Giovanni Schgör (g.schgor)

LOGICA IN JAVA PER SCU

8 June 2011

Proseguendo nell'illustrazione di programmi tipici per le **applicazioni** dello SCU (vedi precedente), si vuole qui esaminare la possibilità di utilizzare la scheda di valutazione per funzioni tipiche di un **PLC**. Ovviamente la grande differenza sta nella forma di programmazione: non Ladder(KOP) né AWL, ma direttamente in linguaggio JAVA.

Operazioni logiche

Entriamo subito in argomento con un programma che mostra l'uso delle 3 funzioni logiche fondamentali (**NOT, AND,OR**) fra 2 variabili (booleane) A e B, fisicamente rappresentate dallo stato degli interruttori d'ingresso RC4 ed RC5:

```
// Dichiarazione variabili
    boolean A,B,R0,R1,R2;
// Loop
  // Lettura ingressi (c4,c5)
    scu.send("get c4\r");
    risp=scu.getAns();
    if (risp.contains("1")) A=true; else A=false;
    scu.send("get c5\r");
    risp=scu.getAns();
    if (risp.contains("1")) B=true; else B=false;

  // Operazioni logiche
     R0=!A;  // NOT A
     R1=A&B; // A AND B
     R2=A|B; // B OR B

  // Comandi uscite (b4,b5,b6)
    if (R0) scu.sendCmd("set b4\r");
       else scu.sendCmd("clr b4\r");
    if (R1) scu.sendCmd("set b5\r");
       else scu.sendCmd("clr b5\r");
    if (R2) scu.sendCmd("set b6\r");
       else scu.sendCmd("clr b6\r");
// Fine Loop
```

Come ormai dovrebbe essere noto dalla lettura del precedente articolo, è necessario dichiarare le variabili in gioco nell'apposito spazio del programma Base_SCU_EVB ed

incollare nello stesso il testo fra Loop e Fine Loop. Avviando il programma di base, si hanno le accensioni dei LED d'uscita della scheda (RB4, RB5, RB6) in corrispondenza degli stati degli ingressi A e B, in accordo con le rispettive operazioni logiche svolte.
Sulla notazione delle funzioni logiche, non resta che prenderne atto ed imparare ad utilizzarle (se i concetti sono chiari, la simbologia non è certo un problema).

Si sottolinea la struttura del programma che, come nei PLC, è costituita da una prima parte di "lettura degli ingressi", da una seconda di "logica" (il programma di controllo vero e proprio) e da una terza per "l'attivazione delle uscite". Si consiglia di mantenere questa struttura in tutte le applicazioni.

Dopo aver avviato il programma, si raccomanda di eseguire tutte le combinazioni di stato della variabili d'ingresso e di verificare sui relativi LED gli stati delle singolo uscite: è così possibile costruire una *tabella della verità*, verificando ciascuna funzione.
Per completare il quadro delle operazioni logiche possiamo aggiungere le cosiddette funzioni derivate (NAND,NOR,XOR). Per questo basta sostituire la parte centrale del programma con le istruzioni seguenti:

```
// Operazioni logiche
  R0=!(A&B);  // NOT(A AND B)->(NAND)
  R1=!(A|B); // Not(A OR B)  ->(NOR)
  R2=A^B; // (NOT A AND B)OR(A AND NOT B) ->(XOR)
```

Anche con queste funzioni si consiglia di "verificare con mano" le corrispondenti nuove tabelle della verità.

Con ciò abbiamo la possibilità di "programmare" qualsiasi problema di logica combinatoria (e il risultato non è davvero da poco!).

Esempio di applicazione

Supponiamo un impianto con 2 pompe che devono mantenere pieno un serbatoio.
La prima (P1) è la principale e la seconda (P2) è di emergenza, dovendo intervenire solo se la prima è in blocco. Ciascuna pompa è infatti dotata di un contatto di abilitazione che, se chiuso, ne permette la marcia: chiamiamo questi contatti AbP1 e AbP2. Se entrambi aperti (quindi impossibilità di marcia per entrambe le pompe) deve essere dato un allarme mediante LED lampeggiante.
Il riempimento del serbatoio è inoltre segnalato da 2 contatti di livello (Liv1 e Liv2, uguali e ridondanti) e la chiusura di almeno uno dei due arresta la pompa che è in quel momento è in marcia.

Sia chiaro che non propongo di fare il controllo di due pompe con un PC (!), ma l'approccio con la scheda in questione è (coma già dichiarato) propedeutico all'uso di microcontrollori: l'attuale tecnologia spinge infatti ad utilizzare sempre meno hardware e sempre più software. E' dunque in quest'ottica che la scheda "didattica" può essere assunta come simulatore del controllo dell'impianto.
Si può ed es. assegnare a due LED (RC4 ed RC5) la simulazione della marcia (LED acceso) di ciascuna pompa, e ad un terzo LED (RC7)la funzione di allarme, mentre gli ingressi possono essere rappresentati dagli interruttori (RB4 ed RB5 per le abilitazioni e RB6 ed RB7 per i contatti di livello).
Abbiamo ora tutti gli elementi sufficienti a scrivere un programma di controllo che risponda alla "specifica" data, ma si lascia questo compito al lettore che potrà verificare direttamente sulla scheda il comportamento del programma stesso.

Logica sequenziale

La logica combinatoria non è quasi mai sufficiente per svolgere problemi di automazione: occorre introdurre il concetto di "memorizzazione dei segnali", quindi le configurazioni tipiche della logica sequenziale. Si ricorda che la differenza fra combinatoria e sequenziale è che nella prima gli stati delle uscite sono univocamente definite dagli stati degli ingressi, mentre nella seconda la configurazioni delle uscite può differire a parità di configurazione degli ingressi.
Essenzialmente un circuito di memorizzazione è realizzato con FlipFlop che nella sua forma più semplice è comandato da due segnali (Set e Reset, quindi indicato come SR-FF). Già questo può coprire gran parte delle applicazioni (altri tipi più sofisticati di FF, saranno visti in seguito), aggiungendo però almeno un altro dispositivo: il temporizzatore (o monostabile).

SR-FF

Il circuito ad elementi NOR che realizza un Set/Reset-FlipFlop è ben noto (2 NOR con collegamenti incrociati, i rispettivi ingressi S ed R e le uscite NOT(Q) e Q). La versione in Java non è altro che l'espressione ricavata da questo circuito, a cui devono aggiungersi la letture degli ingressi (rispettivamente pulsante RC0 per il Set ed il pulsante RC1 per il Reset), nonché il comando dell'uscita (il LED RB4 per Q):

```
// Dichiarazione variabili
    boolean S,R,Q;
//  Inizializzazione
    Q=false;
 // Loop
   // Lettura ingressi c0->S(Set) e c1->R(Reset)
      scu.send("get c0\r");
      risp=scu.getAns();
      if (risp.contains("1")) S=true; else S=false;
      scu.send("get c1\r");
      risp=scu.getAns();
      if (risp.contains("1")) R=true; else R=false;
    // SR-FF
      Q=!(R|!(S|Q));
    // Attivazione uscita (Q->b4)
      if (Q) scu.sendCmd("set b4\r");
         else scu.sendCmd("clr b4\r");
  // Fine Loop
```

Con l'avviamento del programma si può infatti constatare che un breve impulso su S fa accendere Il LED RC4 e questo rimane acceso fino a che non si dà un impulso su R.
E' questa la versione elettronica dei classici relè con contatto di ritenuta (tipici dei circuiti di Marcia/Arresto a pulsanti) e una sua applicazione può trovarsi nel completamento dell'esempio di automatismo visto prima: gli stessi pulsanti della scheda possono avviare od arrestare la marcia della pompa destinata al riempimento del serbatoio, secondo la logica specificata.

Monostabile

Altro fondamentale elemento per dispositivi di automazione è il circuito monostabile, che è l'equivalente del classico "relè temporizzato", oppure della particolare configurazione di un 555.
Ne esistono diverse versioni, fra le quali qui è sviluppata quella che viene attivata dal fronte d'onda del segnale di ingresso, e l'uscita rimane attivata per il tempo stabilito, indipendentemente dalla durata del segnale d'ingresso.
Il programma relativo è il seguente:

```
// Dichiarazione variabili
      boolean A,Ar;
// Inizializzazioni
      Ar=false;
// Loop
 // Lettura ingresso
    scu.send("get c0\r");
    risp=scu.getAns();
    if (risp.contains("1")) A=true; else A=false;
 // Ritardo
    if (A&!Ar) timers[0]=30;
    if (A) Ar=true else Ar=false;
    if (timers[0]!=0)   scu.sendCmd("set b4\r");
       else  scu.sendCmd("clr b4\r");
 // Fine Loop
```

dove **A** è il segnale d'ingresso (supposto dato dall'ingresso a pulsante RC0)e **Ar** l'immagine che permette il rilevamento del fronte d'onda positivo. All'arrivo di questo si attiva un timer (lo [0]) impostandolo al tempo (in decimi di secondo) di ritardo voluto (qui 3s), e di conseguenza si mantiene attivato per questo tempo l'uscita (LED RB4)

T-FlipFlop

Benchè non altrertanto fondamentale come i precedenti, vale però la pena di considerare anche il dispositivo la cui uscita cambia ad ogni impulso ricevuto all'ingresso: in elettronica è noto come T-FlipFlop, ed è l'equivalente del relè passo-passo, utile per quelle applicazioni che richiedono ad es. avviamento ed arresto comandatii da un solo pulsante.
Il programma corrispondente è:

```
// Dichiarazione variabili
      boolean CK,Cp,Q;
// Inizializzazioni
      Q=false;
// Loop
      // Lettura ingresso (c0 Clock)
      scu.send("get c0\r");
      risp=scu.getAns();
      if (risp.contains("1")) CK=true;
         else {CK=false; Cp=false;}
      if (CK & !Cp) { Q=!Q; Cp=true;}
      if (Q) scu.sendCmd("set b4\r");
         else  scu.sendCmd("clr b4\r");
 // Fine Loop
```

dove si sono mantenute le sigle convenzionali di **CK** (per l'ingresso o clock) e di **Q** per l'uscita (**Cp** è una variabile ausiliaria che permette, come già visto nel paragrafo precedente, il rilevamento del fonte d'onda positivo).

Il funzionamento può essere verificato inviando impulsi sul pulsante RC0: con un impulso si accende il LED RB4, con il successivo si spegne.

In elettronica il T-FF è alla base del conteggio (binario asincrono), ma con il software è più semplice ed immediato incrementare una variabile (intera) ed avere il conteggio direttamente in decimale. Non considereremo quindi l'utilizzo del T-FF come contatore.

Conclusioni

Si sono esaminati gli elementi fondamentali necessari (e sufficienti) alla realizzazione della maggior parte dei dispositivi di automazione: dovrebbe quindi risultare facile il loro utilizzo in applicazioni concrete (che ovviamente possono andare al di là del semplice uso della scheda didattica).

In prossimi articoli verranno illustate funzioni particolari che ampliano l'applicazione dello SCU a problemi di controllo più complessi.