Esercitazioni PIC Midrange

Inserto_g: CCP/ECCP

Il modulo **CCP/PWM**, oltre alla funzione PWM vera e propria, permette altre due azioni:

- **Capture**
- **Compare**

Le due funzioni permettono di automatizzare diverse operazioni, basate sulla misura o generazione di tempi o periodi e di impulsi.

Entrambi i modi sono selezionabili dal registro **`CCP1CON`**: 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: uscita =1 alla comparazione**
1001	**Modo Compare: uscita =0 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

Nella tabella abbiamo la situazione per il modulo ECCP/PWM+.

Quella del modulo CCP/PWM base è identica, con la sola differenza che non dispone della modalità ***0010 – Compare con toggle out.***

I bit **`CCP1CON7:4`** del registro non sono usati in questi modi di lavoro.

Entrambe le modalità usano il **Timer1** come base tempi. Quindi abbiamo la seguente situazione:

Modo	Timer
Capture	**Timer1**
Compare	**Timer1**
PWM	**Timer2**

I registri impiegati sono gli stessi dei modi PWM, con queste funzioni:

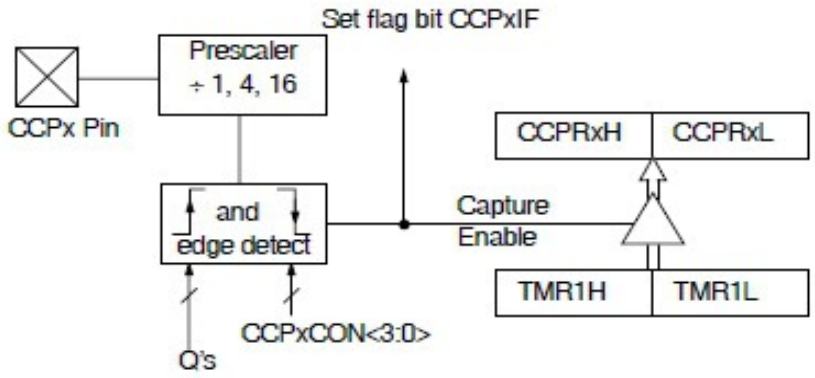
Registro	Capture	Compare	PWM
CCPxCON	Selezione del modo	Selezione del modo	Selezione del modo + LSB del duty cycle
CCPRxL	LSB tempo catturato	LSB tempo comparato	MSB del duty cycle
CCPRxH	MSB tempo catturato	MSB tempo comparato	-
TRISx	CCPx come ingresso	CCPx come uscita	CCPx come uscite
T1CON	Timer e prescaler	Timer e prescaler	-
T2CON	-	-	Timer e prescaler
PR2	-	-	Periodo di Timer2
PIE1	Abilitazione interrupt	Abilitazione interrupt	Abilitazione interrupt
PIR1	Flag interrupt	Flag interrupt	Flag interrupt
INTCON	GIE e PEIE	GIE e PEIE	GIE e PEIE
PWM1CON	-	-	Dead band e auto shutdown/restart
ECCPAS	-	-	Auto shutdown

I registri **PWM1CON** ed **ECCPAS** sono disponibili solo per PWM+.

Vediamo i particolari e come utilizzare queste funzioni.

Capture.

La funzione ***Capture*** ha lo scopo di "catturare" la durata di un evento esterno.
La struttura di base è questa:



Un segnale esterno ad onda quadra viene valutato a seconda di 4 possibili eventi:

- **ogni fronte di discesa**
- **ogni fronte di salita**
- **ogni 4 fronti di salita**
- **ogni 16 fronti di salita**

Queste opzioni sono selezionabili con i bit **`CCP1CON3:0`**.

CCP1M3:0	Funzione
0000	Modulo CCP disabilitato
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**

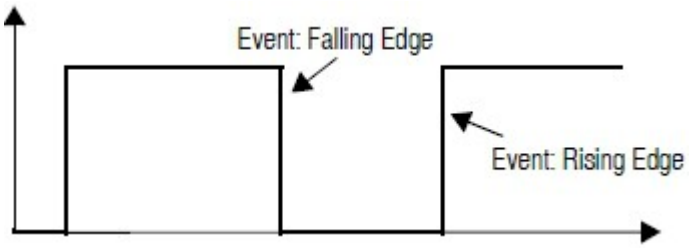
All'evento, i Timer1 che era stato preventivamente avviato, ha raggiunto un certo valore a 16 bit sui registri **`TMR1H:L`**.


Va osservato che il Timer1 deve operare in ***timer mode*** o ***synchronized counter***.

L'evento "fotografa" il contenuto di **`TMR1H:L`** al momento dell'evento e lo copia automaticamente nei registri **`CCPR1H:L`**. Questo consente di salvare il valore per i calcoli successivi, mentre il **Timer1** continua a contare senza arrestarsi.

L'evento genera un interrupt e il programma legge e salva il valore, cancellando il flag **`CCP1IF`** per avere un nuovo interrupt all'evento successivo.
Quando questo accade, si avrà un nuovo valore salvato in **`CCPR1H:L`**; facendo la differenza tra il valore attuale e quello precedente si avrà il tempo della durata tra un evento e il successivo, ovvero

il suo periodo.
L'evento è basato sulla transizione di livello del segnale esterno.



- ***Falling edge*** è la transizione da **livello alto a livello basso**.
- ***Rising edge*** è la transizione da **livello basso a livello alto**.

Il pin **CCP1**, che, come abbiamo detto, è in ingresso, serve a ricevere il segnale esterno da catturare e lo passa ad un prescaler con 4 fattori di divisione.

Attenzione: il pin CCP1 è sempre necessario e deve essere impostato come ingresso. Se è configurato come uscita, una scrittura sul port relativo produce un evento di capture.

Questo prescaler, quando la frequenza di ingresso è costante durante la misura, può essere usato per ottenere una risoluzione molto fine. Ad esempio, se abbiamo un prescaler **1:16**, vengono considerati 16 periodi della frequenza di ingresso e l'errore medio di questi è pari a ***1Tcy***. Questo rende una precisione reale di ***Tcy/16***, che a 20MHz di clock equivale a 12.5ns.
Se usiamo il prescaler **1:1** la precisione sarà ***Tcy***.

Quando la modalità ***Capture*** è modificata, è possibile sia generato un interrupt indesiderato. Se, durante il funzionamento, si vuole cambiare il modo, occorre disabilitare l'interrupt e cancellare il flag **CCP1IF**.
Quando il modulo **CCP** viene disabilitato, il prescaler è cancellato (quindi anche dopo un reset, dato che il default è disabilitato).
Quindi, per cambiare modalità, si consigliano le seguenti istruzioni:

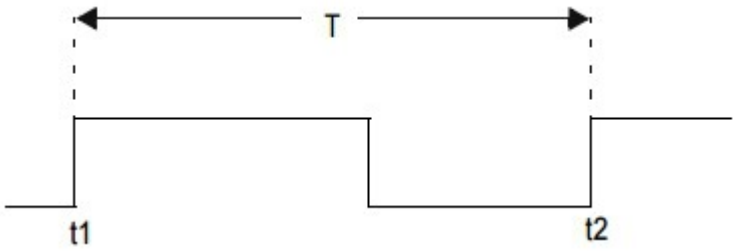
```
; cambia modalità del CCP
     banksel CCP1CON
      clrf    CCP1CON       ; CCP disabilitato
      movlw   new_mode      ; nuovo modo
      movwf   CCP1CON
```

In modalità sleep, il Timer1 non è operativo (mancando il clock), ma il prescaler continua a contare eventi, dato che non è sincronizzato con il clock interno. Quando l'evento di cattura programmato accade, il flag **CCP1IF** vine settato: se l'interrupt del modulo **CCP** è abilitato, si genera un wakeup da sleep.
In queste condizioni, il registro **CCPR1H:L** non viene aggiornato. Però, dato che **TMR1H:L** non sono stati incrementati, il contenuto non ha importanza.

Applicazioni di Capture.

Per chiarire le possibilità della funzione ***Capture***, vediamo alcune possibili applicazioni.

- ## Misura del periodo di un'onda quadra.



Predisponendo il modo Capture sul fronte di salita o discesa, quando si verifica l'evento il contenuto del contatore del Timer1 e' copiato nei registri **CCPR1L** e **CCPR1H** e viene generato un interrupt. Tenendo conto di due eventi e sottraendo il valore precedente dall'ultimo, è possibile determinare il numero di cicli istruzione (***Tcy***) intercorrono trai due eventi e quindi, in sostanza, la durata del periodo.
Se ***N*** e' il numero ottenuto dalla differenza ***t2-t1***, il periodo ***T*** dell'onda quadra è:

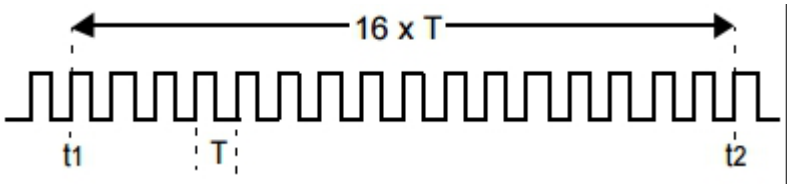
$$T = N * 4/Fosc$$

La sequenza di operazioni necessarie per implementare l'applicazione è la seguente:

1. Programmare il Timer1 come contatore asincrono e clock da ***Fosc/4***
2. Programmare il prescaler con un valore tale da evitare l'overflow del Timer1 prima della prima cattura
3. Programmare la modalità ***Capture*** sul fronte di salita o discesa, ovvero **CCP1M3:0 = 0100** o **0101**
4. Abilitare l'interrupt del modulo **CCP** , il che richiede di settare il bit **CCP1IE** nel registro **PIE1.** Per avere l'interruzione occorre abilitare sia **GIE** che **PEIE**.
5. Ricordarsi di verificare se **CCP1** è configurato in TRIS come ingresso (è il default)
6. Attendere il primo interrupt e salvare il contenuto di **CCPR1H:L**
7. Cancellare il flag **CCP1IF** ed attendere l'interrupt successivo
8. Salvare il contenuto di **CCPR1H:L**
9. Sottrarre i due valori; il risultato è l'ampiezza in ***Tcy*** del periodo dell'onda in ingresso.

- ## Misura mediata del periodo di un'onda quadra.

Come nel caso precedente, con questa variazione: la cattura è programmata per 4 o 16 fronti.



E' possibile discriminare solo fronti di salita.

Questa azione è applicabile solo il periodo (frequenza) della forma d'onda da misurare è costante durante i 4 o 16 periodi.

Il vantaggio è dato dalla maggiore reiezione ai disturbi data dalla media e si otterrà una definizione migliore.
Se ***N*** e' il numero ottenuto dalle operazioni di differenza ***t2-t1***, diviso per il numero dei periodi valutati, il periodo ***T*** dell'onda quadra è:

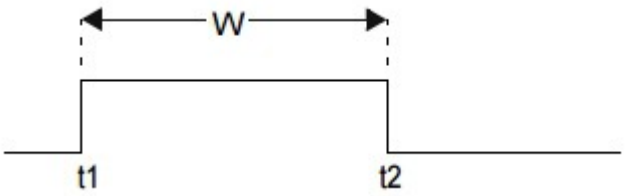
$$T = N/4 * 4/Fosc \quad \text{oppure} \quad T = N/16 * 4/Fosc$$

La sequenza di operazioni necessarie per implementare l'applicazione è la seguente:

1. Programmare il Timer1 come contatore asincrono e clock da ***Fosc/4***
2. Programmare il prescaler con un valore tale da evitare l'overflow del timer per la durata di 4 o 16 periodi dell'onda in ingresso.
3. Programmare la modalità Capture sul fronte di salita o discesa, ovvero `CCP1M3:0 = 0110` o `0111`
4. Abilitare l'interrupt del modulo **CCP** , il che richiede di settare il bit `CCP1IE` nel registro `PIE1`. Per avere l'interruzione occorre abilitare sia **GIE** che **PEIE**.
5. Ricordarsi di verificare se `CCP1` è configurato in TRIS come ingresso (è il default)
6. Attendere il primo interrupt e salvare il contenuto di `CCPR1H:L`
7. Cancellare il flag `CCP1IF` ed attendere l'interrupt successivo
8. Salvare il contenuto di `CCPR1H:L`
9. Sottrarre i due valori e dividere il risultato per 16 (4 shift a sinistra) o per 4 (due shift a sinistra)

• Misura della larghezza di un impulso singolo.

Le prime due modalità **Capture** danno modo di attivare l'evento sia sul fronte di salita che su quello di discesa.



Programmando prima uno e poi l'altro (nel caso in figura, ***t1*** è determinato da un rising edge e ***t2*** da un falling edge) abbiamo due eventi che coincidono con la durata dell'impulso
La sequenza di operazioni necessarie per implementare l'applicazione è la seguente:

1. Programmare il Timer1 come contatore asincrono e clock da ***Fosc/4***
2. Programmare il prescaler con un valore tale da evitare l'overflow del Timer1 in modo tale che non si abbia overflow per la durata dell'impulso
3. Programmare la modalità Capture sul fronte di salita o discesa, ovvero `CCP1M3:0 = 0100`

4. Abilitare l'interrupt del modulo **CCP** , il che richiede di settare il bit **CCP1IE** nel registro **PIE1.** Per avere l'interruzione occorre abilitare sia **GIE** che **PEIE**.
5. Ricordarsi di verificare se **CCP1** è configurato in TRIS come ingresso (è il default)
6. Attendere il primo interrupt e salvare il contenuto di **CCPR1H:L**
7. Cancellare il flag **CCP1IF**
8. Programmare la modalità Capture sul fronte di discesa, ovvero **CCP1M3:0 = 0101** con le avvertenze viste prima
9. Attendere l'interrupt successivo
10. Salvare il contenuto di **CCPR1H:L**
11. Sottrarre i due valori, ottenendo un valore ***N***

La durata dell'impulso è:

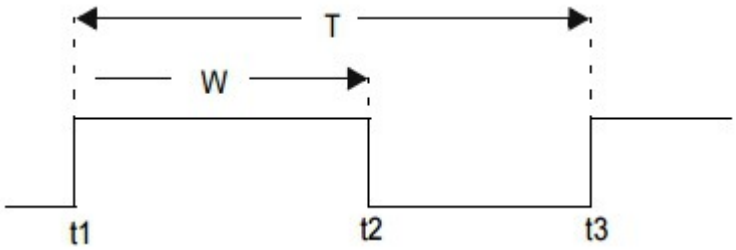
W = 4/Fosc * N

Si potrà misurare anche il duty cycle di un segnale PWM variando ancora una volta il fronte di commutazione ed utilizzando i tempi ottenuti per avere l'ampiezza dei tempo on e di quello off.

Abbiamo detto che il segnale in ingresso deve essere a livello logico e con forma d'onda quadra. In presenza di altri segnali, con forme d'onda diverse o anche di segnali analogici, si potrà utilizzare un comparatore, collegando l'uscita all'ingresso del modulo CCP.

• Misura del duty cycle.

Il duty cycle è il rapporto tra il tempo on e off di un periodo, ovvero tra il tempo on e la durata del periodo stesso.



Occorre la valutazione di tre tempi:

- ***t1*** e ***t2*** per il l'ampiezza dell'impulso ***W*** e ***t1*** e ***t3*** per la durata del periodo ***T***.

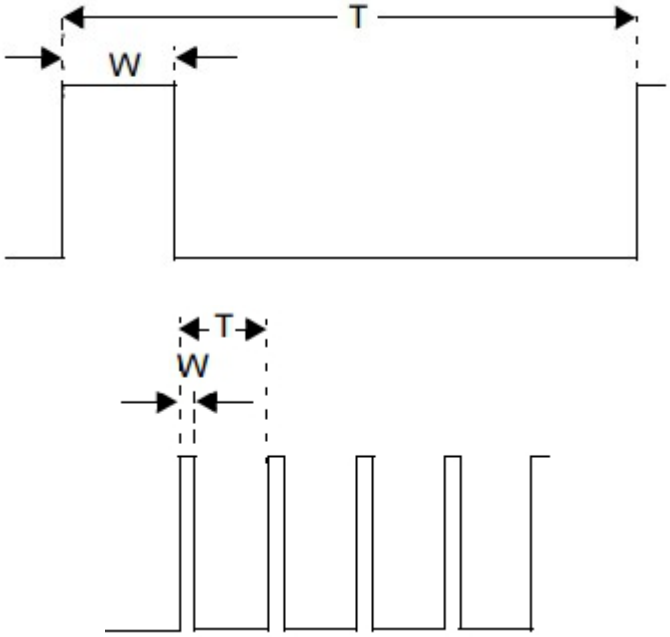
La sequenza di operazioni necessarie per implementare l'applicazione è la seguente:

1. Programmare il Timer1 come contatore asincrono e clock da ***Fosc/4***
2. Programmare il prescaler con un valore tale da evitare l'overflow del Timer1 per la massima durata del periodo **T**
3. Programmare la modalità Capture sul fronte di salita o discesa, ovvero **CCP1M3:0 = 0100**
4. Abilitare l'interrupt del modulo **CCP** , il che richiede di settare il bit **CCP1IE** nel registro **PIE1.** Per avere l'interruzione occorre abilitare sia **GIE** che **PEIE**.
5. Ricordarsi di verificare se **CCP1** è configurato in **TRIS** come ingresso (è il default)
6. Attendere il primo interrupt e salvare il contenuto di **CCPR1H:L** che è ***t1***
7. Cancellare il flag **CCP1IF**

8. Programmare la modalità Capture sul fronte di discesa, ovvero `CCP1M3:0 = 0101` con le avvertenze viste prima
9. Attendre l'interrupt successivo
10. Salvare il contenuto di `CCPR1H:L` che è ***t2***
11. Sottrarre i due valori e operare come nel caso precedente per ottenere ***W***
12. Riconfigurare per il rising edge e catturare ***t3***
13. Eseguire le operazioni su ***t1*** e ***t3*** per ottenere il periodo ***T***
14. Calcolare il duty cycle come ***T/W***

• Misura della rotazione in RPM.

E' comune utilizzare encoder per misurare la velocità di rotazione. Un encoder rende una uscita ad onda quadra proporzionale al numero di giri.



Se la velocità è bassa, l'impulso in uscita avrà una durata W. Se la velocità aumenta, aumenta anche la frequenza dell'onda quadra e W si riduce.

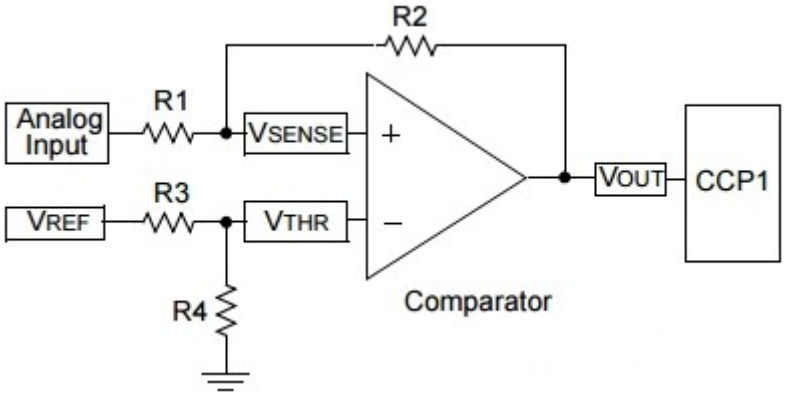
In pratica, misuriamo come visto in precedenza la durata del periodo e ne deriviamo la velocità di rotazione.

Da notare che stiamo considerando encoder di tipo ottico e non meccanico, ovvero esenti da rimbalzi; questo è essenziale per la qualità della misura, dato che essa si basa su una forma d'onda quadra esente da disturbi.
Anche gli encoder basati su sistemi magnetici (effetto Hall) sono adeguati.

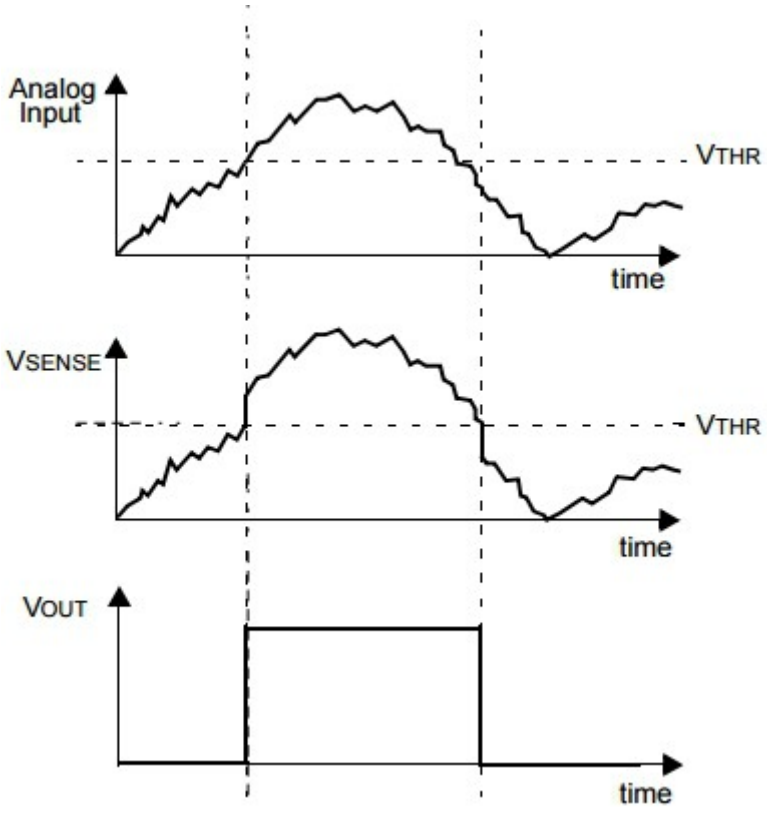
- **Misura di ingressi con forma d'onda irregolare.**

Se abbiamo a che fare con segnali che non sono onde quadre ideali o non sono a livello logico, è necessario anteporre all'ingresso CCP1 un adeguato front end.

Se si tratta di segnali analogici, un sistema semplice è quello di utilizzare un comparatore integrato per squadrare il segnale.



L'ingresso analogico è confrontato con una tensione di riferimento e l'uscita del comparatore è inviata direttamente all'ingresso CCP1.
La tensione di riferimento può essere esterna o interna, dove il chip disponga del modulo Vref.
R1 e R2 provvedo ad introdurre una isteresi per evitare jitter sulle commutazioni del segnale in uscita.

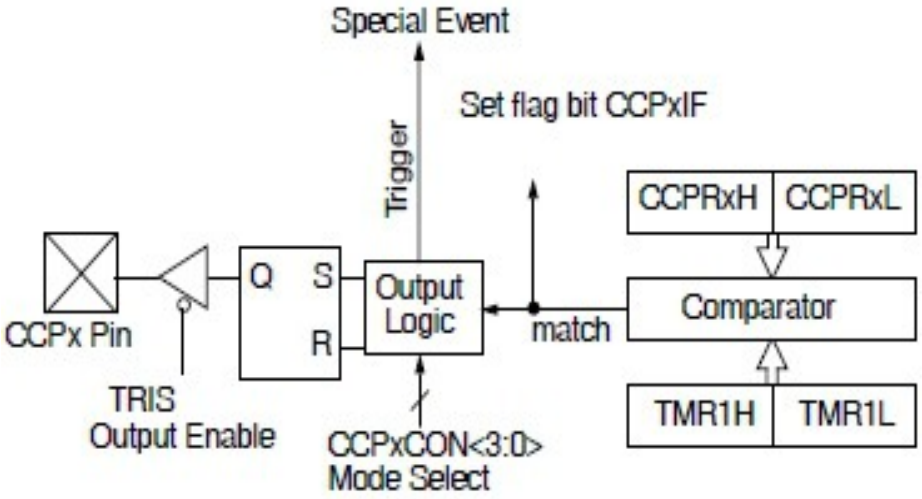


Avendo disponibile l'onda quadra, la potremo valutare con i metodi visti sopra.

Compare.

La funzione Compare ha lo scopo di ha lo scopo di confrontare un valore scritto nel registro **`CCPR1H:L`** con il contenuti del registro **`TMR1H:L`**.

La struttura di base è questa:



Nel momento in cui i due registri si equivalgono, il pin **CCP1** , che è una uscita, può essere :

1. **forzato alto**
2. **forzato basso**
3. **non modificato**
4. **Special Event Trigger**
5. **inversione dello stato (toggle)** – disponibile **solo con ECCP/PWM+**

Attenzione:

- **nei modi 1, 2 e 5 il pin CCP1 viene utilizzato e deve essere impostato come uscita.**
- **nei modi 3 e 4 il pin CCP1 non viene usato e può essere dedicato ad altre funzioni.**

L'azione è controllata dai bit **`CCP1CON3:0`**che selezionano le 5 opzioni.

CCP1M3:0	Funzione
0000	Modulo CCP disabilitato
0010	**Modo Compare con toggle out**
1000	**Modo Compare: uscita =1 alla comparazione**
1001	**Modo Compare: uscita =0 alla comparazione**
1010	**Modo Compare: interrupt senza uscita**
1011	**Modo Compare: trigger per evento speciale**

Come per il PWM, il **CCP1** viene controllato da un latch che non è quello specifico del pin nel suo port.

La base dei tempi è data dal Timer1.


Va osservato che il Timer1 deve operare in ***timer mode*** o ***synchronized counter***.

A quanto sopra si aggiunge uno ***Special Event Trigger*** che serve ad avviare una operazione ausiliaria; in questo caso si tratta di una conversione AD.

Si genera un interrupt se è stato abilitato **`CCP1IE`**, ma questo non azzera il contatore di Timer1, che prosegue il conteggio dal valore corrente.
Se, però, abilitiamo lo **Special Event**, al momento della comparazione il Timer1 viene azzerato.

Compare non opera in sleep in quanto Timer1 è bloccato. Il pin **`CCP1`** resta impostato al valore precedente all'istruzione sleep.

Il modo ***Compare*** consente di realizzare un cronometro di precisione. Normalmente si utilizza un count down caricando nei registri del timer il valore del tempo e decrementandolo fino a 0. Il prossimo ciclo richiede la ricarica dei registri di conteggio del timer con il valore voluto.
In ***Compare*** partiamo da 0 fino ad un tempo pre determinato.
Questo rende possibili azioni a cadenza determinata e interrupt periodici con il minimo di software.
Nei modi in cui il ***Compare*** agisce sul **`CCP1`**, può anche essere usato per generare forme d'onda come modulazioni PWM, PPM, ecc.

Applicazioni di Compare.

Per chiarire le possibilità della funzione ***Compare***, vediamo alcune possibili applicazioni.

- ## Interrupt periodici

La disponibilità di un interrupt a cadenza determinata è utile in molte situazioni. Questa tecnica consente di operare un loop continuo con interruzioni periodiche per eseguire altre task.

Normalmente questo si può ottenere con l'overflow di uno dei timer, ma questo consente solo un range di tempo limitato dalle possibilità del timer, oltre alla necessità di istruzioni per ricaricare I contatori.

Il modo ***Compare*** consente tempi più lunghi e non necessita di altre istruzioni oltre a quelle di avvio.

Ad esempio, se vogliamo ottenere una cadenza di 0.2s con Fosc=8MHz.
Possiamo utilizzare un prescaler 1:8 e un preload di 3036 per ottenere un periodo di interruzione di

0.25s.
Ora calcoliamo il valore da caricare in **CCPR1H:L**.
Avremo:

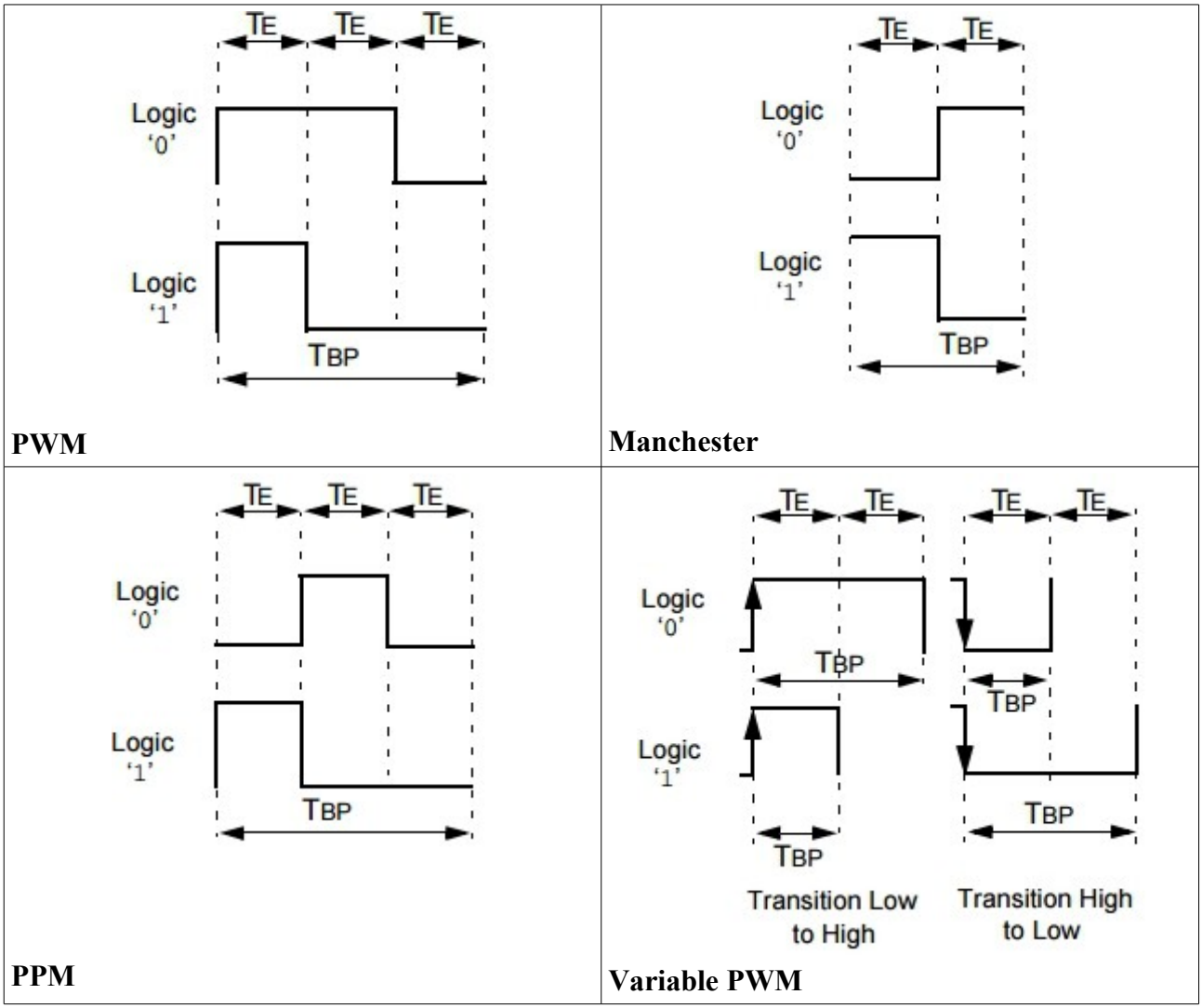
CCPR1H:L= intervallo di tempo/ (Tosc*4*presacler) = 0.2 / (125ns * 4 *8) = 5000 -> 0xC350

cioè **CCPR1H = C3** e **CCPR1L=50**
Possiamo ora impostare il modulo CCP per avere uno special event trigger. Se non abilitiamo il modulo ADC questa situazione ha il solo effetto di generare l'interruzione e azzerare il Timer1. Se il modulo ADC è usato, possiamo impostare la modalità **CCP1CON=00001010** dove viene generato solo l'interruzione; il timer sarà azzerato da programma.

Una simile soluzione può essere usata in sistemi operativi real time (RTOS) per temporizzare lo scheduling delle varie task.

• Modulazione di impulsi.

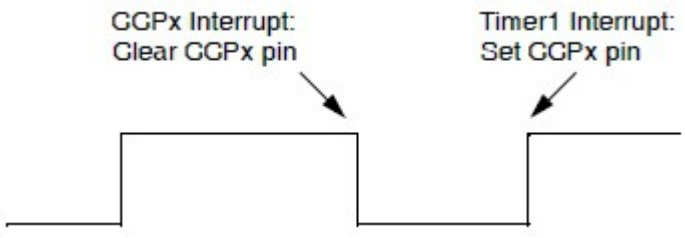
La funzione ***Compare*** consente di generare le più diverse forme di modulazione del segnale in uscita da **CCP1**, secondo tempi prefissati. Ad esempio:



Te è l'unità di tempo base di ogni formato di modulazione, mentre **Tbp** è il periodo..
Lo ***Special Event Trigger*** può essere usato per produrre l'elemento di temporizzazione di base.
Al momento dell'interruzione, il software provvederà a controllare il formato della modulazione.

• PWM a 16bit

Il modulo **PWM** consente di ottenere segnali con la massima definizione di 10bit.
Usando la modalità ***Compare*** si può estendere questa a 16bit.



La sequenza di operazioni necessarie per implementare l'applicazione è la seguente:

1. Programmare la modalità Compare per avere l'uscita a livello basso , ovvero `CCP1M3:0 = 1001`
2. `CCP1` è configurato in `TRIS` come uscita
3. Abilitare l'interrupt del **Timer1**
4. Programmare il periodo della forma d'onda in uscita, considerando anche il prescaler
5. Programmare il duty cycle con `CCPR1H:L`
6. portare a 1 il pin `CCP1` all'interrupt di Timer1

Non è possibile raggiungere il duty cycle del 100% in quanto si ha una certa spesa di tempo nella latenza dell'interrupt di Timer1.
Questo, però, non si riflette sulla durata del periodo, dato che è l'interrupt stesso a determinarlo.

Timer1 ha quattro prescaler configurabili, da 1:1 a 1:8. In base a questo le frequenze possibili sono determinate da:

Fpwm = Fosc / (65536 * 4 * prescaler)

Per una ***Fosc=20MHz*** le frequenze minime saranno 76.3, 38.1, 19.1 e 9.5Hz a seconda del prescaler.

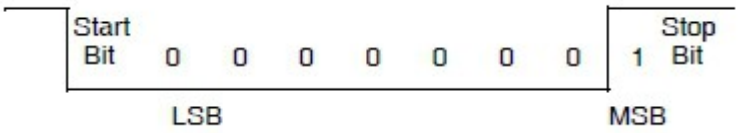
Compare & Capture.

E' possibile usare ***Capture*** e ***Compare*** in combinazione tra di loro per ottenere varie azioni complesse.
Ecco un paio di esempi.

- ## Auto baudrate

Una trasmissione seriale RS-232 si basa su una frequenza di baud rate che può assuere diversi valori. La comunicazione tra due dispositivi sarà possbile solo se i baud rate sono identici.
E' possibile determinare da programma quale è il baud rate ricevuto ed adattarsi, quindi, automaticamente. I più recenti moduli **EUSART** hanno questa funzione, ma d**ove nel chip non è disponibile un UART**, si può utilizzare il modulo **CCP**.

Questo è configurato in ***Capture*** per determinare il baud rate e quindi in modo ***Compare*** per generare la frequenza voluta.
E' necessario ricevere un carattere di calibrazione con una forma come questa:



Si potrà procedere in questo modo:

1. Programmare il CCP per catturare il falling edge, che corrisponde all'inizio del bit di start
2. Salvare il valore ***t1*** al momento della comparazione
3. Programmare il CCP per catturare il rising edge,
4. Salvare il valore ***t2*** al momento della comparazione
5. Eseguire **t2-t1** per ottenere la durata degli 8 bit
6. Dividere il risultato per 8. Questo è il tempo di un bit (***Tb***)
7. Shiftare ***Tb*** a destra di una posizione. Questo è metà periodo d un bit

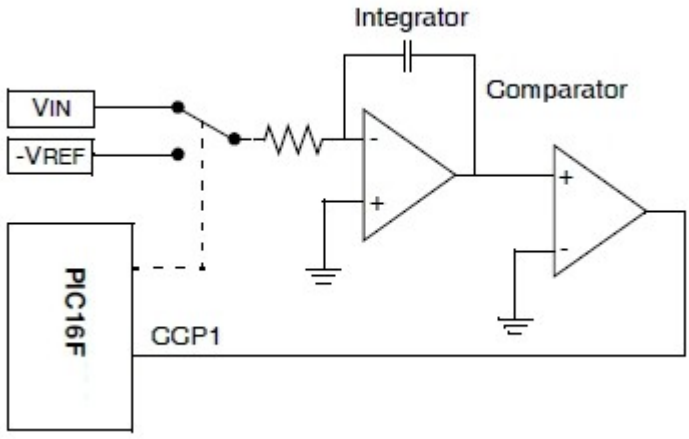
Si potrà poi usare il modo ***Compare*** per generare il baud rate voluto.
Una applicazione Microchip (***AN712 – RS-232 Autobaud***) riporta anche il codice necessario.
Un esempio del codice di trasmissione e ricezione senza uso di UART.

```
TxRoutine:
        movlw  8          ;preload bit counter
        movwf  counter
        bcf    TxLine     ;line initially high,
;toggle low for START bit
TxLoop:
        call   DelayTb    ;wait Tb (bit period)
        rrf    RxByte,f   ;rotate LSB first into Carry
        btfss  STATUS,C   ;Tx line state equals state of Carry
        bcf    TxLine
        btsfc  STATUS,C
         bsf   TxLine
        decfsz Counter,f  ;Repeat 8 times
         goto  TxLoop
        call   Delay Tb   ;Delay Tb before sending STOP bit
        bsf    TxLine     ;send STOP bit
```

```
RxRoutine:
        btfsc   rxline          ;wait for receive line to go low
         goto   Rxroutine
        movlw   8               ;initialize bit counter
        movwf   counter
        call    delay1halftb    ;delay 1/2 tb here plus tb in rxloop
;in order to sample at the right time
rxloop:
       call     delaytb         ;wait tb (bit period)
       btfss    rxline          ;carry flag state equals rx line state
       bcf      status,c
       btfsc    rxline
        bsf     status,c
       btfsc    rxline
        bsf     status,c
       rrf      rxbyte,f       ;rotate lsb first into receive type
       decfsz   counter,f      ;repeat 8 times
        goto    rxloop
```

- ## Conversione AD a doppia rampa.

E' possibile realizzare un convertitore AD a doppia rampa usando il modulo CCP e i comparatori.

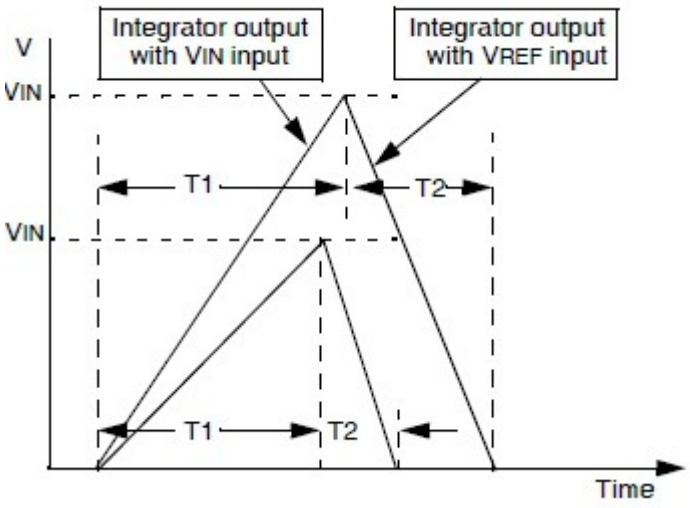


Vengono usati I due comparatori integrati in molti PIC16F.

La converisone a doppia rampa integra il segnale analogico ***Vin*** per un tempo prefissati ***T1***.
Di seguito, l'ingresso viene commutato su una tensione di riferimento (***-Vref***) ed integrata per il tempo ***T2*** fino a che l'uscita dell'integratore è 0.

Vin risulta funzione di ***Vref*** e del rapporto tra ***T1*** e ***T2***.

I componenti principali della conversione sono i tempi ***T1*** e ***T2*** **.**
Il modulo CCP consente di averli in modo Compare & Capture.



Si potrà procedere in questo modo:

1. Programmare il CCP in modo ***Compare*** con Special Event Trigger
2. Collegare l'ingresso a analogico a Vin
3. Attendere il tempo ***T1***, determinato in base alla capacità del condensatore dell'integratore
4. All'interrupt commutare l'ingresso su -Vref e configurare il modulo CCP su ***Capture*** con falling edge
5. All'interrupt il tempo catturato è ***T2***
6. Calcolare Vin come ***Vin = Vref * (T2/T1)***

Riepilogo dei registri usati per il modulo CCP.

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 Timer2 è usato nel modo PWM.

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 CCP 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 **Timer1**, gli SFR **PIE1/PIR1/TRISC** possono essere in altri banchi.

INTCON è accessibile da qualsiasi banco.

Altre informazioni.

CCP/ECCP è una periferica con una certa complessità, che può essere di non immediata comprensione, ma possiede vari automatismi che permettono di risparmiare azioni software ed ottenere temporizzazioni di precisione. ed ha molteplici applicazioni, come:

- registrazione di eventi
- comparazione di tempi
- generazione di interrupt periodici di precisione
- misura del periodo e dell'ampiezza di un impulso
- misura della frequenza e del duty cycle
- generazione di forme d'onda con determinate frequenze e duty cycle
- riferimenti di tempo
- conteggio di eventi
- e molto altro

Altre informazioni nella Documentazioni Microchip:

- ***DD 30673a – Capture/Compare/PWM modules (CCP and ECCP)***
- ***DS 4121 – PICMICRO CCP and ECCP Tips ' Tricks***
- ***microchipdeveloper.com/8bit:ccp***
- ***microchipdeveloper.com/8bit:ccpcapture***
- ***microchipdeveloper.com/8bit:ccpcomparetimer1***
- ***microchipdeveloper.com/8bit:ccpcapturepre***

oltre ai fogli dati dei singoli componenti.