Gli interrupt (II)

Dopo aver visto il concetto, i tipi di interrupt e i processi che il microprocessore esegue quando questi si producono, affrontiamo ora il problema da un altro punto di vista: quello del programmatore.

Aspetto di un programma tipico con interrupt

I vari "punti di contatto" fra il programma principale e le subroutines si possono vedere sia nell'organigramma che nel codice, grazie alle relazioni che esistono fra loro; il codice della RSI invece sembra qualcosa di diverso, dato che non ci sono modi per entrarci, e l'unico riferimento al carattere d'interrupt si trova all'inizio del programma, quando si posiziona all'indirizzo 4 della memoria delle istruzioni. Nella figura possiamo vedere lo schema tipico di un programma.

Dopo aver specificato il microcontroller che si vuole utilizzare, con la direttiva RADIX viene fissato il sistema di numerazione per default, normalmente esadecimale. La definizione delle diverse variabili e delle etichette si fa sempre con la direttiva EQU. Per posizionare istruzioni su indirizzi specifici della memoria delle istruzioni si utilizza ORG. Il suo parametro è l'indirizzo, in cui va inserita l'istruzione che appare di seguito. Nei PIC ci sono due indirizzi speciali; lo 0, chiamato "vector di reset" che è il punto dove si deve trovare la prima istruzione, e la posizione 4, che è il "vector di interrupt" da dove inizia la RSI. Anche se l'inizio viene fissato in questo indirizzo, il resto della routine può essere da qualsiasi altra parte, perché in questo indirizzo normalmente viene inserita un'istruzione "goto". Normalmente nel

```
Direttive principali

LIST        P=tipo_di_Pic
RADIX       HEX
____________

Etichette

nome        EQU     valore
STATUS      EQU     03
Z           EQU     02
PORTAA      EQU     05
TRISA       EQU     05
AUX         EQU     20

Programma principale

            ORG     0
            goto    INIZIO
            ORG     4
            goto    INTER
            ORG     5

INIZIO      Abilitazione interrupt A
            Abilitazione interrupt B
            Abilitazione globale interrupt
            _____________________
            _____________________
            _____________________

CICLO       _____________________
            Chiamata a ROUTINE 1
            _____________________
            _____________________
            Torna a CICLO

Routine di trattamento dell'interrupt

INTER       _____________________
            _____________________
            retfie

Subroutines

ROUTINE 1   _____________________
            _____________________
            return (retlw)

            END
```

Schema di un programma tipo.

```
Memorizza registro W
Memorizza registro STATUS

Se TMR0IF=1 allora vai alla routine ROUT-TMR0
Se no
                    Se RBIF=1 allora vai alla routine ROUT-RBI
                    Se no
                                        Se _____

ROUT-TMR0                               _____
                                        _____
                                        TMR0IF=0
                                        Vai a FINALE

ROUT-RBI                                _____
                                        _____
                                        RBIF=0
                                        Vai a FINALE

FINALE                                  Ripristina registro W
                                        Ripristina registro STATUS
                                        retfie
```

La RSI in pseudocodice.

programma principale si differenziano due parti: un inizio, dove si stabiliscono le variabili, e un ciclo da dove vengono chiamate le differenti routines.

All'inizio, fra le altre cose, si abilitano i differenti interrupt permessi. Dopo il programma principale si trovano le routines, sia quelle "normali" che quella d'interrupt. Dalle prime si può tornare grazie alle istruzioni "return" o "retlw", che abbiamo già studiato, invece la forma più corretta per ritornare da una routine d'interrupt è con "retfie". L'ordine fra il programma principale, le routines e la routine d'interrupt è indifferente, sempre che al termine di una non si possa entrare nell'altra. Tutto il programma termina con la direttiva END.

L'interrupt visto dall'interno

Abbiamo già visto che quando si verifica un interrupt si scatenano una serie di azioni automatiche. Dopo di che, in mezzo a queste azioni automatiche, viene eseguita anche la nostra routine di servizio all'interrupt (RSI). Visto che non abbiamo il controllo di questa sequenza di azioni automatiche, dobbiamo solo preoccuparci di posizionare la routine d'interrupt nell'indirizzo adeguato della memoria di programma. La cosa giusta, anche se non sempre è necessaria, è salvare l'ambiente dove si è verificato l'interrupt, dato che non se ne conosce il momento e potrebbe accadere a metà di un'elaborazione, come ad esempio il movimento di un dato fra dei registri, ed è necessario che al rientro dalla routine tutto torni uguale a prima.

Sono particolarmente importanti i registri W e STATUS, i quali quasi certamente verranno modificati durante il corso di esecuzione della RSI. Dato che tutte le cause d'interrupt arrivano allo stesso punto di partenza, dobbiamo differenziarle per determinare che cosa fare, caso per caso. Per questo verranno testati uno a uno tutti i flag sino a trovare quello con valore 1, che identificherà la causa di interrupt. L'ordine della sequenza dei test lo decide il programmatore, secondo le cause più probabili all'interno del suo sistema. Dato che non si può produrre un interrupt all'interno di un altro, se il programma è fatto correttamente quando si verifica un interrupt e si entra nella RSI, solo uno dei flag avrà valore 1. Per ogni interrupt si salterà a una subroutine differente, dopo la quale verrà cancellato tramite software il flag corrispondente, per non "confondere" il programma la volta successiva; quindi, viene eseguita una parte "Finale" nella quale si esegue la scrittura dei registri interessati con i valori salvati in precedenza. In ultimo, si rientrerà al programma principale con l'istruzione "retfie". Se al posto di "retfie" si utilizza "return" il rientro si verifica ugualmente, però il bit GIE non viene riportato a 1, quindi, non potranno prodursi ulteriori interrupt. Lo pseudocodice rappresentato nella figura è facilmente adattabile all'assembler, quando ne conosceremo tutte le istruzioni come, ad esempio, le operazioni di "Memorizza" e "Ripristina", che si realizzano con le istruzioni di movimento già studiate.