FESTO

Melfa-Basic V. Handbook [EN]

Version: v1.3 (25.07.2013)

Daniel Bolla – FESTO DC-CC

dabo@de.festo.com

Table of Contents

Basic instructions

Delay

```
DLY     0.5     ' Wait 0.5s
```

Servo switch on/off

```
SERVO   ON      ' Switch on servo
SERVO   OFF     ' Switch off servo
```

Open the hand

```
DLY     0.1     ' [s]     Wait for finish movement
HOPEN   1       ' [#]     Open hand ( hand number )
DLY     1       ' [s]     Wait for finish opening
```

Close the hand

```
DLY     0.1     ' [s]     Wait for finish movement
HCLOSE  1       ' [#]     Close hand ( hand number )
DLY     1       ' [s]     Wait for finish Closing
```

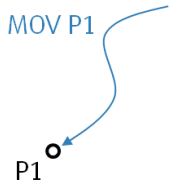
Setup instructions

```
TORQ    1, 40   ' [#, %]    Limit of torque ( axis number, torque value )
ACCEL   10, 20  ' [%, %]    Acceleration, Decelearation
OVRD    10      ' [%]       General speed override
JOVRD   30      ' [%]       Joint override: speed of joint interpolated movements as MOV
SPD     100     ' [mm/s]    Speed of interpolated movements as MVS, MVR
```

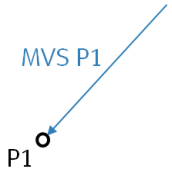
Absolute movements

Basic movements

```
JOVRD   30      ' [%]       Setup the speed
MOV     P1      ' [POS]     Move to position with joint interpolation. We do not know the path.
```

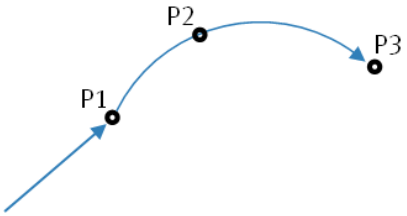



```
SPD     100     ' [mm/s]    Setup the speed
MVS     P1      ' [POS]     Move to position with linear interpolation. The path is a straight line.
```

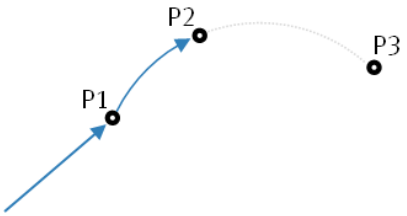


Circular interpolation

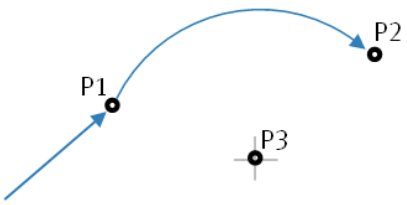
```
MVR     P1, P2, P3  ' [POS, POS, POS]   Move on a circle ( start point, transit point, end point ).
```




```
MVR2    P1, P2, P3  ' [POS, POS, POS]   Move on a circle ( start point, end point, reference point ).
```



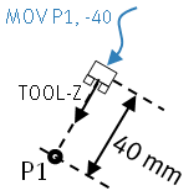

```
MVR3    P1, P2, P3  ' [POS, POS, POS]   Move on a circle ( start point, end point, center point ).
```



Relative movements

Offset in TOOL-Z

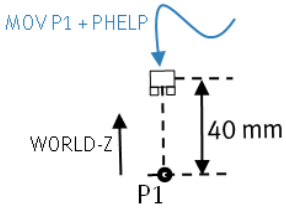
```
MOV     P1, -40          ' [POS, mm]   Move to shifted position. Offset in TOOL-Z = -40.
```




```
MOV     , -40            ' Move relatively to the actual position in TOOL-Z.
```

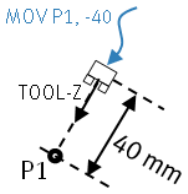
Addition of positions (relative offset in WORLD coordinates)

```
DEF POS  PHELP           ' [name start with „P”]      Define the PHELP position
PHELP =  (0, 0, 40, 0, 0,0)  ' [mm, mm, mm, deg, deg, deg]  Assign a position with coordinates
                                                      ( X, Y, Z, A, B, C )
MOV      P1 + PHELP      ' [POS] + [POS]              Move to calculated position
                                                      ( Offset in WORLD )
```



Multiplication of positions (relative offset in TOOL coordinates)

```
DEF POS  PHELP           ' [name start with „P”]      Define position
PHELP =  (0, 0, -40, 0, 0,0)  ' [mm, mm, mm, deg, deg, deg]  Assign a position with coordinates
                                                      ( X, Y, Z, A, B, C )
MOV      P1 * PHELP      ' [POS] * [POS]              Move to calculated position
                                                      ( Offset in TOOL )
```



Offset without position definition

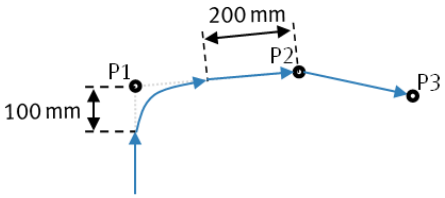
```
MOV      P1 + (10, 20, 30, 10, 20, 30)   ' [POS] * [mm, mm, mm, deg, deg, deg]   Offset in WORLD
MOV      P1 * (10, 20, 30, 10, 20, 30)   ' [POS] * [mm, mm, mm, deg, deg, deg]   Offset in TOOL
```

Continuous movement

```
CNT     0                ' 
MVS     P1
CNT     1, 100, 200      ' [1, mm, mm]
MVS     P2
CNT     0
MVS     P3
```

Switch off continuous movement

Switch on continuous movement
(1: switch on cnt, distance for starting the orbit change, distance for ending the orbit change)



I/O handling

Use Inputs

```
WAIT      M_IN(1) = 1        ' [condition]    Wait until input 1 equals 1. M_IN( number of input )
IF  M_IN(1) = 1 THEN         ' [condition]    IF condition is true, execute the instructions between THEN
    MOV   P1                 '                and ELSE or ENDIF ( if we do not use else )
ELSE                         '                Else execute the instructions between ELSE and ENDIF
    MOV   P2
ENDIF
```

Use outputs

```
M_OUT(1) = 1                 ' [#]            Switch on output 1. M_OUT( number of output )
M_OUT(1) = 0                 ' [#]            Switch off output 1. M_OUT( number of output )
```

Define I/Os

```
DEF IO    button = BIT, 3    ' Define an input for input 3 with „button name”.
DEF IO    lamp = BIT, 3      ' Define an input for output 3 with „lamp name”.
                             ' The definition is the same for input and output. If we read the value, the robot
                             ' will use the input. If we assign the value, then the robot will use the output.
WAIT      button = 1         ' Wait until button press
lamp = 1                     ' Switch on the lamp
```

Interrupts

Define interrupt

```
DEF ACT   1, M_IN(3) = 1     GOSUB *Int1     ' Define interrupt.( interrupt number, event, action )

DEF IO    button = BIT, 3                    ' Define input
DEF ACT   1, button = 1      GOSUB *Int1     ' Define interrupt with button push event.
                                             ' ( interrupt number, event, action )
```

Enable/disable interrupt

```
ACT       1 = 1              ' Enable interrupt ( interrupt number = 1 ).
ACT       1 = 0              ' Disable interrupt ( interrupt number = 0 ).
```

Interrupt routine

```
*Int1                        ' Label of interrupt routine.
    MOV   P1
RETURN 0                     ' Return from the interrupt routine.
```

Programming structures

Jumps

```
*Label1                 ' Define a label ( *label name )
GOTO    *Label1         ' Jump to label.
```

Subroutine

```
*Label1                 ' Define a label ( *label name )
   MOV   P1
RETURN                  ' Continue at the next program line after called the subroutine.
GOSUB   *Label1         ' Call subroutine
```

IF ... THEN ... ELSE

```
IF  M_IN(1) = 1 THEN    ' [condition]    IF condition is true, execute the instructions between THEN
   MOV   P1             '                and ELSE or ENDIF (if we do not use else)
ELSE                    '                Else execute the instructions between ELSE and ENDIF
   MOV   P2
ENDIF
```

FOR loop

```
FOR  M1= 1  TO  6  STEP  1   ' Repeat the same instructions, and increase M1 from 1 until reach 6.
   MOV   P1
   MOV   P2
NEXT M1                      ' End of the FOR loop.
```

WHILE loop

```
WHILE  M_IN(1) = 1      ' Repeat the same instructions while condition is TRUE.
   MOV   P1
   MOV   P2
WEND                    ' End of the WHILE loop
```

SELECT CASE structure

```
SELECT  M1              ' Run different instructions according to the value of M1
  CASE 1                ' If value is 1
    MOV   P1
    BREAK               ' End of the case
  CASE 2 TO 5           ' If value is between 2 and 5
    MOV   P2
    BREAK               ' End of case
  DEFAULT               ' If each case is inactive, run the instructions bellow
    MOV P9
    BREAK               ' End of default case
END SELECT              ' End of SELECT structure
```

Variables

Constants

```
1.0                     ' Decimal number
&H0F1A                  ' Hexadecimal number
&B00101101              ' Binary number
"text"                  ' Char constant
(0, 0, 0, 0, 0, 0)      ' Position constant
```

Standard variables

```
P1 = (0, 0, 0, 0, 0, 0)     ' Position variable ( starts with „P” ).
J1= (0, 0, 0, 0, 0, 0)      ' Joint variable  (starts with „J” ).
M1 = 1                      ' Number variable ( starts with „M” ).
C1$ = "text"                ' Char variable ( starts with „C” ).
```

Define variables

```
DEF INTE    iVar        ' Define an integer ( variable name )
iVar = 1                ' Assign the variable
SPD iVar                ' We can use integer for speed too.

DEF FLOAT   fVar        ' Define a float ( variable name )
fVar = 1.2              ' Assign the variable

DEF DOUBLE  dVar        ' Define a double ( variable name )
dVar = 1.2              ' Assign the variable

DEF CHAR    cVar        ' Define a char ( variable name )
cVar = "text"           ' Assign the variable
```

Array

Define array

```
DIM PPAL(4, 3)          ' Define a 2D array for positions. The size is 4*3.
```

Use array

```
PPAL(1,2) = (0, 0, 0, 0, 0, 0)  ' Assigne an element of the array.
MOV         PPAL(1,2)           ' Read the value of an array
```

Global variables

Create a program with the name **“UBP.MB5”** (User Based Program). If you define variables within this program, it will be a global variable. You can change the name of user Based Program with **“PRGUSR”**

Multitasking

Load and run program

```
XLOAD   2,"PRG1"        '[#, PRG]       Load program into a slot ( slot number, "programname" )
XRUN    2               '[#]            Run the preloaded program ( slot number )

XRUN    2, "PRG1", 0    '[#, PRG, 0/1]  Run a program in a slot ( slot number, "program name",
                        '               0: continuous operation / 1: cycle stop operation )
```

Stop the program

```
XSTP    2               '[#]            Stop the program in a slot.
```

Reset the program

```
XRST    2               '[#]            Reset the program in a slot.
```

Subprogram call

In main program

```
CALL    "PRG1", M1, P1  '[PRG, parameters]  Call a subprogram with parameters.
                        '                   After finishing the sub program, the execution will be
                        '                   continued after call instruction
```

In called program (PRG1)

```
FPRM    M01, P01,P02    '[parameters]   Define the order of parameters
IF  M01 = 1 THEN
    MOV   P01
ENDIF
MVS     P02
```

Parameters

TCP/IP configuration

NETIP = 192.168.0.1	Network IP
NETMSK = 255.255.255.255	Network mask
NETPORT = 10000, 10001, ...	Ethernet ports (realtime control port, OPT11, OPT12, ... OPT19)

Error handling

ROBOTERR = 1	Define when open the robot error output. 0: Never 1: At high level error 2: At low level error 3: At low- and high level error 4: At warning 5: At warning and high level error 6: At warning and low level error 7: At any error levels
BZR = 1	Switch on/off the buzzer (0: off; 1: on)

Hand parameters

HANDINIT = 1, 0, 1, 0, 1, 0, 1, 0	Initial value of hand outputs at power on
HAND-TYPE = D900, S902, ...	Select between double / single solenoid hand for each hand. D900: double at output number 900, 901 S902: single at output number 902

Program execution

SLT1 = "PRG1", REP, START	Define the initial slot configuration (program name, running mode, start mode) **Runing modes:** REP: execute repeatedly CYC: execute for one cycle **Start modes:** START: start after pushing the START button ALWAYS: start automaticly
CTN = 0	Continue the program execution after power off + power on? (0: not, 1: yes)

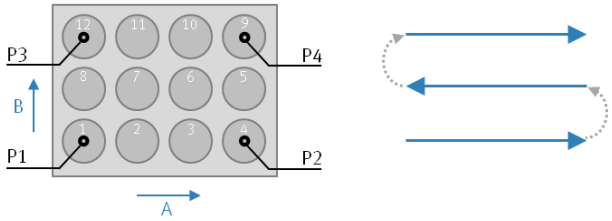
Pallet functions

Define pallet

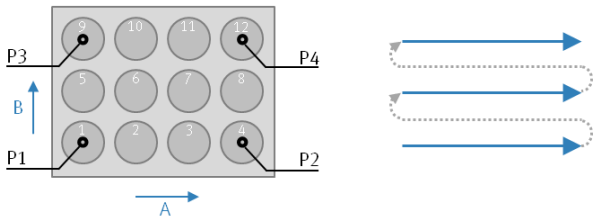
```
DEF PLT 1, P1, P2, P3, P4, 4, 3, 1      ' [#, POS, POS, POS, POS, #, #, #]
                                        ' Define a pallet ( pallet number, ' start position, end position A /
                                        ' transit position, endposition B, diagonal position, Quantity A,
                                        ' quantity B, type of pallet )
DEF PLT 2, P1, P2, P3, , 7, , 3         ' Define an arc pallet.
```

Types of pallets

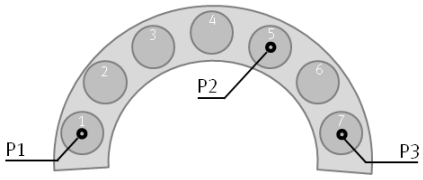
1: Zigzag (posture equal division)



2: Same direction (posture equal division)



3: Arc (posture equal division)



Move to pallet position

```
DEF POS  Pallet1        ' [name start with „P"]   Define the PHELP position
Pallet1 = PLT 1, 5      ' [#, #]                  Assign a position with coordinates of pallet 1, position 5
MOV      Pallet1, -40   '                         Move to pallet position

MOV      (PLT 1, 5), -40 '                        Move above to pallet 1, position 5
```