Volume

1

AxesBrain

Linguaggio di programmazione per l'automazione

A B & T
Tecnologie Informatiche

Linguaggio di programmazione

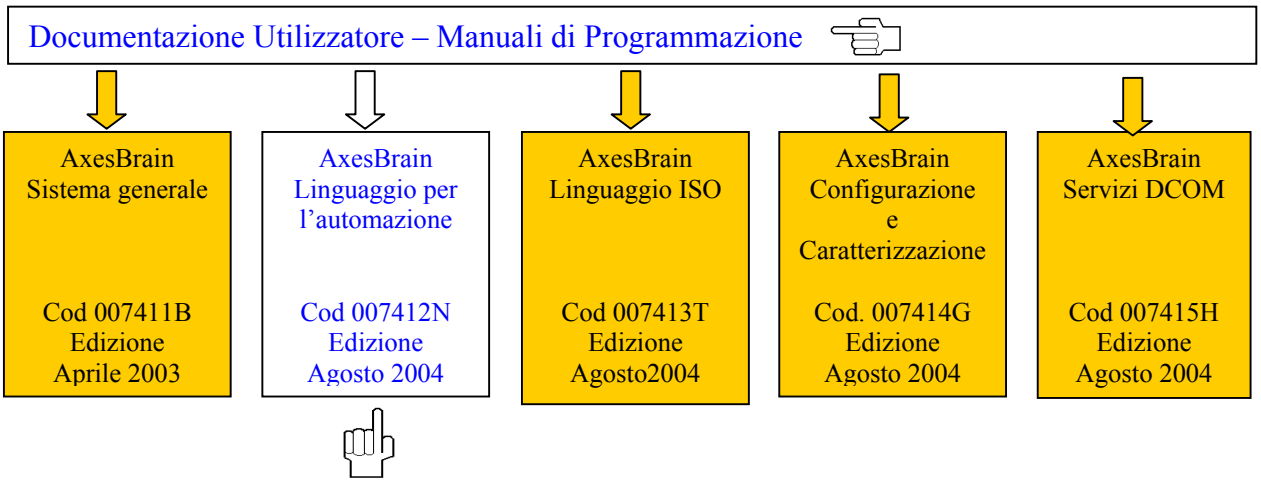
AxesBrain

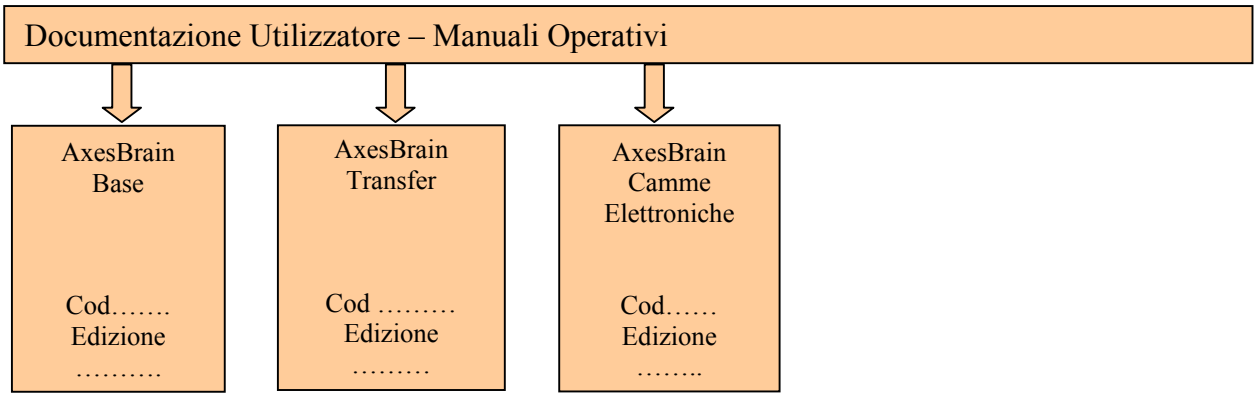
Linguaggio di programmazione AxesBrain

AB&T Tecnologie Informatiche™
Via dell'About, 2/A • 10015 Ivrea
Fax 0125 234397
www.bausano.net
info@bausano.net

Informazioni legali

Panoramica della Documentazione Utilizzatore AxesBrain





INDICE DEGLI ARGOMENTI

CAPITOLO 1

1 Introduzione

1.1 Come utilizzare questo manuale

Questo manuale descrive come utilizzare le risorse fornite dal linguaggio di programmazione AxesBrain.
Il suo scopo è provvedere ai programmatori un rapido aiuto e supporto specialmente per funzionalità che sono utilizzate infrequentemente e fornire offrire una rapida guida in merito.
Il manuale ha, tuttavia, brevi testi descrittivi facilmente comprensibili.

1.2 Linguaggio di programmazione proprietario per descrivere cicli di manipolazione

AxesBrain è un linguaggio di programmazione è indicato per la definizione dei cicli di lavoro per l'automazione.

Per descrivere il ciclo di lavoro di un manipolatore o di una macchina automatica è necessario avere un linguaggio in grado di apprendere come evolvono le fasi. Esistono numerosi linguaggi di programmazione, per lo più sono "proprietari" dell'azienda. Le specifiche che un linguaggio deve avere sono molteplici, programmazione in fasi parallele, sincronizzazione tra le attività programmate, alta integrazione con i dispositivi esterni (visione, laser, eccetera).

Nato dalla specifica del SIGLA (SIGMA LANGUAGE) 1976 dell'Olivetti, come uno dei primi linguaggi di programmazione delle macchine speciali per l'assemblaggio di parti, è stato ampliato in modo da soddisfare le moderne esigenze d'integrazione e flessibilità. Riferimento a :"Robot technology at Olivetti: the sigma system" Olivetti,Milan 1976.

Una sua prerogativa oltre la semplicità di sintassi è la capacità di avere la multiprogrammazione dei singoli cicli, prerogativa indispensabile per poter effettuare delle operazioni di assemblaggio e di manipolazioni di parti.

Il presente manuale intende fornire le nozioni essenziali per la stesura di programmi di lavorazione interpretabili dal sistema .

In particolare esso è così organizzato:

Il **Capitolo 2 Architettura** descrive l'architettura del sotto sistema di automazione, in altre parole di poter eseguire i cicli di lavoro in parallelo tra loro sincronizzandoli con degli eventi.

Il **Capitolo 3 Sintassi del linguaggio** descrive la sintassi del linguaggio di programmazione, la quale è estremamente semplice essendo costituita da un triletterale preceduto da un trattino "-", mentre uno "/" rappresenta il delimitatore dai parametri dell'istruzione.

Il **Capitolo 4 Variabili globali e locali** definisce ed introduce ai concetti delle variabili utilizzate dal sistema suddivise in globali e locali.

Il **Capitolo 5 Multitask e anticollisione** introduce al concetto di poter effettuare più operazioni insieme, in sicurezza e coordinate tra loro o meno, funzionalità di multitask e anticollisione degli assi.

Il **Capitolo 6 Istruzioni logiche matematiche** affronta il problema delle istruzioni logiche matematiche, elencandone la lista e esemplificando ogni istruzione.

Il **Capitolo 7 Istruzioni di controllo** illustra l'elenco delle istruzioni disponibili. Per ogni istruzione è previsto un descrittivo di utilizzo e un esempio di programmazione.

Il **Capitolo 8 Istruzioni di movimentazione** descrive le istruzioni di programma relative alla gestione degli assi. Per ogni istruzione è previsto un descrittivo di utilizzo e un esempio di programmazione.

Il **Capitolo 9 Istruzioni per i segnali di input ed output** illustra l'elenco delle istruzioni disponibili. Per ogni istruzione è previsto un descrittivo di utilizzo e un esempio di programmazione.

Il **Capitolo 10 Istruzioni di sincronizzazione** è dedicato alla descrizione del set di struzioni necessari alla sincronizzazione degli eventi.

Il **Capitolo 11 Istruzioni di servizio** illustra l'elenco delle istruzioni disponibili. Per ogni istruzione è previsto un descrittivo di utilizzo e un esempio di programmazione.

Il **Capitolo 12 Istruzioni per l'integrazione con altri ambienti** affronta il problema della integrazione del sistema con altri ambienti di lavoro , fornisce le informazioni necessarie alla gestione e un elenco delle istruzioni disponibili con esempi di utilizzo..

Il **Capitolo 13 Istruzione di comunicazione** illustra un elenco delle istruzioni disponibili, con esempi di utilizzo ,al fine di mettere in comunicazione il sistema con altre risorse

Il **Capitolo 14 Istruzioni AWL PLC** illustra l'elenco delle istruzioni disponibili relative al linguaggio di programmazione AWL (linguaggio di programmazione letterale per PLC). Per ogni istruzione è previsto un descrittivo di utilizzo e un esempio di programmazione.

L'**Appendice A Assi virtuali** illustra il concetto di assi virtuali su sistemi a geometria SCARA e CILINDRICA.

L'**Appendice B Controllo asse con volantino** definisce la possibilità di posizionamento manuale di un asse tramite un volantino elettronico.

L' **Appendice C Assi a portale (Gantry)** illustra il concetto di assi Gantry gestito dal sistema.

L' **Appendice D Camme elettroniche** illustra il concetto di camma elettronica e la possibilità di gestione da parte del sistema.

L' **Appendice E Movimenti in continuo** definisce e esemplifica le varie possibilità di generazione tipi di profili con movimento continuo.

L' **Appendice F Integrazione con la visione** illustra il concetto di integrazione del sistema in un ambiente di analisi di immagini.

L' **Appendice G Elenco errori** elenca la lista degli errori possibili che vengono generati durante una errata programmazione o configurazione del sistema.

CAPITOLO 2

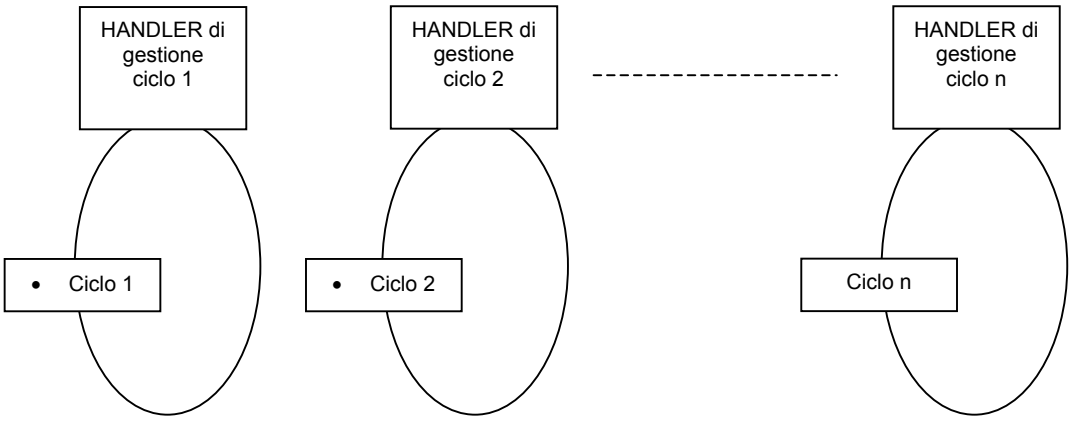
2 ARCHITETTURA

L'architettura del sotto sistema di automazione è quella di poter eseguire i cicli di lavoro in parallelo tra loro sincronizzandoli con degli eventi.

Ogni programma ha al proprio interno un area proprietaria dove sono allocate le variabili LOCALI ,su cui tra l'altro vengono depositati i parametri di chiamata.

Quando al programma viene inserito in esecuzione, da un comando esplicito o richiamato da un CALL o da un'istruzione TSK, il suo codice sorgente viene caricato in memoria ed automaticamente precompilato, in memoria vi rimarrà fino ad un comando esplicito di cancellazione o comando al sottosistema di RESET.

I programmi vengono abbinati ad un HANDLER di lavoro proprietario su cui si appenderanno tutte le attività del ciclo, possiamo aver un numero pressoché infinito di HANDLER, con questa modalità si possono avere configurazione d'impianto estremamente flessibili, come linee di ROBOTS, macchine speciali multi testa, macchine con carichi scarichi integrati , macchine multifunzionali, etc.


HANDLER di gestione ciclo 1
HANDLER di gestione ciclo 2
HANDLER di gestione ciclo n
Ciclo 1
Ciclo 2
Ciclo n

CAPITOLO 3

3 SINTASSI DEL LINGUAGGIO

3.1 Definizioni del linguaggio

La sintassi del linguaggio è estremamente semplice un triletterale preceduto da un trattino "-" rappresenta l'istruzione, uno "/" rappresenta il delimitatore dai parametri dell'istruzione che possono essere:

1. Riferimenti diretti a variabili GLOBALI o LOCALI
2. Riferimenti indiretti a variabili GLOBALI
3. Espressioni numeriche con riferimenti alle variabili GLOBALI o LOCALI
4. Espressioni matematiche con riferimenti alle variabili GLOBALI o LOCALI
5. LABEL di salto
6. Nomi di risorse assi , mandrini, input e output

I parametri sono separati da virgole.

Le etichette o LABEL per i salti condizionati o incondizionati sono messe prima del separatore trattino "-" dell'istruzione.

I commenti sono preceduti da carattere punto e virgola ";"

[label] –XXX/[parametro 1],…[parametro n] [; questo è un commento]

Si possono commentare più righe usando "/*" all'inizio dell'area commentata e "*/" alla fine dell'area

Esempio:

```
;  questo è un esempio di programmazione
     -LET/G1,0
qui  -JNE/G1,1,qui              ; attende che il valore della variabile GLOBALE G1 sia posto a 1
; da un altro ciclo in lavorazione parallela
```

Le espressioni matematiche hanno le seguenti funzioni:

abs	Assoluto di un numero
acos	Arco coseno
and	And tra due numeri
asin	Arco seno
atan	Arco tangente
atanw	Arco tangente di Y,X
ceil	Arrotondamento verso l'alto di un numero decimale in un numero intero
cos	Coseno
cosh	Coseno iperbolico
deg	Trasformazione in gradi di un angolo espresso in radianti
exp	Esponenziale
floor	Arrotondamento verso il basso di un numero decimale in un numero intero
logd	Logaritmo decimale
logn	Logaritmo naturale
lshift	Shift verso sinistra di un numero
max	Massimo fra enne espressioni
min	Minimo fra enne espressioni
mod	Modulo fra due numeri
not	Negazione boleana di un numero
or	Or fra due numeri
pi	P greco
rad	Trasformazione in radianti di un angolo espresso in gradi
rshift	Shift verso destra di un numero
sin	Seno
sinh	Seno iperbolico
sqr	Radice quadrata
tan	Tangente
tanh	Tangente iperbolica
xor	Or esclusivo fra due numeri
tim	restituisce un valore in secondi riferiti dal 1 gennaio 1970

Gli operatori in una espressione sono:

\+ somma
\- sottrazione
/ divisione
* moltiplicazione
^ elevato
(parentesi aperta
) parentesi chiusa

Esempio di espressione matematica

```
    -LET/L1, MAX(SIN(RAD(G1+12),COS(RAD(G1+12*L1/56))
 ; nella variabile LOCALE L1 viene caricato il risultato della espressione:
;                MAX(SIN(RAD(G1+12),COS(RAD(G1+12*L1/56))
```

Note

a) I numeri in esadecimale sono preceduti da 0x

Esempio:

```
    -LET/L1,0x10                    ; 0x10 = 16 decimale
```

L'utilizzo dell'operatore esadecimale "0x" è particolarmerne utile nella mascheratura con le funzioni di "and" e "or", per poi usarli successivamente-

Esempio:

```
    -LET/L1,and(L1,0x8000)
    -JEQ/L1,0x8000,Bit8000Uno
```

3.2 Dimensioni massima dei parametri richiamabili

GLOBALI	32.767
LOCALI	configurate nella voce NumeroLocali= del file di configurazione "SISTEMA.TXT" (ampliabili da –DIM/Numero LOCALI)
WATCH	16
DRT	6
Righe DIS	17
Colonne DIS	128
Point	32000
NETwork	128
Timer (T)	32
Counter	32

CAPITOLO 4

4 Variabili globali e locali

Per poter effettuare delle operazioni logiche, leggere valori numerici sono necessarie le variabili.

Il sotto sistema AXESBRAIN prevede due tipi di variabili :

LOCALI

GLOBALI

Ogni programma all'atto dell'entrata in funzione si alloca un numero di variabili pari a quello configurate nel sistema, vengono tutte azzerate e sono a disposizione delle istruzioni di quel programma, le prime variabili vengono impostate con i parametri di chiamata del comando di esecuzione, le variabile rimarranno in memoria a disposizione per le operazioni di interrogazioni o visualizzazione.

Il numero di LOCALI può essere cambiato dal quello previsto nel "**SISTEMA.TXT**" alla voce "**NumeroLocali**=", utilizzando l'istruzione –**DIM**/numero locali, che rimane definito solo per quella istanza di task.

Un caso particolare sono i programmi richiamati dalle istruzioni di "-CAL" e di "-TSK" con parametri, in questi casi le prime variabili LOCALI sono caricate con i parametri di chiamate nello stesso ordine posizionale, se esistono delle LOCALI nei parametri di chiamate verranno caricate con il nuovo valore al ritorno del programma chiamato.

Esempio:

```
-CAL/,routinemia:L9,12,23,G1+89,L7
```

quando avviene il ritorno "routinemia " L9 ed L7 avranno il valore definito nella routine, in questo caso

L9 avrà il valore di 11

L7 avrà il valore di 3

Infatti il codice di "routinemia" è il seguente:

```
-LET/L1,11
-LET/L5,3
```

All'atto del esecuzione le variabili LOCALI di "routinemia" sono:

L1=valore di L9 del programma chiamante

L2= 12

L3=23

L4=valore della variabile G1 + 89

L5=valore di L7 del programma chiamante

Oltre le variabili LOCALI nel sotto sistema di automazione sono previste **32767** variabili **GLOBALI** che vengono richiamate con la lettera "G" ed il numero della stessa.

Tutte le **GLOBALI** vengono salvate su HardDisk, che quindi possono essere usate per memorizzare dati persistenti.

Le variabili indirette rappresentate dalla lettera "I" seguita dal numero indica che il valore che viene trattato è quella della variabile GLOBALE il cui numero è quello della variabile LOCALE indicata dalla lettera "I".

Esempio:

```
-LET/L12,100
-LET/I12,120.56
```

In questo caso il valore 120.56 è caricato nella variabile GLOBALE 100, infatti il valore 100 indicante la GLOBALE è stato caricato nella LOCALE 12 , di cui si fa riferimento nella istruzione "-LET/I12,120.56".

Per indicizzare Globali o Locali si può usare la parentesi seguito dalla Globale o Locale utilizzate come indice (l'uso della "I" è rimasta per compatibilità)

Esempio:

```
;  questo è un esempio di programmazione indicizzata
 -LET/ L1,1
 -LET/G(L1),0
qui  -JNE/G(L1)1,1,qui                    ; attende che il valore della variabile GLOBALE G1 sia posto a 1
```

Le Globali sono salvate su HardDisk all'uscita del sistema o nella procedura di "SHUT DOWN" è comunque possibile effettuare l'operazione con l'istruzione **SGL.**

L'OPERAZIONE di scrittura delle Globali viene effettuata con una modalità di COMMIT per assicurare l'integrità con l'ultimo salvataggio.

CAPITOLO 5

5 Multitask e anticollisione

Una caratteristica importante nell'automazione è il poter effettuare più operazioni insieme, coordinate tra loro o meno, quindi abbiamo la necessità di avere la funzionalità di "MULTITASK".

Un ciclo di attività può essere eseguito con un comando esplicito, oppure da una istruzione "-TSK", il ciclo o programma viene abbinato ad un HANDLER di lavoro proprietario su cui si appenderanno tutte le attività del ciclo, possiamo aver un numero pressoché infinito di HANDLER.

Un "TASK" può essere cancellato da parte di un altro task o da se stesso con l'istruzione "-TKM" oppure quando viene eseguito il comando di RESET per il sotto sistema di automazione.

Il concetto di HANDLER è utile inoltre vederlo come un canale su cui vengono effettuate operazioni di inizio della movimentazione in continuo , aggregazioni di movimento , attesa che i movimenti siano completati.

Con questa architettura noi siamo in grado di vedere il sistema di movimentazione come una apparecchiatura con numerosi "Bracci" che lavorano insieme coordinati o meno, aggregando dinamicamente gruppi di assi,.

Un esempio interessante può essere immaginare lo riempimento di un vassoio di bicchieri, in una fase iniziale abbiamo due "Bracci" che separatamente riempiono i calici, il sistema gestisce l'anticollisione e abbiamo così due gruppi di assi che lavorano separatamente, quando i calici sono stati riempiti il sistema, raggruppando i due bracci come farebbe un cameriere, porta il vassoio in zona di scarico.

Come prima evidenziato durante il riempimenti dei calici, i due bracci incidendo su uno stesso asse fisico X, grazie alla gestione dell'anticollisione è possibile programmare due cicli indipendenti di riempimento, sincronizzare i due a completamento delle loro rispettive fasi, quindi programmare un unico ciclo di scarico del vassoio con un raggruppamento unico dei due bracci.

CAPITOLO 6

6 Istruzioni logiche matematiche

.1-LET(SET)	Imposta ad una Variabile il valore dell'espressione (LET)
.2-ADD	Somma ad una Variabile il valore dell'espressione (ADDed)
.3-MUL	Moltiplica una Variabile con il valore dell'espressione (MULtiply)
.4-DIV	Dividi una Variabile con il valore dell'espressione (DIVided)
.5-NEG	Nega il valore di una Variabile (NEGation)
.6-LBF	Imposta ad un vettore di Variabili il valore dell'espressione (Load BuFfer)

6.1 LET: LET value (SET)

LET(SET) LET value (SET value)

Funzione: eseguire una operazione aritmetica di inizializzazione di variabile.

Parametri: **a1,b1,a2,b2,...an,bn**

dove:

da a1 .. an = variabile da inizializzare
da b1 .. bn = valori da introdurre nelle variabili

Esempio:

-LET/L1,5.5,G7,L3	Inizializzare la variabile locale 1 con il valore 5.5 e la variabile globale 7 con il valore contenuto nella variabile locale 3.

Note:

a) Se il parametro **b** è una variabile, questa mantiene sempre il valore iniziale.

b) Per ogni istruzione è possibile inizializzare fino a 16 variabili.

c) Il parametro **b** può essere come d' altraparte tutti i parametri statici un'espressione matematica.

d) La "LET" rende le istruzioni "ADD" , "MUL", "DIV" e "NEG" ridondanti, ma sono state implementatate per conformità con le specifiche precedenti.

Esempio:

```
-LET/L1,L1+4  ; corrisponde  -ADD/L1,4
-LET/L1,L1*4  ; corrisponde  -MUL/L1,4
-LET/L1,L1/4  ; corrisponde  -DIV/L1,4
-LET/L1,-L1   ; corrisponde  -NEG/L1
```

6.2 ADD: ADD value

ADD **ADD value**

Funzione: eseguire una somma algebrica (a= a±b)

Parametri: **a1,b1,a2,b2,...an,bn**

dove:

da a1 a an = variabili operandi nelle quali si otterrà il risultato

da b1 a bn = valori da sommare algebricamente

Esempio:

-ADD/L1 ,-5,G2,L3 Sottrarre 5 alla variabile locale 1 e sommare alla variabile globale 2 il valore della variabile locale 3

Note:

a) Se il parametro **b** è una variabile, questa mantiene sempre il suo valore iniziale.

b) Per ogni istruzione è possibile operare su un massimo di 16 variabili.

6.3 MUL: MULtiplication

MUL MULtiplication

Funzione: eseguire una moltiplicazione (a= a * b)

Parametri: **a1,b1,a2,b2...an,bn**

dove:

da a1 a an = variabili operandi nelle quali si ottiene il risultato

da b1 a bn = valori da moltiplicare

Esempio:

-MUL/G1,5.3,L2,G2	Moltiplicare per 5.3 il valore della variabile globale 1 e moltiplicare per il valore della variabile globale 2 il valore della locale 2.

Note:

a) Se il parametro **b** è una variabile, questa mantiene sempre il suo valore iniziale.

b) Per ogni istruzione è possibile operare su un massimo di 16 variabili

6.4 DIV: DIVision

DIV **DIVision**

Funzione: eseguire una divisione (a=a/b)

Parametri: **a1,b1,a2,b2...an,bn**

dove:

da a1 a an = variabili operandi nelle quali si ottiene il risultato

da b1 a bn = valori da dividere

Esempio:

-DIV/L2,5.7,G4,L1 Dividere per 5.7 il valore della variabile locale 2 e dividere per il valore della variabile locale 1 il valore della globale 4.

Note:

a) Se il parametro **b** è una variabile, questa mantiene sempre il suo valore iniziale.

b) Per ogni istruzione è possibile operare su un massimo di 16 variabili.

c) La divisione per zero genera il mesaggio d'errore "errore nell'interpretazione dell'istruzione".

6.5 NEG: NEGation

NEG NEGation

Funzione: cambiare il segno algebrico di una variabile a=-a

Parametri: **a1,a2,...an**

dove:

da a1 a an = variabili da moltiplicare per -1

Esempio:

-NEG/L1,G7 Moltiplicare per -1 i valori delle variabili locale 1 e globale 7

Note:

a) Per ogni istruzione è possibile operare su un massimo di 16 variabili.

6.6 LBF: Load BuFfer

LBF **Load BuFfer**

Funzione: eseguire una operazione aritmetica di inizializzazione di un buffer di variabili.

Parametri: **a,b,c**

dove:

a = variabile Locale o Globale iniziale

b = variabile Locale o Globale finale

c = valore da introdurre nelle variabili

Esempio:

-LBF/G1,G100,0 Azzerare le variabili globali comprese fra 1 e 100.

Note:

a) Se il parametro **c** è una variabile, questa mantiene sempre il valore iniziale.

b) Le variabili indicate nei parametri **a, b** devono essere dello stesso tipo e progressive.

CAPITOLO 7

7 ISTRUZIONI DI CONTROLLO

.1-JMP	Salta incondizionatamente ad una Label (JuMP)
.2-JEQ	Salta ad una Label se le due espressioni sono uguali (Jump if Equal)
.3-JNE	Salta ad una Label se le due espressioni non sono uguali (Jump in Not Equal)
.4-JLT	Salta ad una Label se il valore del primo parametro è minore del secondo (Jump if Less Then)
.5-JLE	Salta ad una Label se il valore del primo parametro è minore o uguale del secondo (Jump if Less then and Equal)
.6-JGT	Salta ad una Label se il valore del primo parametro è maggiore del secondo (Jump if Great Then)
.7-JGE	Salta ad una Label se il valore del primo parametro è maggiore o uguale del secondo (Jump if Great then and Equal)
.8-JRN	Salta se il valore del parametro è compreso nel range (Jump if RaNge)
.9-JNR	Salta se il valore del parametro è fuori del range(Jump if Not Range)
.10-JOS	Salta se almeno un bit del valore del parametro è uno (Jump Or if bit Set)
.11-JOC	Salta se almeno un bit del valore del parametro è zero (Jump Or if bit Clear)
.12-JAS	Salta se tutti i bit del valore del parametro sono uno (Jump And if bit Set)
.13-JAC	Salta se tutti i bit del valore del parametro sono zero (Jump And if bit Clear)
.14-CAL	Chiama un part program passandogli dei parametri (CALl)
.15-RET	Ritorna al chiamante del programma (RETurn)
.16-END	Fine processo (END)
.17-TSK	Esegue in parallelo un ciclo di lavoro (TaSK)
.18-TKM	Sospende, ripristina e cancella un ciclo di lavoro (TasK Manegement)
.19-DIM	Dimensiona il numero di variabile L di un part pogram

7.1 JMP: JuMP

JMP JuMP

Funzione: salto incondizionato alla label indicata (nello stesso part program).

Parametri: **a**

dove:

a = Label di salto

La label è una stringa alfanumerica senza spazi , la label può stare singolarmente su una riga deve però esistere il carattere "-"

Esempio:

```
-JMP/Pippo              Il programma salta alla label  Pippo.
...
...
Pippo - .....
```

Note:

a) Questa funzione è valida solo all'interno di uno stesso part-program.

b) In uno stesso part-program possono coesistere più **JMP** con la stessa label.

c) Un'uso particolare è il il salto indicizzato al valore di una LOCALE o GLOBALE
–JMP/L# oppure –JMP/G#
in questo caso il salto avverrà alla label numerica equivalente al valore intero contenuto nella variabile.

Esempio:

```
-JMP/L1         ; se il contenuto di L1 è 23 salteà alla label 23
...             ; se il contenuto di L1 è 24 salteà alla label 24
...             ; se il contenuto di L1 è 35 salteà alla label 35
23 - .....


24  - .....

35  - .....
```

7.2 JEQ: Jump EQual

JEQ **Jump EQual (Jump se =)**

Funzione: eseguire un test su n valori ed in funzione del risultato eseguire o no il salto condizionato.

Parametri: **a,b1,...bn,c**

dove:

a = variabile o vocabolo di sistema inerente il test

da b1 a bn = valori da confrontare (max. 14)

c = n° di salto se si verifica la condizione prevista cioè se almeno un parametro **c** è uguale al parametro **b**.

Esempio1:

```
-JEQ/L1,6,14,13,Equal        Se il contenuto della variabile locale 1 è 6 oppure 14 oppure 13, il
                             programma salta alla label Equal; in caso contrario procede
…..
…..
…..
Equal….
```

Esempio2:

```
a                            -JEQ/L1,0,1,2,3,4,L1
                             -LET/L1,L1+1
                             -DIS/pippo,L1
                             -JMP/L1
                             -JMP/a
5-
32-
56-
                             -TMM/1000
                             -DIS/>
                             -END

Fine-
                             -TMM/1000
                             -DIS/>          .
```

7.3 JNE: Jump Not Equal

JNE **Jump Not Equal (Jump se diverso)**

Funzione: eseguire un test su n valori ed in funzione del risultato eseguire o no il salto condizionato.

Parametri: **a,b1,...bn,c**

dove:

a = variabile o vocabolo di sistema inerente il test

da b1 a bcn = valori da confrontare (max. 14)

c = n° di salto se si verifica la condizione prevista cioè se tutti i parametri **c** sono diversi dal parametro **b**.

Esempio1:

```
-JNE/L1,10,12,-6,Noequal        Se il contenuto della variabile locale 1 è diverso da 10,12,-6, il programma
                                salta alla label Noequal; in caso contrario procede
…….
…….
Noequal…..
```

Esempio2:

```
a                               -JNE/L1,0,1,2,3,4,L1
                                -LET/L1,L1+1
                                -DIS/pippo,L1
                                -JMP/L1
                                -JMP/a
5-
32-
56-
                                -TMM/1000
                                -DIS/>
                                -END

Fine-
                                -TMM/1000
                                -DIS/>
```

7.4 JGT: Jump Greater Than

7.5 JGE: Jump Greater Equal

7.6 JLT: Jump Less Than

7.7 JLE:Jump Less Equal

JGT Jump Greater Than (Jump se>)
JGE Jump Greater Equal (Jump se ≥)
JLT Jump Less Than (Jump se <)
JLE Jump Less Equal (Jump se ≤)

Funzione: eseguire un test su un valore ed in funzione del risultato eseguire o no il salto condizionato.

Parametri: **a,b,c**

dove:

a = variabile o vocabolo di sistema inerente il test

b = valore da confrontare

c = n° di salto se si verifica la condizione prevista

Esempio:

```
-JGT/L1,4,Salt   Se il contenuto della variabile locale 1 è superiore a 4, il programma salta alla label Salt; in caso contrario procede
...
...
Salt    -
```

7.8 JRN: Jump RaNge

JRN **Jump RaNge**

Funzione: eseguire un test su un range (valore compreso fra 2 limiti) ed eseguire il salto se Tale valore è compreso nel range.

Parametri: **a,b,c,d**

dove:

a = variabile o vocabolo di sistema inerente il test

b = valore minimo previsto

c = valore massimo previsto

d = n° di salto se si verifica la condizione prevista (cioè, se **a** è compreso nel range **bc** (estremi compresi)

Esempio:

```
-JRN/L1,30,50,Salt      Se 50 > L1 > 30, il programma salta alla label Salt; in caso contrario procede
…
Salt-
```

7.9 JNR: Jump Not Range

JNR **Jump Not Range**

Funzione: eseguire un test su un range (valore compreso fra due limiti) ed eseguire il salto se tale valore è esterno al range.

Parametri: **a,b,c,d**

dove:

a = variabile o vocabolo di sistema inerente il test

b = valore minimo previsto

c = valore massimo previsto

d = n° di salto se si verifica la condizione prevista (cioè se **a** è minore o maggiore del range **bc**)

Esempio:

```
-JNR/L1,30,50,Salt      Se L1 è minore di 30 o maggiore di 50, il programma salta alla label Salt; in caso
                        contrario procede
...
Salt-
```

Esempio:

```
-RAV/6:L11,Z      ; 6   quota reale assoluta asse Z in L11
-RAV/12:L12,Z     ; 12 over travel positivo assoluto asse Z in L12
-JNR/L11,L12-3,L12+3,FuoriPosizione
```

7.10 JOS: Jump Or if bit Set

JOS **Jump Or if bit Set**

Funzione: eseguire un test parziale sul valore di una variabile (maschera di bit) ed in funzione del risultato salta se almeno uno dei bit del valore del parametro sono uno.

Parametri: **a,b,c**

dove:

a = variabile da testare

b = valore (maschera) di test

c = JUMP se si verifica la condizione

Esempio:

```
L1 = 11

-JOS/L1,6,Salt
        … no    i bit testati non corrispondono
Salt.
        … ok    i bit testati corrispondono

maschera di L1                  1       0       1       1               (1+2+8)
maschera di 6                           1       1       0               (2+4)
```

Poichè almeno uno dei bit di test corrispondono a quelli della variabile L1 avviene il salto condizionato.

Note:

a) In questa istruzione il test viene effettuato solo sui bit indicati dal parametro **b**; gli altri bit vengono ignorati.

7.11 JOC: Jump OR if bit Clear

JOC **Jump Or if bit Clear**

Funzione: eseguire un test parziale sul valore di una variabile (maschera di bit) ed in funzione del risultato salta se almeno uno dei bit del valore del parametro sono zero.

Parametri: **a,b,c**

dove:

a = variabile da testare

b = valore (maschera) di test

c = JUMP se si verifica la condizione

Esempio:

```
L1 = 15

-JOS/L1,6,Salt
        … no    i bit testati non corrispondono
Salt.
        … ok    i bit testati corrispondono
```

maschera di L1	1	1	1	1	(1+2+4+8)
maschera di 6		1	1	0	(2+4)

Poichè tutti i bit di test della maschera rispetto a quelli della variabile L1 sono a uno non avviene il salto condizionato.

Note:

a) In questa istruzione il test viene effettuato solo sui bit indicati dal parametro **b**; gli altri bit vengono ignorati.

7.12 JAS: Jump And if bit Set

JAS **Jump And if bit Set**

Funzione: eseguire un test parziale sul valore di una variabile (maschera di bit) ed in funzione del risultato salta se tutti i bit del valore del parametro sono uno.

Parametri: **a,b,c**

dove:

a = variabile da testare

b = valore (maschera) di test

c = JUMP se si verifica la condizione

Esempio:

```
L1 = 11

-JAS/L1,6,Salt
        … no    i bit testati non corrispondono
Salt.
        … ok    i bit testati corrispondono
```

maschera di L1	1	0	1	1	(1+2+8)
maschera di 6		1	1	0	(2+4)

Poichè i bit di test non corrispondono a quelli della variabile L1 non avviene il salto condizionato.

Note:

a) In questa istruzione il test viene effettuato solo sui bit indicati dal parametro **b**; gli altri bit vengono ignorati.

7.13 JAC: Jump And if bit Clear

JAC **Jump And if bit Clear**

Funzione: eseguire un test parziale sul valore di una variabile (maschera di bit) ed in funzione del risultato salta se tutti i bit del valore del parametro sono zero.

Parametri: **a,b,c**

dove:

a = variabile da testare

b = valore (maschera) di test

c = JUMP se si verifica la condizione

Esempio:

```
L1 = 9

-JAS/L1,6,Salt
        … no    i bit testati non corrispondono
Salt.
        … ok    i bit testati corrispondono
```

maschera di L1	1	0	0	1	(1+8)
maschera di 6		1	1	0	(2+4)

Poichè i bit di test della maschera rispetto a quelli della variabile L1 sono a zero avviene il salto condizionato.

Note:

a) In questa istruzione il test viene effettuato solo sui bit indicati dal parametro **b**; gli altri bit vengono ignorati.

7.14 CAL: CALl

CAL CALI

Funzione: richiamare il part-program con o senza parametri da inviare e ricevere

Parametri: **a: [c1,...c32]**

dove:

a = label del part program richiamato**,**
nome file del part program richiamato

da c1 a c32 = parametri da trasmettere nelle variabili locali del part program da eseguire

Esempio:

-CAL/,c:\pippo.pp:5.4,L5,G3 Richiamare il part program 3 trasmettendo il valore 5.4 nella variabile locale 1, il valore corrispondente alla variabile locale 5 nella variabile locale 2 e il valore corrispondente alla variabile globale 3 nella variabile locale 3

La label di part program all'interno di un file: PartProgram**[label**]

Nota:

1 se tra **c1** e **c2** c'è i due punti come separatore ed c1 e **c2** sono variabili consecutive, tutte le variabili indicate tra c1 e c2 sono trasferite tra i due programmi.
esempio:
–CAL/pippo:L2:L45 sono trasferite le variabili da L2 fino L45 che andranno nel part program pippo come L1 fino L44

7.15 RET: RETurn

RET **RETurn**

Funzione: provocare l'uscita dal part program durante l'esecuzione del ciclo. Se era un routine ritorna al chiamante del programma.

Parametri: **(nessuno)**

Esempio:

```
-RET
...
...
-RET
...
...
-RET
```

Note:

a) Su uno stesso part-program possono esserci più **RET**.

7.16 END: END

END **End**

Funzione: provocare l'uscita dal part program durante l'esecuzione del ciclo. Se era una routine interrompe il processo comunque.

Parametri: **(nessuno)**

Esempio:

```
-END
...
...
-END
...
...
-END
```

Note:

a) Su uno stesso part-program possono esserci più **END**.

7.17 TSK: TaSK

TSK TaSK

Funzione: esecuzione di un task in parallelo ad altri

Parametri: **a:[b1,...B15]:[c]**

dove:

a	=	Label,Nome file del task (part program) da eseguire
da b1 a b15	=	parametri da trasmettere nelle variabili locali del part program da eseguire
c	=	modalità di controllo del task 0 = non controlla e lancia l'esecuzione di una istanza del TASK (default) 1 = se il task esiste non lo esegue,ma ne accoda l'esecuzione 2 = segnala un errore di task già operante 3 = se il processo chiamante è di tipo sempre attivo anche il task sarà dello stesso tipo(sempre attivo, non resettabile) 4 = Il task sarà sempre attivo, non resettabile

Esempio:

```
-TSK/,C:\task1.pp
-TSK/,\task2.pp:50,G4
-RET
```

Note:

a) I parametri sono trasmessi posizionalmente (**b1** in **L1**, **b2** in **L2**,...**b15** in **L15**). E' possibile trasmettere fino a 32 parametri.
b) I parametri indicati come Locali verranno aggiornati alla fine dell'esecuzione del TASK se il part program che ne richiede l'esecuzione è ancora operante.
c) se tra **c1** e **c2** c'è i due punti come separatore ed c1 e **c2** sono variabili consecutive, tutte le variabili indicate tra c1 e c2 sono trasferite tra i due programmi.

esempio:

–CAL/pippo:L2:L45 sono trasferite le variabili da L2 fino L45 che andranno nel part program pippo come L1 fino L44

Esempio multitask per un azzeramento multitesta:

```
;
;                1   = mask asse ZL
;                2   = mask asse ZR
;                4   = mask asse XL
;                8   = mask asse XR
;                16  = mask asse Y
;                32  = mask asse SL
;                64  = mask asse SR
;                128 = mask asse LU

                 -DIS/0:>
                 -DIS/1:>
                 -DIS/2:>
                 -DIS/3:>
                 -DIS/4:>
                 -DIS/5:>
                 -DIS/6:>
                 -DIS/7:>
                 -DIS/8:>
                 -EVC/1,2,3,4,5,6,7,8,9
                 -LET/L10,1+2+4+8+16+32+64+128-1
                 -LET/L11,AND(L10,1)
                 -JNE/L11,1,NO_ZL
;                0 = Posizione asse ZL con origine  fatta
                 -TSK/OMOZL:0
                 -JMP/DO_ZL
NO_ZL            -EVS/1
DO_ZL            -LET/L11,AND(L10,2)
                 -JNE/L11,2,NO_ZR
;                0 = Posizione asse ZR con origine  fatta
                 -TSK/OMOZR:0
                 -JMP/DO_ZR
NO_ZR            -EVS/2
DO_ZR            -EVW/1,2
                 -LET/L11,AND(L10,12)
                 -JNE/L11,12,NO_XLXR

;                 500 = Posizione asse XR con origine  fatta
;                -550 = Posizione asse XR per ricerca MARKER
;                -500 = Posizione asse XL con origine  fatta
;                 600 = Posizione asse XL per ricerca MARKER
                 -TSK/OMOX:500,-550,-500,600
                 -JMP/DO_XLXR
NO_XLXR          -EVS/9
DO_XLXR          -LET/L11,AND(L10,16)
                 -JNE/L11,16,NO_Y
;                  0 = Posizione asse Y con origine  fatta
;                250 = Posizione asse Y per ricerca MARKER
                 -TSK/OMOY:0,250
                 -JMP/DO_Y
NO_Y             -EVS/5
DO_Y             -LET/L11,AND(L10,32)
                 -JNE/L11,32,NO_SL
;                0 = Posizione asse SL con origine  fatta
                 -TSK/OMOSL:0
                 -JMP/DO_SL
NO_SL            -EVS/6
DO_SL            -LET/L11,AND(L10,64)
                 -JNE/L11,64,NO_SR
;                0 = Posizione asse SR con origine  fatta
                 -TSK/OMOSR:0
```

```
                -JMP/DO_SR
NO_SR           -EVS/7
DO_SR           -LET/L11,AND(L10,128)
                -JNE/L11,128,NO_LU
;               0 = Posizione asse LU con origine  fatta
                -TSK/OMOLU:0
                -JMP/DO_LU
NO_LU           -EVS/8
DO_LU           -EVW/5,6,7,8,9
                -DIS/0: Origine FATTA SU TUTTI GLI ASSI DESIDERATI
                -RET


PartProgram[OMOZL]
;               Origine ASSE ZL
;
;               L1 = Posizione asse ZL con origine  fatta
;
                -HOM/ZL
                -MOV/ZL,L1
                -EVS/1
                -DIS/1: Origine ZL fatta
                -RET


PartProgram[OMOZR]
;               Origine ASSE ZR
;
;               L1 = Posizione asse ZR con origine  fatta
;
                -HOM/ZR
                -MOV/ZR,L1
                -EVS/2
                -DIS/2:Origine ZR fatta
                -RET


PartProgram[OMOX]
;               Origine VELOCE ASSI X
;               L1 = Posizione asse XR con Origine fatta
;               L2 = Posizione per inizio Origine asse XR
;                       (rispetto al micro di Origine)
;               L3 = Posizione asse XL con Origine fatta
;               L4 = Posizione per inizio Origine asse XL
;                       (rispetto al micro di Origine)
                -EVC/1,2,3,4
                -TSK/OMOXR:L1,L2  ;-550 = Pos. inizio ricerca marker XR
                -TSK/OMOXL:L3,L4; ; 600 = Pos. inizio ricerca marker XL
                -EVW/1,2
                -EVS/9
                -DIS/3: Origine FATTA SU ASSI XR e XR
                -RET


PartProgram[OMOXL]
;               Origine VELOCE ASSE XL
;
;               L1 = Posizione asse con Origine fatta
;               L2 = Posizione per inizio Origine asse XL
;                       (rispetto al micro di Origine)
;
                -SFP/5000,1000,1000
                -HOM/3:XL
```

```
                -DIS/2:>
                -TMS/XL,-2000:5:160,0,NTCH      ;-2000 = posiz. finale
                -DIS/2:NON TOCCATO MICRO XL
                -JMP/FINI
NTCH            -EVS/4
                -EVW/3
                -SFP/7000,1000,1000
                -RAV/1:L10,XL
                -MOV/XL,L2+L10
                -SFP/5000,1000,1000
                -HOM/17:XL              ; Origine su Marker dir. positiva
                -MOV/XL,L1              ;-500
                -DIS/4: Origine XL fatta
                -EVS/1
FINI            -
                -RET



PartProgram[OMOXR]
;               Origine VELOCE ASSE XR
;
;               L1 = Posizione asse con Origine fatta
;               L2 = Posizione per inizio Origine asse XR
;                       (rispetto al micro di Origine)
;
                -SFP/5000,1000,1000
                -HOM/3:XR
                -DIS/3:>
                -TMS/XR,2000:5:159,0,NTCH       ;2000 = posiz. finale
                -DIS/3:NON TOCCATO MICRO XR
                -JMP/FINI
NTCH    -EVW/4
                -SFP/7000,1000,1000
                -RAV/1:L10,XR
                -MOV/XR,L2+L10
                -SFP/5000,1000,1000
                -HOM/17:XR              ; Origine su Marker dir. positiva
                -MOV/XR,L1              ;500
                -EVS/3
                -DIS/3: Origine XR fatta
                -EVS/2
FINI            -
                -RET


PartProgram[OMOY]
;               Origine VELOCE ASSE Y
;
;               L1 = Posizione asse con Origine fatta
;               L2 = Posizione per inizio Origine asse Y
;                       (rispetto al micro di Origine)
;
                -SFP/5000,1000,1000
                -HOM/3:Y
                -DIS/1:>
                -TMS/Y,-2000:5:158,0,NTCH       ;-2000 = posiz. finale
                -DIS/1:NON TOCCATO MICRO
                -JMP/FINI
NTCH            -SFP/7000,1000,1000
                -RAV/1:L10,Y
                -MOV/Y,L2+L10
```

```
                -SFP/5000,1000,1000
                -HOM/49:Y               ; Origine su Marker dir. negativa
                -MOV/Y,L1
                -EVS/5
                -DIS/5: Origine Y fatta
FINI            -
                -RET



PartProgram[OMOSL]
;               Origine ASSE SL
;
;               L1 = Posizione asse SL con Origine fatta
;
                -OMO/SL
                -MOV/SL,L1
                -EVS/6
                -DIS/6:OMO SL fatta
                -RET


PartProgram[OMOSR]
;               Origine ASSE SR
;
;               L1 = Posizione asse SR con Origine fatta
;
                -HOM/SR
                -MOV/SR,L1
                -EVS/7
                -DIS/7: Origine SR fatta
                -RET


PartProgram[OMOLU]
;               Origine ASSE LU
;
;               L1 = Posizione asse LU con Origine fatta
;
                -HOM/LU
                -MOV/LU,L1
                -EVS/8
                -DIS/8: Origine LU fatta
                -RET
```

Note:

a) I parametri sono trasmessi posizionalmente (c1 in L1, b2 in L2,...c15 in L15). E' possibile trasmettere fino a 32 parametri.

b) Se i parametri trasmessi sono delle variabili locali, rientrano in input al pari program chiamante con i valori assunti nel part program richiamato. Nell'esempio sopra riportato, dopo l'esecuzione dell'istruzione CAL la variabile locale L5 assumerà il valore corrente della variabile L2 del part program pippo.pp.

7.18 TKM: TasK Management

TKM TasK Management

Funzione: Provoca la chiusura del part program indicato

Parametri: **a:b[:c:d] [,e,f][,g]**
dove:

a = Tipo di operazione (0-7)

0 ;ABORT DEL TASK Kill part program
1 ;SOSPENSIONE TASK CON DECELERAZIONE IMMEDIATA
2 ;RICONTINUAZIONE DEL TASK SOSPESO
3 ;SOSPENSIONE TASK A FINE MOVIMENTI AGGREGATI
4 ;RESTART DEL TASK SOSPESO
5 ;SET FEED PROFILE
6 ;CHANGE FEED PROFILE in percentuale 0-100 nel parametro **d**
7 ;CHANGE SPEED in percentuale 0-100 nel parametro **d**
10H: ;Gestione del mandrino come MASCHIATURA
20H: ;Gestione del mandrino come FRESATURA

b = Label,Nome file del task (part program) da chiudere
c = Flag fermata del mandrino e sua ripartenza vedi comando
caso tipo operazione 0-4:
d = Locale o Globale di stato di ritorno 0 se non trovato il TASK 1 se trovato
caso tipo operazione 5,6,7:
d = Velocità vera
caso tipo operazione 5:
e = Accelerazione
f = Decelerazione
g = Locale o Globale di stato di ritorno 0 se non trovato il TASK 1 se trovato

TKM/(0-7):[nomecall][,nomefile.pp]:[flagMandrino]:[[Vel],[Acc],[Dec]]:Locale stato
TKM/(0-7):[nomefileISO.prg]:[flagMandrino][:[Vel],[Acc],[Dec]:]Locale stato
TKM/(0-5):[numerotesta ISO]:[flagMandrino][:[Vel],[Acc],[Dec]]:Locale stato

Esempio:

```
-       -TKM/1:,A_TKMSet.pp::L1
        -TKM/1:,A_TKMWait.pp::L2
        -DIS/10:Trovato,L1
        -DIS/11:Trovato,L2
        -TMM/2000
        -TKM/2:,A_TKMSet.pp::L3
```

-TKM/2:,A_TKMWait.pp::L4

Note:

a = Tipo di operazione (0-7)

Può essere associato in OR alla scelta del tipo operazione (0-7) i seguenti flag
In esadecimale:

10H: ;Gestione del mandrino come MASCHIATURA
20H: ;Gestione del mandrino come FRESATURA

Nella gestione del mandrino come maschiatura, esso viene fermato
o fatto ripartire contemporaneamente al movimento; eccetto con
la sospensione del task a fine movimenti gia' aggregati, dove
il mandrino viene fermato a movimenti finiti

Nella gestione del mandrino come fresatura, esso viene fermato
dopo che il movimento e' fermato; fatto ripartire prima che il
movimento sia rilanciato.

-TKM/5:,PP1.PP::c1[,d1[,e1]]
c1 = Vel regime sulla traiettoria
d1 = Accelerazione sulla traiettoria
e1 = Decelerazione sulla traiettoria

Cambia la velocita' e le accelerazioni in esecuzione, a partire
dai movimenti ancora da aggregare.
I valori permangono sul task sino ad un prossimo cambio con TKM
o con SFP (istruzione che deve stare sul task chiamato)
Il task deve essere riconosciuto da AxesBrain; ossia prima di
usare una TKM il task chiamato deve avere utilizzato almeno
una istruzione del tipo: SFP/0 o movimentazioni.

-TKM/6:,PP1.PP::c1[,d1[,e1]]
c1 = Vel % sulla traiettoria
d1 = Acc % sulla traiettoria
e1 = Dec % sulla traiettoria

Cambia in percentuale la velocita' e le accelerazioni
dal movimento in esecuzione in poi.
I valori permangono sul task sino ad un prossimo cambio con TKM
Il task deve essere riconosciuto da AxesBrain; ossia prima di
usare una TKM il task chiamato deve avere utilizzato almeno
una istruzione del tipo: SFP/0 o movimentazioni.

-TKM/7:,PP1.PP::c1
c1 = rotazione % del mandrino

Cambia in percentuale di rotazione dal mandrino in esecuzione
in poi. Se il task pone in rotazione piu' di un mandrino, viene
comandato solo l'ultimo che era stato messo in rotazione dal
task.
I valori permangono sul task sino ad un prossimo cambio con TKM
Se sono presenti cambi in percentuale di rotazione,
(istruzione CAP/56:S1,10) esse agiscono solo sul mandrino
comandato in modo diretto, ma ad un seguente comando SPD presente
sul task chiamato, viene riutillizzato il valore caricato con TKM
Il task deve essere riconosciuto da AxesBrain; ossia prima di
usare una TKM il task chiamato deve avere utilizzato almeno
una istruzione del tipo: SFP/0 o movimentazioni.

7.19 DIM : DIMension

DIM DIMension

Funzione: Dimensiona il numero di variabile L di un part pogram

Parametri: **a**

dove:

a = Numero variabili Locali del part program dimensionate

Esempio:

```
-DIM/100        ; Dimensiono a 100 il numero delle variabili Locali per il contesto del programma
```

Note:

a) I parametri alla creazione di un part program sono configurati con il valore definito nella voce **"NumeroLocali"** del file di configurazione "**sistema.txt**", sezione "**[ParametriGenerali]".**

CAPITOLO 8

8 Istruzioni di movimentazione

In questo capitolo vengono descritte le istruzioni di programma relative alla gestione degli assi.

Le istruzioni relative alla gestione degli assi, descritte in modo dettagliato nelle pagine seguenti, sono:

.1-HOM(OMO)	Origine di un asse (HOMing)
.2-MOV	Movimento interpolato linearmente di un gruppo di assi (MOVe)
.3-CIR	Movimento interpolato circolare o ellittico in senso orario di un gruppo di assi (CIrcular Right)
.4-CIL	Movimento interpolato circolare o ellittico in senso antiorario di un gruppo di assi (CIrcular Left)
.5-CRR	Movimento interpolato circolare o ellittico in senso orario di un gruppo di assi con raggio noto (Circular Radius Right)
.6-CRL	Movimento interpolato circolare o ellittico in senso antiorario di un gruppo di assi con raggio noto (Circular Radius Left)
.7-STC	Inizio di una movimentazione in continuo con definizione di percorso (STart Continuous)
.8-HLC	Attesa del completamento della movimentazione in continuo (HaLtContinuous)
.9-ABC	Cancellazione del movimento in continuo (Abort Continuous)
.10-CAP	Cambia i parametri asse (Change Axis Parameter)
.11-HMS	Gestione Master Slave (Handling Master Slave)
.12-HEC	Gestione delle camme (Handling Electronic Cam)
.13-GEI	Legge le informazioni della camma (Get Electronic Cam Information)
.14-CFR	Cambia i parametri dinamici di un asse (Change Feed Rate)
.15-CPL	Cambia il loop di posizione (Change Position Loop)
.16-PRD	Legge le posizioni dell'asse (Position ReaD)
.17-RAV	Legge i parametri dell'asse (Read Axis Value)
.18-RSV	Legge la velocità di un mandrino (Read Speed Value)
.19-SFP	Imposta la velocità del profilo di movimentazione (Set Feed Profile)
.20-SPD	Imposta la velocità di rotazione di un mandrino (SPeeD)
.21-TCH	Movimento con tostatura (TouCH)
.22-TMT	Movimento con ricerca valore di segnale analogico(Test Movement Trasducer)

.23-TMS	Movimento con ricerca valore di sensore digitale (Test Movement Sensor)
.24-TPE	Abilita il tastatore (Touch Probe Enable)
.25-SZP	Definisci la posizione di un set zeri macchina (Set Zero Point)
.26-LZP	Attiva un set di zeri macchina (Load Zero Point)
.27-PIN	(INQ) Flag di incrementale su un asse (Position INcremental)
.28-PAB(ABS)	Flag di assoluto su un asse (Position ABsolute)
.29-MMA	Muove un asse con un movimento manuale (Move Manual Axis)
.30-OPT	Apre un file di punti (Open PoinT file)
.31-MOR	Movimento interpolato linearmente di un gruppo di assi con anticipo (MOv Re)
.32-DCT	Movimento con profondita controllata da tastatore (Deep Control Touch)
.33-DCS	Movimento con profondita controllata da ingresso digitale (Deep Control Sensor)
.34-GRM	Comandi agli assi e mandrini raggruppati (GRoup Management)

8.1 HOM: HOMing (OMO)

HOM(OMO) Origine di un asse (HOMing)

Funzione: azzeramento asse

Parametri: **[b]:a**

dove:

a = numero o nome dell'asse azzerare
b = modalità di azzeramento Default 0

modalità di azzeramento

0 = origine secondo caratterizzazione (default)
1 = origine senza micro solo con marker
2 = origine senza marker solo con micro
3 = origine con azzeramento asse incondizionato senza movimento
4 = origine a "BATTUTA"
8 = test origine su marker per verifica perdita steps encoder
16 = spazio origine caratterizzato in direzione positiva per la ricerca micro
32 = spazio origine caratterizzato con direzione invertita rispetto alla caratterizzazione per la ricerca micro
48 = spazio origine caratterizzato in direzione negativa per la ricerca micro

Quindi:

-HOM/17:A1 significa esecuzione homing su asse A1 con ricerca solo su marker in direzione negativa. Notare che la direzione e' negativa perche' non viene ricercato il micro, che e' virtualmente ricercato in direzione positiva.

-HOM/18:A1 significa esecuzione homing su asse A1 con ricerca solo su micro in direzione positiva.

Esempio:

```
HOM/X1          Azzeramento asse X1
```

Note:

a) Questa istruzione deve essere eseguita per tutti gli assi reali incrementali (funzione caratterizzata) che si intende muovere dopo ogni accensione del sistema, poichè il controllo assume come posizione zero quella corrente all'accensione.

b) Per azzerare gli assi in tempi successivi occorre prevedere n istruzioni **HOM**.

c) L'origine degli assi reali (trasduttore encoder o resolver) viene eseguita come segue:

1 Ricerca veloce del micro di zero, nella direzione (caratterizzata o prevista da una istruzione CAP precedente) e con la velocità caratterizzate; se il micro di zero corrispondente, non è caratterizzato, interviene il messaggio di errore .

2 Riduzione della velocità a quella caratterizzata e rilascio del micro di zero in direzione opposta.

3 Ricerca del "marker", che diventa lo zero elettrico del controllo..

4 Se a questo punto si desidera avere uno zero asse meccanico, occorre verificare il delta fra lo zero elettrico e la posizione di battuta ed introdurlo come offset in caratterizzazione.

e) Prima di azzerare un asse occorre verificare che esso sia nel campo idoneo (positivo se l'azzeramento è in direzione negativa e viceversa); altrimenti l'asse andrebbe in collisione.

f) Se si richiede la funzione **HOM** su di un asse virtuale viene segnalato il messaggio di errore .

g) Se si richiede la funzione HOM su di un asse, l'origine viene effettuata sul 1° marker encoder trovato; .

8.2 MOV: MOVe

MOV MOVe

Funzione: muovere gli assi della macchina "in rapido", alla velocità vettoriale indicata (il vettore è definito dagli assi Specificati) se indicato un output lo pone nella condizione richiesta.

Sintassi 1

Parametri: **[a:]a1,b1,...bn:[c,d]**

dove:

a	=	velocità vettoriale
da a1 a an	=	assi interessati (vedi nota b)
da b1 a b	=	coordinate
c	=	nome o numero di un output digitale
d	=	valore a 1 o O del output digitale

Sintassi 2

Parametri: **[a:]Pb:[c,d]**

dove:

a	=	velocità vettoriale
Pb	=	numero del punto interessato
c	=	nome o numero di un output digitale
d	=	valore a 1 o O del output digitale

Esempio:

MOV/5000:X1,50.5,Y1 ,75	Eseguire un movimento con velocità 5 m/min sul vettore X,Y alle quote indicate.
MOV/5000:X1,50.5,Y1 ,75:12,1	Eseguire un movimento con velocità 5 m/min sul vettore X,Y alle quote indicate, alla fine del movimento viene messo al valore 1 l'output 12
MOV/5000:P10	eseguire un movimento relativo agli assi (e alle coordinate) indicate nel punto P10 con velocità di 5 m/min.

Note:

a) La sintassi 1 permette di eseguire movimenti su indicazione diretta di assi e coordinate. La sintassi 2 permette di eseguire movimenti utilizzando i punti delle matrici.

b) ll vettore può essere formato da un massimo di 6 assi nel caso di utilizzo di punti e 7 assi per indicazione specifica di assi e coordinate.

c) Se il parametro **a** viene omesso, il movimento viene eseguito alla massima velocità possibile, la quale è inoltre utilizzata nel caso il parametro **a** venga specificato, con un valore troppo elevato o con valore zero. Tale velocità è costante lungo tutta la traiettoria se il movimento si effettua su assi reali (cartesiani o non), è invece variabile nel caso di movimenti su assi virtuali perché durante il movimento non è costante la variazione angolare sui due assi reali interessati.

d) Se sulla stessa istruzione viene ripetuto l'asse, in esecuzione compare il messaggio di errore .

e) La richiesta di **MOV** su un asse non azzerato (se caratterizzata) provoca la visualizzazione del messaggio di errore.

h) Con l'istruzione **MOV** è possibile operare sia su assi reali che su assi virtuali, purchè non vi sia abbinamento (reale virtuale) fra gli stessi; in caso contrario interviene la segnalazione di errore (in esecuzione) .

i) Se l'istruzione **MOV** prevede un movimento su più assi, questi si muovono nel seguente modo:
- se si tratta di soli assi reali non cartesiani (per esempio, SCARA), questi si muovono in sincronismo ma non in interpolazione. Praticamente, i vari movimenti iniziano e terminano insieme ma la traiettoria ottenuta non è una retta ma dipende dalla geometria degli assi interessati;
- se si tratta di assi cartesiani (reali o virtuali), questi si muovono in interpolazione lineare.

I) Nel caso di movimento su assi virtuali, va ricordato che l'eventuale asse rotante "**C**" posto nell'estremità del braccio, nel caso non venga menzionato nell'istruzione, avrà o meno una compensazione automatica a seconda che tale situazione sia stata prevista o meno in caratterizzazione (vedi caratterizzazione assi virtuali).

Muove gli assi

Di solito il movimento e' interpolato a meno che non si muovano assi virtuali. In questo caso sono interpolati gli assi fisici e non il movimento virtuale

-MOV/A1,b1,C1,d1,....

A1 C1 .... nomi degli assi
b1 d1 .... posizione finale del movimento

MOVe
Muove gli assi con velocita' eplicita interpolando sempre.
-MOV/v1:A1,b1,C1,d1,....

v1 = velocita' vettoriale del movimento (0 e' il valore
di default; in questo caso il movimento e come MOV
senza switch:)
A1 C1 .... nomi degli assi
b1 d1 .... posizione finale del movimento

8.3 CIR: Circular Interpolation Right CIL: Circular Interpolation Left

CIR **Circular Interpolation Right**
CIL **Circular Interpolation Left**

Funzione: movimento interpolato circolare o ellittico di un gruppo di assi in senso antiorario e antiorario

Parametri: **[a]:b,c,b,d,e,f,e,g,[h,i]**

dove:

a	=	velocità vettoriale
b	=	nome o numero primo asse
c	=	posizione finale del movimento primo asse
d	=	centro del cerchio primo asse
e	=	nome o numero secondo asse
f	=	posizione finale del movimento secondo asse
g	=	centro del cerchio secondo asse
e	=	nome o numero terzo asse (elica)
f	=	posizione finale del movimento terzo asse (elica)

CIrcle Right
Movimento circolare orario (G02)
-CIR/z1:A1,b1,A1,cb1,C1,d1,C1,cd1,

z1 = velocita' vettoriale del cerchio
A1 C1 .... nomi degli assi
b1 d1 .... posizione finale del movimento
cb1 cd1 centro del cerchio

CIrcle Left
Movimento circolare antiorario (G03)
-CIL/z1:A1,b1,A1,cb1,C1,d1,C1,cd1,

z1 = velocita' vettoriale del cerchio
A1 C1 .... nomi degli assi
b1 d1 .... posizione finale del movimento
cb1 cd1 centro del cerchio

Esempio moto rotatorio continuo

```
          -MOV/X1,100,Y1,100
          -STC/0:10
YY        -CIR/X1,100,X1,50,Y1,100,Y1,50
          -JMP/YY

************** ELICA SU X1 Y1 C1 **************
          -DIS/1:-----------------
;       Attesa servo on
INI     -LET/L1,0
          -SFP/10000,100,100
          -MOV/X2,100,Y2,0,Z1,0
          -CAP/0:Z1,128
          -STC/0:10
PIPPO     -CIR/X2,100,X2,50,Y2,0,Y2,0,Z1,10
          -ADD/L1,1
          -JLE/L1,10,PIPPO
          -HLC
          -CAP/0:Z1,-128
          -JMP/INI
```

8.4 CRR: Circular Radius Right CRL: Circular Radius Left

CIR Circular Radius Right
CIL Circular Radius Left

Funzione: movimento interpolato circolare o ellittico di un gruppo di assi in senso antiorario e antiorario con raggio noto

Parametri: **[a]:b,c,b,d,e,f,e,g,[h,i]**

dove:

a	=	velocità vettoriale
b	=	nome o numero primo asse
c	=	posizione finale del movimento primo asse
d	=	raggio del cerchio
e	=	nome o numero secondo asse
f	=	posizione finale del movimento secondo asse
g	=	non usato
e	=	nome o numero terzo asse (elica)
f	=	posizione finale del movimento terzo asse (elica)

CIrcle with Radius Right
Movimento circolare orario con raggio esplicito
-CRR/z1:A1,b1,A1,rb1,C1,d1,C1,rb2

z1 = velocita' vettoriale del cerchio
A1 C1 .... nomi degli assi
b1 d1 .... posizione finale del movimento
rb1 raggio del cerchio
rb2 unused

CIrcle with Radius Left

Movimento circolare antiorario con raggio esplicito

-CRL/z1:A1,b1,A1,rb1,C1,d1,C1,rb2

z1 = velocita' vettoriale del cerchio
A1 C1 .... nomi degli assi
b1 d1 .... posizione finale del movimento
rb1 raggio del cerchio
rb2 unused

8.5 STC: STart Continuous

STC **STart Continuous**

Funzione: abilitare il sistema ad eseguire le varie istruzioni del part program (successive a STC) in modo continuo anziché singolo.

Sintassi 1 (continuo normale o esecuzione di spline)

Parametri: **a:b,c,d,e**

dove:

a = tipo di continuo (switch:)
- 0 = continuo standard (valore di default)
- 1 = spline di tipo 1
- 2 = spline di tipo 2
- 3 = spline di tipo 3
- 5 = caricamento valori speciali
- 10 = movimenti paralleli
- 11 = doppio interpolare (fly motion)
- 21 = numero di enti nel buffer di continuo
- 22 = Percentuale smooth in continuo
- 23 = SPLINE caricamento cicli iterativi massimi
- 24 = SPLINE caricamento percentuale accuratezza velocita'
- 25 = SPAZIO MINIMO dei movimenti in continuo

b = parametro che dipende da a
- a = 0 = parametro per gestione overshoot **>
- a = 1 = parametro per gestione overshoot **>
- a = 2 = parametro per gestione overshoot **>
- a = 3 = parametro per gestione overshoot **>
- a= 5 = HLC wait (>1), nowait (=0) o unused (<0)
- a = 10 = parametro per gestione overshoot **>
- a = 11 = parametro per gestione overshoot **>
- a = 21 = numero di enti nel buffer di continuo (>3)
- a = 22 = Percentuale smooth in continuo
- a = 23 = cicli iterativi massimi in spline (1 - 100)
- a = 24 = percentuale accur. velocita' in spline (>0 =>10)
- a = 25 = SPAZIO MINIMO del movimento in contionuo

**> parametro per gestione overshoot
0 = continuo con velocita' di spigolo uguale a zero
e partenza immediata.
>0 = continuo con velocita' di spigolo maggiore di zero
e partenza immediata.
<0 = continuo con velocita' di spigolo maggiore di zero
e partenza differita.
La partenza del continuo avviene o per fine continuo
(HLC) o per cambio parametro B1 con valore >= 0 o per
fine enti di movimentazione ammessi.

c = parametro che dipende da a
a = 0 non usato
a = 1 modificatore 1 di curvatura (default = 1)
a = 2 modificatore 1 di curvatura (default = 1)
a = 3 modificatore 1 di curvatura (default = 1)
a = 5 = cicli iterativi spline
a = 10 quanti sono i movimenti paralleli (<= 16)
a = 11 spazio di anticipazione del secondo interpolatore
Con valore <0 il secondo interpolatore parte
quando il primo inizia a decelerare
a = 21 non usato
a = 22 non usato
a = 23 non usato
a = 24 non usato
a = 25 non usato

d = parametro che dipende da a
a = 0 non usato
a = 1 modificatore 2 di curvatura (default = 1)
a = 2 modificatore 2 di curvatura (default = 1)
a = 3 modificatore 2 di curvatura (default = 1)
za = 5 = Numero enti del buffer di continuo
a = 10 non usato
a = 11 non usato
a = 21 non usato
a = 22 non usato
a = 23 non usato
a = 24 non usato
a = 25 non usato

e = parametro che dipende da a

a = 0 non usato
a = 1 modificatore 3 di curvatura (default = 1)
a = 2 modificatore 3 di curvatura (default = 1)
a = 3 modificatore 3 di curvatura (default = 1)
a = 5 = Percentuale smooth in continuo ***>
a = 6 non usato
a = 10 non usato
a = 11 non usato
a = 21 non usato
a = 22 non usato
a = 23 non usato
a = 24 non usato
a = 25 non usato

***> Percentuale smooth
<=0 Il parametro non agisce
>0 Percentuale di guadagno di tempo nel continuo per tratti piccoli ove non si accelera se il risparmio di tempo non supera questa percentuale (tratto senza cuspide tra vel.iniziale e vel.finale).

SPLINE

Gli assi dei movimenti spline devono sempre essere gli stessi, non importa l'ordine, ma non si possono omettere anche se la coordinata non cambia.

Spline di tipo 1 :
La curva e' tangente alla corda sottesa all'inizio ed alla fine alla corda successiva

Spline di tipo 2
La curva ha delle tangenti che sono calcolate utilizzando tre punti

Spline di tipo 3
La curva e' simile alla precedente, ma le tangenti sono calcolate utilizzando 5 punti

Esempio 1:

```
-STC/10
-MOV/X1,100,Y1,100      ;Eseguire i due movimenti indicati in modo continuo con partenza imme-
-MOV/X1,200             ;diata dei movimenti.
-TIM/0.5
...
...
-HLC
```

Esempio 2:

```
-STC/2:-10,0.1          ;Eseguire i 3 movimenti indicati con spline di tipo 2 sul profilo con partenza
-MOV/X1,100,Y1,100      ;del primo movimento non appena viene schedulata l'istruzione HLC.
-MOV/X1,150,Y1,150
-MOV/X1,200,Y1,150
..
-HLC
```

Note relative alla sintassi 1:

a) Se il parametro **a** viene omesso, per default assume valore 0 (continuo normale); in tale caso non ha significato l'eventuale parametro **c** (indicato o meno).

b) Se il parametro **c** viene omesso (in tale caso deve essere necessariamente essere omesso anche **d**), per default assume valore **1**.

c) La funzione continuo (per tutti i tipi), rimane valida sino a che non viene annullata da una istruzione HLC (Haft Continuous) o da una istruzione ABC (Abort Continuous).

d) Le istruzioni successive a **STC** e inerenti i movimenti (**MOV, ACC**. ecc.) vengono eseguite nella progressione indicata, mentre le istruzioni di altro tipo (**LET,JMP**, ecc.) pur rispettando la progressione con cui sono indicate vengono eseguite immediatamente; nell'esempio sopracitato l'eccitazione dell'output 1, la pausa di 0,5 secondi e la diseccitazione dell'output avvengono non appena parte il primo movimento di X1.

e) L'utilizzo del parametro **d**, che può essere omesso e che viene accettato solo con parametro "a" uguale a zero, ha significato quando nei movimenti da eseguire in continuo vi sono delle TMS, TMT o TCH (movimenti che possono essere interrotti in funzione di situazioni esterne).
In questi casi, per utilizzare correttamente questo parametro, osservare quanto segue:

- il parametro **c** non ha alcun significato però non è omettibile;
- vengono prese in considerazione 5 words consecutive a partire da quella relativa al bit indicato nel parametro; tali words vengono azzerate automaticamente ad ogni inizio esecuzione della STC;
- nella prima word vengono memorizzate due cose e precisamente:
  - sul primo bit viene memorizzato lo stato dell'ultimo movimento eseguito; il bit assume valore 0 se il movimento è terminato regolarmente mentre assume valore 1 quando il movimento è stato interrotto perché **è stata toccata la parete";**
  - sul secondo bit è memorizzata la differenza fra movimenti "aggregati" al continuo e movimenti "eseguiti"; il bit assume valore 0 quando la quantità di movimenti eseguiti è inferiore a quella di movimenti aggregati; assume valore 1 quando queste due quantità sono uguali;
- nella seconda e terza word, viene memorizzata la quantità di movimenti aggregati al continuo;
- nella quarta e quinta word, viene memorizzata la quantità di movimenti eseguiti;
- poiché la memorizzazione sulle words avviene a livello di bits, è necessario in certi casi eseguire l'istruzione BPI per trasferire il valore da word ad una variabile del GP (locale, globale, indiretta); per esempio BPI su 16 bit per conoscere il valore memorizzato nella seconda più terza word o nella quarta più quinta word;
- è facoltà del programmatore testare queste words prima e/o dopo le varie istruzioni di movimento;

f) Sebbene non ci siano limiti alla quantità di task con continui in esecuzione che possono intervenire in un dato istante, il numero di enti "movimenti" che possono essere aggregati (precalcolati) è:

- 30 nell'interpolazione circolare (CIR,CIL);
- 60 negli altri casi.

Se il numero di enti di uno o più continui è superiore a questi limiti, gli eccedenti vengono accodati e calcolati mano a mano che si liberano gli enti già eseguiti.

g) Se il calcolo degli enti risulta più lento dei movimenti (come nel caso dei movimenti molto brevi), viene eseguita una decelerazione finale (al limite con arresto degli assi). I movimenti vengono ripresi non appena il calcolo aggrega nuovi enti.

h) Per tutti i movimenti eseguiti all'interno di un continuo (fra istruzione **STC** e istruzione **HLC** o **ABC**) non viene attivato il controllo di posizione mentre rimangono sempre attivi sia il controllo di collisione che il controllo blocco conta; dopo ogni overshoot gli assi vengono comunque richiamati in posizione "tirati in asservimento" cioè richiamati dall'asservimento a cui è stata data la massima tensione di riferimento.

i) In tutti i casi di continuo, se al parametro **b** viene assegnato valore 0, si ottiene un profilo seguente:

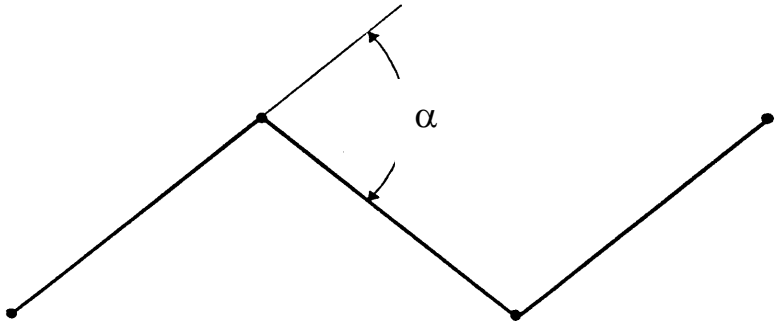
- per ogni punto del profilo vi è fermata a fine di ogni movimento; praticamente i vari movimenti vengono effettuati come se non fosse attivo il continuo con l'unica eccezione che non vengono attivati i tempi di **position time** e **wait cycle**;
- la congiunzione fra i vari punti del profilo è sempre una retta nel caso di continuo normale (parametro **a** = 0); è un arco più o meno schiacciato nel caso di spline (parametro **a** > di 0) in funzione del valore del parametro **c**.

I) In tutti i casi di continuo, se al parametro **b** viene assegnato un valore diverso da 0, si ottiene un profilo seguente:

- il passaggio su ogni punto del profilo (senza fermata) è sempre garantito;
- non vi è mai arresto su ogni punto del profilo, ma solo un rallentamento della velocità; tale rallentamento è più o meno accentuato in funzione del tipo di continuo e del valore assegnato al coefficiente STC.
- la congiunzione fra i vari punti del profilo può essere una retta o un arco più o meno schiacciato in funzione del valore assegnato al parametro **c**.
- dopo ogni punto del profilo può esserci o meno overshoot in funzione del tipo di continuo e del valore assegnato al parametro **c**.

Le note successive **m,n,o,** descrivono il profilo eseguito con i vari tipi di continuo.

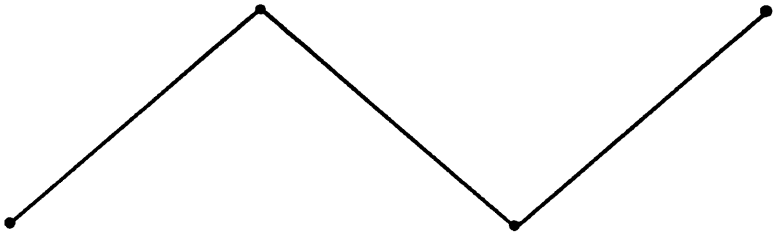
I vari tipi di continuo, si riferiscono al profilo teorico seguente:



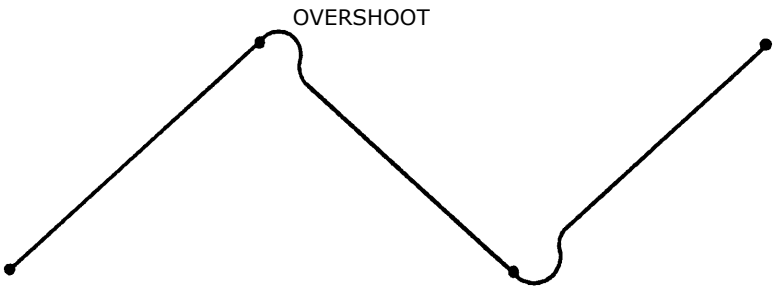
m) Nel caso di continuo normale (parametro **a**= 0), vi è sempre overshoot dopo ogni punto del profilo. La velocità di overshoot dopo ogni movimento è direttamente proporzionale alla accelerazione ed al coefficiente STC e maggiore (con incremento) o minore (con decremento) in modo non proporzionale rispetto all'angolo α (cambio di direzione fra 2 movimenti successivi). Poichè la massima velocità di overshoot non può comunque superare la massima velocità dell'asse, un valore grande di STC è significativo solamente con un valore piccolo di accelerazione; viceversa con una grande accelerazione ha significato un piccolo valore di STC (se questo valore è grande, viene comunque forzato al massimo ammissibile).

Esempi di profilo con continuo normale

Profilo pratico con coefficiente STC (parametro **b**) = 0



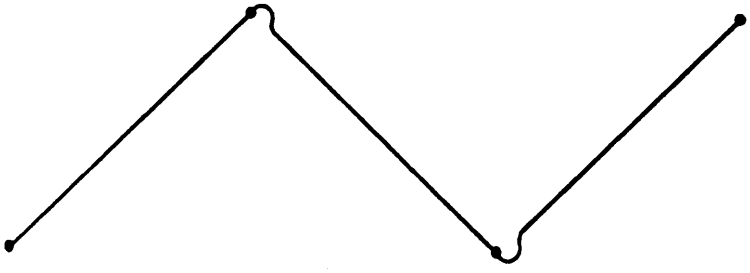
Profilo pratico con coefficiente STC diverso da 0



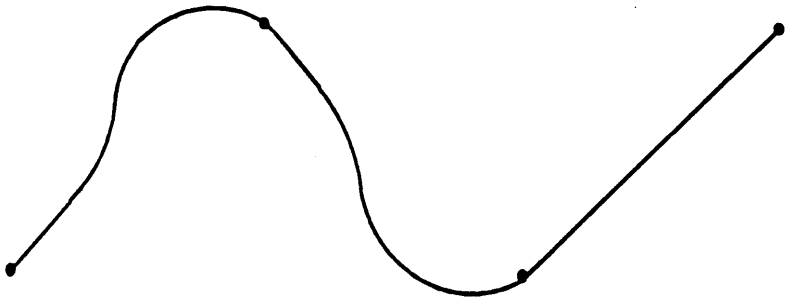
n) Nel caso di spline di tipo 1 (parametro **a** = 1), l'overshoot dopo ogni punto del profilo è nettamente inferiore a quello relativo al continuo normale; praticamente è nullo se si assegna valore 1 al parametro **c**; varia da 0 ad un valore crescente (comunque sempre basso) più il valore del parametro **c** si avvicina allo 0.
Anche per questo tipo di continuo, si ha una variazione di velocità lungo il profilo, per essa è molto meno evidente rispetto al continuo normale; naturalmente la velocità sul profilo sarà tanto più uniforme tanto minore è l'angolo a, tanto più il valore del parametro o si avvicina a 1 e tanto maggiore (anche se poco influente) è il valore del coefficiente STC (parametro **b**).

Esempi di profilo con spline di tipo 1

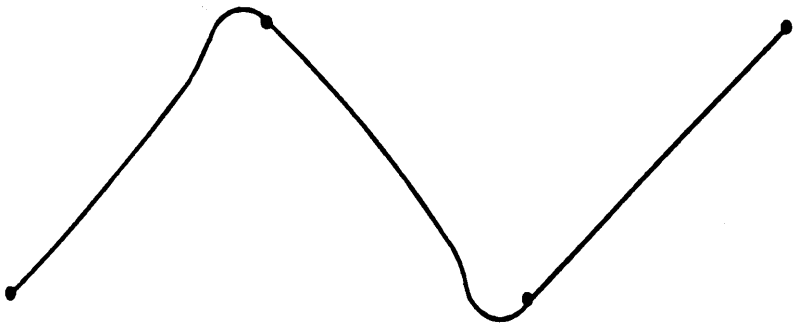
OVERSHOOT PICCOLO



Profilo ottenuto con parametro 0=0 e coefficiente STC diverso da 0; se il coefficiente STC è 0, non vi sono naturalmente gli overshoot.



Profilo ottenuto con parametro **c**= i (con coefficiente STC diverso da 0)

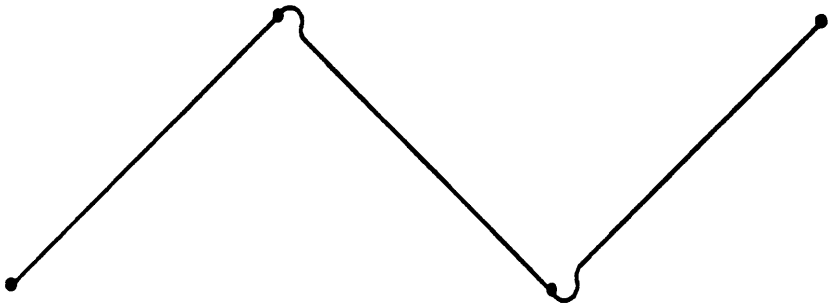


Profilo ottenuto con parametro **c** compreso fra 0 e 1 (coefficiente STC diverso da 0)

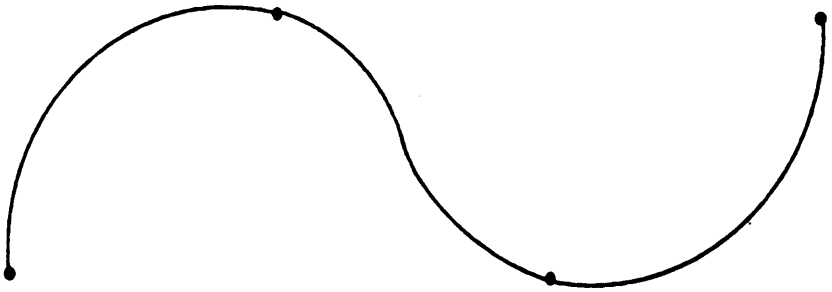
o) Nel caso di spline di tipo 2 (parametro **a** = 2) e di tipo 3 (parametro **a** = 3) si ha un comportamento simile alla spline di tipo 1 ma con la differenza che per la spline di tipo 2 il precalcolo sul profilo viene tatto prendendo in considerazione 3 punti del profilo mentre per la spline di tipo 3 i punti presi in considerazione sono 5.

Esempi di profilo con spline di tipo 2 e 3

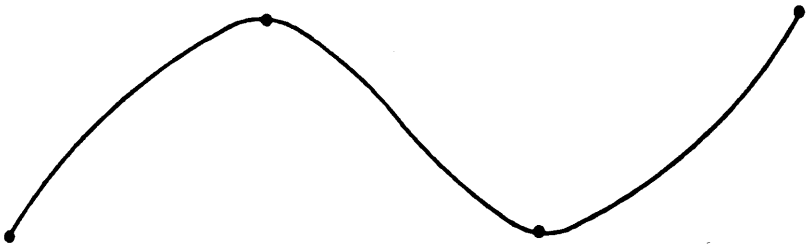
OVERSHOOT PICCOLO



Profilo ottenuto con parametro **c** = 0 e coefficiente STC diverso da 0, se il coefficiente STC è 0, non vi sono naturalmente gli overshoot.



Profilo ottenuto con parametro **c** =1 (coefficiente STC diverso da 0)



Profilo ottenuto con parametro **c** compreso fra 0 e 1 (coefficiente STC diverso da 0).

p) Nel caso delle spline, quando un movimento riguarda più assi, occorre ripetere tutti gli assi anche per i movimenti successivi; se per esempio un asse non è interessato al movimento, occorre ripeterlo ribadendo la coordinata precedente; in caso contrario subentra la segnalazione di errore.

Esempio

```
-STC/2: 10,1
-MOV/X1,100,Y1,100
-MOV/X1,150,Y1,100          ; asse Y1 ribadito con coordinata precedente.
```

q) In base alle considerazioni viste, l'utilizzo di un tipo o l'altro di continuo è legato a diversi fattori e precisamente:

- uniformità del profilo pratico rispetto al teorico
- uniformità della velocità lungo il profilo
- tempo di esecuzione del profilo.

Le considerazioni che si possono fare sono le seguenti:

- il continuo normale è quasi sempre più veloce come tempo di esecuzione però comporta una maggiore variazione di velocità lungo il profilo con una conseguente maggior sollecitazione dei motori degli assi.
- Il continuo tipo spline comporta una maggiore uniformità come velocità lungo il profilo e quindi una minore sollecitazione sui motori a scapito però della traiettoria pratica rispetto alla teorica e a scapito molte volte del tempo di esecuzione.

8.6 HLC: HaLt Continuous

HLC **HaLt Continuous**

Funzione: disabilitare il modo continuo, di forma tale che i movimenti successivi vengano attuati singolarmente.

Parametri: (**nessuno**)

Esempio:

```
-STC/1
-MOV/X1,100,Y1,50
-MOV/X1,125,Y1,200
-HLC
-MOV/X1,150          ;questo movimento verrà effettuato solamente una volta concluso l'ultimo
                     movimento precedente;
```

Note:

Con l'istruzione **HLC** tutti i movimenti (enti) appartenenti al continuo vengono terminati; il programma riprende con la prima istruzione successiva a **HLC**.

8.7 ABC: ABort Continuous

ABC **ABort Continuous**

Funzione: abortire il continuo

Parametri: (**nessuno**)

Esempio:

```
        -STC/10
        -MOV/100:Z1,400
L1      -JLT/T1,150,L1
        -ABC                    ;Il movimento di Z1 è abortito non appena T1 raggiunge il valore 150.
```

Note:

a) Con l'istruzione **ABC** il continuo viene abortito: gli assi in movimento vengono fermati con decelerazione forzata. Il programma prosegue con l'istruzione successiva a **ABC**

b) Nel caso di due blocchi di programma consecutivi che prevedono movimenti (relativi a più assi) in continuo (STC), bisogna inserire, tra il primo blocco e il successivo, una istruzione di movimento relativo agli assi interessati al primo blocco; se viceversa l'asse interessato è uno solo, questo inserimento non è necessario.

Esempio:

```
1° BLOCCO ——┐ -STC
            | ...
            | ...
            | ...
            | ...
            | -MOV/X1,100
            | -MOV/Y1,150
            | -ABC
            | -MOV/X1,n,Y1,n

2° BLOCCO ——┐ -STC
            | ...
            |
            | ...
            | ...
            | -ABC
```

8.8 CAP: Change Axis Parameter

CAP **Change Axis Parameter**

Funzione: modificare i parametri di caratterizzazione asse.

Sintassi

Parametri: **a:b,c**

dove:

a = numero che individua il parametro di caratterizzazione asse su cui operare
b = asse interessato

c = valore da sostituire a quello caratterizzato

I valori di **a sono:**

a1 = tipo di parametro da cambiare (switch:)
- 0 = Informazioni su asse
- 1 = unused
- 2 = unused
- 3 = unused
- 4 = Velocita' asse
- 5 = unused
- 6 = unused
- 7 = unused
- 8 = unused
- 9 = Offset relativo
- 10 = Over travel positivo
- 11 = Over travel negativo
- 12 = Over travel positivo assoluto
- 13 = Over travel negativo assoluto
- 14 = unused
- 15 = KP (proportional gain)
- 16 = VFF (velocity feed forward)
- 17 = Volt a DAC
- 18 = Volt a DAC assoluto
- 19 = KD (derivative gain)
- 20 = KI (integrative gain)

21 = Max integrative error
22 = Volt offset
23 = KD2 (derivative 2 gain)
24 = AFF (acceleration feed forward)
25 = Collision Error
26 = Servo Error
27 = Position Error
28 = Prescrizione Massima (Volts)
29 = Tempo di latenza per STOP COUNTER (Secondi)
30 = Tensione massima per STOP COUNTER (Volts)
31 = Velocita' massima per STOP COUNTER
32 = KS (smorzatore)
33 = Rampe non lineari PENDENZA ACC (variazione acc al sec)
34 = Rampe non lineari PENDENZA DEC (variazione dec al sec)
35 = Rampe non lineari TEMPO PENDENZA ACC
36 = Rampe non lineari TEMPO PENDENZA DEC

38 = KT (predictive)
39 = Velocita' limite dell'asse o di sistema (SYSTEM)
40 = Accelerazione limite dell'asse o di sistema (SYSTEM)
41 = Decelerazione limite dell'asse o di sistema (SYSTEM)
42 = Cambio quota come con resolver per ENCODER_VIRTUALE
43 = Cambio quota diretta per ENCODER_VIRTUALE
46 = Riduzione velocita' per supero errore

50 = Attiva BUFFER CIRCOLARE
51 = Disattiva o Resetta BUFFER CIRCOLARE
52 = Trigger BUFFER CIRCOLARE

55 = CFR (Change FeedRate %)
56 = CSP (Change SPindle %)
57 = RAPPORTO GIRI MANDRINO / MOTORE (Cambio Gamma Mandrino)
58 = GIRI MAX MANDRINO (Cambio Gamma Mandrino)

61 = Shitf Asse (finale)
63 = Shift per tornio (spostamento da centro R a punta tool)
64 = Shitf Quota (finale)

65 = KF (Friction)
66 = Addolcitore di OVERSHOOT

98 = Gestione Sviluppo
99 = Definizione assi usati nel Part Program (LIKE_HOLD_TKM)

110 = Valore dello scostamento dell'asse fisico di un virtuale
111 = Valore LEN_ARM1_V
112 = Valore LEN_ARM2_V
113 = Valore LEN_ARM3_V

b1 c1 .... valori letti dagli assi
A1 B1 .... nomi degli assi
b1 c1 .... valori cambiati agli assi

Con 35 36 il TEMPO PENDENZA ha il seguente significato:
> 0 Scurve (tempo aggiuntivo alla rampa lineare)
= 0 Lineare
< 0 Sinusoidale
Se l'accelerazione e' sinusoidale la decelerazione puo'
essere lineare o sinusoidale, ma non Scurve; in questo
caso viene forzata la decelerazione sinusoidale.

Con 98 i valori passati hanno i seguenti significati:
Primo asse = asse virtuale programmato su figura sviluppata
Suo valore = raggio di sviluppo (con 0 viene annullata)
Secondo asse = asse rotante fisico da movimentare
Suo valore = offset

```
-CAP/98:V,50,B,0
          |   Offset B
         Raggio di sviluppo
con raggio = 0 chiusura prestazione
```

Con 99 il valore passato per l'asse ha questi significati:
> 0 l'asse e' definito come asse da usare nel PP;
piu' precisamente con TKM LIKE_HOLD questo asse e'
riportato in posizione di fermata a prescindere che
sia mosso o no.
<= 0 l'asse viene tolto come asse del PP.
N.B. L'istruzione CAP/99: deve essere data nel
Part Program su cui agira' l'istruzione TKM

INFO
CAP/0:ABS_MOV,1 Attiva sul task i movimenti dall'Origine Assoluta
CAP/0:ABS_MOV,-1 Rimette sul task i movimenti dall'Origine Relativa attuale
Con un solo asse (ABS_MOV) avente il parametro >0 attiva
sul task il movimento dall'origine assoluta anche in

presenza di offset relativi; <=0 disattiva questa prestazione.

CAP/0:X1,4 Forza lo stato di OMO fatta su asse X1 senza azzerare la quota
CAP/0:X1,-4 Toglie lo stato di OMO fatta su asse X1
CAP/0:X1,128 Attiva l'incrementale su asse X1
CAP/0:X1,-128 Toglie l'incrementale da asse X1
CAP/0:X1,256 Attiva l'integrativo con movimento su asse X1
CAP/0:X1,-256 Toglie l'integrativo con movimento su asse X1
CAP/0:X1,512 Forza lo stato di asse bloccato su asse X1
CAP/0:X1,-512 Toglie lo stato di asse bloccato su asse X1
CAP/0:X1,4096 Attiva la correzione su asse X1 (se la tabella e' caricata)
CAP/0:X1,-4096 Toglie la correzione su asse X1
CAP/0:X1,8192 Attiva la gestione su asse X1 di "COLLISION ERROR"
CAP/0:X1,-8192 Toglie la gestione su asse X1 di "COLLISION ERROR"
CAP/0:X1,16384 Attiva la gestione su asse X1 di "SERVO ERROR"
CAP/0:X1,-16384 Toglie la gestione su asse X1 di "SERVO ERROR"
CAP/0:X1,32768 Attiva la gestione su asse X1 di "POSITION ERROR"
CAP/0:X1,-32768 Toglie la gestione su asse X1 di "POSITION ERROR"
CAP/0:X1,65536 Attiva la gestione su asse X1 di "STOP COUNTER ERROR"
CAP/0:X1,-65536 Toglie la gestione su asse X1 di "STOP COUNTER ERROR"
CAP/0:X1,131072 Attiva la gestione su asse X1 di "DRIVER_OK ERROR"
CAP/0:X1,-131072 Toglie la gestione su asse X1 di "DRIVER_OK ERROR"
CAP/0:YM,262144 Scambia un master GANTRY caratterizzato con slave
CAP/0:YM,-262144 Ripristina come da caratterizzazione il GANTRY

INFO supplementare (100)
CAP/100:X1,1 Attiva la gestione del KP in modo SQUARE dell'asse X1
CAP/100:X1,-1 Mette la gestione del KP in modo LINEARE dell'asse X1
CAP/100:X1,2 Attiva la gestione "mediata" di KD dell'asse X1 (4 campionature)
CAP/100:X1,-2 Mette la gestione non "mediata" di KD dell'asse X1 (1 campionatura)
CAP/100:X1,4 Corregge la quota con il tempo di campionatura dell'asse X1
CAP/100:X1,-4 Non corregge la quota letta dell'asse X1
CAP/100:X1,8 Attiva la gestione del KI in modo SQUARE dell'asse X1
CAP/100:X1,-8 Mette la gestione del KI in modo LINEARE dell'asse X1
CAP/100:X1,16 Disabilita l'anticollisione dell'asse X1
CAP/100:X1,-16 Abilita l'anticollisione caratterizzata dell'asse X1
CAP/100:X1,32 Se l'asse e' ROLLOVER UNSIGNED si muove a minor spazio
CAP/100:X1,-32 Se l'asse e' ROLLOVER UNSIGNED si muove a spazio comandato
CAP/100:X1,64 Mette l'accelerazione di tipo S-CURVE
CAP/100:X1,-64 Toglie l'accelerazione di tipo S-CURVE
CAP/100:X1,128 Mette la decelerazione di tipo S-CURVE
CAP/100:X1,-128 Toglie la decelerazione di tipo S-CURVE
CAP/100:X1,256 Esclude l'asse (ASSE FUORI SERVIZIO)
CAP/100:X1,-256 Reinclude l'asse (ASSE IN SERVIZIO)

CAP/100:X1,512 Definisce l'asse come NORMALMENTE FRENATO
CAP/100:X1,-512 Toglie la funzione NORMALMENTE FRENATO
CAP/100:YM,1024 Definisce il gantry come gantry rigido
CAP/100:YM,-1024 Rimette il gantry come "non rigido" se caratterizzato
CAP/100:YM,2048 Disabilita il sistema gantry (solo su asse master)
CAP/100:YM,-2048 Riabilita il sistema gantry (solo su asse master)
CAP/100:YM,4096 Permette il movimento anche senza OMO
CAP/100:YM,-4096 Riabilita il test su OMO fatta (se e' caratterizzato)
CAP/100:YM,65536 (0x10000) Definisce l'asse in fase di plottatura
CAP/100:YM,-65536 (-0x10000) Toglie lo stato di plottatura
CAP/100:YM,131072 (0x20000) Forza lo stato marker vero (se omo fatta) per debug
CAP/100:YM,-131072 (-0x20000)Toglie lo stato marker vero (se omo fatta) per debug
CAP/100:X1,262144 (0x40000) Disabilita la gestione MOV_SWITCH
CAP/100:X1,-262144 (-0x40000) Riabilita la gestione MOV_SWITCH (se caratterizzata)
CAP/100:X1,0x80000 Altro modo per eseguire HOM/3: (toglie MARKER_VERO)
CAP/100:X1,-0x80000 Riporta lo zero sul MARKER (mette MARKER_VERO)
CAP/100:X1,0x100000 Con ROLLOVER_NEAR il movimento e' concorde a CIR CIL
CAP/100:X1,-0x100000 Tolta la condizione di concordanza a CIR CIL
CAP/100:X1,0x200000 Con asse mandrino il comando e con solo VFF = 100
CAP/100:X1,-0x200000 Tolta la condizione di solo VFF all'asse mandrino
CAP/100:X1,0x800000 Con asse norm. frenato fa ripresa SERVO ERROR (1 VOLTA)
CAP/100:X1,0x2000000 Con asse norm. frenato continua ripresa di SERVO ERROR
CAP/100:X1,-0x2000000 Tolta la ripresa continua di SERVO ERROR
CAP/100:CU,0x8000000 ASSE LINMOV con SERVO attivo in norm. frenato

NOTA se l'asse che viene messo o tolto dalla fase di plottatura e'
il master di un sistema GANTRY, anche sull'asse slave viene
eseguita la funzione (CAP/100:YM,0x10000)

Per alcuni valori di caratterizzazione dando come CODICE
il valore corrispondente in OR con 80H viene anche modificato
il valore su statica; ossia il valore permane anche dopo RESET
I codici sono:

(10) (138) OVER TRAVEL POSITIVO
(11) (139) OVER TRAVEL NEGATIVO
(12) (140) OVER TRAVEL POSITIVO ASSOLUTO
(13) (141) OVER TRAVEL NEGATIVO ASSOLUTO
(15) (143) KP
(19) (147) KD
(23) (151) KD2
(20) (148) KI
(16) (144) VFF
(24) (152) AFF
(21) (149) MAX_ERR_I

(25) (153) Collision Error
(26) (154) Servo Error
(27) (155) Position Error
(28) (156) Prescrizione Massima (Volts)
(65) (193) KF

(0) (128) INFO per i seguenti sottocodici
(256) MSK_INTEGRATIVO_MOV
(512) MSK_DYNAMIC_OFFSET
(8192) COLLISION ERROR
(26384) SERVO ERROR
(32768) POSITION ERROR
(65536) STOP COUNTER ERROR
(131072) DRIVER_OK_ERROR

(100) (228) INFO supplementare per i seguenti sottocodici
(1) MSK_KP_SQUARE
(2) MSK_KD_MIDDLE
(4) MSK_EVALUATE
(8) MSK_KI_SQUARE
(64) ACCELERAZIONE S-CURVE
(128) DECELERAZIONE S-CURVE
(512) NORMALMENTE FRENATO

CAP/9:X1,50 Mette l'offset relativo su asse X1: viene spostato lo zero rispetto all'offset assoluto di 50 in positivo. (Se l'asse era a zero dopo questa istruzione diventa -50) Siccome viene spostato lo zero rispetto all'offset assoluto, e non rispetto allo zero relativo, ogni cambio non tiene conto di quello precedente. Quindi per tornare a zero assoluto basta dare un cambio con valore 0.

CAP/50:X1,14,X1,17 Abilita il buffer a memorizzare EPSI per X1 e VOLT per X1. In questo caso il valore che segue l'asse e' il codice RAV corrispondente. Si possono bufferizzare da 1 a 8 assi. Sono validi i seguenti codici:
1 = Quote assi reali
2 = Quote assi teoriche
6 = Quote assi reali assolute
7 = Quote assi teoriche assolute
14 = Errore d'interpolazione

17 = Volt a DAC
18 = Volt a DAC assoluto
48 = Tempo di campionatura
49 = Tempo usato in campionatura
90 = Speed hardware (P.E. digitale passiva)
91 = Current hardware (P.E. digitale passiva)

CAP/51:X1,0 Disabilita la memorizzazione su buffer
L'asse non e' usato.
Se il valore e' >= 0 la memorizzazione del buffer viene fermata (questo avviene naturalmente con il buffer pieno)
Se il valore e' < 0 l'area del buffer viene azzerata

CAP/52:X1,1,X1,100 Attiva il trigger su buffer circolare
L'asse X1 fa partire la memorizzazione quando la quota reale e' >= a 100.
CAP/52: deve precedere CAP/50:, e viene annullata da CAP/51:
La condizione di trigger viene fatta su un solo asse con i seguenti codici:
1 = Quote assi reali
2 = Quote assi teoriche
6 = Quote assi reali assolute
7 = Quote assi teoriche assolute
14 = Errore d'interpolazione
17 = Volt a DAC
18 = Volt a DAC assoluto
90 = Speed hardware (P.E. digitale passiva)
91 = Current hardware (P.E. digitale passiva)
Se il codice e' positivo la condizione e' >= al valore; con codice negativo la condizione e' <= al valore.

CAP/55:X1,10 Permette il change feedrate di un movimento.
Il valore deve essere tra 0 e 100 (percentuale)
Se quando e' dato il comando, l'asse e' fermo, il suo valore e' memorizzato e il movimento che seguira' partira' con il valore di CFR dato con CAP/55:..
A fine movimento il valore di CFR sull'asse e' riportato a 100. Lo stesso succede dopo un reset.

Esempio :

-CAP/16:X1,50 Definire per l'asse X1 il valore di VFF deI 50%.

Esempi di CAP e RAV

Controllo aperto ad un asse per la taratura dell'offset dell' azionamento

```
;       ASSEGNA UNA TENSIONE ALL'ASSE Z
;       E NE VISUALIZZA LA VELCITA'
;
        -TSK/VEL_Z
        -LET/L1,0
AA      -KYB/DAMMI VOLT Z,L1
        -CPL/-1:Z
        -CAP/17:Z,L1
        -JMP/AA
        -RET

PartProgram[VEL_Z]
AA      -RAV/4:L1,Z
        -DIS/1:VELOCITA' Z,L1
;       -TIM/10
        -JMP/AA
        -RET
```

8.9 HMS: Handling Master Slave

HMS **Handling Master Slave**

Funzione: modificare i parametri di caratterizzazione asse riguardo i parametri di master slave.

Sintassi 1

Parametri: **a:b,c,d**

dove:

a = tipo di aggancio slave (switch:)

1	;Asse slave che segue
2	;Asse slave che segue il teorico
4	;Asse slave interpolante ???
16	;Annulla tutti gli slave di un master

b = asse Master interessato

c = asse slave interessato

d = modalità (0, ± valore) di funzionamento:

0	significa disassociare l'asse Slave dal suo Master
± valore	significa associare l'asse Slave al Master definendo per l'asse Slave il valore di following rate.

Esempio 1:

-HMS/X1,0	Dissociare l'asse Slave (caratterizzato) dall'asse Master X1.

Esempio 2:

-HMS/X1,Y1,-50	Associare all'asse Master X1, l'asse Slave Y1 che seguirà il Master in direzione opposta con un valore di Following rate del 50%.

Note

a) Questa sintassi, sempre relativa ad assi Master-Slave, permette sia la modifica di parametri caratterizzati, sia di formare nuove coppie di assi Master-Slave.

b) Devono essere indicati tutti i parametri; in caso contrario subentra il messaggio di errore

c) Se si agisce su una coppia di assi caratterizzata, i nuovi dati si sostituiscono ai dati di caratterizzazione.

d) Se viene indicato +2, significa che l'asse slave pur essendo associato al suo Master, non deve necessariamente seguirlo in tempo reale secondo i parametri definiti; la posizione dell'asse slave non è quindi subordinata alla posizione dell'asse master.
In questo caso se l'asse slave non riesce a seguire il master (l'asse slave arriverà quando potrà), non interviene alcuna segnalazione di errore.
Se viceversa viene indicato -2, significa che l'asse slave deve necessariamente seguire il suo master in tempo reale; in caso contrario interviene l'emergenza COLLISION.

e) L'asse Master può essere o **Reale** o **R.Q.** mentre l'asse Slave può solo essere Reale; segnalazione di errore in tutti gli altri casi.

f) Le istruzioni di movimento relative agli assi Master e Slave possono far parte di un continuo (istruzione STC) però non ne può far parte l'istruzione HMS poichè potrebbe provocare anomalie (in funzione del momento in cui viene eseguita rispetto alle istruzioni di movimento).

c) Se viene richiesta la disassociazione di un asse Slave e questi in quel momento non è associato, interviene il messaggio di essere.

8.10 HEC: Handling Electronic Cam

HEC **Handling Electronic Cam**

Funzione: modificare i parametri di caratterizzazione asse riguardo i parametri gestione delle camme.

Creazione della tabella con:

-HEC/1:a,b[,c]:X,d,Y,d,Z,d...

dove 1: significa creazione tabella

a e' il numero della tabella

b e' lo stato a bit della tabella

```
2      ;MASTER CON QUOTA REALE
4      ;MASTER CON QUOTA REALE STIMATA
8      ; MASTER POSIZIONE TEORICA
0x10   ;CAMMA RIPETITIVA
0x20   ;ASSI SLAVE FERMATI SU ULTIMA POS.
0x40   ;TABELLA CON GESTIONE BUFFERIZZATA
0x80   ;TABELLA NON CANCELLABILE DA RESET
       ;Se la tabella non e' ancora "CHIUSA" viene annullata dal RESET
0x100  ;ATTESA DI LOAD RICORSIVA DI 1 GIRO
```

c sono il massimo numero di righe della tabella

d e' lo stato assoluto (0) incrementale (1) ecc.

```
1      ;ASSE CON TABELLA CARICATA IN INCREM.
2      ;ASSE CHE DEVE SEGUIRE IL MASTER
4      ;RIDUZIONE VELOCITA' PER ERRORE
```

Il primo asse (X) e' il MASTER; gli altri gli SLAVE

Caricamento della tabella con:

-HEC/2:a:X,b,Y,b,Z,b

dove 2: significa caricamento tabella

a e' il numero della tabella

b e' il valore da assegnare ad ogni asse

Gli assi senza variazione possono non esserci
L'ordine tra gli assi non e' importante
L'asse master deve sempre esserci perche' la sua posizione non puo' essere uguale alla precedente; inoltre la direzione deve essere concorde in tutti i punti di tabella (direzione positiva o negativa)

Chiusura dati in tabella con:
-HEC/3:a:X,b

dove 3: significa CHIUSURA tabella HEC
a e' il numero della tabella
b non usato
Questa informazione deve sempre essere data dopo l'ultimo punto, altrimenti l'esecuzione non termina

Start all'elaborazione della tabella con:
-HEC/4:a:X,b

dove 4: significa EXEC tabella HEC
a e' il numero della tabella
b con assi master rollover e' la precorsa da fare all'attacco delle tabella
Questa informazione fa partire l'eleborazione della tabella. Se la tabella e' HEC_WAIT_LOAD l'istruzione puo' essere data anche subito dopo la creazione della tabella; la partenza effettiva avverra' quando sara' caricato almeno un giro di camma.

Attesa fine esecuzione tabella con:
-HEC/5:a:X,b

dove 5: significa ATTESA fine tabella HEC
a e' il numero della tabella
b non usato
Questa istruzione e' paragonabile a HLC

Interruzione della tabella con:
-HEC/6:a:X,b

dove 6: significa INTERRUZIONE tabella HEC
a e' il numero della tabella
b non usato
L'esecuzione della tabella puo' essere interrotta da un RESET o da questa istruzione.
E' bene usare questa istruzione dopo aver fermato il master, per evitare alla fermata posizioni degli slave non su tabella.

Cancellazione della tabella con:
-HEC/7:a:X,b

dove 7: significa CANCELLAZIONE tabella HEC
a e' il numero della tabella
b non usato
Questa istruzione deve essere data ad esecuzione finita, fa rilasciare la memoria utilizzata dalla tabella

Notare che a prescindere dal tipo di asse, il primo punto della tabella e' sempre assoluto

La camma chiusa deve avere il primo e l'ultimo punto della tabella uguale, e la tabella tutta caricata.

Quando l'asse MASTER e' rollover bisogna defire la tabella con questo asse come INCREMENTALE (ASSE_HEC_INCREMENTALE = 1) e quindi caricare i valori del master in incrementale.

Caricamento della tabella con:
-HEC/8:a:b,c,d

dove 8: significa caricamento tabella

a e' il numero della tabella
b e' il numero dei punti da trasferire alla camma
c e' il numero dei valori per ogni punto
d e' il file dove vengono letti i valori da trasferire

Esempio:

```
-HEC/1:1,0x8,1000000:B,1,X,2,Z,2    ;CREA HEC 1
-HEC/8:1:1000000,3,NomeFile     ;carica da memoria  1000000 di punti ognuno con 3 double
-HEC/3:1:B,0                        ;chiude HEC
-HEC/4:1:B,0                        ;esecuzione con attacco in accelerazione
-HEC/5:1:B,0                        ;aspette fine HEC
```

Esempio di un caricamento di una camma chiusa da file

```
;   creazione ed esecuzione di una tabella HEC
;   leggendo i dati da file testo
;
        -RST
        -CAP/100:X,-512 ; Asse sempre abilitato
        -TMM/10
/*
        0x2             ;MASTER CON QUOTA REALE
        0x4             ;MASTER CON QUOTA REALE STIMATA
        0x8             ;MASTER CON QUOTA TEORICA
        0x10    ;CAMMA CHIUSA
        0x20    ;ASSI SLAVE FERMATI SU ULTIMA POS.
        0x80    ;TABELLA NON CANCELLABILE DA RESET
*/
        -MOV/A2,0,X,0 ; precorsa di attacco dell'asse slave
        -HEC/1:1,0x38,36000:CM,1,A2,2,X,2  ;CREA HEC 1

                -LET/L21,0
                -LET/L3,0  ; posizione da leggere
                -LET/L4,0  ; numero di righe
                -LET/L30,0
REP             -TIM/-1
                -LET/L30,L30+1

                -FRD/C:\CAMMA.TXT,L3,L1,L2,L11,L14
                -JEQ/L1,-1,FINI
                -JNE/L3,0,CaricoPunto
PrimoPunto-
                -HEC/2:1:CM,0,A2,L12,X,L13*360          ;LOAD ECM 1 punto
;               -HEC/2:1:CM,0,A2,0,X,1*360      ;LOAD ECM 1 punto
                -LET/L20,L14
                -LET/L4,L4+1
                -LET/L3,L1+1
                -JMP/REP

CaricoPunto-

        -DIS/1:Valore pos. da leggere =,L3
        -DIS/2:Valore pos. letta      =,L1
        -DIS/3:Valore variabili lette =,L2
        -DIS/4:Valore N  =,L11
        -DIS/5:Valore A2 =,L12
        -DIS/6:Valore X =,L13
        -DIS/7:Valore CM =,L14
        -DIS/8:Valore Cunta =,L30

                -JEQ/L1,-1,FINI
                -LET/L4,L4+1
                -LET/L3,L1+1

                -LET/L20,L14-L20
```

```
                -JLE/L20,180,LD1
                -LET/L20,L20-360
                -JMP/LD9
LD1             -JGE/L20,-180,LD9
                -LET/L20,L20+360

LD9             -HEC/2:1:CM,L20,A2,L12,X,L13*360      ;LOAD ECM
                -JLT/L4,358,DOPO
;               -KYB/PIPP,L4
DOPO-
                -LET/L20,L14
                -JMP/REP
FINI    -
                -HEC/3:1:CM,0                 ;chiude HEC

                -MOV/CM,-5,A2,0,X,1*360

        -HEC/4:1:CM,0           ; EXEC HEC
        -TSK/Esegui
        -HEC/5:1:CM,0
        -END


PartProgram[Esegui]
        -INQ/CM
        -STC/1
Loop-
        -MOV/4*60*360:CM,360
        -JMP/Loop

        -END
```

Esempio di un caricamento di una camma chiusa con creazione della funzione

```
;       ESECUZIONE CAMMA ELETTRONICA CON TABELLA
;       BUFFERIZZATA

        -RST
        -CAP/100:X,-512 ; Asse sempre abilitato
        -TMM/10
/*
        0x2             ;MASTER CON QUOTA REALE
        0x4             ;MASTER CON QUOTA REALE STIMATA
        0x8             ;MASTER CON QUOTA TEORICA
        0x10    ;CAMMA CHIUSA
        0x20    ;ASSI SLAVE FERMATI SU ULTIMA POS.
        0x80    ;TABELLA NON CANCELLABILE DA RESET
*/

        -LET/L11,1      ; Ampiezza
        -LET/L3,0
        -LET/L4,1                       ; Delta gradi

        -HEC/1:1,0x38,36000:CM,1,A2,2,X,2  ;CREA HEC 1

        -HEC/2:1:CM,0,A2,(L11*sin(rad(L3))),X,(L11*cos(rad(L3)))*360

RPT     -TIM/-1
        -LET/L3,L3+L4
        -LET/L2,(L11*sin(rad(L3)))
        -LET/L7,(L11*cos(rad(L3)))
        -HEC/2:1:CM,L4,A2,L2,X,L7*360
        -JLT/L3,360,RPT
        -HEC/3:1:CM,0                   ;CHIUDE HEC

        -MOV/CM,-5,A2,(L11*sin(rad(L3))),X,(L11*cos(rad(L3)))*360

        -HEC/4:1:CM,0           ; EXEC HEC
        -TSK/Esegui
        -HEC/5:1:CM,0
        -END

PartProgram[Esegui]
        -INQ/CM
        -STC/1
Loop-
        -MOV/4*60*360:CM,360
        -JMP/Loop

        -END
```

Esempio di un caricamento di una tabella continua bufferizzata

```
;       ESECUZIONE CAMMA ELETTRONICA CON TABELLA
;       BUFFERIZZATA

        -DIM/64
        -SPD/S,100

        -LET/L31,25             ;RAGGIO LENTE
        -LET/L32,100            ;H LENTE
        -LET/L33,90             ;INCREMENTO ANGOLARE
        -LET/L34,0.5            ;INCREMENTO RADIALE (SU 360)
;
        -LET/L35,L31    ;X INIZIALE
        -LET/L36,L32    ;C INIZIALE
        -LET/L39,360/L33-mod(360/L33,1)
        -LET/L40,L31/L34*L39 ;Numero totale punti
        -LET/L41,0
        -LET/L33,360/L39

        -MOV/X,L35,C,L36

        -HEC/1:1,104,6:B,1,X,2,C,2                  ;CREA HEC 1
        -HEC/4:1:B,0                        ;EXEC HEC

        -HEC/2:1:B,0,X,L35,C,L36    ;LOAD HEC (RIGA 1)
RPT     -LET/L41,L41+1
        -LET/L35,L31*(L40-L41)/L40
        -LET/L36,L32/2+cos(rad(360*L41/L39*2))*L32/2*(L40-L41)/L40
        -HEC/2:1:B,L33,X,L35,C,L36  ;LOAD HEC (RIGHE 2 - n)
        -JNE/MOD(L41,L39),0,HH
        -HEC/2:1:B,360    ;LOAD HEC PER ESECUZIONE GIRO COMPLETO
HH  -
        -JLT/L41,L40,RPT
        -HEC/3:1:B,0                        ;CHIUDE HEC
        -SPD/S,0
        -END
```

8.11 GEI: Get Electronic cam Information

GEI **Get Electronic Information**

Funzione: Restituisce nella varaibili L o G le informazioni inerenti alla tabella electronic cam

Parametri: **a:b,c,d,e**

dove:

a = NUMERO DELLA TABELLA HEC
b = Variabile Locale o Globale RIGHE HEC INTRODOTTE IN TABELLA
c = Variabile Locale o Globale RIGA IN ESECUZIONE
d = Variabile Locale o Globale STATO HEC
e = Variabile Locale o Globale VELOCITA' TEORICA DEL MOVIMENTO

Descrizione dello STATO:
bit 1 ; TABELLA ESISTENTE
bit 2 ; TABELLA COMPLETA DI TUTTI I DATI
bit 4 ; TABELLA IN ESECUZIONE
bit 8 ; MOVIMENTO DEL MASTER INCORRETTO
bit 16 ; AUTOINTERRUZIONE FATTA E FINITA

Esempio:

-GEI/1:L1,L2,L3,L4

8.12 CFR: Change Feed Rate

CFR **Change Feed Rate**

Funzione: variare la feedrate di un handle in movimento

Parametri: **a,b,c**

dove:

a = percentuale di feedrate (da 0 a 100)
b = percentuale di accelerazione (da 0 a 100)
c = percentuale di decellerazione (da 0 a 100)

8.13 CPL: Change Position Loop

CPL **Change Position Loop**

Funzione: variare (aprire, chiudere) il loop di posizione sugli assi

Parametri: **[a:]b1,b2,...bn**

dove:

a = tipo di "loop" (switch:)
-3 = Tolto asservimento ma attivato Enable
-2 = Tolto asservimento + Enable
-1 = Tolto asservimento (DAC)
0 = Tolto asservimento (SAC) (Default)
1 = Rimesso asservimento + Enable
2 = Phase axis
3 = Phase axis con controllo spostamento

b1..n = nome dell'asse
Attivando il "loop" -1 (DAC) si puo' comandare l'asse a loop aperto con le istruzione :
-CAP/4:.. , -CAP/17:.. e -CAP/18:.. .
Attivando il "loop" 0 (SAC) si puo' comandare l'asse a loop aperto con l'istruzione -SAC/..

Con il comando CPL/3:Nome Asse la fasatura viene lanciata e viene controllato lo spazio eseguito durante la fasatura.
Se viene superato il valore di "ServoError_motion=" viene tolta potenza e segnalato l'errore di collision. Siccome lo spazio fatto in fasatura puo' essere diverso dal valore voluto di collision questo puo' essere cambiato con CAP/25:Nome Asse,Nuovo Valore.

NB con gli assi FUORI SERVIZIO o LOCK DEFAULT l'istruzione non si puo' usare. Occore prima mettere gli assi in stato normale con le opportune CAP/100:...

Esempio:

```
                -CPL/1:X1        aprire il loop di posizione sull'asse X1

                -DRT/1,PR,X1
A               -CPL/-2:X1
                -TIM/100
                -CPL/1:X1
                -TIM/100
                -JMP/A
```

Nota:

a) Se il parametro **a** viene omesso, per default assume valore 0.

b) Quando viene aperto il loop di posizione, viene automaticamente forzato 0 come valore di Kv sull'asse e vengono disabiIitati i vari controlli asse (servo error, collision, posizione e blocco conta); quando il loop di posizione viene richiuso, vengono ripristinati i valori caratterizzati.

c) Quando viene richiuso il loop di posizione, è indispensabile che la richiesta di movimento asse avvenga con un ritardo di almeno una campionatura rispetto alla richiusura loop.
Esempio:

```
-CPL/1:X1       richiusura loop di posizione
-TIM/2          ritardo
-MOV/X1,        movimento asse
```

d) Se interviene una emergenza dopo che è stato aperto un loop di posizione, eliminando la condizione di emergenza tale loop viene automaticamente richiuso.

e) Le funzioni di Reset richiudono automaticamente eventuali loop di posizione aperti precedentemente.

f) Le richieste di apertura loop di posizione asse su di un asse in movimento, oppure la richiesta di movimento asse con loop di posizione aperto, non vengono accettate e generano il messaggio di anomalia.

8.14 PRD: Position ReaD

PRD **Position ReaD**

Funzione: memorizzare nelle variabili le quote corrispondenti alla posizione degli assi.

Parametri: **[a:]b1,c1,...bn,cn**

dove:

a = switch che determina il tipo di memorizzazione e precisamente:

0 = INFOrmazioni su asse o sistema
1 = Quote assi reali
2 = Quote assi teoriche
3 = Servizio per marker (dopo OMO)
4 = Velocita' asse
5 = Quote assi staticizzate (da TCH TPE)
6 = Quote assi reali assolute
7 = Quote assi teoriche assolute
8 = Quote assi staticizzate assolute
9 = Offset relativo
10 = Over travel positivo
11 = Over travel negativo
12 = Over travel positivo assoluto
13 = Over travel negativo assoluto
14 = Errore d'interpolazione
15 = KP (proportional gain)
16 = VFF (velocity feed forward)
17 = Volt a DAC
18 = Volt a DAC assoluto
19 = KD (derivative gain)
20 = KI (integrative gain)
21 = Max integrative error
22 = Volt offset
23 = KD2 (derivative 2 gain)
24 = AFF (acceleration feed forward)
25 = Collision Error
26 = Servo Error
27 = Position Error

28 = Prescrizione Massima (Volts)
29 = Tempo di latenza per STOP COUNTER
30 = Tensione massima per STOP COUNTER
31 = Velocita' massima per STOP COUNTER
32 = KS (smorzatore)
33 = Rampe non lineari PENDENZA ACC (variazione acc al sec)
34 = Rampe non lineari PENDENZA DEC (variazione dec al sec)
35 = Rampe non lineari TEMPO PENDENZA ACC
36 = Rampe non lineari TEMPO PENDENZA DEC
37 = Valore della correzione ASSE
Quota reale = Quota Encoder + Valore Correzione
38 = KT (predictive)
39 = Velocita' limite dell'asse o di sistema (SYSTEM)
40 = Accelerazione limite dell'asse o di sistema (SYSTEM)
41 = Decelerazione limite dell'asse o di sistema (SYSTEM)
42 = Quota da marker con correzioni
43 = Quota da marker senza correzioni
44 = Distanza reale dalla meta
45 = Distanza teorica (calcolo) dalla meta
46 = Riduzione velocita' per supero errore
47 = Step encoder persi in modo cumulativo
48 = SAMPLE Tempo di campionatura (millisecondi)
49 = SAMPLE Tempo effettivamente usato (millisecondi)
50 = Lettura livelli fatti nel BUFFER CIRCOLARE **1
51 = Lettura livelli liberi del BUFFER CIRCOLARE **2

55 = CFR (Change FeedRate %)
56 = CSP (Change SPindle %)
57 = RAPPORTO GIRI MANDRINO / MOTORE (Cambio Gamma Mandrino)
58 = GIRI MAX MANDRINO (Cambio Gamma Mandrino)

60 = Passo Encoder Caratterizzato
61 = Shitf Asse (corrente)
62 = Velocita' teorica dell'asse
63 = Shift per tornio (spostamento da centro R a punta tool)
64 = Shitf Quota (corrente)

65 = KF (Friction)
66 = Addolcitore di OVERSHOOT
67 = Quote assi teoriche in evoluzione (continuo ...)
68 = Quote assi teoriche assolute in evoluzione (continuo ...)

-CAP/98:V,50,B,0

| Offset B
Raggio di sviluppo
con raggio = 0 chiusura prestazione

99 = assi usati nel PP (1 = DEFINITI 0 = NON DEFINITI)

100 = INFOrmazioni supplementari su asse
101 = INFOrmazioni hardware
PRIMA ELECTRINICS = Status_R
NEWTON = ENCODER_NEWTON
SIX = ENCODER_SIX
SERCOS = ENCODER_SERCOS
YASKAWA = ENCODER_YASKAWA
ETEL = ENCODER_ETEL
UNIVERSALE = POSITION_ACTUAL_UNI
SYSTEM = VERSIONE
102 = INFOrmazioni hardware
PRIMA ELECTRINICS = Device_R
NEWTON = ENCODER_MARKER
SIX = ENCODER_MARKER
SERCOS = ERROR_SERCOS
YASKAWA = ERROR_YASKAWA
ETEL = ERROR_ETEL
UNIVERSALE = DRIVE_FOLLOWING_ERROR_UNI
Asse STEPPER valore PULSE (generatore di frequenza)
SYSTEM = STATUS_PP
103 = INFOrmazioni hardware
PRIMA ELECTRINICS = ErrorCode_R
NEWTON = QUANTI_MARKER
SIX = QUANTI_MARKER
SERCOS = DAC_SERCOS
YASKAWA = DAC_YASKAWA
ETEL = DAC_ETEL
UNIVERSALE = POSITION_SERPOINT
ETHERBOX_FRQ valore FREQUENZA (letta da DAC_OUTPUT)
Asse STEPPER valore FREQUENZA ottenuta con "PULSE"
SYSTEM = NOT USED
104 = INFOrmazioni hardware
ETEL = ENCODER_START_ETEL
UNIVERSALE = LATCH_POSITION_UNI
SYSTEM = NOT USED

110 = Valore dello scostamento dell'asse fisico di un virtuale
111 = Valore LEN_ARM1_V

112 = Valore LEN_ARM2_V
113 = Valore LEN_ARM3_V

**1 il valore ritornato e' 0 o il numero di livelli memorizzati
**2 il valore ritornato e' 0 o il numero di livelli liberi,
se non e' attiva la gestione buffer il valore e' -1

b1 c1 .... valori letti dagli assi
A1 B1 .... nomi degli assi

INFO
valore letto (a bits) :

```
  1H    ; ASSE DI SOLA LETTURA
  2H    ; ASSE CON LOOP DI POSIZIONE APERTO (DAC)
  4H    ; ASSE CON OMO FATTA
  8H    ; ASSE IN SERVO ERROR (ASSE FERMO) **E
 10H    ; ASSE IN COLLISION (ASSE IN MOVIMENTO) **E
 20H    ; ASSE NON IN POSIZIONE **E
 40H    ; ASSE IN BLOCCO CONTA **E
 80H    ; ASSE INCREMENTALE
100H    ; ASSE CON PID INTEGRATIVO SU MOVIMENTO
200H    ; ASSE CHE NON SI DEVE MUOVERE (UNMOVED)
400H    ; ASSE CON ASSERVIMENTO ATTIVO
800H    ; ASSE IN DRIVER KO **E
1000H   ; ASSE CON CORREZIONE ATTIVA
2000H   ; ASSE CON COLLISION ERROR TESTATO
4000H   ; ASSE CON SERVO ERROR TESTATO
8000H   ; ASSE CON POSITION ERROR TESTATO
10000H  ; ASSE CON STOP COUNTER ERROR TESTATO
20000H  ; ASSE CON DRIVER OK ERROR TESTATO
40000H  ; ASSE GANTRY SCAMBIATO IN SLAVE (MASTER)
80000H  ; ASSE CON MOVIMENTO AGGREGATO
100000H ; ASSE CON ANTICOLLISION TESTATA
200000H ; ASSE CON MOVIM. FERMATO DA ANTIC. o CFR/0
400000H ; ASSE IN ANTICOLLISION
800000H ; ASSE IN DEADLOCK
1000000H ; ASSE CON ASSERVIMENTO IN CORRENTE
2000000H ; ASSE IN DRIVER FAULT
4000000H ; ASSE IN OVER TRAVEL POSITIVO (SWITCH FINE CORSA + )
8000000H ; ASSE IN OVER TRAVEL NEGATIVO (SWITCH FINE CORSA - )
```

Se il nome asse e' SYSTEM

```
        valore letto (a bits) :
          1H    ; LETTURA ASSI ATTIVA
          2H    ; ASSERVIMENTI ATTIVI
          4H    ; RUN DI PART PROGRAMA
          8H    ; CYCLE START
         10H    ; CYCLE STOP
         20H    ; EMERGENZA
         40H    ; SISTEMA IN RIPARTENZA
         80H    ; SISTEMA IN RESET
        100H    ; SISTEMA CON ALMENO UN ASSE CON DRIVER KO
        200H    ; SISTEMA CON ASSI IN ANTICOLLISIONE O DEADLOCK
        400H    ; GRUPPI IN EMERGENZA


      ------------------------------------------------------
      INFO supplementare (100)
        valore letto (a bits) :
          1H    ; ASSE CON KP TIPO SQUARE
          2H    ; ASSE CON KD MEDIATA SU 4 CAMPIONATURE
          4H    ; ASSE CON LETTURA CORRETTA DA SAMPLE
          8H    ; ASSE CON KI TIPO SQUARE
         10H    ; ASSE CON ANTICOLLISIONE DISABILITATA
         20H      ; ASSE ROLLOVER UNSIGNED CON MOVIMENTO A MINOR
SPAZIO
         40H    ; ASSE CON ACCELERAZIONE S-CURVE
         80H    ; ASSE CON DECELERAZIONE S-CURVE
        100H    ; ASSE FUORI SERVIZIO
        200H    ; ASSE NORMALMENTE FRENATO
        400H    ; GANTRY CHE NON SEGUE (GANTRY RIGIDO)
        800H    ; GANTRY DISABILITATO
       1000H    ; MOVIMENTO PERMESSO ANCHE SENZA OMO
       2000H    ; ASSE MOSSO IN ACCELERAZIONE
       4000H    ; ASSE MOSSO IN DECELERAZIONE
       8000H    ; ASSE MOSSO A VELOCITA' COSTANTE
      10000H    ; ASSE IN PLOTTATURA
      20000H    ; ASSE CON MARKER VERO
      40000H    ; ASSE CON GESTIONE MOV_SWITCH DISABILITATA
                ;
     100000H    ; ASSE "C" ROLLOVER NEAR CONCORDE CIR CIL
     200000H    ; ASSE MANDRINO COMANDATO SOLO CON VFF **
     400000H    ; ASSE VIRTUALE
     800000H      ; ASSE NORMAL. DISABIL. FRENATO FA RIPRESA SERVO
ERROR
```

1000000H ; ASSE CON FRENO ATTIVO
2000000H ; ASSE NORMAL. DISABIL. CON RIPRESA CONTINUA DEL SERVO ERROR

** In questo caso l'errore di inseguimento e messo uguale all'errore
che l'asse fa in una campionatura (errore della rotazione in UDM asse)

Se il nome asse e' SYSTEM
valore letto (a bits) : MASCHERA DEI GRUPPI DI ASSI IN EMERGENZA

INFO hardware (101)
valore letto : INFOrmazioni hardware (PRIMA ELECTRINICS Status_R)
Asse NEWTON o SIX valore ENCODER letto in impulsi

Se il nome asse e' SYSTEM
valore letto VERSION di AxesBrain

INFO hardware (102)
Asse STEPPER valore PULSE (generatore di frequenza)
Asse NEWTON o SIX ENCODER letto sul MARKER in impulsi

Se il nome asse e' SYSTEM
valore letto CONTATORE stati DISABLE ENABLE che deve
essere 0 (DEBUG)

INFO hardware (103)
Asse STEPPER valore FREQUENZA ottenuta con "PULSE"
Asse NEWTON o SIX QUANTI_MARKER incontrati

Se il nome asse e' SYSTEM
valore letto CONTATORE di HANDLING_MOVE bloccata con
multi CPU o Hyper Treading (con mono CPU deve essere 0)

INFO hardware (104)

Asse ETEL valore ENCODER_START_ETEL "impulsi"

Se il nome asse e' SYSTEM
valore letto CONTATORE di SAMPLE bloccata con
multi CPU o Hyper Treading (con mono CPU deve essere 0)

3 = Servizio per marker
Il valore estratto ha i seguenti significati:
1) Dopo una OMO con micro e marker, la distanza del micro dal marker.
2) Dopo una OMO senza micro e/o senza marker , la posizione dove l'asse era stato azzerato
3) Dopo una OMO/8: o una TCH/8: la posizione del marker; tale valore e' tipicamente zero esatto (a meno che non si sia fatta una OMO su marker precedentemente).
Il non ritrovare zero significa perdita di conteggio.

da b1 a bn = variabili (solo locali o globali) nelle quali si intende memorizzare le quote

da c1 a cn = assi interessati

Esempio:

-PRD/1:L1,X1,G3,Y1 Memorizzare nella variabile locale L1 la quota corrispondente all'asse X1 e nella variabile globale 3 la quota corrispondente all'asse Y1.

Note:

a) Se il parametro **a** viene omesso, per default assume valore 0.

b) Per ogni istruzione è possibile memorizzare un massimo di 8 quote.

c) Le quote memorizzate sono sempre **assolute** ma riferite alla situazione precedente l'istruzione **PRD** (riferite allo zero macchina o ad eventuali set point).

d) Se l'asse interessato è in movimento, viene memorizzata la quota di passaggio, cioè quella esistente al momento dell'esecuzione di **PRD**.

e) La memorizzazione delle quote relative alla piastra trasduttori (parametro **a** = 5), deve essere effettuata dopo una esecuzione dell'istruzione **TCH** o **TPE** cioè dopo che è stata eseguita una strobbatura delle quote da parte del tastatore; in caso contrario verrebbero memorizzate delle quote fasulle cioè le ultime quote staticizzate dal tastatore della piastra.

f) La memorizzazione delle quote teoriche degli assi (parametro **a** = 2) può essere utilizzata per eseguire sul video una plottatura del percorso teorico effettuato dagli assi con *I* senza gli assi che si muovono realmente; è chiaro che se gli assi non si muovono, devono essere disabilitati i seguenti controlli: SERVO ERROR, COLLISION, STOP COUNTER, K.O. POSITION.

8.15 RAV: Read Axis Value

RAV **Read Axis Value**

Funzione: memorizzare nelle variabili le quote corrispondenti i parametri degli assi.

Parametri: **a:b1,c1,...bn,cn**

dove:

a = switch che determina il tipo di memorizzazione e precisamente:

0 = INFOrmazioni su asse o sistema
1 = Quote assi reali
2 = Quote assi teoriche
3 = Servizio per marker (dopo OMO)
4 = Velocita' asse
5 = Quote assi staticizzate (da TCH TPE)
6 = Quote assi reali assolute
7 = Quote assi teoriche assolute
8 = Quote assi staticizzate assolute
9 = Offset relativo
10 = Over travel positivo
11 = Over travel negativo
12 = Over travel positivo assoluto
13 = Over travel negativo assoluto
14 = Errore d'interpolazione
15 = KP (proportional gain)
16 = VFF (velocity feed forward)
17 = Volt a DAC
18 = Volt a DAC assoluto
19 = KD (derivative gain)
20 = KI (integrative gain)
21 = Max integrative error
22 = Volt offset
23 = KD2 (derivative 2 gain)
24 = AFF (acceleration feed forward)
25 = Collision Error
26 = Servo Error
27 = Position Error
28 = Prescrizione Massima (Volts)

29 = Tempo di latenza per STOP COUNTER
30 = Tensione massima per STOP COUNTER
31 = Velocita' massima per STOP COUNTER
32 = KS (smorzatore)
33 = Rampe non lineari PENDENZA ACC (variazione acc al sec)
34 = Rampe non lineari PENDENZA DEC (variazione dec al sec)
35 = Rampe non lineari TEMPO PENDENZA ACC
36 = Rampe non lineari TEMPO PENDENZA DEC
37 = Valore della correzione ASSE
Quota reale = Quota Encoder + Valore Correzione
38 = KT (predictive)
39 = Velocita' limite dell'asse o di sistema (SYSTEM)
40 = Accelerazione limite dell'asse o di sistema (SYSTEM)
41 = Decelerazione limite dell'asse o di sistema (SYSTEM)
42 = Quota da marker con correzioni
43 = Quota da marker senza correzioni
44 = Distanza reale dalla meta
45 = Distanza teorica (calcolo) dalla meta
46 = Riduzione velocita' per supero errore
47 = Step encoder persi in modo cumulativo
48 = SAMPLE Tempo di campionatura (millisecondi)
49 = SAMPLE Tempo effettivamente usato (millisecondi)
50 = Lettura livelli fatti nel BUFFER CIRCOLARE **1
51 = Lettura livelli liberi del BUFFER CIRCOLARE **2

55 = CFR (Change FeedRate %)
56 = CSP (Change SPindle %)
57 = RAPPORTO GIRI MANDRINO / MOTORE (Cambio Gamma Mandrino)
58 = GIRI MAX MANDRINO (Cambio Gamma Mandrino)

60 = Passo Encoder Caratterizzato
61 = Shitf Asse (corrente)
62 = Velocita' teorica dell'asse
63 = Shift per tornio (spostamento da centro R a punta tool)
64 = Shitf Quota (corrente)

65 = KF (Friction)
66 = Addolcitore di OVERSHOOT
67 = Quote assi teoriche in evoluzione (continuo ...)
68 = Quote assi teoriche assolute in evoluzione (continuo ...)

-CAP/98:V,50,B,0
| Offset B
Raggio di sviluppo
con raggio = 0 chiusura prestazione

99 = assi usati nel PP (1 = DEFINITI 0 = NON DEFINITI)

100 = INFOrmazioni supplementari su asse
101 = INFOrmazioni hardware
PRIMA ELECTRINICS = Status_R
NEWTON = ENCODER_NEWTON
SIX = ENCODER_SIX
SERCOS = ENCODER_SERCOS
YASKAWA = ENCODER_YASKAWA
ETEL = ENCODER_ETEL
UNIVERSALE = POSITION_ACTUAL_UNI
SYSTEM = VERSIONE
102 = INFOrmazioni hardware
PRIMA ELECTRINICS = Device_R
NEWTON = ENCODER_MARKER
SIX = ENCODER_MARKER
SERCOS = ERROR_SERCOS
YASKAWA = ERROR_YASKAWA
ETEL = ERROR_ETEL
UNIVERSALE = DRIVE_FOLLOWING_ERROR_UNI
Asse STEPPER valore PULSE (generatore di frequenza)
SYSTEM = STATUS_PP
103 = INFOrmazioni hardware
PRIMA ELECTRINICS = ErrorCode_R
NEWTON = QUANTI_MARKER
SIX = QUANTI_MARKER
SERCOS = DAC_SERCOS
YASKAWA = DAC_YASKAWA
ETEL = DAC_ETEL
UNIVERSALE = POSITION_SERPOINT
ETHERBOX_FRQ valore FREQUENZA (letta da DAC_OUTPUT)
Asse STEPPER valore FREQUENZA ottenuta con "PULSE"
SYSTEM = NOT USED
104 = INFOrmazioni hardware
ETEL = ENCODER_START_ETEL
UNIVERSALE = LATCH_POSITION_UNI
SYSTEM = NOT USED

110 = Valore dello scostamento dell'asse fisico di un virtuale

111 = Valore LEN_ARM1_V
112 = Valore LEN_ARM2_V
113 = Valore LEN_ARM3_V

**1 il valore ritornato e' 0 o il numero di livelli memorizzati
**2 il valore ritornato e' 0 o il numero di livelli liberi,
se non e' attiva la gestione buffer il valore e' -1

b1 c1 .... valori letti dagli assi
A1 B1 .... nomi degli assi

INFO
valore letto (a bits) :

```
  1H    ; ASSE DI SOLA LETTURA
  2H    ; ASSE CON LOOP DI POSIZIONE APERTO (DAC)
  4H    ; ASSE CON OMO FATTA
  8H    ; ASSE IN SERVO ERROR (ASSE FERMO) **E
 10H    ; ASSE IN COLLISION (ASSE IN MOVIMENTO) **E
 20H    ; ASSE NON IN POSIZIONE **E
 40H    ; ASSE IN BLOCCO CONTA **E
 80H    ; ASSE INCREMENTALE
100H    ; ASSE CON PID INTEGRATIVO SU MOVIMENTO
200H    ; ASSE CHE NON SI DEVE MUOVERE (UNMOVED)
400H    ; ASSE CON ASSERVIMENTO ATTIVO
800H    ; ASSE IN DRIVER KO **E
1000H   ; ASSE CON CORREZIONE ATTIVA
2000H   ; ASSE CON COLLISION ERROR TESTATO
4000H   ; ASSE CON SERVO ERROR TESTATO
8000H   ; ASSE CON POSITION ERROR TESTATO
10000H  ; ASSE CON STOP COUNTER ERROR TESTATO
20000H  ; ASSE CON DRIVER OK ERROR TESTATO
40000H  ; ASSE GANTRY SCAMBIATO IN SLAVE (MASTER)
80000H  ; ASSE CON MOVIMENTO AGGREGATO
100000H ; ASSE CON ANTICOLLISION TESTATA
200000H ; ASSE CON MOVIM. FERMATO DA ANTIC. o CFR/0
400000H ; ASSE IN ANTICOLLISION
800000H ; ASSE IN DEADLOCK
1000000H ; ASSE CON ASSERVIMENTO IN CORRENTE
2000000H ; ASSE IN DRIVER FAULT
4000000H ; ASSE IN OVER TRAVEL POSITIVO (SWITCH FINE CORSA + )
8000000H ; ASSE IN OVER TRAVEL NEGATIVO (SWITCH FINE CORSA - )
```

Se il nome asse e' SYSTEM
valore letto (a bits) :
1H ; LETTURA ASSI ATTIVA
2H ; ASSERVIMENTI ATTIVI
4H ; RUN DI PART PROGRAMA
8H ; CYCLE START
10H ; CYCLE STOP
20H ; EMERGENZA
40H ; SISTEMA IN RIPARTENZA
80H ; SISTEMA IN RESET
100H ; SISTEMA CON ALMENO UN ASSE CON DRIVER KO
200H ; SISTEMA CON ASSI IN ANTICOLLISIONE O DEADLOCK
400H ; GRUPPI IN EMERGENZA

INFO supplementare (100)
valore letto (a bits) :
1H ; ASSE CON KP TIPO SQUARE
2H ; ASSE CON KD MEDIATA SU 4 CAMPIONATURE
4H ; ASSE CON LETTURA CORRETTA DA SAMPLE
8H ; ASSE CON KI TIPO SQUARE
10H ; ASSE CON ANTICOLLISIONE DISABILITATA
20H ; ASSE ROLLOVER UNSIGNED CON MOVIMENTO A MINOR SPAZIO
40H ; ASSE CON ACCELERAZIONE S-CURVE
80H ; ASSE CON DECELERAZIONE S-CURVE
100H ; ASSE FUORI SERVIZIO
200H ; ASSE NORMALMENTE FRENATO
400H ; GANTRY CHE NON SEGUE (GANTRY RIGIDO)
800H ; GANTRY DISABILITATO
1000H ; MOVIMENTO PERMESSO ANCHE SENZA OMO
2000H ; ASSE MOSSO IN ACCELERAZIONE
4000H ; ASSE MOSSO IN DECELERAZIONE
8000H ; ASSE MOSSO A VELOCITA' COSTANTE
10000H ; ASSE IN PLOTTATURA
20000H ; ASSE CON MARKER VERO
40000H ; ASSE CON GESTIONE MOV_SWITCH DISABILITATA
;
100000H ; ASSE "C" ROLLOVER NEAR CONCORDE CIR CIL
200000H ; ASSE MANDRINO COMANDATO SOLO CON VFF **
400000H ; ASSE VIRTUALE

800000H ; ASSE NORMAL. DISABIL. FRENATO FA RIPRESA SERVO ERROR
1000000H ; ASSE CON FRENO ATTIVO
2000000H ; ASSE NORMAL. DISABIL. CON RIPRESA CONTINUA DEL SERVO ERROR

** In questo caso l'errore di inseguimento e messo uguale all'errore
che l'asse fa in una campionatura (errore della rotazione in UDM asse)

Se il nome asse e' SYSTEM
valore letto (a bits) : MASCHERA DEI GRUPPI DI ASSI IN EMERGENZA

INFO hardware (101)
valore letto : INFOrmazioni hardware (PRIMA ELECTRINICS Status_R)
Asse NEWTON o SIX valore ENCODER letto in impulsi

Se il nome asse e' SYSTEM
valore letto VERSION di AxesBrain

INFO hardware (102)
Asse STEPPER valore PULSE (generatore di frequenza)
Asse NEWTON o SIX ENCODER letto sul MARKER in impulsi

Se il nome asse e' SYSTEM
valore letto CONTATORE stati DISABLE ENABLE che deve
essere 0 (DEBUG)

INFO hardware (103)
Asse STEPPER valore FREQUENZA ottenuta con "PULSE"
Asse NEWTON o SIX QUANTI_MARKER incontrati

Se il nome asse e' SYSTEM
valore letto CONTATORE di HANDLING_MOVE bloccata con
multi CPU o Hyper Treading (con mono CPU deve essere 0)

INFO hardware (104)
Asse ETEL valore ENCODER_START_ETEL "impulsi"

Se il nome asse e' SYSTEM
valore letto CONTATORE di SAMPLE bloccata con
multi CPU o Hyper Treading (con mono CPU deve essere 0)

3 = Servizio per marker
Il valore estratto ha i seguenti significati:
1) Dopo una OMO con micro e marker, la distanza del micro dal marker.
2) Dopo una OMO senza micro e/o senza marker , la posizione dove l'asse era stato azzerato
3) Dopo una OMO/8: o una TCH/8: la posizione del marker; tale valore e' tipicamente zero esatto (a meno che non si sia fatta una OMO su marker precedentemente).
Il non ritrovare zero significa perdita di conteggio.

da b1 a bn = variabili (solo locali o globali) nelle quali si intende memorizzare le quote

da c1 a cn = assi interessati

Esempio:

-RAV/1:L1,X1,G3,Y1 Memorizzare nella variabile locale L1 la quota corrispondente all'asse X1 e nella variabile globale 3 la quota corrispondente all'asse Y1.

NOTE SULLA GESTIONE PERDITA STEP ENCODER

Con gli assi NEWTON e PCI 2000 si possono recuperare le perdite di steps dell'encoder utilizzando i seguenti campi di caratterizzazione

StepsEncoder=4000
impulsi per giro encoder. usato per aggiustare la posizione se nel giro si perdono steps
MaxStepsLost=100
impulsi per giro encoder che possono essere persi
MarkerSteps=2
Larghezza in impulsi del MARKER

Con 0x200 in Kind_axis_0 (aggiustamento in una sola direzione) il valore >=0 significa in direzione positiva; <0 in direzione negativa.
Se la perdita passo nel giro supera il valore di MaxStepsLost= viene marcato l'errore e tolta sull'asse la condizione di zero fatto.
Con l'asservimento attivo la segnalazione diviene effettiva e permane sino all'esecuzione del primo reset. Una nuova segnalazione di perdita passo eccessiva verra diagnosticata dopo l'azzeramento dell'asse che deve essere rifatto.
Si possono leggere il numero cumulativo di steps persi con l'istruzione RAV/47:L1,X011
In L1 viene caricata la distanza persa/aggiunta sulla posizione encoder

in Unita' Di Misura
Caricando una variabile con RAV/60:G1,Z011 e' possibile avere il numero di steps persi dividendo il valore ottenuto con RAV/47: con questa variabile
Questo valore viene resettato da un azzeramento asse
Utilizzando l'istruzione RAV/102:L1,X011 si puo visualizzare l'ultima posizione encoder del MARKER (in steps encoder)
Utilizzando l'istruzione RAV/103:L1,X011 si puo visualizzare quante volte il MARKER e' stato incontrato. Il conteggio e' resettato da un azzeramento asse.
Anche con la funzione perdita passo non attiva si possono utilizzare le istruzione RAV/102 RAV/103 che attivano la fase di caricamento marker. Con reset questa fase viene eliminata.
E' quindi possibile fare PART-PROGRAM che verificano se il MARKER e' presente e il controllo del numero di steps per giro encoder nonche' la larghezza del marker.

E.G.

```
II                  -RAV/103:L3,A                       ;Leggi Num. MARKER
AA                  -RAV/103:L1,A                       ;Leggi Num. MARKER
                    -JEQ/L1,L3,AA
                    -DIS/4:MARKER TROVATO ,L1
                    -RAV/102:L2,A                       ;Leggi Quota MARKER
BB                  -RAV/103:L3,A                       ;Leggi Num. MARKER
                    -JEQ/L3,L1,BB
                    -DIS/4:MARKER TROVATO ,L3
                    -RAV/102:L4,A                       ;Leggi Quota MARKER
                    -DIS/5:Numero STEPS ,abs(L4-L2)
                    -JMP/II
                    -END
```

Questo part program puo' dare il numero di steps per giro se l'encoder ruota (non troppo velocemente) sempre dalla stessa parte, o la larghezza del marker se l'encoder oscilla a cavallo dello stesso.

8.16 RSV: Read Speed Value

RSV **Read Speed Value**

Funzione: legge i giri dei mandrini (da 1 a 8 mandrini)

Parametri: **a:b1,c1….bn,cn**

dove:

a = tipo di parametro da leggere (switch:)
0 = INFOrmazioni su mandrino
1 = Giri mandrino reali
2 = Giri mandrino teorici
3 = Volts dati al mandrino

INFO
valore letto (a bits) :
1H ; MANDRINO IN ROTAZIONE
2H ; MANDRINO A REGIME

b1..n = variabili (solo locali o globali) nelle quali si intende memorizzare il valore di rotazione

c1..n = mandrini interessati

Esempio:

-RSV/L1,M1 Memorizzare nella variabile locale L1 la quota corrispondente al mandrino M1

8.17 SFP: Set Feed Profile

SFP **Set Feed Profile**

Funzione: definisce un valore generico di feed ,accelerazione e decelerazione da utilizzare sulle istruzioni di movimento che seguono.

Parametri: **a,b,c**

dove:

a = valore massimo di velocita' sul profilo
b = valore massimo di accelerazione sul profilo
c = valore massimo di decelerazione sul profilo

Esempio:

```
-SFP/500
-MOV/X1,100,Y1,100      Tutte queste istruzioni (ad eccezione della MOV/200:X1 300) vengono
-MOV/200:X1,300         eseguite con una feed di 500.
-MOV/X1,400
-MOV/1000:X1 ,500
…
…
```

Note:

a) Ogni funzione **SFP** annulla la precedente e rimane valida fino alla prossima.

b) Per annullare la funzione **SFP** occorre definirla con parametro **a=** zero; le funzioni di Reset annullano eventuali **SFP** attive.

c) I movimenti che seguono l'istruzione **SFP**, vengono eseguiti con il valore di feed minore fra: quello indicato nella istruzione SFP; quello eventualmente indicato nelle istruzioni di movimento (nell'esempio sopra citato la MOV/200:X1,300); e quello possibile in funzione degli assi da muovere (vettore formato da più assi con dati diversi fra loro; spazio da percorrere diverso fra i vari assi, ecc.).

8.18 SPD: SPeeD

SPD **Speed**

Funzione: imposta la velocità di rotazione di un mandrino (SPeeD).

Parametri: **a,b,[c,d]**

dove:

a = numero o nome mandrino
b = valore in giri al minuto da assegnare al mandrino
c = accelerazione in giri al sec2 (0 = max)
d = decelerazione in giri al sec2 (0 = max)

Esempio:

```
-SPD/M1,11000
```

Note:

La rotazione del mandrino e' lanciata con SPD/M1,giri
dove M1 e' il nome simbolico del mandrino e giri sono giri/min.
Ci sono alcuni I/O che gestiscono la rotazione.

```
Number_ENABLE_switch_port="Enable Spindle ZB"        // Output per abilit. mandrino "OUT11"
FalseTrue_logic_ENABLE_switch=1                      // Output per abilitazione mandrino

Number_DRVOK_switch_port=""                          // Input per mandrino OK
FalseTrue_logic_DRVOK_switch=2                       // Input per mandrino OK

Number_STOP_switch_port=""                           // Input per mandrino fermo
FalseTrue_logic_STOP_switch=2                        // Input per mandrino fermo

Number_LOCK_switch_port=""                           // Output per mandrino BLOCCATO
FalseTrue_logic_LOCK_switch=2                        // Output per mandrino BLOCCATO

Number_RUN_switch_port=""                            // Input per mandrino a REGIME
FalseTrue_logic_RUN_switch=2                         // Input per mandrino a REGIME
```

8.19 TCH: TouCH

TCH **TouCH**

Funzione: eseguire un test sul tastatore delle piastre trasduttori durante un movimento

Parametri: **a,b:c:d,e:f**

dove:

a = Asse mosso
b = Posizione finale per NO TOUCH
c = Spazio max di fermata
d = Globale / Locale per staticizzazione quota touch
e = Asse staticizzato
f = Label di salto per touch fatta

Esempio di TCH

```
        -DIS/1:>
        -DIS/2:>
INIZIO-
        -MOV/Y,0
        -TCH/Y,100:10:L1,Y:TOCCATO
        -DIS/1:NON TOCCATO IN ANDATA
        -TMM/2000
        -JMP/RITORNO
TOCCATO -
        -DIS/1:VALORE ,L1
        -TMM/2000
RITORNO         -
        -TCH/Y,0:1:L1,Y:SENTITO
        -DIS/2:NON SENTITO IN RITORNO
        -TMM/2000
        -JMP/INIZIO
SENTITO -
        -DIS/2:VALORE ,L1
        -TMM/2000
        -JMP/INIZIO
```

8.20 TMT: Test Movement Transducer

TMT **Test Movement Transducer**

Funzione: eseguire un test sui trasduttori di forza (input analogici) durante un movimento.

Parametri: **[a]:a1..n,b1..n:c:d,e:f:g**

dove:
a = Velocità vettoriale
a1..n = Numero o nome asse
b1..n = Posizione finale se non tocca
c = Spazio di fermata
d = Numero / Nome trasduttore
e = Valore trasduttore
f = 1 (>= al valore) 0 (<= al valore)
g = Label di salto per toccata fatta

; TMT a Trasduttore . TMT/A1,b1:SpaceStop:C1,Valore:d1:LABEL
A1 = Asse mosso
b1 = Posizione finale se non tocca
SpaceStop = Spazio di fermata
C1 = Numero / Nome trasduttore
Valore = Valore trasduttore
d1 = 1 (>= al valore) 0 (<= al valore)
LABEL = Label di salto per toccata fatta

; TMT a Battuta TMT/A1,b1:Time:A1,Volt:d1:LABEL
A1 = Asse mosso
b1 = Posizione finale se non tocca
SpaceStop = Tempo di battuta in secondi
C1 = Asse testato (uguale ad asse mosso
Valore = Tensione in Volt
d1 = non usato e considerato 1
LABEL = Label di salto per toccata fatta

8.21 TMS: Test Movement Sensor

TMS **Test Movement Sensor**

Funzione: eseguire un test sui sensori (input digitali) durante un movimento

Sintassi 1

Parametri: **[a]:a1..n ,b1..n:c:d,e:f**

dove:

a = Velocità vettoriale
a1..n = Numero o nome asse
b1..n = Posizione finale se non trovato switch
c = Spazio di fermata
d = Numero / Nome del digital input/ Nome del digital output
e = 1 (digital input on) 0 (digital input off)
f = Label di salto per micro trovato

; TMS/A1,b1:SpaceStop:Switch,Tipo:LABEL
A1 = Asse mosso
b1 = Posizione finale se non trovato switch
SpaceStop = Spazio max di fermata
Switch = Numero / Nome del digital input
Tipo = 1 (digital input on) 0 (digital input off)
LABEL = Label di salto per micro trovato

Esempio:

```
-HOM/2:REV041

-TMS/REV041,-720:0.01:I406,0:TrPTO
-LET/L1,-97
-JMP/ErroreBloccoRev

TrPTO-
```

8.22 TPE: Touch Probe Enable

TPE **Touch Probe Enable**

Funzione: eseguire un test sui tastatori delle piastre trasduttori senza effettuare alcun movimento

Parametri: **a1..n,b1..n**

dove:

a1.. n = Globale / Locale per staticizzazione quota touch
b1.. n = Numero o nome asse

Esempio:

```
;                              TPE/G1,X1,G2,Y1,....
```

8.23 SZP: Set Zero Point

SZP **Set Zero Point**

Funzione: consente di definire le origini dei sistemi di riferimento degli assi.

Parametri: **a:b1,c1,..bn,cn**

dove:

a = n° di origine (da 1 a 32)

da b1 a bn = assi interessati

da c1 a cn = coordinate, cioè distanza fra lo zero macchina di ciascun asse e l'origine.

Esempio:

```
SZP/1:X1 ,50,Y1 ,75.43,Z ,-4.3
```

Note:

a) Gli assi interessati possono appartenere anche a più bracci fisici (vedi la nota g dell'istruzione **LZP**).

b) Se i parametri sono immediati e non variabili, l'istruzione **SZP** diventa esecutiva immediatamente dopo [Enter] sia in TEACH che in EDIT (e non occorre eseguire il part program); in caso contrario è necessario eseguire il part program (una sola volta, dopo TEACH o dopo EDIT).

c) Ogni **SZP** annulla la precedente con io stesso numero, anche se gli assi interessati sono diversi.

d) E' conveniente programmare tutte le **SZP** relative al robot raggruppate su uno stesso part-program.

e) Con ogni istruzione **SZP** è possibile operare su un massimo di 7 assi.

8.24 LZP: Load Zero Points

LZP **Load Zero Points**

Funzione: consente di programmare in base alle varie origini definite dall'istruzione SZP

Parametri: **a**

dove:

a = n° dell'origine interessata (da 1 a 32)

Esempio:

```
LZP/1
MOV/X1,-10.5
```

La coordinata X1 risultante sarà 39,5 mm (equivalente a 50 mm dell'istruzione SZP meno 10,5 mm del movimento).

Note:

a) Ogni funzione **LZP** annulla la precedente e rimane valida fino alla prossima.

b) Le quote risultanti dopo l'istruzione **LZP** saranno uguali alla somma algebrica delle coordinate del movimento e quelle dell'origine.

c) **LZP** può essere annullata dal comando di RESET

d) Se non si prevede alcuna **LZP** tutte le coordinate si riferiscono agli zeri macchina (vedere la nota g).

e) Se **LZP** viene introdotta senza la relativa **SZP** compare il messaggio di errore **ZERO POINT NOT SET**.

f) Se dopo una **LZP**/n viene mosso un asse non indicato nella relativa **SZP**/n, questo asse sarà riferito alla **LZP** precedente.

g) Se in una stessa istruzione **SZP** vengono indicati assi non riguardanti lo stesso braccio fisico, possono verificarsi degli inconvenienti durante i lavori multitask (task paralleli), è quindi importante tener presente quanto segue:

- la prima istruzione **LZP** eseguita su uno qualsiasi dei task rende esecutivi tutti i delta relativi agli assi indicati nella **SZP** corrispondente.

- di conseguenza la **LZP** attiva sugli altri task non è quella prevista; sono prevedibili le conseguenze in caso di cambio **LZP**.

Per ciò si consiglia di indicare assi relativi a più bracci fisici sulla stessa SZP solo quando questi verranno movimentati dallo stesso task.

8.25 PIN: Position Incremental(INQ)

PIN(INQ) **Position Incremental(INcremental Quote)**

Funzione: abilitare il sistema ad effettuare gli spostamenti con quote incrementali.

Parametri: **a1,a2,...an**

dove:

da a1 a an = assi che si vogliono programmare in modo incrementale (massimo 16)

Esempio:

```
-PIN/X1,Z2        abilitare il sistema a ricevere quote incrementali sugli assi X1 e Z2
```

Note:

a) I resti delle operazioni di conversione metrica non vengono persi ma considerati nei movimenti successivi.

L'istruzione **PIN** è valida finchè non viene annullata da **PAB** (relativa agli stessi assi) o comando di RESET

8.26 PAB: Position Absolute(ABQ)

PAB(ABQ) **Position Absolute (ABsolute Quote)**

Funzione: disabilitare l'incrementale in modo che i movimenti successivi vengano di nuovo riferiti allo zero di partenza.

Parametri: **a1 ,a2,...an**

dove:

da a1 a an = assi che si desidera riprogrammare in assoluto (massimo 16)

Esempio:

```
-PAB/X1      ;Le coordinate relative all'asse X1 che seguono questa istruzione saranno di nuovo in assoluto.
```

8.27 MMA: Move Manual Axis

MMA **Move Manual Axis**

Funzione: Muove un asse con un movimento manuale (Move Manual Axis)

Parametri: **a,b,c**

dove:

a	=	0 ferma il movimento, 1 lo attiva
b	=	asse interessato
a	=	percentuale di velocita del movimento

Esempio:

```
;
; Gestione JOG
;
        -RAI/L30,AD1      ; potenziometro assi
        -JGE/L30,0,ADAOK
        -LET/L30,-L30
ADAOK-
        -LET/L30,L30/16300*30
        -JLE/L30,30,ADA_1OK
        -LET/L30,30
ADA_1OK-

MM_Z011-
        -IDI/IN6,1,MuoviZ11Piu
        -IDI/IN7,1,MuoviZ11Meno
FermoAsseZ11-
        -JEQ/L31,0,MM_Z021
        -MMA/0:Z011
        -SDO/M1.4,0
        -LET/L31,0
        -JMP/MM_Z021
MuoviZ11Piu-
        -JNE/and(G14011,2),2,MM_Z021
        -JEQ/L31,1,MM_Z021
        -SDO/M1.4,0
        -MMA/1:Z011,L30
        -LET/L31,1
        -JMP/MM_Z021
MuoviZ11Meno-
        -JNE/and(G14011,2),2,MM_Z021
        -JEQ/L31,-1,MM_Z021
        -SDO/M1.4,0
        -MMA/1:Z011,-L30
        -LET/L31,-1
        -JMP/MM_Z021
```

```
MM_Z021-
        -IDI/IN4,1,MuoviZ21Piu
        -IDI/IN5,1,MuoviZ21Meno
FermoAsseZ21-
        -JEQ/L32,0,MM_Z
        -SDO/M1.4,0
        -MMA/0:Z021
        -SDO/M1.4,0
        -LET/L32,0
        -JMP/MM_Z
MuoviZ21Piu-
        -JNE/and(G14021,2),2,MM_Z
        -JEQ/L32,1,MM_Z
        -SDO/M1.4,0
        -MMA/1:Z021,L30
        -LET/L32,1
        -JMP/MM_Z
MuoviZ21Meno-
        -JNE/and(G14021,2),2,MM_Z
        -JEQ/L32,-1,MM_Z
        -SDO/M1.4,0
        -MMA/1:Z021,-L30
        -LET/L32,-1
        -JMP/MM_Z
MM_Z-

MM1-
        -JMP/a
;
; Non in Manuale Fermare gli assi se in moto
;
NoManuale-

NO_Z011-
        -JEQ/L31,0,NO_Z021
        -MMA/0:Z011
        -SDO/M1.4,0
        -LET/L31,0
        -JMP/MM_Z021
NO_Z021-
        -JEQ/L32,0,NO_Z
        -SDO/M1.4,0
        -MMA/0:Z021
        -SDO/M1.4,0
        -LET/L32,0
        -JMP/NO_Z
NO_Z-
```

8.28 OPT: Open PoinT file

OPT **Open PoinT file**

Funzione: Apre un file di punti (Open PoinT file)

Parametri: **a**

dove:

a = Nome file di punti

Esempio:

```
-OPT/Point.pt
```

8.29 MOR: MORe

MOR MORe

Funzione: muovere gli assi della macchina "in rapido", alla velocità vettoriale indicata (il vettore è definito dagli assi Specificati) se indicato un output lo pone nella condizione richiesta, la segnalazione di completamento della movimentazione è anticipata al raggiungimento della distanza dal punto finale.

Sintassi 1

Parametri: **[a:]a1,b1,...bn:c:[d,e]**

dove:

a	=	velocità vettoriale
da a1 a an	=	assi interessati (vedi nota b)
da b1 a b	=	coordinate
c	=	distanza dal punto finale
d	=	nome o numero di un output digitale
e	=	valore a 1 o O del output digitale

Sintassi 2

Parametri: **[a:]Pb:c:[d,e]**

dove:

a	=	velocità vettoriale
Pb	=	numero del punto interessato
c	=	distanza dal punto finale
d	=	nome o numero di un output digitale
e	=	valore a 1 o O del output digitale

Esempio:

-MOR/5000:X1,50.5,Y1 ,75:1	Eseguire un movimento con velocità 5 m/min sul vettore X,Y alle quote indicate, con anticipo di segnalazione a 1 mm dalla fine.
-MOR/5000:X1,50.5,Y1 ,75:1:12,1	Eseguire un movimento con velocità 5 m/min sul vettore X,Y alle quote indicate, alla fine del movimento viene messo al valore 1 l'output 12, con anticipo di segnalazione a 1 mm dalla fine.

-MOR/5000:P10:1	Eseguire un movimento relativo agli assi (e alle coordinate) indicate nel punto P10 con velocità di 5 m/min , con anticipo di segnalazione a 1 mm dalla fine.

Note:

Per attendere il completamento del movimento utilizzare l'istruzione **HLC.**

Esempio:

```
-MOR/5000:X1,50.5,Y1 ,75:25
-SDO/output21,1 ; viene settato a 1il segnale "output21" a 25mm dalla fine del movimento
-HLC                        ; attende la fine del movimento
```

8.30 DCT: Deep Control Touch

DCT **DeepControlTouch**

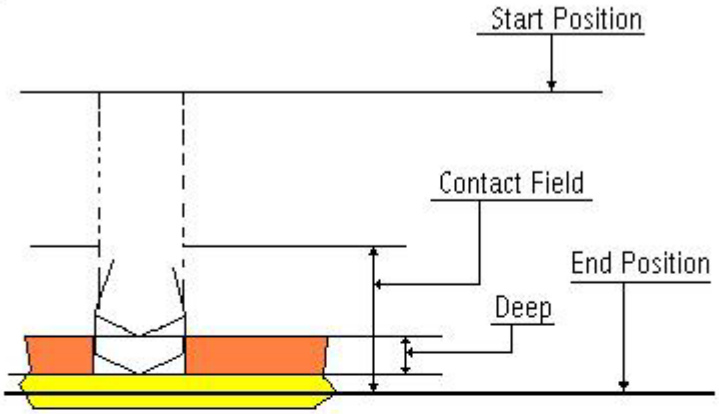
Funzione: muovere gli assi della macchina, alla velocità vettoriale indicata (il vettore è definito dagli assi Specificati) il contatto di tastatura, cambia la velocità del movimento e la posizione finale che sarà definita dal valore che aveva al contatto più un delta programmato.

Sintassi 1
Parametri: **[a:]a1,b1.an,bn:Deep,Field,Feed:G1:LABEL**

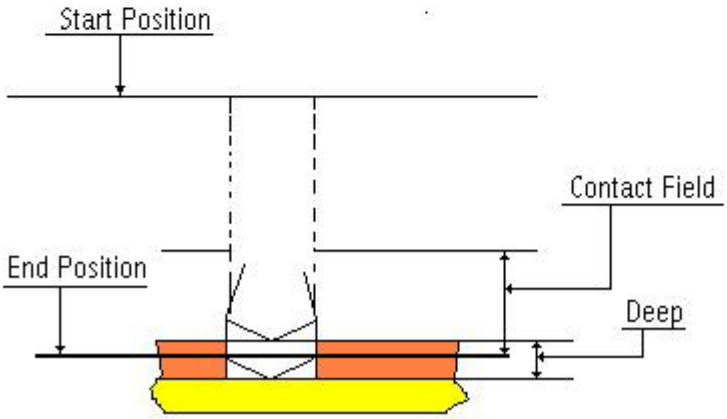
dove:

a	=	Velocità vettoriale
da a1 a an	=	Assi interessati
da b1 a bn	=	Posizione finale per NO TOUCH
Deep	=	Profondita' di foratura dalla quota tastata (se in campo)
Field	=	Campo di tastatura (riferita a b1)
Feed	=	Feed di tastatura (quella di foratura e quella standard)
G1	=	Globale / Locale per stato tastatura -2 = tastatura dopo il campo di sondaggio (SURFACE CONTROL) -1 = tastatura prima del campo di sondaggio (DEEP CONTROL e SURFACE CONTROL) 0 = tastatura corretta e foratura eseguita 1 = NO TOUCH
LABEL	=	Label di salto per contatto non trovato

Mode A



Mode B



Esempio:

```
          -SFP/0,0,0
          -MOV/Z1,0
          -SFP/2500,0,0
          -DCT/Z1,0.5:0.8,0.2,500:G1:FRKO
          -SFP/0,0,0
          -MOV/Z1,0
          -JMP/EXIT
FRKO      -JLT/G1,0,TRUC
          -DIS/NO TOUCH
          -JMP/EXIT
TRUC    -DIS/TRUCIOLO
EXIT      -RET
```

8.31 DCS: Deep Control Sensor

DCS **DeepControlSensor**

Funzione: muovere gli assi della macchina, alla velocità vettoriale indicata (il vettore è definito dagli assi Specificati) il contatto di un segnale digitale, cambia la velocità del movimento e la posizione finale che sarà definita dal valore che aveva al contatto più un delta programmato.

Sintassi 1
Parametri: [a:]a1,b1.an,bn:Deep,Field,Feed:Input Digitale,Valore:G1:LABEL
dove:

a	=	Velocità vettoriale
da a1 a an	=	Assi interessati
da b1 a bn	=	Posizione finale se non trovato switch
Deep	=	Profondita' di foratura dalla quota tastata (se in campo)
Field	=	Campo di tastatura (riferita a b1)
Feed	=	Feed di tastatura (quella di foratura e quella standard)
Input digitale	=	Numero / Nome del digital input
Valore	=	1 (digital input on) 0 (digital input off)
G1	=	Globale / Locale per stato tastatura -2 = tastatura dopo il campo di sondaggio (SURFACE CONTROL) -1 = tastatura prima del campo di sondaggio (DEEP CONTROL e SURFACE CONTROL) 0 = tastatura corretta e foratura eseguita 1 = NO TOUCH
LABEL	=	Label di salto per contatto non trovato

Casi particolari :

Field = 0 Funzionamento tipo TCH TMS con quota tastata
Le istruzione si fermano con decelerazione dopo
aver sentito la parete. La velocita di tastatura e' quella
di Feed se Deep <= 0 , altrimenti la minore tra Feed e quella
necessaria a fermarsi con uno spazio pari a Deep.

E.G.

```
                -SFP/0,0,0
                -MOV/Z1,0
                -SFP/2500,0,0
                -DCS/Z1,0.5:0.8,0,500:INP1,1:G1:FRKO
                -RAV/5:L1,Z1
                -DIS/1:POSIZIONE TASTATA ,L1
                -SFP/0,0,0
                -MOV/Z1,0
                -JMP/EXIT
FRKO            -
                -DIS/NO TOUCH
EXIT            -RET
```

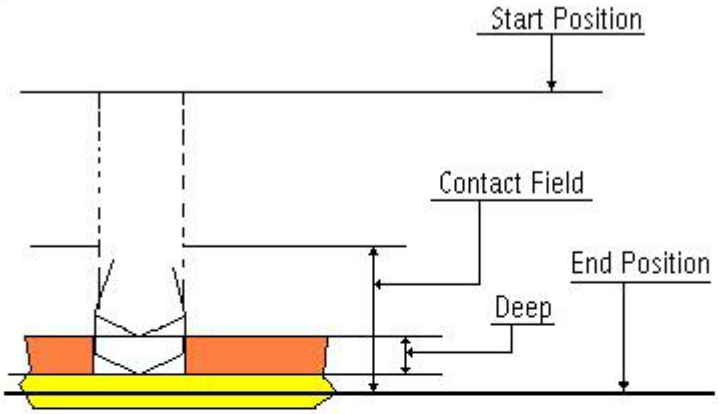
Field > 0 Funzionamento DEEP CONTROL
La posizione finale per NO TOUCH e' la fine del range
di tastatura. Se la parete non e' incontrata il quadrotto
puo' venire solo intaccato. Il movimento di lavoro e'
quasi interrotto nel range di sondaggio parete.
Valori di Deep <= 0 funzionano come una sorta di RETRACT.

E.G.

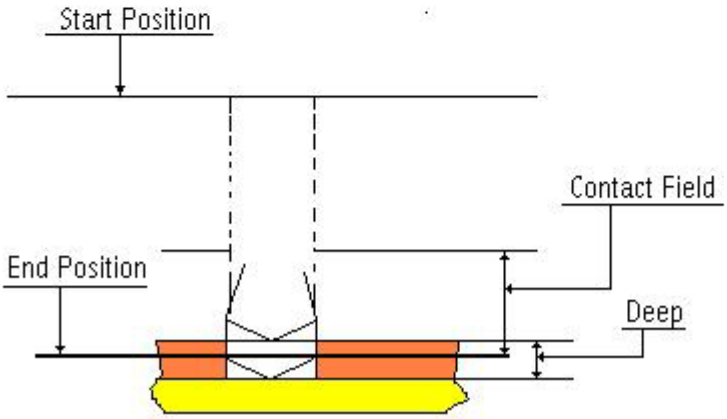
```
;               DEEP CONTROL
                -SFP/0,0,0
                -MOV/Z1,0
                -SFP/2500,0,0
                -DCS/Z1,0.5:1.2,0.5,0:INP1,1:G1:FRKO
                -RAV/5:L1,Z1
                -DIS/1:POSIZIONE TOCCATA ,L1
                -SFP/0,0,0
                -MOV/Z1,0
                -JMP/EXIT
FRKO            -JLT/G1,0,TRUC
                -DIS/NO TOUCH
                -JMP/EXIT
TRUC            -DIS/TRUCIOLO
EXIT            -RET

;               RETRACT
                -SFP/0,0,0
                -MOV/Z1,0
                -DCT/Z1,50:-0.2,50,200:G1:FRKO
                -RAV/5:L1,Z1
                -DIS/1:POSIZIONE TASTATA ,L1
                -JMP/EXIT
FRKO            -JLT/G1,0,TRUC
                -DIS/NO TOUCH
                -JMP/EXIT
TRUC            -DIS/TASTATURA PRIMA DEL RANGE
EXIT            -RET
```

```
                        Field < 0   Funzionamento SURFACE CONTROL
                        La posizione finale per NO TOUCH e' la foratura piu'
                        profonda prevista (quadrotto sfondato).
                        Se la parete non e' incontrata il quadrotto
                        viene quindi completamente forato. Il movimento di lavoro e'
                        continuo. In questo caso il range e' compreso tra Deep+Field
                        e Deep (rispetto alla posizione finale).
                        Non e' accettato Deep <= 0.
                        In questo caso e' previsto un nuovo stato di tastatura in G1
                       G1 = Globale / Locale per stato tastatura
                                    -2 = tastatura dopo il campo di sondaggio
E.G.
;                       SURFACE CONTROL
                       -SFP/0,0,0
                       -MOV/Z1,0
                       -SFP/2500,0,0
                       -DCS/Z1,-2:1.2,-0.5,0:INP1,1:G1:FRKO
                       -RAV/5:L1,Z1
                       -DIS/1:POSIZIONE TOCCATA ,L1
                       -SFP/0,0,0
                       -MOV/Z1,0
                       -JMP/EXIT
FRKO                   -JEQ/G1,-1,TRUC
                       -JEQ/G1,-2,SCHEG
                       -DIS/NO TOUCH - PUNTA ROTTA
                       -JMP/EXIT
TRUC                   -DIS/TRUCIOLO
                       -JMP/EXIT
SCHEG                  -DIS/PUNTA SCHEGGIATA
EXIT                   -RET
```

Mode A



Mode B



Esempio:

```
          -SFP/0,0,0
          -MOV/Z1,0
          -SFP/2500,0,0
          -DCS/Z1,0.5:0.8,0.2,500:Bit,1,G1:FRKO
          -SFP/0,0,0
          -MOV/Z1,0
          -JMP/EXIT
FRKO      -JLT/G1,0,TRUC
          -DIS/NO TOUCH
          -JMP/EXIT
TRUC    -DIS/TRUCIOLO
EXIT      -RET
```

8.32 GRM: GRoup maschine Management

GRM GRoup maschine Mnagement

Funzione: Gestione degli assi e mandrini associati in gruppi macchina

Parametri: **a:b1,„„bn**
dove:

a = Tipo di operazione (0-7)

0: ;SERVO OFF
1: ;// Sospensione gruppo con decelerazione immediata
2: ;// Ricontinuazione del gruppo sospeso
3: ;SERVO ON
4: ;RESET
5: ;// SFP da un task chiamante
6: ;// CFR da un task chiamante
5: ;// CSP da un task chiamante
10H: ;// Gestione del mandrino come MASCHIATURA
20H: ;// Gestione del mandrino come FRESATURA

b1,„bn = Numero identificativo Gruppo Macchina interessato all'operazione

Nota:

Nella gestione del mandrino come maschiatura, esso viene fermato o fatto ripartire contemporaneamente al movimento; eccetto con la sospensione del task a fine movimenti gia' aggregati, dove il mandrino viene fermato a movimenti finiti

Nella gestione del mandrino come fresatura, esso viene fermato dopo che il movimento e' fermato; fatto ripartire prima che il movimento sia rilanciato.

Esempio:

- -GRM/4:1,2,3,4

Effettua l'operazione di RESET ASSI e MANDRINI dei GRUPPI MACCHINA 1,2,3 e 4

CAPITOLO 9

9 Istruzioni per i segnali di input ed output

.1-WDI (WIN)	Attende che un segnale d'ingresso digitale si porti ad un determinato stato (Wait Digital Input)
.2-WAI	Attende che un segnale d'ingresso analogico si porti ad un determinato stato (Wait Analog Input)
.3-AIN	Attendi per il case uno degli input mettendo il numero sulla G o L (Attesa INput)
.4-TDI(TES)	Effettua un'operazione di test su un segnale d'ingresso digitale (Test Digital Input)
.5-TDO	Effettua un'operazione di test su un segnale d'uscita digitale (Test Digital Output)
.6-IDI	Effettua un'operazione di test su un segnale d'ingresso digitale (If Digital Input)
.7-IDO	Effettua un'operazione di test su un segnale d'uscita digitale (If Digital Output)
.8-TAI	Effettua un'operazione di test su un segnale d'ingresso analogico (Test Analog Input)
.9-SDO	(SAX) Setta o resetta dei segnali d'uscita digitale (Set Digital Output)
.10-EDO	Setta o resetta dei segnali d'uscita digitale in base ad un test (Enanced Digital Output)
.11-SAO	(SAC) Scrive il valore di un segnale analogico d'uscita (Set Analog Output)
.12-GDI	(RBI) Legge il valore di un segnale digitale d'ingresso (Get Digital Input)
.13-GDO	Legge il valore di un segnale digitale d'uscita (Get Digital Output)
.14-GAI(RAI)	Legge il valore di un segnale analogico d'ingresso (Get Analog Input)
.15-WBD(BPO)	Scrive un blocco di segnali digitali in uscita (Write Buffer Digital input)
.16-RBD(BPI)	Legge un blocco di segnali digitali in ingresso (Read Buffer Digital input)
.17-CPI	Attende che uno dei segnali d'ingresso digitale subisca una variazione
.18-CDI	Al cambiamento del segnale d'ingresso digitale attiva un task o processo
.19-CDO	Al cambiamento del segnale d'ingresso digitale attiva un task o processo
.20-RDI	Effettua un'operazione di test su un segnale d'ingresso digitale attivando un Task se il test è positivo (Run Digital Input)

.21-RDO Effettua un'operazione di test su un segnale d'uscita digitale attivando un Task se il test è positivo (Run Digital Output)

9.1 WDI: Wait Digital Input

WDI (Wait Digital Input)

Funzione: attende che un segnale d'ingresso digitale si porti ad un determinato stato.

Parametri: **a,b:[,c,d]**

dove:

a = numero o nome dell'input digitale

b = valore dell'input digitale

c = tempo massimo di attesa

d = label di salto se il time out è trascorso

Esempio:

```
-WDI/11,1          attende che il segnale digitale numero 11 diventi 1
-WDI/11,1:2000,NonArrivaSegnale
```

Note:

a) I parametri c e d sono opzionali

9.2 FWI: Fast Wait Digital Input

FWI (Fast Wait Digital Input)

Funzione: attende che un segnale d'ingresso digitale si porti ad un determinato stato.Il loop di controllo viene effettuato con una frequena molto più alta della istruzione WDI

Parametri: **a,b[,c,d]**

dove:

a = numero o nome dell'input digitale

b = valore dell'input digitale

c = tempo massimo di attesa

d = label di salto se il time out è trascorso

Esempio:

-WDI/11,1 attende che il segnale digitale numero 11 diventi 1

Note:

a) I parametri c e d sono opzionali

9.3 WAI: Wait Analog Input

WAI (Wait Analog Input)

Funzione: attende che un segnale d'ingresso digitale si porti ad un determinato stato.

Parametri: **a:b,c,d[,e,f]**

dove:

a = modalità di attesa valore
1= range interno attende che il valore entri nel range
2= range esterno attende che il valore esca dal range

b = numero o nome dell'input analogico

c = valore dell'input analogico soglia inferiore

d = valore dell'input analogico soglia superiore

e = tempo massimo di attesa

f = label di salto se il time out è trascorso

Esempio:

-WAI/1:Pressure,75,255 attende che il segnale analogico Pressure entri nella soglia tra 75 e 255

Note:

a) I parametri e e f sono opzionali

9.4 AIN: Attesa INput

AIN (Attesa INput)

Funzione: Attendi per il case uno degli input mettendo il numero sulla G o L.

Parametri: **a:b1-bn:c[:d,e]**

dove:

a = 1 o 0 per il valore degli input

da b1 a bn = numero o nome degli input digitali

c = Globale o Locale in cui viene inserito il numero dell'ingresso richiesto

d = time out

e = label di salto se il segnale di input digitale è uguale al valore indicato in **b**

Nella Locale o Globale viene caricato 0=se nessun input ottiene il valore richiesto nel tempo desiderato, altrimenti inizia da 1 se il primo della lista, 2 se il secondo e così via fino a n

Esempio:

```
-AIN/1:inp1,inp2,inp3:g1,1000,labelsalto
```

9.5 TDI: Test Digital Input

TDI (Test Digital Input)

Funzione: effettua un'operazione di test su un segnale d'ingresso digitale.

Parametri: **a,b,c[:d]**

dove:

a = numero o nome dell'input digitale

b = valore dell'input digitale

c = label di salto se il segnale di input digitale è diverso al valore indicato in **b**

d = Valore in millisecondi nel quale il segnale deve rimanere stabile(antirimbalzo)

Esempio:

-TDI/11,1, SignalZero	Effettua una operazione di test sul segnale digitale numero 11 e se 0 effettua un salto alla label SignalZero, altrimenti continua all'istruzione successiva
-TDI/11,1, SignalZero:1000	Effettua una operazione di test sul segnale digitale numero 11 e se 0 effettua un salto alla label SignalZero, altrimenti continua all'istruzione successiva se il segnale è stabile per 1 secondo (se non stabile salta a SignalZero)

9.6 TDO: Test Digital Output

TDO (Test Digital Output)

Funzione: effettua un'operazione di test su un segnale d'uscita digitale.

Parametri: **a,b,c[:d]**

dove:

a = numero o nome dell'input digitale

b = valore dell'output digitale

c = label di salto se il segnale di input digitale è diverso al valore indicato in **b**

d = Valore in millisecondi nel quale il segnale deve rimanere stabile(antirimbalzo)

Esempio:

-TDO/11,1, SignalZero	Effettua una operazione di test sul segnale digitale numero 11 e se 0 effettua un salto alla label SignalZero, altrimenti continua all'istruzione successiva
-TDO/11,1, SignalZero:1000	Effettua una operazione di test sul segnale digitale numero 11 e se 0 effettua un salto alla label SignalZero, altrimenti continua all'istruzione successiva se il segnale è stabile per 1 secondo (se non stabile salta a SignalZero)

9.7 IDI: If Digital Input

IDI (If Digital Input)

Funzione: effettua un'operazione di test su un segnale d'ingresso digitale.

Parametri: **a,b,c[:d]**

dove:

a = numero o nome dell'input digitale

b = valore dell'input digitale

c = label di salto se il segnale di input digitale è uguale al valore indicato in **b**

d = Valore in millisecondi nel quale il segnale deve rimanere stabile(antirimbalzo)

Esempio:

-IDI/11,1, SignalOne	Effettua una operazione di test sul segnale digitale numero 11 e se 1 effettua un salto alla label SignalOne
-IDI/11,1, SignalOne:1000	Effettua una operazione di test sul segnale digitale numero 11 e se 1 effettua un salto alla label SignalOne se rimane ad 1 per 1 secondo

9.8 IDO: If Digital Output

IDO (If Digital Output)

Funzione: effettua un'operazione di test su un segnale d'uscita digitale.

Parametri: **a,b,c[:d]**

dove:

a = numero o nome dell'input digitale

b = valore dell'output digitale

c = label di salto se il segnale di input digitale è uguale al valore indicato in **b**

d = Valore in millisecondi nel quale il segnale deve rimanere stabile(antirimbalzo)

Esempio:

-IDO/11,1, SignalOne	Effettua una operazione di test sul segnale digitale numero 11 e se 1 effettua un salto alla label SignalOne
-IDO/11,1, SignalOne:1000	Effettua una operazione di test sul segnale digitale numero 11 e se 1 effettua un salto alla label SignalOne se rimane ad 1 per 1 secondo

9.9 TAI: Test Analog Input

TAI (Test Analog Input)

Funzione: effettua un'operazione di test su un segnale d'ingresso analogico.

Parametri: **a,b,c,d**

dove:

a = numero o nome dell'input analogico

b = valore dell'input analogico soglia inferiore

c = valore dell'input analogico soglia superiore

d = label di salto se il segnale analogico entra nel range di valori espressa da b e c

Esempio:

```
-TAI/ Pressure,75,255, SignalOnRange    ;Effettua una operazione di test sul segnale analogico Pressure e se entra nel range da 75 a 255 effettua un salto alla label SignalOnRange
```

9.10 SDO: Set Digital Output

SDO (Set Digital Output)

Funzione: effettua un'operazione di scrittura su dei segnali d'uscita digitale.

Parametri: **[n:]a1,b1…an,bn**

dove:

[n:] = caso particolare di gestione dei segnali vedi nota 1

da a1 a an = numero o nome dell'output digitale

da b1 a bn = Valore dell'output digitale 1 e 0 sono i due valori consentiti

Esempio:

-SDO/Elevator,1,Driver,0	Effettua una operazione di scrittura a 1 sul segnale Elevator e 0 sul segnale Driver

Nota 1:

Viene effettuato l'output consecutivo di bit a partire dal parametro indicato da a1 con i valori della Locale o Globale o numero diretto indicato nel parametro b1.
Simile alla WBO è però per complementare con l'istruzione GDO

Esempio:

-SDO/16:M2.1,L1	Effettua una operazione di scrittura del valore contenuto nella locale L1 sul merker WORD M2

Esempio :

```
        -SDO/16:M3.1,L1
AS-
        -GDO/16:L1,M3.1
        -LET/L1,L1+1
        -SDO/16:M3.1,L1
        -TMM/1
        -JMP/AS


        -SDO/16:IM1.1,L2
AZ-
        -GDI/16:L2,IM1.1
        -LET/L2,L2+1
        -SDI/16:IM1.1,L2
        -TMM/1
        -JMP/AZ
```

9.11 SDI: Set Digital Input

SDI (Set Digital Input)

Funzione: effettua un'operazione di scrittura su dei segnali d'uscita digitale.

Parametri: **[n:]a1,b1…an,bn**

dove:

[n:] = caso particolare di gestione dei segnali vedi nota 1
da a1 a an = numero o nome dell'input digitale

da b1 a bn = Valore dell'input digitale 1 e 0 sono i due valori consentiti

Esempio:

-SDI/M1.3,1,M1.4,0 Effettua una operazione di scrittura a 1 sul segnale M1.3 e 0 sul segnale M1.4

Nota 1:

Viene effettuato l'output consecutivo di bit a partire dal parametro indicato da a1 con i valori della Locale o Globale o numero diretto indicato nel parametro b1.
Simile alla WBO, agisce però sugli input , inoltre è per complementare con l'istruzione GDI

Esempio:

-SDI/16:M1.1,L1 Effettua una operazione di scrittura del valore contenuto nella locale L1 sul merker WORD M1

Esempio :

```
        -SDO/16:M3.1,L1
AS-
        -GDO/16:L1,M3.1
        -LET/L1,L1+1
        -SDO/16:M3.1,L1
        -TMM/1
        -JMP/AS


        -SDO/16:IM1.1,L2
AZ-
        -GDI/16:L2,IM1.1
        -LET/L2,L2+1
        -SDI/16:IM1.1,L2
        -TMM/1
        -JMP/AZ
```

9.12 EDO: Enanced Digital Output

EDO (Enanced Digital Output)

Funzione: Setta o resetta dei segnali d'uscita digitale in base ad un test (Enanced Digital Output).

Parametri: **a,b:c1,d1…cn,dn[:e]**

dove:

a	= variabile Locale o Globale, oppure il nome o il numero di un segnale d'ingresso oppure il nome di un segnale d'uscita.
b	= valore del test per cui se uguale ad **a** gli output **c1 a cn** dei parameri vengono messi al valore indicato da **d1 a dn**
da c1 a cn	= numero o nome dell'output digitale
da d1 a dn	= valore dell'output digitale 1 e 0 sono i due valori consentiti
e	= valore se zero indica in caso non uguale di **a** a **b** i segnali di output **c1 a cn** non verranno settati al modo contrario di quello indicato da **d1 a dn** (Default 1)

Esempio:

```
-EDO/input,1:out1,1,out2,1:1

-EDO/ISA8255_OUT9,1:ISA8255_OUT10,1,ISA8255_OUT11,1,ISA8255_OUT12,1,ISA8255_OUT13,1:0
```

9.13 SAO: Set Analog Output

SAO (Set Analog Output)

Funzione: effettua un'operazione di scrittura su un segnale d'uscita analogico.

Parametri: **a1,b1…an,bn**

dove:

da a1 a an = numero o nome dell'output analogico

da b1 a bn = Valore dell'output analogico

Esempio:

-SAO/Climp,200,Rout,0	Effettua una operazione di scrittura a 200 sul segnale Climp e 0 sul segnale Rout

9.14 GDI: Get Digital Input

GDI Get Digital Input

Funzione: eseguire una operazione di lettura di un valore di un segnale digitale d'ingresso e lo mette in una variabile.

Parametri: **[n:]a1,b1,a2,b2,...an,bn**

dove:

[n:] = caso particolare di gestione dei segnali vedi nota 1
da a1 a an = variabile da scrivere
da b1 a bn = valori dell' segnale d'ingresso digitale da introdurre nelle variabili

Esempio:

-GDI/L1,Pulse1,G7,Pulse2	Scrivo la variabile locale 1 con il valore di Pulse1 e la variabile globale 7 con il valore di Pulse2.

Nota 1:

Viene effettuato la lettura consecutivo di bit a partire dal parametro indicato da **b1** ed il valore è posto nella Globale o Locale indicata nel parametro **a1.**
Simile alla RBI è però per complementare con l'istruzione SDI

Esempio:

-GDI/16:L1, M2.1	Effettua una operazione di lettura del merker WORD M2 lo scrive su L1

Esempio :

```
        -SDO/16:M3.1,L1
AS-
        -GDO/16:L1,M3.1
        -LET/L1,L1+1
        -SDO/16:M3.1,L1
        -TMM/1
        -JMP/AS


        -SDO/16:IM1.1,L2
AZ-
        -GDI/16:L2,IM1.1
        -LET/L2,L2+1
        -SDI/16:IM1.1,L2
        -TMM/1
        -JMP/AZ
```

9.15 GDO: Get Digital Output

GDO Get Digital Output

Funzione: eseguire una operazione di lettura di un valore di un segnale digitale d'uscita e lo mette in una variabile.

Parametri: **[n:]a1,b1,a2,b2,...an,bn**

dove:
[n:] = caso particolare di gestione dei segnali vedi nota 1
da a1 a an = variabile da scrivere
da b1 a bn = valori dell' segnale d'uscita digitale da introdurre nelle variabili

Esempio:

-GDO/L1,Pulse1,G7,Pulse2	Scrivo la variabile locale 1 con il valore di Pulse1 e la variabile globale 7 con il valore di Pulse2.

Nota 1:

Viene effettuato la lettura consecutivo di bit a partire dal parametro indicato da **b1** ed il valore è posto nella Globale o Locale indicata nel parametro **a1.**
Simile alla RBI, ma agisce sugli output ed inoltre è complementare con l'istruzione SDO

Esempio:

-GDO/16:L1, M1.1	Effettua una operazione di lettura del merker WORD M1 lo scrive su L1

Esempio :

```
        -SDO/16:M3.1,L1
AS-
        -GDO/16:L1,M3.1
        -LET/L1,L1+1
        -SDO/16:M3.1,L1
        -TMM/1
        -JMP/AS


        -SDO/16:IM1.1,L2
AZ-
        -GDI/16:L2,IM1.1
        -LET/L2,L2+1
        -SDI/16:IM1.1,L2
        -TMM/1
        -JMP/AZ
```

9.16 GAI: Get Analog Input

GAI **Get Analog Input**

Funzione: eseguire una operazione di lettura di un valore di un segnale analogico d'ingresso e lo mette in una variabile.

Parametri: **a1,b1,a2,b2,...an,bn**

dove:

da a1 a an = variabile da scrivere
da b1 a bn = valori dell' segnale d'ingresso analogico da introdurre nelle variabili

Esempio:

-GAI/L1,Pressure,G7,Temp	Scrivo la variabile locale 1 con il valore di Pressure e la variabile globale 7 con il valore di Temp.

9.17 WBD: Write Buffer Digital input (BPO)

WBD (BPO) Write Buffer Digital output (Bit Parallel Output)

Funzione: eseguire una operazione di scrittura di un blocco di segnali digitali in uscita.

Parametri: **a,b,c**

dove:

a = numero di bit su cui agire (output da settare) 1-16

b = valore decimale che tradotto in maschera binaria corrisponde allo stato degli output da settare
(la modalità di scrittura in esadecimale 0X.. può essere molto utile)

c = n° dell'output di partenza

Esempio:

-WBD/6, 11,8 Agire sugli output (fisici normali) 8:13, settandoli in ON/OFF

Note:

a) Gli output devono essere progressivi a partire da quello specificato nel parametro **c**.

b) Se non esistono gli output (fisici del pacco A) corrispondenti alla somma dei parametri **b+d** viene visualizzato il messaggio di errore.

c) Il valore binario dell'output eccitato è 1 e diseccitato è 0.

d) Nell'esempio sopra riportato, vengono eccitati gli output 8,9 e 11 e diseccitati gli output 10,12,13 (perchè al valore decimale 11 corrisponde la maschera binaria 001011):

output	13	12	11	10	9	8
maschera binaria	0	0	1	0	1	1
	2^5	2^4	2^3	2^2	2^1	2^0
eccitati			X		X	X

valore decimale 11=1+2+8

e) La maschera binaria di un valore negativo corrisponde al complemento a 2 del valore positivo, che si ottiene sommando 1 al complemento della maschera:

Esempio:

+5	maschera		0	1	0	1
	complemento		1	0	1	0
		+				1
		---	---	---	---	---
maschera negativa			1	0	1	1

Esempi di valori positivi e negativi con la stessa maschera:

operando su4bit	0 = -16	maschera	0	0	0	0
(val.max.15)	1 = -15		0	0	0	1
	2 = -14		0	0	1	0
	.					
	.					
	.					
	15 = -1		1	1	1	1

operandosu8bit	0 = -256	0	0	0	0	0	0	0	0
(val.max.255)	1 = -255	0	0	0	0	0	0	0	1
	2 = -254	0	0	0	0	0	0	1	0
.									
.									
.									
.									
	255 = -1	1	1	1	1	1	1	1	1

operando su 16 bit	0 = -65536	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(val. max. 65535)	1 = -65535	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	2 = -65534	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	.																	
	.																	
	.																	
	.																	
	65535 = -1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

; L1 = numero passi
; L2 = motore 1-2-3

Esempio:

```
; Impostazione velocità
        -JEQ/L2,1,mot1
        -JEQ/L2,2,mot2
        -JEQ/L2,3,mot3
        -JEQ/L2,4,mot4
        -JEQ/L2,5,mot5
        -JEQ/L2,6,mot6
        -JEQ/L2,7,mot7
        -JEQ/L2,8,mot8
        -JEQ/L2,9,mot9
        -JEQ/L2,10,mot10
        -JEQ/L2,11,mot11
        -JEQ/L2,12,mot12
        -JMP/errore

mot1 -
        -LET/L17,201,L18,L17+15                 ; Motore 1
        -JMP/aa
mot2 -
        -LET/L17,217,L18,L17+15                 ; Motore 2
        -JMP/aa
mot3 -
        -LET/L17,233,L18,L17+15                 ; Motore 3
        -JMP/aa
mot4 -
        -LET/L17,249,L18,L17+15                 ; Motore 4
        -JMP/aa
mot5 -
        -LET/L17,265,L18,L17+15                 ; Motore 5
        -JMP/aa
mot6 -
        -LET/L17,281,L18,L17+15                 ; Motore 6
        -JMP/aa
mot7 -
        -LET/L17,297,L18,L17+15                 ; Motore 7
        -JMP/aa
mot8 -
        -LET/L17,313,L18,L17+15                 ; Motore 8
        -JMP/aa
mot9 -
        -LET/L17,329,L18,L17+15                 ; Motore 9
        -JMP/aa
mot10 -
        -LET/L17,345,L18,L17+15                 ; Motore 10
        -JMP/aa
mot11 -
        -LET/L17,361,L18,L17+15                 ; Motore 11
        -JMP/aa
mot12 -
        -LET/L17,377,L18,L17+15                 ; Motore 12
        -JMP/aa



aa  -
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0701,L17              ; enable motore
        -WDI/L18,1:1000,Errore1
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
```

```
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0D00,L17              ; azzeramento encoder
        -WDI/L18,1:1000,Errore1

        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:10000,Errore0
        -JGE/L1,0,dir_pos
        -BPO/16,0x0901,L17              ; direzione motore negativo
        -JMP/con_dir
dir_pos -
        -BPO/16,0x0900,L17              ; direzione motore positivo
con_dir -
        -WDI/L18,1:1000,Errore1

b   -
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0A00,L17              ; esecuzione 1 step
        -WDI/L18,1:1000,Errore1
        -BPI/15,L19,L17                 ; esecuzione 1 step
        -DIS/L2:Stato ,L19,4,-1
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0C00,L17              ; comando Leggi Encoder
        -WDI/L18,1:1000,Errore1
        -BPI/14,L20,L17                 ; esecuzione 1 step
        -LET/L21,L2+5
        -DIS/L21:Encoder ,L20,4,-1
        -LET/L19,and(L19,1)
        -JEQ/L19,0,b
//// trovato micro

// inverto direzione
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:10000,Errore0
        -JGE/L1,0,dir_posc
        -BPO/16,0x0900,L17              ; direzione motore positivo
        -JMP/con_dirc
dir_posc-
        -BPO/16,0x0901,L17              ; direzione motore negativo
con_dirc-

        -WDI/L18,1:1000,Errore1

// reset marker
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0E00,L17              ; reset marker
        -WDI/L18,1:1000,Errore1

// micro a zero
z   -
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0A00,L17              ; esecuzione 1 step
        -WDI/L18,1:1000,Errore1
        -BPI/15,L19,L17                 ; esecuzione 1 step
        -DIS/L2:Stato ,L19,4,-1
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0C00,L17              ; comando Leggi Encoder
        -WDI/L18,1:1000,Errore1
        -BPI/14,L20,L17                 ; esecuzione 1 step
```

```
        -LET/L21,L2+5
        -DIS/L21:Encoder ,L20,4,-1
        -LET/L19,and(L19,1)
        -JEQ/L19,1,z

// ricerca marker
j   -
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0A00,L17              ; esecuzione 1 step
        -WDI/L18,1:1000,Errore1
        -BPI/15,L19,L17             ;
        -DIS/L2:Stato ,L19,4,-1
        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0C00,L17              ; comando Leggi Encoder
        -WDI/L18,1:1000,Errore1
        -BPI/14,L20,L17                 ; esecuzione 1 step
        -LET/L21,L2+5
        -DIS/L21:Encoder ,L20,4,-1
        -LET/L19,and(L19,0x10)
        -JNE/L19,0x10,j
//

        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0700,L17              ; tolgo enable
        -WDI/L18,1:1000,Errore1
        -BPI/15,L19,L17             ;
        -DIS/L2:Stato ,L19,4,-1

        -BPO/16,0x0,L17                 ; richiesta esecuzione comando
        -WDI/L18,0:1000,Errore0
        -BPO/16,0x0C00,L17              ; comando Leggi Encoder
        -WDI/L18,1:1000,Errore1
        -BPI/14,L20,L17                 ; esecuzione 1 step
        -LET/L21,L2+5
        -DIS/L21:Encoder ,L20,4,-1


        -DIS/finito
        -RET


Errore0 -
        -DIS/Errore 0
        -END
Errore1  -
        -DIS/Errore 1
        -END
```

9.18 RBD: Read Buffer Digital input (BPI)

RBD(BPI) Read Buffer Digital input (Bit Parallel Input)

Funzione: ricevere su un variabile lo stato (ON/OFF) di un buffer di inputs digitali

Parametri: **a,b,c**

dove:

a = numero di input (bit) di cui si riceve la situazione (da 1 a 16)

b = variabile nella quale viene memorizzato il valore decimale corrispondente alla maschera binaria funzione dello stato degli input

c = n° dell'input di partenza

Esempio:

-RBDI/5,L1,7 memorizzare nella variabile locale 1 il valore corrispondente allo stato degli input 7-11 (input fisici normali).

Note:

a) Gli input interessati devono essere progressivi a partire da quello definito dal parametro **c**.

b) Se gli input (fisici del pacco A) corrispondenti alla somma dei parametri **a** e **c** non esistono, in esecuzione compare il messaggio di errore.

c) All'input eccitato corrisponde il bit 1; all'input diseccitato corrisponde il bit 0.

- Nell'esempio sopra riportato, supponiamo che lo stato degli input è il seguente:

  7,9,10 eccitati

  8,11 diseccitati

input	11	10	9	8	7	
eccitato		X	X		X	
diseccitato	X			X		
	2^4	2^3	2^2	2^1	2^0	
maschera	0	1	1	0	1	(1+4+8) = 13

La variabile L1 assume il valore 13.

9.19 CPI: Change Parallel Input

CPI **Change Parallel Input**

Funzione : attende che uno dei segnali d'ingresso digitale subisca una variazione

Parametri: **a,b,c:[d,e]**

dove:

a = numero di bit su cui agire (output da settare) 1:16
b = valore decimale che tradotto in maschera binaria corrisponde allo stato degli output da settare
c = n° dell'inpput di partenza
e = timeout in millisecondi
f = label di salto

Esempio:

- CPI/8,L1,Input:1000,labeltimeout

9.20 CDI: Change Digital Input

CDI **Change Digital Input**

Funzione : al cambiamento del segnale d'ingresso digitale attiva un task o processo

Parametri: **a,b,c:d:e:f1..fn:g**

dove:

a	=	variabile Locale o Globale in viene registrato il numero del segnale d'ingresso
b	=	variabile Locale o Globale in viene registrato il segnale 0 o 1 del segnale d'ingresso
c	=	numero o nome segnale d'ingresso
d	=	nomefile, nomecall del TASK attivato quando il segnale commuta ad valore logico 1
e	=	nomefile.pp, nomecall del TASK attivato quando il segnale commuta ad valore logico 0
f 1..n	=	parametri trasferiti al TASK nelle sue Locali da 1.. n
g	=	modalità di controllo sull'esistenza del TASK : 0=viene sempre lanciato creando una nuova istanza 1=non viene lanciato , ma aspetta l'uscita di quello attivo per lanciarlo 2=segnala l'errore

Esempio:

```
-CDI/L1,L2,INPUT: nomefile1.pp , nomecall1 :nomefile0.pp , nomecall0:I2,56,I1:0
```

9.21 CDO: Change Digital Output

CDO **Change Digital Output**

Funzione : al cambiamento del segnale d'uscita digitale attiva un task o processo

Parametri: **a,b,c:d:e:f1..fn:g**

dove:

a	=	variabile Locale o Globale in viene registrato il numero del segnale d'uscita
b	=	variabile Locale o Globale in viene registrato il segnale 0 o 1 del segnale d'uscita
c	=	numero o nome segnale d'uscita
d	=	nomefile, nomecall del TASK attivato quando il segnale commuta ad valore logico 1
e	=	nomefile.pp, nomecall del TASK attivato quando il segnale commuta ad valore logico 0
f 1..n	=	parametri trasferiti al TASK nelle sue Locali da 1.. n
g	=	modalità di controllo sull'esistenza del TASK : 0=viene sempre lanciato creando una nuova istanza 1=non viene lanciato , ma aspetta l'uscita di quello attivo per lanciarlo 2=segnala l'errore

Esempio:

```
--CDO/L1,L2,OUTPUT: nomefile1.pp, nomecall1 : nomefile0.pp, nomecall0 :l2,56,l1:1
```

Esempio gestione movimento manuale con I/O

```
                -LET/G1,0,G2,0
pp              -CDO/L1,L2,1:avanti:zero:L1,L2:1
                -CDO/L1,L2,2:indietro:zero:L1,L2:1
                -JMP/pp

PartProgram[indietro]
                -DIS/1:Muovi manuale indietro
                -MMA/1:X,-10
                -RET

PartProgram[avanti]
                -DIS/1:Muovi manuale avanti
                -MMA/1:X,10
                -RET

PartProgram[zero]
                -DIS/1:Ferma manuale
                -MMA/0:X
                -RET



;////////////////////////////////////////////////////////
;     NUOVA GESTIONE STATI STOP MACCHINA
;////////////////////////////////////////////////////////
;--------------------------------------------------------
;                VARIABILI GLOBALI USATE
;--------------------------------------------------------
;                G50      = codice di errore
;                G50-bit1=1              emergenza
;                G50-bit2=1              mancanza aria generale
;                G50-bit3=1              mancanza aria mandrino sinistro
;                G50-bit4=1              mancanza aria pinza sinistra
;                G50-bit5=1              mancanza aria mandrino destro
;                G50-bit6=1              mancanza aria pinza destra
;                G50-bit7=1              failure centralina raffreddamento
;--------------------------------------------------------
;                G51=            stato del programma p.p.
;                G51=0           idle
;                G51=1           programma in esecuzione
;--------------------------------------------------------
;                        G58 = FLAG DI EMERGENZA
;                G58 = 1 emergenza in atto
;--------------------------------------------------------
;                        G350-bits
;                0x100 = HOME.PP running (set reset da HOME.PP)
;                0x002   mancanza aria generale
;                0x004   mancanza aria mandrino sinistro
;                0x008   mancanza aria pinza sinistra
;                0x010   mancanza aria mandrino destro
;                0x020   mancanza aria pinza destra
;                0x040   failure centralina raffreddamento
;--------------------------------------------------------
;                        G500 = ABILITAZIONE A DARE POTENZA
;                G500 = 0  disabilitazione a dare potenza
;                G500 = 1  abilitazione a dare potenza
;************************************************
;
;                  INIZIO PROGRAMMA
;************************************************
;
```

```
                -JEQ/G500,0,AVANT                       ;disabilitazione a dare potenza
                -SDO/40,1                               ;abilitazione SW al power-on
                -SDO/51,0                               ;spengo lampada emergenza
AVANT           -RST/1
                -TMM/10                                 ;mando un reset alla scheda assi
                -LET/G50,0,G350,0                       ;reset variabili globali
INIZIO          -LET/G51,1                              ;segnalo programma in esecuzione
;               -TMM/10
                -LET/L11,L11+1
                -JGE/L11,0,StartNeg
                -DIS/10:PLC Running.. ,L11
                -JLT/L11,0,StartCon
                -LET/L11,10
                -JMP/StartCon
StartNeg                -DIS/10:  ,L11
                -JLT/L11,10,StartCon
                -LET/L11,-10
StartCon-
;               -TSK/,gestpa~1.PP,L32:1                 ;chiamo gestione pannello
;               -TSK/,lampcontr.PP,L32:1                ;chiamo gestione lampade
;               -JEQ/G500,0,CONT                        ;disabilitazione a dare potenza
                -SDO/40,1                               ;abilitazione SW alpower-on
CONT            -CDI/L1,L2,1:RESET1:SEGNAL1::1;controllo stato emergenza
                -CDI/L1,L2,2:RESET2:SEGNAL2::1;controllo aria generale
                -CDI/L1,L2,3:RESET3:SEGNAL3::1;controllo aria mandrino sinistro
                -CDI/L1,L2,4:RESET4:SEGNAL4::1;controllo aria mandrino destro
                -CDI/L1,L2,5:RESET5:SEGNAL5::1;controllo aria pinza sinistra
                -CDI/L1,L2,6:RESET6:SEGNAL6::1;controllo aria pinza destra
                -CDI/L1,L2,7:RESET7:SEGNAL7::1;controllo centralina
                -JMP/INIZIO
;-----------------------------------------------
;             LANCIO I VARI TASK
;-----------------------------------------------
PartProgram[SEGNAL1]
                -DIS/9:0:\\\\\-
                -TDI/24,1,SALT1                 ;controllo aria generale
                -LET/G50,OR(G50,0x1)                    ;segnalo stato di emergenza sul bit 1
                -LET/G58,1                              ;segnalo emergenza in atto
                -LET/G250,1                             ;setto flag emergenza
                -LET/G500,1                             ;abilitazione a dare potenza***
                -SDO/34,0,35,0                          ;spengo sorgenti
                -TDI/18,0,SALT                          ;controllo se il gripper sinistro è sù
                -SDO/21,1                               ;apro il gripper
                -TMM/100                                ;attesa 1 sec
                -SDO/22,0                               ;abbasso il gripper
                -TMM/1000                               ;attesa 1 sec
                -SDO/21,0                               ;chiudo il gripper
SALT    -TDI/20,0,SALT1                         ;controllo se  gripper destro è sù
                -SDO/23,1                               ;apro il gripper
                -TMM/100                                ;attesa 1 sec
                -SDO/24,0                               ;abbasso il gripper
                -TMM/100                                ;attesa 1 sec
                -SDO/23,0                               ;chiudo il gripper
SALT1   -DIS/9:0:EmerG
                -END

;-------------------------------------------------------------------------------
PartProgram[SEGNAL2]
SEGNAL2         -DIS/9:6:\\\\\-
                -TDI/24,0,FINE2                         ;ricontrollo aria generale
                -TMM/3000                               ;attesa
                -TDI/24,0,SEGNAL2                       ;ricontrollo aria generale
                -LET/G50,OR(G50,0x2)                    ;segnalo mancanza aria generale sul bit 2
```

```
                -LET/G58,1                              ;segnalo emergenza in atto
                -SDO/65,0,66,0,67,0,68,0                        ;fermo carrelli
                -SDO/34,0,35,0                          ;spengo le sorgenti RX
                -RST/1                                  ;resetto  i movimenti in corso
                -SPD/M1,0                               ;fermo mandrino SX
                -SPD/M2,0                               ;fermo mandrino DX
                -CAL/ZLUP::1                            ;ZL UP
                -CAL/ZRUP::1                            ;ZR UP
                -SDO/40,0                               ;disabilitazione SW al power-on
                -LET/G50,AND(G50,NOT(0x6D))
                -LET/G500,0                             ;disabilitazione a dare potenza***
                -SDO/51,1,49,0,50,0                     ;accendo lampada emergenza e spengo le altre
FINE2           -DIS/9:6:AriaG
                -END

;-------------------------------------------------------------------------
PartProgram[SEGNAL3]
SEGNAL3         -DIS/9:12:\\\\\-
                -TDI/1,0,FINE3                          ;ricontrollo aria mandrino sinistro
                -TMM/500
                -TDI/1,0,SEGNAL3                        ;ricontrollo aria mandrino sinistro
                -LET/G50,OR(G50,0x4)                    ;segnalo mancanza aria mandrino sinistro sul bit 3
                -LET/G58,1                              ;segnalo emergenza in atto
                -SPD/M1,0                               ;fermo mandrino SX
                -SDO/51,1,49,0,50,0                     ;accendo lampada emergenza e spengo le altre
FINE3           -DIS/9:12:Ar.MS-
                -END
;-------------------------------------------------------------------------
PartProgram[SEGNAL4]
SEGNAL4         -DIS/9:18:\\\\\-
                -TDI/4,0,FINE4                          ;ricontrollo aria mandrino destro
                -TMM/500
                -TDI/4,0,SEGNAL4                        ;ricontrollo aria mandrino destro
                -LET/G50,OR(G50,0x8)                    ;segnalo mancanza aria mandrino destro sul bit 4
                -LET/G58,1                              ;segnalo emergenza in atto
                -SDO/51,1,49,0,50,0                     ;accendo lampada emergenza e spengo le altre
FINE4           -DIS/9:18:Ar.MD-
                -END
;-------------------------------------------------------------------------
PartProgram[SEGNAL5]
SEGNAL5         -DIS/9:24:\\\\\-
                -TDI/2,0,FINE5                  ;ricontrollo aria pinza sinistra
                -TMM/500
                -TDI/2,0,SEGNAL5                        ;ricontrollo aria pinza sinistra
                -LET/G50,OR(G50,0x10)           ;segnalo mancanza aria pinza sinistra sul bit 5
                -LET/G58,1                              ;segnalo emergenza in atto
                -SPD/M1,0                               ;fermo mandrino SX
                -SDO/51,1,49,0,50,0                     ;accendo lampada emergenza e spengo le altre
FINE5           -DIS/9:24:Ar.PS-
                -END
;-------------------------------------------------------------------------
PartProgram[SEGNAL6]
SEGNAL6         -DIS/9:30:\\\\\-
                -TDI/5,0,FINE6                  ;ricontrollo aria pinza destra
                -TMM/500
                -TDI/5,0,SEGNAL6                        ;ricontrollo aria pinza destra
                -LET/G50,OR(G50,0x20)           ;segnalo mancanza aria pinza destra sul bit 6
                -LET/G58,1                              ;segnalo emergenza in atto
                -SDO/51,1,49,0,50,0                     ;accendo lampada emergenza e spengo le altre
FINE6           -DIS/9:30:Ar.PD-
                -END
;-------------------------------------------------------------------------
PartProgram[SEGNAL7]
```

```
SEGNAL7         -DIS/9:36:\\\\\-
                -TDI/39,0,FINE7          ;ricontrollo input centralina
                -TMM/300
                -TDI/39,0,SEGNAL7                ;ricontrollo input centralina
                -LET/G50,OR(G50,0x40)            ;segnalo anomalia centralina raffreddamento sul bit 7
                -LET/G58,1                       ;segnalo emergenza in atto
                -SPD/M1,0                        ;fermo mandrino SX
                -SDO/51,1,49,0,50,0              ;accendo lampada emergenza e spengo le altre
FINE7           -DIS/9:36:Centr-
                -END

;-------------------------------------------------
;               TASK DI RESET
;-------------------------------------------------
;Reset segnalazione mancanza potenza
PartProgram[RESET1]
                -LET/G50,AND(G50,NOT(0x1))       ;resetto bit 1
                -JEQ/G500,0,AVANT1               ;disabilitazione a dare potenza
                -SDO/40,1                        ;abilitazione SW alpower-on
                -LET/G58,0                       ;resetto flag emergenza
                -CAL/,SpinPreset.PP,l32          ;chiamo preset mandrini
AVANT1 -DIS/9:0:------
                -END

;-------------------------------------------------
;Reset segnalazione mancanza aria generale
PartProgram[RESET2]
                -LET/G50,AND(G50,NOT(0x2))               ;resetto bit 2
                -LET/G350,AND(G350,NOT(0x2))
                -DIS/9:6:------
                -END
;-------------------------------------------------
;Reset segnalazione mancanza aria mandrino sinistro
PartProgram[RESET3]
                -LET/G50,AND(G50,NOT(0x4))               ;resetto bit 3
                -LET/G350,AND(G350,NOT(0x4))
                -DIS/9:12:------
                -END
;-------------------------------------------------
;Reset segnalazione mancanza aria mandrino destro
PartProgram[RESET4]
                -LET/G50,AND(G50,NOT(0x8))               ;resetto bit 4
                -LET/G350,AND(G350,NOT(0x8))
                -DIS/9:18:------
                -END
;-------------------------------------------------
;Reset segnalazione mancanza aria pinza sinistra
PartProgram[RESET5]
                -LET/G50,AND(G50,NOT(0x10))              ;resetto bit 5
                -LET/G350,AND(G350,NOT(0x10))
                -DIS/9:24:------
                -END
;-------------------------------------------------
;Reset segnalazione mancanza aria pinza destra
PartProgram[RESET6]
                -LET/G50,AND(G50,NOT(0x20))              ;resetto bit 6
                -LET/G350,AND(G350,NOT(0x20))
                -DIS/9:30:-----
                -END
-------------------------------------------------
;Reset segnalazione anomalia centralina raffreddamento
PartProgram[RESET7]
                -LET/G50,AND(G50,NOT(0x40))              ;resetto bit 7
```

```
                -LET/G350,AND(G350,NOT(0x40))
                -DIS/9:36:------
                -END

;*************************************************
;*************************************************
;                       SUBROUTINE
;*************************************************
;PartProgram[EMERGENZA]
;               Emergenza assi
;               -SDO/40,0                                       ;disabilitazione SW al power-on
;               -RET
;-------------------------------------------------
PartProgram[ZLUP]
;               Asse ZL alto
;               L1 = Valore ZL alto
                -RAV/0:L2,SYSTEM
                -LET/L2,AND(L2,2)
                -JEQ/L2,0,VIA
RIP             -RAV/0:L2,ZL
                -LET/L3,AND(L2,0x4)
                -JEQ/L3,0,VIA
                -LET/L2,AND(L2,0x80000)
                -JNE/L2,0,RIP
                -MOV/ZL,L1
VIA             -RET
;-------------------------------------------------
PartProgram[ZRUP]
;               Asse ZR alto
;               L1 = Valore ZR alto
                -RAV/0:L2,SYSTEM
                -LET/L2,AND(L2,2)
                -JEQ/L2,0,VIA
RIP             -RAV/0:L2,ZR
                -LET/L3,AND(L2,0x4)
                -JEQ/L3,0,VIA
                -LET/L2,AND(L2,0x80000)
                -JNE/L2,0,RIP
                -MOV/ZR,L1
VIA             -RET
;-------------------------------------------------
;               -EMC/11:Servo OFF
;               -PWO/10:Power ON
;               -CYC/10:Cycle start
```

9.22 RDI: Run Digital Input

RDI (RunDigital Input)

Funzione: effettua un'operazione di test su un segnale d'ingresso digitale e se uguale attiva un task o processo.

Parametri: **a,b,c,d:e1..n:f**

dove:

a = numero o nome dell'input digitale
b = valore dell'input digitale
c = nomefile, nomecall del TASK attivato quando il segnale commuta ad valore logico 1
d = nomefile.pp, nomecall del TASK attivato quando il segnale commuta ad valore logico 0
e 1..n = parametri trasferiti al TASK nelle sue Locali da 1.. n
f = modalità di controllo sull'esistenza del TASK :
0=viene sempre lanciato creando una nuova istanza
1=non viene lanciato , ma aspetta l'uscita di quello attivo per lanciarlo
2=segnala l'errore

Esempio:

```
-RDI/11,1, SignalOne     Effettua una operazione di test sul segnale digitale d'ingresso numero 11 e se 1
                         attiva il task al processo SignalOne
..


PartProgram[SignalOne]
..
```

9.23 RDO: Run Digital Output

RDO (Run Digital Output)

Funzione: effettua un'operazione di test su un segnale d'uscita digitale.

Parametri: **a,b,c,d:e1..n:f**

dove:

a = numero o nome dell'input digitale
b = valore dell'output digitale
c = nomefile, nomecall del TASK attivato quando il segnale commuta ad valore logico 1
d = nomefile.pp, nomecall del TASK attivato quando il segnale commuta ad valore logico 0
e 1..n = parametri trasferiti al TASK nelle sue Locali da 1.. n
f = modalità di controllo sull'esistenza del TASK :
0=viene sempre lanciato creando una nuova istanza
1=non viene lanciato , ma aspetta l'uscita di quello attivo per lanciarlo
2=segnala l'errore

Esempio:

```
-RDO/11,1, SignalOne    Effettua una operazione di test sul segnale digitale d'uscita numero 11 e se  1 attiva il task al processo SignalOne
..

PartProgram[SignalOne]
..
```

CAPITOLO 10

10 Istruzioni di sincronizzazione

.1-EVS Setta degli eventi di sincronizzazione (EVent Set)
.2-EVC Resetta degli eventi di sincronizzazione (EVent Clear)
.3-EVW Attende alcuni eventi di sincronizzazione (EVent Wait)
.4-EVG Legge lo stato degli eventi (EVent Get)
.5-CSA Crea una sincronizzazione per l'uso di un asse tra più task (Create SyncroAxes)
.6-WSA Attende la sincronizzazione per l'uso di un asse tra piu task (Wait Syncro Axes)
.7-DSA Elimina la sincronizzazione per l'uso di un asse tra piu task (Delete Syncro Axes)

10.1 EVS: EVent Set

EVS **EVent Set**

Funzione: da via libera al semaforo (viene considerato avvenuto un evento)

Parametri: **a1..an**

dove:

da a1 a an = = numero del semaforo interessato (1 ÷ 64) (65÷ 128)

Esempio:

-EVS/1,3 Al semaforo 1 e 3.

Note:

a) L'istruzione **EVS** presente in un part program (relativo ad un task in lavorazione), opera parallelamente all'istruzione **EVW** presente su un altro part program (relativo ad un altro task in lavorazione).
Praticamente l'istruzione **EVS**, che viene sempre eseguita immediatamente, da via libera al semaforo mentre l'istruzione **EVW** viene eseguita solamente se il semaforo è libero; in quest'ultimo caso neutralizza l'effetto dell'istruzione **EVS** ribloccando il semaforo.

b) Gli eventi sono 128 di cui i primi 64 nascono settati 0 cioè azzerati, mentre da 65-128 nascono a 1 cioè settati

10.2 EVC: EVent Clear

EVC **EVent Clear**

Funzione: da il blocco al semaforo (viene considerato avvenuto un evento)

Parametri: **a1..an**

dove:

da a1 a an = = numero del semaforo interessato (1 ÷ 64) (65÷ 128)

Esempio:

-EVC/1,3 Al semaforo 1 e 3.

Note:

a) L'istruzione **EVC** presente in un part program (relativo ad un task in lavorazione), opera parallelamente all'istruzione **EVW** presente su un altro part program (relativo ad un altro task in lavorazione).
Praticamente l'istruzione **EVC**, che viene sempre eseguita immediatamente, da uno stop al semaforo mentre l'istruzione **EVW** viene eseguita solamente se il semaforo è libero; in quest'ultimo caso neutralizza l'effetto dell'istruzione **EVC** ribloccando il semaforo.

10.3 EVW: EVent Wait

EVW **EVent Wait**

Funzione: eseguire un test sui semafori e proseguire o meno a seconda che questi siano liberi o bloccati

Parametri: **a1, an:[,b,c]**

dove:

da a1 a an = numero del semaforo interessato (1 ÷ 64) nascono bloccati (65÷ 128)nascono liberi

b = tempo massimo di attesa in millisecondi

c = label di salto se il time out è trascorso

Esempio:

-EVW/1,5,7 Il programma passa all'istruzione successiva solo se i semafori 1,5,7 sono tutti liberi.

-EVW/5:2000, NonArrivaEvento Il programma passa all'istruzione successiva solo se il semaforo 5 è verde (libero) , se dopo 2 sec questo non avviene viene eseguito un salto a lla label "NonArrivaEvento"

Note:

a) L'istruzione **EVW** opera parallelamente all'istruzione **EVS** (vedi quest'ultima istruzione).

b) Ogni qualvolta viene eseguita una istruzione **EVW**, se almeno uno dei semafori interessati non è libero, il programma si arresta su questa istruzione sino a che tutti i semafori non si rendono liberi; a tale punto il programma prosegue con l'istruzione successiva e viene azzerata la condizione di semaforo libero definita dall'istruzione **EVS**.

c) Quando un task è in attesa su di una istruzione **EVW**, questo task non viene più schedulato sino a che in interrupt non interviene la condizione di semaforo libero.
L'utilizzo delle istruzioni **EVS** e **EVW** è quindi molto conveniente ogni qualvolta una istruzione su di un task deve essere eseguita in funzione di una condizione che si verifica su di un altro task.
È ovvio che se un task è in attesa su di una istruzione **EVW** e non interviene una istruzione **EVS** a sbloccarlo, questi rimane bloccato all'infinito,a meno del time out se dichiarato.

10.4 EVG: EVent Get

EVG **EVent Get**

Funzione: Leggere lo stato degli eventi

Parametri: **a1,b1… an,bn**

dove:

da a1 a an = variabili (solo locali o globali) nelle quali si intende memorizzare lo stato dell'evento o semaforo
da b1 a bn = numero dell'evento (semaforo) interessato (1 ÷ 64) (65÷ 128)

Esempio:

-EVG/L1,1,L2,5,L3,7 Carica nella Locale L1 lo stato (1o 0) dell'evento 1, carica nella Locale L2 lo stato (1o 0) dell'evento 5, carica nella Locale L3 lo stato (1o 0) dell'evento 7

Note:

a) L'istruzione **EVG** non interferisce sullo stato di un evento, quindi è da utilizzare esclusivamente come lettura, **non utilizzare come funzione di sincronizzazione**

10.5 CSA: Create SyncroAxes

CSA **Create Syncro Axes**

Funzione: Crea una sincronizzazione per l'uso di un asse tra più task

Parametri: **a,b**

dove:

a	=	numero del sincronismo 1-32
b	=	numero o nome dell'asse interessato

Esempio:

```
-CSA/1:X
```

10.6 WSA: Wait SyncroAxes

WSA **Wait Syncro Axes**

Funzione: Attende la sincronizzazione per l'uso di un asse tra piu task

Parametri: **a**

dove:

a = numero del sincronismo 1-32

Esempio:

```
-WSA/1
```

10.7 DSA: Delete SyncroAxes

DSA **Delete Syncro Axes**

Funzione: Elimina la sincronizzazione per l'uso di un asse tra piu task

Parametri: **a**

dove:

a = numero del sincronismo 1-32

Esempio:

-DSA/1

CAPITOLO 11

11 Istruzioni di servizio

.1-FOC(AZZ)	Azzera il contenuto un file o la crea se non esiste (File Open Create)
.2-FWR	(SCR) Scrive un record su file (File WRite)
.3-FWA	Scrive un record su file (File Write Ascii)
.4-FRD	(LEG) Legge un record da file (File ReaD)
.5-TIM	Temporizzatore in secondi (TIMe)
.6-TMM	Temporizzatore in millesimi di secondo (TiMe Millisecond)
.7-SWA	Inizializza un orologio (Start WAtch)
.8-RWA	Legge un orologio (Read WAtch)
.9-HWA	Ferma un orologio (Halt WAtch)
.10-CWA	Continua un orologio (Continue WAtch)
.11-KYB	Attende un'operazione da tastiera (KeYBoard)
.12-DRT	Visualizza continuamente i valori di assi, globali, segnali (Display Real Time)
.13-DIS	Visualizza una riga di messaggio (DISplay)
.14-HLD	Manda il sistema nello stato di Cycle Stop (HoLD)
.15-PWO	Manda il sistema nello stato di Power ON (PoWerOn)
.16-EMC	Reset del sistema (ReSeT)
.17-LCK	Lock il task ed eventualmente segnala con una SEC
.18-ULK	Rilascia tutti i task in stato di LOCK
.19-RST	Reset del sistema (ReSeT)
.20-SDW	Shut down del sistema (ShutDoWn)
.21-SOR	Ordinamento di una sequenza di valori (SORt)
.22-GTK	Rileva le informazioni inerenti ad un TASK o processo (Get TasK information)
.23-MDI	Esegue un programma ISO - GCODE (ISO)
.24-OTC	Esegue un programma ISO - GCODE (ISO)
.25-ISO	Esegue un programma ISO - GCODE (ISO)
.26-WND	Attende una segnalazione di stato di errore delle risorse assi o mandrino (Wait Notify Detected)
.27-WKY	Attende la premuta di un tasto (Wait KeYboard)
.28-NHL	No Hold
.29-YHL	Yes Hold
.30-GDT	Get Date and Time
.31-GAT	Get Absolute Data e Time
.32-GLN	Get Numero locali

.33-GMI	Get Motion Information
.34-RTC	Read Timer o Counter
.35-SGL	Save Globali
.36-SHL	Shell di applicativi o procedure
.37-G80	Fine ciclo fisso (G80)
.38-G81..89	G81-G89 Attiva il ciclo fisso specificato
.39-ESE	Esegue delle sequenze esterne (Exec Sequence) **sistema ETEL**
.40-ERR	Leggi registri esterni (External Read Registry) **sistema ETEL**
.41-EWR	Scrivi registri esterni(External Write Registry) **sistema ETEL**
.42-ECM	Esegue un comando esterno (External CoMmand) **sistema ETEL**
.43-EWS	Attende una segnalazione (External Wait Signal) **sistema ETEL**
.44-CLM	Comando da Logica di Macchina **sistema ETEL**
.45-SND	Effettua l'emissione di un file Wav sull' uscita audio del PC

11.1 FOC: File Open Create(AZZ)

FOC(AZZ) File Open Create

Funzione: azzera il contenuto un file o lo crea se non esiste

Parametri: **[t:]a**

dove:

t = Tipo se uno (1) il file viene se esiste cancellato e non più creato vuoto (Default 0)
a = Nome del file

Attenzione se nel nome del file vi è il carattere '#' con numero d una variabile LOCALE o GLOBALE, viene comosto con il numero intero indicato.
Esempio:

```
-GDT/L1,L3
-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
-FOC/c:\pippo#L11.txt

-GDT/L1,L3
in L1= 2003
  L2= 12
  L3= 20

-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
in L11=201203
```

Per cui
c:\ pippo#L11.txt il nuovo formato : c:\ pippo201203.txt

Esempio:

-FOC/c:\check\date.dat il file se esiste viene cancellato e ceato vuoto

Esempio:

-FOC/1:c:\check\date.dat il file se esiste viene cancellato

11.2 FWR: File Write(SCR)

FWR(SCR) File WRite

Funzione: scrive un record su

Parametri: **a,[b],[c],[d1,dn] [,e][,f,g,h]**

dove:

a = Nome del file
Attenzione se nel nome del file vi è il carattere '#' con numero d una variabile LOCALE o GLOBALE, viene comosto con il numero intero indicato.
Esempio:

```
-GDT/L1,L3
-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
-FOC/c:\pippo#L11.txt

-GDT/L1,L3
in L1= 2003
  L2= 12
  L3= 20

-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
in L11=201203
```

Per cui
c:\ pippo#L11.txt il nuovo formato : c:\ pippo201203.txt

b = TIME scrive l'ora e il giorno
c = stringa da scrivere
d1,dn = Prima Variabile da scrivere , Ultima variabile da Scrivere
e = Numero progressivo nel record
f = variabile il cui contenuto (valore numerico) si vuole aggiungere dopo la stringa **c**
g = numero di cifre intere da visualizzare (0:10)
h = numero di cifre decimali da visualizzare (0:5)

Note:

a) Se viene messo valore di –1 nel parametro **e** viene visualizzato il valore **c** in esadecimale

Esempi:

```
-FWR/c:\pippo,TIME, cosa, G1, G12, L1,G5,5,-1
-FWR/c:\pippo,TIME, cosa, G1, G12, L1
-FWR/c:\pippo, cosa, G1, G12
-FWR/c:\pippo,TIME, G1, G12
-FWR/c:\pippo, G1, G12
-FWR/c:\pippo,cosa
-FWR/c:\pippo,TIME,cosa
```

Esempio:

```
                -DIS/1:Inizio
                -DIS/2:>
                -AZZ/\DINX1.TXT
                -SCR/\DINX1.TXT,N Pos_X1          Volt
                -LET/L10,0
                -CAP/27:X1,0.1
                -CAP/9:X1,-300
INIZ            -LET/L1,0
                -CAP/51:X1,0                      ; buffer off
                -CAP/50:X1,1,X1,17,X1,14          ; buffer on
POSIZ           -MOV/X1,L1
                -TMM/2
                -LET/L1,L1-2.54*2
                -JGE/L1,-200,POSIZ
                -CAP/51:X1,0                      ; buffer off
                -MOV/X1,0
                -CAP/9:X1,0
LOOPS           -GAV/L11,L12,L1,L4
                -JLE/L11,0,FINE
                -JEQ/L12,0,DOPOS
                -SCR/\DINX1.TXT,Sbordamento,L12,L12
                -DIS/2:Overflow,L12
                -JMP/FINE
DOPOS           -ADD/L10,L11
                -ADD/L9,1
                -SCR/\DINX1.TXT,L1,L3,L9
                -JMP/LOOPS
FINE            -DIS/1:Finito,L10
```

Esempio:

```
                -SFP/10000
                -DIS/1:Inizio
                -DIS/2:>
                -AZZ/\DINX1.TXT
                -SCR/\DINX1.TXT,N  Pos_YEncoder  Pos_YScala
                -RAV/1:L1,Y,L2,YS
                -MOV/Y,-100
                -TMM/500
                -RAV/1:L1,Y,L2,YS
```

```
                -CAP/9:Y,L1,YS,L2          ;Azzeramento dei 2 assi
                -CAP/51:Y,0 ; buffer off
                -CAP/50:Y,1,YS,1 ; buffer on
                -STC/0
                -MOV/Y,-400
                -TMM/2000
                -CAP/51:Y,0 ; buffer off
                -MOV/Y,0
                -CAP/9:Y,0,YS,0  ;Annullamento offset dei 2 assi
                -SET/L9,0,L10,0
LOOPS           -GAV/L11,L12,L1,L2
                -JLE/L11,0,FINE
                -JEQ/L12,0,DOPOS
                -SCR/\DINX1.TXT,Sbordamento,L12,L12
                -DIS/2:Overflow,L12
                -JMP/FINE
DOPOS           -ADD/L10,L11
                -ADD/L9,1
                -SCR/\DINX1.TXT,L1,L2,L9
                -JMP/LOOPS
FINE            -DIS/1:Finito,L10
                -END
```

11.3 FWA: File Write Ascii

FWA **File Write Ascii**

Funzione: scrive un record su file

Parametri: **a,[b:c,d]**
dove:

a = Nome del file
Attenzione se nel nome del file vi è il carattere '#' con numero d una variabile LOCALE o GLOBALE, viene comosto con il numero intero indicato.
Esempio:

```
-GDT/L1,L3
-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
-FOC/c:\pippo#L11.txt

-GDT/L1,L3
in L1= 2003
  L2= 12
  L3= 20

-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
in L11=201203
```

Per cui
c:\ pippo#L11.txt il nuovo formato : c:\ pippo201203.txt

b = stringa contenuta tra " …. " oppure valore numerico
c = numero di cifre intere da visualizzare (0:10)
d = numero di cifre decimali da visualizzare (0:5)

Note:

b) Se viene messo valore di –1 nel parametro **d** viene visualizzato il valore **b** in esadecimale

c) Se il nome del file è FileSystem, alora il nome è costruito con la seguente modalità:
Nel file sistema.txt viene ricercato il percorso

[ParametriGenerali]
…
FileSystem = %FileName.txt

Nel file c:\…….\FileName.txt

[SystemName]
FileName=c:\Prova.txt

Il nome del file sarà perciò c:\Prova.txt

d) Se il primo parametro dopo il nome del file è la parola TIME viene inserita la data all'inizio del file.

Esempio 1:

-FOC/FileSystem
Loop-
-FWA/FileSystem:TIME,"N",L1+2:7,0," G1 ","X ",L1:7,3,"Y ",L2:7,3
-LET/L1,L1+1,L2,L2+1

-JLT/L1,120,Loop

Genera il seguente file:

```
16/12/2007 10:06:43  N2 G1 X 0.000Y 0.000
16/12/2007 10:06:43  N3 G1 X 1.000Y 1.000
16/12/2007 10:06:43  N4 G1 X 2.000Y 2.000
16/12/2007 10:06:43  N5 G1 X 3.000Y 3.000
16/12/2007 10:06:43  N6 G1 X 4.000Y 4.000
16/12/2007 10:06:43  N7 G1 X 5.000Y 5.000
16/12/2007 10:06:43  N8 G1 X 6.000Y 6.000
16/12/2007 10:06:43  N9 G1 X 7.000Y 7.000
16/12/2007 10:06:43  N10 G1 X 8.000Y 8.000
16/12/2007 10:06:43  N11 G1 X 9.000Y 9.000
16/12/2007 10:06:43  N12 G1 X 10.000Y 10.000
16/12/2007 10:06:43  N13 G1 X 11.000Y 11.000
16/12/2007 10:06:43  N14 G1 X 12.000Y 12.000
16/12/2007 10:06:43  N15 G1 X 13.000Y 13.000
```

Esempio 2:

```
Loop-
        -FWA/c:\pipo:"N",L1+2:7,0," G1 ","X ",L1:7,3,"Y ",L2:7,3
        -LET/L1,L1+1,L2,L2+1
        -JLT/L1,120,Loop
```

Genera il seguente file:

```
N2 G1 X 0.000Y 0.000
N3 G1 X 1.000Y 1.000
N4 G1 X 2.000Y 2.000
N5 G1 X 3.000Y 3.000
N6 G1 X 4.000Y 4.000
N7 G1 X 5.000Y 5.000
N8 G1 X 6.000Y 6.000
N9 G1 X 7.000Y 7.000
N10 G1 X 8.000Y 8.000
N11 G1 X 9.000Y 9.000
N12 G1 X 10.000Y 10.000
N13 G1 X 11.000Y 11.000
N14 G1 X 12.000Y 12.000
N15 G1 X 13.000Y 13.000
N16 G1 X 14.000Y 14.000
N17 G1 X 15.000Y 15.000
N18 G1 X 16.000Y 16.000
N19 G1 X 17.000Y 17.000
N20 G1 X 18.000Y 18.000
N21 G1 X 19.000Y 19.000
N22 G1 X 20.000Y 20.000
N23 G1 X 21.000Y 21.000
N24 G1 X 22.000Y 22.000
N25 G1 X 23.000Y 23.000
N26 G1 X 24.000Y 24.000
N27 G1 X 25.000Y 25.000
```

11.4 FRD: File ReaD(LEG)

FRD(LEG) File ReaD

Funzione: legge un record da file

```
//      -FRD/c:\pippo, Posizione Da Leggere ,
//                   , Ultima Posizione Letta( -1 fine file )
//                   , Numero Globali Lette
//                   , G1, G12
```

Parametri: **a,b,c,d,e1,en**

dove:

a = Nome del file
Attenzione se nel nome del file vi è il carattere '#' con numero d una variabile LOCALE o GLOBALE, viene comosto con il numero intero indicato.
Esempio:

```
-GDT/L1,L3
-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
-FOC/c:\pippo#L11.txt

-GDT/L1,L3
in L1= 2003
  L2= 12
  L3= 20

-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
in L11=201203
```

Per cui
c:\ pippo#L11.txt il nuovo formato : c:\ pippo201203.txt

b = Posizione da leggere
c = variabile su cui viene caricato l'ultima posizione letta (se –1 fine file)
d = variabile su cui viene caricato il numero variabili lette
e1,en = Prima variabile da caricare .. Ultima variabile da caricare

Esempio:

```
-FRD/c:\check\date.dat,12,L1,L2,G100,G2001
```

Esempio:

```
          -DIS/Inizio

          -LET/L3,1  ; posizione da leggere
aa        -FRD/c:\dante.txt,L3,L1,L2,L11,L12

          -DIS/1:Valore pos. da leggere =,L3
          -DIS/2:Valore pos. letta      =,L1
          -DIS/3:Valore variabili lette =,L2
          -DIS/4:Valore 1 =,L11
          -DIS/5:Valore 2 =,L12
          -DIS/6:Valore 3 =,L13

          -JEQ/L1,-1,Finito
          -LET/L3,L1+1
          -TMM/100
          -JMP/aa
Finito-
          -DIS/Finito
```

Il file dante.txt contiene i seguenti dati :

```
 1 X 0 Y 0
 2 X -170 Y -260
 3 X -110 Y -190
 4 X -50 Y -60
 5 X 50 Y 60
 6 X 110 Y 190
 7 X 170 Y 260
 8 X -170 Y 260
 9 X -110 Y 190
10 X -50 Y 60
11 X 50 Y -60
12 X 110 Y -190
13 X 170 Y -260
14 X -170 Y -190
15 X -110 Y -120
16 X -50 Y 0
17 X 0 Y 60
18 X 50 Y 190
19 X 110 Y 260
20 X 170 Y 190
21 X 110 Y 120
22 X 50 Y 0
23 X 0 Y -60
24 X -50 Y -190
25 X -110 Y -260
26 X 110 Y -260
27 X 50 Y -190
28 X -110 Y 120
29 X -170 Y 190
30 X -110 Y 260
31 X -50 Y 190
32 X 110 Y -120
33 X 170 Y -190
34 X 50 Y -260
35 X -170 Y 120
36 X -50 Y 260
37 X 170 Y -120
38 X -50 Y -260
39 X -170 Y -120
40 X 50 Y 260
41 X 170 Y 120
42 X -140 Y -225
43 X -80 Y -155
44 X -25 Y -30
45 X 25 Y 30
46 X 80 Y 155
47 X 140 Y 225
48 X -140 Y 225
49 X -80 Y 155
50 X -25 Y 30
51 X 25 Y -30
52 X 80 Y -155
53 X 140 Y -225
54 X -140 Y -155
55 X 140 Y 155
56 X 80 Y 225
57 X -80 Y 225
58 X -140 Y 155
```

59 X 140 Y –155

LEG Leggi File

Funzione: legge un record da file

```
//      -LEG/c:\pippo, Posizione Da Leggere ,
//              , Ultima Posizione Letta( -1 fine file )
//              , Numero Globali Lette
//              , G1, G12
//      -LEG/1:c:\pippo, Posizione Da Leggere
//              , Ultima Posizione Letta( -1 fine file )
//              , Numero Globali CARICATE
//
//              , G1      Tipo 1 = numero
//                              2 = lettera
//                G2      Valore del numero o della lettera
//                ...
//                Gn-1    Tipo 1 = numero
//                              2 = lettera
//                Gn      Valore del numero o della lettera
```

Parametri: [**t:]a,b,c,d,e1,en**

dove:

t = Tipo se uno (1) vengono caricati in modo diverso i parametri el,en

a = Nome del file

Attenzione se nel nome del file vi è il carattere '#' con numero d una variabile LOCALE o GLOBALE, viene comosto con il numero intero indicato.

Esempio:

```
-GDT/L1,L3
-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
-FOC/c:\pippo#L11.txt

-GDT/L1,L3
in L1= 2003
   L2= 12
   L3= 20

-LET/L11,L1-2000,L11,L11+L2*100,L11,L11+L3*10000
in L11=201203
```

Per cui
c:\ pippo#L11.txt il nuovo formato : c:\ pippo201203.txt

b = Posizione da leggere

c = variabile su cui viene caricato l'ultima posizione letta (se –1 fine file)
d = variabile su cui viene caricato il numero variabili lette
e1,en = Prima variabile da caricare .. Ultima variabile da caricare

se
t = Tipo se uno (1) allora
el = 1 numero
2 lettera
el+1 = Valore del numero o della lettera
……..
en = 1 numero
2 lettera
en+1 = Valore del numero o della lettera

Esempio:

```
        -DIS/Inizio

        -LET/L31,1
        -SWA/1


        -LET/L3,1  ; posizione da leggere

aa      -LBF/L11,L22,0
        -LEG/1:c:\temp\Prova.txt,L3,L1,L2,L11,L22
        -JEQ/L1,-1,Finito

        -DIS/1:Valore pos. da leggere =,L3
        -DIS/2:Valore pos. letta      =,L1
        -DIS/3:Valore variabili lette =,L2
        -DIS/4,0:Valore 1 =,L11
        -DIS/4,25:Valore 2 =,L12
        -DIS/5,0:Valore 3 =,L13
        -DIS/5,25:Valore 4 =,L14
        -DIS/6,0:Valore 5 =,L15
        -DIS/6,25:Valore 6 =,L16
        -DIS/7,0:Valore 7 =,L17
        -DIS/7,25:Valore 8 =,L18
        -DIS/8,0:Valore 9 =,L19
        -DIS/8,25:Valore10 =,L20
        -DIS/9,0:Valore11 =,L21
        -DIS/9,25:Valore12 =,L22


        -DIS/10:Record letti,L31
        -TMM/2000
        -LET/L31,L31+1
```

```
        -RWA/L32,1
        -DIS/11:Tempo,L32
        -JLE/L32,0.1,li
        -DIS/12:Quanti sec,L31/L32
li-
        -JEQ/L1,-1,Finito
        -LET/L3,L1+1
        -JMP/aa
Finito-
        -HWA/1
        -RWA/L32,1
        -DIS/11:Quanti sec,L31/L32
        -DIS/10:Tempo,L32
        -DIS/9:Record letti,L31
        -DIS/Finito
```

Il file c:\temp\Prova.txt contiene i seguenti dati :

```
N01 O1
N02 T1 M8
N03 M3 S1000
N04 G4 1000
N05 G00 Z17.5
N06 G01 Z15 F30
N07 G01 Z7 F75
N08 M3 S500
N09 G01 Z6.5 F50
N10 G00 Z93.4
;N07 G80
N11 M28
N12 M5
```

11.5 TIM: TIMe

TIM TIMe

Funzione: effettua un ritardo definito in secondi

Parametri: **a**

dove:

a = ritardo in centesimi di secondo

Esempio:

```
-TIM/100                ; Ritardo di 1 secondo
```

11.6 TMM: TiMe Millisecond

TMM **TiMe Millisecond**

Funzione: effettua un ritardo definito in millisecondi

Parametri: **a**

dove:

a = ritardo in millisecondi

Esempio:

-TMM/10 Ritardo di 10 millisecondi

11.7 SWA: Start WAtch

SWA **Start WAtch**

Funzione: Inizializza un orologio

Parametri: **a**

dove:

a = orologio 1-16

Esempio:

```
-SWA/1
```

11.8 RWA: Read WAtch

RWA **Read WAtch**

Funzione: legge il conteggio di un orologio in una variabile

Parametri: **a,b**

dove:

a = variabile su cui viene letto il valore dell'orologio
b = orologio 1-16

Esempio:

```
-RWA/L1,1
```

11.9 HWA: Halt WAtch

HWA **Halt WAtch**

Funzione: arresta il conteggio di un orologio

Parametri: **a**

dove:

a = orologio 1-16

Esempio:

```
-HWA/1
```

11.10 CWA: Continue WAtch

CWA **Continue WAtch**

Funzione: continua il conteggio di un orologio precedentemente interrotto

Parametri: **a**

dove:

a = orologio 1-16

Esempio:

```
-CWA/1
```

11.11 KYB: KeYBoard

KYB KeYBoard

Funzione: Attende un input da tastiera

Parametri: **a,b[:c,d,e]**

dove:
a = Stringa da visualizzare
b = Globale o Locale da acquisire
c = variabile il cui contenuto (valore numerico) si vuole visualizzare dopo la stringa **a**
d = numero di cifre intere da visualizzare (0:10)
e = numero di cifre decimali da visualizzare (0:5)

Note:

e) Se viene messo valore di –1 nel parametro **e** viene visualizzato il valore **c** in esadecimale

Esempio:

```
-KYB/Input Key,G1
```

Esempio:

```
-LET/L1,123
-KYB/Input Key,G1:L1,7,0     Visualizza :  Input Key  123
```

Note:

a) Viene mandato un evento ai Client EVENTO_ATTESA_DA_TASTIERA

11.12 DRT: DispIay Real Time

DRT **Display Real Time**

Funzione: visualizzare un messaggio con o senza il contenuto di una variabile

Parametri: **a,b,c,[a2,b2,c2..an,bn,cn]**

dove:

a = posizione 1-6
b = Tipo elemento
PR posizione reale asse
PT posizione teorica asse
ER errore asse
G Globale
c = Identificativo elemento

Esempio:

```
-DRT/1,PR,A1,2,G,100
```

11.13 DIS: DlSpIay

DIS DlSplay

Funzione: visualizzare un messaggio con o senza il contenuto di una variabile

Parametri: **a,[b]:c,[d,e,f]**

dove:

a	=	riga di visualizzazione 0 -17
b	=	colonna di inizio visualizzazione 0-127
c	=	messaggio da visualizzare
d	=	variabile il cui contenuto (valore numerico) si vuole visualizzare
e	=	numero di cifre intere da visualizzare (0:10)
f	=	numero di cifre decimali da visualizzare (0:5)

Esempio:

-DIS/2,34;PEZZI ESEGUITI,G2 visualizzare sulla riga 2 (partendo dalla colonna 34) il messaggio PEZZI ESEGUITI = seguito dal contenuto della variabile G2.

Note:

f) Se viene indicata la riga –1nel parametro **a** , viene mandato un evento ai Client EVENTO_DISPLAY
g) Se viene indicata la riga –2 nel parametro **a**, viene mandato un evento ai Client EVENTO_TRACE
h) Se viene messo valore di –1 nel parametro **f** viene visualizzato il valore **d** in esadecimale

Esempio :

```
-DIS/commento=0x,L1,4,-1
                viene visualizzato nella riga 0:
                commento=0x0fc1|->4 cifre esadecimali valore in L1( locale 1)
```

Esempio :

```
a       -LET/L11,L11-1
        -JLE/L11,0,StartNeg
        -DIS/0,0:PlcRunning...
        -JMP/StartCon
StartNeg        -
        -DIS/0,0:
        -JGT/L11,-10,StartCon
        -LET/L11,10
StartCon        -

        -JMP/a
```

11.14 HLD: HoLD

HLD HoLD

Funzione: manda il sistema nello stato di Cycle Stop

Parametri: **[a],[b]:[c],[d,e,f]**

dove:

a	=	riga di visualizzazione 0 -17
b	=	colonna di inizio visualizzazione 0-127
c	=	messaggio da visualizzare
d	=	variabile il cui contenuto (valore numerico) si vuole visualizzare
e	=	numero di cifre intere da visualizzare (0:10)
f	=	numero di cifre decimali da visualizzare (0:5)

Esempio:

```
-HLD                    manda il sottosistema in stato di Cycle Stop
```

11.15 PWO: PoWer On

PWO **PoWer ON**

Funzione: Manda il sistema nello stato di Power ON

Parametri: **[a],[b]:[c],[d,e,f]**

dove:

a	=	riga di visualizzazione 0 -17
b	=	colonna di inizio visualizzazione 0-127
c	=	messaggio da visualizzare
d	=	variabile il cui contenuto (valore numerico) si vuole visualizzare
e	=	numero di cifre intere da visualizzare (0:10)
f	=	numero di cifre decimali da visualizzare (0:5)

Esempio:

```
-PWO
```

11.16 EMC: EMergenCy

EMC **Emergency**

Funzione: Manda il sistema nello stato di Emergenza (EMergenCy)

Parametri: **[a],[b]:[c],[d,e,f]**

dove:

a = riga di visualizzazione 0 -17
b = colonna di inizio visualizzazione 0-127
c = messaggio da visualizzare
d = variabile il cui contenuto (valore numerico) si vuole visualizzare
e = numero di cifre intere da visualizzare (0:10)
f = numero di cifre decimali da visualizzare (0:5)

Esempio:

```
-EMC
```

11.17 LCK: LOCK and set event client

LCK Lock task

Funzione: Lock il task and set event client

Parametri: **[[a],b,c]**

dove:

a = iCookie (default –1)
b = Stringa di comunicazione
c = valore di comunicazione

Esempio:

```
// -LCK/iCookie:MACHINE_LOGIC,1  ; Info
// -LCK/iCookie:MACHINE_LOGIC,2  ; Warning
// -LCK/iCookie:MACHINE_LOGIC,3  ; Allarm
// -LCK/iCookie:MACHINE_LOGIC,4  ; Dinamic Area
// -LCK

        -LCK/MACHINE_LOGIC,3
        -RET

        -WDI/ingresso,1,10000,errore
        ..
        ..
errore  -LCK
```

11.18 ULK: UnLOCK

ULK Lock task

Funzione: Unlock libera i task in LCK

Parametri:

Esempio:

```
-ULK
```

11.19 RST: ReSeT

RST **Reset**

Funzione: Reset del sistema (ReSeT)

Parametri: **a**

dove:
a = Modalità di reset
0 reset assi e mandrini, reset PartProgram, reset Livelli IO, reset Socket, reset Seriali
1 reset assi e mandrini
2 reset assi e mandrini, reset Livelli IO, reset Socket, reset Seriali

Esempio:

```
-RST/1          ; reset assi e mandrini
```

11.20 SWN: Shut down

SWN **Shut down**

Funzione: Shut down del sistema (ShutDoWn)

Parametri:

dove:

Esempio:

```
-SDW
```

11.21 SOR: SOrt

SOR SORt

Funzione: Ordinamento di una sequenza di valori (SORt)

Parametri: **a:b,c**

dove:
a = Modalità di ordinamento
1 ordina dal piu grande
–1 ordina dal piu piccolo
b = Indirizzo Inizio variabile
c = Indirizzo Fine variabile

Esempio:

```
-LET/L1,93,L2,48,L3,4,L4,7,L5,8
-LET/L6,12,L7,9,L8,76,L9,200,L10,45,L11,95,L12,58
-SOR/1:L1,L12                                   ; ordina dal piu grande
-DIS/Valore L1 ,L1
-DIS/1:Valore L2 ,L2
-DIS/2:Valore L3 ,L3
-DIS/3:Valore L4 ,L4
-DIS/4:Valore L5 ,L5
-DIS/5:Valore L6 ,L6
-DIS/6:Valore L7 ,L7
-DIS/7:Valore L8 ,L8
-DIS/8:Valore L9 ,L9
-DIS/9:Valore L10,L10
-DIS/10:Valore L11,L11
-DIS/11:Valore L12,L12
```

11.22 GTK: Get TasK information

GTK **Get TasK information**

Funzione: Rileva le informazioni inerenti ad un TASK o processo (Get TasK information)

Parametri: **a:b,[c,d,e]**

dove:
a = nomefile.pp , nomecall :
b = Variabile dove viene caricato: 1 tsk attivo, 0 tsk non attivo
c = Variabile dove viene caricato: il numero locali
d = Prima variabile dove viene caricato il valore della prima locale
e = Ultima variabile dove viene caricato il valore della ennesima locale

Esempio:

```
-GTK/TaskName:L1,L2,L10,L32
```

L1 Variabile dove viene caricato: 1 tsk attivo, 0 tsk non attivo
L2 Variabile dove viene caricato: il numero locali
L10 Prima variabile dove viene caricato il valore della prima locale L1 del part program TaskName
L32 Ultima variabile dove viene caricato il valore della locale L22 del part program TaskName

11.23 MDI: Esegue una istruzione ISO

MDI **Esegue una istruzione ISO**

Funzione: Esegue una istruzione ISO - GCODE (ISO)

Parametri: **a:btesto itruzione iso**

a = ID (Identificativo numerico) del processo ISO
oppure nome del processo ISO
b = testo dell'istruzione

Esempio:

```
-MDI/ Scara:X123Y45F1000
```

11.24 OTC: Carica campo della tabella ISO

OTC Carica campo della tabella Origine -Tool –Correttore -Parametrica ISO

Funzione: Carica i campi della tabella Origine-Tool -Correttore – Parametrica ISO - GCODE (ISO)

Parametri: **a:b:c:d1,e1……dn ,en**

a = ID (Identificativo numerico) del processo ISO
oppure nome del processo ISO
b = O oppure T oppure C oppure P oppure R
c = Numero della tabella

d1..n = Identificativo campo della tabella o Valore del campo della Tabella P
el..n = Valore del campo della Tabella O ,T,C,R

Nota 1 caso speciale :
scambio di record della tabella interessata
-OTC/3:T,1,T2,2
Scambia il record della tabella utensili 1 con il 2

Nota 2 nel caso R viene letto nella variabile il valore di O attivo
Esempio
-OTC/3:R,L1
-DIS/-1:Origine,L1

Gli identificativi dei campi delle tabelle sono:

Tabella Origini:
X origine asse X
Y origine asse Y
Z origine asse Z
A origine asse A
B origine asse B
C origine asse C
U origine asse U
V origine asse V
W origine asse W

Attenzione il valore dell'offset viene posto a zero

DX offset asse X
DY offset asse Y
DZ offset asse Z
DA offset asse A
DB offset asse B
DC offset asse C
DU offset asse U
DV offset asse V
DW offset asse W

Tabella Tool (utensili):

LC Lunghezza Tool
RT Raggio Tool
Free1 Campo 1
Free2 Campo 2
Free3 Campo 3
Free4 Campo 4
Size
Spec
NM
PM
PD
CP
SC
Diametro
Passo
Inch

Tabella Corretori:

LC Lunghezza Tool
RT Raggio Tool
Free1 Campo 1
Free2 Campo 2
Free3 Campo 3
Free4 Campo 4

Esempio:

```
-OTC/ Scara:O ,1:X,234
-OTC/ Scara:O ,1:DX,0.1
-OTC/ Scara:T ,2:LC,11
-OTC/ Scara:C ,9:LC,23.8
-OTC/ Scara:P ,10:23.8
```

11.25 ROP: Legge campo della tabella ISO

ROP **Legge campo della tabella Origine -Tool -Correttore -Parametrica ISO**

Funzione: Legge un campo della tabella Origine-Tool –Correttore- Parametrica ISO - GCODE (ISO)

Parametri: **a:b:c:d1,e1……dn ,en**

a = ID (Identificativo numerico) del processo ISO
oppure nome del processo ISO
b = O oppure T oppure C oppure P
c = Numero della tabella

d1..n = Locale o Globale su cui caricare il valore del campo
el..n = Identificativo campo della tabella o nulla per della Tabella P

Nota 1 nel caso che si voglia leggere l'origine o il tool o il correttore allora si usa b e c scambiati

Esempio

```
-ROP/3:L1,O
-DIS/-1:Origine,L1
-ROP/3:L1,T
-DIS/-1:Utensile,L1
-ROP/3:L1,C
-DIS/-1:Correttore,L1
```

Gli identificativi dei campi delle tabelle sono:

Tabella Origini:

X registro origine asse X
Y registro origine asse Y
Z registro origine asse Z
A registro origine asse A
B registro origine asse B

C registro origine asse C
U registro origine asse U
V registro origine asse V
W registro origine asse W
DX offset asse X
DY offset asse Y
DZ offset asse Z
DA offset asse A
DB offset asse B
DC offset asse C
DU offset asse U
DV offset asse V
DW offset asse W

Per ottenere il valore dell'origine di un asse bisogna eseguire la seguente somma:

Origine asse = registro origine asse – offset asse

Tabella Tool (utensili):

LC	Lunghezza Tool
RT	Raggio Tool
Free1	Campo 1
Free2	Campo 2
Free3	Campo 3
Free4	Campo 4
Size	
Spec	
NM	
PM	
PD	
CP	
SC	
Diametro	
Passo	
Inch	

Tabella Corretori:

LC	Lunghezza Tool
RT	Raggio Tool
Free1	Campo 1
Free2	Campo 2
Free3	Campo 3
Free4	Campo 4

Esempio:

```
-ROP/ Scara:O,1:L1,X
-ROP/ Scara:T,2:L1,LC
-ROP/ Scara:C ,9:L1,LC
-ROP/ Scara:P ,10:L1
```

11.26 ISO: Esegue un part program ISO

ISO **Esegue un part program ISO**

Funzione: Esegue un programma ISO - GCODE (ISO)

Parametri: **a,b:[c:d,e,f:g,h:i,l:m,n]**

dove:

a = nomefile.prg oppure parola chiave **ISOCNC** (Vedi Nota 1)

b = ID (Identificativo numerico) del processo ISO che eseguirà il programma che si deve trovare nell directory di sistema CNC;
oppure nome del processo ISO in questo caso la directory del file . prg è la directory del processo ISO definito nel "sistema.txt"

c = Time di attesa completamento lavorazione : 0 non attende Default 0
-1 infinito
> 0 tempo di attesa

d,e,f = offset x,y,z

g,h = parametri da passare da P1-Pn

i,l = parametri ritorno da P1-Pn

m = nomefile, nomecall del TASK attivato quando il processo ISO è interrotto da una anomalia assi

n = modalità di controllo sull'esistenza del TASK :
0=viene sempre lanciato creando una nuova istanza
1=non viene lanciato
2=segnala l'errore

Nota 1: se al posto del nome di un programma ISO, viene indicata la parola chiave **ISOCNC** il sistema eseguirà il programma selezionato nel modulo CNC dell'ambiente CNC, correspettivo al processo ISO identificato dall'ID.

Esempio:

modo 1 (esplicito) -ISO/A.PRG,Scara:-1:0,0,0:L141,L145:L146,L149:,SistemaInErrore

modo 2 (directory CNC) -ISO/CAVALLO.PRG,13:-1:0,0,0:L141,L145:L146,L149:,SistemaInErrore

modo 3 (ultimo selezionanato) -ISO/ISOCNC,3:-1:0,0,0:L141,L145:L146,L149:,SistemaInErrore

in caso di anomalia sugli assi viene eseguito il Task "SistemaInErrore"

i parametri di ingresso uscita possono essere omessi -ISO/FresaB.PRG,2:-1:0,0,0::,0_ErroreISO

Esempio chiamata di un part program ISO con orientamento da Visione

```
                -DIM/150
INIZIO  -
                -SAX/101,0
                -MOV/X,3,Y,0
                -LET/L1,0               // Left
                -LET/L2,0
                -LET/L3,1               // Numero integrazioni
                -LET/L4,10              // Size blob
                -LET/L5,0               // Shape


                -DIS/1:  Inizio Measure
                -DIS/2:
                -DIS/3:
                -DIS/4:
                -DIS/5:
                -DIS/6:
;
                -EVW/71
                -SWA/1

                -JEQ/L1,0,Cam2
//
//  Numero telecamere
//
Cam1            -LET/L11,1      // iNumberCameraActive
                -JMP/ViaPr

Cam2            -LET/L11,2      // iNumberCameraActive
ViaPr           -ARI/1,12,VisAlgo,SetActiveCamera,L11,L11,L31,L31


//
//  Leggi immagine
//
                -LET/L11,0              // iTipoPalette,
                -LET/L12,L3             // Numero integrazioni
;               -ARI/1,12,VisionBase,ReadMultiImages,L11,L12,L31,L31
                -ARI/1,12,VisionBase,ReadImage,L11,L12,L31,L31
////////////////////////////////////////
//      Prendi 1 blob               //
////////////////////////////////////////
                -LET/L11,-215           // iLeft,
                -LET/L12,200            // iTop,
                -LET/L13,55             // iWidth,
                -LET/L14,55             // iHeight,
                -LET/L15,77             // 77 GetTreshold
                -JEQ/L2,0,DopoNegativo1
                -JEQ/L2,1,DopoPositivo1
                -JMP/fine
DopoNegativo1   -LET/L16,1              // iModeLowParam,
                -JMP/Continua1
DopoPositivo1   -LET/L16,0              // iModeLowParam,
                -JMP/Continua1
Continua1       -LET/L17,2              // iModeHighParam Tipo di Sogliatore Funny
                -LET/L18,0              // iTypeBlob   0 vuoto 1 pieno
                -LET/L19,200            // iLowArea
                -LET/L20,0              // iTypeFilter
```

```
                -LET/L21,0              // iFilterParam
                -LET/L22,1              // Distanza minima dal centro
                -LET/L23,30
                -ARI/1,12,VisAlgo,GetAllBlobs,L11,L23,L31,L31
                -LET/L48,L31
                -LET/L49,1
LoopB1          -JGT/L49,L48,NOBLOB
//
//  Valore blob
//
                -LET/L11,L49     // iNumberBlobBySize
                -ARI/1,12,VisAlgo,GetValueBlob,L11,L11,L31,L51
                -LET/L50,L31    //
                -LET/L51,L32    // X Baricentro geometrico
                -LET/L52,L33    // Y Baricentro geometrico
                -LET/L53,L34    // Area
                -LET/L54,L35    // Perimetro
                -JMP/contin1
                -JLT/L34,20000,contin
                -LET/L49,L49+1
                -JMP/LoopB1



//
//  Cerchio blob
//
contin1  -LET/L11,L49            // iNumberBlobBySize ,
                -LET/L12,L141   // dRaggioNoto,
                -ARI/1,12,VisAlgo,GetCircleBlob,L11,L12,L30,L38
                -LET/L55,L31    // Raggio
                -LET/L56,L32    // X Centro
                -LET/L57,L33    // Y Centro
                -LET/L58,L34     // Rotondita

//
//  Disegna blob
//
                -LET/L11,L49            // iNumberBlobBySize ,
                -LET/L12,0              // iFlagBlobPicture,
                -LET/L13,80             // uRGBPictureColor,
                -LET/L14,1              // iFlagBlobEdge,
                -LET/L15,008            // uRGBEdgeColor,
                -LET/L16,0              // iFlagValue,
                -LET/L17,0              // iFlagDimension,
                -LET/L18,17             // iFlagCircle,
                -LET/L19,0              // iFlagXAxesRect,
                -LET/L20,0              // iFlagYAxesRect,
                -LET/L21,1              // iFlagBordoROI,
                -LET/L22,0              // iFlagBordoInternoROI,
                -LET/L23,0              // iFlagCoordinateROI,
                -LET/L24,0              // iFlagEllisse,
                -ARI/1,12,VisAlgo,DrawBlob,L11,L24,L30,L30
//
//  Testo delle quote su video
//
                -LET/L11,L51-70 // iX,
                -LET/L12,L52            // iY,
                -LET/L13,0              // iFlagStringa
                -LET/L14,0              // pStringa,
                -LET/L15,0              // iFlagAng,
```

```
                -LET/L16,0              // iFlagAngMilliToSessa,
                -LET/L17,L55            // Raggio,
                -LET/L18,1              // iFlagX,
                -LET/L19,L56            // dX,
                -LET/L20,1              // iFlagY,
                -LET/L21,L57            // dY,
                -LET/L22,2              // Numero font,
                -LET/L23,888            // colore,
                -ARI/1,12,VisAlgo,DrawTextOnVideo,L11,L23,L30,L30
////////////////////////////////////////
//        Prendi 2 blob               //
////////////////////////////////////////
                -LET/L11,145            // iLeft,
                -LET/L12,200            // iTop,
                -LET/L13,55             // iWidth,
                -LET/L14,55             // iHeight,
                -LET/L15,77             // 77 GetTreshold
                -JEQ/L2,0,DopoNegativo2
                -JEQ/L2,1,DopoPositivo2
                -JMP/fine
DopoNegativo2   -LET/L16,1              // iModeLowParam,
                -JMP/Continua2
DopoPositivo2   -LET/L16,0              // iModeLowParam,
Continua2       -LET/L17,2              // iModeHighParam Tipo di Sogliatore Funny
                -LET/L18,0              // iTypeBlob   0 vuoto 1 pieno
                -LET/L19,200            // iLowArea
                -LET/L20,0              // iTypeFilter
                -LET/L21,0              // iFilterParam
                -LET/L22,1              // Distanza minima dal centro
                -LET/L23,30
                -ARI/1,12,VisAlgo,GetAllBlobs,L11,L23,L31,L31

                -LET/L48,L31
                -LET/L49,1
LoopB2          -JGT/L49,L48,NOBLOB
//
//  Valore blob
//
                -LET/L11,L49     // iNumberBlobBySize
                -ARI/1,12,VisAlgo,GetValueBlob,L11,L11,L31,L51
                -LET/L60,L31    //
                -LET/L61,L32    // X Baricentro geometrico
                -LET/L62,L33    // Y Baricentro geometrico
                -LET/L63,L34    // Area
                -LET/L64,L35    // Perimetro
                -JMP/contin2
                -JLT/L34,20000,contin
                -LET/L49,L49+1
                -JMP/LoopB2
//
//  Cerchio blob
//
contin2         -LET/L11,L49     // iNumberBlobBySize ,
                -LET/L12,L141   // dRaggioNoto,
                -ARI/1,12,VisAlgo,GetCircleBlob,L11,L12,L30,L38
                -LET/L65,L31    // Raggio
                -LET/L66,L32    // X Centro
                -LET/L67,L33    // Y Centro
                -LET/L68,L34     // Rotondita
//
//  Disegna blob
//
```

```
                -LET/L11,L49            // iNumberBlobBySize ,
                -LET/L12,0              // iFlagBlobPicture,
                -LET/L13,80             // uRGBPictureColor,
                -LET/L14,1              // iFlagBlobEdge,
                -LET/L15,008    // uRGBEdgeColor,
                -LET/L16,0              // iFlagValue,
                -LET/L17,0              // iFlagDimension,
                -LET/L18,17             // iFlagCircle,
                -LET/L19,0              // iFlagXAxesRect,
                -LET/L20,0              // iFlagYAxesRect,
                -LET/L21,1              // iFlagBordoROI,
                -LET/L22,0              // iFlagBordoInternoROI,
                -LET/L23,0              // iFlagCoordinateROI,
                -LET/L24,0              // iFlagEllisse,
                -ARI/1,12,VisAlgo,DrawBlob,L11,L24,L30,L30
//
//  Testo delle quote su video
//
                -LET/L11,L61-70 // iX,
                -LET/L12,L62            // iY,
                -LET/L13,0              // iFlagStringa
                -LET/L14,0              // pStringa,
                -LET/L15,0              // iFlagAng,
                -LET/L16,0              // iFlagAngMilliToSessa,
                -LET/L17,L65            // Raggio,
                -LET/L18,1              // iFlagX,
                -LET/L19,L66            // dX,
                -LET/L20,1              // iFlagY,
                -LET/L21,L67            // dY,
                -LET/L22,2              // Numero font,
                -LET/L23,888            // colore,
                -ARI/1,12,VisAlgo,DrawTextOnVideo,L11,L23,L30,L30


////////////////////////////////////////
/         Prendi 3 blob               //
////////////////////////////////////////
                -LET/L11,-215           // iLeft,
                -LET/L12,-145           // iTop,
                -LET/L13,55              // iWidth,
                -LET/L14,55             // iHeight,
                -LET/L15,77             // 77 GetTreshold
                -JEQ/L2,0,DopoNegativo3
                -JEQ/L2,1,DopoPositivo3
                -JMP/fine
DopoNegativo3   -LET/L16,1              // iModeLowParam,
                -JMP/Continua3
DopoPositivo3   -LET/L16,0              // iModeLowParam,
                -JMP/Continua3
Continua3 -
                -LET/L17,2               // iModeHighParam Tipo di Sogliatore Funny
                -LET/L18,0              // iTypeBlob   0 vuoto 1 pieno
                -LET/L19,200            // iLowArea
                -LET/L20,0              // iTypeFilter
                -LET/L21,0              // iFilterParam
                -LET/L22,1              // Distanza minima dal centro
                -LET/L23,30
                -ARI/1,12,VisAlgo,GetAllBlobs,L11,L23,L31,L31
                -LET/L48,L31
                -LET/L49,1
LoopB3          -JGT/L49,L48,NOBLOB
//
```

```
//  Valore blob
//
                -LET/L11,L49            // iNumberBlobBySize
                -ARI/1,12,VisAlgo,GetValueBlob,L11,L11,L31,L51
                -LET/L70,L31            //
                -LET/L71,L32    // X Baricentro geometrico
                -LET/L72,L33            // Y Baricentro geometrico
                -LET/L73,L34    // Area
                -LET/L74,L35    // Perimetro
                -JMP/contin3
                -JLT/L34,20000,contin
                -LET/L49,L49+1
                -JMP/LoopB3
//
//  Cerchio blob
//
contin3         -LET/L11,L49                    // iNumberBlobBySize ,
                -LET/L12,L141   // dRaggioNoto,
                -ARI/1,12,VisAlgo,GetCircleBlob,L11,L12,L30,L38
                -LET/L75,L31            // Raggio
                -LET/L76,L32    // X Centro
                -LET/L77,L33            // Y Centro
                -LET/L78,L34            // Rotondita
//
//  Disegna blob
//
                -LET/L11,L49            // iNumberBlobBySize ,
                -LET/L12,0              // iFlagBlobPicture,
                -LET/L13,80             // uRGBPictureColor,
                -LET/L14,1              // iFlagBlobEdge,
                -LET/L15,008            // uRGBEdgeColor,
                -LET/L16,0              // iFlagValue,
                -LET/L17,0              // iFlagDimension,
                -LET/L18,17             // iFlagCircle,
                -LET/L19,0              // iFlagXAxesRect,
                -LET/L20,0              // iFlagYAxesRect,
                -LET/L21,1              // iFlagBordoROI,
                -LET/L22,0              // iFlagBordoInternoROI,
                -LET/L23,0              // iFlagCoordinateROI,
                -LET/L24,0              // iFlagEllisse,
                -ARI/1,12,VisAlgo,DrawBlob,L11,L24,L30,L30

//
//  Testo delle quote su video
//
                -LET/L11,L71-70 // iX,
                -LET/L12,L72            // iY,
                -LET/L13,0              // iFlagStringa
                -LET/L14,0              // pStringa,
                -LET/L15,0              // iFlagAng,
                -LET/L16,0              // iFlagAngMilliToSessa,
                -LET/L17,L75            // Raggio,
                -LET/L18,1              // iFlagX,
                -LET/L19,L76            // dX,
                -LET/L20,1              // iFlagY,
                -LET/L21,L77            // dY,
                -LET/L22,2              // Numero font,
                -LET/L23,888            // colore,
                -ARI/1,12,VisAlgo,DrawTextOnVideo,L11,L23,L30,L30
////////////////////////////////////////
//        Prendi 4 blob               //
////////////////////////////////////////
```

```
                -LET/L11,145            // iLeft,
                -LET/L12,-145           // iTop,
                -LET/L13,55             // iWidth,
                -LET/L14,55             // iHeight,
                -LET/L15,77             // 77 GetTreshold
                -JEQ/L2,0,DopoNegativo4
                -JEQ/L2,1,DopoPositivo4
                -JMP/fine
DopoNegativo4   -LET/L16,1              // iModeLowParam,
                -JMP/Continua4
DopoPositivo4   -LET/L16,0              // iModeLowParam,
                -JMP/Continua4
Continua4 -
                -LET/L17,2               // iModeHighParam Tipo di Sogliatore Funny
                -LET/L18,0              // iTypeBlob   0 vuoto 1 pieno
                -LET/L19,200            // iLowArea
                -LET/L20,0              // iTypeFilter
                -LET/L21,0              // iFilterParam
                -LET/L22,1              // Distanza minima dal centro
                -LET/L23,30
                -ARI/1,12,VisAlgo,GetAllBlobs,L11,L23,L31,L31
                -LET/L48,L31
                -LET/L49,1
LoopB4  -JGT/L49,L48,NOBLOB
//
//  Valore blob
//
                -LET/L11,L49            // iNumberBlobBySize
                -ARI/1,12,VisAlgo,GetValueBlob,L11,L11,L31,L51
                -LET/L80,L31             //
                -LET/L81,L32    // X Baricentro geometrico
                -LET/L82,L33    // Y Baricentro geometrico
                -LET/L83,L34    // Area
                -LET/L84,L35    // Perimetro
                -JMP/contin4
                -JLT/L34,20000,contin
                -LET/L49,L49+1
                -JMP/LoopB4
//
//  Cerchio blob
//
contin4  -LET/L11,L49     // iNumberBlobBySize ,
                -LET/L12,L141    // dRaggioNoto,
                -ARI/1,12,VisAlgo,GetCircleBlob,L11,L12,L30,L38
                -LET/L85,L31             // Raggio
                -LET/L86,L32    // X Centro
                -LET/L87,L33    // Y Centro
                -LET/L88,L34            // Rotondita
//
//  Disegna blob
//
                -LET/L11,L49            // iNumberBlobBySize ,
                -LET/L12,0              // iFlagBlobPicture,
                -LET/L13,80             // uRGBPictureColor,
                -LET/L14,1              // iFlagBlobEdge,
                -LET/L15,008            // uRGBEdgeColor,
                -LET/L16,0              // iFlagValue,
                -LET/L17,0              // iFlagDimension,
                -LET/L18,17             // iFlagCircle,
                -LET/L19,0              // iFlagXAxesRect,
                -LET/L20,0              // iFlagYAxesRect,
                -LET/L21,1              // iFlagBordoROI,
```

```
                -LET/L22,0              // iFlagBordoInternoROI,
                -LET/L23,0              // iFlagCoordinateROI,
                -LET/L24,0              // iFlagEllisse,
                -ARI/1,12,VisAlgo,DrawBlob,L11,L24,L30,L30
//
//  Testo delle quote su video
//
                -LET/L11,L81-70 // iX,
                -LET/L12,L82            // iY,
                -LET/L13,0              // iFlagStringa
                -LET/L14,0              // pStringa,
                -LET/L15,0              // iFlagAng,
                -LET/L16,0              // iFlagAngMilliToSessa,
                -LET/L17,L85            // Raggio,
                -LET/L18,1              // iFlagX,
                -LET/L19,L86            // dX,
                -LET/L20,1              // iFlagY,
                -LET/L21,L87            // dY,
                -LET/L22,2              // Numero font,
                -LET/L23,888            // colore,
                -ARI/1,12,VisAlgo,DrawTextOnVideo,L11,L23,L30,L30
//
// retta tra due centri
//
                -LET/L11,(L86-L56)*(L86-L56)
                -LET/L12,(L87-L57)*(L87-L57)
                -LET/L13, sqr(L11+L12)  // Raggio
                -JLE/L13,0.001,noele1
                -LET/L22,(L87-L57)/L13
                -LET/L23,(L86-L56)/L13
                -LET/L24,-1
                -LET/L21,(L22*L24*L56)+(L23*L57)
;               -ARI/1,12,VisAlgo,DrawElementLine,L21,L23,L20,L20
noele1  -
                -LET/L11,(L76-L66)*(L76-L66)
                -LET/L12,(L77-L67)*(L77-L67)
                -LET/L13, sqr(L11+L12)  // Raggio
                -JLE/L13,0.001,noele2
                -LET/L22,(L77-L67)/L13
                -LET/L23,(L76-L66)/L13
                -LET/L24,-1
                -LET/L21,(L22*L24*L66)+(L23*L67)
;               -ARI/1,12,VisAlgo,DrawElementLine,L21,L23,L20,L20
noele2  -
                -LET/L31,200    // MM X
                -LET/L32,200    // MM y
                -LET/L41,L66-L56 // DELTA Xsup
                -LET/L42,L86-L76 // DELTA Xsup
                -LET/L43,abs(L41+L42)/2
                -LET/G501,1
                -JLE/L43,0.001,nomisx
                -LET/G501,L31/L43
nomisx  -
                -LET/L41,L77-L57 // DELTA Ysin
                -LET/L42,L87-L67 // DELTA Ydes
                -LET/L43,abs(L41+L42)/2
                -LET/G502,1
                -JLE/L43,0.001,nomisy
                -LET/G502,L32/L43
nomisy  -
                -DIS/10:Valore pixel X,G501
                -DIS/11:Valore pixel Y,G502
```

```
//
//  Prendi blob
//
                    -LET/L11,-215               // iLeft,
                    -LET/L12,200                // iTop,
                    -LET/L13,415                // iWidth,
                    -LET/L14,400                // iHeight,
                    -LET/L15,77                 // 77 GetTreshold
                    -JEQ/L2,0,DopoNegativo
                    -JEQ/L2,1,DopoPositivo
                    -JMP/fine
DopoNegativo        -LET/L16,1                  // iModeLowParam,
                    -JMP/Continua
DopoPositivo        -LET/L16,0                  // iModeLowParam,
                    -JMP/Continua
Continua            -LET/L17,2                  // iModeHighParam Tipo di Sogliatore Funny
                    -LET/L18,0                  // iTypeBlob   0 vuoto 1 pieno
                    -LET/L19,200                // iLowArea
                    -LET/L20,0                  // iTypeFilter
                    -LET/L21,0                  // iFilterParam
                    -LET/L22,1                  // Distanza minima dal centro
                    -LET/L23,30
                    -ARI/1,12,VisAlgo,GetAllBlobs,L11,L23,L31,L31
                    -LET/L48,L31
                    -LET/L49,1
LoopB               -JGT/L49,L48,NOBLOB



//
//  Valore blob
//
                    -LET/L11,L49                // iNumberBlobBySize
                    -ARI/1,12,VisAlgo,GetValueBlob,L11,L11,L31,L51
                    -LET/L50,L31                //
                    -LET/L51,L32      // X Baricentro geometrico
                    -LET/L52,L33                // Y Baricentro geometrico
                    -LET/L53,L34      // Area
                    -LET/L54,L35      // Perimetro
                    -LET/L55,L36                // X Baricentro gravita
                    -LET/L56,L37      // Y Baricentro gravita
                    -LET/L57,L38      // Area di gravita
                    -JMP/contin
                    -JLT/L34,20000,contin
                    -LET/L49,L49+1
                    -JMP/LoopB
//
//  Cerchio blob
//
contin              -LET/L11,L49                // iNumberBlobBySize ,
                    -LET/L12,0.1                // dPixelGap,
                    -ARI/1,12,VisAlgo,GetCircleBlob,L11,L12,L30,L38
                    -LET/L61,L31      // Raggio
                    -LET/L62,L32      // X Centro
                    -LET/L63,L33      // Y Centro
                    -LET/L64,L34                // Rotondita

                    -LET/L11,2                  // iTypeFirstElement, // 1=line 2=radius
                    -LET/L12,2                  // dLineDistance dCircleRadius ,
                    -LET/L13,2                  // dLineSin dCircleXCenter
```

```
                -LET/L14,2              // dLineCos dCircleYCenter
                -LET/L15,2               // iTypeSecondElement, // 1=line 2=radius
                -LET/L16,2              // dLineDistance dCircleRadius ,
                -LET/L17,2              // dLineSin dCircleXCenter
                -LET/L18,2              // dLineCos dCircleYCenter
                -ARI/1,12,VisAlgo,PointsAcross2Elements,L11,L18,L30,L36
                -LET/L111,L31   // piNumberPoint,
                -LET/L112,L32           // pPAX,
                -LET/L113,L33   // pPAY,
                -LET/L114,L34   // pPBX,
                -LET/L115,L35   // pPBY,
                -LET/L116,L36   // pDist
                -LET/L11,L49    // iNumberBlobBySize ,
                -ARI/1,12,VisAlgo,GetEllipseBlob,L11,L11,L30,L44
                -LET/L101,L31    // pdXc
                -LET/L102,L32    // pdYc
                -LET/L103,L33    // pdAngA
                -LET/L104,L34    // pdRgA
                -LET/L105,L35    // pdAngB
                -LET/L106,L36    // pdRgB
                -LET/L107,L37    // pdX1
                -LET/L108,L38   // pdY1
                -LET/L109,L39   // pdX2
                -LET/L110,L40   // pdX2
                -LET/L111,L41   // pdX3
                -LET/L112,L42   // pdY3
                -LET/L113,L43   // pdX4
                -LET/L114,L44   // pdX4
//
//  Dimensione blob
//
                -LET/L11,L49            // iNumberBlobBySize ,
                -ARI/1,12,VisAlgo,GetDimensionBlob,L11,L11,L30,L46
                -LET/L71,L31            // XHigh
                -LET/L72,L32            // YHigh
                -LET/L73,L33            // XLow
                -LET/L74,L34            // YLow
                -LET/L75,L35    // XRigh
                -LET/L76,L36    // YRight
                -LET/L77,L37    // XLeft
                -LET/L78,L38    // YLeft
                -LET/L79,L39    // XHighAlongBaricenter
                -LET/L80,L40    // YHighAlongBaricenter
                -LET/L81,L41    // XLowAlongBaricenter
                -LET/L82,L42    // YLowAlongBaricenter
                -LET/L83,L43    // XRightAlongBaricenter
                -LET/L84,L44    // YRightAlongBaricenter
                -LET/L85,L45    // XLeftAlongBaricenter
                -LET/L86,L46    // YLeftAlongBaricenter
//
//  Disegna blob
//
                -LET/L11,L49            // iNumberBlobBySize ,
                -LET/L12,0              // iFlagBlobPicture,
                -LET/L13,80             // uRGBPictureColor,
                -LET/L14,1              // iFlagBlobEdge,
                -LET/L15,008            // uRGBEdgeColor,
                -LET/L16,1              // iFlagValue,
                -LET/L17,0              // iFlagDimension,
                -LET/L18,0              // iFlagCircle,
                -LET/L19,0              // iFlagXAxesRect,
                -LET/L20,0              // iFlagYAxesRect,
```

```
                -LET/L21,1              // iFlagBordoROI,
                -LET/L22,1              // iFlagBordoInternoROI,
                -LET/L23,0              // iFlagCoordinateROI,
                -LET/L24,0x61           // iFlagEllisse,
                -ARI/1,12,VisAlgo,DrawBlob,L11,L24,L30,L30
//
//  Testo delle quote su video
//
                -LET/L11,L51-70         // iX,
                -LET/L12,L52            // iY,
                -LET/L13,0              // iFlagStringa
                -LET/L14,0              // pStringa,
                -LET/L15,1              // iFlagAng,
                -LET/L16,0              // iFlagAngMilliToSessa,
                -LET/L17,L103           // dAng,
                -LET/L18,0              // iFlagX,
                -LET/L19,L51            // dX,
                -LET/L20,0              // iFlagY,
                -LET/L21,L52            // dY,
                -LET/L22,2              // Numero font,
                -LET/L23,000            // colore,
                -ARI/1,12,VisAlgo,DrawTextOnVideo,L11,L23,L30,L30
;
;  Show l'immagine
;
                -ARI/1,12,VisionBase,ShowImage,L11,L11,L30,L31
                -HWA/1
                -RWA/L1,1
                -DIS/Orologio,L1
                -LET/G71,0
                -MOV/X,L51*G501,Y,L52*G502
                -DIS/1:X pixel,L101
                -DIS/2:Y pixel,L102
                -DIS/3:Ang rilevato ,L103
                -LET/L141,L101*G501     ; Delta X
                -LET/L142,L102*G502     ; Delta Y

                -LET/L143,L103-166.2    ; Angolo
                -LET/L144,24.5*G501             ; Delta X Nominale
                -LET/L145,-7.5*G502             ; Delta Y Nominale

                -DIS/4:X Reale mm ,L141
                -DIS/5:Y Reale mm,L142
                -DIS/6:Ang Rotazione ,L143


                -EVS/71
                -ISO/CAVALLOECS.PRG,11:-1:0,0,0:L141,L145:L146,L149
                -MOV/X,0,Y,0

                -JMP/INIZIO

NOBLOB          -LET /L62,0     // X Centro
                -LET /L63,0     // Y Centro
                -LET /L53,0     // X Baricentro gravita
                -LET /L54,0     // Y Baricentro gravita
                -LET /L61,0     // Raggio
                -LET /L61,0     // Raggio
                -LET /L77,0     // XLeft
                -LET /L72,0     // YHigh
                -LET /L75,0     // XRigh
                -LET /L74,0     // YLow
```

```
-LET/G71,0
-LET/L141,0,L142,0,L143,0,L144,0,L145,0

-EVS/71
-ISO/CAVALLOECS.PRG,11:-1:0,0,0:L141,L145:L146,L149


-JMP/INIZIO
```

11.27 WND: Wait Notify Detected

WND **Wait Notify Detected**

Funzione: Attende una segnalazione di stato di errore delle risorse assi
(Wait Notify Detected)

Parametri: **a,b,c:[d,e]**

dove:

a	=	maschera degli errori di cui si vuole la segnalazione oppure 0 per tutte le segnalazioni di anomalia
b	=	Variabile locale dove viene caricato il numero asse o mandrino o gruppo macchina se maschera degli errori = 0x80000000
c	=	Variabile locale dove viene caricato il tipo errore o maschera dei gruppi macchina se maschera degli errori = 0x80000000
e	=	Timeout in millisecondi
f	=	Label di salto per timeout scaduto

Caricando la maschera errore o segnalazione viene attivata la richiesta di segnalazione se un asse qualsiasi va in errore con una di quelle modalità

Maschera degli errori di cui si vuole la segnalazione

0	Tutte 7 le possibili anomalie
0x8	ASSE IN SERVO ERROR (ASSE FERMO)
0x10	ASSE IN COLLISION (ASSE IN MOVIMENTO)
0x20	ASSE NON IN POSIZIONE
0x40	ASSE IN BLOCCO CONTA
0x2000000	ASSE DRIVER FAULT
0x4000000	ASSE IN OVER TRAVEL POSITIVO
0x8000000	ASSE IN OVER TRAVEL NEGATIVO

caso speciale :
0x80000000 ANOMALIA GRUPPI MACCHINA

Il tipo di anomalia asse viene restituito nella variabile locale **b** con i seguenti valori:

TIPO ANOMALIA	
1	ASSE IN SERVO ERROR (ASSE FERMO)
2	ASSE IN COLLISION (ASSE IN MOVIMENTO)
3	ASSE NON IN POSIZIONE
4	ASSE IN BLOCCO CONTA
5	ASSE DRIVER FAULT
6	ASSE IN OVER TRAVEL POSITIVO
7	ASSE IN OVER TRAVEL NEGATIVO

Esempio 1:

```
-WND/0x8,L2,L3:1000,LabelTimeout

0x8 ASSE IN SERVO ERROR (ASSE FERMO)
L2 = numero asse o mandrino
L3 = tipo errore
```

Esempio 2:

```
A       -LET/L1,0
        -WND/L1,L2,L3
```

Nota: se la maschera degli errori richiesti (parametro **a**) è:

0x80000000		ANOMALIA GRUPPI MACCHINA nei parametri **b** e **c**
b	=	Il numero del gruppo macchina in anomalia asse
c	=	La maschera di tutti i gruppi macchina in anomalia

Esempio 3:

```
-WND/0x80000000,L2,L3:1000,LabelTimeout

L2 = numero gruppo macchina in anomalia
L3 = maschera gruppi macchina in anomalia
```

11.28 WKY: Wait KeYboard

WKY **Wait KeYboard**

Funzione: Attende la premuta di un tasto (Wait KeYboard)

Parametri: **a**

dove:

a = SCAN code del dasto da attendere

Esempio:

```
-WKY/1 ;   Attende il tasto ESC

-WKY/16 il Tasto Q
-WKY/17 il Tasto W
-WKY/18 il Tasto E
-WKY/1   il Tasto ESC
-WKY/29 il Tasto Ctrl
-WKY/56 il Tasto Alt
-WKY/28 il Tasto Enter
-WKY/73 il Tasto PGup
-WKY/81 il Tasto Pgdown
```

Tabella dei Codici Ascii Estesi corrispondenti ai System Scan Code.(per Consolle ECS)

RESET	0x6B
AXISH	0x71
START	0x6E
REL	0x6D
HOLD	0x70

Tabella dei System Scan Code (in uscita dal Keyboard Controller)

SP1	0x72
SP2	0x73
SP3	0x74
SP4	0x75
SP5	0x77
SP6	0x78
SP7	0x79
SP8	0x55
SGOUP	0x58

11.29 NHL: NoHoLd

NHL **NoHoLd command**

Funzione: Disabilita l'attuazione di istruzioni TKM/1: da altri part program

Parametri: **nessuno**

L'istruzione serve per disabilitare delle porzioni di part program dala possibilità di essere messe nello stato HOD da parte di altri part program tramite l'istruzione TKM/1:…..
Alla fine della parte di codice critico, tramite l'istruzione YHL si può ritornare nelle condizioni normali se durante l'esuzione della parte tra NHL.. YHL, è stata richiesta l'operazione di Hold tale funzione verrà attivata immediatamente dopo YHL.

Esempio:

```
-NHL
..
.               Zona non sospendibile da TKM/1:….
…
…
-YHL
```

11.30 YHL: YesHoLd

YHL **YesHoLd command**

Funzione: Abilita l'attuazione di istruzioni TKM/1: da altri part program disabilitata dal'istruzione NHL

Parametri: **nessuno**

L'istruzione di YHL è la complementare della NHL, viene utilizzata per ripristinare lo stato normale di sospensionabilità tramite l'istruzione TKM/1: … attivata da altri part program che operano in parallelo.

Esempio:

```
-NHL
..
.          Zona non sospendibile da TKM/1:….
…
…
-YHL
```

11.31 GDT: Get Date and Time

GDT **Get Date and Time**

Funzione: Restituisce nelle varaibili L o G la data e l'ora attuale

Parametri: **a,b**

dove:

a = Prima Variabile Locale o Globale
b = Ultima Variabile Locale o Globale

Nella a = anno
a+1 = mese
a+2 = giorno
a+3 = ora
a+4 = minuti
a+5 = secondi
a+6 = millisecondi
a+7 = secondi dal 1 gennaio 1970

Esempio:

```
-GDT/L1,L8
-DIS/1:Anno                =,L1,4,0
-DIS/2:Mese                =,L2,4,0
-DIS/3:Giorno              =,L3,4,0
-DIS/4:Ora                 =,L4,4,0
-DIS/5:Minuti              =,L5,4,0
-DIS/6:Secondi             =,L6,4,0
-DIS/7:Millisecondi        =,L7,4,0
-DIS/8:Secondi dal 1970    =,L8,11,0
```

11.32 GAT: Get Absolute data eTime

GAT **Get Absolute data eTime**

Funzione: Restituisce nelle varaibili L o G la data e l'ora attuale

Parametri: **a,b,c**

dove:

a = Data in secondi dal 1 gennaio 1970
b = Prima Variabile Locale o Globale
c = Ultima Variabile Locale o Globale

Nella b = secondi
b+1 = minuti
b+2 = ora
b+3 = giorno
b+4 = mese
b+5 = anno

Esempio:

```
-GAT/135678: L1,L6
-DIS/1:Anno          =,L6,4,0
-DIS/2:Mese          =,L5,4,0
-DIS/3:Giorno        =,L4,4,0
-DIS/4:Ora           =,L3,4,0
-DIS/5:Minuti        =,L2,4,0
-DIS/6:Secondi       =,L1,4,0
```

11.33 GLN: Get Local Number

GLN **Get Local Number**

Funzione: Restituisce nella varaibile L o G il numero di variabili Locali del part program

Parametri: **a**

dove:

a = Variabile Locale o Globale in cui viene restituito il numero di locali definite nel part program

Esempio:

-GLN/L1
-DIS/1:Numero locali definite =,L1,4,0

11.34 GMI: Get Motion Information

GMI **Get Motion Information**

Funzione: Restituisce nella varaibili L o G il numero di enti o movimenti eseguiti e da fare di un part program di automazione o di un unità ISO

Parametri: **a:b,c[,d,e,f]**

dove:

a = Label,Nome file del task (part program) da chiudere
[nomecall][,nomefile.pp]
[nomefileISO.prg]
[numerotesta ISO]
b = Variabile Locale o Globale movimenti caricati
c = Variabile Locale o Globale movimenti caricati movimenti eseguiti
d = Variabile Locale o Globale Feed vettoriale
e = Variabile Locale o Globale maschera dei movimenti
f = Variabile Locale o Globale stato dell'operazione 1 se trovato il part program o 0

Esempio 1:

```
Tool-
        -TMM/10
        -GMI/,Cos_Crash:L1,L2,L3,L4  ; Part program Cos_Crash.PP
        -DIS/1:Movimenti Fatti,L2
        -DIS/2:Movimenti Indrodottti,L1
        -DIS/3:Feed,L3
        -DIS/4:Delta, L1-L2
        -JMP/Tool
```

Esempio 2:

```
Tool-
        -TMM/10
        -GMI/11:L1,L2,L3,L4       ; 11 Numero testa ISO
        -DIS/1:Movimenti Fatti,L1
        -DIS/2:Movimenti Indrodottti,L2
        -DIS/3:Feed,L3

        -JMP/Tool
```

11.35 RTC: Read Timer and Counter

RTC **Read Timer e Conter**

Funzione: Restituisce nella varaibili L o G il valore di un Timer (TON e TOR) o di un Counter(CTU e CUD)

Parametri: **a,b[,c,d]**

dove:

a = Variabile Locale o Globale contenente il valore del timer o del contatore
b = T o C con il relativo numero
c = Valore massimo impostato
d = Stato 1 o 0 del timer o contatore

Esempio:

-RTC/L1,T1,L2,L3
-DIS/1:Valore del timer attuale =,L1,7,0
-DIS/2:Valore del timer finale =,L2,7,0
-DIS/1:Valore dello stato del timer =,L3,7,0

Parametri: **a:b,c[,d,e]**

dove:

a = Tipo di lettura
1=Timer
2=Counter
3=Counter esterno 1-6 contatori
1-3 su 8255 prima PCI board
4-6 su 8255 seconda PCI board

b = Variabile Locale o Globale contenente il valore del timer o del contatore
c = T o C con il relativo numero
d = Valore massimo impostato
e = Stato 1 o 0 del timer o contatore

Esempio:

```
        -TSK/Clock1
        -SDO/Gate0,1
        -LET/L10,20
Loop-
        -TMM/L10
        -RTC/3:L1,1
        -DIS/1:Conteggio = ,L1,12,0
        -LET/L3,(L1-L2)*1000/L10*60
        -LET/L2,L1
        -DIS/-1:Velocità =,L3/1000,7,3
        -DIS/2:Velocità =,L3/1000,12,3
        -JMP/Loop

PartProgram[Clock1]
        -SDO/Gate0,1
Loop-
        -TMM/1
        -SDO/CLK0,1
        -TMM/1
        -SDO/CLK0,0
        -JMP/Loop
```

11.36 SGL: Save GLobal

SGL **Save GLobal**

Funzione: Salva su disco con "commit" le variabili Globali, le quali sono persistenti su "hard disk"

Parametri: nessuno

Esempio:

-SGL

Vengono salvate sul HardDisk tutte le 32768 variabili Globali con una tecnica che assicura la scrittura in modlità „Commit" tale da garantire il completamento della scrittura mantenendo comunque il sistema precedente in caso di anomalia del sistema.

11.37 SHL: SHeLl

SHL SHeLl

Funzione: Esegue una "Shell" di un applicativo (.EXE) o di una procedura (.BAT) del sistema operativo Windows.

Parametri: [a]:b[,c,d]

dove:
a = Attende il completamento della SHELL se 1 oppure 0 non attende
(Default = 0 non attende)
b = Nome della procedura (Bat) o dell'eseguibile/applicativo (.EXE)
c = Visualizzazione se 1 della shell oppure 0 se non si desidera la visualizzazione.
(Default = 1 Visualizza)
d = Modalità di visualizzazione:
0=SW_SHOW (Default 0= SW_SHOW)
1= SW_SHOWMAXIMIZED
2= SW_SHOWMINIMIZED
3= SW_RESTORE
4= SW_HIDE

-SHL/Wait:Calc.exe,Show,ModeShow

Esempio:

-SHL/Calc.exe

-SHL/Esplorer.exe

-SHL/C:AxesBrain\Save.bat

11.38 G80: Fine ciclo fisso

G80 **Fine ciclo fisso**

Funzione: Fine ciclo fisso (G80)

Parametri:

dove:

Esempio:

```
-G80
```

11.39 G81..89 Attiva il ciclo fisso specificato

G81..89 Attiva il ciclo fisso specificato

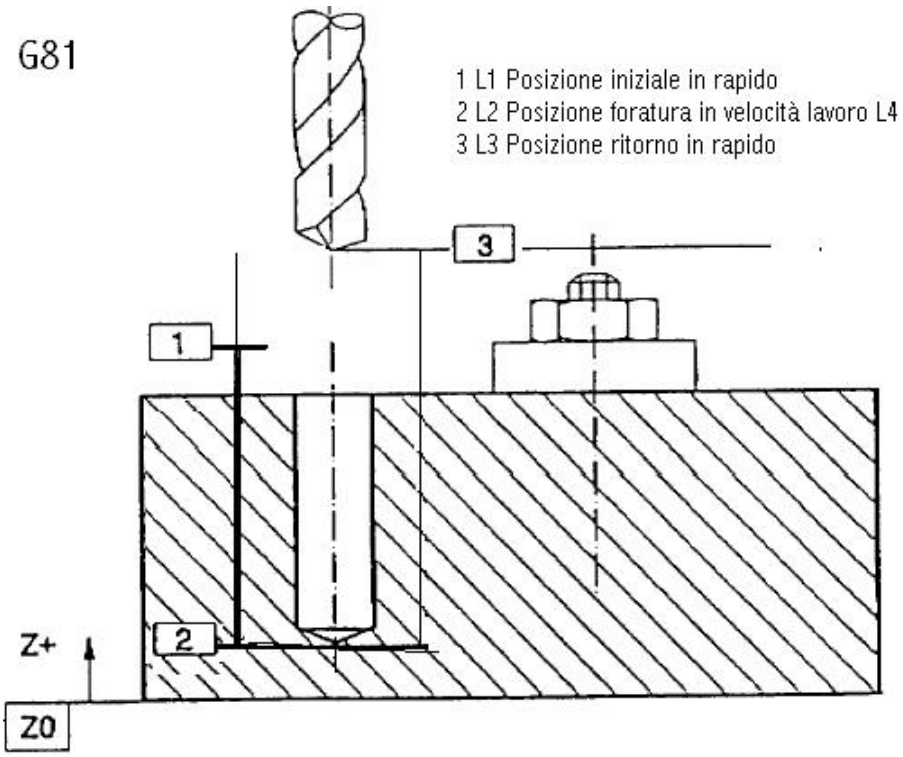
Funzione: G81-G89 Attiva il ciclo fisso specificato

Parametri:

dove:

Esempio:

```
        -G81..89

        -G81/Z:L1,L2,L3,L4      ( L1=PosZ_Inizio L2=PosZ_Foratura L3=PosZ_Ritorno L4=VelZ_Foratura )

        -G82/Z:L1,L2,L3,L4,L5   ( L1=PosZ_Inizio L2=PosZ_Foratura L3=PosZ_Ritorno L4=VelZ_Foratura
                                  L5= Tempo in Millisec )
        -G83/Z: L1= PosZ_Inizio
                L2= PosZ_Foratura
                L3= PosZ_Ritorno
                L4= VelZ_Foratura
                L5= Tempo in Millisec
                L6= Primo incremento da PosZ_Inizio
                L7= Incremento Successivo
                L8= Incremento Sicurezza

;
-G84/Z,M1:L2,L3,L4,L5,L6

L1= PosZ_Inizio
L2= PosZ_Foratura
L3= PosZ_Ritorno
L4= Passo
L5= Perc . riduzione velocità per compensatore o zero per maschiatura rigida
L6= Speed mandrino
  -G84/Z,Sp:120,200,80,1.5,12,300 ; Maschiatura con compensatore
  -G84/Z,Sp:120,140,120,1.5,0,300 ; Maschiatura rigida
  -MOV/X12,Y12
  -MOV/X12,Y24
  -MOV/X12,Y12
  -G80
```



G81 Foratura

-G81/Z:L1,L2,L3,L4

L1=PosZ_Inizio
L2=PosZ_Foratura
L3=PosZ_Ritorno
L4=VelZ_Foratura

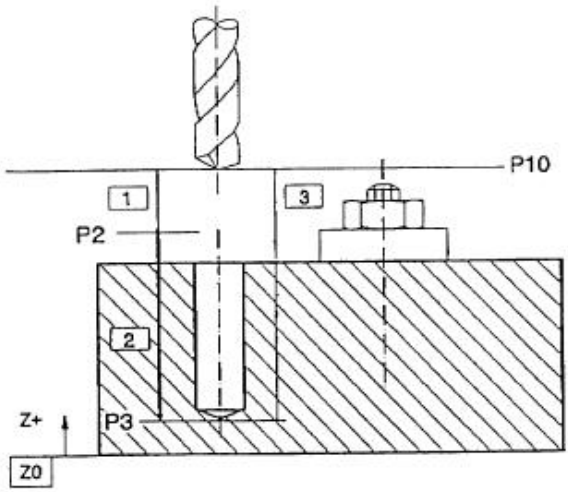


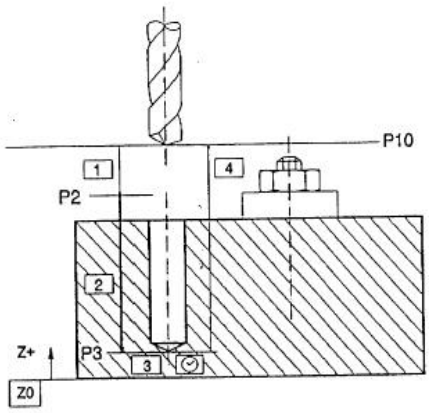
Fig. 7.1-2: Drilling cycle G81

Sequence of the drilling cycle G81:
1. Rapid traverse in the Z direction to the reference plane (P2).
2. Drill to the final depth required (P3) using the current feed rate.
3. Pull out in rapid traverse to the retract plane (P10).

G82 Foratura con sosta

-G82/Z:L1,L2,L3,L4,L5

L1=PosZ_Inizio
L2=PosZ_Foratura
L3=PosZ_Ritorno
L4=VelZ_Foratura
L5= Tempo in Millisec



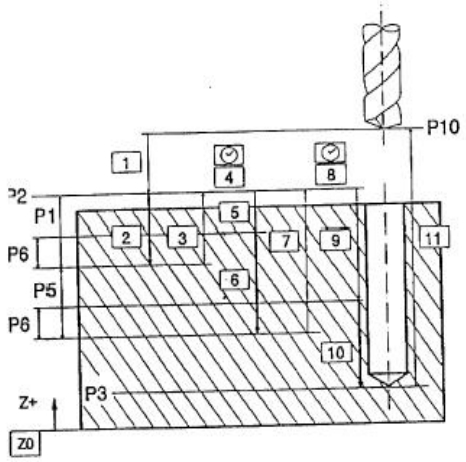
– **Dwell time**

Fig. 7.1-3: Drilling cycle G82

Sequence of the drilling cycle G82:
1. Rapid traverse in the Z direction to the reference plane (P2).
2. Drill to the final hole depth (P3) using the current feed rate.
3. Wait for the dwell time (P4) to elapse before breaking contact with workpiece.
4. Pull out in rapid traverse to the retract plane (P10).

G83 Foratura con rottura trucciolo

-G83/Z: L1= PosZ_Inizio
L2= PosZ_Foratura
L3= PosZ_Ritorno
L4= VelZ_Foratura
L5= Tempo in Millisec
L6= Primo incremento da PosZ_Inizio
L7= Incremento Successivo
L8= Incremento Sicurezza



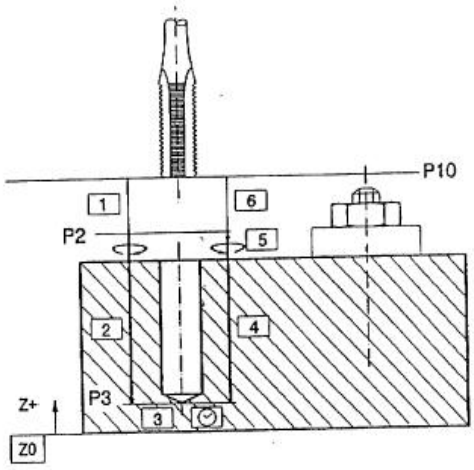
– Dwell time

Fig. 7.1-4: Drilling cycle G83

Sequence of the drilling cycle G83:
1. Rapid traverse in the Z direction to the reference plane (P2).
2. Drill using the current feed rate with the first feed-in value (P1) to the depth 1.
3. Pull out in rapid traverse to the reference plane (P2).
4. To allow the drill bit to cool, the Z axis remains on the reference plane (P2) during the dwell time (P4).
5. Move in rapid traverse to P1-P6 (first feed-in minus safety clearance) in the hole.
6. Drill to the depth 2: P6+P5 (safety clearance plus feed-in) using the current feed rate.

G84 Maschiatura

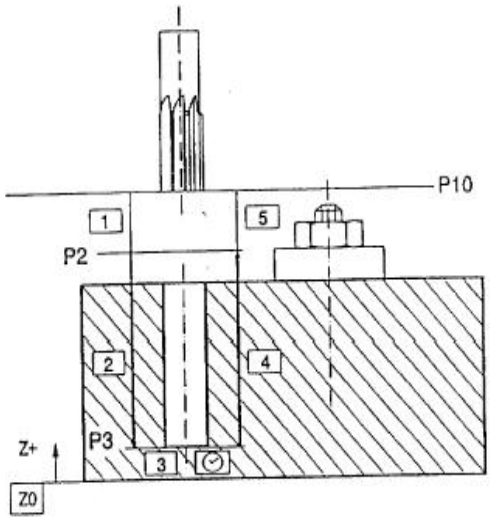
-G84/Z,Mandrino:
L1= quota rapido entrata 1
L2= quota lavoro finale 3
L3= quota rapido uscita 6
L4= Passo
L5=Perc.Stiramentoo 0 Rigida
L6= Speed Mandrino



– Dwell time

Fig. 7.1-5: Drilling cycle G84

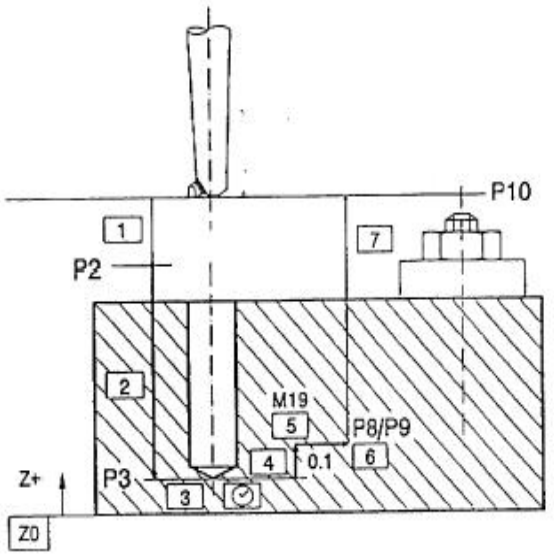
Sequence of the drilling cycle G84:
1. Rapid traverse in the Z direction to the reference plane (P2).
2. Drill using the current feed rate and clockwise rotating spindle (M03) to the final hole depth (P3).
3. Reverse spindle, i.e. the direction of rotation changes; a pause is made for the dwell time (P4).
4. Pull out using the current feed rate to the reference plane (P2).
5. Reverse spindle, i.e. spindle's direction of rotation is again clockwise.
6. Move in rapid traverse to the retract plane (P10).



= Dwell time

Fig. 7.1-6: Drilling cycle G85

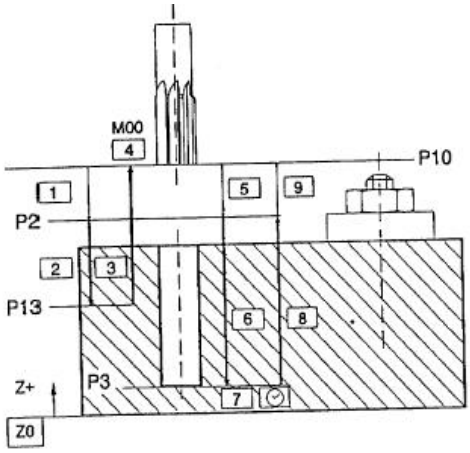
Sequence of the drilling cycle G85:
1. Rapid traverse in the Z direction on the reference plane (P2).
2. Drill using the current feed rate to the final hole depth (P3).
3. Wait for the dwell time (P4) to elapse.
4. Pull out using the current feed rate to the reference plane (P2).
5. Move in rapid traverse to the retract plane (P10).



= Dwell time

Fig. 7.1-7: Drilling cycle G86

Sequence of the drilling cycle G86:
1. Rapid traverse in the Z direction to the reference plane (P2).
2. Bore to the final hole depth (P3) using the current feed rate.
3. Wait for the dwell time (P4) to elapse.
4. Move away 0.1 mm using the current feed rate.
5. Spindle is orientated to 0 degrees (M19).
6. Spindle is moved in the X or Y axis by the lift distance (P8 or P9).
7. Pull out to the retract plane (P10) in rapid traverse.



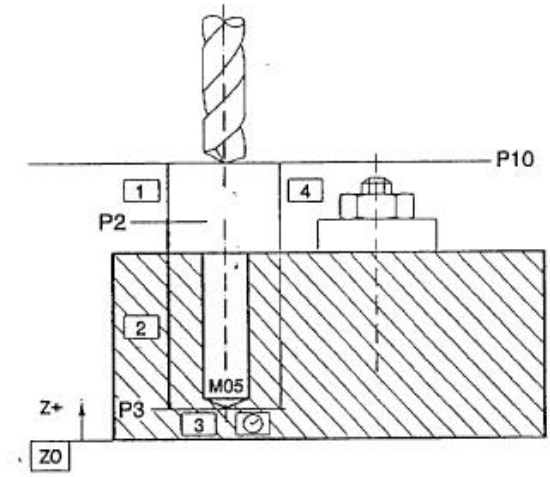
= **Dwell time**

Fig. 7.1-8: Drilling cycle G87
Sequence of the drilling cycle G87:

1. Rapid traverse in the Z direction to the reference plane (P2).
2. Ream with the processing feed rate (P11) to the first reamed depth (P13).
3. Pull out to the retract plane (P10) with the retract feed rate (P12).
4. Halt feed rate to allow measuring of the hole, press START to continue with the processing.
5. Rapid traverse to the reference plane (P2).
6. Ream with the processing feed rate (P11) to the final hole depth (P3).
7. Wait for the dwell time (P4) to elapse.
8. Pull out with the retract feed rate (P12) to the reference plane (P2).
9. Move to the retract plane (P10) in rapid traverse.

Attention:

After leaving the drilling cycle G87 the retract feed rate is active!

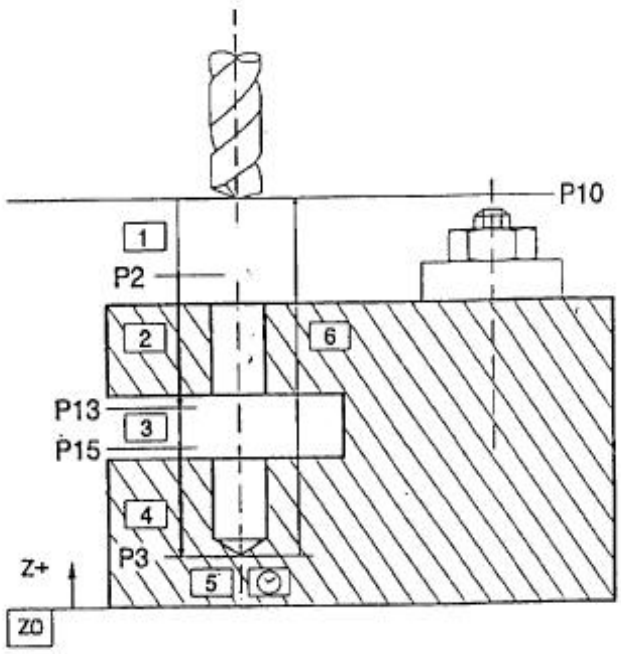


= **Dwell time**

Fig. 7.1-9: Drilling cycle G88

Sequence of the drilling cycle G88:

1. Rapid traverse in the Z direction to the reference plane (P2).
2. Bore to the final hole depth (P3) using the current feed rate.
3. Wait for the dwell time (P4) to elapse, after that the spindle stops.
4. Pull out to the retract plane (P10) in rapid traverse with stopped spindle.



= **Dwell time**

Fig. 7.1-10: Drilling cycle G89

Sequence of the drilling cycle G89:

1. Rapid traverse in the Z direction to the reference plane (P2).
2. Bore to the first drilling depth (P13) using the current feed rate.
3. Rapid traverse in the Z direction to the second drilling plane (P15).
4. Bore to the final hole depth (P3) using the current feed rate.
5. Wait for the dwell time (P4) to elapse.
6. Pull out in rapid traverse to the retract plane (P10).

TABELLE di conversione per istruzioni per il sistema ETEL

DMD_TYP_NONE	0	no type
DMD_TYP_IMMEDIATE	0	disabled or immediate value
DMD_TYP_USER	1	user registers
DMD_TYP_PPK	2	drive parameters
DMD_TYP_MONITOR	3	monitoring registers
DMD_TYP_SEQUENCE	5	sequence buffer
DMD_TYP_TRACE	6	trace buffer
DMD_TYP_ADDRESS	7	address value
DMD_TYP_LKT	8	movement lookup tables
DMD_TYP_TRIGGER	9	triggers buffer
DMD_TYP_REALTIME	10	realtime buffer
DMD_TYP_HALL_LKT	11	hall lookup tables (deprecated)
DMD_TYP_Y	11	Y type variables
DMD_TYP_FLOAT	12	float register

convertion constants

DMD_CONV_DWORD	0	double word value without conversion
DMD_CONV_BOOL	1	boolean value
DMD_CONV_INT	2	integer value without conversion
DMD_CONV_LONG	3	long integer value without conversion
DMD_CONV_K8	3	
DMD_CONV_K14	3	
DMD_CONV_STRING	4	packed string value
DMD_CONV_FLOAT	5	float value
DMD_CONV_UPI	10	user position increment
DMD_CONV_USI	11	user speed increment
DMD_CONV_UAI	12	acceleration, user acceleration increment
DMD_CONV_DPI	15	drive position increment
DMD_CONV_DSI	16	drive speed increment
DMD_CONV_DAI	17	drive acceleration increment
DMD_CONV_C13	20	current 13bit range
DMD_CONV_C14	21	current 14bit range
DMD_CONV_C29	22	current 29bit range
DMD_CONV_C15	23	current 15bit range
DMD_CONV_CUR2	25	i², dissipation value
DMD_CONV_CUR2T	26	i²t, integration value
DMD_CONV_M82	28	current limit in 10 mA unit, 100 <=> 1.0A
DMD_CONV_STI	30	slow time increment (500us-2ms)

DMD_CONV_FTI	31	fast time increment (125us-166us)
DMD_CONV_CTI	32	current loop time increment (41us)
DMD_CONV_EXP10	33	ten power factor
DMD_CONV_HSTI	34	half slow time increment
DMD_CONV_M242	35	quartz frequency in Hz
DMD_CONV_QZTIME	36	interrupt time in sec = inc / m242
DMD_CONV_SPEC2F	37	filter time, T = [fti] * (2ⁿ-1)
DMD_CONV_TEMP	38	2⁰ = 1 <=> 1.0
DMD_CONV_UFTI	39	user friendly time increment
DMD_CONV_AVI	40	analog voltage increment -8192 <=> 10V 8192 <=> -10V
DMD_CONV_VOLT	41	2⁰ = 1 <=> 1.0
DMD_CONV_ENCOFF	42	11bit with 2048 offset
DMD_CONV_VOLT100	43	(2⁰)/100 = 1 <=> 1.0
DMD_CONV_PH11	44	2¹¹ = 2048 <=> 360°
DMD_CONV_PH12	45	2¹² = 4096 <=> 360°
DMD_CONV_PH28	46	2²⁸ = 65536*4096 <=> 360°
DMD_CONV_AVI12BIT	47	analog voltage increment 2048 <=>10V 2048 <=> -10V
DMD_CONV_AVI16BIT	48	analog voltage increment 32767 <=>10V-32768 <=> -10V
DMD_CONV_BIT0	50	2⁰ = 1 <=> 1.0
DMD_CONV_BIT5	55	2⁵ = 32 <=> 1.0
DMD_CONV_BIT8	58	2⁸ = 256 <=> 1.0
DMD_CONV_BIT9	59	2⁹ = 512 <=> 1.0
DMD_CONV_BIT10	60	2¹⁰ = 1024 <=> 1.0
DMD_CONV_BIT11	61	2¹¹ = 2048 <=> 1.0
DMD_CONV_BIT15	65	2¹⁵ = 32768 <=> 1.0
DMD_CONV_BIT24	74	2²⁴ = 256*65536 <=> 1.0
DMD_CONV_BIT31	81	2³¹ = 32768*65536 <=> 1.0
DMD_CONV_BIT11P2	82	
DMD_CONV_UFPI	85	user friendly position increment
DMD_CONV_UFSI	86	user friendly speed increment
DMD_CONV_UFAI	87	user friendly acceleration increment
DMD_CONV_MSEC	88	milliseconds
DMD_CONV_K1	90	pl prop gain, k(A/m) = k1 * Iref * dpi_factor / 2²⁹
DMD_CONV_K2	92	pl speed feedback gain, k(A/(m/s)) = k2 * Iref * dsi_factor ²⁹
DMD_CONV_K4	94	pl integrator gain, k(A/(m*s)) = k1 * Iref * dpi_factor / 2²⁹ / pl_time
DMD_CONV_K5	96	anti-windup K[m/A]=K5*4096/(dpi_factor * Iref)
DMD_CONV_K9	98	1st order filter in pl
DMD_CONV_K10	100	1st order filter in s.

DMD_CONV_K20	102	speed feedback, sec unit (m/(m/s)), F = k20 / 2^{16-k50} * dsi_factor / dpi_factor
DMD_CONV_K21	104	speed feedback, sec unit (m/(m/s²)), F = k20 / 2^{24-k50} * dai_factor / dpi_factor
DMD_CONV_K23	106	commutation phase advance period/(m/s)
DMD_CONV_K75	108	encoder multiple index distance, 1/1024 * encoder perion unit
MD_CONV_K80	110	cl prop gain delta[1/A]
MD_CONV_K81	112	cl prop integrator delta[1/(A*s)]
MD_CONV_K82	114	filter time, T = [cti] * (2ⁿ-1)
MD_CONV_K94	116	time in 2x current loop increment
MD_CONV_K95	118	current rate for k95
MD_CONV_K96	120	phase rate for k96
MD_CONV_K205	122	came value, 100 <=> 1.0
MD_CONV_PER_1000	123	per thousand unit
MD_CONV_K239	124	motor Kt factor in mN(m)/A, 1000 <=> 1.0mN(m)/A

Sistema ETEL fast :

Nella configurazione DSTEB3 i comandi implementati sono leggermente diversi dai comandi per la DSMAX ed in particolare:

1) Funzione esegui comandi ETEL (vedi tabella comandi Etel)

-ESE/NomeAsse:Numero comando,L#,L#+1

esegue la funzione ETEL :

param1= L#
param2= L#+1

execute_command_s(NomeAsse, cmd , param1 , param2, FAST)

Esempio: per eseguire l'esecuzione di una sequenza il cui numero è 1 sull'asse X2

-LET/L1,1
-LET/L2,0
-ESE/X2:26:L1,L2

dove:

26 = JMP	command
1 = Numero sequenza chiamata	param1
0 = Non significativo comando 26	param2

2) Funzione leggi registri ETEL (vedi tabella registri Etel)

-ERR/NomeAsse:Variabile su cui caricato il valore letto:L#,L#+2:

Parametri: **a:b: c,d**
dove:

a = Numero o nome asse
b = Variabile in cui viene caricato il valore del registro esterno
c = Variabile di partenza dove sono dichiararti i parametri del registro esterno
d = Variabile finale dove sono dichiararti i parametri del registro esterno

typ = Tipo registro L#
idx = Numero registro L#+1
sidx = Numero sotto registro L#+2

esegue la funzione ETEL :

get_register(int axis, int typ, int idx, int sidx, long *value)

3) Funzione scrivi registri ETEL (vedi tabella registri Etel)

-EWR/NomeAsse:L#,L#+2:valore

Parametri: **a:b,c:d**

dove:

a = Numero o nome asse
b = Variabile di partenza dove sono dichiararti i parametri del registro esterno
c = Variabile finale dove sono dichiararti i parametri del registro esterno
d = Valore con cui viene caricato il valore del registro esterno

typ = Tipo registro L#
idx = Numero registro L#+1
sidx = Numero sotto registro L#+2

esegue la funzione ETEL :

set_register(int axis, int typ, int idx, int sidx, long value)

I tipi di registri ETEL sono:

TYP_NONE	0	/* no type */
TYP_IMMEDIATE	0	/* disabled or immediate value */
TYP_USER	1	/* user registers */
TYP_PPK	2	/* drive parameters */ K
TYP_MONITOR	3	/* monitoring registers */ M
TYP_SEQUENCE	5	/* sequence buffer */
TYP_TRACE	6	/* trace buffer */
TYP_ADDRESS	7	/* address value */
TYP_LKT	8	/* movement lookup tables */
TYP_TRIGGER	9	/* triggers buffer */
TYP_REALTIME	10	/* realtime buffer */
TYP_HALL_LKT	11	/* hall lookup tables (deprecated)*/
TYP_Y	11	/* Y type variables */

4) Funzione leggi status ETEL

-EWS/NomeAsse:3:L#,L#

Parametri: **a:b:c,d**

dove:

a = Numero o nome asse
b = NON significativo
c = Variabile di partenza dove viene caricato lo stato
d = Valore finale dove viene caricato lo stato

Bit0-31	Valore	Significato
0	0x00000001	= User bit 0, could be modified by trigger fuctions or by K177
1	0x00000002	= User bit 1, could be modified by trigger fuctions or by K177
2	0x00000004	= User bit 2, could be modified by trigger fuctions or by K177
3	0x00000008	= User bit 3, could be modified by trigger fuctions or by K177
4	0x00000010	= User bit 4, could be modified by trigger fuctions or by K177
5	0x00000020	= User bit 5, could be modified by trigger fuctions or by K177
6	0x00000040	= User bit 6, could be modified by trigger fuctions or by K177
7	0x00000080	= User bit 7, could be modified by trigger fuctions or by K177
8	0x00000100	= User bit 8, could be modified by trigger fuctions or by K177
9	0x00000200	= User bit 9, could be modified by trigger fuctions or by K177
10	0x00000400	= User bit 10, could be modified by trigger fuctions or by K177
11	0x00000800	= User bit 11, could be modified by trigger fuctions or by K177
12	0x00001000	= User bit 12, could be modified by trigger fuctions or by K177
13	0x00002000	= User bit 13, could be modified by trigger fuctions or by K177
14	0x00004000	= User bit 14, could be modified by trigger fuctions or by K177
15	0x00008000	= User bit 15, could be modified by trigger fuctions or by K177
16	0x00010000	= The drive is in power on
17	0x00020000	=
18	0x00040000	=
19	0x00080000	= Motor axis present .Bit normaly set to 1
20	0x00100000	= Motor is executing a trajectory(moving)
21	0x00200000	= Motor in the position windows
22	0x00400000	=
23	0x00800000	= Drive global warning
24	0x01000000	= **Drive executing an internal sequence**
25	0x02000000	=
26	0x04000000	= Drive in error mode

27 0x08000000 = Trace busy flag is set during a register trace acquisition
28 0x10000000 =
29 0x20000000 =
30 0x40000000 =
31 0x80000000 =

Esempio:

```
-EWS/Y:0:L9,L9
-DIS/3:Stato asse Y,L9,8,-1
-JEQ/and(L9,0x01000000) , 0x01000000,DriveExeInternalSequence
```

Esempi istruzioni ETEL DSteb3 :

```
;
; Esempio lettura Regitro M7 position actual
;
        -LET/L1,3     ; 3=M 2=K 1=X
        -LET/L2,7
        -LET/L3,0
        -ERR/Y:L10:L1,L3
        -DIS/-1:M7(Y),L10
        -DIS/1:Valore M7 asse Y,L10
;
; Esempio scrittura Registro User X3
;
        -LET/L1,1     ;  3=M 2=K 1=X
        -LET/L2,3
        -LET/L3,0
        -EWR/Y:L1,L3:12345
;
; Esempio lettura Registro User X3
;
        -LET/L1,1     ; 3=M 2=K 1=X
        -LET/L2,3
        -LET/L3,0
        -ERR/Y:L10:L1,L3
        -DIS/2:Valore X3 asse Y,L10


;
; Esempio comando power off 124 param 1  0=off
;
        -LET/L1,0     ; param 1  0=off
        -LET/L2,0     ; param 2
                                ; 124 power
        -ESE/Y:124:L1,L2

        -TMM/1000
```

```
;
; Esempio comando power on 124 param 1  1=on
;
        -LET/L1,1      ; param 1  1=on
        -LET/L2,0      ; param 2
                                    ; 124 power
        -ESE/Y:124:L1,L2

;
; Nel power on bisogna dare anche il comando WTM
;
        -LET/L1,0      ; param 1
        -LET/L2,0      ; param 2
                                    ; 8 WTM
        -ESE/Y:8:L1,L2

;
        -TMM/1000

;
; Esempio lettura stato
;
        -EWS/Y:0:L9,L9
        -DIS/3:Stato asse Y,L9,8,-1
        -JEQ/and(L9,0x01000000) , 0x01000000,DriveExeInternalSequence
```

11.40 ESE Exec Sequenze (Sistema ETEL)

ESE **Exec Sequence**

Funzione: Esegue delle sequenze esterne (Exec Sequence)

Parametri: **a:b:[c,d]**

dove:

a = Numero o nome asse
b = Numero sequenza
c = Variabile di partenza dove sono dichiararti i parametri della sequenza
d = Variabile finale dove sono dichiararti i parametri della sequenza

external sequence -ESE/ Asse:
Numero sequenza:
Da,A(Da+5)*n
Da 1=Tipo registro
2=Numero registro
3=Numero sotto registro
4=Valore
5=Costante conversione

Esempio:

```
- ESE/X1:100:L101,L110
```

11.41 ERR External Read Registry (Sistema ETEL)

ERR External Read Registry

Funzione: Leggi registri esterni (External Read Registry)

Parametri: **a:b:c,d**

dove:

a = Numero o nome asse
b = Variabile in cui viene caricato il valore del registro esterno
c = Variabile di partenza dove sono dichiararti i parametri del registro esterno
d = Variabile finale dove sono dichiararti i parametri del registro esterno

external read register -ERR/ Asse:
Locale o Globale:
Da,A(Da+5)
Da 1=Tipo registro
2=Numero registro
3=Numero sotto registro
4=Valore
5=Costante conversione

Esempio:

```
ERR/X1:G11:L101,L105
```

11.42 EWR External Write Registry (Sistema ETEL)

EWR External Write Registry

Funzione: Scrivi registri esterni(External Write Registry)

Parametri: **a:b,c:d**

dove:

a = Numero o nome asse
b = Variabile di partenza dove sono dichiarati i parametri del registro esterno
c = Variabile finale dove sono dichiarati i parametri del registro esterno
d = Valore con cui viene caricato il valore del registro esterno

external write register -EWR/ Asse:
Da,A(Da+5):
Valore o Locale o Globale
Da 1=Tipo registro
2=Numero registro
3=Numero sotto registro
4=Valore
5=Costante conversione

Esempio:

```
- EWR/X1:L101,L105:123
```

11.43 ECM External CoMmand (Sistema ETEL)

ECM External CoMmand

Funzione: Esegue un comando esterno (External CoMmand)

Parametri: **a:b:c,d:e:f,g**

dove:

a = Numero o nome asse
b = Numero comand
c = False o True Fast command
d = False o True report drive errors
e = Type parameters
f = Variabile di partenza dove sono dichiararti i parametri del comando
g = Valore finale dove sono dichiararti i parametri del comando

external command -ECM/Asse:
Numero comand
False o True Fast command,
False o True report drive errors:
Type parameters
Type parameter 0 = no param
1 = d 1 long
2 = i 1 double
3 = dd 2 long
4 = id 1 double 2 long
5 = di 1 long 2 double
6 = ii 2 double
7 = Number parameter free
Da,A(Da+3)*n
Da 1=Type
2=Valore
3=Costante conversione

Esempio:

```
- ECM/X1:109:0,0:3:L11,L19
```

11.44 EWS External Wait Signal (Sistema ETEL)

EWS External Wait Signal

Funzione: Attende una segnalazione esterna (External Wait Signal)

Parametri: **a:b:c,d**

dove:

a = Numero o nome asse
b = FlagEqualOrNotEqua
c = Variabile di partenza dove sono dichiarati i parametri della segnalazione
d = Valore finale dove sono dichiarati i parametri della segnalazione

external sequence -EWS/ Asse:
FlagEqualOrNotEqual:
Da,A
Valore 1 = mask.drive.present
2 = mask.drive.moving
3 = mask.drive.in_window
4 = mask.drive.sequence
5 = mask.drive.error
6 = mask.drive.trace
7 = mask.drive.warning
8 = mask.drive.breakpoint
9 = mask.drive.user

Esempio:

```
-EWS/X1:1:L101,L110
```

11.45 CLM Comando da Logica di Macchina (Sistema ETEL)

CLM **Comando da logica di macchina**

Funzione: Comando da Logica di Macchina (Sistema ETEL)

Parametri: **a:b:c,d**

dove:
a = Numero o nome asse
b = Valore con cui viene caricato il valore del registro esterno
c = Variabile di partenza dove sono dichiararti i parametri del registro esterno
d = Variabile finale dove sono dichiararti i parametri del registro esterno

Comando da logica di machina -CLM/ Asse:
Numero comando:
Da,A(Da+5)
Da 1=Tipo registro
2=Numero registro
3=Numero sotto registro
4=Valore
5=Costante conversione

Nota: Il segno '-' davanti all'asse, indica che il comando è rivolto ai registri della DSMAX e non ai DRIVE.

I comandi sono:

1 Andata in foratura -CLM/ZA:1:Pos.finale foratura,
Pos.inizio foratura,
Velocita Lavoro

2 Ritorno veloce -CLM/ZA:2:Pos.finale ritorno,
Pos.Sgancio X e Y,
Pos.FuoriPezzo,
Vel.Salita

7 Imposta la distanza movimento completato
–CLM/7:ZA,Distanza

8 ZOVR aumento in percentuale Z e S
-CLM/ZA:8,
Incremento in percentuale

11.46 SND Effettua l'emissione di un file WAV sull' uscita audio del PC

SND SouND Play file WAV su linea audio

Funzione: Emette un file WAV sull' uscita audio del PC

Parametri: **a,b**

dove:
a = nome file WAV da utilizzare come suono
b = Locale o Globale di stato dove viene caricato l'esito dell'istruzione
0 = OK
-1 = Porta audio per l'emissione di un file già impegnata

Nota: se non specificato nella dichiarazione del file, viene ricercato nella directory ..\DAT

Esempio:

```
-SND/Glass.wav,L1
-TMM/100
-SND/Glass.wav,L1
-TMM/100

-DIS/1:Stato,L1,3,0
```

CAPITOLO 12

12 Istruzioni per l'integrazione con gli altri ambienti

.1-ARI Richiesta di esecuzione di una istruzione per l'ambiente specificato nel primo parametro e attesa della risposta (ritorno effettuato con la funzione di "WriteServiceParametersAndContinue" al sottosistema di automazione AXESBRAIN)
(Ambient Request Instruction)

.2-SEC Set event client

12.1 ARI: Ambient Request Instruction

ARI **Ambient Request Instruction**

Funzione: richiesta di esecuzione di una istruzione per l'ambiente specificato nel primo parametro e attesa della risposta (ritorno effettuato con la funzione di "WriteServiceParametersAndContinue" al sottosistema di automazione AXESBRAIN)

Parametri: **a:b,c,d,e1…en:f1..fn:[g,h]**

dove:

a = ambiente interessato
- 1 Ambiente "CLIENT"
- 2 Ambiente o sotto sistema CNC GMSISO
- 3 Ambiente o sotto sistema PLC GMSAWL
- 4 Ambiente di analisi delle immagine GVS
- 101 Ambiente o sotto sistema 1 OEM
- 102 Ambiente o sotto sistema 2 OEM
- 10n Ambiente o sotto sistema n OEM

b = chiave di risposta
c = messaggio
d = funzione
da **e1 a en** = variabili Locali o Globali da trasferire nella richiesta
da **f1 a fn** = variabile Locali o Globali da ricevere nella risposta alla richiesta
g = Timeout in caso di non risposta 0=infinito (Default)
h = Label di salto per Timeout scaduto

Esempio:

```
-ARI/1:23,Messaggio,Funzione,G12,G45:G100,G200
-ARI/1,12,VisAlgo,GetValueBlob,L11,L11,L31,L51:1000,NonRisponde
```

Richiede un servizio dal applicativo utente in termini di dati, che quando reperiti verranno forniti tramite l'opportuna funzione di scrittura od ad altri sottosistemi.
L'applicazione dell'utente predisposto ad eseguire la richiesta,una volta che ha terminato il compito deve segnalare tramite la funzione
"WriteServiceParametersAndContinue" che l'operazione è stata eseguita.

Note:

Invia alle applicazioni connesse il seguente evento :

```
HRESULT OnRequestServiceEvent(      BSTR    sMsg,
                                    int     lSubSystem ,
                                    int     lKey,
                                    int     iNumberParameter.
                                    VARIANT * pVar
                              )
```

Parametri

BSTR	sMsg	Messaggio dipendente dal servizio richiesto
int	lSubSystem	Sottosistema che ha prodotto il messaggio 0 Sistema centrale **1 AXESBRAIN** 2 GMSISO 3 GMSAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
int	lKey	Chiave per didtinguere le richieste 101 Richiesta di un servizio generico 102 Richiesta di un input da tastiera 103 Richiesta di una funzione GVS 104 Richiesta di una procedura GVS
int	iNumberParameter	Numero di parametri
VARIANT *	pVar	Parametri della richiesta

Il processo del sotto sistema si pone in attesa inviando una richiesta si servizio, sarà appunto il l'esecutore del servizio a chiamare il metodo: "WriteServiceParametersAndContinue"

STDMETHODIMP WriteServiceParametersAndContinue(

int	**iNumberSubSystem,**
BSTR	**sMsg,**
int	**lParam1,**
int	**lKey,**
int	**iNumberParameter,**
VARIANT*	**pVarParameter,**
int *	**piError**
)	

Valore di ritorno

S_OK	ritorno ok
S_FALSE	Errore nei parametri

Parametri

int iNumberSubSystem	Tipo di identificazione del sottosistema **1 AXESBRAIN** 2 GMSISO 3 GMSAWL 101 Primo sottosistema OEMCapability 102 Secondo sottosistema OEMCapability ... ... 109 Nono sottosistema OEMCapability
BSTR sMsg	Messaggio
int lParam1	Parametro
int lKey	Chiave di riconoscimento
int iNumberParameter	Numero di parametri
VARIANT* pVarParameter	Variant con il valore di ogni parametro
int * piError	Ritorna il tipo di errore nel caso che stdmethodimp sia S_FALSE ed in particolare: 1 iNumberSubSystem errato 2 sMsg errato 3 lParam1 errato 4 lKey errato 5 iNumberParameter errato

12.2 SEC: Set Event Client

SEC **Set Event Client**

Funzione: Set event client

Parametri: **[a],b,c**

dove:

a = iCookie (default –1)
b = Stringa di comunicazione
c = valore di comunicazione

Esempio:

```
// set evento client -SEC/iCookie:MACHINE_LOGIC,1  ; Info
// set evento client -SEC/iCookie:MACHINE_LOGIC,2  ; Warning
// set evento client -SEC/iCookie:MACHINE_LOGIC,3  ; Allarm
// set evento client -SEC/iCookie:MACHINE_LOGIC,4  ; Dinamic Area
// set evento client -SEC/iCookie: prova di comunicazione,g1

PartProgram[InOne]
        -DIS/11:AlarmInOne Input  =,L1,3,0
        -DIS/12:AlarmInOne Status =,L2,1,0
        -RET

PartProgram[InZero]
        -DIS/11:AlarmInZero Input  =,L1,3,0
        -DIS/12:AlarmInZero Status =,L2,1,0
        -RET


PartProgram[OutOne]
        -DIS/8:AlarmOutOne Output =,L1,3,0
        -DIS/9:AlarmOutOne Status =,L2,1,0
        -SEC/MACHINE_LOGIC,3
        -RET

PartProgram[OutZero]
        -DIS/8:AlarmOutZero Output =,L1,3,0
        -DIS/9:AlarmOutZero Status =,L2,1,0
        -SEC/MACHINE_LOGIC,3
        -RET
```

CAPITOLO 13

13 Istruzioni di comunicazione

.1-CSO	Connessione ad un socket TCP/IP (Connect SOcket)
.2-LSO	Listen ad un socket TCP/IP (Listen SOcket)
.3-RSO	Leggi i dati che transitano su un socket TCP/IP (Read SOcket)
.4-TSO	Scrive su un socket TCP/IP (Write SOcket)
.5-DSO	Cancella una connessione a un socket TCP/IP (Destroy SOcket)
.6-FSO	Azzera i dati eventualmente ricevuti su un socket TCP/IP (Free SOcket)
.7-GSO	Acquisisce le informazioni di un socket TCP/IP (Get information SOcket)
.8-OSL	Apre una linea seriale (Open Serial Line)
.9-RXL	Riceve dati da una linea seriale(Receive Serial Line)
.10-TXL	Trasmette dati su una linea seriale(Trasmit Serial Line)
.11-CSL	Chiude la linea seriale(Close Serial Line)
.12-FSL	Azzera i dati eventualmente ricevuti da una linea seriale(Free Serial Line)
.13-RFB	Leggi dati da FieldBus
.14-WFB	Scrivi dati su FieldBus
.15-RGS	Reset linea GSM/GPRS
.16-SMS	Send SMS su GSM/GPRS
.17-WMS	Wait message SMS da GSM/GPRS
.18-CGS	Effettua una chiamata su GSM/GPRS
.19-WRG	Attende una chiamata da GSM/GPRS
.20-CTL	Effettua chiamata su linea telefonica
.21-WTL	Attende una chiamata da linea telefonica
.22-STL	Chiude la linea telefonica
.23-GTL	Acquisisce un numero da linea telefonica
.24-PTL	Invia un file registrato su linea telefonica
.25-EML	Invia un E-Mail

13.1 CSO: Connect SOcket

CSO **Connect SOcket**

Funzione: Connessione ad un socket TCP/IP (Connect SOcket)

Parametri: **a,b:c:d,e,f,g:[h]**

dove:

a	=	Numero socket	1-10	
b	=	Numero Porta	1025->	
c	=	Loc o Globale ritorno Stato:	0=Ok 1=NON connesso	
d	=	IP connectA	000	Esempio 127
e	=	IP connectB	000	Esempio 17
f	=	IP connectC	000	Esempio 2
g	=	IP connectD	000	Esempio 7
h	=	Caratteri Buffer Ricezione	Default1024	

Esempio:

- CSO/2,1027:L1:192,103,5,7

Note:

a) La lunghezza del buffer circolare che per default è 1024, viene utilizzata per acquisire i caratteri che arrivano una volta che è stat aperta la connessione, con la istruzione RSO vengono acquisiti dal part program per laloro elaborazione. Con l'istruzione FSO vengono cancellati tutti i byte ricevuti.

13.2 LSO: ListenSOcket

LSO **Listen SOcket**

Funzione: Listen ad un socket TCP/IP (Listen SOcket)

Parametri: **a,b:[c,e]**

dove:

a	= Numero socket	1-10	
b	= Numero Porta	1025->	
c	= Loc o Globale ritorno Stato:	0=Ok 1=Busy	
e	= Caratteri Buffer Ricezione	1024	Default 1024

Esempio:

```
- LSO/1,1027:L1          ; Listen (si mette in ascolto sulla porta 1027)
```

13.3 RSO: ReadSOcket

RSO **Read SOcket**

Funzione: Leggi i dati che transitano su un socket TCP/IP (Read SOcket)

Parametri: **a:b,c:[d,e,f]**

dove:

a = Numero socket
b = Indirizzo Inizio variabile
c = Indirizzo Fine variabile
d = Indirizzo variabile di Stato 0=OK 1=Linea non aperta 2=timeout scaduto
e = Timeout (sec):0=infinito Default 0
f = Indirizzo variabile numero caratteri ricevuti

Esempio:

- RSO/1:L11,L19:L2:L3

L11 : primo carattere da ricevere
L19 : ultimo carattere da ricevere
L2 : stato 0=OK 1=Linea non aperta 2=timeout scaduto
L3 : numero caratteri ricevuti

13.4 TSO: WriteSOcket

TSO **Write SOcket**

Funzione: Scrive su un socket TCP/IP (Write SOcket)

Parametri: **a:b,c:[e,f]**

dove:

a = Numero socket
b = Indirizzo Inizio variabile
c = Indirizzo Fine variabile
e = Indirizzo variabile di Stato 0=OK 1=Linea non aperta
f = Indirizzo variabile numero caratteri trasmessi L22

Esempio:

```
-TSO/2:L11,L19:L2:L3
```

L11 : primo carattere da trasmettere
L19 : ultimo carattere da trasmettere
L2 : stato socket 0=OK 1=Linea non aperta
L3 : Indirizzo variabile numero caratteri trasmessi L22

13.5 DSO: DestroySOcket

DSO **Destroy SOcket**

Funzione: Cancella una connessione a un socket TCP/IP (Destroy SOcket)

Parametri: **a**

dove:

a = Numero socket

Esempio:

- DSO/1

13.6 FSO: FreeSOcket

FSO **Free SOcket**

Funzione: Azzera i dati eventualmente ricevuti su un socket TCP/IP (Free SOcket)

Parametri: **a**

dove:

a = Numero socket

Esempio:

- FSO/1

13.7 GSO: Get information SOcket

GSO **Get information SOcket**

Funzione: Acquisisce le informazioni di un socket TCP/IP (Get information SOcket)

Parametri: **a,b**

dove:

a = Numero socket
b = Variabile di stato dove : 0 =Nulla 1=WaitForListen 2=ListenWithConnect 3=Connect

Esempio:

```
        -GSO/1,L1
```

Esempio sulla trasmissione su socket TCP/IP:

```
; Connect
        -CSO/2,1027:L1:192,103,5,7
        -LET/L1,1
        -LET/L11,0x41,L12,0x42,L13,0x43
aa      -TSO/2:L11,L19:L2:L3
        -RSO/2:L11,L19:L2:L3
        -DIS/1:locale,L11
        -TMM/1000
        -LET/L1,L1+1
        -LET/L11,L11+1,L12,L12+1,L13,L13+1
        -JLE/L1,10,aa
        -DSO/2:

;   Listen
        -LSO/1,1027:L1
aa      -
        -RSO/1:L11,L19:L2:L3
        -TSO/1:L11,L19:L2:L3
        -JLE/L11,0x41+10,aa
        -DSO/1
```

Esempio doppio collegamento socket

```
; Programma di test della trasmissione su rete (Client)

        -LET/L11,0                      ; Inizializzazione valore
        -DIS/1:Connessione in corso ...  ; Stampa "Connessione in corso ..."
        -DIS/2:                         ; Pulisce la riga
        -DIS/3:                         ; Pulisce la riga
        -CSO/1,1027:L1:192,103,5,1      ; Connect (si connette all' indirizzo 192.103.5.1 e sulla porta 1027)
        -DIS/1:Connesso          ; Stampa "Connesso"
LOOP    -
        -TSO/1:L11,L19:L2:L3            ; Trasmette il valore
        -DIS/2:Trasmesso ,L11           ; Stampa il valore trasmesso
        -ADD/L11,1                      ; Incrementa il valore
        -TMM/300                        ; Pausa di 300 ms
        -JLT/L11,256,LOOP               ; Se il valore è maggiore di 255 termina la comuncazione     -
DSO/1                                   ; Chiude la comunicazione
        -DIS/1:                         ; Pulisce la riga
        -DIS/3:Connessione terminata !  ; Stampa "Connessione terminata !"


; Programma di test della trasmissione su rete (Server)

        -DIS/1:Attesa connessione ...    ; Stampa "Attesa connessione ..."
        -DIS/2:                         ; Pulisce la riga
        -DIS/3:                         ; Pulisce la riga
        -LSO/1,1027:L1                  ; Listen (si mette in ascolto sulla porta 1027)
LOOP    -
        -RSO/1:L11,L19:L2:L3            ; Si mette in attesa di ricevere qualcosa
        -DIS/1:Connesso          ; Stampa "Connesso"
        -DIS/2:Ricevuto ,L11            ; Stampa il valore rivevuto
        -JLE/L11,254,LOOP               ; Se il valore è maggiore di 254 termina la comuncazione
        -DSO/1                          ; Chiude la comunicazione

        -DIS/1:                         ; Pulisce la riga
        -DIS/3:Connessione terminata !  ; Stampa "Connessione terminata !"
```

13.8 OSL: Open Serial Line

OSL **Open Serial Line**

Funzione: Apre una linea seriale (Open Serial Line)

Parametri: **a:b:[c:d,e,f,g,h,i]**

dove:

a	= Numero seriale	1-n	
b	= Tipo Protocolllo:	0=free,1=C.U.	
c	= Loc o Globale ritorno Stato	0= Ok 1=Busy	
d	= BaudRate	1200-115000	Default 9600
e	= Parità	0=No parity 1=Dispari 2= Pari	Default 0
f	= NumeroBit	7-8	Default 8
g	= BitStop	1-2	Default 1
h	= Caratteri Buffer Ricezione	1024	Default 1024
i	= Flag RS485	0=no 1=si	Default 0

Esempio:

```
-OSL/1:0:L1:9600,0,8,1
```

Note:

b) La lunghezza del buffer circolare che per default è 1024, viene utilizzata per acquisire i caratteri che arrivano una volta che è stat aperta la linea seriale, con la istruzione RXL vengono acquisiti dal part program per laloro elaborazione. Con l'istruzione FSL vengono cancellati tutti i byte ricevuti.

c) Le seriali sono quelle gestite dal sistema operativo della macchina.

d) Il flag RS485 se 1 attiva la gestione secondo la modalità RS485

13.9 RXL: Receive Serial Line

RXL **Receive Serial Line**

Funzione: Riceve dati da una linea seriale(Receive Serial Line)

Parametri: **a:b,c:[d,e,f]**

dove:
a = Numero seriale
b = Indirizzo Inizio variabile
c = Indirizzo Fine variabile
d = Indirizzo variabile di Stato 0=OK 1=Linea non aperta 2=timeout scaduto
e = Timeout(sec) 0=infinito -1= Ritorna in **f** il numero di caratteri nel buffer di ricezione senza caricare nulla in **b** e **c**
f = Indirizzo variabile numero caratteri ricevuti

Nota se **e** Timeout(sec) = -1 non vengono caricati caratteri, ma in **f** viene messo il numero di caratteri ricevuti nel buffer interno di ricezione

Esempio:

```
- RXL/1:L1,L1:L21:0:L3
```

L1 : primo carattere da ricevere
L20 : ultimo carattere da ricevere
L21 : 0=OK 1=Linea non aperta 2=timeout scaduto
L3 : numero caratteri ricevuti

13.10 TXL: Trasmit Serial Line

TXL **Trasmit Serial Line**

Funzione: Trasmette dati su una linea seriale(Trasmit Serial Line)

Parametri: **a:b,c:[d:e]**

dove:
a = Numero seriale
b = Indirizzo Inizio variabile
c = Indirizzo Fine variabile
d = Indirizzo variabile di Stato 0=OK 1=Linea non aperta
e = Indirizzo variabile numero caratteri trasmessi

Esempio:

- TXL/1:L1 ,L20:L2:L3

L1 : primo carattere da trasmettere
L20 : ultimo carattere da trasmettere
L2 : 0=OK 1=Linea non aperta
L3 : Indirizzo variabile numero caratteri trasmessi

13.11 CSL: Trasmit Serial Line

CSL **Close Serial Line**

Funzione: Chiude la linea seriale(Close Serial Line)

Parametri: **a**

dove:
a = Numero seriale

Esempio:

- CSL/1

13.12 FSL: FreeSerial Line

FSL **FreeSerial Line**

Funzione: Azzera i dati eventualmente ricevuti da una linea seriale(Free Serial Line)

Parametri: **a**

dove:
a = Numero seriale

Esempio:

- FSL/1

Esempio gestione della linea seriale 1 su un dispositivo cambio utensili:

```
                -CAL/AzzTamburo:L1
                -DIS/1:Azzeramento Eseguito=,L1
                -CAL/ToolTamburo:L1,449
                -DIS/2:Comando Eseguito,L1


PartProgram[AzzTamburo]
                -LET/L11,8,L12,0x44,L13,3
                -LET/L14,3,L15,0,L16,0,L17,0,L18,0

                -LET/L19,L11+L12+L13+L14+L15+L16+L17+L18
                -LET/L19,and(-L19,0xFF)

                -OSL/1:0:L1:9600,0,8,1
                -FSL/1
                -TXL/1:L11,L19:L2:L3
                -RXL/1:L1,L1:L2:0:L3
                -JEQ/L1,6,O_OK
                -CSL/1
                -LET/L1,-1
                -RET
O_OK            -RXL/1:L1,l5
                -CSL/1
                -LET/L1,L4
                -RET

PartProgram[ToolTamburo]
; L2 Numero tool
                -LET/L11,8,L12,0x44,L13,3
                -LET/L14,1,L15,0,L16,0
                -LET/L17,and(L2/256,0xFF),L18,and(L2,0xFF)

                -LET/L19,L11+L12+L13+L14+L15+L16+L17+L18
                -LET/L19,and(-L19,0xFF)

                -OSL/1:0:L1:9600,0,8,1
                -FSL/1
                -TXL/1:L11,L19:L2:L3
                -RXL/1:L1,L1:L2:0:L3
                -JEQ/L1,6,O_OK
                -CSL/1
                -LET/L1,-1
                -RET
O_OK            -RXL/1:L1,l5
                -CSL/1
                -LET/L1,L4
                -RET
```

13.13 RFB: Leggi dati da FieldBus

RFB **Read FieldBus**

Funzione: Legge dati da FieldBus

Parametri: **a:b,c,d,e,f1,g1[,f2,g2….fn,gn]**

dove:
a = Tipo di fieldbus
0 Canopen default
b = Indirizzo modulo identificaivo sulla rete
c = Indirizzo dell'oggetto da leggere
e = Sottoindice dell'oggetto da leggere
f1..n = Tipo di lettura dei dati dell'oggetto
1= Byte
2=Word (2 bytes)
3=Dword (4 bytes)
4=Float (4 bytes)
5=Double (8 bytes)
g1..n = Locale o Globale dove vene letto il valore

i parametri **f g** sono ripetibili fino a completare la lettura dei valori contenuti nell'oggetto richiesto

Esempio:

```
-RFB/0:10,0x6700,2,2,L1
```

Legge l'oggetto index=0x6700 sub index=2 dal modulo indirizzo= 10 e mette la WORD letta (2 bytes) nella locale L1

13.14 WFB: Scrivi dati su FieldBus

WFB **Write FieldBus**

Funzione: Scrive dati su modulo FieldBus

Parametri: **a:b,c,d,e,f1,g1[,f2,g2….fn,gn]**

dove:
a = Tipo di fieldbus
0 Canopen default
b = Indirizzo modulo identificaivo sulla rete
c = Indirizzo dell'oggetto da scrivere
e = Sottoindice dell'oggetto da scrivere
f1..n = Tipo di scrittura del dato nell'oggetto
1= Byte
2=Word (2 bytes)
3=Dword (4 bytes)
4=Float (4 bytes)
5=Double (8 bytes)
g1..n = Valore da scrivere nel campo 1..n

i parametri **f g** sono ripetibili fino a completare la scrittura dei valori nei campi nell'oggetto

Esempio:

-WFB/0:10,0x8100,2,2,213

Scrive nel campo unico dell'oggetto index=0x8100 sub index=2 del modulo indirizzo= 10 il valore word 213

13.15 RGS Reset linea GSM/GPRS

RGS Reset GSm/gprs

Funzione: Reset la linea modem GSM/GPRS

Parametri: nessuno

Esempio:

-RGS

I parametri della linea seriale abbinata al modem si trovano nel File :

COMUNICATION.TXT

Il file viene ricercato nella directory \DAT\ COMUNICATION.TXT

I cui contenuti del file sono:

[ParametriGenerali]
NumeroSeriale=5
BaudRate=9600
Parity=0
NumeroBit=8
BitStop=1
MassimoNumeriTelefonici=1
MassimoSMS=24
ServiceCenterNumber=+393492000200

[Rubrica]
NumeroTelefono_1=3474493103

[SMS]
Messaggio_1=Test CEAN Magazzino
Messaggio_2=Test CEAN Stazione Magazzino

13.16 SMS Send SMS su GSM/GPRS

SMS Send SMS su GSM/GPRS

Funzione: Scrive dati su modulo FieldBus

Parametri: **a[,b:c,d,e:f]**

dove:

a = Numero messaggio da inviare il cui numero e specificato nel file GSM_GPRS.TXT sezione [SMS] voce Messaggio_**a**
se non viene specificato **b,** nella sezione [Rubrica] voce NumeroTelefono_ **a** si può indicare il numero del cellulare da chiamare

b = Numero del cellulare da chiamare

c = Valore da accodare altra stringa definita da **a**

d = Numero di cifre intere da visualizzare (0:10)

e = Numero di cifre decimali da visualizzare (0:5)

f = Globale o Locale su cui viene restituito lo stato dell'operazione di SMS 0=OK

Esempio;

```
-GDT/L11,L18
-DIS/5:Anno         =,L11
-DIS/6:Mese         =,L12
-DIS/7:Giorno       =,L13
-DIS/8:Ora          =,L14
-DIS/9:Minuti       =,L15
-DIS/10:Secondi     =,L16
-DIS/11:MilliSecondi =,L17

-LET/L20,L14
-LET/L20,L20+(L15/100)
-LET/L21,393476820942
-SMS/2,L21:L20,5,2:G100
```

13.17 WMS Wait message SMS da GSM/GPRS

WMS Wait SMS da GSM/GPRS

Funzione: Legge i messaggi SMS presenti nel cellulare

Parametri: **a,b,c,:d,e:,f,g,h**

dove:
a = Tipo di modalità di lettura dei messaggi :
1 dammi il primo
99 dammi l'ultimo
-1 dammi il primo e cancellalo
-99 dammi l'ultimo e cancellalo
-999 dammi l'ultimo e cancella tutti i messaggi

b = numero messaggi presenti , se negativo errore di comunicazione

c = Viene restituito il numero chiamante
d,e(d+5) = Giorno Mese Anno Ora minuti secondi
f = Viene restituito il Numero campi numerici caricati
g,h = Campi numeri presenti nel messaggio
Gxx,Gzz Gxx 1 primo campo numerico
Gzz ultimo campo numerico

Esempio:

```
-WMS/-1,L1,L2:L11,L16:L3,L21,L25
;  modo
;          1     dammi il primo
;          99    dammi l'ultimo
;          -1    dammi il primo e cancellalo
;          -99   dammi l'ultimo e cancellalo
;          -999 dammi l'ultimo e cancella tutti i messaggi
;      L1,     numero messaggi presenti
;                          se negativo  errore di comunicazione
;      L2:     numero chiamante
;      G1,G6:  Giorno Mese Anno Ora minuti secondi
;      L3,       Numero campi numerici
```

```
;          Gxx,Gzz   Gxx 1 primo campo numerico
;                    Gzz ultimo campo numerico


-DIS/0:Numero messaggi      =,L1,5,0
-DIS/1:Numero Chiamante =,L2,15,0
-DIS/2:Numero campi          =,L3,5,0
-DIS/3:Primo campo      =,L21
-DIS/4:Secondo campo    =,L22
-DIS/5:Terzo campo      =,L23
-DIS/6:Quarto campo     =,L24
-DIS/7:Quinto campo     =,L25

-DIS/8:Anno             =,L11,5,0
-DIS/9:Mese             =,L12,5,0
-DIS/10:Giorno           =,L13,5,0
-DIS/11:Ora             =,L14,5,0
-DIS/12:Minuti           =,L15,5,0
-DIS/13:Secondi        =,L16,5,0
```

13.18 CGS Effettua una chiamata su GSM/GPRS

CGS Effettua una chiamata su GSM/GPRS

Funzione: Si collega al telefono indicato nel numero di chiamata

Parametri: **a,b**

dove:
a = nella sezione [Rubrica] voce NumeroTelefono_ **a** si può indicare il numero del ctelefono da chiamare
b = Globale o Locale in cui viene restituito lo stato dell'operazione 0= OK

Esempio:

```
-CGS/2,L1  ;     numero di telefono definito GSM_GPRS.TXT
                 nella sezione [Rubrica]  voce NumeroTelefono_ 2=
```

13.19 WRG Attende una chiamata da GSM/GPRS

WRG **Wait RinG on GSM/GPRS**

Funzione: Attende una chiamata da GSM/GPRS

Parametri: **a,b,c**

dove:
a = Tipo di modo
0 Attendi la telefonata
-1 Azzera eventuali pendenze su GSM/GPRS
b = Globale o Locale su cui viene restituito lo stato dell'operazione 0=OK
c = Globale o Locale su cui viene restituito il numero di telefono chiamante

Esempio:

```
-DIS/2:Attesa telefonata
-DIS/3:
-WRG/-1
-WRG/0,L1,L2
-DIS/2:Stato  = ,L1,3,0
-DIS/3:Numero =+,L2,12,0
-KYB/Telefonata arrivata
```

Esempio:

```
Test-
        -DIS/2:Attesa telefonata
        -DIS/3:

        -WRG/0,L1,L2
        -DIS/2:Stato Ric  = ,L1,3,0
        -DIS/3:Numero     =+,L2,12,0
        -TMM/10000
        -JNE/L1,0,Test

        -LET/G100,L2
        -LET/G101,0                 ; Sensore 1
        -LET/G102,0                 ; Sensore 2
        -LET/G103,0                 ; Sensore 3
        -LET/G104,0                 ; Sensore 4
        -LET/G105,0                 ; Sensore 5
        -LET/G106,0                 ; Sensore 6
        -LET/G107,0                 ; Sensore 7
        -LET/G108,0                 ; Sensore 8
        -LET/G111,0                 ; PT100 1
        -LET/G112,0                 ; PT100 2
        -LET/G113,0                 ; PT100 3
        -LET/G114,0                 ; PT100 4

        -GDT/L11,L17
/*
        -DIS/5:Anno         =,L11
        -DIS/6:Mese         =,L12
        -DIS/7:Giorno       =,L13
        -DIS/8:Ora          =,L14
        -DIS/9:Minuti       =,L15
        -DIS/10:Secondi     =,L16
        -DIS/11:MilliSecondi =,L17
*/
        -LET/L20,L14
        -LET/L20,L20+(L15/100)

        -SMS/1,L1:L20,5,2:L2
        -DIS/2:Stato Trsm = ,L1,3,0

        -JMP/Test
```

Esempio:

```
PartProgram[Temperatura1]
        -JGT/G99,0,Fine
        -JGT/G121,0,Fine
        -LET/G121,1
        -SMS/21,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Temperatura2]
        -JGT/G99,0,Fine
        -JGT/G122,0,Fine
        -LET/G122,1
        -SMS/22,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Temperatura3]
        -JGT/G99,0,Fine
        -JGT/G123,0,Fine
        -LET/G123,1
        -SMS/23,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Temperatura4]
        -JGT/G99,0,Fine
        -JGT/G124,0,Fine
        -LET/G124,1
        -SMS/24,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore1]
        -JGT/G99,0,Fine
        -JGT/G111,0,Fine
        -LET/G111,1
        -SMS/11,L1:G840,5,2:G100
Fine-
        -RET
```

```
PartProgram[Sensore2]
        -JGT/G99,0,Fine
        -JGT/G112,0,Fine
        -LET/G112,1
        -SMS/12,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore3]
        -JGT/G99,0,Fine
        -JGT/G113,0,Fine
        -LET/G113,1
        -SMS/13,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore4]
        -JGT/G99,0,Fine
        -JGT/G114,0,Fine
        -LET/G114,1
        -SMS/14,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore5]
        -JGT/G99,0,Fine
        -JGT/G115,0,Fine
        -LET/G115,1
        -SMS/15,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore6]
        -JGT/G99,0,Fine
        -JGT/G116,0,Fine
        -LET/G116,1
        -SMS/16,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore7]
        -JGT/G99,0,Fine
        -JGT/G117,0,Fine
```

```
        -LET/G117,1
        -SMS/17,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[Sensore8]
        -JGT/G99,0,Fine
        -JGT/G118,0,Fine
        -LET/G118,1
        -SMS/18,L1:G840,5,2:G100
Fine-
        -RET

PartProgram[UPS]
        ;-JGT/G99,0,Fine
        -JGT/G119,0,Fine
        -LET/G119,1
        -SMS/19,L1:G840,5,2:G100
Fine-
        -RET
```

13.20 CTL Effettua chiamata su linea telefonica

CTL Effettua una chiamata su linea telefonica

Funzione: Effettua una chiamata su linea telefonica

Parametri: **a,b,c**

dove:
a = Numero telefonico da chiamare
b = Globale o Locale su cui viene restituito lo stato dell'operazione
Stato 0 = OK connesso
1 = occupato
2 = non risponde dopo il tempo prefissato (default 5 sec)
-1 = nessun modem trovato
-2 = modem già occupato
c = Tempo massimo da numero fatto a risposta (default 5 secondi)

Esempio:

-CTL/125632826,L1

13.21 WTL Attende una chiamata da linea telefonica

WTL **Attende una chiamata da linea telefonica**

Funzione: Attende una chiamata da linea telefonica

Parametri:

dove:
a = Globale o Locale su cui viene restituito lo stato dell'operazione
Stato 0 = OK connesso
1 =
2 = Time out scaduto
-1 = nessun modem
-2 = modem già occupato

b = Globale o Locale su cui viene restituito il numero chiamante
c = Tempo massimo di attesa(default sempre)in millisecondi

Esempio:

-WTL

13.22 STL Chiude la linea telefonica

STL Chiude la linea telefonica

Funzione: Chiude la linea telefonica

Parametri: **a**

dove:
a = Globale o Locale su cui viene restituito lo stato dell'operazione
Stato 0 = OK sconnesso
1 =
-1 = nessun modem
-2 = modem già occupato
-3 = Linea non aperta

Esempio :

-STL

13.23 GTL Acquisisce un numero da linea telefonica

GTL Acquisisce un numero da linea telefonica

Funzione: Acquisisce un numero da linea telefonica digitato sulla tastiera del telefono

Parametri: **a:b,c,d**

dove:

a = Globale o Locale su cui viene restituito lo stato dell'operazione

Stato 0 = OK
1 =
2 = Time out scaduto
-1 = nessun modem
-2 = modem già occupato
-3 = Linea non aperta
-1 = nessun modem
-2 = modem già occupato

b = Modo di ottenere il numero

0 codice alfanumerico un carattere
1 codice numerico un carattere 0-9 #=11 *=12
2 numero chiuso da * o #

c = Globale o Locale su cui viene restituito il **numero**

d = Tempo massimo di attesa(default 5 secondi)in millisecondi

Esempio:

```
-GTL/L1,1,L2:5000
-JNE/L1,0,Fine
```

13.24 PTL Invia un file registrato su linea telefonica

PTL Invia un file registrato su linea telefonica

Funzione: Emette un file WAV sulla linea telefonica

Parametri: **a,b**

dove:
a = nome file WAV da utilizzare come suono
b = Locale o Globale di stato dove viene caricato l'esito dell'istruzione
0 = OK
-1 = Porta audio per l'emissione di un file già impegnata

Nota: se non specificato nella dichiarazione del file, viene ricercato nella directory ..\DAT

Esempio:

```
-CTL/0125632826,L1      ; chiama il numero telefonico 0125632826
-TMM/1000               ; attende 1 sec
-PTL/Start.wav          ; invia su telefono un messaggio di benvenuto
```

Esempio:

```
        -DIS/1:Attendo Chiamata
        -WTL
        -DIS/1:Chiamato
        -PTL/Start.wav
        -DIS/1:Fine Play
Loop-
        -GTL/L1,1,L2:5000
        -JNE/L1,0,Fine
        -DIS/5:valore carattere,L2,3,0
        -JEQ/L2,12,Fine
        -JNR/L2,0,9,Loop
        -JMP/L2
0       -PTL/Msg1.wav
        -JMP/Loop
1       -PTL/Msg1.wav
        -JMP/Loop
2-      -PTL/Msg1.wav
        -JMP/Loop
3       -PTL/Msg1.wav
        -JMP/Loop
4       -PTL/Msg1.wav
        -JMP/Loop
5       -PTL/Msg5.wav
        -JMP/Loop
6       -PTL/Msg5.wav
        -JMP/Loop
7       -PTL/Msg5.wav
        -JMP/Loop
8       -PTL/Msg5.wav
        -JMP/Loop
9       -PTL/Msg5.wav
        -JMP/Loop


Fine-
        -STL
        -DIS/1:Fine chiamata
```

13.25 EML Invia una E-MAIL

EML Invia una E-Mail

Funzione: Invia una E-Mail

Parametri: **a,b:c,d,e,f**

dove:
a = Numero della E-MAIL
b = Locale o Globale di stato dove viene caricato l'esito dell'istruzione
0 = OK
-1 = Non collegato

c = stringa che viene inviata come corpo della E-Mail
d = variabile il cui contenuto (valore numerico) viene accodato alla stringa suddetta
e = numero di cifre intere da visualizzare (0:10)
f = numero di cifre decimali da visualizzare (0:5)

I parametri della e-mail si trovano nel file COMUNICATION.TXT , che viene ricercato nella directory ..\DAT

Esempio:

```
-EML/1,L1:Messaggio ,G1,3,0        ;
```

Nota: Contenuto nel file COMUNICATION.TXT

```
[ParametriGenerali]
NumeroMail=1

[E-MAIL-Server]
ServerPosta=smtp…….xxxt
User=….@……..
Password=……

[E-MAIL]
Nome_Mittente_1=Tornio 1
Indirizzo_Mittente_1=andrea@bausano.net

Numero_Destinatari_1=2
Nome_Destinatario_1_1=Tullio
Indirizzo_Destinatario_1_1=tullio@bausano.net
Nome_Destinatario_1_2=Andrea
Indirizzo_Destinatario_1_2=andrea@bausano.net

Numero_Allegati_1=2
Allegato_1_1=c:\File1Minuto.TXT
Allegato_1_2=c:\File10Secondi.txt

Oggetto_1=Oggetto ( Prova di test )
Contenuto_1=""
```

Esempio:

```
;
; Inizializzazione
;
        -LBF/G1,G6:0
        -TSK/IncementaG1G3
        -TSK/IncementaG4G6

        -FOC/c:\File1Minuto.TXT
        -FOC/c:\File10Secondi.TXT

        -GDT/L11,L18
        -LET/L1,L18,L2,L18,L3,L18
;
; Attività temporizzate
;
Loop-
        -GDT/L11,L18

        -DIS/5:Anno         =,L11
        -DIS/6:Mese         =,L12
        -DIS/7:Giorno       =,L13
        -DIS/8:Ora          =,L14
        -DIS/9:Minuti       =,L15
        -DIS/10:Secondi     =,L16
        -DIS/11:MilliSecondi =,L17
        -DIS/12:Secondi 1970 =,L18,18,0

        -JGE/L18-L1,10*60,InviaEmailOgni10Minuti
        -JGE/L18-L2,1*60,ScrifiFileOgniMinuto
        -JGE/L18-L3,10,ScrifiFileOgni10Secondi

        -JMP/Loop
;
; Routine di servizio
;
InviaEmailOgni10Minuti-
        -LET/L1,L18

        -LET/L20,L14
        -LET/L20,L20+(L15/100)
```

```
        -EML/1,L21:Email di controllo alle ore,L20,5,2
        -JNE/L21,0,Loop

        -FOC/c:\File1Minuto.TXT
        -FOC/c:\File10Secondi.txt

        -JMP/Loop

ScrifiFileOgniMinuto-
        -LET/L2,L18
        -FWR/c:\File1Minuto.TXT,TIME,Ogni minuto, G1, G3
        -JMP/Loop

ScrifiFileOgni10Secondi-
        -LET/L3,L18
        -FWR/c:\File10Secondi.TXT,TIME,Ogni 10 secondi, G4, G6
        -JMP/Loop


;
; Tasks di gestione dati
;
PartProgram[IncementaG1G3]
Loop-
        -LET/G1,G1+1,G2,G2+2,G3,G3+3
        -JMP/Loop

PartProgram[IncementaG4G6]
Loop-
        -LET/G4,G4+1,G5,G5+2,G6,G6+3
        -JMP/Loop
```

CAPITOLO 14

14 Istruzioni AWL PLC

.1-NET	PLC Network
.2-LD	PLC Carica operazione
.3-LDN	PLC Carica operazione negata
.4-A	PLC Combina il valore di bit tramite AND
.5-AN	PLC Combina il valore di bit negato tramite AND
.6-O	PLC Combina il valore di bit tramite OR
.7-ON	PLC Combina il valore di bit negato tramite OR
.8-EU	PLC Rilevamento di fronte positivo
.9-ED	PLC Rilevamento di fronte negativo
.10-EQU	PLC Assegna, copia nel parametro specificato il valore superiore dello stack
.11-S	PLC Imposta ad 1 il numero di punti specificato se lo stack è 1
.12-R	PLC Imposta ad 0 il numero di punti specificato se lo stack è 1
.13-LPP	PLC Prelevamento logico
.14-LPS	PLC Duplicazione logica
.15-LRD	PLC Copiatura logica
.16-ALD	PLC Combina il primo e il secondo elemento tramite AND
.17-OLD	PLC Combina il primo e il secondo elemento tramite OR
.18-NOT	PLC Modifica del valore superiore
.19-LEQ	PLC Confronta due valori se uguali carica con lo stack 1 se no 0
.20-LGE	PLC Confronta due valori se uguali o maggiore carica con lo stack 1 se no 0
.21-LLE	PLC Confronta due valori se uguali o minore carica con lo stack 1 se no 0
.22-AEQ	PLC Confronta due valori se uguali fa l'AND di 1 con lo stack
.23-AGE	PLC Confronta due valori se uguali o maggiore fa l'AND di 1 con lo stack
.24-ALE	PLC Confronta due valori se uguali o minore fa l'AND di 1 con lo stack
.25-OEQ	PLC Confronta due valori se uguali fa l'OR di 1 con lo stack
.26-OGE	PLC Confronta due valori se uguali o maggiore fa l'OR di 1 con lo stack
.27-OLE	PLC Confronta due valori se uguali o minore fa l'OR di 1 con lo stack
.28-PEX	PLC Esegui su 1 in stack l'istruzione AXESBRAIN
.29-TON	PLC Timer senza ritenzione
.30-TOR	PLC Timer con ritenzione
.31-CTU	PLC Counter Up
.32-CUD	PLC Counter Up e Down

Operazioni speciali a contatti

Il **Contatto normalmente aperto** è chiuso (on) se il valore del bit interrogato dell'indirizzo **n è** 1.
In AWL il contatto normalmente aperto è rappresentato dalle operazioni del tipo:
Carica operazione, Combina il valore di bit tramite And, Combina il valore di bit tramite OR. Queste operazioni, rispettivamente, caricano il valore di bit dall'indirizzo n nel valore superiore dello stack logico, o combinano tramite And o OR il valore bit dell'indirizzo n con il valore superiore dello stack logico.

Contatto normalmente chiuso è chiuso (on) se il valore del bit interrogato dell'indirizzo n è 0
In AWL il contatto normalmente chiuso è rappresentato dalle operazioni del tipo:
Carica il valore di bit negato, Combina il valore di bit negato tramite And, Combina il valore di bit negato tramite OR.
Queste operazioni caricano il valore di bit dall'indirizzo n nel valore superiore dello stack logico, o combinano tramite And o Or il valore bit dell'indirizzo n con il valore superiore dello stack logico.

Le descritte operazioni ottengono il valore indirizzato dal registro delle immagini di processo, se tale valore è aggiornato all'inizio di ogni ciclo CPU.

Contatti diretti
Il **Contatto diretto normalmente aperto** è chiuso (on) se il valore del bit dell'ingresso fisico indirizzato n è **1.**
In AWL il contatto diretto normalmente aperto è rappresentato dalle operazioni del tipo
Carica il valore di bit direttamente, Combina bit direttamente tramite And, e Combina bit direttamente tramite OR.
Queste operazioni, rispettivamente, caricano direttamente il valore di bit dall'indirizzo n al valore superiore dello stack logico, o combinano direttamente tramite And o OR il valore bit dell'ingresso fisico indirizzato n con il valore superiore dello stack logico.

Il Contatto diretto normalmente chiuso è chiuso (on) se il valore del bit dell'ingresso fisico indirizzato n è 0.
In AWL il contatto diretto normalmente chiuso è rappresentato dalle operazioni del tipo **Carica il valore di bit negato direttamente, Combina direttamente** il **valore di bit negato tramite And, e Combina direttamente il valore di bit negato tramite OR.** Queste operazioni, rispettivamente, caricano direttamente il valore di bit negato dall'indirizzo n nel valore superiore dello stack logico, o combinano direttamente tramite And o OR il valore bit negato dell'indirizzo n con il valore superiore dello stack logico.
Le descritte operazioni ottengono il valore indirizzato dall'ingresso fisico quando vengono eseguite, ma il registro delle immagini di processo non viene aggiornato.

Contatto Not
Il contatto NOT modifica lo stato dei segnali. Se il flusso di corrente raggiunge il contatto Not, questo viene bloccato. Se il flusso non raggiunge il contatto Not, questo genera flusso di corrente.
In AWL l'operazione **Negazione del valore superiore** modifica il valore superiore dello stack da 0 a 1, o da 1 a O.

Normative semantiche

Il carattere **'#'** indica che il valore dell'espressione numerica è interpretata come valore 0 oppure 1.

Esempio 1:

```
-NET/1
-LD/#1                  ; Viene caricato il valore 1 nello STACK
```

Esempio 2:

; Esempio della dgestione dei Driver OK per dare e togliere le abilitazioni agli assi

```
-NET/1
-LD/DriveOKZ011
-A/DriveOKZ021
-A/DriveOKS011
-A/DriveOKS021
-PEX/-LET/L1,1
-NOT
-PEX/-LET/L1,0

-NET/2
-LD/#L1
-EU
-PEX/-PWO     ; Abilita potenza assi
-PEX/-RST/5   ; Reset Sistema senza condizionare i fronti degli I/O

-NET/3
-LD/#L1
-ED
-PEX/-EMC     ; Disabilita potenza assi
-PEX/-RST/5   ; Reset Sistema senza condizionare i fronti degli I/O
```

Esempio 3: Esempio di accenzione Macchina

ingresso	**"IN1"**	**Pulsante Inserimento Comandi**
uscita	**"OUT12"**	**Marcia Comandi**
Merker	**"M1.1"**	**Merker Marcia Comandi inserito**
Merker	**"M1.2"**	**Merker Merker Abilitazione Assi OK**

```
Loop   -TMM/100

       -NET/1
       -LD/M1.1
       -TON/1,5000

       -IDO/M1.1,1,MacchinaInserita

       -NET/2
       -LD/IN1
       -EU
       -EQU/M1.1          ; Inserimento comandi
       -S/OUT12           ; TLR marcia comandi ON
       -R/ M1.2           ; Merker Abilitazione Assi KO

       -IDO/M1.1,0, Loop

MacchinaInserita-

       -NET/3
       -LD/DriveOKZ011
       -A/DriveOKZ021
       -A/DriveOKS011
       -A/DriveOKS021
       -PEX/-LET/L1,1
       -NOT
       -PEX/-LET/L1,0

       -NET/4
       -LD/#L1
       -EU
       -PEX/-PWO          ; Abilita potenza assi
       -PEX/-RST/5        ; Reset Sistema senza condizionare i fronti degli I/O
       -S/M1.2            ; Merker Abilitazione Assi OK
       -PEX/-SEC/CLR_MESSAGE,1
```

```
        -NET/5
        -LD/#L1
        -ED
        -PEX/-EMC           ; Disabilita potenza assi
        -PEX/-RST/5         ; Reset Sistema senza condizionare i fronti degli I/O
        -R/M1.2             ; Merker Abilitazione Assi KO

        -IDO/M1.2,1, Loop

        -NET/4
        -LD/T1
        -PEX/-JMP/AzionamentiNonSalgono
        -JMP/ Loop

AzionamentiNonSalgono-
        -SEC/SET_MESSAGE,1
        -JMP/Loop
```

14.1 NET: PLC Network

NET PLC Network

Funzione: Inizializza un ramo NETWORK logico

Parametri: **a**

dove:
a = Nome del ramo da 1 a 128

Esempio:

- NET/1

14.2 LD: PLC Carica operazione

LD **PLC Carica operazione nello stack**

Funzione: Carica l'operatore nello stack

Parametri: **a**

dove:
a = Nome dell'operatore:
Input
Output
Timer 1-32
Contatore 1-32

Esempio:

```
-NET/1
-LD/T1
```

14.3 LDN: PLC Carica operazione negata

LDN **PLC Carica operazione negata nello stack**

Funzione: Carica l'operatore negato nello stack

Parametri: **a**

dove:
a = Nome dell'operatore:
Input
Output
Timer 1-32
Contatore 1-32

Esempio:

```
-NET/1
-LDN/T1
```

14.4 A: PLC Combina il valore di bit tramite AND

A PLC combina il valore di bit tramite AND

Funzione: Combina il valore del bit con lo stack tramite AND

Parametri: **a,…,an**

dove:

a,…,an = Nome dell'operatore:
Input
Output
Timer 1-32
Contatore 1-32

Esempio:

```
-NET/1
-LD/I_1
-A/I_12
```

14.5 AN: PLC Combina il valore di bit negato tramite AND

A PLC combina il valore di bit negato tramite AND

Funzione: Combina il valore del bit negato con lo stack tramite AND

Parametri: **a,…,an**

dove:

a,…,an = Nome dell'operatore:
Input
Output
Timer 1-32
Contatore 1-32

Esempio:

```
-NET/1
-LD/I_1
-AN/I_12
```

14.6 O: PLC Combina il valore di bit tramite OR

O **PLC combina il valore di bit tramite OR**

Funzione: Combina il valore del bit con lo stack tramite OR

Parametri: **a,…,an**

dove:

a,…,an = Nome dell'operatore:
Input
Output
Timer 1-32
Contatore 1-32

Esempio:

```
-NET/1
-LD/I_1
-O/I_12
```

14.7 ON: PLC Combina il valore di bit negato tramite OR

ON **PLC combina il valore di bit negato tramite OR**

Funzione: Combina il valore del bit negato con lo stack tramite OR

Parametri: **a,…,an**

dove:

a,…,an = Nome dell'operatore:
Input
Output
Timer 1-32
Contatore 1-32

Esempio:

```
-NET/1
-LD/I_1
-ON/I_12
```

14.8 EU: PLC Rilevamento fronte positivo

EU **PLC rileva fronte positivo**

Funzione: Rileva il fronte positivo nel valore superiore dello stack

Transizione positiva permette alla corrente di circolare per un ciclo di scansione, per ogni transizione da 0 (off) a 1 (on).
In AWL questa operazione viene rappresentata come Rilevamento di **fronte positivo.** Se essa rileva da un ciclo di scansione all'altro una transizione da **0** a **1** nel valore superiore dello stack, tale valore viene **impostato a 1**; altrimenti a 0.

Parametri: **nessuno**

Esempio:

```
-NET/1
-LD/I_1
-ED
```

14.9 ED: PLC Rilevamento fronte negativo

ED **PLC rileva fronte positivo**

Funzione: Rileva il fronte positivo nel valore superiore dello stack

Transizione negativa permette alla corrente di circolare per un ciclo di scansione, per ogni transizione da 1 (on) a 0 (off).
In AWL questa operazione viene rappresentata come **Rilevamento di fronte negativo.** Se essa rileva da un ciclo di scansione all'altro una **transizione da 1 a 0** nel valore superiore dello stack, tale valore viene **impostato a 1**; altrimenti a 0.

Parametri: **nessuno**

Esempio:

```
-NET/1
-LD/I_1
-ED
```

14.10 EQU: PLC Assegna, copia nel parametro specificato il valore superiore dello stack

EQU PLC Assegna, copia nel parametro specificato il valore superiore dello stack

Funzione: Assegna, copia nel parametro specificato il valore superiore dello stack

Assegna

Quando viene eseguita l'operazione Assegna, viene attivato il parametro specificato (n).

In AWL l'operazione Assegna copia nel parametro specificato (n) il valore superiore dello stack.

Parametri: **a,…,an**

dove:

a,…,an = Nome dell'operatore:
Output
Input

Esempio:

```
-NET/1
-LD/I_1
-EQU/O_12
```

14.11 S: PLC Imposta ad 1 il numero di punti specificato se lo stack è 1

S PLC Imposta ad 1 il numero di punti specificato se lo stack è 1

Funzione: Imposta ad 1 il numero di punti specificato se lo stack è 1

Imposta
Quando sono eseguite le operazioni **Imposta** viene impostato (attivato) il numero di punti specificato dall'operando

Parametri: **a,…,an**

dove:
a,…,an = Nome dell'operatore:
Output
Input

Esempio:

```
-NET/1
-LD/I_1
-S/O_12
```

14.12 R: PLC Imposta ad 0 il numero di punti specificato se lo stack è 1

R PLC Imposta ad 0 il numero di punti specificato se lo stack è 1

Funzione: Imposta ad 0 il numero di punti specificato se lo stack è 1

Resetta

Quando sono eseguite le operazioni **Resetta** (disattivato) il numero di punti specificato dall'operando
Utilizzando l'operazione Resetta, se l'operando è indicato come un bit T o C, vengono azzerati il bit di temporizzazione/conteggio o il valore corrente di temporizzazione/conteggio.

Parametri: **a,…,an**

dove:
a,…,an = Nome dell'operatore:
Output
Input

Esempio:

```
-NET/1
-LD/I_1
-R/O_12
```

14.13 LPS: PLC Duplicazione logica

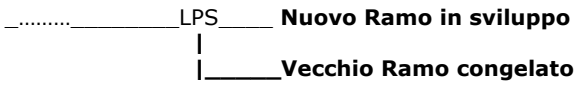
LPS PLS Duplicazione logica

Funzione: Duplicazione logica

L'operazione **Duplicazione logica** duplica il valore superiore dello stack e colloca questo valore nello stack.
Il valore più basso al fondo dello stack viene traslato fuori e va perso.

Parametri: **nessuno**

Questa funzione è importante essendo l'inizio di apertura di un ramo in parallelo



Esempio:

```
-NET/1
-LD/I_1
-A/I_2
-LPS
-A/I_3
```

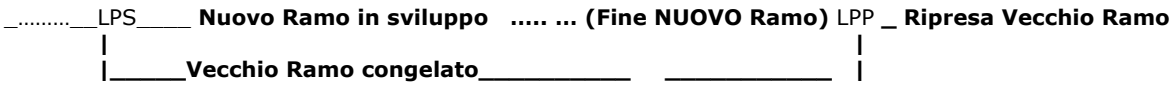
14.14 LPP: PLC Prelevamento logico

LPP LPP Prelevamento logico

Funzione: Prelevamento logico

L'operazione **Prelevamento logico** trasla fuori il valore alla sommità dello stack.
Il secondo valore dello stack diventa il nuovo valore alla sommità dello stack.
Dopo l'esecuzione di LPP, la profondità dello stack viene decrementata di uno.

Parametri: **nessuno**



Esempio:

```
-NET/1
-LD/I_1
-LD/I_1
-A/I_2
-LPS     ; Nuovo ramo con salvataggio del vecchio
-A/I_3
-PEX/-DIS/Valore Ramo Nuovo a 1
-NOT
-PEX/-DIS/Valore Ramo Nuovo a 0
-LPP     ; Abbandono nuovo ramo e ripresa del vecchio
-A/I_2
-……
```

14.15 LRD: PLC Copiatura logica

LRD PLC Copiatura logica

Funzione: Copiatura logica

L'operazione **Copiatura logica** carica il secondo valore dello stack nella sommità dello stesso.
Il valore non viene né collocato né prelevato dallo stack, ma il valore che stava alla sua sommità viene sovrascritto da quello nuovo.

Parametri: **nessuno**

```
_.........__LPS_____ Nuovo Ramo in sviluppo   ... (Fine NUOVO Ramo) LPP _ Ripresa Vecchio Ramo
          |                                                                  |
          |_____Vecchio Ramo congelato_____________   _____________|_____ Vecchio Ramo congelato
```

Esempio:

```
-NET/1
-LD/I_1
-LD/I_1
-A/I_2
-LPS      ; Nuovo ramo con salvataggio del vecchio
-A/I_3
-PEX/-DIS/Valore Ramo Nuovo a 1
-NOT
-PEX/-DIS/Valore Ramo Nuovo a 0
-LRD      ; Abbandono nuovo ramo e ripresa del vecchio con sua continuazione del ramo congelato
-A/I_2
-……
```

14.16 ALD: PLC Combina il primo e il secondo elemento tramite AND

ALD PLC Combina il primo e il secondo elemento tramite AND

Funzione: Combina il primo e il secondo elemento tramite AND

L'operazione **Combina primo e secondo livello tramite AND** combina i valori del primo e del secondo livello dello stack usando un'operazione logica combinatoria AND.
Il risultato viene caricato nella sommità dello stack.
Dopo l'esecuzione di ALD, la profondità dello stack viene decrementata di uno.

Parametri: **nessuno**

```
_.........__LPS_____ Nuovo Ramo in sviluppo   ... _________ALD ( Continua  Nuovo in AND con il Vecchio )
          |                                                  |
          |_____Vecchio Ramo congelato_____________    _|
```

Esempio:

```
a         -
          -SDO/M1.1,1  ; a = ( M1.1 And M1.2  )
          -SDO/M1.2,1
          -SDO/M1.3,1  ; b = ( M1.3 Or M1.4 )
          -SDO/M1.4,0
          -SDO/M1.5,1  ; r = M1.5 And  a And  b


          -NET/1
          -LD/M1.1   ; a = ( M1.1 And M1.2  )
          -A/M1.2
          -LPS
          -LD/M1. 1  ; b = ( M1.3 Or M1.4 ))
          -O/M1.4
          -LPS
          -LD/M1.5   ; r = M1.5 And  a And  b
          -ALD
          -ALD

          -PEX/-DIS/Stack valore 1
          -NOT
          -PEX/-DIS/Stack valore 0
          -JMP/a
```

Il valore visualizzato dalla DIS è uguale a **Stack valore 1**

14.17 OLD: PLC Combina il primo e il secondo elemento tramite OR

OLD PLC Combina il primo e il secondo elemento tramite OR

Funzione: Combina il primo e il secondo elemento tramite OR

L'operazione Combina primo e secondo livello tramite OR esegue una combinazione logica OR dei valori di bit nel primo (sommità) e secondo livello dello stack. lì risultato viene caricato nella sommità dello stack.
Dopo l'esecuzione di OLD, la profondità dello stack si riduce di uno.

Parametri: **nessuno**

```
_.....__LPS____ Nuovo Ramo in sviluppo   ... _________OLD ( Continua  Nuovo in OR con il Vecchio )
        |                                                |
        |_____Vecchio Ramo congelato_____________    _|
```

Esempio:

```
a       -
        -SDO/M1.1,1  ; a = ( M1.1 And M1.2  )
        -SDO/M1.2,0
        -SDO/M1.3,1  ; b = ( M1.3 And M1.4 )
        -SDO/M1.4,0
        -SDO/M1.5,0  ; r = M1.5 Or a Or b


        -NET/1
        -LD/M1.1   ; a = ( M1.1 And M1.2  )
        -A/M1.2
        -LPS
        -LD/M1.3   ; b = ( M1.3 And M1.4 )
        -A/M1.4
        -LPS
        -LD/M1.5   ; r = M1.5 Or a Or b
        -OLD
        -OLD

        -PEX/-DIS/Stack valore 1
        -NOT
        -PEX/-DIS/Stack valore 0
        -JMP/a
```

Il valore visualizzato dalla DIS è uguale a **Stack valore 0**

14.18 NOT: PLC Modifica del valore superiore

NOT PLC Modifica del valore superiore dello stack

Funzione: Modifica del valore superiore dello stack

Parametri: **nessuno**

Esempio:

```
-NET/1
-LD/I_1
-NOT
```

14.19 LEQ: PLC Confronta due valori se uguali carica con lo stack 1 se no 0

LEQ PLC Confronta due valori se uguali carica con lo stack 1 se no 0

Funzione: PLC Confronta due valori se uguali carica con lo stack 1 se no 0.

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 = n2

In AWL, se il confronto è vero, questa operazione rispettivamente caricano un 1 nel valore superiore dello stack logico

I confronti di numeri reali sono con segno.

Avvertenza: si potrà creare un confronto **<>,** <o > utilizzando l'operazione Not con l'operazione di confronto =. La sequenza qui riportata è equivalente a un confronto <> di L1 con 50:

-LEQ L1, 50.0
-NOT

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-LEQ/L1,6
```

14.20 LGE: PLC Confronta due valori se uguali o maggiore carica con lo stack 1 se no 0

LEQ PLC Confronta due valori se maggiore uguali carica con lo stack 1 se no 0

Funzione: PLC Confronta due valori se uguali o maggiore carica con lo stack 1 se no 0.

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 >= n2

In AWL, se il confronto è vero, questa operazione rispettivamente caricano un 1 nel valore superiore dello stack logico

I confronti di numeri reali sono con segno.

Avvertenza: si potrà creare un confronto **<,** utilizzando l'operazione Not con l'operazione di confronto > =. La sequenza qui riportata è equivalente a un confronto < di L1 con 50:

```
-LGE  L1, 50.0
-NOT
```

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-LGE/L1,6
```

14.21 LLE: PLC Confronta due valori se uguali o minore carica con lo stack 1 se no 0

LEQ PLC Confronta due valori se minore uguali carica con lo stack 1 se no 0

Funzione: PLC Confronta due valori se uguali o minore carica con lo stack 1 se no 0.

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 <= n2

In AWL, se il confronto è vero, questa operazione rispettivamente caricano un 1 nel valore superiore dello stack logico

I confronti di numeri reali sono con segno.

Avvertenza: si potrà creare un confronto **>,** utilizzando l'operazione Not con l'operazione di confronto < =. La sequenza qui riportata è equivalente a un confronto **>** di L1 con 50:

-LLE/ L1, 50.0
-NOT

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-LLE/L1,6
```

14.22 AEQ: PLC Confronta due valori se uguali fa l'AND di 1 con lo stack

AEQ PLC Confronta due valori se uguali fa l'AND di 1 con lo stack

Funzione: PLC Confronta due valori se uguali fa l'AND di 1 con lo stack

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 = n2

In AWL se il confronto è vero combina tramite AND un 1 con il valore superiore dello stack logico.

I confronti di numeri reali sono con segno.

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-AEQ/L1,6.
```

14.23 AGE: PLC Confronta due valori se uguali o maggiore fa l'AND di 1 con lo stack

AEQ PLC Confronta due valori se uguali o maggiore fa l'AND di 1 con lo stack

Funzione: PLC Confronta due valori se uguali o maggiore fa l'AND di 1 con lo stack

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1> = n2

In AWL se il confronto è vero combina tramite AND un 1 con il valore superiore dello stack logico.

I confronti di numeri reali sono con segno.

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-AGE/L1,6.
```

14.24 ALE: PLC Confronta due valori se uguali o minore fa l'AND di 1 con lo stack

AEQ PLC Confronta due valori se uguali o minore fa l'AND di 1 con lo stack

Funzione: PLC Confronta due valori se uguali o minore fa l'AND di 1 con lo stack

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 < = n2

In AWL se il confronto è vero combina tramite AND un 1 con il valore superiore dello stack logico.

I confronti di numeri reali sono con segno.

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-ALE/L1,6.
```

14.25 OEQ: PLC Confronta due valori se uguali fa l'OR di 1 con lo stack

AEQ PLC Confronta due valori se uguali fa l'OR di 1 con lo stack

Funzione: PLC Confronta due valori se uguali fa l'OR di 1 con lo stack

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 = n2

In AWL se il confronto è vero combina tramite OR un 1 con il valore superiore dello stack logico.

I confronti di numeri reali sono con segno.

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-OEQ/L1,6.
```

14.26 OGE: PLC Confronta due valori se uguali o maggiore fa l'OR di 1 con lo stack

AEQ PLC Confronta due valori se uguali o maggiore fa l'OR di 1 con lo stack

Funzione: PLC Confronta due valori se uguali o maggiore fa l OR di 1 con lo stack

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1> = n2

In AWL se il confronto è vero combina tramite OR un 1 con il valore superiore dello stack logico.

I confronti di numeri reali sono con segno.

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-OGE/L1,6.
```

14.27 OLE: PLC Confronta due valori se uguali o minore fa l'OR di 1 con lo stack

AEQ PLC Confronta due valori se uguali o minore fa l'OR di 1 con lo stack

Funzione: PLC Confronta due valori se uguali o minore fa l'OR di 1 con lo stack

L'operazione **Confronto di numeri reali** viene utilizzata per confrontare due valori: n1 e n2. Si potrà eseguire un confronto n1 < = n2

In AWL se il confronto è vero combina tramite OR un 1 con il valore superiore dello stack logico.

I confronti di numeri reali sono con segno.

Parametri: **a,b**
dove:

a = variabile o vocabolo di sistema inerente il test

b = valori da confrontare

Esempio:

```
-NET/1
-OLE/L1,6.
```

14.28 PEX: PLC Esegue su 1 in stack l'istruzione AXESBRAIN

PEX PLC Esegue su 1 in stack l'istruzione AXESBRAIN

Funzione: Esegue su 1 in stack l'istruzione AXESBRAIN

Parametri: **a**

dove:

a = istruzione AXESBRAIN

Esempio:

```
-NET/1
-LD/I_12
-PEX/-DIS/1 :prova
```

14.29 TON: PLC Timer senza ritenzione

14.30 TOR: PLC Timer con ritenzione

TON TOR PLC Timer senza ritenzione e Timer con ritenzione

Funzione: PLC Timer senza ritenzione e Timer con ritenzione
Avvia temporizzazione come ritardo all'inserzione (con memoria)

Se abilitate, le operazioni Avvia temporizzazione come ritardo all'inserzione e Avvia temporizzazione come ritardo all'inserzione con memoria misurano il tempo sino al valore massimo. Quando il valore corrente (Txxx) è >= al tempo di default (PI), il bit di temporizzazione viene attivato.

Se disattivati, il temporizzatore come ritardo all'inserzione si resetta, mentre il temporizzatore come ritardo all'inserzione con memoria semplicemente si arresta. Entrambi i temporizzatori si arrestano al raggiungimento del valore massimo.
Operandi: Txxx: TON TOR

I temporizzatori vengono anche utilizzati per implementare le funzioni di conteggio comandate a tempo. Il sistema fornisce due diverse operazioni di temporizzazione: avvia temporizzazione come ritardo all'inserzione (TON) e avvia temporizzazione come ritardo all'inserzione con memoria (TOR). I temporizzatori differiscono nel modo in cui reagiscono allo stato dell'ingresso di abilitazione. Sia TON che TOR contano in avanti mentre è attivo (on) l'ingresso di abilitazione:
entrambi non contano in avanti mentre è disattivato (0ff) l'ingresso di abilitazione, ma mentre il temporizzatore TON si resetta automaticamente, il temporizzatore TOR non si resetta e mantiene il suo ultimo valore. Pertanto si utilizzerà al meglio TON per la temporizzazione di un singolo intervallo. TOR sarà più appropriato per accumulare un numero di intervalli di tempo.

I temporizzatori hanno le seguenti caratteristiche.

- I temporizzatori vengono controllati con un singolo ingresso di abilitazione, e hanno un valore corrente che conserva il tempo trascorso da quando il temporizzatore è stato abilitato, I temporizzatori hanno anche un valore di default (PT), che viene confrontato con il

valore corrente ogni volta che quest'ultimo viene aggiornato, e se viene eseguita l'operazione di temporizzazione.

- Un bit di temporizzazione viene impostato o resettato in base al risultato del confronto del valore corrente con il valore di default.

- Se il valore corrente è maggiore o uguale al valore di default, il bit di temporizzazione (bit T) viene attivato.

Avvertenza
Alcuni valori correnti di temporizzazione possono essere a ritenzione, I bit di temporizzazione non sono a ritenzione, e vengono impostati solo come risultato di un confronto tra valore corrente e valore di default.

Se si resetta un temporizzatore, viene azzerato il suo valore corrente e disattivato il bit T. Qualsiasi temporizzatore può essere resettato con l'operazione Resetta, ma un temporizzatore TOR può essere resettato unicamente da questa operazione. Il bit di temporizzazione di un temporizzatore non viene resettato scrivendo uno zero come valore corrente del temporizzatore. Lo stesso vale al rovescio: scrivendo uno zero nel bit T del temporizzazione non si resetta il suo valore corrente.

Aggiornamento di temporizzatori con risoluzione a 1 ms

La CPU fornisce temporizzatori che si aggiornano una volta a millisecondo (temporizzatori da i ms) per mezzo della routine di sistema che memorizza la base di tempo di sistema. Si tratta dì temporizzatori che garantiscono l'esatto controllo delle operazioni.

L'aggiornamento è automatico in quanto il valore corrente di un temporizzatore attivo da i ms viene aggiornato appunto da una routine di sistema. Una volta abilitata, l'esecuzione di una operazione TON/TOR che controlla un temporizzatore da i ms è richiesta solo per controllare lo stato attivato/disattivato del temporizzatore.

Poiché il valore corrente e il bit T dei temporizzatori da i ms vengono aggiornati da una routine di sistema (indipendente dal ciclo del controllore programmabile e del programma utente), questi valori possono essere aggiornati in qualsiasi punto del ciclo, e ciò awerrà più di una volta per ciclo se il tempo di ciclo supera un millisecondo. Non è pertanto garantito che tali valori rimangano costanti per tutta una determinata esecuzione del programma utente principale.
Il reset di un temporizzatore da 1 ms abilitato lo disattiva, resetta a zero il suo valore corrente e azzera il suo bit .

Parametri: **a,b**

dove:

a = numero timer (massimo 32)
b = valore in millisecondi

Esempio :

```
-NET/6
-LD /Q_InMac.15
-TON/1,3000

-NET/7
-LD /T1
-EQU/Q_InMac.16
```

Esempio :

```
a       -NET/1
        -LDN/T1
        -TON/2,1000

        -NET/2
        -LD/T2
        -TON/1,1000

        -NET/3
        -LD/T2
        -PEX/-DIS/pippo
        -NOT
        -PEX/-DIS/

        -JMP/a
```

14.31 CTU: PLC Counter Up

14.32 CUD: PLC Counter Up e Down

CTU CUD PLC Counter Up e Counter Up e Down

Funzione: PLC Contatore in avanti e Contatore avanti ed indietro

Dettagli delle operazioni di conteggio

L'operazione Conta in avanti (CTU) conta in avanti a partire dal valore corrente del contatore ogni volta che l'ingresso di conteggio in avanti passa da off a on. lì contatore viene resettato quando si attiva l'ingresso di reset o quando viene eseguita l'operazione Reset. Esso si arresta al raggiungimento del valore massimo (32.767).

L'operazione Conta in avanti/indietro (CUD) conta in avanti ogni volta che l'ingresso di conteggio in avanti passa da off a on e conta all'indietro ogni volta che l'ingresso di conteggio all'indietro passa da off a on. lì contatore viene resettato quando si attiva l'ingresso di reset o quando viene eseguita l'operazione Reset. Al raggiungimento del valore massimo (32.767), il fronte di salita successivo dell'ingresso di conteggio in avanti farà sì che il valore corrente si raccolga intorno al valore minimo (-32.768). Analogamente, al raggiungimento del valore minimo (-32.768) il successivo fronte di salita dell'ingresso di conteggio all'indietro farà sì che il valore corrente si raccolga intorno al valore massimo (32.767).

Se l'utente resetta un contatore con l'operazione Resetta, sia il bit di conteggio sia il valore corrente di conteggio saranno resettati.

I contatori in avanti e in avanti/all'indietro hanno un valore corrente che memorizza il conteggio corrente. Essi hanno anche un valore di default (PV) che viene confrontato con il valore corrente ogni volta che viene eseguita l'operazione di conteggio. Se il valore corrente è maggiore o uguale al valore di default, si attiva il bit di conteggio (bit C). Altrimenti il bit C si disattiva.

Si utilizzi il numero del contatore per far riferimento sia al valore corrente che al bit C del contatore stesso.

Avvertenza
Vi è solamente un valore corrente per ogni contatore. Non si assegni perciò lo stesso numero a più di un contatore (i contatori in avanti e in avanti/indietro accedono allo stesso valore corrente).

Definisci modo per contatore veloce, Attiva contatore veloce

Parametri: **a,b**

dove:

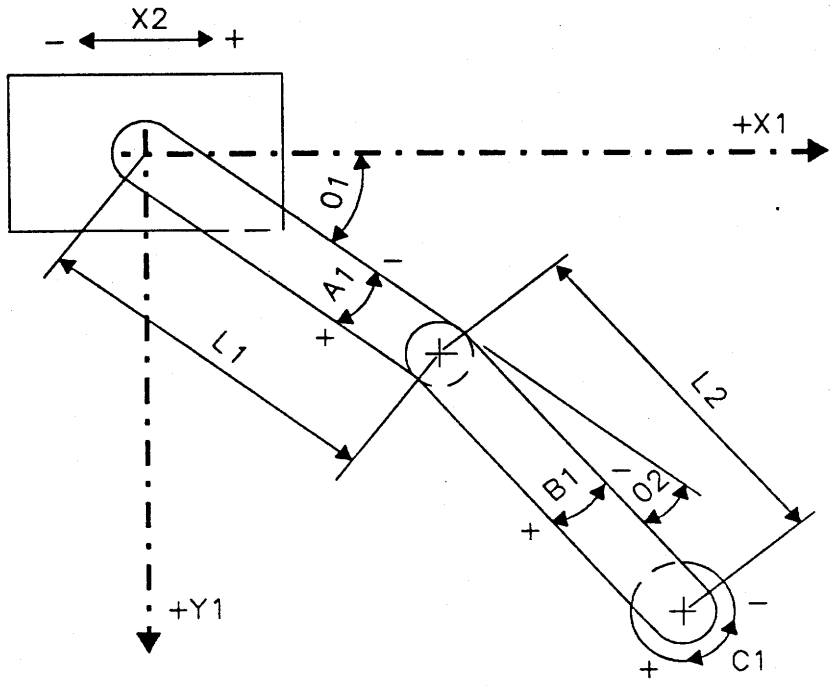
a = numero contatore (massimo 32)
b = valore finale

APPENDICE A

A. Assi Virtuali

Gli assi virtuali, come dice la parola stessa, non sono assi fisici ma assi che per comodità utente vengono caratterizzati su sistemi a geometria SCARA o CILINDRICA in modo che oltre ad effettuare movimenti sugli assi fisici si possa anche effettuare movimenti secondo un sistema cartesiano.
Con tale sistema, inoltre si possono effettuare interpolazioni lineari e circolari.

Sistema SCARA



Numero di assi SCARA: indica il numero di assi da prendere in considerazione; minimo 2 (i primi 2); tre (1,2,3) oppure 4 (tutti)

Nell'esempio riportato, vengono presi in considerazione tutti e 4 (i 2 snodi A1 e B1,il 3° asse rotante C1 posto all'estremità del secondo snodo e il 4° asse traslante X2)

Nome del primo asse: indica il primo asse reale del sistema (1° snodo); nell'esempio l'asse A1

Lunghezza del primo asse: indica la lunghezza del primo asse (distanza fra il primo e secondo snodo); nell'esempio L1

Nome del secondo asse: indica il secondo asse reale del sistema (2° snodo); nell'esempio B1

Lunghezza del secondo asse: indica la lunghezza del secondo asse (distanza fra snodo centrale e asse pinza); nell'esempio L2.
Nel caso sia presente il terzo asse (nell'esempio C1), l'asse pinza deve essere concentrico con l'asse di rotazione del terzo asse (in questo caso la lunghezza del secondo asse è fissa). Se invece vi è eccentricità, la lunghezza del secondo asse (rispetto all'asse pinza) è variabile, essendo dipendente dalla posizione angolare della pinza. Di conseguenza, poiché si imposta una sola lunghezza, il sistema cartesiano ottenuto sarà valido solo in quelle condizioni.

In base a queste considerazioni, in caso di eccentricità, occorre tener presente quanto segue:

- non è possibile ottenere traiettorie rettilinee e circolari e l'errore di traiettoria sarà tanto maggiore quanto maggiore è l'eccentricità.

Nome del terzo asse: indica l'eventuale asse rotante posto all'estremità del secondo asse; nell'esempio C1.
L'indicazione o meno di questo asse in tabella, influisce solo sulla geometria del sistema agli effetti della compensazione automatica di detto rotante e precisamente:

- Asse non indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante viene ignorato (naturalmente se non richiesto nel movimento)

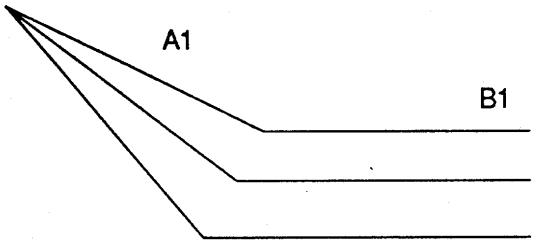
- Asse indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante (naturalmente se non richiesto nel movimento) viene automaticamente compensato; la pinza posta su tale asse rimane quindi sempre parallela a se stessa.
  E' certo che se questo asse viene richiesto nel movimento, l'indicazione o meno dell'asse nella tabella non ha più alcun significato essendo il movimento richiesto determinante.

Nome del quarto asse: indica l'eventuale asse traslante (traslazione di tutto Io SCARA); nell'esempio X2

Compensazione meccanica: questo dato serve per individuare il tipo di robot secondo la convenzione seguente:

0 = robot di tipo SCARA tradizionale
± 1 = robot di tipo SCARA ma con sistema a pantografo

dove muovendo un solo asse (nell'esempio sottostante A1) il secondo asse (nell'esempio B1) si mantiene parallelo a se stesso.



Il segno algebrico va inteso nel modo seguente:
Con il segno positivo si intende la compensazione automatica sull'asse B1 sommando l'angolo effettuato dall'asse A1.

Con il segno negativo si intende la compensazione automatica sull'asse B1 sottraendo l'angolo effettuato dall'asse A1.

Offset assoluto: questo dato viene utilizzato ogni volta si desideri ottenere uno zero asse pratico diverso da quello teorico.

Massima velocità: corrisponde alla massima velocità voluta sull'asse virtuale; se questa velocità fosse impossibile da effettuare (occorre tener presente che i due assi reali abbinati all'asse virtuale hanno ognuno una massima velocità) per default viene utilizzata la massima possibile.

Massima accelerazione e massima decelerazione: corrispondono alla massima accelerazione e decelerazione volute sull'asse virtuale; anche per questi dati se i valori indicati fossero superiori al possibile, per default vengono utilizzati i massimi possibili.

Note:

a) Poiché tutti i calcoli per gli assi virtuali vengono eseguiti secondo la geometria classica operando su seni e coseni di angoli espressi in gradi, occorre tenere presente quanto segue:

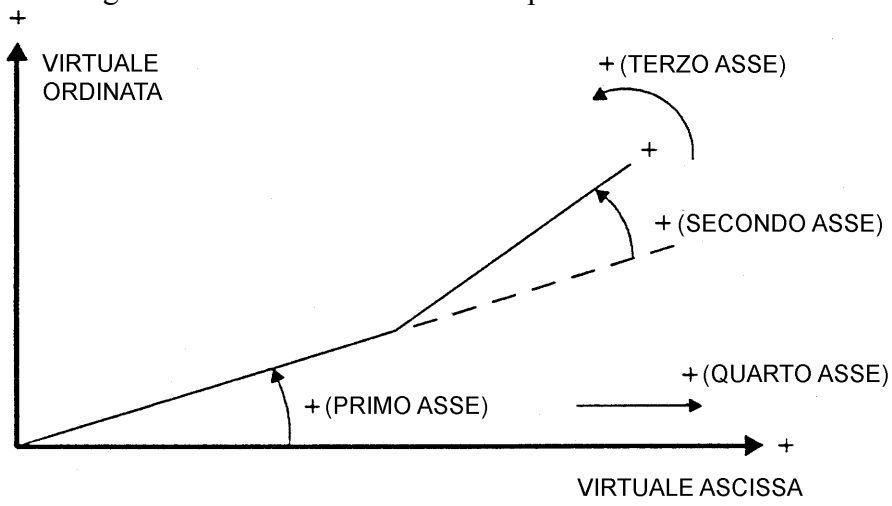
1 - L'unità di misura dei primi 3 assi reali deve essere il grado.

2 - L'unità di misura del quarto asse, può essere il mm o pollici a seconda che le lunghezze primo e secondo asse vengano date rispettivamente in mm o inc.

3 - L'offset assoluto del primo asse reale deve essere dato (se è presente il 4° asse traslante) in modo che l'asse virtuale ascissa sia parallelo al quarto asse SCARA.

4 - L'offset assoluto del secondo asse reale deve essere dato in modo che i primi due assi risultino allineati (braccio tutto teso).

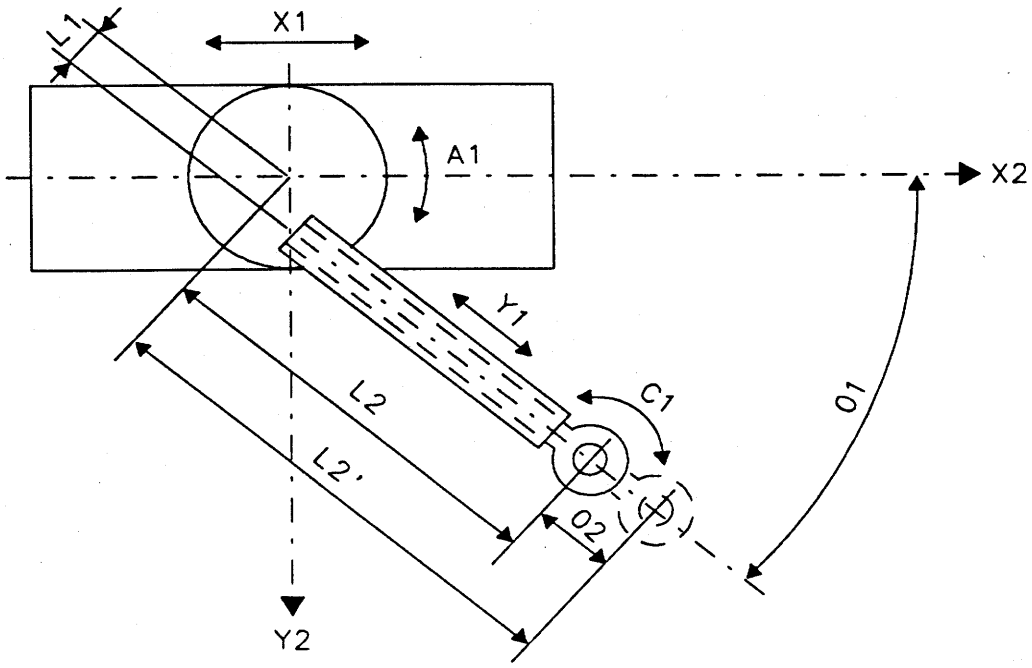
5 - Le direzioni di movimento (positiva, negativa) degli assi del sistema devono essere tutte concordi: per i primi 3 assi rotanti direzione positiva significa andare da ascissa positiva a ordinata positiva; l'eventuale quarto asse deve avere la stessa direzione della ascissa. La figura successiva fornisce un esempio di dette direzioni.



6 - In base alle considerazioni viste nei punti 1-5 precedenti, si avrà che dopo un movimento a zero di tutti e 4 gli assi reali SCARA, l'asse virtuale ascissa si troverà ad una coordinata equivalente alla somma delle lunghezze del primo e secondo asse; l'asse virtuale ordinata a coordinata zero.

b) Con un sistema SCARA, non è possibile richiedere tramite una stessa istruzione di movimento, assi virtuali unitamente ai primi due assi reali del sistema (uno dei due o entrambi.
E' viceversa possibile richiedere assi virtuali unitamente al 3° e/o 4° asse reale del sistema.

Sistema CILINDRICO



Numero di assi del sistema CILINDRICO: indica il numero di assi da prendere in considerazione; 1°+2°,1°+2°+ 3°,1°+2°+3°+ 4°. Si consiglia di indicare sempre 4 e se qualche asse non è presente (2°, 3°, 4°) non indicarlo nelle righe apposite.
Nell'esempio riportato, vengono presi in considerazione tutti e 4 gli assi.

Nome del primo asse: indica il primo asse reale del sistema; nell'esempio A1. Il primo asse è sempre l'asse rotante generale e deve essere sempre presente

Lunghezza del primo asse: indica la distanza (può anche essere 0) fra l'asse di rotazione del primo asse e l'asse pinza; tale distanza va data perpendicolarmente all'asse cannotto (se cannotto mancante, al braccio porta pinza); nell'esempio L1.

Nome del secondo asse: indica il secondo asse reale del sistema; nell'esempio Y1.
Tale asse è sempre l'asse cannotto e potrebbe anche non essere presente; in tale caso deve però essere necessariamente presente il quarto asse

Lunghezza del secondo asse: indica la distanza fra l'asse di rotazione del primo asse e l'asse pinza; tale distanza va data parallelamente all'asse cannotto (se cannotto mancante, al braccio porta pinza); nell'esempio, L2 se il secondo asse (cannotto) non ha offset assoluto; 12', se questo asse ha un suo offset (02).
Nel caso sia presente il terzo asse (nell'esempio C1), l'asse pinza deve essere concentrico con l'asse di rotazione del terzo asse (in questo caso la lunghezza del secondo asse è fissa). Se invece vi è eccentricità, la lunghezza del secondo asse (rispetto all'asse pinza) è variabile, essendo indipendente dalla posizione angolare della pinza. Di conseguenza, poiché si imposta una sola lunghezza, il sistema cartesiano ottenuto sarà valido solo in quelle condizioni.
In base a queste considerazioni, in caso di eccentricità, occorre tener presente quanto segue:

- non è possibile ottenere traiettorie rettilinee e circolari e l'errore di traiettoria sarà tanto maggiore quanto maggiore è l'eccentricità.

Nome del terzo asse: indica l'eventuale asse rotante posto all'estremità del cannotto (o braccio porta pinza se cannotto mancante); nell'esempio C1.
L'indicazione o meno di questo asse in tabella, influisce solo sulla geometria del sistema agli effetti della compensazione automatica di detto rotante e precisamente:

- Asse non indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante viene ignorato (naturalmente se non richiesto nel movimento)

- Asse indicato in tabella: in caso di richiesta movimento assi (reali o virtuali) tale asse rotante (naturalmente se non richiesto nel movimento) viene automaticamente compensato; la pinza posta su tale asse rimane quindi sempre parallela a se stessa.
  E' certo che se questo asse viene richiesto nel movimento, l'indicazione o meno dell'asse nella tabella non ha più alcun significato essendo il movimento richiesto determinante.

Nome del quarto asse: indica l'eventuale asse traslante (traslazione di tutto il CILINDRICO); nell'esempio X1. Tale asse potrebbe anche non essere presente purché sia presente il secondo asse;

Compensazione meccanica: tale dato va sempre dato con valore 0.

Offset assoluto: questo dato viene utilizzato ogni qualvolta si desideri ottenere uno zero asse pratico diverso da quello teorico.

Massima velocità: corrisponde alla massima velocità voluta sull'asse virtuale; se questa velocità fosse impossibile da effettuare, (ogni asse reale appartenente al sistema ha una propria massima velocità) per default viene utilizzata la massima possibile.

Massima accelerazione e decelerazione: corrispondono alla massima accelerazione e decelerazione volute sull'asse virtuale; anche per questi dati se i valori indicati fossero superiori al possibile, per default vengono utilizzati i massimi possibili.

Note:

a) Poiché tutti i calcoli per gli assi virtuali vengono eseguiti secondo la geometria classica, operando su seni e coseni di angoli espressi in gradi, occorre tenere presente quanto segue:

1 - L'unità di misura degli assi rotanti (primo e terzo asse) reali deve essere il grado.

2 - L'unità di misura del quarto asse reale può essere il mm o l'inc. a seconda che la lunghezza secondo asse venga data rispettivamente in mm o inc.

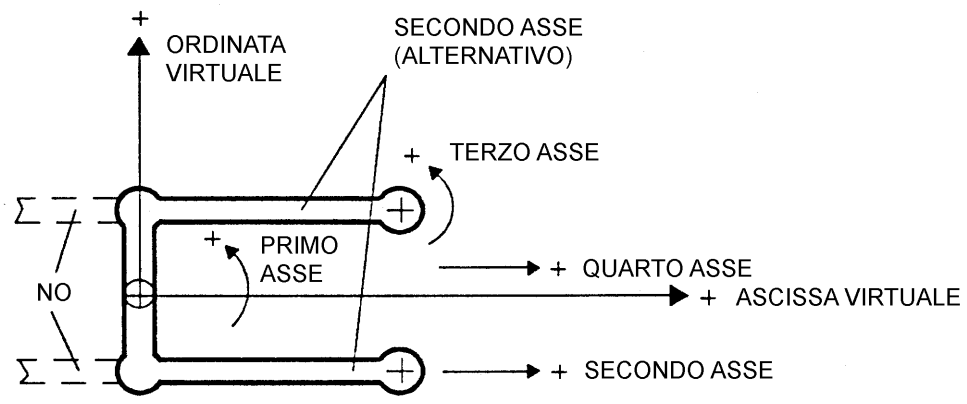
3 - L'offset assoluto del primo asse reale deve essere dato (se è presente il 4 asse traslante) in modo che l'asse virtuale ascissa sia parallelo al quarto asse CILINDRICO.

4 - ll secondo asse (cannotto) può o meno avere offset; tale offset implica chiaramente una diversa lunghezza del secondo asse.

5 - Le direzioni di movimento (positive, negative) degli assi del sistema devono essere tutte concordi: per gli assi rotanti (primo e terzo asse) direzione positiva significa andare da ascissa positiva a ordinata positiva; gli assi lineari (secondo e quarto) devono avere la stessa direzione delle ascisse.

6 - ll secondo asse (cannotto) non può trovarsi in ascissa negativa in caso contrario ruotare il primo asse in modo che si trovi in ascissa positiva.

La figura successiva fornisce un esempio di come devono essere messi gli assi su di un sistema cilindrico



7 - In base alle considerazioni viste nei punti 1-6 precedenti, si avrà che dopo un movimento a zero di tutti e 4 gli assi reali CILINDRICI, l'asse virtuale ascissa si

troverà ad una coordinata pari alla lunghezza del secondo braccio e l'asse virtuale ordinata ad una coordinata pari alla lunghezza del primo braccio. (Per quest'ultimo asse la coordinata sarà positiva o negativa)

b) Con un sistema CILINDRICO, non è possibile richiedere tramite una stessa istruzione di movimento, assi virtuali unitamente ai due assi reali (uno dei due o entrambi) costituenti il sistema cilindrico e precisamente: 1° asse del sistema (Rotante generale), 2° asse del sistema (cannotto) o 4° asse del sistema (traslante) nel caso di assenza dell'asse cannotto

E' invece possibile richiedere assi virtuali unitamente al 3° e/o 4° asse (se è presente il 2° asse)
Riepilogo degli assi reali non abbinabili ai virtuali in una stessa istruzione di movimento:

- Sistema cilindrico composto dal 1° e 2° asse (rotante generale + cannotto) Non sono ammessi virtuali +1° e/o 2° asse reale

- Sistema cilindrico composto dal 1° e 4° asse (rotante generale + traslante) Non sono ammessi virtuali + 1° e/o 4° asse reale

- Sistema cilindrico completo di 1°, 2°, e 4° asse

Non sono ammessi virtuali + 1° e/o 2° asse (da solo o con il 4°).

APPENDICE B

B. Controllo asse con volantino

Il posizionamento di un asse in manuale può essere abbinato ad un dispositivo chiamato volantino che viene visto dal sistema come un asse di sola lettura .

Il valore di posizione letto dal volantino modifica la posizione dell'asse abbinato, è così possibile dare degli incrementi micrometrici all' asse stesso.

Il volantino è visto come un asse di sola lettura, che tramite opportuni comando viene agganciato ad un asse che ne rimarrà controllato.

Nel linguaggio di automazione AXESBRAIN, l'istruzione "-HMS" permette l'inserimento e il disinserimento di un asse volantino "master" ad un asse di posizionamento "slave".

Per gestire l'abbinamento asse volantino con asse di posizionamento in ambiente "DCOM", deve essere utilizzato il servizio "WriteAxesRegister" per entrambi gli assi.

APPENDICE C

C. Assi a portale (Gantry)

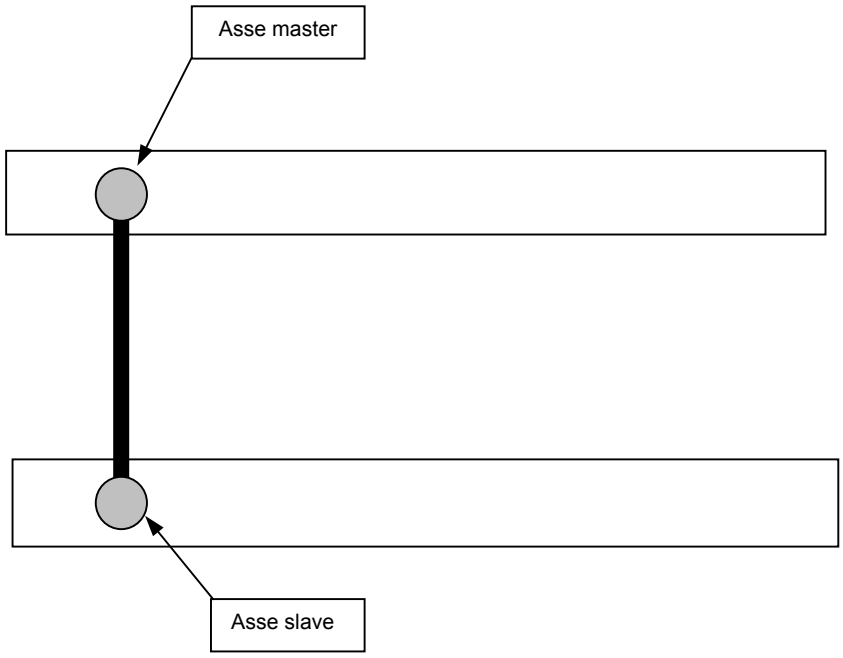
L'asse a portale(Gantry)è una struttura meccanicamente rigida (normalmente è una struttura a portale) e corrisponde pertanto ad un asse unico, ma è trattata dal controllo come se fosse costituita da una coppia di assi (asse master e asse slave, ciascuno con i propri sistemi di conteggio ed il proprio azionamento.
Una delle funzioni del controllo è quella di mantenere la posizione dell'asse "slave" più prossima possibile a quella dell'asse "master".

I movimenti richiesti sugli assi Master-Slave, vengono eseguiti nel modo seguente:

La richiesta di movimento può essere effettuata tramite tutte le istruzioni di movimento.

Nel caso di movimento relativo al solo asse Master, l'asse Slave lo segue o rimane fermo rispettivamente in caso di associazione o disassociazione.

Durante il movimento, l'asse Slave segue il suo Master in tempo reale



APPENDICE D

D. Camme elettroniche

La camma elettronica permette di abbinare la posizione di un gruppo di assi ad un asse "master" ed una tabella di posizioni multiple.

E' così possibile simulare elettronicamente il comportamento delle camme, sostituirne il funzionamento meccanico con un sistema analogo formato da un gruppo di assi asservito ad un asse "master" che può essere di sola lettura.

Le leggi del moto dei cedenti è definita come una tabella di vettori, che ne defisse le posizioni rispetto al moto della camma.

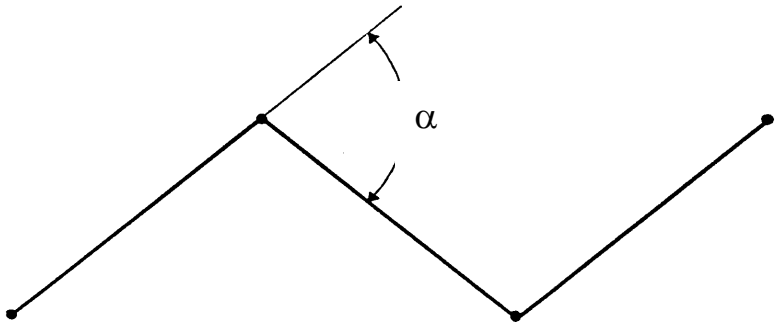
Nel linguaggio di automazione AXESBRAIN, l'istruzione "-HEC" permette la gestione delle camme elettroniche.

Per gestire le camme elettriche in ambiente "DCOM", deve essere utilizzato il servizio "WriteAxesRegister" per tutti gli assi .

APPENDICE E

E. Movimenti in continuo

I vari tipi di continuo, si riferiscono al profilo teorico seguente:



Parametri del continuo

a = switch (da 0 a 3) che determina il tipo di continuo nel modo seguente:
0 significa continuo normale
1 significa spline di tipo 1
2 significa spline di tipo 2
3 significa spline di tipo 3

b = coefficiente (da 0 a n) di riduzione velocità ad ogni cambio di direzione fra i vari punti del profilo.

+ = partenza immediata dei movimenti.

- = partenza dei movimenti condizionata al verificarsi delle seguenti condizioni:
- istruzione HCL trovata nel Part Program;
- nuova istruzione STC con parametro **b** positivo trovata nel Part Program;
- saturazione del buffer contenente gli enti movimenti relativo al continuo;

c = valore (da 0, a 1) che determina il tipo di congiunzione (schiacciamento o meno dell'arco) fra i punti del profilo nel modo seguente:
0 significa retta (arco tutto schiacciato) fra i vari punti;
1 significa arco normale fra i vari punti;
da >0 a <1 significa arco più o meno schiacciato; tanto più piccolo è il valore, tanto più schiacciato sarà l'arco;

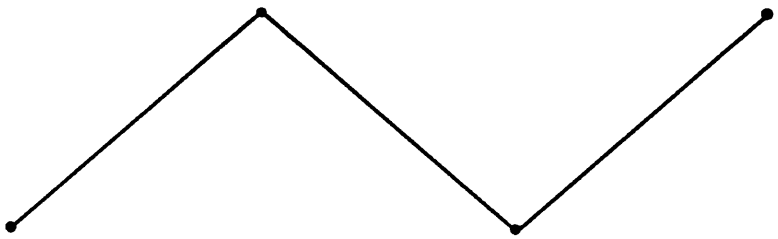
a) Nel caso di continuo normale (parametro **a**= 0), vi è sempre overshoot dopo ogni punto del profilo. La velocità di overshoot dopo ogni movimento è direttamente proporzionale alla accelerazione ed al coefficiente è maggiore (con incremento) o minore (con decremento) in modo non proporzionale rispetto all'angolo α (cambio di direzione fra 2 movimenti successivi). Poichè la massima velocità di overshoot non può comunque superare la massima velocità dell'asse, un valore grande di coefficiente è significativo solamente con un valore piccolo di accelerazione; viceversa con una grande accelerazione ha significato un piccolo valore di coefficiente (se questo valore è grande, viene comunque forzato al massimo ammissibile).

Esempi di profilo con continuo normale

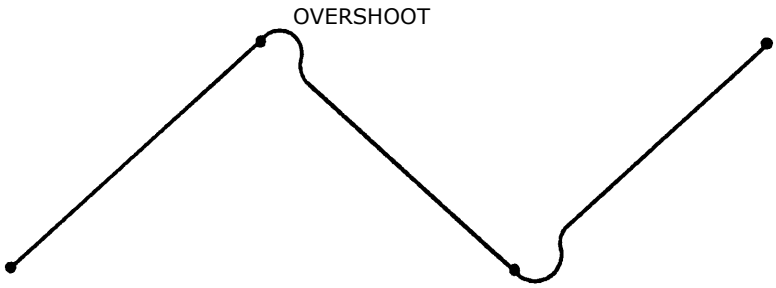
Profilo pratico con coefficiente del parametro **b** = 0

Esempi di profilo con continuo normale

Profilo pratico con coefficiente di riduzione velocità ad ogni cambio di direzione fra i vari punti del profilo. = 0



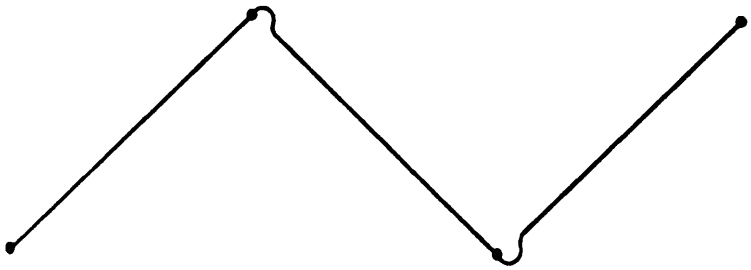
Profilo pratico con coefficiente diverso da 0



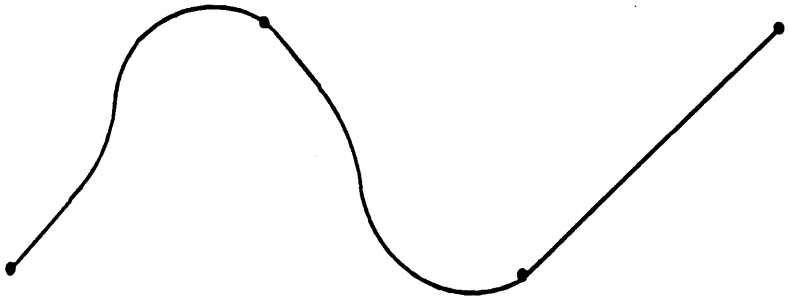
b) Nel caso di spline di tipo 1 ,l'overshoot dopo ogni punto del profilo è nettamente inferiore a quello relativo al continuo normale; praticamente è nullo se si assegna valore 1 al parametro **c**; varia da 0 ad un valore crescente (comunque sempre basso) più il valore del parametro di congiunzione (schiacciamento o meno dell'arco) si avvicina allo 0 (retta con arco tutto schiacciato ,fra i vari punti).
Anche per questo tipo di continuo, si ha una variazione di velocità lungo il profilo, per essa è molto meno evidente rispetto al continuo normale; naturalmente la velocità sul profilo sarà tanto più uniforme tanto minore è l'angolo a, tanto più il valore del parametro o si avvicina a 1 e tanto maggiore (anche se poco influente) è il valore del coefficiente di riduzione velocità

Esempi di profilo con spline di tipo 1

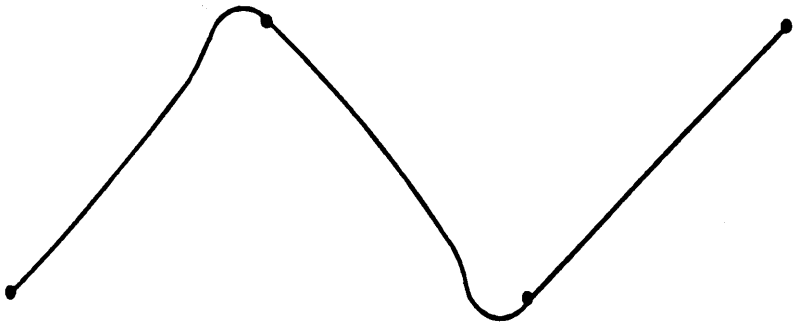
OVERSHOOT PICCOLO



Profilo ottenuto con parametro b=0 e coefficiente di riduzione velocità diverso da 0; se il coefficiente di riduzione velocità è 0, non vi sono naturalmente gli overshoot.



Profilo ottenuto con parametro **c**= i (coefficiente di riduzione velocità diverso da 0)

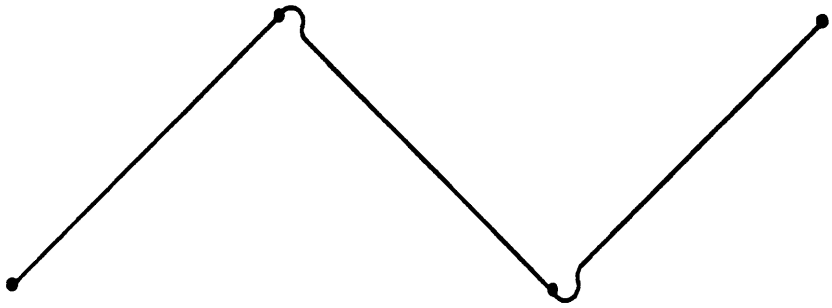


Profilo ottenuto con parametro **c** compreso fra 0 e 1 (coefficiente di riduzione velocità diverso da 0)

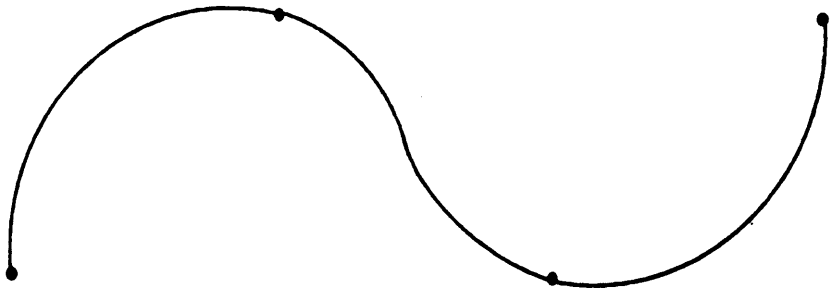
o) Nel caso di spline di tipo 2 (parametro **a** = 2) e di tipo 3 (parametro **a** = 3) si ha un comportamento simile alla spline di tipo 1 ma con la differenza che per la spline di tipo 2 il precalcolo sul profilo viene tatto prendendo in considerazione 3 punti del profilo mentre per la spline di tipo 3 i punti presi in considerazione sono 5.

Esempi di profilo con spline di tipo 2 e 3

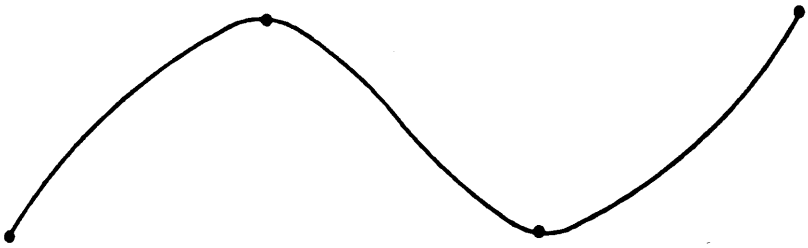
OVERSHOOT PICCOLO



Profilo ottenuto con parametro **c** = 0 e coefficiente STC diverso da 0, se il coefficiente STC è 0, non vi sono naturalmente gli overshoot.



Profilo ottenuto con parametro **c** =1 (coefficiente STC diverso da 0)

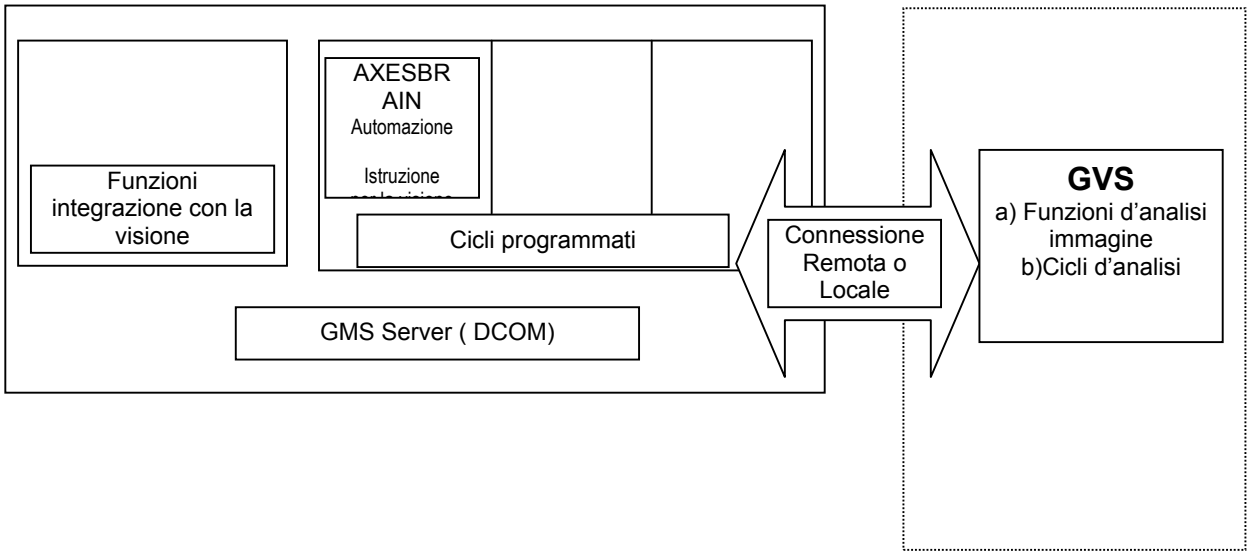


Profilo ottenuto con parametro **c** compreso fra 0 e 1 (coefficiente STC diverso da 0).

APPENDICE F

F. Integrazione con la visione

L'integrazione con l'ambiente di analisi d'immagine GVS può essere operato in due modalità o attraverso delle specifiche richieste del GMSServer che effettuano delle chiamate alle funzioni base o richieste di eseguire cicli di analisi, una seconda strada è utilizzare le istruzioni d'interfacciamento visione del linguaggio AXESBRAIN.



Schema integrazione analisi d'immagine

Come si può notare dallo schema la macchina predisposta per l'analisi d'immagine può essere remotata su un altro PC, naturalmente collegata in rete locale o rete internet.

GVS

Il pacchetto GVS lavora tramite richieste di funzioni d'analisi dell'immagine che effettuano operazioni di :

- Filtro e manipolazione dell'immagina

- Analisi luce e estrazioni automatiche di soglie
- Blobs su rettangoli d'interesse
- Estrazioni di contorni su intersezione di figure su rettangoli d'interesse
- Estrazioni di bordi
- Misurazione di contorni
- Estrazioni di enti sui contorni
- Misurazione sugli enti

Inoltre è previsto un linguaggio GVSL in grado di programmare cicli d'elaborazione di immagini , in modo di ottenere dei risultati mirati alle situazioni reali.

APPENDICE G

G. Definizione di Globali speciali e funzianilità generali

Nel file di caratterizzazione/configurazione del sistema "sistema.txt", vedi manuale di caratterizzazione, è stato previste **se dichiarate** delle Globali speciali che si integrano con l'ambiente di programmazione e di alcune funzionalità generali.
Le dichiarazioni si collocano nel file "sistema.txt" sotto la sessione "**[ParametriGenerali]**" e sono:

a- GlobaleNumeroAllarmi
b- GlobaleBloccoMovimentazione
c- NumeroLocali
d- NumeriDecimaliDRT
e- NumeriCifreDRT
f- NumeriDecimaliFILE
g- NumeriCifreFILE
h- NumeroIstruzioniMax
i- PartProgramG89
l- FlagFastWaitIO
m- TempoLatenzaErrori
n- WaitSEC
o- TempoScansione
p- SeGlobaleErroreNoDISM1
q- FlagHoldG8183

GlobaleNumeroAllarmi

Il numero indicato se diverso da zero indica la Globale nella quale il sistema carica il numero di allarmi presenti.
Default = 0

GlobaleBloccoMovimentazione

Il numero indicato se diverso da zero indica la Globale nella quale il sistema verifica l'inibizione dei movimenti manuali (non programmati), nel caso caso che il valore sia diverso da zero 0.
Default = 0

NumeroLocali

Il valore indica il numero di variabili Locali che vengno definite per ogni part program ed inizializzate a 0.
L'istruzione "DIM" ridimensiona questo valore nella sola istanza del part program
Default = 32

NumeriDecimaliDRT

Il valore indica il numero di decimali che vengono rappresentati nella visualizzazione dei campi dell'istruzione "DRT"
Default = 3

NumeriCifreDRT

Il valore indica il numero di cifre che vengono rappresentate nella visualizzazione dei campi dell'istruzione "DRT"
Default = 7

NumeriDecimaliFILE

Il valore indica il numero di decimali che vengono rappresentati nella visualizzazione dei campi dell'istruzione "FWR"
Default = 3

NumeriCifreFILE

Il valore indica il numero di cifre che vengono rappresentate nella visualizzazione dei campi dell'istruzione "FWR"
Default = 7

NumeroIstruzioniMax

Per evitare che una sequenza di istruzioni logiche e di controllo condizionino il tempo dedicato dalla CPU, viene indicato il numero di istruzioni nel quale se il task non viene sospeso, il sistema forza una pausa di **TempoScansione** millisecondi per il part program in esame.
Default = 64

PartProgramG89

Nome del part program che viene eseguito con le modalità dei cicli fissi.
Default = G89.pp

FlagFastWaitIO

Valore che attiva la modalità fast per le istruzioni "WDI" attive.
Default = 0

TempoLatenzaErrori

Tempo in secondi che deve intercorrere tra due messaggi con lo stesso numero d'errore generati tramite "-SEC/SET_MESSAGE,#" , per attualizzarlo sul zona di video dedicata.
Default = 1

WaitSEC

Tempo si sospensione automatico che viene introdotto ad ogni istruzione di "SEC" e di "DIS/-1:"
Default = 0.03

TempoScansione

Il tempo di scansione per ogni part program se non viene sospeso da attività non logiche o matematiche
Default = 0.010

SeGlobaleErroreNoDISM1

Per evitare la sovrascrittura nella parte di segnalazione con l'istruzione –DIS/-1: …. se sono presenti segnalazioni, verificando la **GlobaleNumeroAllarmi**, il sistema non invia il messaggio.
Default = 1

FlagHoldG8183

Flag che definisce**, se** a 1 che i cicli fissi di G81,G82 e G83 possono andare in HOLD
Default = 0

APPENDICE H

H. Elenco Errori

Errore	1	Funzionalità ancora attiva
Errore	2	Nome asse/mandrino/IO non corretto
Errore	3	OVERFLOW Tabelal nomi assi
Errore	4	OVERFLOW Tabella assi
Errore	5	Tipo asse incongruente
Errore	6	MANCA ASSE FISICO PER SISTEMA VIRTUALE
Errore	7	ASSE HELP NON USABILE (CYL_TRASLANTE)
Errore	8	OVERFLOW Tabella JOB di movimentazione
Errore	9	JOB di movimentazione NON DEFINITO
Errore	10	OVERFLOW AREA movimentazione
Errore	11	PARAMETRI Start continuo ERRATI
Errore	16	PARAMETRI Movimentazione ERRATI
Errore	17	TROPPE TPE CONTEMPORANEE
Errore	18	ASSE GIA' USATO IN ALTRI JOBS
Errore	19	ASSE FUORI CAMPO DI LAVORO
Errore	20	RAGGIO FINALE <> RAGGIO INIZIALE
Errore	21	JOB NON DEFINIBILE
Errore	22	TROPPI ASSI (MAX 8)
Errore	23	NON CI SONO PIU' TIMER
Errore	24	ASSI IN MOVIMENTO
Errore	25	STATO INCORRETTO
Errore	26	ERRORE DI SISTEMA NELL'USO DEL JOB
Errore	27	ASSE FRENATO
Errore	28	CLOSE O OPEN POSITION LOOP INUTILE
Errore	29	MANCA AREA SLAVE
Errore	30	CAMPIONATURA NON ATTIVA
Errore	31	TROPPI PARAMETRI
Errore	32	MANCA ASSERVIMENTO ATTIVO
Errore	33	TROPPE PIASTRE I/O CONFIGURATE
Errore	34	PIASTRA I/O SBAGLIATA
Errore	35	BIT NON SU PIASTRA
Errore	36	OVERFLOW DEL BIT LOGICO
Errore	37	BIT I/O SBAGLIATO
Errore	38	TROPPE PIASTRE ANALOGICHE CONFIGURATE
Errore	39	PIASTRA ANALOGICA SBAGLIATA
Errore	40	CANALE ANALOGICO NON SU PIASTRA

Errore	41	OVERFLOW DEL CANALE ANALOGICO (LOGICO)
Errore	42	CANALE ANALOGICO SBAGLIATO
Errore	43	SCELTA SCONOSCIUTA
Errore	44	TEMPO SCADUTO (Time OUT)
Errore	45	INTERPOLAZIONE SPLINE ERRATA
Errore	47	AZZERAMENTO ASSE NON FATTO (MANCA HOMING)
Errore	48	OVERFLOW TABLE SPINDLE
Errore	49	MANDRINO IN ATTIVITA' SAMPLE
Errore	50	MANDRINO IN ROTAZIONE
Errore	51	ASSERVIMENTO DISABILITATO
Errore	52	QUOTA CORREZIONE ERRATA (UGUALE)
Errore	53	OVERFLOW AREA CORREZIONE
Errore	54	BUFFER CIRCOLARE NON ATTIVO
Errore	55	OVERFLOW COUNTER NELL'ASSE STEPPER
Errore	56	COUNTER NON DEFINITO NELL'ASSE STEPPER
Errore	58	MOVIMENTO IN DEADLOCK
Errore	59	TASK IN ABORT (TKM)
Errore	60	OVERFLOW TKM
Errore	61	ASSE SU MICRO DI OLTRE CORSA POSITIVO
Errore	62	ASSE SU MICRO DI OLTRE CORSA NEGATIVO
Errore	63	FASE NON FINITA
Errore	64	TASK CON TKM ANCORA ATTIVA
Errore	65	GRM SU GRUPPO ANCORA ATTIVO
Errore	66	TABELLA ELECTRONIC CAM NON DEFINITA
Errore	67	OVERFLOW TABLE ELECTRONIC CAM
Errore	68	DIREZIONE SBAGLIATA IN ELECTRONIC CAM
Errore	69	HEC BUFFER PIENO
Errore	70	HEC VELOCITA' MASTER ECCESSIVA
Errore	71	TABELLA HEC NON CORRETTA
Errore	72	FUORI SERVIZIO
Errore	73	SPD BLOCCATO DA GRM/1:..
Errore	74	TABELLA HEC IN SAMPLE (NON CANCELLABILE)
Errore	75	Movimento con ABC (DEFAULT LOCK)

Errore	1001	HANDLE INCORRETTO
Errore	1002	DRIVER OCCUPATO
Errore	1003	INIZIALIZZAZIONE ERRATO
Errore	1004	ANOMALIA IN FASE DI AZZERAMENTO
Errore	1006	TROPPI HANDLER DI MOVIMENTAZIONE ATTIVI

Errore Driver Fault	0x5	Configuration error - Decision Table
	0x7	Calibration error
	0x10E	Fiber not connected to digital driver
	0x111	Digital driver assh't hardware enable
	0x113	Fiber rx Frame error
	0x114	Fiber rx Parity error
	0x116	Zero command refused for movement
	0x117	Lost zero axis