Introducción a la programación

Melfa Basic IV

Programación estructurada

- En este lenguaje la programación se estructura como un conjunto de instrucciones cuyo flujo de proceso se realiza en un lenguaje BASIC estándar.
- El aspecto de un programa es un conjunto de instrucciones propias del sistema de Robot entre sentencias ya conocidas de BASIC.
- Se obtiene así una forma intuitiva de programación , sencilla incluso para aquellos usuarios con pocos conocimientos de BASIC.

Ejemplo de programa

```
...
10 DEF INTE VEL
12 VEL=50
15 OVRD VEL

20 FOR T= 0 TO 6
30 MVS P1
40 MOV P2
50 MVS P4
60 NEXT T

70 GOSUB 1000
75 HCLOSE 1
80 MOV P7
85 OVRD  30
90 MVS P9
95 RETURN
100 GOTO 20

1000 MOV P3
1005 DLY 1
1200 ...
```

Sentencias propias de BASIC para el flujo y condiciones de programa

Sentencias propias de MELFA para el movimiento del Robot

Carácteres con significado especial

Apóstrofe (´)

Las líneas de comentarios están indicadas con apóstrofes, y serán transferidas también a la drive unit.

Ejemplo:

```
100 ´ posición de inicio
```

Asterisco (*)

El asterisco define marcas de salto (etiquetas). No serán tranferidas a la drive unit

Ejemplo:

```
110 *TABLA1
```

Carácteres con significado especial(2)

Coma (,)

La coma sirve de separador cuando se especifican muchos parámetros consecutivos.

Ejemplo:

100 P50 = (450,100,300, ...)

Punto (.)

Para datos múltiples ,como los datos posicionales, el punto sirve como separador de cada componente singular.

Ejemplo:

110 M10 = P10.X

Carácteres con significado especial (3)

Espacio ( ) ␣

Debe guardarse entre instrucciones y datos individuales, y tras los números de línea

Ejemplo:

100 ␣MOV␣P10

- **Cada línea debe contener como máximo un comando**

Declaración de variables

**Los nombres de variables del tipo de posición, articulación (joint), aritmética, y
cadena de caracteres , empiezan con un carácter particular.**

La norma es:

P = *Positional* (variable de posición)
J = *Joint* (articulaciones)
M = Aritméticas
C = *Character string* (cadena de caracteres)

Constantes numéricas

Ejemplos:

decimal : 234, 7471, -435, +546, -5454

hexadecimal : &H03FA, &H1AE5, &HA5

binario : &B0101, &B110110101, &B10101111

Constantes alfanuméricas

Ejemplo:

"MELFA BASIC es altamente eficiente"
“Siguiente posición"
“Esperando entrada 5"

Constantes angulares

Ejemplo:

90DEG

120DEG

El seno de un ángulo de 100º se representa como:

SIN(100DEG)

Declaración de variables: ejemplos

Position P	Joint J	Arithmetic M	Character string C
P1	J100	M10	C30$
P124	J100.W	M99	C$[M5+4] (!)
P100.X	J10.T	M[M6+3] (!)	
P110.Z			
P[M5+3] (!)			
P[M10].Z (!)			

(!) Sólo en Melfa Basic III

Expresiones con variables de tipo Posicional

Ejemplos:

P14 = P100
P20 = P_CURR
P30 = P[M4*2+5] (!)
P5.Z = 10*M5
P[M10] = P1 + P20
P15.Z = P15.Z+30

-**Variables de Posición**: Datos de coordenadas de espacio ortogonales, X, Y Z,(normalmente en mm) y orientación A, B. (en DEG) . Todas las variables de este tipo empiezan con **P**.

(!) Sólo en Melfa Basic III

- **Expresiones con variables del tipo Joint**

Ejemplos:

J10 = J_CURR
J10.W = J10.W+RAD(M5)

-**Variables de "articulación"**: Datos que hacen referencia a posición de los ejes. Todas las variables de este tipo empiezan con **J**.

- **Expresiones con variables de Cadena de carácteres**

Ejemplos:

C30$ = "Nº de paso de secuencia."
C$[M100] = "Siguiente posición"(!)
C$[M10*2] = "Número de posición:"(!)

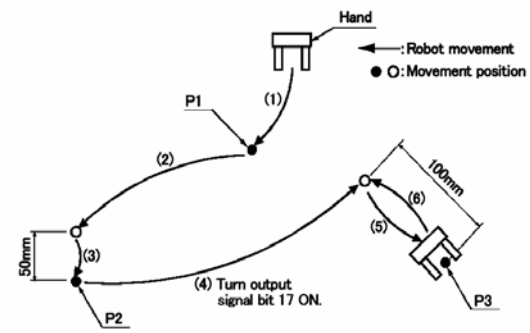
(!) Sólo en Melfa Basic III

Funciones de movimiento: MOV

MOV : Movimiento interpolación de ejes

Descripción:
Esta instrucción mueve a un punto determinado mediante interpolación de ejes. La trayectoria de un punto a otro no es lineal, es decir, no describe una línea recta en el espacio, sino que la CPU procesa y mueve los ejes a su conveniencia , por su camino más sencillo. Por lo tanto la trayectoria no es 100% predecible por el usuario

Funciones de movimiento: MOV



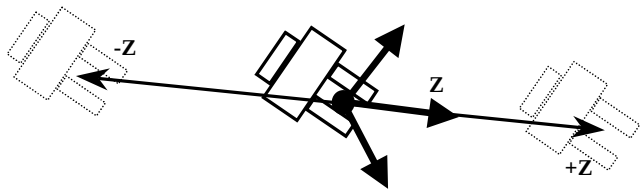

```
10 MOV P1                          ;mueve hacia P1
20 MOV P2,-50                      ;mueve respecto P2, 50mm atrás de la
                                   ;posición de la mano (desp. relativo)
30 MOV P2                          ;mueve hacia P2
40 MOV P3,-100 WTH M_OUT(17)=1     ;mueve respecto P3,100mm atrás, mientras activa
                                   ;salida bit nº 17
50 MOV P3                          ;mueve hacia P3
60 MOV P3,-100                     ;mueve respecto P3, 100mm atrás(desp.relativo)
70 END                             ;fin de programa
```

Funciones de movimiento: MOV

Comentarios:

-El movimiento en este tipo de instrucciones no es lineal, por lo tanto, no es 100% predecible. Utilizar esta instrucción con cautela, para evitar **colisiones** del brazo con alguna parte del entorno del robot.

-Cuando se usa el desplazamiento relativo desde un punto, (por ejemplo `MOV P3,-100`) el sentido de avance viene determinado por el **signo** de éste, en su coordenada **Z**:

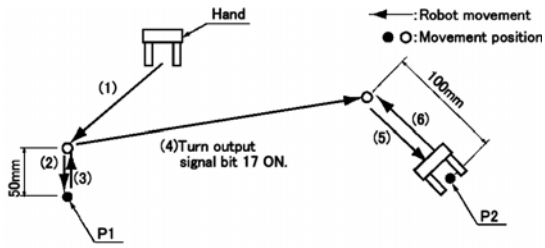


Funciones de movimiento: MVS

MVS : Movimiento en interpolación lineal

Descripción:
Esta instrucción mueve a un punto determinado mediante interpolación lineal. La trayectoria de un punto a otro es lineal, es decir, describe una línea recta en el espacio.

Funciones de movimiento: MVS




```
10 MVS P1,-50              ;mueve respecto P2, 50mm atrás, en línea recta
                           ;(desp. relativo)
20 MVS P1                  ;mueve hacia P1 en línea recta
30 MVS,-50                 ;mueve 50mm atrás desde la posición actual
                           ; en línea recta (desp. relativo)
40 MVS P2,-100 WTH M_OUT(17)=1     ;mueve respecto P2,en línearecta,
                                   ;100mm atrás, mientras activa
                                   ;salida bit nº 17
50 MVS P2                  ;mueve hacia P2, en línea recta
60 MVS,-50                 ;mueve respecto P2, en línea recta 50mm
                           ;atrás (desp. relativo)
70 END                     ;fin de programa
```


Funciones de movimiento: MVS

Comentarios:

-Cuando se usa el desplazamiento relativo desde un punto, (por ejemplo `MVS P3,-100`) el sentido de avance viene determinado por el **signo** de éste, en su coordenada **Z**, tal como se hace con **MOV**.

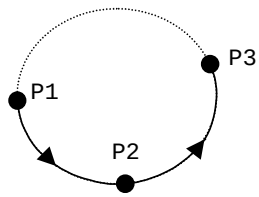
-Al usar el desplazamiento relativo, la posición destino es ficticia (no consta en tabla de coordenadas, por lo tanto no es un punto registrado). Usarlo con cuidado para evitar **colisiones** del brazo con alguna parte del entorno del robot.

Movimientos en interpolación circular: MVR

MVR : Designado un punto de comienzo, un punto de tránsito y un punto final, se realiza un movimiento a través de ellos (describe un arco en el espacio), mediante interpolación circular de ejes.

Ejemplo:

```
MVR P1,P2,P3
```

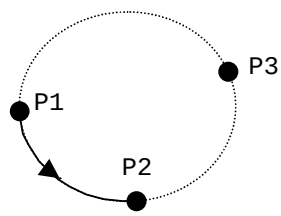


Movimientos en interpolación circular: MVR2

MVR2 : Designado un punto de comienzo, un punto de final y un punto de referencia, se realiza un movimiento del punto inicial al punto final sin pasar por el punto de referencia. La trayectoria seguida es la que correspondería a un arco que incluye el punto de referencia, pero sólo se traza el segmento correspondiente al tramo del punto inicial al final.

Ejemplo:

```
MVR2 P1,P2,P3
```

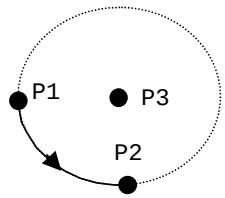


Movimientos en interpolación circular: MVR3

MVR3 : Designado un punto de comienzo, un punto de centro y un punto de final, se describe un arco desde el punto de inicio hasta el de final, cuya trayectoria es trazada respecto al punto de centro. El ángulo trazado debe ser entre 0 y 180º :

Ejemplo:

```
MVR3 P1,P2,P3
```

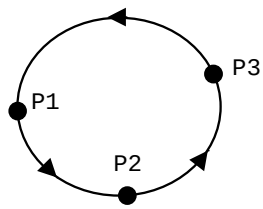


Movimientos en interpolación circular: MVC

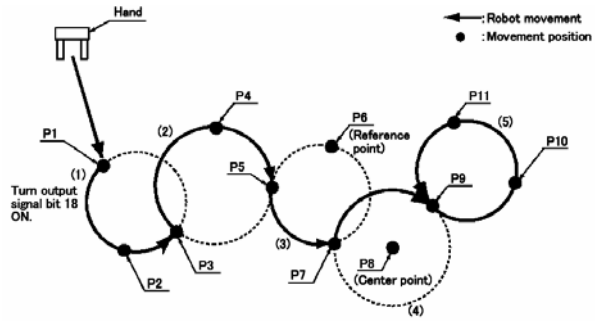
MVC : Designado un punto de comienzo igual al final, un punto de tránsito 1 y un punto de tránsito 2, describe un arco desde el punto de comienzo pasando por el punto de tránsito 1, luego por el 2 y finaliza en le punto final, que es el mismo de comienzo. Describe por lo tanto un círculo o una elipse completa.

Ejemplo:

```
MVC P1,P2,P3
```



Movimientos en interpolación circular




```
10 MVR P1,P2,P3 WHT M_OUT(18)      ;Mueve P1→P2→P3 como un arco,
                                   ; mientras activa la salida bit 18
20 MVR P3,P4,P5                    ;Mueve P3→P4→P5 como un arco
30 MVR2 P5,P7,P6                   ;Arco de P5→P7 con trayectoria de P6
40 MVR3 P7,P9,P8                   ;Describe P7→P9 con centro en P8
50 MVC P9,P10,P11                  ;Arco cerrado P9→P10→P11→P9
60 END
```

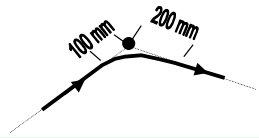

Movimientos sin interrupciones: CNT

CNT : Movimiento sin interrupciones

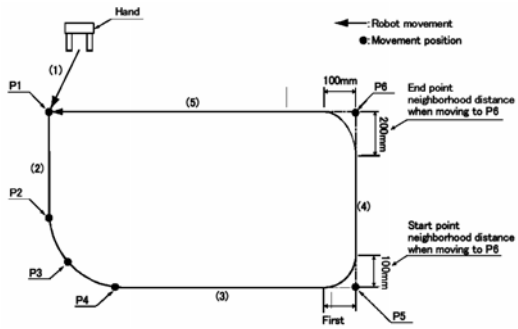
Descripción:

Esta instrucción permite que se realiza un movimiento entre puntos múltiples - definidos por MOV,MVS,MVR, etc, sin interrupciones, es decir, sin aceleraciones ni deceleraciones. En el momento que se declara esta instrucción, todas las intrucciones de movimiento a partir de ella se hacen de esta forma.

```
CNT 1                  ;Designa la activación    de la función CNT
CNT 0                  ;Designa la desactivación de la función CNT
CNT 1,100,200          ;Designa la activación de la función, y
                       ;define que el punto De comienzo a 100mm y
                       ;el de final a 200mm del punto destino
```



Movimientos sin interrupciones




```
10 MOV P1
20 CNT 1          ; a partir de las siguientes líneas se habilita CNT
30 MVR P2,P3,P4
40 MVS P5                          ;
50 CNT 1,100,200                   ; El punto de comienzo más cercano a 100mm,
                                   ;el de final más cercano a 200mm
60 MVS P6
70 MVS P1
80 CNT 0
90 END
```


Definición de velocidades y acc. decc.

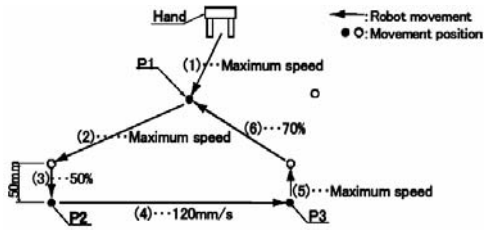
ACCEL : Designa aceleración y deceleración en % respecto a la máxima permitida

OVRD : Designa la velocidad de trabajo del robot en %.

JOVRD : Designa la velocidad de interpolación de ejes en % respecto a la máxima permitida

SPD : Designa la velocidad en mm/s de la interpolación circular y lineal, velocidad de la parte móvil (punto de trabajo, mano, etc)

Definición de velocidades y acc. decc.

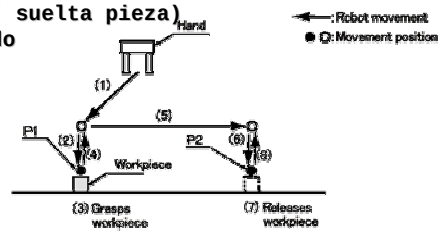



```
10 OVRD 100              ;fija la velocidad de trabajo al 100%
20 MOV P1
30 MOV P2,-50
40 OVRD 50               ; fija la velocidad de trabajo al 50%
50 MVS P2
60 SPD 120               ; movimientos lineales a 120mm/s
70 OVRD 100              ;velocidad general al 100%
80 ACCEL 70,70           ;aceler. y deceler. a 70%
90 MVS P3
100 SPD M_NSPD           ;fija la velocidad lineal a su valor nominal
110 JOVRD 70             ;velocidad de interpolación ejes a 70%
120 ACCEL                ;fija la aceleración y deceleración al 100%
130 MVS, -50
140 MOV P1
150 END
```


Control de utensilios (Pinza)

HOPEN :Abre la pinza designada
HCLOSE :Cierra la pinza designada

```
10 MOV P1,-50
20 OVRD 50
30 MVS P1
40 HCLOSE 1     ; cierra pinza nº 1 (atrapa pieza)
50 DLY 0.5      ; ejecuta un retardo de medio segundo
60 OVRD 100
70 MVS ,-50
80 MOV P2,-50
90 OVRD 50
100 MVS P2
110 HOPEN 1     ; abre pinza nº 1 ( suelta pieza)
120 DLY 0.5     ; realiza un retardo
130 OVRD 100
140 MVS,-50
150 END
```



Función de Paletizado

Descripción:
Esta función desarrolla un movimiento programado en filas y columnas, para realizar operaciones del tipo de manipulación en cajas compartimentadas, operaciones seriadas en una superficie, etc.

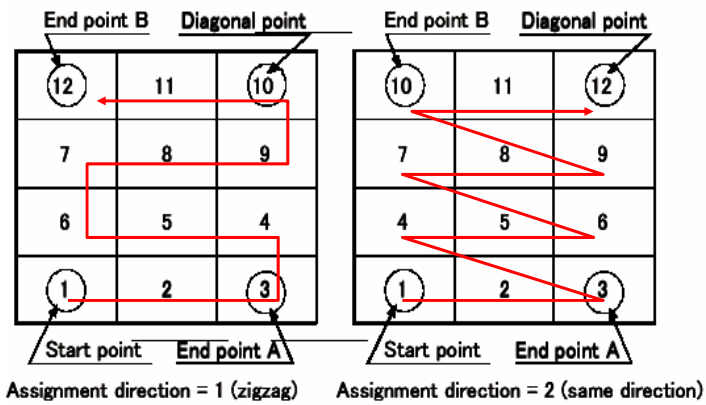
DEF PLT : Define el pallet a ser usado

PLT : calcula la posición actual de una casilla del pallet usado

Sintaxis:

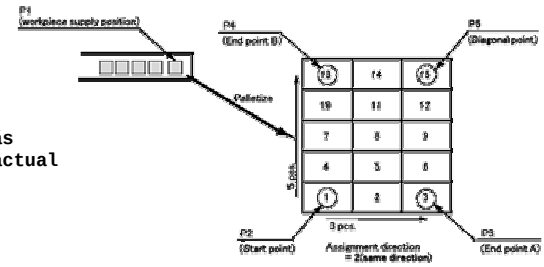
```
DEF PLT <Pallet No.>, <punto START>, <punto FINAL A>, <punto final B>,
[<punto Diagonal >], <cantidad A>, <cantidad  B>, <dirección de
avance>
```


Función de Paletizado



Función de Paletizado

```
10 DEF PLT 1,P2,P3,P4,P5,3,5,2     ;Define el pallet nº 1, punto
                                   ;START=P2, punto END A=P3, punto END
                                   ;B=5, dirección
20 M1=1
30 *BUCLE         ;Designa una etiqueta llamada BUCLE
40 MOV P1,-50
50 OVRD 50
60 MVS P1
70 HCLOSE 1
80 DLY 0.5
90 OVRD 100
100 MVS,-50       ;Mov.50mm atrás de posición actual
110 P10=PLT 1,M1  ;Opera en posición M1 del pallet 1
120 MOV P10,-50
130 OVRD 50
140 MVS P10
150 HOPEN 1
160 DLY 0.5
170 OVRD 100
180 MVS,-50       ;Mov.50mm atrás
                  ;de posición actual
190 M1=M1+1
200 IF M1<=15 THEN *BUCLE
210 END
```



Comandos de control de programa

Descripción general:
Estos comandos realizan las mismas funciones que el BASIC estándar, y sirve para transferir el control del programa a líneas determinadas de éste, condicionalmente a un caso particular o incondicionalmente.

GOTO : Salto incondicional a línea
Sintaxis:
`GOTO <línea o label>`

Ejemplos:

```
GOTO 200          ;salta a línea nº 200
GOTO *FINAL       ;salta a línea marcada como * FINAL
```

Comandos de control de programa

ON …GOTO :Salto condicional a línea designada por una variable entera. El programa seguirá el valor de orden de esta variable (0,1,2,3,4…)

Sintaxis:

```
ON <Variable entera> GOTO<destino><destino><destino>…
```

Ejemplos:

```
ON M1 GOTO 100,200,300      ;SI M1=1 salta a 100,
                            ;si M1=2 salta a
                            ;200…si no
                            ;corresponde, salta a
                            ;siguiente
```


Comandos de control de programa

IF… THEN… ELSE :Salto condicionado, si no se da la circunstancia se ejecuta el salto designado en **ELSE** .El comando **ELSE** es opcional.

Sintaxis:

```
IF <condición> THEN <línea> ELSE <línea o
label>
```

Ejemplos

```
IF M1=1 THEN 130                ;Salta a 130 si M1=1
IF M1=1 THEN 130 ELSE 150       ;Salta a 130 si M1=1,
                                ;si no salta a 150
```

Comandos de control de programa

SELECT...CASE : Salto condicional, según la condición se ejecuta lo designado en **CASE**

Sintaxis:

```
SELECT <variable>
      CASE <condición>
       <sentencias>
      CASE <condición>
       <sentencias>
      CASE <condición>
 <sentencias>

DEFAULT <sentencias>

End SELECT
```

DEFAULT corresponde al grupo de instrucciones que se ejecuta cuando ninguno de los casos anteriores se ha cumplido

Comandos de control de programa

Ejemplos

```
SELECT M1
CASE 10                    ; Si M1=10 ejecuta sólo
   :                       ;las líneas entre CASE 10
   :                       ;y CASE IS 11
CASE IS 11                 ; Si M1=11 ejecuta sólo
   :                       ;las líneas entre CASE IS
   :                       ;11 y CASE IS <5
 CASE IS < 5               ;
    :
 CASE 6 TO 8               ;ejecuta si está entre 6 y 8
    :
DEFAULT
    :
END SELECT
```

Comandos de control de programa

WAIT : Espera en esta línea hasta que la condición ha sido alcanzada

Sintaxis:

```
WAIT <condición>
```

Ejemplo:

```
WAIT M_IN(1)    ;Espera en esta línea hasta que la señal
                 de entrada 1 está activa
```


Repetición incondicional

FOR…NEXT : Repite las instrucciones comprendidas entre FOR y NEXT las veces que indique la sentencia FOR. El comando STEP es opcional

Sintaxis:

```
FOR <variable> = <const./variable> TO <const./variable>
STEP <paso>
(sentencias)
NEXT
```

Ejemplo:

```
10 FOR M1 = 1 TO 10     ;Las líneas entre 10 y 60 se
   :                            ;repetirán 10 veces
:
60 NEXT
70 FOR M2= 0 TO 50 STEP 10     ;Se incrementa M2 en pasos de
   :                                   ; 10 unidades
:
100 NEXT
```

Repetición condicional

WHILE…WEND: Repite las sentencias comprendidas entre WHILE y WEND hasta que se cumpla una condición determinada

Sintaxis:

```
WHILE <variable condición>
(sentencias)
WEND
```

Ejemplo:

```
10 WHILE (M1>=1)AND(M1<=10)   ;Las líneas entre 10 y 60
:                             ;se repetirán HASTA que M1
:                             ;esté entre 1 y 10
:
60 WEND
```


Llamada a subrutinas (incondicional)

GOSUB: llama a línea determinada en la instrucción y vuelve a ella tras encontrar RETURN en la subrutina.

Sintaxis:

```
GOSUB <LINEA/LABEL>
```

Toda subrutina debe acabar en RETURN para retornar el control a la línea siguiente tras GOSUB

Ejemplo:

```
10 GOSUB 1000                 ;LLAMADA a subrutina
20 <sentencias>               ;transfiere el control a
:                             ;la línea 1000 hasta que
:                             ;encuentra RETURN, tras
:                             ;ello vuelve a 20
1000 <sentencias>
:
:
1400 RETURN
```

Llamada a subrutinas (condicional)

ON …GOSUB :Salto condicional a línea designada por una variable entera. El programa seguirá el valor de orden de esta variable (0,1,2,3,4…)

Sintaxis:

```
ON <Variable entera> GOSUB <destino><destino><destino>…
```

Toda subrutina debe acabar en RETURN para retornar el control a la línea siguiente tras ON...GOSUB

Ejemplos:

```
ON M1 GOSUB 100,200,300          ;SI M1=1 salta a 100,
:                                ;si M1=2 salta a
:                                ;200…si no
100 <sentencias>                 ;corresponde,salta a
:
200 <sentencias>                 ;siguiente
:
300 <sentencias>
1000 RETURN
```


Llamada a sub-programas

CALLP :Transfiere el control del programa a otro programa almacenado en la Drive Unit; una vez lo ejecuta, vuelve a pasar el control al programa principal. Opcionalmente puede pasar parámetros .

FPRM :Recibe parámetros y variables desde el programa principal, hacia el programa llamado.

Sintaxis:

```
CALLP <Nombre programa> <parámetros y argumentos>
```

Ejemplo:

Programa principal

```
10 MOV P1
20 CALLP "2",P2,P7
30 END
```

Progr. No.2

```
10 FPRM P200,P700
100 MOV P200
110 MOV P700
120 END
```

Interrupciones

DEF ACT :Define las condiciones de la interrupción y la instrucción a realizar tras ella .
ACT :Establece la prioridad de esta interrupción sobre las otras

Sintaxis:

```
DEF ACT <Nºde Int.> <condición> <Proceso> <L>
```

Ejemplo:

```
10 DEF ACT 1,M_IN(17)=1 GOSUB 100          ; Si entrada 17 es ON, salta
                                           ;inmediatamente a línea 100

30 DEF ACT 3,M_TIMER(1)>10.5 GOSUB 300     ;Cuando pasan 10.5 segundos
                                           ;transfiere el control a línea
                                           ;subrutina 300.


100 M_TIMER(1)=0

110 ACT 3=1                                ;Establece prioridades
```


Interrupciones

Notas:

•Los saltos por interrupción que lleven a una instrucción GOSUB, deben retornar con RETURN, en este caso:

```
RETURN <nº de interrupción>
```

•Las prioridades se establecen con ACT<nºint>=<nivel> y van de 1(mayor) a 8(menor)

•Prioridad 0 significa interrupción desabilitada (ACT<nºint>=0)

•Cuando se pone una L al final significa que la interrupción se ejecutará al finalizar la instrucción en curso :

```
DEF ACT 1,M_IN(17)=1 GOSUB 100, L
```

Paro incondicional

HLT :Para el programa en aquel punto

Sintaxis:
HLT

Ejemplo:
```
10 IF M_IN(20)THEN HLT  ;detiene el programa si la
                        ;entrada 20 es ON
```

Retardos

DLY :establece un retardo
Sintaxis:
DLY <segundos (0.05 mínimo)>

Ejemplo:
```
10 DLY 0.8  ;detiene el programa durante 0.8 s.
```


Entradas y Salidas

Sintaxis:

Entradas:
<variable>=M_IN(<bit>)
<variable>=M_INB(<byte>)
<variable>=M_INW(<word>)
Salidas:
M_OUT(<bit>)=<1/0>
M_OUT(<byte>)=<byte>
M_OUT(<word>)=<word>

Ejemplos:
```
M1=M_INB(20)      ;BitsOUT 20 a 27 pasan a M1
WAIT M_IN(3)=1    ;Espera hasta que bitIN 3 es ON
M_OUT(1)=1 DLY 0.5 ;conmuta bitOUT 1 a ON durante 0.5S
```

Operaciones aritméticas / lógicas / funciones

MELFA Basic permite realizar operaciones con números y comparaciones. Ejemplos:

sustitución:
`P1=P2`
`P10.Z=100`

Aritmética:
`+,-,*,/,^,...`

Comparación:
`>,<,<>,=>,<=,AND,OR,NOT,XOR...`

Funciones:
`MAX,MIN,RAD,SQR,TAN,SIN,COS,TAN...`