www.asap-firmware.com

Prefazione

Tecnica Pulse Width Modulation (PWM)
Periferica PPG su microcontrollori Fujitsu
PPG a 16 bit
Conversione Proporzionale
La soluzione ASAP
Esempi in linguaggio C

Titolo: FWS0003A

Uscita PWM Proporzionale

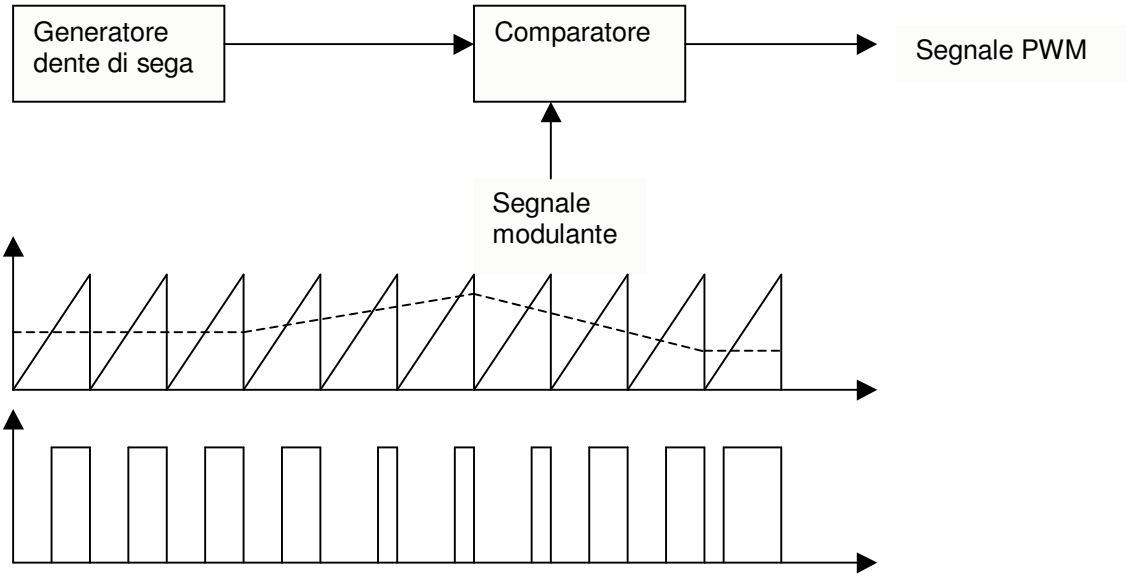
Tecnica Pulse Width Modulation (PWM)

In generale la modulazione consiste nel modificare le caratteristiche di un segnale, detto *portante*, in funzione di un altro segnale, detto *modulante*; la forma d'onda che si ottiene è detta *segnale modulato*.

Scopo della modulazione è di trasformare un segnale in un altro di forma più appropriata per essere utilizzato per determinate finalità; ad esempio nel caso di un controllo luminosità di una lampada (dimmer) la modulazione consiste nel traslare il segnale di regolazione in modo da renderlo atto a parzializzare la tensione di alimentazione alternata.

Nella modulazione di larghezza ad impulsi (PWM) è la larghezza (durata) degli impulsi che varia in funzione dell'informazione, mantenendo costanti l'ampiezza e il periodo; offre il vantaggio di un più alto rendimento, potendo il trasmettitore lavorare a livelli di saturazione.

Un metodo per ottenere la modulazione PWM è dimostrato nelle seguenti figure:



Una forma d'onda a dente di sega è applicata al comparatore, il cui livello di soglia è funzione del segnale modulante e quindi lo sarà anche la durata degli impulsi all'uscita del comparatore; la demodulazione può essere ottenuta applicando il segnale PWM ad un filtro passa basso.

Il passaggio da questo sistema analogico ad un sistema digitale è molto simile: in un microcontrollore la forma d'onda a dente di sega corrisponde al valore di un timer ed il comparatore corrisponde ad un registro che contiene il valore di "match"/ "compare".

Periferica PPG su microcontrollori Fujitsu

Nella famiglia di microcontrollori a 16 bit F2MC-16LX Fujitsu la periferica PWM è denominata PPG timer (Programmable Pulse Generator).

Esistono dei modelli con funzioni dedicate al controllo motori (dead time e pin per converter trifase), mentre in genere troviamo una periferica 8/16bit PPG timer con quattro registri di reload a 8 bit (PRLH0, PRLL0, PRLH1, and PRLL1) e due PPG down counters (PCNT0 e PCNT1).

Il "count clock" del timer può essere selezionato tra più clocks interni; il periodo dell'impulso è determinato dalla somma dell'ampiezza dell'intervallo High e dell'intervallo Low; un interrupt può essere generato alla fine di ogni intervallo; l'impulso viene trasferito al pin dedicato solo se abilitato da software.

L'ampiezza dell'intervallo è determinata dal valore presente nei registri di reload (PRLxn) e poi caricato nel down counter (PCNTn); è importante sapere che il valore utilizzato corrisponde a quello presente in PRLxn alla fine dell'intervallo High (fronte di discesa del pin).

Per non ottenere impulsi errati, dovuti al fatto che il match-compare può avvenire nell'esatto momento del reload, sono obbligatorie le seguenti precauzioni:

1. se PPG timer è usato in modo "8-bit PPG indipendent output" oppure in modo "8+8-bit PPG output", utilizzare una istruzione a word per impostare al durata di entrambe gli intervalli scrivendo i reload registers (PRLLn/PRLHn) nello stesso momento
2. quando PPG timer e usato come "16bit output", i valori di reload scritti nei registri PRLL0/PRLH0, vengono memorizzati in un latch temporaneo, e trasferiti simultaneamente alla scrittura nei registri PRLL1/PRLH1; perciò se si scrive il valore usando un'istruzione a word è necessario seguire l'ordine PPG0 e poi PPG1, oppure si scriverà il valore con una sola istruzione a long.

PPG a 16 bit

In questa Soluzione si è utilizzata la periferica PPG nella modalità a 16bit, ottenendo così una risoluzione di 2^16 bit per ogni intervallo (High e Low) e la stessa forma d'onda sui due pin dedicati.

Utilizzando la possibilità di abilitare o meno il pin di uscita si è realizzata una periferica PWM "duale", con lato positivo e lato negativo; i pin possono essere sempre connessi al trasduttore ma diventano "attivi" solo in corrispondenza della propria "zona di lavoro".

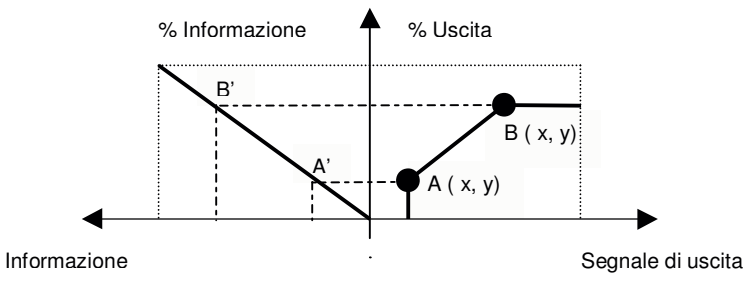
In questo modo il valore dell'informazione da convertire viene suddiviso in due zone indipendenti, ognuna con una propria caratteristica di uscita.

Conversione Proporzionale

Il valore dell'informazione da trasferire può provenire da una qualsiasi fonte eleborata dal microcontrollore, ad esempio il risultato di un algoritmo di controllo digitale, il valore di un segnale acquisito tramite A/D converter, un dato trasmesso dall'unità "master" all'unità "driver" via CAN Bus (I/O distribuito), ottenendo così massima flessibilità.

Il modo più semplice per trasferire l'informazione è convertirla direttamente nell'impulso PWM, ma a volte può risultare necessario legare la conversione ad una certa legge matematica.

In questa Soluzione, il valore dell'informazione viene trasferito all'attuatore in modo Proporzionale secondo l'equazione di una retta, come evidenziato nella figura seguente:



Nella parte sinistra del grafico troviamo una retta passante per l'origine che identifica l'andamento dell'informazione da 0 al 100%; nella parte destra del grafico troviamo una retta che identifica la caratteristica del segnale di uscita (PWM) in base al relativo valore dell'informazione di ingresso.

L'inizio e la fine della retta della caratteristica di uscita sono determinati dalle coordinate dei due punti A e B; il punto A definisce la compensazione dell'offset e il valore iniziale, il punto B definisce il guadagno e il limite massimo.

Analizzando il grafico si vede come è possibile mantenere l'uscita nulla fino al valore A' dell'ingresso, quindi incrementare con un determinato guadagno fino al punto B' e quindi saturare al valore massimo.

Rendendo tarabili le coordinate "x,y" dei due punti A e B si ottengono delle configurazioni particolari che si possono adattare alle caratteristiche dell'attuatore; infatti questo significa modificare l'equazione della retta come evidenziato dai seguenti grafici:

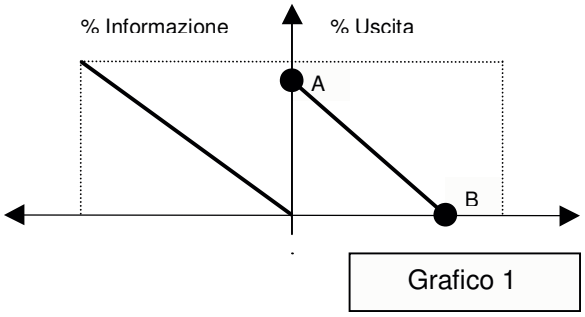


Grafico 1

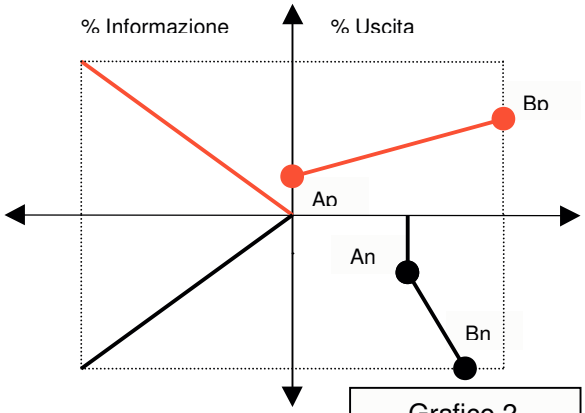


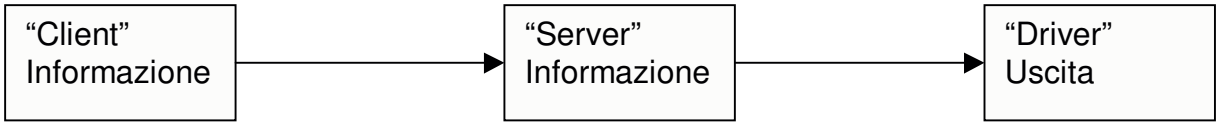
Grafico 2

Nel Grafico 1 all'aumentare del segnale di ingresso otteniamo una diminuzione del segnale di uscita.

Nel Grafico 2 sono riportate le caratteristiche di entrambe le uscite, lato positivo e lato negativo; nel lato positivo si inizia con un valore fisso e si incrementa molto lentamente, mentre nel lato negativo si rimane con uscita nulla fino a metà scala per poi aumentare molto rapidamente.

La soluzione ASAP

Il principio di funzionamento può essere schematizzato come segue:

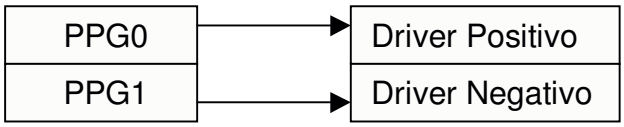


L'unità che acquisisce l'informazione viene denominata Client; l'unità che contiene l'informazione e la elabora viene denominata Server; l'unità che trasferisce l'informazione all'attuatore viene denominata Driver.

Queste unità possono essere contenute nello stesso microcontrollore, oppure essere schede diverse collegate via CAN Bus.

Utilizzando l'unità PPG del microcontrollore Fujitsu a 16 bit si ottiene sui due pin di uscita dedicati la stessa forma d'onda.

E' possibile collegare i due pin ad un attuatore per il lato positivo e ad uno per il lato negativo, oppure ad un unico componente che controlla entrambe le direzioni.



Il software in base al valore contenuto nell'informazione di ingresso abiliterà una delle due uscite PWM.

Esiste però un problema di sincronizzazione nel passaggio tra i due lati; infatti bisogna evitare che le due uscite PWM siano attive contemporaneamente.

Per questo è necessario realizzare un driver software dedicato in grado di gestire il cambio di uscita come pure il cambio di periodo PWM run-time.

Il momento di sincronizzazione è stato individuato nella fine dell'intervallo High della PWM, sul fronte di discesa del pin PWM.

A causa dei tempi di latenza e di esecuzione dell'interrupt le condizioni diventano più critiche sia al diminuire del periodo PWM sia in prossimità degli estremi del duty-cycle.

Può infatti succedere che l'interrupt venga eseguito dopo la durata di uno dei due intervalli PWM, con il pericolo di sincronizzarsi in modo errato.

L'algoritmo del driver software introdurrà dei controlli per verificare se il momento corrisponde a quello desiderato; infatti se il livello del pin PWM è basso il sincronismo si considera corretto, e se il livello del pin PWM è alto si può essere all'inizio dell'intervallo High, oppure in ritardo per rilevare il pin a livello basso.

Per ogni ulteriore informazione, ASAP è raggiungibile al seguente indirizzo di posta elettronica :
info@asap-firmware.com

Esempi in linguaggio C

Funzione per l'impostazione del periodo PWM:

```
boolean PwmUpdateFreq (byte nch, word freq)
{
boolean esito;

   esito= FALSE;
   /* controllo N. di canale e valore della frequenza */
   if ( (nch<MAX_CH_PWM) && (freq<MAX_FREQ_PWM) && (freq>_MIN_FREQ_PWM) )
      {
      pwm[nch].freq= freq;
      /* attiva richiesta di modifica */
      DriverPwm[nch].FreqUpd= 1;
      esito= TRUE;
      }

   return (esito);
}
```

Funzione per l'impostazione del duty-cycle:

```
boolean PwmUpdateDuty (byte nch, word duty)
{
boolean esito;

   esito= FALSE;
   /* controllo N. di canale e valore del duty */
   if ( (nch<MAX_CH_PWM) && (abs(duty)<=MAX_DC_PWM) )
      {
      pwm[nch].duty= duty;
      /* attiva richiesta di modifica */
      DriverPwm[nch].DutyUpd= 1;
      esito= TRUE;
      }

   return (esito);
}
```

Descrizione funzione driver software:

```
1) Scansione canali PWM positivo e negativo …
for (i=0; i<MAX_CH_PWM; i++)
    {……
2) Verifica richiesta aggiornamento periodo PWM ……
   if (DriverPwm[i].FreqUpd==1) ……
   {……
3) Scelta della sorgente di clock più adatta per ottenere la massima risoluzione
4) Verifica richiesta aggiornamento duty cycle
   if (DriverPwm[i].DutyUpd) ……
      {……
5) Calcolo dei nuovi valori di impostazione dei registri di reload
   DriverPwm[i].prl= ValueLow;
   DriverPwm[i].prh= ValueHigh;……
```

Descrizione funzione servizio interrupt:

```
1) Rilevazione sicura punto di aggiornamento
2) Verifica richiesta di aggiornamento……
3) if (DriverPwm[CH_PWM0].HwUpd)……
     {……
4) Impostazione frequenza PWM
5) Impostazione duty-cycle
```