# 5. LE ISTRUZIONI BASIC

Nel presente capitolo diamo una spiegazione dettagliata di tutte le istruzioni del linguaggio BASIC corredata di esempi di impiego. Il capitolo è stato scritto con l'intenzione di fornire la possibilità di un rapido riferimento ai punti riguardanti l'esatta codifica delle istruzioni e l'impiego corretto delle medesime nell'ambito di un programma BASIC. Nei paragrafi precedenti tale descrizione sono richiamate le nozioni fondamentali che permettono la lettura delle pagine successive che sono, insieme al capitolo 3, le parti più importanti del manuale.

## Il programma BASIC e le istruzioni BASIC

Un programma BASIC è composto da un insieme di istruzioni l'ultima delle quali è sempre una istruzione END. Le istruzioni BASIC si classificano in:

- istruzioni eseguibili
- istruzioni non eseguibili

Le istruzioni eseguibili specificano al sistema di compiere una azione ben determinata; sono esempi di istruzioni eseguibili: l'istruzione LET che assegna un valore ad una o più variabili, l'istruzione DISP che visualizza il contenuto delle variabili di programma in essa specificate, l'istruzione GOSUB che modifica l'ordine sequenziale di esecuzione di un programma.

Le istruzioni non eseguibili si limitano a specificare delle informazioni che sono utili per l'esecuzione del programma o per il programmatore; sono esempi di istruzioni non eseguibili: l'istruzione DIM che specifica le dimensioni di una variabile multipla, l'istruzione DCL che specifica l'occupazione di memoria principale di una variabile, l'istruzione REM che specifica un commento utile per il programmatore che è stampato quando si richiede il listing del programma.

Le istruzioni non eseguibili possono essere interposte con le istruzioni eseguibili nell'ambito del programma,

ma per facilitarne la leggibilità è bene raggruppare tutte le istruzioni di tipo dichiarativo (DIM, DCL) in un'unica parte del programma.

Ogni istruzione in un programma BASIC è detta linea BASIC e deve iniziare con un numero di linea che è costituito da un numero intero compreso tra 1 e 9999. Il numero di linea determina l'ordine di esecuzione delle istruzioni del programma; ad esso possono riferirsi altre istruzioni di programma (vedi GOTO, GOSUB etc.) oppure dei comandi di sistema (vedi LIST, FETCH etc.). Tutte le istruzioni sono eseguite in ordine di numero di linea, prescindendo dall'ordine con cui sono state introdotte, a meno che la sequenza di esecuzione sia alterata da salti od iterazioni (vedi le istruzioni GOTO, GOSUB, FOR/NEXT etc.).

## Introduzione di linee BASIC

Le linee BASIC sono introdotte da tastiera ed ognuna può avere al massimo 80 caratteri; ogni linea è trasferita in memoria principale premendo il tasto END OF LINE (vedi capitolo 2). Non vi possono essere due linee con lo stesso numero di linea in uno stesso programma. In questo caso l'ultima linea digitata è inserita nel programma, mentre la precedente è cancellata.

Una linea BASIC in generale è composta, oltre che dal numero di linea, da:

- una o più parole chiave BASIC
- uno o più operandi

Le parole chiave BASIC sono parole inglesi con lettere maiuscole (vedi FOR, STEP) o caratteri speciali (vedi due punti ":" per l'istruzione immagine) che specificano al sistema l'azione o le azioni da eseguire. Gli operandi possono specificare:

- su quali elementi di programma (costanti, variabili o espressioni) deve essere compiuta l'azione espressa dalle parole chiave BASIC, ad esempio:
  100 PRINT"OLIVETTI", A,A*B, A$

- quali condizioni (relazione di confronto) si devono verificare perchè l'azione espressa dalla parola chiave BASIC si attui, ad esempio:
  50 IF A=B THEN 1000
  dove A=B indica la condizione che si deve verificare

perchè l'esecuzione del programma prosegua dalla istruzione con numero di linea 1000

- con quali modalità l'azione espressa dalla parola chiave BASIC si deve attuare, ad esempio:
  70 DELAY 100
  dove 100 indica che l'esecuzione dell'istruzione successiva deve essere ritardata di 10 secondi

Una linea BASIC può essere digitata senza introdurre spazi tra le parti (numero di linea, parola chiave BASIC ed operandi) che la compongono. Quando però si esegue una stampa (vedi comando LIST) od una visualizzazione (vedi comando FETCH) delle istruzioni di un programma, le istruzioni sono stampate e visualizzate con un formato diverso da quello con cui sono state introdotte per poter permettere una facile lettura del programma stesso.

Le parole chiave BASIC possono essere digitate da tastiera premendo un solo tasto della sezione alfanumerica mentre si mantiene premuto il tasto [SHIFT]. Questo rende più rapida l'introduzione da tastiera delle istruzioni BASIC e riduce le possibilità di errore. Dopo l'introduzione l'istruzione BASIC è analizzata. Se è rilevato un errore sintattico viene immediatamente visualizzato un messaggio di errore; altrimenti la istruzione è trasferita in memoria principale nell'ambito del programma.

## Notazioni

Le seguenti notazioni sono impiegate nella descrizione delle istruzioni BASIC:

{ } racchiude due o più parametri che non sono opzionali; uno di essi deve essere specificato

[ ] racchiude uno o più parametri che sono opzionali, un parametro o nessun parametro può essere specificato

... indica che il precedente operando può essere ripetuto più di una volta

, separa gli operandi di una istruzione BASIC (nelle istruzioni PRINT, MAT PRINT e DISP ha inoltre una funzione di tabulazione standard)

I seguenti simboli sono usati per definire il formato delle istruzioni, ma non devono essere digitati:

| | |
|---|---|
| - | trattino di unione |
| { } | parentesi graffe |
| [ ] | parentesi quadre |
| ... | puntini |

Le lettere minuscole e le parole con lettere minuscole rappresentano informazioni variabili che deve fornire l'utente. Le parole con lettere maiuscole (parole chiave BASIC) ed i seguenti simboli devono essere digitati esattamente come indicato nella definizione dell'istruzione:

| | |
|---|---|
| # | segno di numero |
| * | asterisco, prodotto scalare o prodotto tra matrici |
| $ | segno di dollaro |
| : | due punti |
| ; | punto e virgola |
| " | apici |
| ( ) | parentesi tonde |
| + | addizione |
| - | sottrazione |

## Elenco e funzione delle istruzioni BASIC

Le istruzioni BASIC e la loro funzione sono elencate in ordine alfabetico, con tutte le istruzioni per il calcolo sulle matrici nell'ultima parte, come segue:

| Istruzione | Funzione |
|---|---|
| APPEND: | Permette di aggiungere dati in coda ad un file dati esterno, sequenziale |
| ASSIGN | Assegna ad una o più variabili uno o più dati contenuti in una stringa di caratteri e separati da un delimitatore |
| BASSIGN | Assegna ad una o più variabili di programma le stringhe e/o i dati numerici compresi nel valore di una espressione stringa |
| BBUILD | Trasferisce il valore di una o più espressioni ad una variabile stringa, modificandone il formato |
| BEEP | Produce una segnalazione acustica |
| BPAD | Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione, aggiungendo in coda dei caratteri binari |

| Istruzione | Funzione |
|---|---|
| BUILD | Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa |
| BUILD USING | Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa ponendo i caratteri in posizioni predefinite da una istruzione immagine |
| CALL | Carica in memoria principale il modulo assembler specificato e ne inizia l'esecuzione |
| CHAIN | Termina l'esecuzione di un programma presente in memoria principale e carica in memoria principale il programma specificato lanciandone l'esecuzione |
| CONVERT | Converte ogni carattere di una espressione stringa nel corrispondente codice numerico ISO e viceversa |
| DATA | Crea un file dati interno al programma |
| DCL | Dichiara la lunghezza di allocazione delle variabili stringa e la singola precisione per le variabili numeriche |
| DEF | Definisce una funzione monolinea |
| DEF/FNEND | Definiscono una funzione multilinea |
| DELAY | Ritarda l'esecuzione dell'istruzione successiva |
| DEPAD | Rimuove in una variabile stringa i caratteri di riempimento specificati |
| DIM | Specifica le dimensioni delle variabili multiple di programma |
| DISP | Visualizza dati e testi sul display in formato standard |
| DISP USING | Visualizza dati e testi sul display in un formato predefinito in una istruzione immagine |
| END | Definisce la fine di un programma |
| FILES | Specifica quali file dati esterni possono essere elaborati dal programma |

| Istruzione | Funzione |
|---|---|
| FILE: | Chiude ed apre l'accesso di un programma ad un file dati esterno |
| FKEY # | Assegna un contenuto ai tasti funzione |
| FNEND | Termina la definizione di una funzione multilinea |
| FOR | Inizia l'esecuzione di un ciclo iterativo |
| GOSUB | Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma |
| GOTO | Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata |
| IF...THEN | Trasferisce il controllo dell'esecuzione di un programma ad una istruzione specificata, nel caso che si verifichi la condizione predefinita |
| Istruzione Immagine | Specifica un formato predefinito utilizzato dalle istruzioni PRINT USING, DISP USING, BUILD USING, MAT PRINT USING |
| INPUT | Assegna i valori introdotti da tastiera alle variabili di programma specificate |
| INTERRUPT ENABLE | Consente la gestione da parte dell'utente di interruzioni sia interne che da periferiche esterne |
| LET | Assegna valori alle variabili di programma |
| NEXT | Definisce il termine di un ciclo iterativo |
| ON...GOSUB | Trasferisce il controllo dell'esecuzione di un programma ad un sottoprogramma scelto tra un insieme di sottoprogrammi in funzione del valore assunto da una espressione specificata |
| ON...GOTO | Trasferisce il controllo dell'esecuzione di un programma ad una istruzione scelta tra un insieme di istruzioni in funzione del valore assunto da una espressione specificata |
| PAD | Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione aggiungendo in coda dei caratteri predefiniti |
| PRINT | Stampa dati e testi in un formato standard |

| Istruzione | Funzione |
|---|---|
| PRINT USING | Stampa dati e testi in un formato predefinito in una istruzione immagine |
| RANDOMIZE | Permette la generazione di numeri casuali |
| READ | Assegna alle variabili di programma specificate i valori contenuti nel file dati interno al programma, generato con le istruzioni DATA |
| READ: | Assegna alle variabili di programma specificate i valori contenuti in un file dati esterno |
| REMARK | Permette di inserire in un programma dei commenti che rendono facile la lettura del relativo listing |
| RESTORE | Posiziona il pointer del file dati interno all'inizio del file stesso |
| RESTORE: | Posiziona il pointer di un file dati esterno all'inizio del file e, se il file è di tipo sequenziale, ne permette la lettura |
| RETURN | Trasferisce il controllo della esecuzione di un programma all'istruzione successiva ad una istruzione GOSUB |
| RKB | Assegna ad una variabile stringa i caratteri introdotti da tastiera che sono scelti tra i caratteri del SET P6066 |
| SCRATCH: | Posiziona il pointer di un file dati esterno all'inizio di un file sequenziale e permette di registrare in esso dei dati |
| SETW: | Posiziona il pointer all'inizio della parola specificata di un file dati esterno ad accesso diretto |
| STOP | Interrompe l'esecuzione di un programma e commuta il sistema nello stato di debugging |
| TRACE OFF | Termina la stampa dei numeri di linea delle istruzioni di programma eseguite |
| TRACE ON | Richiede la stampa dei numeri di linea di ogni successiva istruzione di programma eseguita |

| Istruzione | Funzione |
|---|---|
| WHERE: | Determina la posizione su cui è posizionato il pointer di un file dati esterno nell'ambito del file |
| WRITE: | Registra in un file dati esterno i valori delle espressioni specificate |
| MAT...= | Assegna i valori degli elementi di una matrice agli elementi di un'altra matrice |
| MAT...+ | Esegue l'operazione di addizione tra due matrici e ne assegna il risultato alla matrice specificata prima del segno uguale |
| MAT...- | Esegue l'operazione di sottrazione tra due matrici e ne assegna il risultato ad una matrice specificata |
| MAT...* (scalare) | Moltiplica ogni elemento di una matrice per il valore di una espressione numerica e ne assegna il risultato ad un'altra matrice specificata |
| MAT...* | Esegue il prodotto, righe per colonne, tra due matrici e ne assegna il risultato ad una matrice specificata |
| MAT...CON | Assegna il valore uno ad ogni elemento di una matrice |
| MAT...IDN | Assegna il valore uno a tutti gli elementi della diagonale principale di una matrice quadrata ed il valore zero a tutti gli altri elementi della matrice |
| MAT...INV | Calcola la matrice inversa di una matrice quadrata e la assegna ad una matrice specificata |
| MAT...TRN | Assegna ad una matrice specificata gli elementi di un'altra matrice scambiandone le righe con le colonne |
| MAT...ZER | Assegna il valore zero a tutti gli elementi di una matrice |
| MAT INPUT | Assegna agli elementi di una variabile multipla i dati introdotti da tastiera |
| MAT PRINT | Stampa i valori degli elementi di una o più variabili multiple nel formato standard |
| MAT PRINT USING | Stampa i valori degli elementi di una o più variabili multiple in un formato predefinito in una istruzione immagine |

| Istruzione | Funzione |
|---|---|
| MAT READ | Assegna agli elementi di una o più variabili multiple i dati contenuti nel file interno definito mediante le istruzioni DATA |
| MAT READ: | Assegna agli elementi di una o più variabili multiple i dati contenuti in un file dati esterno |
| MAT WRITE: | Registra in un file dati esterno i valori degli elementi di una o più variabili multiple specificate. |

## Descrizione delle istruzioni BASIC

Nel seguito sono descritte dettagliatamente tutte le istruzioni del linguaggio BASIC. Tutte le istruzioni sono disposte in ordine alfabetico, meno le istruzioni che permettono di elaborare le matrici che sono raggruppate nell'ultima parte del paragrafo. Alcune istruzioni sono descritte insieme, poichè sono sempre utilizzate in combinazione, comunque vengono riportati gli eventuali riferimenti. La descrizione di ogni istruzione è realizzata secondo la medesima struttura che è composta dai seguenti punti fondamentali:

- funzione : è una breve descrizione della funzione dell'istruzione
- formato : è una descrizione sintetica del formato più generale dell'istruzione
- azione : è una descrizione dettagliata delle azioni eseguite dal sistema quando l'istruzione è eseguita
- note : è un insieme di osservazioni e di regole da osservare per un corretto impiego dell'istruzione
- esempi : è una raccolta di esempi d'impiego della istruzione descritta che permette di chiarire eventuali dubbi riguardo la codifica corretta dell'istruzione. Per motivi di spazio gli esempi sono brevi e non coprono applicazioni significative.

Istruzione APPEND:

Funzione

Permette di aggiungere dati in coda ad un file dati esterno di tipo sequenziale

Formato

**APPEND: file-designator**

dove:
file-designator
è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica un designatore di file.

Azione

Il puntatore del file specificato con file-designator è posizionato all'inizio della parola successiva all'ultimo dato contenuto nel file dati esterno.

Il file suddetto è aperto al programma per operazioni di registrazione.

Note

1. Il valore di file-designator, arrotondato all'intero più prossimo, deve essere maggiore di zero e minore od uguale al numero di file specificati nella istruzione FILES.

2. Il designatore di file è un numero che indica su quale file dati esterno deve essere fatta l'operazione specificata (vedi istruzioni FILES e FILE:).

3. Il designatore di file dell'istruzione APPEND: deve indicare un file dati di tipo sequenziale.

Esempi

1. Si crei un file sequenziale di 256 byte nella libreria MAT sul floppy disk nell'unità FDU1. Si esegua il programma indicato nel listing sottostante. Il programma registra sul file APPEND i numeri

interi da 1 a 10 e quindi li legge e stampa con la tabulazione standard. Con l'istruzione 130 il pointer si pone in coda al file per cui la stringa successiva viene registrata, istruzione 140, subito dopo il numero 10, come si vede dalla relativa stampa del contenuto del file che è fatta da programma.

```
CRE APPEND,(MAT,FDU1),,256

LIST
FILE     ZA


0010 DCL 80(I$)
0020 FILES APPEND
0030 SCRATCH :1
0040 FOR I=1 TO 10 STEP 1
0050 WRITE :1,I
0060 NEXT I
0070 RESTORE :1
0080 FOR I=1 TO 10 STEP 1
0090 READ :1,I
0100 PRINT I,
0110 NEXT I
0120 PRINT
0130 APPEND :1
0140 WRITE :1,"COME SI VEDE QUESTA STRINGA E' IN CODA AL FILE"
0150 RESTORE :1
0160 FOR I=1 TO 10 STEP 1
0170 READ :1,I
0180 PRINT I,
0190 NEXT I
0200 READ :1,I$
0210 PRINT I$;
0220 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
 1             2             3             4             5
 6             7             8             9             10

 1             2             3             4             5
 6             7             8             9             10
COME SI VEDE QUESTA STRINGA E' IN CODA AL FILE
```

Istruzione ASSIGN

Funzione

Assegna ad una o più variabili uno o più dati contenuti in una stringa di caratteri e separati da un delimitatore.

Formato

**ASSIGN string-exp,** { **num-var** / **string-var** } [ , { **num-var** / **string-var** } ] ... ; **delimiter**

dove:

string-exp
: è una espressione stringa il cui valore può essere composto da uno o più dati separati da un delimitatore

num-var
: è una variabile numerica alla quale viene assegnato un valore numerico contenuto in string-exp

string-var
: è una variabile stringa alla quale è assegnato un valore contenuto in string-exp

delimiter
: è un numero intero compreso tra zero e 255 che indica quale carattere della tabella ISO (vedi appendice E) separa i diversi dati contenuti nel valore di string-exp.

Azione

L'espressione stringa viene calcolata ed il valore ottenuto viene considerato composto da diversi dati ognuno dei quali è separato dal successivo dal carattere che corrisponde, nella tabella ISO, al numero indicato con delimiter.

Ogni dato viene trasformato nel formato elaborabile dal sistema e quindi assegnato nell'ordine, da sinistra a destra, alle variabili specificate dopo string-exp.

Note

1. Un dato stringa non può essere assegnato ad una variabile numerica.

2. Il numero di dati che compongono il valore di string-exp deve essere maggiore od uguale al numero di variabili specificate dopo string-exp.

3. Se il valore della espressione stringa string-exp contiene due caratteri consecutivi equivalenti al delimitatore delimiter, il dato assegnato alla variabile corrispondente è la stringa nulla.

4. L'istruzione ASSIGN, insieme all'istruzione BUILD, è utile quando si vogliano trasferire dati tra la memoria principale del P6066 ed una unità periferica esterna (vedi manuale P6066 I/O con periferiche esterne).

Esempi

1. Nel listing sottostante si può vedere un programma che assegna a diverse variabili di programma i dati contenuti in una stringa separati dal carattere spazio (a cui corrisponde il numero 32, nella tabella ISO).

```
LIST
FILE

0010 ASSIGN "LUNGHEZZA 155 METRI ALTEZZA 120 CENTIMETRI",L$,L,M$,A$,A,C$;32
0020 PRINT "L$=";L$,"L=";L,"M$=";M$
0030 PRINT "A$=";A$,"A=";A,"C$=";C$
0040 END

END OF LISTING


RUN
L$=LUNGHEZZA     L= 155          M$=METRI
A$=ALTEZZA       A= 120          C$=CENTIMETRI
```

2. Nel programma precedente si è diminuito il numero di variabili a cui assegnare gli elementi della stringa. Come si vede dalla stampa prodotta alle variabili A e C$ è assegnato rispettivamente il valore zero e stringa nulla; viene segnalato un errore di tipo recuperabile ed il sistema è nello stato di debugging in attesa di una decisione da parte dell'utente.

```
LIST
FILE


0010 ASSIGN "LUNGHEZZA 155 METRI ALTEZZA 120 CENTIMETRI",L$,L,M$,A$;32
0020 PRINT "L$=";L$,"L=";L,"M$=";M$
0030 PRINT "A$=";A$,"A=";A,"C$=";C$
0040 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
L$=LUNGHEZZA     L= 155           M$=METRI
A$=ALTEZZA       A= 0             C$=
ERROR 1    IN LINE 30
```

# BASSIGN

Istruzione BASSIGN

Funzione

Assegna a una o più variabili di programma le stringhe e/o i dati numerici contenuti nel valore di una espressione stringa e rappresentati nel formato con cui sono registrati in un file dati esterno.

Formato

**BASSIGN** string-exp, num-var | string-var [, num-var | string-var] ...

dove

string-exp
: è una espressione stringa composta di una o più variabili stringa i cui valori sono rappresentati nel formato con cui sono rappresentati in un file dati esterno

num-var
: è una variabile numerica alla quale viene assegnato il corrispondente dato numerico contenuto nel valore di string-exp

string-var
: è una variabile stringa alla quale viene assegnato il corrispondente dato stringa contenuto nel valore di string-exp

Azione

L'espressione string-exp viene valutata e riconosciuta come composta da uno o più dati ognuno dei quali è rappresentato nel formato con cui i dati sono registrati in un file dati esterno. Ogni dato componente del valore di string-exp viene modificato nel formato elaborabile dal sistema ed assegnato, nell'ordine, da sinistra a destra, alle variabili specificate dopo string-exp.

Note

1. Un dato di tipo stringa non può essere assegnato ad una variabile numerica; un dato di tipo numerico non può essere assegnato ad una variabile stringa.

2. Il numero di variabili specificate deve essere minore od uguale al numero di dati componenti il valore di string-exp.

3. In string-exp non possono comparire delle stringhe di caratteri ISO, ad esempio: "AREA", bensì le variabili che compaiono nella espressione devono avere un contenuto assegnato ad esse da una istruzione BBUILD (vedi istruzione BBUILD).

4. L'istruzione BASSIGN, insieme all'istruzione BBUILD, è utile quando si vogliano trasferire dati tra la memoria principale del P6066 ed una unità periferica esterna senza modificare il formato dei dati in stringhe ISO.

Esempio

1. Nel listing seguente è indicato un programma che mostra come il contenuto delle variabili A$ e B$ viene separato ed assegnato, solo nella parte valore, alle variabili specificate nella istruzione 40.

```
LIST
FILE


0010 DCL 30(A$,B$)
0020 BBUILD A$,"AREA","123",249
0030 BBUILD B$,"VOLUME","10",100
0040 BASSIGN A$+B$,S$,N$,N,V$,M$,M
0050 PRINT "A$=";A$,"B$=";B$
0060 PRINT "S$=";S$,"N$=";N$,"N=";N,"V$=";V$
0070 PRINT "M$=";M$,"M=";M
0080 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A$=■■AREA■■123 ■■■■■■■■     B$=■■VOLUME  ■■10  ■■■■■■■■
S$=AREA          N$=123          N= 249          V$=VOLUME
M$=10            M= 100
```

Istruzione BBUILD

Funzione

Trasferisce il valore di una o più espressioni ad una variabile stringa, modificandone il formato.

Formato

**BBUILD string-var,** { **num-exp** / **string-exp** } [ , { **num-exp** / **string-exp** } ] ...

dove:

string-var
è una variabile stringa in cui vengono trasferiti i valori delle espressioni specificate

num-exp
è una espressione numerica il cui valore è trasferito nella variabile stringa specificata

string-exp
è una espressione stringa il cui valore è trasferito nella variabile stringa specificata

Azione

Le espressioni specificate sono valutate ed i relativi valori sono assegnati nell'ordine, da sinistra a destra, alla variabile stringa specificata dopo la parola chiave.

Note

1. Ogni valore è assegnato a string-var nel formato con cui è registrato in un file dati esterno per cui l'occupazione sara:

   - 4 byte per i valori numerici in singola precisione
   - 8 byte per i valori numerici in doppia precisione
   - INT((n-1)/4+2)*4 byte per i valori di tipo stringa (dove n è il numero di caratteri che compongono la stringa).

2. L'istruzione BBUILD è utile per trasferire dati dalla memoria principale del P6066 ad una periferica esterna. In questo caso, infatti, i dati numerici devono essere assegnati ad una variabile stringa(con una istruzione del tipo: 50 BBUILD A$, A,B,C che trasferisce i valori di A,B e C in A$) prima di essere trasferiti alla periferica esterna. I dati numerici registrati su un supporto di una periferica esterna sono trasferiti in memoria principale ed assegnati ad una variabile stringa. Dalla variabile stringa possono poi essere assegnati ad una variabile numerica eseguendo una istruzione BASSIGN del tipo: 100 BASSIGN A$,A,B,C con cui tre dati registrati su un supporto e ricevuti in A$ sono assegnati nell'ordine alle variabili A,B e C.

   Si noti che per eseguire il trasferimento di dati tra P6066 e periferica esterna si possono utilizzare anche le istruzioni BUILD ed ASSIGN. Le istruzioni BBUILD e BASSIGN sono eseguite più rapidamente dal sistema perchè l'esecuzione della istruzione BBUILD non modifica il formato dei dati in stringhe ISO.

Esempi

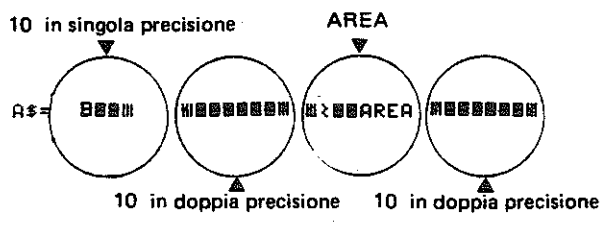
1. Nella stampa prodotta dal programma sottostante si può vedere in quale formato sono rappresentati i dati trasferiti in A$ dall'istruzione BBUILD.

```
FILE      QW

0010 DCL SA,28A$
0020 LET A=10
0030 LET B=10
0040 BBUILD A$,A,B,"AREA",10
0050 PRINT "A=";A,"B=";B
0060 FOR I=1 TO 10 STEP 1
0070 PRINT
0080 NEXT I
0090 PRINT "A$=";A$
0100 END

END OF LISTING


A=  10              B=  10
```



2. Nella stampa prodotta dal programma sottostante si può vedere come i dati stringa vengono ad occupare un numero intero di parole (4 caratteri ognuna) per cui le parole sono riempite con spazi in coda (vedi i due spazi dopo VOLUME).

```
LIST
FILE


0010 DCL 20(A$)
0020 BBUILD A$,"VOLUME",10
0030 PRINT "A$=";A$
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A$=■/■■VOLUME  ■■■■■■■■
```

Gli ultimi 8 caratteri stampati dopo A$= rappresentano il numero 10 in doppia precisione.

Istruzione BEEP

Funzione — Produce una segnalazione acustica.

Formato — **BEEP**

Azione — Il segnalatore acustico emette un suono della durata di 0,2 secondi.

Nota — L'istruzione è particolarmente utile per richiamare l'attenzione dell'operatore su un messaggio visualizzato sul display.

Esempio — La seguente routine richiama l'attenzione dell'operatore sul messaggio: I non supera 98. L'istruzione DELAY impone un ritardo nell'esecuzione dell'istruzione successiva per cui il suono permane e ad ogni esecuzione del ciclo FOR/NEXT riappare il messaggio suddetto.

```
LIST
FILE


0010 FOR I=1 TO 100 STEP 1
0020 BEEP
0030 DISP "I non supera 98"
0040 DELAY 2
0050 DISP
0060 NEXT I
0070 DISP "I e' maggiore di 98"
0080 DELAY 100
0090 END

END OF LISTING
```

Istruzione BPAD

Funzione

Eguaglia la lunghezza attuale di una variabile stringa alla sua lunghezza di allocazione, aggiungendo in coda dei caratteri binari.

Formato

**BPAD string-var**

dove:
string-var
è una variabile stringa di cui si vuole rendere la lunghezza attuale uguale alla lunghezza di allocazione.

Azione

Se la variabile string-var ha un valore il cui numero di caratteri è inferiore alla sua lunghezza di allocazione, in coda ad essa sono aggiunti altrettanti caratteri di riempimento che nella tabella ISO corrispondono al numero decimale 255.

Note

1. L'istruzione BPAD permette la generazione di record di dati aventi la stessa lunghezza, quando si genera un file dati esterno oppure un file su di un supporto di una periferica esterna.

2. Se la variabile specificata nella istruzione ha lunghezza attuale uguale alla sua lunghezza di allocazione, l'istruzione è ignorata.

Esempi

1. Dopo aver creato il file dati esterno ad accesso diretto, di nome BPAD, utilizzando l'istruzione 60 si generano dei record di 60 caratteri.

   In risposta alla istruzione 40 si forniscono le

parole corrispondenti ai numeri ordinali da 1 a 32. Si genera così un file che indica quali sono i caratteri della tabella ISO, dal primo al trentaduesimo.

```
CRE BPAD,(MAT,FDU1),R

LIST
FILE     ISOT

0010 DCL 20(A$),60(B$)
0020 FILES BPAD
0030 FOR I=0 TO 31 STEP 1
0040 INPUT A$
0050 LET B$="I1 "+A$+" carattere della Tabella ISO e' "+CHR$(I)
0060 BPAD B$
0070 WRITE :1,B$
0080 NEXT I
0090 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
?
primo
?
secondo
?
terzo
?
quarto
?
```

2. Con il programma sottostante si leggano i record desiderati del file dati generato con l'esempio precedente. Avendo generato dei record tutti della medesima lunghezza la ricerca del dato desiderato risulta facile: ogni dato occupa 16 parole per cui il pointer viene posizionato sulle parole 16*(N-1) +1 dove N è il numero d'ordine del dato desiderato. Prima di stampare il dato si eliminano i caratteri di riempimento con l'istruzione 65 (vedi istruzione DEPAD).

```
LIST
FILE     READIS


0010 DCL 60(A$)
0020 FILES BPAD
0030 DISP "QUALE ELEMENTO ISO VUOI";
0040 INPUT N
0050 SETW :1 TO 16*(N-1)+1
0060 READ :1,A$
0065 DEPAD A$,255
0070 PRINT A$
0080 PRINT
0090 GOTO 30
0100 END

END OF LISTING
RUN
QUALE ELEMENTO ISO VUOI?
1
Il primo carattere della Tabella ISO e' ■


QUALE ELEMENTO ISO VUOI?
2
Il secondo carattere della Tabella ISO e' ┌

QUALE ELEMENTO ISO VUOI?
3
Il terzo carattere della Tabella ISO e' ┴

QUALE ELEMENTO ISO VUOI?
6
Il sesto carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
30
Il trentesimo carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
32
Il trentaduesimo carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
```

Istruzione BUILD

Funzione

Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa separate con un delimitatore.

Formato

**BUILD string-var, [num-exp | string-exp] [, [num-exp | string-exp]] ... [; delimiter]**

dove:

string-var
è una variabile stringa in cui vengono trasferite le stringhe ricavate dai valori delle espressioni specificate

num-exp
è una espressione numerica il cui valore viene trascodificato in una stringa di caratteri ISO e trasferito nella variabile stringa specificata

string-exp
è una espressione stringa il cui valore viene trasferito nella variabile stringa specificata

delimiter
è un numero intero compreso tra zero e 255 il cui carattere che gli corrisponde nella tabella ISO viene interposto, come delimitatore, tra le stringhe ISO che sono trasferite nella variabile stringa specificata.

Azione

Le espressioni sono valutate ed i valori ottenuti sono trascodificati in stringhe di caratteri e quindi trasferiti nell'ordine, da sinistra a destra, nella variabile stringa specificata.

Se è specificato l'operando delimiter, il carattere che gli corrisponde nella tabella ISO viene interposto tra le stringhe suddette nell'ambito della variabile stringa specificata.

Note

1. Il valore corrispondente alla valutazione di una espressione stringa viene trascodificato nella sequenza di caratteri che lo costituisce.

2. Il valore corrispondente alla valutazione di una espressione numerica viene trascodificato in una sequenza di caratteri così composta:

   - il primo carattere è uno spazio oppure il segno meno (-) se il valore suddetto è rispettivamente positivo o negativo

   - i successivi caratteri sono delle cifre decimali che corrispondono al valore suddetto ed assumono la forma di un:

     . numero intero se il valore è un numero intero rappresentabile correttamente con non più di 8 cifre

     . numero decimale in virgola fissa, con 8 cifre significative, se il valore è un numero in valore assoluto compreso tra 0.0999999995 e 99999999.4 (se il numero è minore di 1 viene tralasciato lo zero che precede la parte decimale) o anche se il valore è un numero in valore assoluto minore di 0.0999999995 ma rappresentabile con non più di 8 cifre dopo il punto decimale

     . numero decimale in virgola mobile, con 8 cifre significative, in tutti i rimanenti casi

   - l'ultimo carattere è uno spazio

3. L'istruzione BUILD è utile per trasferire dati dalla memoria principale del P6066 ad una periferica esterna. In questo caso, infatti, i dati numerici devono essere assegnati ad una variabile stringa con una istruzione del tipo: 50 BUILD A$,A,B,C,44 che trasferisce i valori di A,B e C in A$ separati con il carattere virgola. I dati numerici registrati su un supporto di una periferica esterna sono trasferiti in memoria principale ed assegnati ad una variabile stringa. Dalla variabile stringa possono poi essere assegnati ad una variabile numerica eseguendo una istruzione ASSIGN del tipo:
   100 ASSIGN A$ A,B,C,44

con cui tre dati registrati su un supporto esterno e ricevuti in A$ sono assegnati nell'ordine alle variabili A,B e C. Per informazioni sulle istruzioni BASIC che attuano il trasferimento di dati tra P6066 e periferiche esterne si veda il manuale P6066 I/O con periferiche esterne.

Esempi

1. Il seguente programma utilizza l'istruzione BUILD per trasferire nelle variabili D$ e C$ le stringhe di caratteri stampate qui sotto. Si noti che il valore 0.0999999995 è generato nel formato in virgola fissa ma arrotondato a 0.1 perchè le cifre significative sono più di 8. Si noti come il valore -0.0099999999 è generato nel formato in virgola mobile perchè al di fuori del campo definito nella relativa nota qui sopra.

```
FILE     RBUIL1

0010 DCL 30(D$),18(C$)
0020 BUILD D$,0.0999999995,-0.0099999999;44
0030 PRINT "D$=";D$
0040 LET A=100
0050 LET B=10
0060 LET A$="AREA"
0070 LET B$=" DI BASE"
0080 BUILD C$,A$+B$,A*B;44
0090 PRINT "A$=";A$,"B$=";B$,"A$+B$=";A$+B$
0100 PRINT "A*B=";A*B
0110 PRINT "C$=";C$
0120 END .

END OF LISTING


D$= .10000000 ,-9.9999999E-03
A$=AREA          B$= DI BASE      A$+B$=AREA DI BASE
A*B= 1000
C$=AREA DI BASE, 1000
```

2. In questo esempio non si è specificato alcun separatore e si può notare come in questo caso le stringhe di caratteri assegnate ad A$ si susseguono consecutivamente.

```
LIST
FILE


0010 DCL 25(A$)
0020 BUILD A$,"AR","EA DI BA","SE DE","L CILI","NDRO"
0030 PRINT "A$=";A$
0040 END

END OF LISTING



RUN
**** FORMALLY CORRECT PROGRAM ****
A$=AREA DI BASE DEL CILINDRO
```

Istruzione BUILD USING

Funzione

Trascodifica i valori di una o più espressioni in altrettante stringhe ISO e le trasferisce in una variabile stringa ponendo i caratteri in posizioni predefinite da una istruzione immagine (si veda Istruzione IMMAGINE).

Formato

**BUILD USING** {**line-num** / **string-var$_1$**}, **string-var$_2$**, {**num-exp** / **string-exp**} [, {**num-exp** / **string-exp**}]...

dove:

line-num
: è un numero intero che indica il numero di linea di una istruzione IMMAGINE

string-var$_1$
: è una variabile stringa il cui contenuto specifica come devono disporsi i caratteri nella variabile stringa

string-var$_2$
: è una variabile stringa in cui devono essere trasferiti i valori trascodificati delle espressioni specificate

num-exp
: indica una espressione numerica il cui valore deve essere trascodificato in una stringa di caratteri numerici ISO e trasferito, nell'ordine indicato, nella variabile stringa specificata con string-var

string-exp
: indica una espressione stringa il cui valore deve essere trasferito, nell'ordine indicato, nella variabile stringa specificata con string-var .

Azione

Le espressioni specificate sono valutate ed i valori ottenuti sono trascodificati in sequenze di caratteri ISO, quindi queste ultime sono trasferite, nell'ordine da sinistra a destra, nella variabile stringa indicata con string-var$_2$. I caratteri, nella variabile stringa

specificata con string-var$_2$, sono disposti come è specificato dai campi della istruzione immagine il cui numero di linea è line-num o del contenuto della variabile specificata con string-var$_1$.

Nota

Si veda l'istruzione IMMAGINE per un completa descrizione di come i caratteri sono disposti in string-var$_2$.

Esempi

1. Il programma sottostante pone nella variabile stringa B$ i valori numerici trascodificati in sequenze di caratteri numerici ISO, posizionati come indicato nell'istruzione 10. Nell'istruzione 50 l'immagine con cui i valori delle rispettive espressioni devono essere posti in C$ è indicata dal contenuto della variabile A$.

```
LIST
FILE


0005 PRINT
0010 : ###   $$$$   ##.###   $$$.###   ###.#↑↑↑↑
0020 DCL 70(A$,B$,C$)
0030 LET A$="'LLLLLLLLLL   LLL   'RRRRRRRRRR   RRR   'CCCCCCCCCC   CCC"
0040 BUILD USING 10,B$,12.7,15.8,-5.75,-5.8,-75.5E-7
0050 BUILD USING A$,C$,"Olivetti","Olivetti","Olivetti"
0060 PRINT "B$=";B$
0065 PRINT
0070 PRINT "C$=";C$
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

B$=  12    $15   -5.750   $-5.800   -75.5E-07

C$=Olivetti      LLL      Olivetti   RRR    Olivetti     CCC
```

2. Nel programma sottostante si mostra cosa accade quando i valori delle espressioni nell'istruzione BUILD USING hanno più caratteri di quelli dei relativi campi nell'immagine.

```
LIST
FILE


0005 PRINT
0010 : ###    $$$$    ##.###    $$$.###    ###.#↑↑↑↑
0020 DCL 70(A$,B$,C$)
0030 LET A$="'LLLLLLLLLL   LLL   'RRRRRRRRRR   RRR   'CCCCCCCCCC   CCC"
0040 BUILD USING 10,B$,12.7,15.8,-5.75,-5.8,-75.5E-7
0050 BUILD USING A$,C$,"Olivetti P6066","Olivetti P6066","Olivetti P6066"
0060 PRINT "B$=";B$
0065 PRINT
0070 PRINT "C$=";C$
0080 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

B$=  12    $15    -5.750    $-5.800    -75.5E-07

C$=Olivetti      LLL      Olivetti   RRR    Olivetti     CCC
```

Istruzione CALL

Funzione

Carica in memoria principale un modulo assembler e ne inizia l'esecuzione.

Formato

**CALL string-exp$_1$[,{num-exp / string-exp}]...[;DEB]**

dove:

string-exp$_1$
: è un'espressione stringa il cui valore indica il nome di un modulo assembler presente in una delle librerie aperte

num-exp
: è un'espressione numerica il cui valore è passato come argomento al modulo assembler richiamato

string-exp
: è un'espressione stringa il cui valore è passato come argomento al modulo assembler richiamato

DEB
: è la stringa omonima e specifica al sistema di caricare il modulo di sistema che permette di eseguire il debugging del modulo assembler caricato in memoria principale

Azione

Il modulo assembler specificato con il valore di string-exp$_1$ è caricato in memoria principale ed eseguito.

Note

1. Il numero di argomenti che si possono passare al modulo assembler specificato è limitato solamente dal numero di caratteri che possono costituire una istruzione BASIC (80 caratteri).

2. Al termine della esecuzione del modulo assembler suddetto viene eseguita l'istruzione esecutiva suc-

cessiva alla CALL.

3. Per ulteriori informazioni sull'impiego dell'istruzione si veda la pubblicazione:"P6066-Ambiente di programmazione in Assembler",codice 3978380 T.

# CHAIN

Istruzione CHAIN

Funzione

Termina l'esecuzione di un programma presente in memoria principale e carica in memoria principale il programma specificato, lanciandone l'esecuzione.

Formato

**CHAIN** |«filename»|
|string-var|

dove:

filename
: è il nome di un file programma presente in una delle librerie

string-var
: è una variabile stringa il cui contenuto è il nome di un file programma presente in una delle librerie

Azione

L'esecuzione del programma residente in memoria principale è terminata, come se fosse eseguita una istruzione END (vedi END). Viene caricato in memoria principale, ed eseguito, il programma con il nome indicato tra apici nell'istruzione o contenuto nella variabile stringa specificata come operando dell'istruzione.

Il programma chiamante è cancellato dalla memoria principale; quindi deve essere registrato in una libreria (comando SAVE) prima che l'istruzione CHAIN sia eseguita.

Note

1. Il programma da caricare in memoria principale è ricercato tra le librerie aperte secondo l'ordine di apertura.

2. La comunicazione di dati tra programmi concatenati può avvenire utilizzando l'istruzione COMMON, vedi COMMON.

3. L'istruzione CHAIN chiude l'accesso da parte del programma utente chiamato ai file dati esterni aperti con il programma chiamante, ma mantiene in memoria principale alcune informazioni relative ai file che prima della esecuzione della istruzione suddetta avevano un numero designatore di file compreso tra 1 e 4. Questo permette al sistema di passare più rapidamente alla esecuzione del programma chiamato se i file dati esterni da esso utilizzati sono quelli suddetti. Il programma chiamato deve specificare i nomi dei file suddetti nella istruzione FILES.

   Quindi per passare più rapidamente l'esecuzione ad un programma chiamato dalla istruzione CHAIN, se abbiamo un programma chiamante come il seguente:

```
 10 FILES A,B,C,D,E,F
    -
    -
    -
100 FILE: 3,"H"
    -
    -
    -
200 FILE: 1,"Z"
    -
    -
    -
300 CHAIN "MAT"
    -
    -
    -
9999 END
```

   il programma chiamato non avrà la seguente struttura:

```
 10 FILES A,B,C,D
    -
    -
    -
100 FILE: 3,"H"
    -
    -
    -
9999 END
```

   ma bensì la struttura seguente:

```
 10 FILES Z,B,H,D
    -
    -
    -
9999 END
```

Esempio

1. IL programma sottostante, dopo aver registrato su file esterno i primi undici caratteri della tabella ISO, richiama un programma che legge il file dati appena prodotto.

```
LIST
FILE      ISOT

0010 DCL 20(A$),60(B$)
0020 FILES BPAD
0030 FOR I=0 TO 10 STEP 1
0040 INPUT A$
0050 LET B$="Il "+A$+" carattere della Tabella ISO e' "+CHR$(I)
0060 BPAD B$
0070 WRITE :1,B$
0080 NEXT I
0090 CHAIN "READIS"
0100 END

END OF LISTING


RUN
?
primo
?
secondo
?
terzo
?
quarto
?
quinto
?
sesto
?
settimo
?
ottavo
?
nono
?
decimo
?
undicesimo
**** FORMALLY CORRECT PROGRAM ****
QUALE ELEMENTO ISO VUOI?
1
Il primo carattere della Tabella ISO e' ■

QUALE ELEMENTO ISO VUOI?
11
Il undicesimo carattere della Tabella ISO e' ≡

QUALE ELEMENTO ISO VUOI?
```

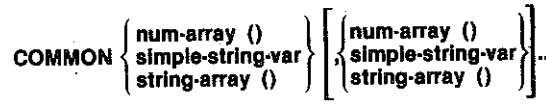
# COMMON

Istruzione COMMON

Funzione

Permette di definire delle variabili di programma i cui valori possono essere comunicati tra programmi distinti.

Formato




dove:

num-array
: è una variabile multipla numerica i cui elementi ricevono (o trasferiscono) dati numerici da (a) un programma BASIC

simple-string-var
: è una variabile semplice stringa che riceve (o trasferisce) una stringa di caratteri da (a) un programma BASIC

string-array
: è una variabile multipla stringa i cui elementi ricevono (o trasferiscono) stringhe di caratteri da (a) un programma BASIC

Azione

L'istruzione specifica quali sono le variabili che il programma può utilizzare per trasferire valori ad un altro programma o per ricevere valori da un altro programma.

Per poter effettuare tale trasferimento di dati tra programmi il sistema utilizza un'area di memoria, detta area comune, in cui memorizza i valori delle suddette variabili. L'utente può specificare il numero di byte da riservare come area comune introducendo un comando SAVE con l'operando COM=m.

Quando in memoria principale è caricato un programma registrato in una libreria mediante un comando SAVE con l'operando COM=m, il sistema riserva i primi m byte della memoria all'area comune e quindi carica il codice del programma. Così, se il programma che è stato eseguito in precedenza utilizzava un'area comune con lo stesso numero di byte o meno, i valori memorizzati dal programma nell'area suddetta non sono alterati -- e sono disponibili per il programma appena caricato.

Come esempio, supponiamo che sia eseguito il seguente programma che era stato registrato nella libreria MAT con il comando SAV PRO1, (MAT,UD), COM=108

```
 10 COM X(),A$(),Y(),B$
 20 DIM X(2,2),A$(4),Y(5)
 30 DCL SX(),6A$()
 50 X(1,1) = 1
 60 X(1,2) = 2
 70 X(2,1) = 3
 80 X(2,2) = 4
 90 A$(1) = "ROMA"
100 A$(2) = "TORINO"
110 A$(3) = "GENOVA"
120 A$(4) = "MILANO"
130 FOR I=1 TO 5
140 Y(I) = I * 10
150 NEXT I
160 B$ = "OLIVETTI P6066"
          -
          -
          -
1000 CHAIN "PRO2"
9999 END
```

I valori assegnati ai componenti della matrice numerica X(), ai componenti della variabile multipla stringa ad una dimensione A$(), ai componenti del vettore Y() ed alla variabile stringa B$, sono registrati, durante l'esecuzione, a partire dall'inizio della memoria utente, nello stesso ordine con cui le variabili sono specificate nell'istruzione COMMON. La figura 5-1 descrive l'area comune e mostra il numero totale di byte occupato da tale area.

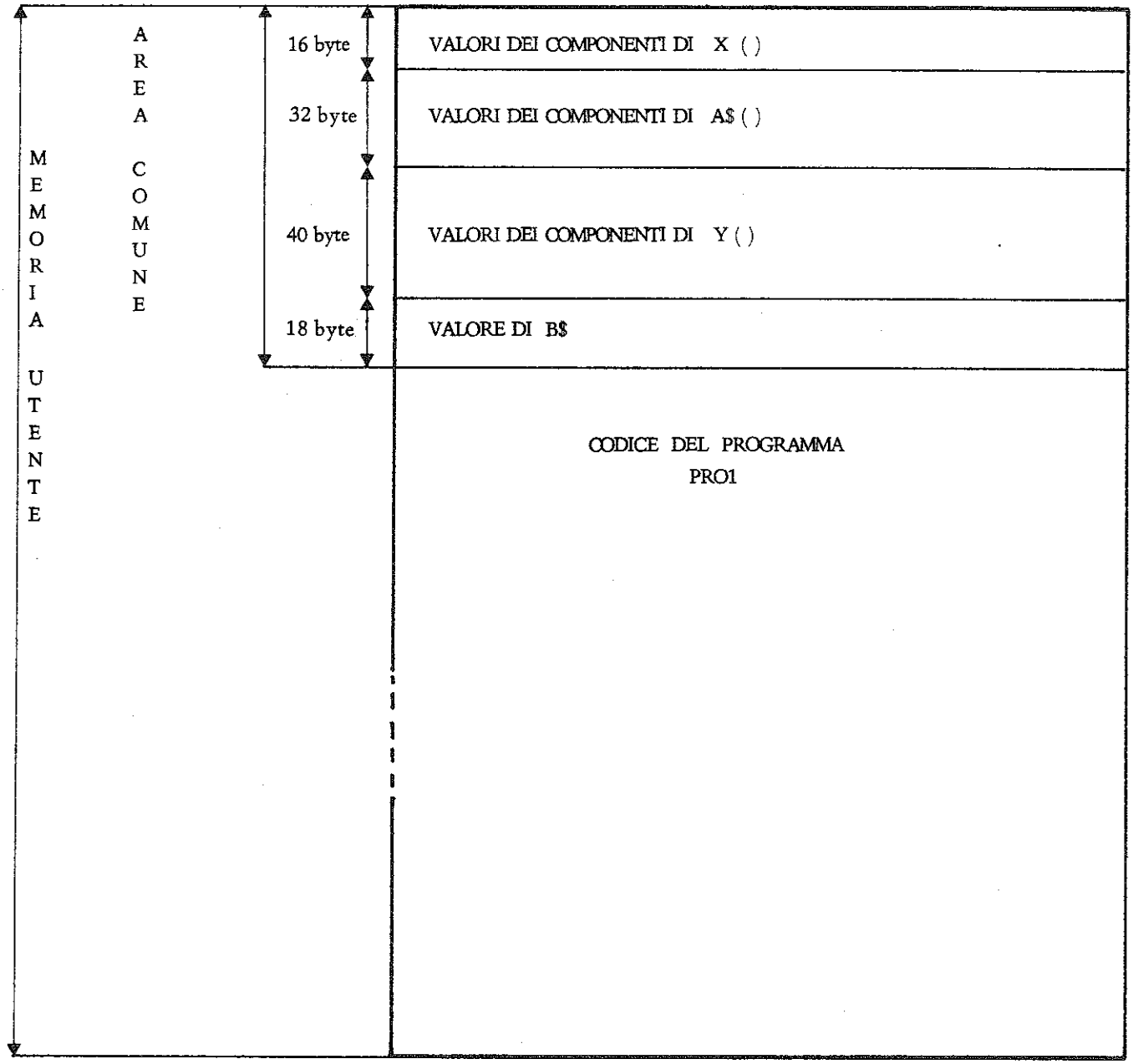



Figura 5-1 L'area comune

Se anche il programma PRO2, caricato in memoria principale con l'istruzione CHAIN di PRO1, è stato registrato in una libreria con la richiesta di un'area comune di 108 byte i valori memorizzati nell'area comune da PRO1 possono essere usati da PRO2, come segue:

```
 10 COMMON X(),A$(),Y(),B$
 20 DIM X(2,2),A$(4),Y(5),N(5),C$(4),C$(4)
 30 DCL S(X(),Z()),6(A$(),C$())
 50 Z(1,1) = X(1,1)
 60 Z(1,2) = X(1,2)
 70 Z(2,1) = X(2,1)
 80 Z(2,2) = X(3,1)
 90 C$(1) = A$(1)
100 C$(2) = A$(2)
110 C$(3) = A$(3)
120 C$(4) = A$(4)
130 FOR I = 1 TO 5
140 N(I) = Y(I)
150 NEXT I
160 D$ = B$
     -
     -
     -
999 END
```

I valori 1, 2, 3 e 4 sono assegnati, rispettivamente, alle variabili Z(1,1), Z(1,2), Z(2,1) e Z(2,2). I valori ROMA, TORINO, GENOVA e MILANO sono assegnati rispettivamente alle variabili C$(1), C$(2), C$(3) e C$(4). I valori 10, 20, 30, 40 e 50 sono assegnati, rispettivamente, alle variabili N(1), N(2), N(3), N(4) ed N(5). Infine, il valore OLIVETTI P6066 è assegnato alla variabile D$. Questi valori sono prelevati dalla area comune, come si può vedere analizzando i programmi PRO1 e PRO2.

Note

1. In un programma BASIC vi può essere una sola istruzione COMMON.

2. Un programma BASIC con una istruzione COMMON non può essere eseguito prima che sia stato registrato in una libreria mediante un comando SAVE con l'operando COM = m.

3. Un programma che utilizza i valori registrati in un'area comune da un programma eseguito in precedenza può essere caricato in memoria principale da una istruzione CHAIN, un comando OLD od un comando RUN filename.

4. Le variabili che si comunicano i valori attraverso un'area comune possono avere nomi differenti.

5. Per richiamare in modo corretto i valori registrati in un'area comune vi deve essere una corretta corrispondenza di locazione del valore rispetto alla variabile che lo utilizza. La posizione di un valore nell'area comune è determinata dalla posizione della corrispondente variabile nell'istruzione COMMON e dal numero di byte occupato dai diversi valori registrati nell'area stessa.

   I valori registrati in un'area comune occupano il seguente numero di byte:

   - ogni elemento di una variabile multipla numerica dichiarata in singola precisione occupa quattro byte

   - ogni elemento di una variabile multipla numerica dichiarata in doppia precisione occupa otto byte

   - ogni elemento di una variabile multipla stringa occupa il numero di byte della sua lunghezza di allocazione, più due byte

   - il valore di una variabile semplice stringa occupa il numero di byte della sua lunghezza di allocazione, più due byte.

   Si noti, infine, che ogni valore numerico è registrato nell'area comune iniziando da una posizione multipla di quattro byte. Così, un valore numerico che è registrato immediatamente dopo una stringa di caratteri può, in alcuni casi, essere registrato dopo tre byte dalla fine della stringa suddetta.

6. I valori numerici registrati nell'area comune come valori di una variabile multipla numerica A possono essere assegnati da un altro programma agli elementi di più variabili multiple numeriche. In questo caso: (1) il numero di elementi di A deve essere uguale alla somma del numero di elementi delle variabili multiple specificate nel secondo programma, (2) il tipo di precisione per la rappresentazione dei valori deve essere lo stesso.

7. Le stringhe di caratteri registrate nell'area comune come valori degli elementi di una variabile multipla stringa A$ possono essere assegnati da un altro programma agli elementi di più variabili multiple stringa oppure a variabili semplici stringa.

Nel primo caso: (1) il numero di elementi di A$ deve essere uguale alla somma del numero di elementi delle variabili multiple stringa specificate nel secondo programma, (2) la lunghezza di allocazione deve essere la stessa.

8. Una stringa di caratteri registrata nell'area comune come valore di una variabile semplice stringa non può essere divisa in sottostringhe da assegnare come valori agli elementi di una o più variabili multiple stringa di un altro programma.

9. Lo spazio richiesto per registrare i valori delle variabili specificate in una istruzione COMMON non può superare il numero di byte specificato per l'area comune dall'operando COM = m del comando SAVE.

10. Lo spazio riservato come area comune per un programma può essere ridotto od allargato quando un altro programma è caricato in memoria principale. Se è ridotto, alcuni valori registrati nell'area comune dal primo programma possono essere perduti. Se è allargato, lo spazio che è aggiunto non conterrà dati significativi finchè non vi sono registrati dal secondo programma.

11. L'utente deve assicurarsi che le variabili usate per comunicare dati tra programmi differenti siano dello stesso tipo.

12. L'area comune non contiene dati significativi finchè non vi siano registrati dal programma.

Istruzione CONVERT

Funzione

Converte ogni carattere di una espressione stringa nel corrispondente codice numerico ISO e viceversa.

Formato

1. **CONVERT string-exp TO num-vector LENGTH num-var**

2. **CONVERT num-vector TO string-var LENGTH num-exp**

dove:

string-exp
: è una espressione stringa il cui valore viene convertito, carattere per carattere, nel corrispondente numero intero, in base dieci, della tabella ISO (vedi appendice E)

num-vector
: è un vettore numerico ai cui elementi vengono assegnati, ordinatamente, i numeri interi corrispondenti ai caratteri del valore dell'espressione stringa specificata; oppure, nel secondo formato, è un vettore numerico di cui vengono codificati i valori degli elementi nei corrispondenti caratteri ISO

num-var
: è una variabile numerica alla quale viene assegnato il numero di caratteri che compongono il valore string-exp

string-var
: è una variabile stringa a cui viene assegnata la sequenza di caratteri ISO corrispondente ai valori numerici degli elementi di num-vector

num-exp
: è una espressione numerica il cui valore, arrotondato all'intero più prossimo, indica quanti elementi di num-vector, iniziando dal primo, devono essere convertiti nei corrispondenti caratteri ISO e quindi assegnati a string-var

Azione

Quando è eseguita l'istruzione indicata con il primo formato, l'espressione string-exp viene valutata ed i caratteri corrispondenti al suo valore vengono convertiti nei corrispondenti numeri in base 10 della tabelle ISO ed assegnati, nell'ordine da sinistra a destra, agli elementi del vettore numerico specificato con num-vector.

All variabile num-var viene assegnato un numero che indica la lunghezza attuale del valore di string-exp.

Quando è eseguita l'istruzione indicata con il secondo formato, vengono convertiti nei corrispondenti caratteri ISO i valori numerici dei primi n elementi del vettore num-vector, arrotondati all'intero più prossimo. La sequenza dei caratteri ISO ottenuta è assegnata alla variabile stringa string-var.

Note

1. Quando è utilizzata l'istruzione CONVERT nel secondo formato, i valori numerici contenuti nel vettore num-vector, arrotondati all'intero più prossimo, devono essere compresi tra zero e 255.

2. Se il numero di caratteri del valore di string-exp è maggiore del numero di componenti del vettore num-vector, nel caso del primo formato, allora viene convertito un numero di caratteri uguale al numero di componenti del vettore suddetto, iniziando dal primo. Viene data una segnalazione di errore recuperabile (appendice D) ed il sistema commuta nello stato di debugging.

3. Se, nel secondo formato, il valore di num-exp arrotondato all'intero più prossimo è maggiore della lunghezza di allocazione della variabile stringa specificata con string-var, allora viene convertito un numero di componenti del vettore num-vector pari al numero di caratteri dichiarati (esplicitamente o implicitamente) per la variabile stringa string-var iniziando dal primo componente. Viene data una segnalazione di errore recuperabile ed il sistema commuta nello stato di debugging.

Esempi

1. Vediamo come si possono convertire le lettere dell'alfabeto nei corrispondenti valori decimali della tabella ISO e viceversa. Come si vede 21.43 è arrotondato a 21.

```
LIST
FILE      CONVER

0010 DCL 21(A$),S(A())
0020 DIM A(21)
0040 CONVERT "ABCDEFGHILMNOPQRSTUVZ" TO A LENGTH B
0050 PRINT
0060 FOR I=1 TO B STEP 1
0070 PRINT "A(";I;")=";A(I),
0080 NEXT I
0090 PRINT
0100 CONVERT A TO A$ LENGTH 21.43
0110 PRINT "A$=";A$
0120 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****

A( 1 )= 65      A( 2 )= 66      A( 3 )= 67      A( 4 )= 68      A( 5 )= 69
A( 6 )= 70      A( 7 )= 71      A( 8 )= 72      A( 9 )= 73      A( 10 )= 76
A( 11 )= 77     A( 12 )= 78     A( 13 )= 79     A( 14 )= 80     A( 15 )= 81
A( 16 )= 82     A( 17 )= 83     A( 18 )= 84     A( 19 )= 85     A( 20 )= 86
A( 21 )= 90
A$=ABCDEFGHILMNOPQRSTUVZ
```

2. In questo caso poichè il vettore A ha solo 10 componenti, quando viene eseguita l'istruzione 30, il sistema fornisce una segnalazione di errore recuperabile e si pone nello stato di debugging (la luce del tasto STEP è accesa). Premendo il tasto di console CONTINUE l'esecuzione del programma riprende e vengono convertiti solo i primi 10 caratteri della stringa dell'alfabeto.

```
LIST
FILE     CONVER

0010 DCL 21(A$),S(A())
0030 CONVERT "ABCDEFGHILMNOPQRSTUVZ" TO A LENGTH 8
0040 PRINT
0050 FOR I=1 TO 10 STEP 1
0060 PRINT "A(";I;")=";A(I),
0070 NEXT I
0080 PRINT
0090 CONVERT A TO A$ LENGTH 8
0100 PRINT "A$=";A$
0110 END

END OF LISTING


RUN
**** FORMALLY CORRECT PROGRAM ****
ERROR 8   IN LINE 30

A( 1 )= 65      A( 2 )= 66      A( 3 )= 67      A( 4 )= 68      A( 5 )= 69
A( 6 )= 70      A( 7 )= 71      A( 8 )= 72      A( 9 )= 73      A( 10 )= 76

A$=ABCDEFGHIL
```