www.asap-firmware.com

Prefazione

Teoria della progettazione della logica di controllo
Organizzazione delle informazioni
Elaborazione delle informazioni
La soluzione ASAP
Esempi in linguaggio C

Titolo: FWS0002A

Logica di controllo

Teoria della progettazione della logica di controllo

In una apparecchiatura le informazioni vengono visualizzate attraverso l'interfaccia utente, come già discusso ampliamente nella soluzione firmware FWS0001A.

L'obiettivo principale è consentire all'operatore di avere una buona consapevolezza sullo stato del sistema e per questo è necessario progettare in modo appropriato anche la logica di controllo dell'apparecchiatura.

Per logica di controllo possiamo intendere l'insieme delle modalità che consentono l'acquisizione e l'elaborazione delle informazioni, le quali devono essere implementate nel firmware in modo da garantire affidabilità, velocità di esecuzione e flessibilità combinatoria.

Per garantire l'*affidabilità* bisogna evitare di considerare come valide informazioni derivanti da eventi transitori e indesiderati e per questo è consigliabile uno studio approffondito sul funzionamento del sistema da controllare, in collaborazione con i progettisti delle varie parti che lo costituiscono.

Queste parti si possono pensare come dei blocchi aventi dati in ingresso e in uscita, che comunicano tra di loro in cascata per evitare di manipolare dati in posizioni diverse ottenendo contemporaneamente risultati non omogenei e quindi non affidabili.

La *velocità di esecuzione* si aumenta organizzando le informazioni da elaborare nel modo più omogeneo possibile e utilizzando variabili in memoria che consentono un accesso veloce; per questo è necessario anche uno studio della struttura del controllore che si sta utilizzando, analizzando le caratteristiche delle aree di memorie e delle istruzioni dedicate ai relativi accessi.

Ovviamente la memoria interna in genere è quella che consente maggiori prestazioni, meglio ancora se è possibile indirizzarla a singoli bit.

Per ottenere la *flessibilità combinatoria* è necessario utilizzare le operazioni booleane tipicamente a disposizione (OR,AND,XOR ecc.) integrate con altre che realizzano funzioni speciali come ad esempio filtri temporizzati e Flip-Flop.

Organizzazione delle informazioni

Le informazioni dovranno essere organizzate in gruppi omogenei.

Generalmente in una apparecchiatura si possono individuare :

1. segnali provenienti da sensori suddivisibili in discreti (on/off) e analogici (A/D)
2. segnali da fornire agli attuatori suddivisibili in discreti (on/off) e analogici (D/A)
3. stati di funzionamento
4. condizioni di allarme
5. dati eterogenei
6. comandi utente

Le informazioni di tipo on/off (1-0) saranno rappresentate con valori a bit, mentre le informazioni analogiche saranno rappresentate con variabili che dipendono dalla precisione desiderata (byte,integer, long, float..).

Per non diminuire la velocità di esecuzione è consigliabile comunque non utilizzare floating point e long preferendo formati integer amplificandone il valore per poter rappresentare anche cifre decimali; ad esempio il valore 100.03 può essere contenuto nella variabile integer come valore 10003 e rappresentato con due cifre decimali solo al momento della visualizzazione nel pannello di controllo.

Elaborazione delle informazioni

Il primo passo nell'elaborazione delle informazioni consiste nell'acquisizione delle grandezze di ingresso provenienti dal campo, come ingressi discreti e analogici.

Quest'ultimi saranno acquisiti tramite il convertitore A/D e convertiti in valori numerici in base ad un valore di fondoscala; gli ingressi discreti saranno copiati in una mappa che conterrà la loro immagine durante l'elaborazione.

A questo punto le informazioni ottenute sono combinate tra loro in base alla logica di controllo per generare dati in uscita come stati di funzionamento, condizioni di allarme informazioni di visualizzazione.

L'ultimo passo consiste nel comandare gli attuatori.

E' consigliabile mantenere questa sequenza ed eseguirla ciclicamente con un periodo di tempo inferiore alla velocità del sistema da controllare.

La soluzione ASAP

L'esperienza maturata da ASAP nella progettazione del firmware di questo tipo ha portato alla creazione di metodi e modelli ormai consolidati.

Come base di partenza si definisce la durata del periodo di controllo; questo dipende dalla velocità di esecuzione che può garantire il controllore e dalla velocità con cui cambiano i dati del sistema da controllare.

Per dare un'idea si può dire che in genere questo tempo può variare da pochi a centinaia di millisecondi.

Acquisizione ingressi discreti

L'acquisizione degli ingressi discreti viene eseguita da una funzione che legge i valori di input e li converte in bit.

Per aumentare l'affidabilità, viene eseguito un filtro software per non acquisire valori transitori e solo ad acquisizione ultimata verranno avvisati i processi in cascata.

Acquisizione ingressi analogici

Se non esistono particolari problemi nelle tempistiche, generalmente l'acquisizione degli ingressi analogici viene eseguita a rotazione.

Ogni periodo di acquisizione è formato da più letture dal convertitore A/D e concluso con un calcolo della media dei valori; per tradurre i valori in bit in grandezze reali la media viene moltiplicata per un fondocala e divisa per la risoluzione del convertitore (processo di normalizzazione della misura).

A livello firmware un indice progressivo potrà puntare ad una serie di vettori di dimensione pari al numero di misure che conterranno i valori necessari ai calcoli eseguiti durante questo procedimento.

Stati di funzionamento

Gli stati di funzionamento sono rappresentati con valori a bit, risultano dall'elaborazione degli ingressi digitali e analogici acquisiti e si riferiscono a condizioni di funzionamento normali.

Condizioni di allarme

Gli allarmi sono rappresentati con valori a bit, risultano dall'elaborazione degli ingressi digitali e analogici acquisiti e si riferiscono a condizioni di funzionamento critiche.

Dati eterogenei

Con questo termine ci riferiamo a valori a bit che risultano dall'eleborazione di informazioni diverse da quelle elencate; un esempio può essere il confronto di una misura con una soglia di preallarme, l'informazione della presenza di uno stato per un certo tempo o l'invio di un comando da parte dell'utente.

Comandi utente

Anche l'utente può influenzare la logica di controllo per mezzo di comandi inviati per eseguire determinate azioni.

I comandi vengono acquisiti e accodati in un buffer circolare in modo da evitare eventuali sovrapposizioni ed eseguirli nella sequenza corretta.

Funzioni

La logica combinatoria di tutte queste informazioni si esegue sfruttando gli operatori booleani già presenti nel linguaggio C :

AND – operatore &
OR – operatore |
XOR – operatore ^
NOT – operatore !

A volte infatti si vuole creare un'informazione in uscita che deriva dalla presenza o dall'assenza di un ingresso, oppure si vuole memorizzare una condizione e azzerarla in seguito, e per questo sono state create delle funzioni speciali aggiuntive:

MonostabileRitardatoInSalita – finchè l'input non dura per il tempo specificato ritorna un output nullo

MonostabileRitardatoInDiscesa – mantiene attivo l'output dopo l'assenza dell'input per il tempo specificato

Flip-Flop RS – mantiene l'output Q attivo dopo aver rilevato l'input S e lo azzera solo se rileva l'input R.

Le tempistiche vengono gestite attraverso dei vettori contenenti i temporizzatori.

Questi funzionano come conto alla rovescia e si decrementano a piacere in base alla risoluzione desiderata.

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

Esempi in linguaggio C

Vettore contenente i fondoscala misure :

```
const int fondoscala[MAX_MISURE]=
/* 10.0  -  200  -  100.00*/
{100,       200,     10000 };
```

Funzione NormalizzazioneMisura :

```
int NormalizzaMisura (int media, int fondoscala)
{
   return(media*fondoscala/RIS_AD_CONVERTER);
}
```

Funzione MonostabileRitardatoInSalita :

```
boolean MonoRS(boolean input,word *ptTempo,word reload)
{
boolean output;

   if (input)
      {
      if (*ptTempo==0)
         output= TRUE;
      }
  else
     {
      *ptTempo = reload;
      output= FALSE;
     }
   return(output);
}
```

Funzione MonostabileRitardatoInDiscesa :

```
boolean MonoRD(boolean input,word *ptTempo,word reload)
{
boolean output;

   if (input)
      {
      *ptTempo = reload;
      output= TRUE;
      }
  else
     {
     if (*ptTempo==0)
         output= FALSE;
     }
   return(output);
}
```

Funzione FlipFlopRS :

```
boolean FlipFlopRS(boolean S, boolean R, boolean old)
{
boolean output;

   if (S==TRUE)
      output= TRUE;
   else if (R==TRUE)
      output= FALSE;
   else
      output= old;
   return(output);
}
```