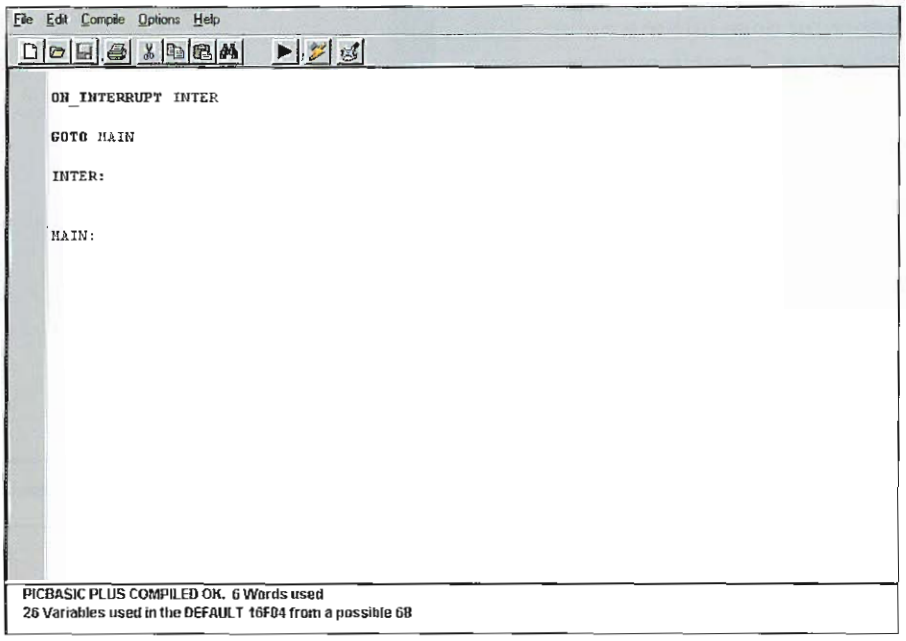
Interrupt

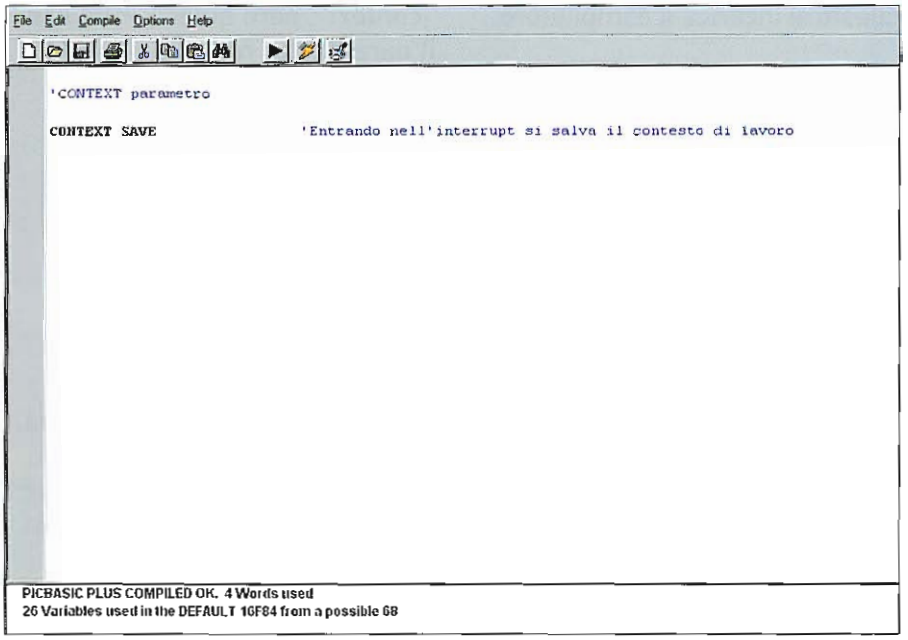
Nella sezione corrispondente abbiamo già spiegato in che cosa consistono gli interrupt nel lavoro con i microcontroller e abbiamo trattato i tipi di interrupt che si possono trovare nei PIC16F87x. Anche se i concetti sono uguali sia che si stia lavorando in assembler che in BASIC, nel PicBasicPlus esistono istruzioni speciali per controllare gli interrupt. Ora le vedremo nel dettaglio.

Salto a interrupt

Quando si verifica un interrupt, il flusso del programma si deve dirigere alla subroutine che è dedicata a questo interrupt. Internamente, questo indirizzamento punta alla posizione 4 della memoria delle istruzioni, quindi in assembler è necessario utilizzare la direttiva ORG per posizionare la routine a questo indirizzo. Se invece della routine di interrupt si trovano istruzioni "normali", il sistema non funzionerà correttamente, ma questo non potrà essere rilevato sino all'esecuzione del programma. Nel PicBasicPlus non dobbiamo preoccuparci di come lavora internamente il microcontroller, però dobbiamo specificare a quale routine dobbiamo andare quando si produce un interrupt. Questo si fa con l'istruzione "on_interrupt". Il parametro di questa istruzione



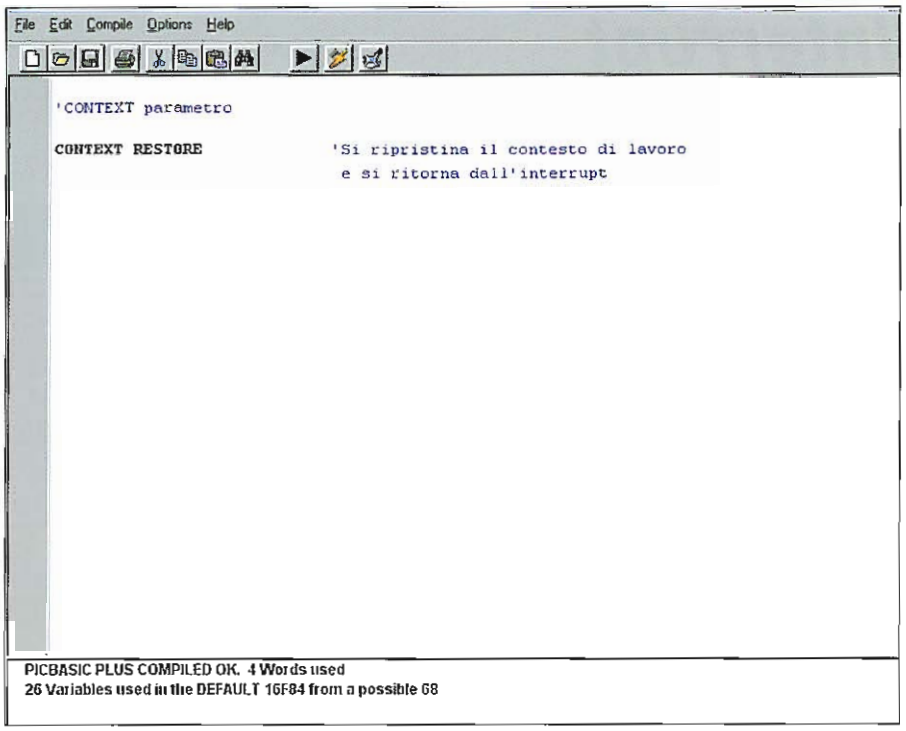
È necessario seguire un ordine tra il programma principale e la routine di servizio di interrupt.



La prima cosa da fare all'interno dell'interrupt sarà salvare il contesto di lavoro.

è un nome che identifica la subroutine. Nella posizione 0 della memoria delle istruzioni deve trovarsi innanzitutto il programma principale, e la routine di trattamento degli interrupt deve iniziare nella posizione 4, perciò pur non avendo bisogno di ricordarci degli indirizzi, anche il PicBasicPlus deve seguire un certo ordine, che è il modo con cui il compilatore differenzia il tipo di subroutine e colloca ognuna di esse nella posizione adeguata.

Nel programma della pagina precedente, quando si produce un interrupt si salterà alla subroutine "inter"; il programma tuttavia inizierà in "main", che è la posizione a cui si salta all'inizio. Possiamo notare che lo stesso accade in assembler con la direttiva ORG, ma esiste una differenza dell'assembler: non è necessario specificare l'indirizzo in cui si mette ogni subroutine, perché di questo si incarica il compilatore.



Anche per tornare dall'interrupt, e ripristinare il contesto di lavoro precedente, si utilizza il comando "context".

L'ambiente di lavoro

Una volta all'interno della subroutine di servizio degli interrupt la prima cosa sarà salvare l'ambiente di lavoro. Nel PicBasiPlus esiste un comando anche per questo chiamato "context". Come parametro dovremo impostare "save", che eseguirà il salvataggio dei registri quali STATUS, FSR e PCLATH, oltre al registro di lavoro W. Il bit di abilitazione globale di interrupt (GIE) verrà impostato automaticamente a 0 (via hardware) per fare in modo che non si verifichi un altro interrupt fino a quando non si esce da quello attuale. Per tornare al programma principale utilizzeremo lo stesso comando "context", però questa volta con il parametro "restore". Questo parametro inoltre riporterà i registri citati in precedenza alla loro condizione iniziale.

Aspetto di un programma tipico con interrupt

Fra il salvataggio e il ripristino del contesto di lavoro si sviluppa la subroutine di servizio all'interrupt. Come in assembler la prima cosa sarà determinare la causa per poter eseguire la subroutine adeguata, e prima di tornare, cancellare il flag che si era attivato. La figura della pagina successiva mostra l'aspetto di un programma tipo con interrupt. Come si può vedere è molto simile all'assembler, tranne il fatto che mancano gli indirizzi di cui abbiamo già parlato e lo stesso comando che ripristina la situazione precedente all'interrupt, serve anche come istruzione di ritorno dall'interrupt. Tuttavia, anche se lo schema è simile in entrambi i linguaggi, le strutture di controllo e le istruzioni di cui dispone il BASIC facilitano enormemente la traduzione di questo schema in un codice eseguibile.

Precauzioni lavorando in BASIC

Anche se non è indispensabile che il codice della routine di interrupt sia in assembler, bisogna fare attenzione alle