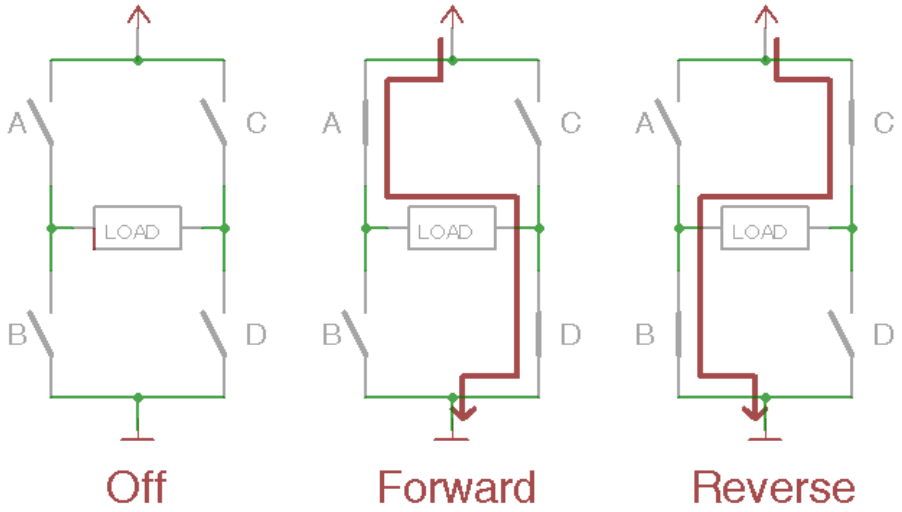
Se vogliamo controllare la direzione di rotazione di un motore in modo più funzionale è opportuno impiegare la configurazione Full-bridge.

Full-Bridge.

Nella modalità di uscita a ponte intero (Full-bridge), il bit **P1M1** (**CCP1CON<7>**) permette all'utente di controllare la direzione.

CCP1CON deve essere programmato con **01yy11xx** per il modo forward e **11yy11xx** per il modo reverse, dove **yy** sono i bit **DC1B1:0**della risoluzione a 10bit e **xx** identificano la polarità delle uscite.

L'hardware tipico di ponte full.bridge o H-bridge è composto da switch che si possono chiudere a coppie, alimentando un carico:



Gli interruttori sono denominati **A, B, C** e **D**.

Se tutti e quattro sono aperti, nessuna corrente scorre nel carico.
Se sono chiusi **A** e **D**, al carico è applicata una tensione e la corrente scorre in un senso, detto ***forward*** (avanti)
Se sono chiusi **B** e **C**, al carico è applicata la tensione opposta e la corrente scorre in senso inverso (***reverse***)..

Questo permette di invertire la direzione di un motore applicato come carico.

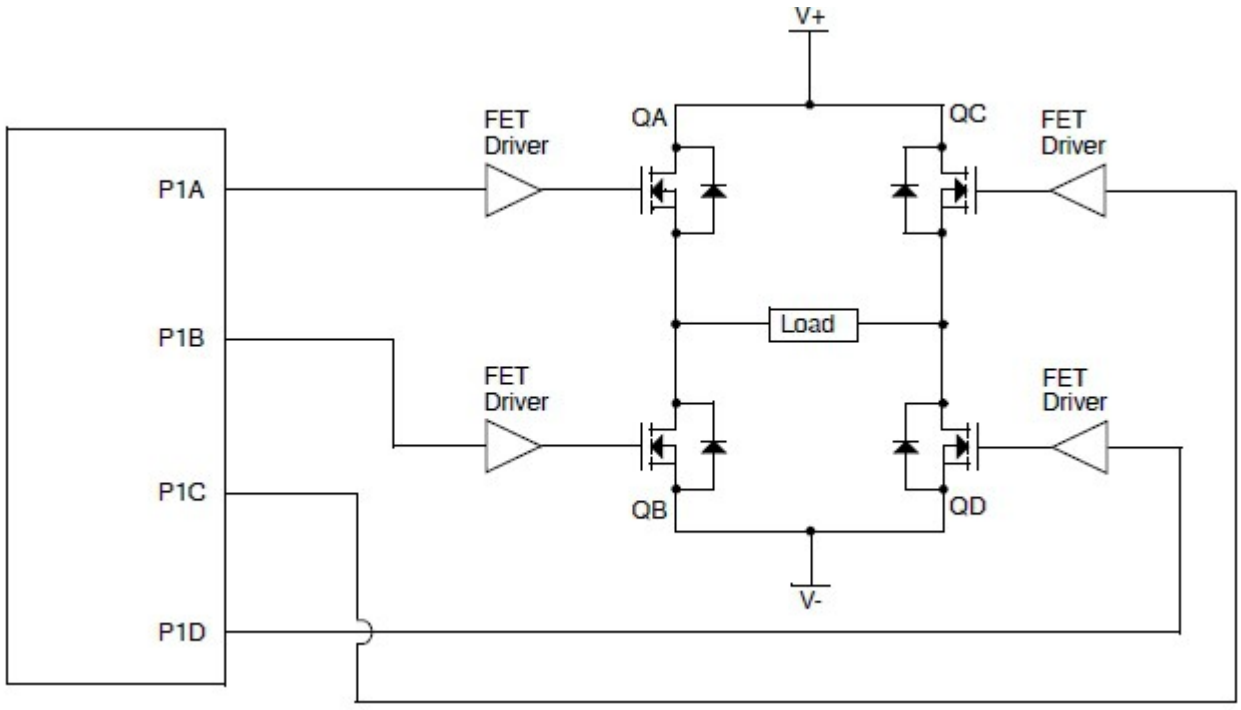
Se, in aggiunta, moduliamo con un PWM uno switch (tipicamente **D** nel forward e **B** nel reverse possiamo anche variare la velocità di rotazione.

Ovviamente non deve mai prodursi la situazione in cui **A / B** o **C / D** sono entrambi chiusi: questo metterebbe in corto circuito l'alimentazione.

Invece è possibile ere contemporaneamente **B** / **D** (oppure **A** / **C**) per produrre una frenatura elettrica del motore che si troverebbe chiuso in corto circuito.

Dal punto di vista realizzativo, il ponte ad H si può costruire utilizzando al posto degli interruttori dei transistor come MOSFET o BJT, che garantiscono velocità di commutazione elevate.
Sono possibili diverse topologie, usando BJT N e P o MOSFET N e P.

Qui un esempio di H-bridge realizzato con MOSFET a canale N.



Nell'H-bridge tutte e quattro le uscite PWM sono utilizzate.
Osserviamo che, a meno di utilizzare MOSFET detti Logic Level, ovvero modelli specificamente realizzati per essere pilotati direttamente dai pin del microcontroller, occorre inserire opportuni gate driver.

`CCP1CON` deve essere programmato con **`01yy11xx`** per il modo forward e **`11yy11xx`** per il modo reverse, dove **`yy`** sono i bit **`DC1B1:0`**della risoluzione a 10bit e **`xx`** identificano la polarità delle uscite.

Il **`P1A,P1B,P1C`** e **`P1D`** richiedono che i bit **`TRIS`** associati siano a livello 0 (uscite).

Osserviamo che i bit1 e 0 del **`CCP1CON`** consento di avere polarità di uscita diverse:

CCP1M3:0	Funzione
11**00**	**PWM P1A/1C attivi alto, P1B/1D attivi alto**
11**01**	**PWM P1A/1C attivi alto, P1B/1D attivi basso**
11**10**	**PWM P1A/1C attivi basso, P1B/1D attivi alto**
11**11**	**PWM P1A/1C attivi basso, P1B/1D attivi basso**

Questo consente di adattarsi a diverse tipologie di ponti e relative polarità del driver (senza richiedere driver esterni invertenti). Così si possono pilotare ponti realizzati con sia con componenti N che P o con un mix dei due.

A questo si aggiunge l'informazione imposta dai bit PM1:0 che modificano la direzione di rotazione (***forward*** = **avanti**, ***reverse*** = **indietro**):

PM1:0	Funzione
01	**Full-bridge forward**: `P1D` modulato; `P1A` attivo; `P1B/P1C` inattivi
11	**Full-bridge reverse**. `P1B` modulato, `P1C` attivo, `P1A/P1D` inattivi

Ovviamente, le definizioni ***forward*** e ***reverse*** sono standard e non sono relative alla situazione meccanica di uno specifico motore: indicano che in un modo la polarità applicata sarà all'opposto dell'altro modo. Il senso di rotazione sarà relativo all'applicazione.

Osserviamo che la modulazione PWM è imposta ai driver dei rami inferiori del ponte. I transistor sui rami superiori fungono da switch per il percorso della corrente.

Possiamo raccogliere in una tabella le condizioni delle uscite.

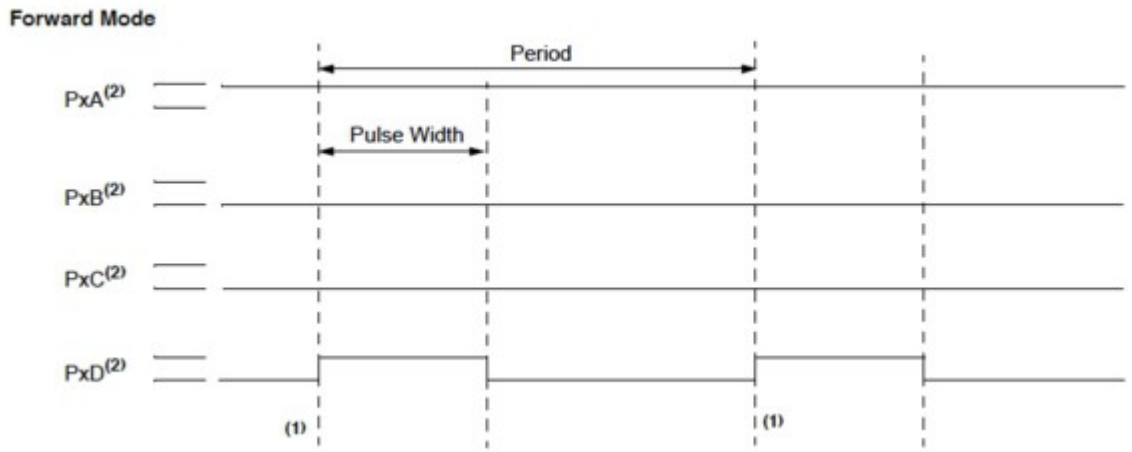
Le **coppie funzionali** sono **P1A/P1D** e **P1B/P1C** che si scambiano la modalità attiva/inattiva a seconda della direzione imposta.

Le **coppie attivi/inattivi**, invece, sono **P1A/1C** e **P1B/1D** ovvero un driver alto con quello basso del ramo opposto.

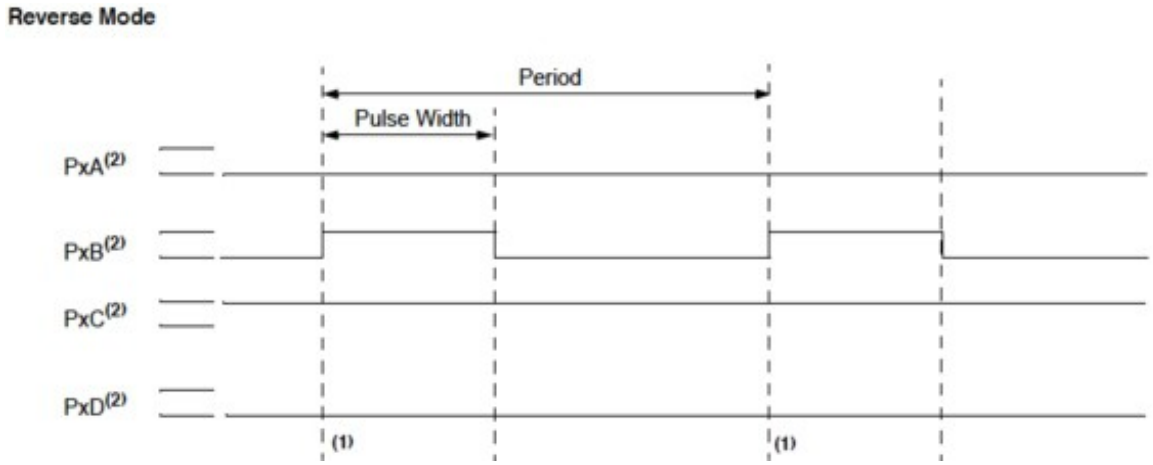
CCP1M3:0	Livello logico	Direzione	P1A	P1B	P1C	P1D
1100	**P1A/1C attivi H** **P1B/1D attivi H** P1A/1C inattivi L P1B/1D inattivi L	***Forward***	attivo **H**	inattivo **L**	inattivo **L**	attivo **PWM**
		Reverse	inattivo **L**	attivo **PWM**	attivo **H**	inattivo **L**
1101	**P1A/1C attivi H** **P1B/1D attivi L** P1A/1C inattivi L P1B/1D inattivi H	***Forward***	attivo **H**	inattivo **H**	inattivo **L**	attivo **PWM**
		Reverse	inattivo **L**	attivo **PWM**	attivo **H**	inattivo **H**
1110	**P1A/1C attivi L** **P1B/1D attivi H** P1A/1C inattivi H P1B/1D inattivi L	***Forward***	attivo **L**	inattivo **L**	inattivo **H**	attivo **PWM**
		Reverse	inattivo **H**	attivo **PWM**	attivo **L**	inattivo **L**
1111	**P1A/1C attivi L** **P1B/1D attivi L** P1A/1C inattivi H P1B/1D inattivi H	***Forward***	attivo **L**	inattivo **H**	inattivo **H**	attivo **PWM**
		Reverse	inattivo **H**	attivo **PWM**	attivo **L**	inattivo **H**

Graficamente due esempi per `CCP1M<3:0> = 1100` , condizione adatta per pilotare il ponte della figura precedente:

Nella modalità direzione in avanti (***forward***), il pin `P1A` è posto allo stato attivo a livello alto e il pin `P1D` è modulato dal PWM, mentre `P1B` e `P1C` sono inattivi livello basso



Nella modalità direzione ***reverse***, **`P1C`** è allo stato attivo a livello alto e **`P1B`** è modulato dal PWM, mentre **`P1A`** e **`P1D`** sono inattivi a livello basso.

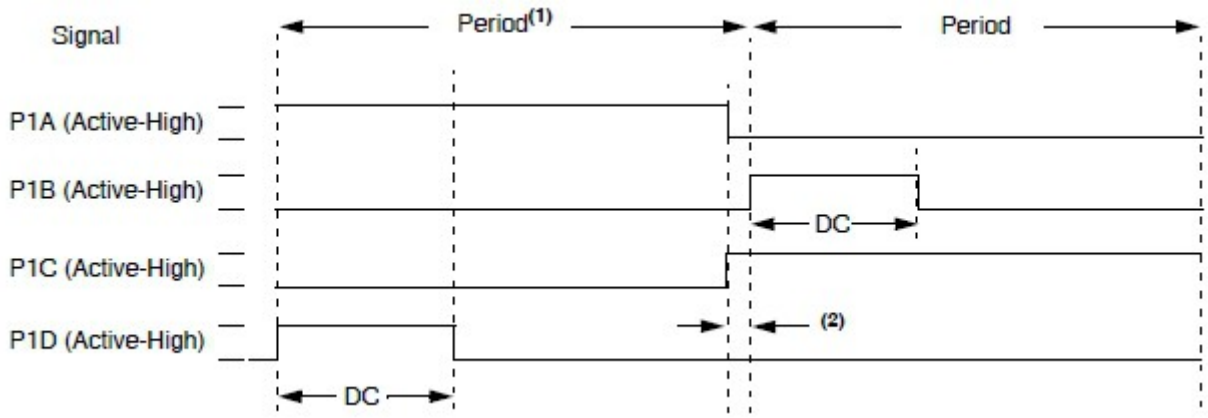


Dato che il cambio di direzione avviene agendo sul bit **`PM1`** nel registro **`CCP1CON`**, basterà invertire questo bit per invertire la direzione.

PM1:0	Funzione
01	***forward***
11	***reverse***

Quando si cambia direzione, le coppie di pin sono scambiate e la modulazione PWM riprende all'inizio del prossimo periodo.

Ciò si verifica in un intervallo di tempo di ***(4 TOSC * (Timer2prescaler))*** prima del prossimo periodo di PWM.

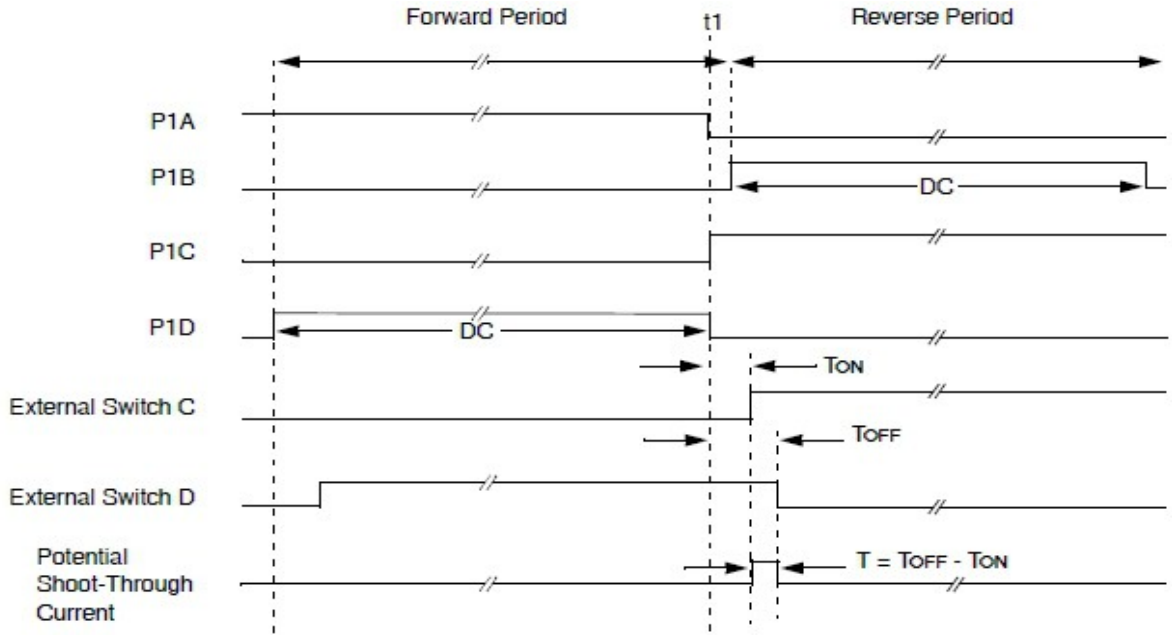


(2) quando avviene il cambio di direzione, **`P1A`** e **`P1C`** vengono commutati alla fine del periodo corrente con un intervallo di ***4Tosc***, ***16Tosc*** o ***64Tosc*** a seconda del valore del prescaler del Timer2. I segnali modulati **`P1B`** e **`P1D`** sono inattivi durante questo tempo.

Si noti che nella modalità di uscita full-bridge il modulo non fornisce alcuno specifico ritardo di dead band: dato che una sola uscita è modulata in ogni momento, questo ritardo non è necessario.

Tuttavia, c'è una situazione in cui potrebbe essere necessario, ovvero quando:
1. La direzione di uscita cambia quando il duty cycle è pari o vicino al 100%.
2. Il tempo di spegnimento dell'interruttore di potenza, compreso il circuito di pilotaggio, è maggiore del tempo di on.

La figura seguente mostra un esempio dove avviene il cambio di direzione per un PWM di quasi il 100%.



Al tempo ***t1***, **`P1A`** e **`P1D`** diventano inattivi, mentre diventa attivo **`P1C`**. In questo esempio, poiché il tempo di spegnimento dei driver è più lungo del tempo di accensione, si determina una finestra di tempo in cui la corrente può attraversare i MOSFET QC e QD, entrambi accesi. Lo stesso fenomeno si verifica per dispositivi di potenza QA e QB nel cambio di direzione inverso.

Se è necessario cambiare direzione PWM duty cycle elevato, occorre applicare una delle seguenti azioni:

1. Ridurre duty cycle PWM per un periodo di PWM prima del cambio di direzione.
2. Usare driver che possono portare in off i dispositivi di potenza in un tempo minore di quello di on.

Auto shutdown/Auto restart

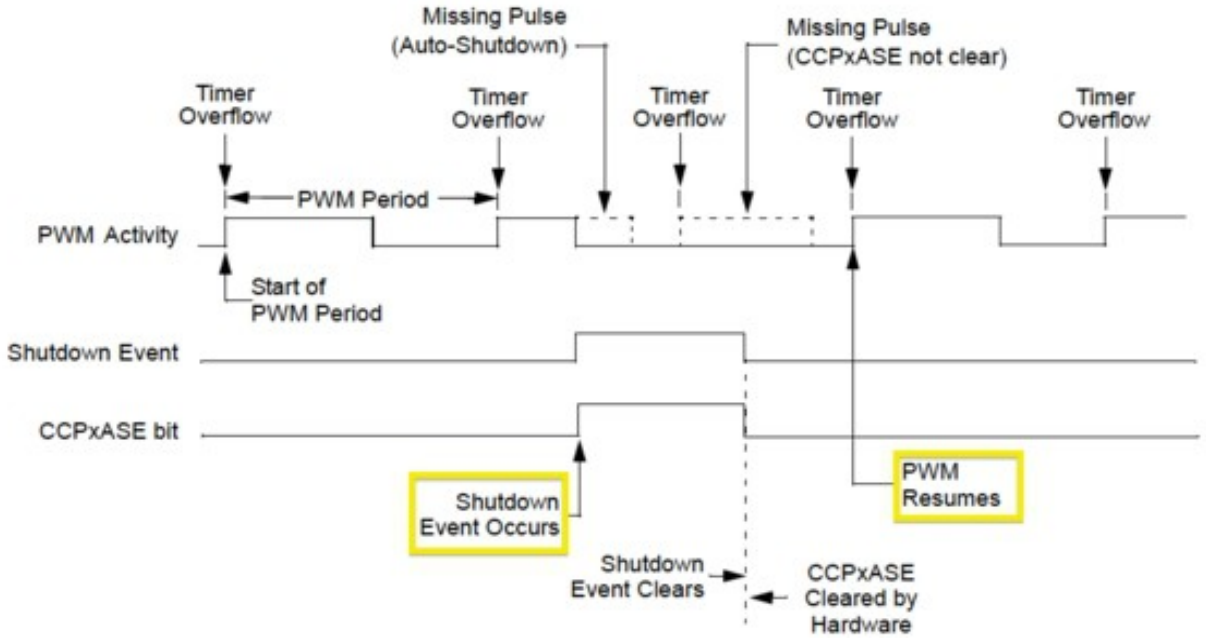
Il modulo PWM Enhanced supporta la funzione di auto spegnimento (auto shutdown) e di auto riavvio (auto restart).

La funzione è disponibile per qualunque dei modi PWM che sono programmati.

Auto-Shutdown disattiva le uscite PWM quando si verifica un evento di arresto esterno e posiziona i pin in una situazione pre determinata. Questa funzione viene utilizzata per prevenire danno quanto collegato esternamente.

Ad esempio, il ponte controllato può disporre di una uscita che indica una sovra corrente dovuta al blocco del rotore del motore. In questo caso il modulo PWM sospende la modulazione e dispone le uscite in modo da cessare di far scorrere corrente nel carico.

La figura seguente mostra le forme d'onda relative:



Le fonti di auto-spegnimento sono selezionate usando un apposito registro:

ECCPAS – ENHANCED CAPTURE/COMPARE/PWM+ AUTO-SHUTDOWN CONTROL REGISTER(1) (ADDRESS: 1Dh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0
bit 7							bit 0

Si possono selezionare diverse fonti:

- un livello 0 sul pin INT
- livelo 1 su uno o l'altro comparatore
- una combinazione delle due precedenti

I bit interessati sono **ECCPAS<2:0>**, secondo la seguente tabella:

ECCPAS<2:0>	Funzione
000	Auto shutdown disabilitato
001	Uscita comparatore 1
010	Uscita comparatore 2
011	Uno o l'altro dei comparatori
100	Livello logico 0 sul pin INT
101	Livello logico 0 sul pin INT o comparatore 1
110	Livello logico 0 sul pin INT o comparatore 2
111	Una o l'altra delle situazioni

La scelta del comparatore consente di monitorare internamente il livello di una tensione analogica esterna, ad esempio la caduta di tensione su una resistenza di misura della corrente senza l'aggiunta di componenti esterni.

La scelta del pin **INT** consente di accettare un segnale di shutdown a livello logico proveniente dall'esterno.

La condizione di blocco è indicata dal bit **ECCPASE.**
- Se il bit è '0', i pin funzionano normalmente.
- Se il bit è '1', le uscite PWM sono nello stato di blocco.

Quando si verifica un evento di arresto, accadono due cose:

1. Il bit **ECCPASE** è impostato su '1' e rimarrà impostato fino a che è cancellato nel firmware o si verifica un riavvio automatico.
2. I pin PWM abilitati vengono posizionati in modo asincrono nei loro stati di arresto.

I pin di uscita PWM sono raggruppati in coppie [**P1A/P1C**] e [**P1B/P1D**].
Lo stato di ciascuna coppia è determinata dai bit **PSSAC1:0** e **PSSBD1:0** secondo la seguente logica:

- **PSSAC1:0**
  - **00** P1A/P1C = 0
  - **01** P1A/P1C = 1
  - **1x** P1A/P1C in tri-state

- **PSSBD1:0**
  - **00** P1B/P1D = 0
  - **01** P1B/P1D = 1
  - **1x** P1B/P1D in tri-state

Non sono disponibili tutte le combinazioni possibili e immaginabili, come non lo sono per le polarità selezionabili con i bit **CCP1M3:0** : si dovrà adeguare la scelta dei driver invertenti o non invertenti per comandare gli elementi di potenza.
In particolare, nel caso di uso della condizione tri-state, ad alta impedenza, occorrerà che l'ingresso dei driver esterni sia dotato di pull-up o pull-down appropriati a garantire il livello logico voluto.

All'automatismo di arresto si aggiunge quello di riavvio: il modulo può essere configurato per riavviare automaticamente il segnale PWM una volta che la condizione di auto-spegnimento è stato rimossa.

Auto-restart è attivato impostando il bit **PRSEN** nel registro **PWM1CON**.
Se il riavvio automatico è abilitato, il bit **ECCPASE** rimane impostato fino a quando la condizione di auto-spegnimento è attivo.
Quando la condizione di auto-spegnimento viene rimossa, il bit **ECCPASE** verrà cancellato via hardware e il funzionamento normale riprenderà.

Questo può essere osservato nella figura precedente. In realtà, il riavvio automatico si verifica in realtà subito dopo il **ECCPASE** viene cancellato, ma la forma d'onda PWM non può iniziare fino alla fine del periodo corrente (overflow del Timer2).

Le due azioni sono utili per implementare una sicurezza, ad esempio, sul comando di motori elettroutensili: il modulo PWW rileva un blocco della rotazione e disattiva il motore. Nel momento in cui la condizione di errore viene rimossa, il funzionamento riprende normalmente in modo automatico, il tutto con la supervisione possibile da parte del firmware del microcontroller.

Pulse steering.

Una ulteriore funzione del PWM+ è il ***Pulse Steering***, che è attivabile quando il PWM è abilitato in modalità singola uscita.

Il pin **P1A** è l'uscita modulata a larghezza di impulso. Tuttavia, il segnale PWM ciò può essere indirizzato ad altri I/O contemporaneamente.

La funzione è gestita da un apposito registro:

PSTRCON – PULSE STEERING CONTROL REGISTER(1, 2) (ADDRESS: 19Dh)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	STRSYNC	STRD	STRC	STRB	STRA
bit 7							bit 0

I bit interessati sono **STRD,SRTC,STRB,STRA**, secondo questa tabella:

STRD	STRC	STRB	STRA	P1D	P1C	P1B	P1A
0	0	0	0	Port	Port	Port	Port
0	0	0	1	Port	Port	Port	P1A
0	0	1	0	Port	Port	P1B	Port
0	0	1	1	Port	Port	P1B	P1A
0	1	0	0	Port	P1C	Port	Port
0	1	0	1	Port	P1C	Port	P1A
0	1	1	0	Port	P1C	P1B	Port
0	1	1	1	Port	P1C	P1B	P1A
1	0	0	0	P1D	Port	Port	Port
1	0	0	1	P1D	Port	Port	P1A
1	0	1	0	P1D	Port	P1B	Port
1	0	1	1	P1D	Port	P1B	P1A
1	1	0	0	P1D	P1C	Port	Port
1	1	0	1	P1D	P1C	Port	P1A
1	1	1	0	P1D	P1C	P1B	Port
1	1	1	1	P1D	P1C	P1B	P1A

L'indicazione **Port** specifica che il pin è disponibile per le sue altre funzioni.

Osserviamo che, per default, viene settata come uscita **P1A**, in modo compatibile col **CCP1** del modulo base. Inoltre, attraverso i 4 bit **STRx** è possibile:

- annullare tutte le uscite
- attribuire la funzione di **CCP1** ad uno qualunque dei quattro **P1x**
- avere l'uscita modulata su una combinazione dei 4 pin

Questo consente una ampia possibilità di gestione dei pin di uscita del segnale PWM.

E' possibile anche una sicronizzazione, controllata dal bit **STRSYNC** che fornisce all'utente due opzioni per il momento in cui avviene lo steering:

- se **STRSYNC** è '0', l'evento avverrà alla fine dell'istruzione che scrive al registro **PSTRCON**. In questo caso il segnale di uscita al pin **P1<D:A>** può essere una forma d'onda PWM incompleto. Questa situazione è utile quando il firmware utente deve rimuovere immediatamente un segnale PWM dal pin.

- se **STRSYNC** è '1', l'aggiornamento avverrà all'inizio del successivo periodo PWM. In questo l'uscita PWM produrrà sempre una forma d'onda completa.

PWM e risoluzione.

Parlando della risoluzione in bit, abbiamo indicato che essa arriva a 10bit, ma dipende dalla relazione

risoluzione = log (4(PR2+1))) / log(2)

ovvero è legata al valore di PR2, il quale a sua volta dipende dalla Fosc e dal valore di frequenza che si vuole ottenere:

frequenza PWM = Fosc / (4* (PR2+1) * prescaler)

In relazione a quanto detto, è utile consultare una delle applet, ad esempio **questa**, che fornisce una panoramica interessante delle possibilità di impostazione per ottenere una certa frequenza.

Attraverso questa, vediamo, ad esempio, quale sia il senso della risoluzione in bit.

Prendiamo per esempio la generazione di un PWM a 500Hz con una Fosc di 4MHz.
L'applet indica che è possibile ottenere il valore preciso con una risoluzione a 10bit.

PWM		TIMER2 Prescaler	REGISTERS			
Frequency (Herz)	Resolution (Bits)		PR2	T2CON	CCPR1L	CCP1CON
477.10	10	÷16	0b10000010	0b00000111	0b01000001	0b00011100
480.77	10	÷16	0b10000001	0b00000111	0b01000000	0b00111100
484.50	10	÷16	0b10000000	0b00000111	0b01000000	0b00011100
488.28	10	÷16	0b01111111	0b00000111	0b00111111	0b00111100
492.13	10	÷16	0b01111110	0b00000111	0b00111111	0b00011100
496.03	10	÷16	0b01111101	0b00000111	0b00111110	0b00111100
500.00	10	÷16	0b01111100	0b00000111	0b00111110	0b00011100
504.03	10	÷16	0b01111011	0b00000111	0b00111101	0b00111100
508.13	10	÷16	0b01111010	0b00000111	0b00111101	0b00011100
512.30	10	÷16	0b01111001	0b00000111	0b00111100	0b00111100
516.53	10	÷16	0b01111000	0b00000111	0b00111100	0b00011100
520.83	10	÷16	0b01110111	0b00000111	0b00111011	0b00111100

Osserviamo che il valore da applicare a **`PR2`** è pari a 124, con un prescaler di1:16 per il Timer2.
In effetti:

frequenza PWM = Fosc / (4* (PR2+1) * prescaler) = 4000000 / (4*(124+1) *16) = 500Hz

Il valore da caricare in **`DCB9:0`**, ovvero **`CCPR1L`** e **`CCP1CON5:4`** è di 249. Abbiamo detto che

Duty Cycle Ratio = (DCxB9:DCxB0) / 4 * (PR2+1)

ovvero:

Duty Cycle Ratio * 4 * (PR2+1) = (DcxB9:DCxB0)

e anche questo è confermato:

(DcxB9:DCxB0) = Duty Cycle Ratio * 4 * (PR2+1) = 0.5 * 4 * (124+1) = 249

In effetti, dato che il periodo di **Fosc/4** è 1us, occorrono 249 (+1)us per ottenere i 500Hz voluti.

Se verifichiamo per il duty cycle pari a 0, otteniamo ***DcxB9:DCxB0=0***.
Per il duty ccyle del 100% abbiamo ovviamente ***DcxB9:DCxB0=499*** , così divisi:

CCP1RL = 01111100 CCP1CON = 00111100

0111110011 = 499

Quindi il senso della risoluzione a 10bit non è che sono necessari tutti e dieci I bit, ma che ne possono essere impeganti anche una quantità minore, superiore comunque a 8 bit.

Questo è importante se, ad esempio, vogliamo scalare un numero a 8 bit per usarlo come carico di **CCPR1L**.

E' importante anche considerare che un aumento della frequenza del PWM rispetto alla Fosc tende a diminuire la risoluzione.

Ad esempio, se invece di 500Hz volessimo ottenere 50kHz, avremmo la seguente situazione:

PWM		TIMER2 Prescaler	REGISTERS			
Frequency (Herz)	Resolution (Bits)		PR2	T2CON	CCPR1L	CCP1CON
47619.05	6	÷1	0b00010100	0b00000100	0b00001010	0b00011100
50000.00	6	÷4	0b00000100	0b00000101	0b00000010	0b00011100
50000.00	6	÷1	0b00010011	0b00000100	0b00001001	0b00111100

La frequenza è ottenibile, ma con una risoluzione di soli 6 bit. Notiamo che una diversa configurazione del prescaler consente di ottenere comunque valori abbastanza vicini a quello voluto, che, comunque, in questo caso con 1:4 è ottenibile con precisione.

Se verifichiamo per il duty cycle pari a 0, otteniamo ***DcxB9:DCxB0=0***.
Per il duty ccyle del 100% abbiamo ovviamente ***DcxB9:DCxB0=19***

In questo caso è evidente che la definizone del PWM è estrememamente ridotta. Valori maggiori di 19 imporranno comunque un duty del 100% e sono inutilizabili.

Se volessimo ottenere 10kHz, avremmo una definizone di 8bit, con un range tra 0 e 99.
Ma se portiamo la Fosc 1 16MHz, la definizione risale a 10bit e il range si allarga.

In sostanza, il forzare frequenze di PWM elevate con bassa Fosc riduce la risoluzione e il range di variazione da applicare a ***DcxB9:DCxB0***; aumentando la Fosc si corregge il problema.

Un ulteriore considerazione va fatta nel caso in cui si utilizzi l'interrupt del Timer2, che determina la cadenza delle interruzioni al secondo. E' evidente che una cadenza troppo elevata rispetto alle necessità di esecuzione delle istruzioni dell'aplicazione, la rende di difficile gestione.

Riepilogo dei registri usati per il modulo PWM.

Questi sono i registri interessati alle operazioni su EEPROM e Flash (dal foglio dati di 16F690):

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
0Bh/8Bh/ 10Bh/18Bh	INTCON	GIE	PEIE	T0IE	INTE	RABIE	T0IF	INTF	RABIF	0000 000x	0000 000x
0Ch	PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	-000 0000
0Eh	TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
0Fh	TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
10h	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0000 0000	uuuu uuuu
11Bh	CM2CON1	MC1OUT	MC2OUT	—	—	—	—	T1GSS	C2SYNC	00-- --10	00-- --10
15h	CCPR1L	Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	uuuu uuuu
16h	CCPR1H	Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	uuuu uuuu
17h	CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	0000 0000
87h/187h	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	1111 1111
8Ch	PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	-000 0000

Da osservare che Timer1 è usato nel modo CCP.

La tabella indica anche la situazione dei bit nei registri a seguito di reset per **POR/BOR** (caduta della tensione sotto i minimi ammissibili) o per altri reset che non comportano direttamente la mancanza di tensione (**WDT, MCLR**).

I registri di gestione del modulo solitamente sono raccolti in un solo banco.

CCPR1L	15h
CCPR1H	16h
CCP1CON	17h
RCSTA	18h
TXREG	19h
RCREG	1Ah
	1Bh
PWM1CON	1Ch
ECCPAS	1Dh

Attenzione perchè i registri relativi a **Timer2**, gli SFR **PIE1/PIR1/TRISC** possono essere in altri banchi.

INTCON è accessibile da qualsiasi banco.