SETTOREZERO.COM
PLAY EMBEDDED ELECTRONICS

Generare segnali PWM con i moduli Output Compare sui PICmicro a 16 e 32bit

Indice

Informativa

Avvertenze

I progetti presentati non hanno la certificazione CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea. Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.

L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto delle informazioni, del materiale, dei dispositivi o del software presentati nella seguente opera. Si fa inoltre presente che quanto riportato viene fornito cosi com'e, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza e senza supporto alcuno.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Introduzione

Prima di continuare con la lettura di questa brief-note, vi invito a leggere il tutorial di Mauro Laurenti sul PWM, linkato nell'ultimo paragrafo, qualora non l'aveste già fatto. Prendo difatti atto che chi legge questo documento sappia già come è strutturato un segnale PWM e abbia chiari i concetti di frequenza, periodo, duty cycle e a cosa serve un segnale modulato in ampiezza.

Molti PICmicro a 16 e 32bit dispongono di un modulo che, tra le altre cose, ha la funzione di generare segnali PWM. Tale periferica prende il nome di **Output Compare,** abbreviato come **OC** (sui PICmicro ad 8 bit abbiamo invece quello che viene chiamato modulo *CCP - Capture, Compare, PWM*. Il modulo OC è simile a questi a meno della funzione di *Capture* che viene invece eseguita da moduli dedicati chiamati *Input Compare*).

I moduli OC hanno la funzione di confrontare il valore assunto da un counter/timer con uno o due registri (a seconda della modalità di funzionamento impostata). Quando si verifica il "match" di questi due valori, *accade qualcosa*: un interrupt, un cambio di stato dei pin.

> *Su altri PICmicro a 16 e 32 bit, la generazione hardware di segnali modulati in ampiezza avviene invece mediante altre periferiche più sofisticate come il Motor Control PWM (MCPWM) o l'High Speed PWM. In questa trattazione non mi occuperò di questi due moduli.*

I moduli OC sui diversi PICmicro a 16 e 32bit hanno varie modalità di funzionamento sebbene buona parte dei moduli abbia molte cose in comune. In particolare illustrerò i moduli chiamati "*Output Compare*" e i moduli "*Output Compare with dedicated Timers*": si tratta di due tipologie di moduli leggermente differenti. I primi per il loro funzionamento si appoggiano ad un timer esterno, i secondi hanno un proprio timer dedicato e possono quindi fare a meno dei timer esterni.

In questa brief-note cerco di unificare i concetti relativi ai due diversi tipi di moduli fino a quando è possibile farlo, in maniera tale da rendere immediato il passaggio dall'uno all'altro tipo di periferica. E' comunque compito del lettore leggere il datasheet del PICmicro che sta utilizzando ma soprattutto fare riferimento al Family Reference Manual relativo. In questo documento difatti non approfondirò tutti i concetti e le considerazioni relative ai moduli Output Compare.

Per questa trattazione, dovendo illustrare anche esempi pratici, prendo come riferimento i **PIC24FJ64GB002** e il **PIC32MX250F128B** utilizzati su ORbit16™. E' chiaro che le informazioni e gli esempi valgono per tutti i PICmicro appartenenti alle stesse famiglie e/o con la stessa tipologia di modulo.

Per evitare di scrivere la sigla delle MCU al completo, da questo momento in poi mi riferirò a questi 2 PICmicro presi ad esempio semplicemente come "PIC24" e "PIC32", per cui tenete conto che se in questa brief-note leggete "PIC24" non mi riferisco a *tutti*

i PIC24 ma solo al PIC24FJ64GB002 : altri PIC24 potrebbero difatti avere impostazioni differenti.

Su questi 2 PICmicro abbiamo **5** moduli OC, indicati genericamente come OCx (quando il lettore vedrà la x pensi che va sostituita con un numero da 1 a 5 per identificare il numero del modulo: OC1, OC2 ecc).

Bisogna innanzitutto sapere che, come tante altre periferiche, su questi due PICmicro i moduli OCx non hanno un'assegnazione fissa e possono (devono) essere assegnati su un pin RPn a piacere mediante la funzione di Peripheral Pin Select.

> *Su altri PICmicro, invece (quelli che non hanno la funzione PPS), i moduli OC hanno uscite fisse, per cui fate sempre riferimento al datasheet della MCU che state utilizzando.*

I PIC32, in particolare, hanno un'implementazione della funzione PPS diversa da quella dei PIC24. Sui PIC32, difatti, le periferiche (funzioni) sono suddivise in gruppi e non possono essere assegnate a qualsiasi pin RPn ma solo a quelli appartenenti al proprio gruppo. Per questo fate riferimento al documento *Microchip-PIC32MX-Peripheral-Library.chm* presente nella cartella Docs dell'installazione di *MPLAB C Compiler for PIC32 MCUs*.

I moduli OC sono singoli: ogni modulo genera un solo segnale PWM, a differenza del singolo modulo CCP dei PICmicro a 8 bit che fornisce generalmente 2 segnali.

Registri coinvolti nella generazione di segnali PWM

Cominciamo con il prendere familiarità con i registri coinvolti nella generazione del segnale PWM:

TMR$_Y$ - Ovvero il Timer Y (dove Y identifica il numero del timer). Il Timer può svolgere diverse funzioni a seconda del tipo di modulo OC e della funzionalità selezionata. I moduli OC con timer dedicato hanno un proprio timer per cui possono utilizzare i timer esterni come sorgente di clock (determinano la velocità di incremento del conteggio) e/o per la sincronizzazione con il timer dedicato (ovvero determinano il periodo). I moduli OC che io definisco "semplici", invece, non hanno un proprio timer e devono appoggiarsi ai timer esterni per tutte le funzioni. Il PIC24 ha i moduli OC con timers dedicati e tali moduli possono sfruttare come sorgente di clock e/o sincronizzazione i Timers dall'1 al 5. Il PIC32 ha un modulo OC semplice e può usare come sorgente di clock soltanto i Timers 2 e 3 nonostante anche qui si abbiano a disposizione 5 timers.

PR$_Y$ - Il registro di confronto (Period Register) del Timer Y. Viene generalmente utilizzato per l'impostazione del periodo (e quindi della frequenza) del segnale PWM tranne che per alcune modalità che non necessasitano di questo registro (quelle che usano il timer solo come sorgente di clock e non come sorgente di sincronizzazione).

OCxTMR –E' il timer dedicato di cui parlavo prima, presente solo sui moduli "OC with dedicated timers". E' quindi presente nel PIC24 ma non nel PIC32.

OCxCON – Unico registro di controllo presente esclusivamente sui PIC con modulo OC semplice (PIC32).

OCxCON1 – Primo registro di controllo del modulo OC.

OCxCON2 - Secondo registro di controllo del modulo OC.

I registri OCxCON1 e OCxCON2 sono presenti soltanto sui moduli OC con timers dedicati (PIC24). Notate che l'unico registro di controllo del PIC32 non ha l'1 finale, dato che è l'unico registro di controllo presente. Sul PIC24 abbiamo invece 2 registri di controllo per ogni modulo (il modulo OC con timer dedicati è molto più laborioso da impostare!) e sono quindi numerati 1 e 2.

OCxR e **OCxRS** – Questi due registri (Output Compare Register e Output Compare Secondary Register) hanno scopi diversi a seconda della funzionalità PWM scelta. Contengono in ogni caso dei valori che hanno la funzione di determinare il duty cycle (tranne in una modalità, in cui invece OCxRS determina il periodo, vedremo dopo).

Prima di continuare chiarisco anche un altro concetto forse banale ma che spesso crea confusione: nei Datasheet della Microchip, così come nei Family Reference Manual, ci si riferisce alla parola "Duty Cycle" come alla durata dell'impulso a livello alto e non come *percentuale* del livello alto sul totale come si è invece normalmente abituati a

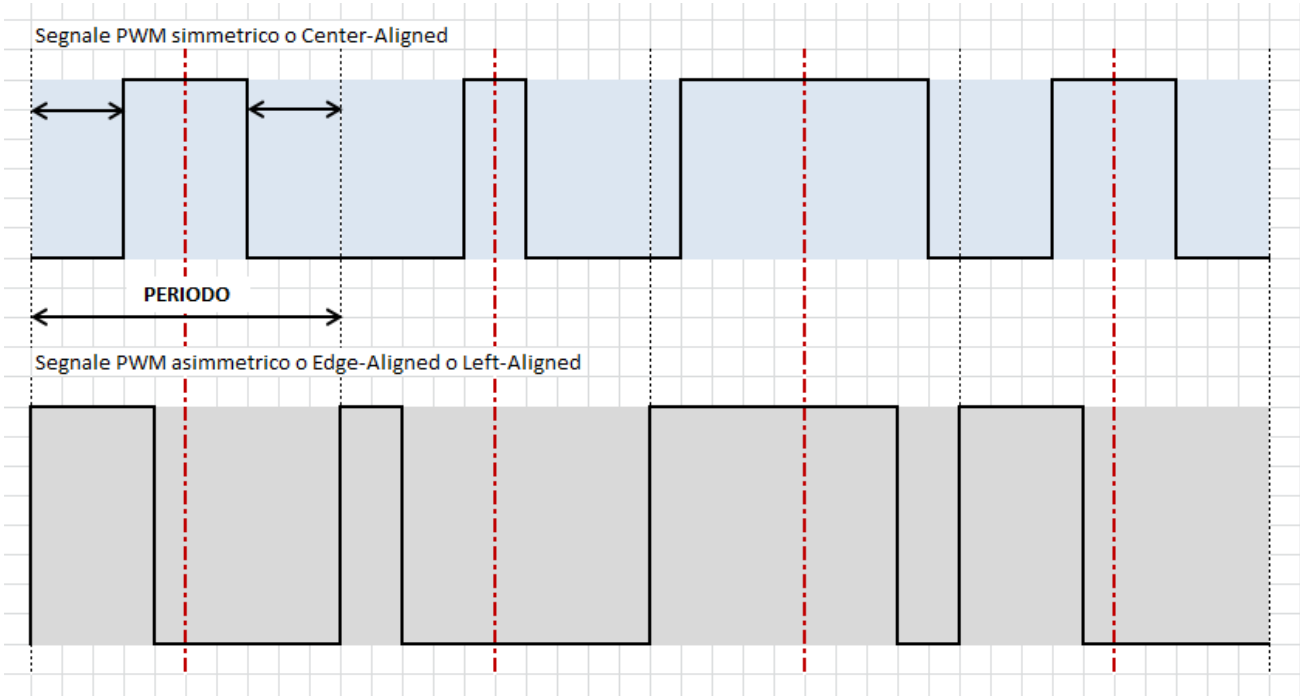
pensare. Per tale motivo da questo momento in poi quando leggerete “duty cycle”, interpretatelo come “durata del livello alto” se non altrimenti specificato (es.: valore riportato come percentuale).

Modalità PWM simmetrica e asimmetrica

I moduli OC con timers dedicati, come avrete già cominciato ad intuire, sono molto più complessi (hanno due registri per le impostazioni, possono usare più timers ecc). Per tale motivo, prima di continuare è doveroso fare una distinzione tra le due diverse modalità che si hanno a disposizione per la generazione di segnali PWM con tali moduli:

- modalità **center-aligned**, conosciuta più comunemente come **modalità simmetrica**
- modalità **edge-aligned**, conosciuta più comunemente come **modalità asimmetrica** o ancora modalità **left-aligned**

Potete incominciare a capire con un colpo d’occhio la differenza tra le due modalità con questo disegno:



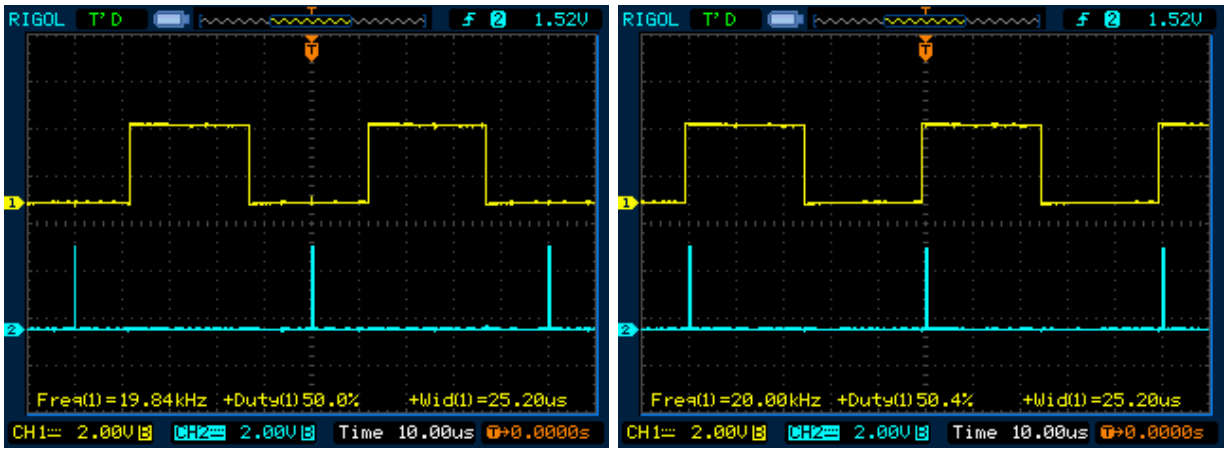
Notate che entrambi i segnali hanno la stessa variazione di duty cycle nel tempo (il fronte alto del segnale ha la stessa durata, il periodo è uguale), ma lo variano in maniera differente. In modalità simmetrica, il “centro” del livello alto è posizionato al centro del periodo (center-aligned : il segnale è allineato al centro del periodo, ed è quindi simmetrico), mentre in modalità asimmetrica il livello alto del segnale è allineato a sinistra del periodo (il fronte di salita -edge- è spostato a sinistra -left- del centro del periodo: segnale asimmetrico).

La differenza la si può notare nel momento in cui dobbiamo variare il duty cycle: in modalità simmetrica, il *centro* dell'onda quadra si troverà sempre nello stesso punto (con la stessa temporizzazione insomma) e all'atto pratico, quando andiamo a generare il segnale con il PICmicro, ci dobbiamo piuttosto preoccupare di variare i tempi di ingresso e di uscita del livello alto anziché la sola durata.

Ho preparato due video che aiutano a capire ancora meglio cosa accade:

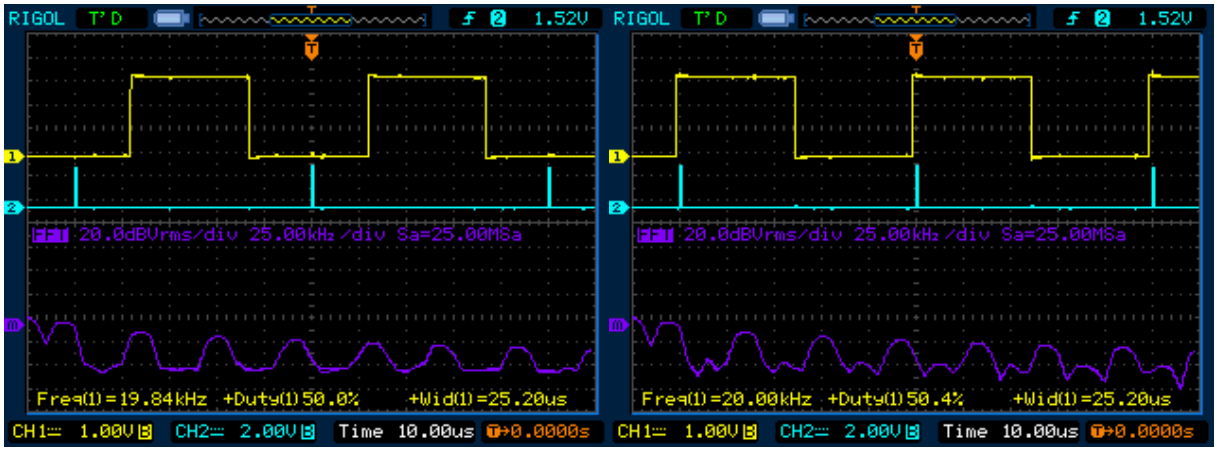
PWM Simmetrico: http://www.youtube.com/watch?v=JoeUPGMcUqE
PWM Asimmetrico: http://www.youtube.com/watch?v=pTKhkrXPMJA

Nei video sopra linkati, il picco della linea azzurra rappresenta l'inizio del periodo. Osservate come varia il segnale PWM (linea gialla) nelle due modalità.



Linea gialla: segnale PWM. Linea azzurra: periodo
Modalità simmetrica (a sinistra) contro modalità asimmetrica (a destra)

La modalità simmetrica diventa importante in particolari applicazioni dal momento che tale tipo di generazione del segnale crea meno armoniche ed è una modalità richiesta, ad esempio, per il pilotaggio di motori brushless o per buoni alimentatori switching.



Linea gialla: segnale PWM. Linea azzurra: periodo. Linea viola:FFT segnale PWM
Modalità simmetrica (a sinistra) contro modalità asimmetrica (a destra)

D'altro canto la modalità simmetrica, per poter essere realizzata, richiede ulteriori calcoli oltre a quelli necessari per impostare la frequenza del segnale: dobbiamo difatti calcolare ad ogni valore di duty cycle, i tempi in cui il segnale deve andare a livello alto e quindi tornare a livello basso, e questo non è automatico come per la modalità allineata a sinistra.

I moduli OC semplici (quelli senza timer dedicati), invece, non hanno la possibilità di selezionare la modalità simmetrica o asimmetrica e prevedono unicamente la modalità asimmetrica.

Calcolare Frequenza e Duty Cycle

Il periodo del PWM sui PICmicro a 16 e a 32 bit viene calcolato con la formula:

$$Periodo_{PWM} = (PR+1) * T_T * Prescaler$$

Dove PR è il valore del periodo da impostare nel registro che determina il periodo: PR_Y oppure OCxRS a seconda della modalità PWM selezionata, come vedremo dopo.

Chiarisco il significato di T_T dato che è una mia notazione e non la troverete nei datasheet (l'ho fatto per utilizzare un'unica formula valida sia per il PIC32 che per il PIC24):

Sui PICmicro a 32 bit questo che io ho chiamato T_T viene indicato come T_{PB} (dove PB sta per Peripheral Bus clock) e rappresenta il valore del tempo impiegato dalle periferiche per compiere un ciclo istruzioni. Su questi PIC, ricordo, CPU e periferiche hanno la possibilità di "girare" a frequenze differenti in base alla direttiva `#pragma` di configurazione `FPBDIV`.

Mettiamo il caso di avere impostato il PIC32 per lavorare a 40MHz –massima frequenza possibile sul PIC32 preso ad esempio (che, ricordo: non è la frequenza del quarzo/oscillatore ma la frequenza operativa *finale* della CPU, che tiene conto del divisore PLL in ingresso `FPLLIDIV`, divisore PLL in uscita `FPLLODIV` e moltiplicatore `FPLLMUL`) e abbiamo impostato nella configurazione `FPBDIV = DIV_1`: in queste condizioni anche le periferiche lavoreranno a quindi a 40MHz.

Dal momento che sui PICmicro a 32bit un ciclo di istruzioni viene eseguito in un solo colpo di clock, abbiamo che:

$$T_{PB} = \frac{1}{F_{OSC}} = \frac{1}{40\,MHz} = 25\,nS$$

Sui PICmicro a 16 bit il "mio" T_T è invece indicato come T_{CY} (dove CY sta per cycle) e rappresenta il tempo impiegato da un ciclo istruzioni. CPU e periferiche, qui, operano sempre alla stessa frequenza. Su tali PICmicro, un ciclo istruzioni viene eseguito in 2

colpi di clock per cui T_{CY}=2/FOSC. Supponiamo che il PICmicro sia impostato per lavorare a 32MHz – massima frequenza possibile sul PIC24 preso ad esempio, allora:

$$T_{CY}=\frac{2}{32\,MHz}=62.5nS$$

Stiamo parlando sempre di *ciclo istruzioni* sia per il PIC32 che per il PIC24, ma, dato che sul PIC32 il ciclo istruzioni può essere impostato su un valore diverso da quello della CPU, la Microchip utilizza giustamente una notazione differente. Quindi non fate confusione: parliamo sempre della stessa cosa.

Ora, dato che a noi interessa più che altro avere la possibilità di impostare un valore di frequenza del segnale PWM e quindi ricavare tutti gli altri parametri, sapendo che:

$$Frequenza=\frac{1}{Periodo}$$

e invertendo la formula precedente abbiamo:

$$PR=\left(\frac{F_{OSC}}{Frequenza_{PWM}*X*Prescaler}\right)-1$$

dove X vale 1 per i PICmicro a 32bit (formula che tiene conto del fatto che le periferiche viaggiano alla stessa velocità della CPU!) e X=2 per i PICmicro a 16bit. Molti per semplificare il calcolo eliminano il -1.

I valori di frequenza vanno tutti espressi con la stessa unità di misura: tutti in Hz, tutti in MHz ecc.

Una volta scelta la frequenza che deve avere il nostro segnale PWM, l'ultimo valore da impostare è il Prescaler (il divisore di frequenza) da assegnare al timer.

PR è un numero intero senza segno a 16bit, per cui il valore massimo che può assumere è 65535. Bisogna scegliere il prescaler in maniera tale da avere PR il più alto possibile ma senza ovviamente superare 65535. In questo modo il nostro segnale PWM avrà la risoluzione più alta e possiamo di conseguenza permetterci di variare il duty cycle in un range di valori più o meno ampio.

Vedete che il valore di Prescaler si trova al denominatore per cui più lo aumentiamo, più PR diminuisce, cerco quindi di impostarlo sempre a valori più bassi possibili, parto da 1, mi faccio i calcoli e controllo che PR non superi un numero a 16bit.

> *Sui PICmicro a 16bit e sui PICmicro a 32 bit relativamente ai Timer di tipo A (quali sono il Timer 2 e 3), il prescaler assegnato ai timer può essere impostato soltanto ai valori: 1, 8, 64 e 256.*

Supponiamo di voler impostare la frequenza del nostro segnale PWM a 20KHz con il prescaler a 1. Per il PIC24F a 32MHz abbiamo:

$$PR = \left(\frac{32MHz}{0.02MHz * 2 * 1} \right) - 1 = 799$$

Il che vuol dire che possiamo variare il Duty Cycle in un range da 0 a 799 (800 valori), che corrispondono a:

$$Risoluzione_{PWM} = \frac{LOG_{10}(800)}{LOG_{10}(2)} = 9.64bit$$

Se vogliamo fare la stessa cosa con il PIC32 che lavora a 40MHz abbiamo invece:

$$PR = \left(\frac{40MHz}{0.02MHz * 1 * 1} \right) - 1 = 1999$$

Il che vuol dire che possiamo variare il Duty Cycle in un range da 0 a 1999 (2000 valori) che corrispondono a 10,96bit:

$$Risoluzione_{PWM} = \frac{LOG_{10}(2000)}{LOG_{10}(2)} = 10.96bit$$

Capite quindi come su questi PICmicro la risoluzione del PWM non è fissa ma varia in base alle frequenze operative. Questo consente di ottenere una certa flessibilità: possiamo in pratica permetterci di impostare la frequenza del segnale PWM sul valore che desideriamo senza essere vincolati a valori tabellati come accade per altre MCU.

Ricordo ancora una volta che il suffisso “x” nella designazione dei moduli, va sostituito con il numero del modulo con il quale dobbiamo lavorare (da 1 a 5), mentre il suffisso “y” riferito ai Timers (Es: TMR_Y , PR_Y) è riferito al numero di timer scelto (2 o 3 sul PIC32 e da 1 a 5 sul PIC24).

Registri di controllo sul PIC24FJ64GB002

OCxCON1

E' il primo registro di controllo del modulo OCx. La mappa del registro è la seguente:

REGISTER 14-1: OCxCON1: OUTPUT COMPARE x CONTROL REGISTER 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2(2)	ENFLT1
bit 15							bit 8

R/W-0	R/W-0, HCS	R/W-0, HCS	R/W-0, HCS	R/W-0	R/W-0	R/W-0	R/W-0
ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2(1)	OCM1(1)	OCM0(1)
bit 7							bit 0

All'avvio tutti i bit si trovano già a zero. I settaggi principali di questo registro sono:

- Bit 0, 1 e 2 (OCM<2:0>). Impostano la modalità operativa del modulo OC. Per generare un segnale PWM questi 3 bit vanno impostati a **111** se vogliamo utilizzare la modalità PWM simmetrica (center-aligned), o a **110** se vogliamo utilizzare la modalità PWM asimmetrica (edge-aligned). Il modulo OCx non ha un bit di ON/OFF ma viene abilitato non apena si seleziona la modalità operativa: difatti di default questi tre bit si trovano a **000** che corrispondono a modulo OCx disattivato.

- Bit 12,11 e 10 (OCTSEL<2:0>). Permettono di selezionare il clock che va ad alimentare il timer dedicato OCxTMR. I valori sono presi dalla tabella seguente:

111	Clock di sistema
110	Riservato (non usare)
101	Riservato (non usare)
100	Timer 1
011	Timer 5
010	Timer 4
001	Timer 3
000	Timer 2 (impostazione di default)

Il Bit OCSIDL serve per fare in modo che il modulo OCx continui a funzionare anche se la CPU va in idle.

I Bit con il suffisso FLT (fault) servono per impostare la funzionalità del modulo OCx in caso di fault (malfunzionamento).

OCxCON2

Registro di controllo 2. Questo registro contiene alcuni settaggi avanzati, anch'essi disattivati di default e di cui non ci occuperemo. Ci interessano principalmente i bit da 0 a 4 (SYNCSEL<0:4>) che servono ad impostare la *sorgente di sincronizzazione*, ovvero: chi/cosa deve determinare il periodo del segnale.

La sorgente di sincronizzazione fa in modo che il registro timer dedicato, OCxTMR, si azzeri in concomitanza di un particolare evento. I settaggi possibili sono davvero tantissimi dato che si ha la possibilità di sincronizzare il modulo OCx anche con altre periferiche. Qui elenco solo alcune impostazioni:

11111	Auto-sincronizzazione
01011	Timer 1
01100	Timer 2
01101	Timer 3
01110	Timer 4
01111	Timer 5

Il settaggio 11111 : Auto-sincronizzazione significa che il timer dedicato OCxTMR si azzera quando raggiunge il valore impostato in OCxRS, per cui vedremo poi che è OCxRS, in una particolare modalità, ad impostare la lunghezza del periodo al posto del registro PR_Y.

Altro bit importante del registro OCxCON2 è il bit 12 (OCINV) che permette, se impostato a 1, di invertire la forma d'onda in uscita dal modulo OCx. Potrebbe tornare utile nel momento in cui due moduli OC devono generare lo stesso segnale ma invertito l'uno rispetto all'altro, ad esempio per il pilotaggio di un ponte H.

Registri di controllo sul PIC32MX250F128B

OCxCON

E' l'unico registro di controllo presente sul PIC32 e su tutti i PICmicro con modulo OC semplice. La mappa del registro è la seguente:

REGISTER 15-1: OCxCON: OUTPUT COMPARE 'x' CONTROL REGISTER

Bit Range	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
	ON(1)	—	SIDL	—	—	—	—	—
7:0	U-0	U-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
	—	—	OC32	OCFLT(2)	OCTSEL	OCM<2:0>		

Trattandosi di un PIC32, il registro è ovviamente a 32bit anche se i 16 bit alti sono tutti inutilizzati. Notate che la disposizione dei bit è pressapoco la stessa dell' OCxCON1 del PIC24 visto prima. Le differenze sono poche:

- Abbiamo un bit 15, chiamato ON, che permette di attivare/disattivare la periferica (sul PIC24 abbiamo invece visto che la periferica si attiva non appena si seleziona la modalità operativa).

- Abbiamo un solo bit OCTSEL, che anche qui serve a selezionare la sorgente di clock, ma possiamo scegliere solo 2 Timers: Timer3 (bit settato a 1) o Timer2 (bit a zero - opzione di default). Il Timer scelto viene utilizzato anche per il "conteggio". Sul PIC24 abbiamo invece visto che il conteggio viene mantenuto dal timer dedicato, per il quale vengono selezionati un clock e una sorgente di sincronia. Qui fa tutto lo stesso timer, per cui abbiamo delle limitazioni: il modulo OC può difatti generare solo segnali PWM asimmetrici.

- I Bit OCM<2:0> assumono anch'essi un'altro significato rispetto al PIC24: posti a 111 abilitano la modalità PWM con pin di fault abilitati. Posti a 110 abilitano la modalità PWM con pin di fault disabilitati.

Come dicevo prima, il PIC32 ha una sola modalità PWM che è quella asimmetrica, per cui entrambi i settaggi dei bits OCM producono sempre un segnale asimmetrico. La differenza tra i due valori sta quindi solo nella possibilità o meno di utilizzare i pin di fault (indicazione di malfunzionamento). Per applicazioni normali i pin di fault

generalmente non vengono utilizzati. Ne do comunque un cenno, starà al lettore approfondire.

> *Il modulo OC "semplice" presente su alcuni dsPic ha il registro OCxCON uguale a questo, con la sola differenza che è a 16bit e non ha i bit ON e OC32*

Pin di Fault

Quando abilitato/i, il/i pin di fault permette lo spegnimento del modulo OC attraverso un segnale logico a livello basso applicato ai pin contrassegnati come OCFA e OCFB.

> *Anche i pin OCFA e OCFB generalmente non hanno un'assegnazione fissa e vanno impostati tramite la funzione PPS. Altri PICmicro invece hanno questi pin assegnati. Fate riferimento al datasheet.*

Sul PIC32, il pin OCFA influenza i moduli OC da 1 a 4 mantre il pin OCFB influenza il modulo OC5. Sul PIC24F invece si può scegliere per ogni modulo se essere disattivato da OCFA o OCFB.

Se, in pratica, viene inviato un livello logico basso su tali pin e tale funzione è abilitata, il pin relativo al modulo associato va in uno stato di alta impedenza (sul PIC32) o in uno stato impostato (sul PIC24) e il modulo viene fermato.

I pin OCFA e OCFB devono inoltre essere impostati, oltre che con la funzionalità PPS, anche nel relativo registro TRIS come ingressi, se si desidera utilizzarli.

Impostazione registri su PIC24FJ64GB002

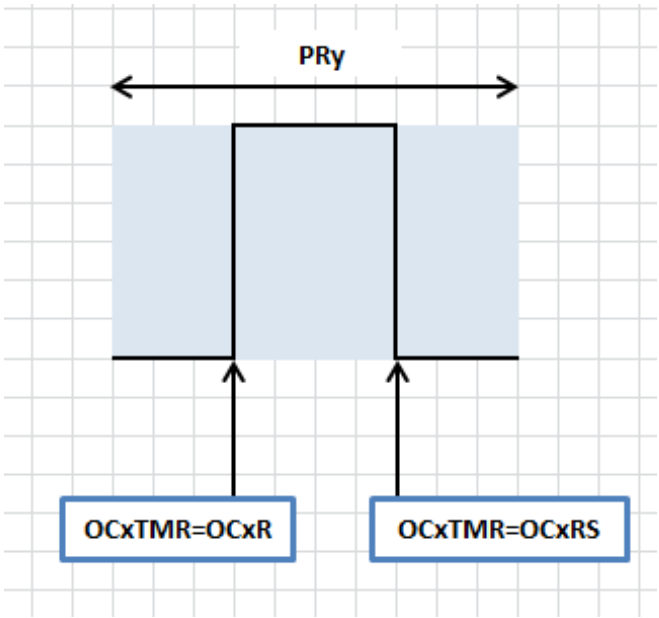
Generare un segnale PWM simmetrico (center-aligned)

Per questa modalità abbiamo visto che bisogna impostare a 111 i bits OCM del registro OCxCON1. Non appena tale modalità viene abilitata, il pin designato per l'uscita del modulo OCx si porta a livello logico basso.

Il Timer dedicato, OCxTMR, comincia il conteggio con una velocità determinata dalla sorgente di clock scelta tramite i bit OCTSEL. Non appena il conteggio eguaglia il valore impostato nel registro OCxR, il segnale va a livello alto.

OCxTMR continua ad incrementare e non appena arriva al valore impostato in OCxRS, il segnale va quindi a livello basso. In questo momento viene anche generato un interrupt, intercettabile dal flag _OCxIF.

Generalmente viene scelto un Timer come sorgente di sincronizzazione, per cui la lunghezza del periodo viene impostata dal registro PR_Y:



Il duty cycle è quindi determinato dalla differenza tra OCxRS e OCxR. Il programmatore deve pertanto preoccuparsi di variare questi due valori per poter modificare il duty cycle a proprio piacimento.

Il lettore più attento noterà che, in realtà, la modalità simmetrica non è necessariamente *allineata al centro*: dipende difatti da come vengono impostati i due registri. La microchip afferma sul datasheet che modalità center-aligned deve essere intesa come una modalità in cui si ha la possibilità di spostare il centro del segnale a

piacimento lungo il periodo. L'allineamento al centro, quindi, non è automatico ma è il programmatore che si deve impegnare a calcolare i valori giusti per fare in modo che lo sia.

Esempio di codice:

```
// Usando il PIC24FJ64GB002 con FOSC=32MHz
// prescaler del Timer2 a 1:1
// e desiderando una frequenza PWM a 20KHz
// il valore da caricare in PR2 è 799
PR2=799;

// Imposto il duty cycle agendo su OC1R e OC1RS
// da 0 a 200 = livello basso
OC1R=200;
// da 200 a 600 = livello alto
OC1RS=600;

// imposto OC1 per usare il Timer2 sia come
// sorgente di clock che come sorgente di
// sincronizzazione, pin di fault disabilitati
OC1CON1=0b0000000000000111; // center-aligned, clock=TMR2
OC1CON2=0b0000000000001100; // sincronia da TMR2
TMR2=0; // azzero il timer 2
T2CON=0;  // imposto il prescaler a 1:1
T2CONbits.TON=1;    // attivo il timer
```

Il codice di esempio mostrato fa uso del Timer2 sia come sorgente di clock (bits OCTSEL=000) sia come sorgente di sincronia (SYNCSEL=01100). Avendo selezionato il Timer2 come sorgente di sincronia, il periodo va quindi impostato nel registro periodo del Timer2, PR2. In questo esempio il livello alto dura 600-200=400, ovvero la metà del periodo impostato, per cui avremo un duty cycle del 50% e le durate del livello basso prima e dopo il livello alto varranno 200 e 200 (da 0 a 200 e da 600 a 800). Il segnale quindi è davvero allineato al centro!

Per poter visualizzare su un oscilloscopio il momento in cui il periodo inizia, potete attivare l'interrupt su overflow del Timer2 (_T2IE=1) e quindi intercettare, tramite ISR, il flag _T2IF e fare il toggle di un pin. Sull'oscilloscopio impostate il trigger su questo pin piuttosto che sul segnale PWM.

In questa modalità avremo un duty cycle dello 0% (segnale sempre a livello logico basso) quando OC1RS vale zero e OC1R è più grande del periodo impostato.

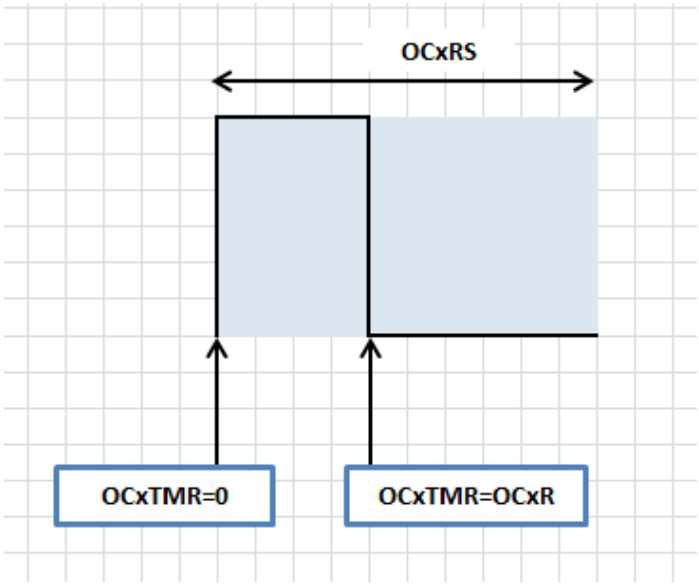
Avremo un duty cycle del 100% (segnale sempre a livello logico alto) quando OC1R vale zero e OC1RS è uguale al periodo.

Generare un segnale PWM asimmetrico (edge-aligned)

Per questa modalità abbiamo visto che bisogna impostare a 110 i bits OCM del registro OCxCON1.

Il pin designato per l'uscita del modulo OCx si porta a livello logico alto quando OCxTMR=0 (sincronia), in questo momento viene anche impostato il flag di interrupt _OCxIF.

Quando il timer dedicato raggiunge il valore di OCxR, il pin va a livello logico basso. Per questa modalità viene generalmente scelto il modulo stesso come sorgente di sincronia (auto-sincronizzazione) pertanto è il registro OCxRS a determinare il periodo:



In questo caso il Timer eventualmente scelto fornisce unicamente il segnale di clock (bit OCTSEL), ovvero determina la velocità di incremento del timer, ma non il periodo.

Esempio di codice:

```
// Usando il PIC24FJ64GB002 con FOSC=32MHz
// prescaler del Timer2 a 1:1
// e desiderando una frequenza PWM a 20KHz
// il valore da caricare in OC1RS è 799
OC1RS=799;

// Imposto il duty cycle agendo su OC1R
// duty cycle del 50%
OC1R=400;

// imposto OC1 per usare il Timer2 come
// sorgente di clock mentre come sorgente di
// sincronizzazione scelgo il modulo stesso
OC1CON1=0b0000000000000110; // center-aligned, clock=TMR2
```

```
OC1CON2=0b0000000000011111; // auto-sincronia
TMR2=0; // azzero il timer 2
T2CON=0;  // imposto il prescaler a 1:1
T2CONbits.TON=1;    // attivo il timer
```

Il codice di esempio mostrato fa uso del Timer2 solo come sorgente di clock (bits OCTSEL=000) mentre come sorgente di sincronia è stato scelto il modulo stesso (SYNCSEL=11111). Avendo selezionato il modulo stesso come sorgente di sincronia, il periodo va impostato nel registro OC1RS. Il duty cycle (durata livello alto) va invece impostato in OC1R.

Per poter visualizzare su un oscilloscopio il momento in cui il periodo inizia, potete attivare l'interrupt del modulo OC1 (_OC1IE=1) e quindi intercettare, tramite ISR, il flag _OC1IF e fare il toggle di un pin. Sull'oscilloscopio impostate il trigger su questo pin piuttosto che sul segnale PWM.

In questa modalità avremo un duty cycle dello 0% (segnale sempre a livello logico basso) quando OC1R vale zero.

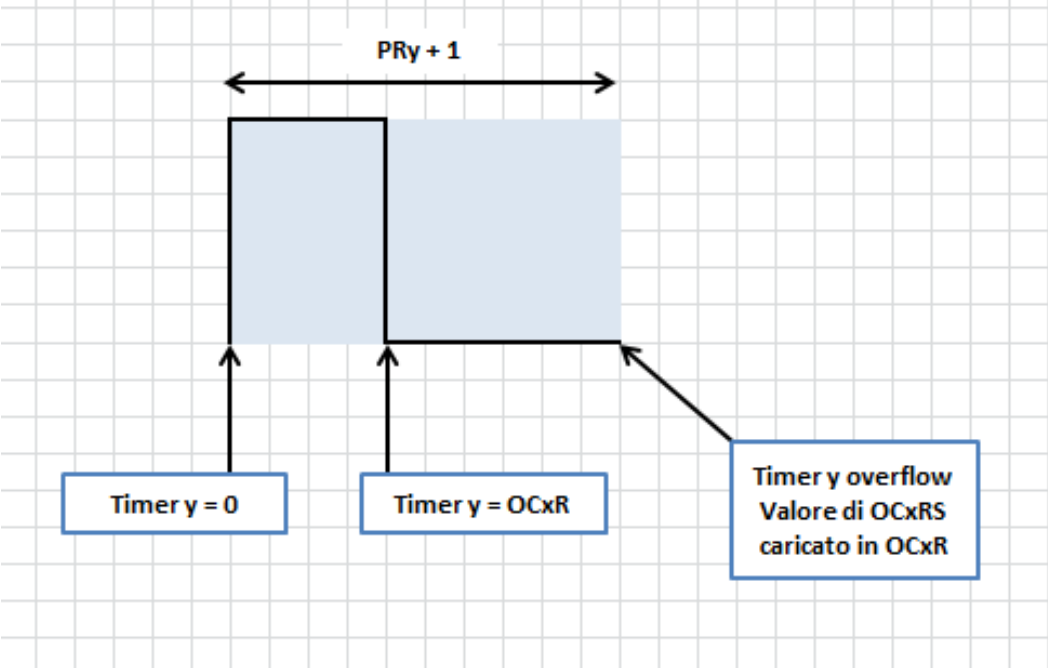
Avremo un duty cycle del 100% (segnale sempre a livello logico alto) quando OC1R è più grande del periodo.

Impostazione registri su PIC32MX250F128B

Abbiamo visto che il PIC32 preso ad esempio ha un modulo OC semplice, quindi senza timers dedicati, e abbiamo un solo registro OCxCON che ci permette di selezionare quale timer deve fare tutto il lavoro (TMR2 o TMR3).

Sincronia e clock vengono pertanto forniti dallo stesso Timer selezionato e il periodo viene di conseguenza fornito caricando il registro PR_Y.

I registri OCxR e OCxRS hanno la stessa funzione: quella di determinare il duty cycle. In particolare il registro OCxR viene caricato solo una volta durante il setup, con il valore di duty cycle iniziale. Dopo che il modulo è stato attivato, OCxR diviene in sola lettura e il valore di duty cycle verrà impostato unicamente in OCxRS. OCxR in pratica diviene un buffer secondario in cui in automatico il modulo trasferisce il valore che noi carichiamo in OCxRS:



Esempio di codice:

```
// Usando il PIC32MX250F128B con FOSC=40MHz
// prescaler del Timer2 a 1:1
// e desiderando una frequenza PWM a 20KHz
// il valore da caricare in PR2 è 1999
PR2=1999;

// Imposto il duty cycle iniziale agendo su OC1R
// duty cycle del 50%
OC1R=1000;
// dopo che il modulo viene avviato, il duty cycle
// verrà prelevato da OC1RS
OC1RS=1000;
// imposto OC1 per usare il Timer2 come sorgente
OC1CON=0b0000000000000110; // pin fault disattivati, clock=TMR2
TMR2=0; // azzero il timer 2
T2CON=0;  // imposto il prescaler a 1:1
T2CONbits.TON=1;    // attivo il timer
OC1CONbits.ON=1;    // attivo il modulo
```

Vedete che durante il setup, non abbiamo messo a 1 il bit 5 di OC1CON (ON) ma questa operazione viene fatta in ultima istanza. Il periodo è determinato dal valore di PR_Y + 1.

Da questo momento in poi il duty cycle verrà impostato modificando il valore in OC1RS.

In questa modalità avremo un duty cycle dello 0% (segnale sempre a livello logico basso) quando OC1R / OC1RS vale zero.

Avremo un duty cycle del 100% (segnale sempre a livello logico alto) quando OC1R / OC1RS è più grande del periodo.

Links

- PIC24F Family Reference Manual (fate riferimento alla sezione 35: Output Compare with dedicated Timer)
- PIC32 Family Reference Manual (fate riferimento alla sezione 16: Output Compare)
- Tutorial PWM Mauro Laurenti
- Video PWM Simmetrico: http://www.youtube.com/watch?v=JoeUPGMcUqE
- Video PWM Asimmetrico: http://www.youtube.com/watch?v=pTKhkrXPMJA

Per gli utenti di ORbit16™ sono disponibili alcune demo, relativamente all'utilizzo dei moduli OC per la generazione di segnali PWM (sia su PIC2 che su PIC32), alla pagina:

http://www.settorezero.com/orbit16/home/examples/

Le demo illustrate sono utili sia perché illustrano come variare il duty cycle sfruttando un potenziometro, sia perché illustrano come ridirezionare il modulo OC su un pin a piacere sfruttando la funzione PPS, che sul PIC32 ha un utilizzo differente rispetto a quella del PIC24.

Supporta Settorezero.com con una ORbit16™