# COSIMIR® · MELFA-BASIC IV

## Alphabetical Overview of MELFA-BASIC-IV commands

| Command | Function |
|---|---|
| ACCEL | (Accelerate) |
| ACT | (Act) |
| BASE | (Base) |
| CALLP | (Call P) |
| CLOSE | (Close) |
| CLR | (Clear) |
| CMP POS | (Composure Posture) |
| CMP TOOL | (Composition Tool) |
| CMP OFF | (Composition OFF) |
| CMPG | (Composure Gain) |
| CNT | (Control) |
| COM OFF | (Communication OFF) |
| COM ON | (Communication ON) |
| COM STOP | (Communication Stop) |
| DEF ACT | (Define Act) |
| DEF CHAR | (Define Character) |
| DEF DOUBLE | (Define Double) |
| DEF FLOAT | (Define Float) |
| DEF FN | (Define Function) |
| DEF INTE | (Define Integer) |
| DEF IO | (Define IO) |
| DEF JNT | (Define Joint) |
| DEF PLT | (Define Pallet) |
| DEF POS | (Define Position) |
| DIM | (Dim) |
| DLY | (Delay) |
| END | (End) |
| ERROR | (Error) |
| FINE | (Fine) |
| FOR NEXT | (For Next) |
| FPRM | (FPRM) |

# COSIMIR® · MELFA-BASIC IV

## ACCEL (Accelerate)

**Function:**

Designate the robot's acceleration and deceleration speeds as a percentage (%).

**Format**

ACCEL[<Acceleration rate>] [, <Deceleration rate>]

**Terminology**

<Acceleration/Deceleration>

Designate the acceleration/deceleration to reach the maximum speed from speed 0 as a percentage.

This can be described as a constant or variable. When omitted, it will be interpreted as 100,100. Unit:[%]

**Explanation**

(1) The maximum acceleration/deceleration is determined according to the robot being used. Set the corresponding percentage. The system default value is 100,100.

(2) The acceleration percentage changed with this command is reset to the system default value when the program is reset or the END statement executed.

(3) The smooth operation when CNT is valid will have a different locus according to the acceleration speed or operation speed. To move smoothly at a constant speed, set the acceleration and deceleration to the same value. CNT is invalid in the default state.

**Reference Program**

10 ACCEL 50,100' Heavy load designation (when acceleration / deceleration is 0.2 seconds, the acceleration will be 0.4, and the deceleration will be 0.2 seconds)

20 MOV P1

30 ACCEL 100,100' Standard load designation

40 MOV P2



# COSIMIR® · MELFA-BASIC IV

**ACT (Act)**

**Function:**

Designates the enable/disable status of the interrupt.

**Format**

ACT<Priority No.> = <Enable/disable>

**Terminology**

<Priority No.>

Either enables or disables the entire interrupt.

1 ~ 8: Designate the priority No. for the interrupt defined in the DEF ACT statement. When entering the priority No., always leave a space (character) after the ACT command. If described as ACT1, it will be a variable name declaration statement.

<Enable/disable>

To enable interrupts, choose 1, to disable them, choose 0.

**Explanation**

(1) When the program starts, the status of <Priority No.> 0 is "enabled." When <Priority No.> 0 is "disabled," even if <Priority No.> 1 to 8 are set to "enabled," no interrupt will be enabled.

(2) The statuses of <Priority No.> 1 to 8 are all "disabled" when the program starts.

(3) An interrupt will occur only when all of the following conditions have been satisfied:

<Priority No.> 0 is set to "enabled."

The status of the DEF ACT statement has been defined.

When the <Priority No.> designated by DEF ACT is made valid by an ACT statement.

(4) The return from an interrupt process should be done by describing either RETURN 0 or RETURN 1.

(5) Even if the robot is in the middle of interpolation, an interrupt defined by a DEF ACT statement will be executed.

(6) During an interrupt process, that <Priority No.> will be executed with the status as "disable."

(7) A communications interrupt (COM) has a higher priority than an interrupt defined by a DEF ACT statement.

(8) The relationship of priority rankings is as shown below:

COM > ACT > WTHIF (WTH) > Pulse substitution

**Reference Program**

10 DEF ACT 1,M_IN(1)=1 GOSUB *INTR' Assign input signal 1 to the interrupt 1 condition

20 MOV P1

30 ACT 1=1' Enable interrupt 1

40 MOV P2

50 ACT 1=0' Disable interrupt 1

:

100 *INTR' If input signal 1 changes to ON (to 1) while the robot

' is moving from P1 to P2, it will stop.

110 HLT' Stop

120 RETURN 0

**Reference**

DEF ACT, RETURN

# COSIMIR® · MELFA-BASIC IV

## BASE (Base)

**Function:**

Designates the base conversion data.

**Format**

BASE<Base conversion data>

**Terminology**

<Base conversion data>

Designate the base conversion data in terms of a position operation expression.

**Explanation**

(1) The X, Y, and Z components of the position data represent the amount of parallel movement from the origin of the world coordinate system to the origin of the base coordinate system. The base conversion data can be changed only with the BASE command. The components A, B, and C of the position data represent the amount that the base coordinate system is tilted in relation to the world coordinate system.

X ... Distance to move parallel to X axis

Y ... Distance to move parallel to Y axis

Z ... Distance to move parallel to Z axis

A ... Angle to turn toward the X axis

B ... Angle to turn toward the Y axis

C ... Angle to turn toward the Z axis

L 1 .. Movement amount of additional axis 1

L 2 .. Movement amount of additional axis 2

(2) For A. B and C. the clockwise direction looking from the front of the origin of the coordinate, used as the

center of rotation, is the forward rotation direction.

(3) The contents of the structural flag have no meaning.

(4) The base coordinate system changed by the BASE command will keep the changes even after the power has

been turned OFF.

(5) The system's default value for this data is P_NBASE=(0,0,0,0,0,0) (0,0). This is calculated with the 6-axis three dimensional regardless of the mechanism structure.

**Reference Program**

10 BASE (50,100,0,0,0,0,0,0)' Input the conversion data as a constant

20 MVS P1

30 BASE P2' Input the conversion data as a variable

40 MVS P1

50 BASE P_NBASE ' Reset the conversion data to the default values.

**Reference**

TOOL, Robot status variables



# COSIMIR® · MELFA-BASIC IV

## CALLP (Call P)

**Function:**

Executes the designated program. (Program call version of GOSUB)

**Format**

CALLP "<Program name>" [, <Argument> [,<Argument>] ... ]

**Terminology**

<Program name>

Designate the program name with a character string constant or character string variable.

<Argument>

Designate the variable to be transferred to the program when the program is

called up. Up to 16 variables can be transferred.

**Explanation**

(1) If the argument differs from the type of argument defined (FPRM) in the CALLP called program, an error will occur. If the number of arguments differs from the CALLP called program, an error will occur when execution is started. When the program is reset, the control will return to the head of the main program (program in which host program does not exist.)

(2) The definition statements (DEF ACT, DEF FN, DEF PLT, DIM) executed in the CALLP calling program are invalid in the CALLP called program. These will be validated again when the program returns from CALLP.

(3) The speed and tool data are all valid.

(4) Up to seven programs can be called out from one program. This number of programs includes when a program is called from the subprogram.

(5) A file already opened and executed cannot be called from another slot and used. It can be called repeatedly from the same slot.

(6) A program cannot call itself out.

(7) A host program cannot be called.

(8) The argument is received with the FPRM command by the receiving program.

(9) The results calculated with the CALLP called program cannot be substituted in the argument variable and used in the CALLP calling program. In this case, use external variables to transfer the value.

**Reference Program**

10 CALLP "P10" ,M1,P1,P2

"P10" Program side

10 FPRM M1, P1, P2

**Reference**

END, FPRM

## CLOSE (Close)

**Function:**

Closes the designated file.

**Format**

CLOSE[[#] <File No.> [, [[#] <File No.> ... ]

**Terminology**

<File No.>

Designate the No. of the file to be closed.

**Explanation**

(1) If a file has been opened for input/output the CLOSE statement will sweep out the data in the buffer. Consequently, the output processing for the file can be completed properly.

(2) Executing an END statement will also close a file.

**Reference Program**

```
10 OPEN "COM1:" AS #1' Open "COM1:" as file No. 1

20 PRINT #1,M1

:

100 INPUT #1,M2

110 CLOSE #1' Close file No. 1, "COM1:"

:

200 CLOSE' Close all open files
```

**Reference**

END

### CLR (Clear)

**Function:**

Clears the general-purpose output signals, local variables in program, and global variables between programs.

**Format**

CLR<Type>

**Terminology**

<Type>

0All steps 1 to 3 below are executed.

1The general-purpose output signals are cleared based on the output reset pattern.

The output reset pattern is designated with parameters ORST0 to ORST224.

(0: OFF, 1: ON, *:Hold)

2All local numeric variables and numeric array variables used in the program are cleared to zero.

3All global numeric variables and numeric array variables used between programs are cleared to zero.

**Explanation**

(1) Either a constant or variable can be used for <Type>.



# COSIMIR® · MELFA-BASIC IV

### CMP OFF (Composition OFF)

**Function:**

Cancels the robot,s soft state.

Note: Excluding model RP-*

**Format**

CMP OFF

**Explanation**

(1) Cancels the robot,s soft state.



# COSIMIR® · MELFA-BASIC IV

## CMP POS (Composure Posture)

**Function:**

Designates which axis to softly move the robot in the orthogonal coordinate system.

Note: Excluding model RP-*

**Format**

CMP POS<Axis designation>

**Terminology**

<Axis designation>

Designate axis to move softly with a bit pattern.

00000000 ... from low-order bit X, Y, Z, ... C, L1, L2

**Explanation**

(1) The robot can be moved softly with the orthogonal coordinate system.

(2) For example, when inserting a pin in the vertical direction, if the X, Y, A and B axes are set to soft operation, the pin can be inserted smoothly.

(3) The degree of softness can be designated with the CMPG command.

(4) The soft state is maintained even after the robot program execution is stopped until the CMP OFF is executed or until power is turned ON again after being turned OFF.

(5) When pressing in the soft state, the robot cannot move to positions that exceed the operation limit of each joint axis.

(6) If the deviation between the actual robot position and the target position is larger than the parameter designated value NMPCDIST due to pressing, etc., the robot will stop. Note that the robot will not stop while the program

is running.

(7) If the CMP POS command is executed while the CMP TOOL command is functioning, an error will occur. Cancel the CMP OFF command before executing the CMP POS command.

(8) If the servo turns from OFF to ON while this command is active, the robot position could change.

(9) Jogging is possible in the soft state.

**Reference Program**

```
10MOV P1

20CMPG 0.5,0.5,1.0,0.5,0.5' Set softness

30CMP POS, &B00011011' Enter soft state

40MVS P2

50M_OUT(10)=1

60MVS P1

70CMP OFF' Return to normal state
```



# COSIMIR® · MELFA-BASIC IV

## CMPG (Composition Gain)

**Function:**

Designates the robots,s softness.

Note: Excluding model RP-*

**Format**

CMPG[<X axis gain>], [<Y axis gain>], [<Z axis gain>], [<A axis gain>], [<B axis gain>],[<C axis gain>], [<L1 axis gain>], [<L2 axis gain>]

**Terminology**

<X to L2 axis gain>

The softness can be set for each axis.

1.0 is the normal state, and 0.0 is the softest state.

If the value is omitted, the current setting value will be applied.

**Explanation**

(1) The softness can be designated in each axis units.

(2) The soft state will not be entered unless validated with the CMP POS or CMP TOOL command.

(3) A spring-like force will be generated in the proportion to the deviation of the command position and actual position. CMPG designates that spring constant.

(4) The deviation of the command position and actual position can be read with M_CMPDST. The success/failure of pin insertion can be checked using this variable.

(5) If a small gain is set, and the soft state is entered with the the CMP POS and CMP TOOL commands, the robot position could drop. Set the softness state gradually while checking it.

(6) If this command is executed in the soft state, the softness can be changed during the operation.



# COSIMIR® · MELFA-BASIC IV

## CNT (Continuous)

**Function:**

Designates continuous operation control for interpolation operations.

**Format**

CNT<Continuous operation mode/acceleration/deceleration operation mode>][, <Numer
value 1 > ] [Numeric value 2>]

**Terminology**

<valid/invalid>

Designate the continuous operation or acceleration/deceleration operation mode. Designa
either 1 or 0. 1 is continuous operation, and 0 is acceleration/deceleration operation.

<Numeric value 1> Designate the neighborhood distance for starting the orbit change wi
a mm unit.

<Numeric value 2> Designate the neighborhood distance for ending the orbit change wit
mm unit.

**Explanation**

(1) To make the change start position and end position both less than the designated valu
one is changed with a distance shorter than the designated value.

(2) The CNT default value is "CNT0".

(3) If numeric value 1 and numeric value 2 are omitted, these will be connected with the
default setting value.

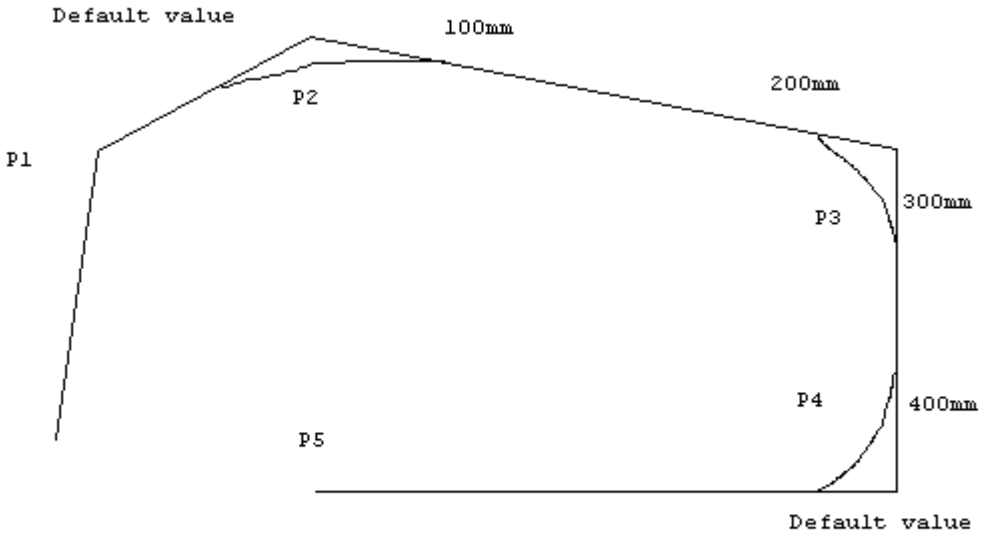
(4) If numeric value 2 is omitted, the same value as numeric value 1 will be applied.

(5) The FINE valid designation is invalid for smooth operation.

(6) If the numeric value designation is reduced, the operation time may take longer than
when CNT is invalid.

**Reference Program**

```
10 CNT 0' Invalidate CNT

20 MVS P1' Operate with acceleration/deceleration

30 CNT 1' Validate CNT (default setting value)

40 MVS P2' Execute with smooth operation

50 CNT 1, 100, 200' Start distance 100mm. end position 200mm

60 MVS P3' Execute with smooth operation

70 CNT 1, 300' Start distance 300mm, end position 300mm

80 MOV P4' Execute with smooth operation

90 CNT 0' Invalidate CNT

100 MOV P5
```




**Reference**

Movement related commands, speed related commands, acceleration/deceleration related commands, FINE, OADL



# COSIMIR® · MELFA-BASIC IV

## COM OFF (Communication OFF)

**Function:**

Designates "disable" for any interrupts from a communication line.

**Format**

COM[(<Communication line No.>)] OFF

**Terminology**

<Communication Line No.>

Describe 1, 2 or 3.

**Explanation**

(1) When COMMON OFF is executed, even if communications are attempted, the interrupt will not be generated.

(2) For information on communication line Nos., refer to the page for OPEN.

**Reference Program**

10 COM(1) OFF' Disable communication interrupt from communication line No. 1.

**Reference**

OPEN, COM_ON, COM STOP, ON COM GOSUB



# COSIMIR® · MELFA-BASIC IV

## COM ON (Communication ON)

**Function:**

Designates "enable" for interrupts coming from a communication line.

**Format**

COM[(<Communication line No.>)] ON

**Terminology**

<Communication Line No.>

Describe 1, 2 or 3.

**Explanation**

(1) For information on communication line Nos., refer to the page for OPEN.

**Reference Program**

10 COM(1) ON' Enable communication interrupt from communication line No. 1.

**Reference**

OPEN, COM_OFF, COM STOP, ON COM GOSUB

# COSIMIR® · MELFA-BASIC IV

## DEF ACT (Define act)

**Function:**

Defines the status of the interrupt and how it will be processed.

**Format**

DEF ACT<Priority No.>, <Expression>, <Process>

**Terminology**

<Priority No.>

This is the priority No. of the interrupt. It can be set with constant Nos. 1 to 8.

<Expression>

For the interrupt status, use the formats described below: (Refer to the syntax diagram)

<Numeric type data> <Comparison operator> <Numeric type data> or

<Numeric type data> <Logical operator> <Numeric type data>

<Numeric type data> refers to the following:

<Numeric type constant>| <Numeric variable>|<Numeric array variable>|<Component data>

<Process>

Refers to a GOTO statement or a GOSUB statement used to process an interrupt when it occurs.

**Explanation**

(1) The priority level for the interrupts is decided by the <Priority No.>, and the priority level, from the highest ranges from 1 to 8.

(2) There can be up to 8 settings for the interrupts. Use the <Priority No.> to differentiate them.

(3) When two interrupts have been defined with the same priority level, the one defined later is validated.

(4) Since DEF ACT defines only the interrupt, always use the ACT command to designate the enable/disable status of the interrupt.

(5) The communications interrupt (COM) has a higher priority level than any of the interrupts defined by DEF ACT.

(6) DEF ACT definitions are valid only in the programs where they are defined. These are invalid when called up in a program by CALL P. If necessary, the data in a subprogram may need to be redefined.

(7) If an interrupt is generated when a GOTO command is designated by <Process> for a DEF ACT command, during execution of the remaining program, the interrupt in progress will remain, and only interrupts of a higher level will be accepted. The interrupt in progress for a GOTO statement can be canceled with the execution of an END statement.

**Reference Program**

10 DEF ACT 1,M_IN(10)=1 GOSUB 100' Defines the subroutine at line 100 to be the one

to be called up when the status for the general

purpose input signal No.17 is ON.

20 DEF ACT 2,MFG1 AND MFG2 GOTO 200' Defines the line 200 as the one to jump to when

the logic operation of AND applied to MFG1 or MFG2

results in "true".

30 DEF ACT 3,M_TIMER(1)>10.5 GOSUB 300' When 10.5 seconds pass, the program jumps to

the line 300 subroutine.

100 M_TIMER(1)=0

110 ACT 3=1

**Reference**

ACT, GOSUB, RETURN, Robot status variables



# COSIMIR® · MELFA-BASIC IV

### DEF CHAR (Define Character)

**Function:**

Declares a character string variable

**Format**

DEF CHAR<Character string position variable name> [, <Character string variable name >] ...

**Terminology**

<Character string variable name>

Designate the variable name.

**Explanation**

(1) The variable name can have up to eight characters. Refer to the "Types of characters that can be used" for the characters that can be used.

(2) When designating multiple variable names, the maximum value (123 characters including command) can be set on one line.

(3) The declared variable follows the "Character string variable,, rules.

(4) After declaration, the variable can be used in the same manner as the C variable.

**Reference Program**

10 DEF CHAR MESSAGE' Declare MESSAGE as a character string variable.

20 MESSAGE = "WORKSET"' Substitute "WORKSET" in the MESSAGE variable.



# COSIMIR® · MELFA-BASIC IV

### DEF FN (Define Function)

**Function:**

Defines a function and gives it name.

**Format**

DEF FN<Name> [(Dummy argument> [, <Dummy argument>] ... )] =
<Function definition expression>

**Terminology**

<Name>

Describe an identifier character + character string.

<Dummy argument>

When a function has been called up, it is transferred to the function. It is possible to describe all the variables, and up to 16 variables can be used.

<Function Definition Expression>

Describe the expression for what operation to use as a function.

**Explanation**

(1) FN + <Name> becomes the name of the function. The function name can be up to 8 characters long.

**Example:**

Numeric typeFNMAX Identifier character: M

Character string typeFNCAME$ Identifier character:

C (Describe $ at the end of the name)

(2) A function defined with DEF FN is called a user-defined function. A function as long as one line can be described.

(3) Built-in functions and user-defined functions that have already been defined can be used in the function definition expression. In this case, up to 16 levels of user-defined functions can be written.

(4) If the variables used in <Function Definition Expression> are not located in <Dummy Argument>, then the value that the variable has at that time will be used. Also, an error will occur if during execution, the number or argument type (numeric value or character string) of arguments differs from the number or type declared.

(5) A user-defined function is valid only in the program where it is defined. It cannot be used by a CALLP designation program.

**Reference Program**

10 DEF FNMAVE(MA,MB)=(MA+MB)/2' Defines FNMAVE to be used to find the average

' of two numeric values.

20 MDATA1=20

30 MDATA2=30

40 MAVE=FNMAVE(MDATA1,MDATA2)' 25, the average of 20 and 30, is substituted

' for the numeric variable MAVE.

**Reference**

Variables, Array variables, Functions



# COSIMIR® · MELFA-BASIC IV

## DEF INTE/FLOAT/DOUBLE (Define Integer/Float/Double)

**Function:**

Declares the arithmetic variable.

**Format**

DEF INTE<Arithmetic variable name> [, <Arithmetic variable name>] ...

DEF FLOAT<Arithmetic variable name> [, <Arithmetic variable name>] ...

DEF DOUBLE<Arithmetic variable name> [, <Arithmetic variable name>] ...

**Terminology**

<Arithmetic Variable Name>

Designate the variable name.

**Explanation**

(1) The variable name can have up to eight characters. Refer to the "Types of characters that can be used" for the characters that can be used.

(2) When designating multiple variable names, the maximum value (123 characters including command) can be set on one line.

(3) The variable declared with INT will be an integer type. (-32768 ~ +32767)

(4) The variable declared with FLOAT will be a single-precision type. (+/- 1.70141E+38)

(5) The variable declared with DOUBLE will be a double-precision type. (+/- 1.701411834604692E+308)

**Reference Program**

10 DEF INTE WORK' Declare WORK as an arithmetic variable name.

20 WORK = 100 ' Substitute the value 100 in WORK.



# COSIMIR® · MELFA-BASIC IV

## DEF IO (Define IO)

**Function:**

Declares an input/output variable.

**Format**

DEF IO<Input/Output variable name> = <Type designation>, <Input/Output bit No.> [, <Mask information>]

**Terminology**

<Input/Output Variable Name>

Designate the variable name.

<Type Designation>

Designate BIT, BYTE, WORD or INTEGER.

<Input/Output Bit No.>

Designate the input or output bit No.

<Mask Information>

Designate when only a specific signal is to be validated.

**Explanation**

(1) The variable name can have up to eight characters. Refer to the „Types

of characters that can be used,, for the characters that can be used.

(2) When mask information is designated, onIy the specified signal will be validated.
Example: For the 2Oth line, mask processing is carried out with hexadecimal 0F, so Nos. 107 to 110 will be validated, and Nos. 111 to 114 will be invalidated (always 0).
InvalidValid
0 0 0 0 1 1 1 1
No.114 No. 107 (Input/output bit No.)

**Reference Program**

10 DEF IO PORT1 = BIT,6' Assign the input/output variable named PORT1 to the

' input/output bit No. 6 with the BIT type.

20 DEF IO PORT2 = BYTE,8,&H0F' Assign the input/output variable named PORT2 to the

' input/output bit No. 8 with the BYTE type, and designate

' the mask information as hexadecimal 0F.



# COSIMIR® · MELFA-BASIC IV

## DEF JNT (Define Joint)

**Function:**

Declares a joint variable.

**Format**

DEF JNT<Joint variable name> [, <Joint variable name>] ...

**Terminology**

<Joint Variable Name>

Designate the variable name.

**Explanation**

(1) The variable name can have up to eight characters. Refer to the "Types of characters that can be used" for the characters that can be used. The

declared variable follows the "Joint variable" rules. After declaration, the variable can be used in the same manner as the J variable.

**Reference Program**

10 DEF JNT SAFE' Declare SAFE as the joint variable

20 MOV SAFE' Move to SAFE



# COSIMIR® · MELFA-BASIC IV

## DEF PLT (Define Pallet)

**Function:**

Defines the pallet. (3-point pallet, 4-point pallet)

**Format**

DEF PLT<Pallet No.>, <Start point>, <End point A>, <End point B>, [<Diagonal point>], <Quantity A>, <Quantity B>, <Assignment direction>

**Terminology**

<Pallet No.>

This is the selection No. of the set pallet. (1 to 8)

<Start Point>

Refers to the pallet's start point. Position operation expressions can be described.

<End Point A>

One of the ending points for the pallet. Transit point of arc for arc pallet. Position operation expressions can be described.

<End Point B>

Another ending point for the pallet. Transit point of arc for arc pallet. Position operation expressions can be described.

<Diagonal Point>

The diagonal point from the pallet's start point. Insignificant for arc pallet. Position operation expressions can be described.

<Quantity A>

The No. of workpieces from the pallet's start point to the end point A. The No. of workpieces between the pallet start point and arc end point when using an arc pallet. Numeric operation expressions can be described.

<Quantity B>

The No. of workpieces from the pallet's start point to the end point A. Insignificant for an arc pallet. (0, etc., must be designated.) Numeric operation expressions can be described.

<Assignment Direction>

Describe 1, 2 or 3 for the direction of assignment for when the grid is divided and given numbers.

Numeric operation expressions can be used. (Refer to the explanation section for details on the assignment direction.)

**Explanation**

(1) A 3-point pallet or a 4-point pallet can be selected.
- A 3-point pallet offers simplified teaching.
- A 4-point pallet is effective for better precision.

(2) The assignment direction is as follows.
Zigzag = 1 , Same direction = 2, Arc pallet = 3

(3) The command is valid onIy within the program being executed. The command is invalid in the program that calls up the command from another program. If necessary, redefine.

(4) Quantity A and B should be a non-zero positive number, while if 0 or a negative number is assigned, an error will occur.

(5) If Quantity A x Quantity B exceeds 32,767, an error will occur when operation starts.

(6) The value of quantity B is insignificant for the arc pallet, but it must not be omitted. The diagonal point will be insignificant even when designated.

**Reference Program**

```
10 DEF PLT 1,P1,P2,P3, ,4,3,1 ' 3-point pallet definition

20 DEF PLT 1,P1,P2,P3,P4,4,3,1' 4-point pallet definition
```

**Reference**

PLT, Movement commands

# COSIMIR® · MELFA-BASIC IV

## DEF POS (Define Position)

**Function:**

Declares a position variable.

**Format**

DEF POS<Position variable name> [, <Position variable name>] ...

**Terminology**

<Position Variable Name>

Designate the variable name.

**Explanation**

(1) The variable name can have up to eight characters. Refer to the "Types of characters that can be used" for the characters that can be used.

(2) When designating multiple variable names, the maximum value (123 characters including command) can be set on one line.

(3) The declared variable follows the "Position variable" rules.

(4) After declaration, the variable can be used in the same manner as the P variable.

**Reference Program**

10 DEF POS WORKSET' Declare WORKSET as the position variable.

20 MOV WORKSET' Move to WORKSET

©·2000·EFR·IRF·GERMANY

# COSIMIR® · MELFA-BASIC IV

## DIM (Dim)

**Function:**

Declares the quantity of elements in the array variable. (Arrays up to the third dimension are possible.)

**Format**

DIM<Variable name> (<Max. value of subscript>, [<Max. value of subscript>] ) [,<Variable name> (<Max. value of subscript> [, <Max. value of subscript>])]

**Terminology**

<Variable Name>

Describe the name of the array variable.

<Max. Value of Subscript>

Describe in terms of constants, the number of elements in an array variable.

**Explanation**

(1) A one-dimensional, two-dimensional or three-dimensional array can be used.

(2) <Max. Value of Subscript> can be described with numeric constants from 1 to 999. Numeric operation expressions cannot be used.

(3) If a subscript with a value larger than the maximum value defined with DlM is used, an error will occur at the execution.

(4) When the DlM statement is executed, the array variable does not have a default value, and instead is non-specified.

(5) An array cannot be used without the DlM statement. If the number of elements is a real number, the decimal numbers will automatically be rounded off to create an integer.

(6) The command is valid onIy within the program being executed. The command is invalid in the program that calls up the command from another program.

(7) This command must be redefined when used in a subprogram. A value (teachable) can be directly set or read from the position array variable data declared with this command.

**Reference Program**

10 DIM PDATA(10)' Defines the position array variable PDATA having ten elements.

20 DIM MDATA(2,3)' Defines the two-dimensional numeric array variable

MDATA having 2x3 elements.

**Reference**

Array variables



# COSIMIR® · MELFA-BASIC IV

## DLY (Delay)

**Function:**

When used as a single command:

At a designated time, it causes a wait.

When used as an additional pulse output:

Designates an output time for a pulse

**Format**

DLY<Time>

**Terminology**

<Time>

Describes the waiting time or the output time for the pulse output, in terms of a numeric operation expression. Unit: [Seconds]

The minimum value that can be set is 0.05 seconds.

**Explanation**

(1) This command is used to create delay times in programs, or to write in the OUT statement and set the pulse output time.

(2) The pulse output will be executed simultaneously as the next command in the lines that follow.

(3) Up to 4 pulse outputs can be issued simultaneously. Exceeding this, an error will occur when the program tries to execute it.

(4) After the designated time has elapsed, the condition that existed just before the pulse output was issued will resume.

(5) Within the designated time, if an END command, or the last line of the program is executed, or if an emergency stop is made to stop the program, the signal output status will keep its current status.

(6) The relation of the priority levels for other interrupts is as shown below:

COM> ACT> WTHIF (WTH) >Pulse output (Time setting ON)

**Reference Program**

10 DLY 30' Wait for 30 seconds

20 M_OUT(17)=1 DLY 10.0' Send the signal output to the general-purpose

' output signal 17 for 10 seconds.

**Reference**

Robot Status Variables, Substitution



# COSIMIR® · MELFA-BASIC IV

**END (End)**

**Function:**

Ends the program execution.

**Format**

END

**Explanation**

(1) Several END statements can be described within one program.

(2) The END statement does not need to be described at the end of the program. Used in the host program, the command will close all files before closing.

(3) An END statement in a program called up by a CALLP command will transfer control to the program that issued the CALL P command.

(4) Used in the host program, the command will close all files before closing.

(5) At program END; the SPD, ACCEL, OADL, JOVRD, OVRD, FINE,

and CNT settings will be initialized.

**Reference Program**

100 END

**Reference**

CALLP, FPRM, GOSUB, RETURN



# COSIMIR® · MELFA-BASIC IV

## ERROR (Error)

**Function:**

Generates an error in the user program.

**Format**

ERROR<Error no.>

**Terminology**

<Error no.>

Either a constant or a numeric operation expression can be set. Designate the No. within the range of 9000 to 9299.

**Explanation**

(1) By executing this command, the error can be transmitted to an external source.

(2) The system will operate as follows according to the number.

| No. | System behavior |
|---|---|
| 9000 - 9099 | The program execution is stopped, and the servo power is shut off. |
| (H level error) | The error state is reset when error reset is input. |
| 9100 - 9199 | The program execution is stopped. |
| (L level error) | The error state is reset when error reset is input. |
| 9200 - 9299 | The program execution is continued. |

(CAUTION) The error output turns OFF with error reset.

**Reference Program**

100 ERROR 9000


# COSIMIR® · MELFA-BASIC IV

## FINE (Fine)

**Function:**

Designates the positioning end conditions for interpolation when not in smooth operation control.

**Format**

FINE<Invalid/No. of pulses> [, <Axis No.>]

**Terminology**

<No. of pulses>

Designate the positioning pulses. This will be invalid to when set to 0. The default value is 0.

<Axis No.>

Designate the axis No. to which the positioning pulses are to be designated. The positioning pulses will be applied on all axes when omitted.

**Explanation**

(1) FINE is invalid in the program until the FINE command is executed. Once FINE is validated, it remains valid until invalidated.

(2) FINE is invalidated at the end of the program.

(3) When the CNT valid state (in smooth operation control) is entered, the FINE command will be ignored even if it is valid (i.e., it will be treated as invalid, but the status will be kept).

**Reference Program**

10 FINE 300' Designate 300 for the positioning pulses.

20 MOV P1

30 FINE 100,2' Change the 2nd axis positioning pulses to 100.

40 MOV P2

50 FINE 0' Invalidate the positioning pulse designation.

60 MOV P3

70 FINE 100' Designate 100 for the positioning pulses.

80 MOV P4

**Reference**

CNT,movement command



# COSIMIR® · MELFA-BASIC IV

## FOR NEXT (For Next)

**Function:**

Repeatedly executes the program between the FOR statement and NEXT statement until the end conditions are satisfied.

**Format**

FOR <Counter> = <Default value> TO <End value> [STEP <Increment>]
.
.
NEXT [<Counter 1> [, <Counter2>] ... ]

**Terminology**

<Counter>

Describes the numeric value data that represents the counter for the number of repetitions.

Same for <Counter 1> and <Counter 2>.

<Default Value>

Sets default value of the counter for the number of repetitions as a numeric operation expression.

<End Value>

Set the end value of the counter for the number of repeats as a numeric operation expression.

<Increment>

Sets the value of the increments for the counter for the number of repetitions as a numeric operation expression.

**Explanation**

(1) Program depth
- It is possible to describe FOR-NEXT statements between other FOR-NEXT statements.
- With each FOR-NEXT process, the control structure of the program becomes one level deeper.
In this system, it is possible for the control structure to have up to 16 levels within a program. Exceeding 16 levels will cause an error during execution.

**Reference Program**

A program that adds the numbers 1 to 10

10 MSUM=0' Initialize the total MSUM.

20 FOR M1=1 TO 10 STEP 1' Increase the counter by 1 from 1 to 10 for the numeric variable M1.

If 10 is exceeded, go to line 50. Step 1 can be omitted.

30 MSUM=MSUM+M1' Add M1 value to numeric variable MSUM.

40 NEXT M1' Return to line 20.

50 END' End the program.

A program that puts the result of a product of two numbers into a 2-dimensional array variable

(Using FOR-NEXT as a nesting structure)

10 DIM MBOX(10,10)' Reserve space for a 10 x 10 array.

20 FOR M1=1 TO 10 STEP 1' Increase the counter by 1 from 1 to 10 for the numeric variable M1.

If 10 is exceeded, transfer control to line 70. Step 1 can be omitted.

30 FOR M2=1 TO 10 STEP1' Increase the counter by 1 from 1 to 10 for the numeric variable M2.

If 10 is exceeded, transfer control to line 60. Step 1 can be omitted.

40 MBOX(M1,M2)=M1*M2' Substitute the value of M1*M2 for the array

variable MBOX (M1, M2).

50 NEXT M2' Retum to line 30.

60 NEXT M1' Retum to line 20.

70 END' End the program.

Cases in which the repeated control is not executed are shown below.

- The counter's <Default Value> is greater than <End Value> and <Increment> is a positive number.

- The counter's <Default Value> is smaller than <End Value>, and <Increment> is a negative number.

If the FOR statement and NEXT statement contradict each other, an error will occur.

Note that when the FOR and NEXT statements are in a nesting structure and have the same end value, the statement can be described with one NEXT statement. For example, line 50 and 60 in the example can be combined to be written "NEXT M2, M1" in one line.

When the NEXT statement corresponds to the closest FOR statement, the variable name in the NEXT statement can be omitted. In the example, "M2" in line 50 and "M1" in line 60 can be omitted.

**Reference**

GOSUB, WHILE, END, Numeric operation expressions (Syntax diagram)



# COSIMIR® · MELFA-BASIC IV

## FPRM (FPRM)

**Function:**

Defines the order of the arguments, the type, and number for the main program that uses arguments in a subprogram (i.e., when the host program uses another program with CALLP).

**Format**

FPRM<Dummy argument> [, <Dummy argument> ... ]

**Terminology**

<Dummy Argument>

The variable in the subprogram that is transferred to the main statement when executed.

All variables can be used. Up to 16 variables may be used.

**Explanation**

(1) FPRM is unnecessary if there are no arguments in the subprogram that is called up.

(2) If a variable used in a program is not in <Dummy Argument> the variable will use the value it currently holds.

(3) Depths between programs
- The calling up of programs allows the level of the control structure between programs to go deeper.
- One calling up of a program results in the program structure becoming one level deeper.

(4) It is possible for the control structure in a program to go as deep as 7 levels. If 7 levels are exceeded, an error will occur during execution.

(5) The calculation results in the subprogram cannot be transferred to the host program using temporary arguments. In this case, transfer the results using external variables.

**Reference Program**

10 FPRM M1,P1,P2

**Reference**

CALLP, Variables, Array variables, Functions



# COSIMIR® · MELFA-BASIC IV

## GETM (Get Mechanism)

**Function:**

Acquires the mechanism resource for the designated mechanism No.

**Format**

GETM<Mechanism no.>

**Terminology**

<Mechanism no.>

Designate either a numeric value or variable.

The standard system,s robot arm is assigned to mechanism 1.

**Explanation**

(1) If the mechanism,s resource is not acquired with GETM, the movement commands of motor power ON/OFF commands cannot be executed..

(2) GETM cannot be commanded to a mechanism for which GETM has already been executed.

(3) If the argument is omitted from the system status variable requiring the mechanism designation, the currently acquired mechanism will be designated.

(4) Main slot (slot 1) acquires mechanism 1 in the default state.

(5) If the program is stopped, RELM will be executed automatically by the system. When the program is restarted, GETM will be executed automatically.

**Reference Program**

```
10GETM 1' Acquire mechanism 1 resource

20SERVO ON' Turn mechanism 1 servo ON

30MOV P1

40MVS P2

45P3=P_CURR' Substitute P3 in mechanism 1 current position.

50SERVO OFF' Turn mechanism 1 servo OFF.

60RELM' Release mechanism 1 resource.

70END
```

### GOSUB (Go Subroutine)

**Function:**

Calls up the subroutine at the designated line No. or line label.

**Format**

GOSUB<Call destination>

**Terminology**

<Call Destination>

Describes the line No. or label name.

**Explanation**

(1) When calling a subroutine in the base program, use a label name starting with "L_". Note that if the same label is found in the local program, the local program's routine will be executed.

**Reference Program**

```
100 GOSUB 1000

110 END

1000 MOV P1

1010 RETURN
```

**Reference**

RETURN, END, GOTO, DEF ACT, ACT



# COSIMIR® · MELFA-BASIC IV

### GOTO (Go To)

**Function:**

Unconditionally branches to a designated line No. or label.

**Format**

GOTO<Branch destination>

**Terminology**

<Branch Destination>

Describes the line No. or label name.

**Explanation**

(1) GOTO cannot be commanded to a label in the base program. Thus, even if the designated label name is in the base program, the no skip destination error will occur.

**Reference**

IF THEN ELSE, GOSUB, DEF ACT, ACT



# COSIMIR® · MELFA-BASIC IV

## HLT (Halt)

**Function:**

Interrupts the execution of the program and movement of the robot, and stops.

**Format**

HLT

**Explanation**

(1) Interrupts the execution of a program and decelerates the robot to a stop. The system will enter the waiting state.

(2) To restart, start the O/P or issue the start signal from an external source. The program will be restarted at the next line after the HLT statement.

(3) Note that if the HLT statement is an appended statement, the operation will restart from the same line of the program where it was interrupted.

**Reference**

WTHIF, WTH, SKIP

# COSIMIR® · MELFA-BASIC IV

## HOPEN/HCLOSE (Hand Open/Close)

**Function:**

Commands the hand to open or close.

**Format**

HOPEN<Hand No.> [, <Starting grasp force>, <Holding grasp force>, <Starting grasp force holding time>]

HCLOSE<Hand No.>

**Terminology**

<Hand No.>
Select a numeric value between 1 and 8.

<Starting grasp forcer>
This parameter is valid for the motorized hand, and invalid for any other type of hand.
Set the required grasping force for starting the hand open/close.
Set the grasping force as a step between 0 and 63 (63 = 3.5kgf).
The default value is 63. When omitted, the previous setting value will be applied.

<Holding grasp force>
This parameter is valid for the motorized hand, and invalid for any other type of hand.
Set the required grasping force for holding the hand open / close.
Set the grasping force as a step between 0 and 63 (63 = 3.5kgf).
The default value is 63. When omitted. the previous setting value will be applied.

<Starting grasp force holding timer>
This parameter is valid for the motorized hand.
Set the time to hold the starting grasping force as a value from 0.00 (sec.).
The default value is 0.3 sec.

**Explanation**

(1) The operation (single/double) of each hand is set with parameter HANDTYPE.

(2) The status of the hand output signal when the power is turned ON is set with parameter HANDINIT.

(3) The hand input signal can be confirmed with the robot status variable M-HNDCQ ("Hand input number"). The signal can also be confirmed with the input signals No.900 to 907 (when there is one mechanism).
10 HCLOSE 1
20 IF M_HNDCQ(1)<>1 THEN GOTO 20
30 MOV P1

(4) There are related parameters. Refer to "1.6 Functions set with parameters" on page 33 of this manual.

**Reference Program**

10 HOPEN 1' Open hand 1.

20 HCLOSE 2' Close hand 2.

**Reference**

Robot Status Variable



# COSIMIR® · MELFA-BASIC IV

## IF THEN ELSE (If Then Else)

**Function:**

A process is selected and executed according to the results of an expression.

**Format**

IF<Expression> THEN <Process> [ELSE <Process>]

**Terminology**

<Expression>

Describe the expression targeted for comparison as a comparison operation expression or logic operation expression.

<Process>

Describe the process following THEN for when the comparison results are true, and the process following ELSE for when the comparison results are false.

**Reference Program**

100 IF M1>10 THEN 1000

110 IF M1>10 THEN GOTO 1000 ELSE GOTO 2000

**Reference**

ON GOSUB, ON GOTO, Numeric operation expressions (syntax diagram)



# COSIMIR® · MELFA-BASIC IV

## INPUT (Input)

**Function:**

Inputs data from a file (input device). All data uses the ASCII format.

**Format**

INPUT# <File No.>, <Input data name> [, <Input data name>] ...

**Terminology**

<File No.>

Describe a number between 1 and 8.

This corresponds to the file No. assigned with the OPEN command.

<Input Data Name>

Describe the variable name for saving the input data. All variables can be described.

**Explanation**

(1) Data is input from file (input device) having the file No. opened with the OPEN statement, and is substituted in the variable. If the OPEN statement has not been executed, an error will occur.

(2) The type of data input and the type of variable that is substituting it must be the same.

(3) When describing multiple variable names, use a comma (,) between variable names as delimiters.

(4) When the INPUT statement is executed, the status will be "standby for input". The input data will be substituted for the variables at the same time

as the carriage return (CR and LF) are input.

(5) If multiple elements are input due to the number of arguments in the INPUT statement, reading will continue to the next INPUT statement. When the END or CLOSE statement is executed, the data saved in the buffer will be erased.

Example: To input both a character string, numeric value and position.

10 INPUT# 1,C1$,M1,P1

PRN MELFA,125.75,(130.5,-117.2,55.1,16.2,0,0)(1,0)

MELFA is substituted in C1$, 125.75 in M1, and (130.5, -117.2,55.1,16.2,0,0)(1,0) in P1.

**Reference Program**

10 OPEN "COM1:" AS #1 ' Assign RS-232-C for file No. 1.

20 INPUT# 1,M1 ' When data is input from the keyboard, that value

' will be input in the numeric variable M1

**Reference**

OPEN, PRINT



# COSIMIR® · MELFA-BASIC IV

## JOVRD (J Override)

**Function:**

Designates the override that is valid only during the robot,s joint movements.

**Format**

JOVRD<Designated override>

**Terminology**

<Designated override>

Describes the override as a real number. A numeric operation expression can also be described. Unit: [%] (Recommended range: 1 to 100.0)

**Explanation**

(1) The JOVRD command is valid only during joint interpolation.

(2) The actual override is = (Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Joint override (JOVRD command)). The JOVRD command changes only the override for the joint interpolation movement.

(3) The 100% <Designate override> is the maximum capacity of the robot. Normally. the system default value (M_NOVRD) is set to 100%. The value is reset to the default value when the END statement is executed or the program is reset.

**Reference Program**

10 JOVRD 50

20 MOV P1

30 JOVRD M_NJOVRD' Set the default value

**Reference**

OVRD, SPD, Movement Commands, Robot Status Variables



# COSIMIR® · MELFA-BASIC IV

## (Label) *

**Function:**

This indicates the jump site.

**Format**

* <Label Name>

**Terminology**

<Label Name>

Describe a character string that starts with an alphabetic character.

Up to 8 characters can be used. (Up to 9 characters including *.)

**Explanation**

(1) An error will not occur even if this is not referred to during the program.

(2) If the same label is defined several times in the same program, an error will occur at the execution.

**Reference Program**

100 * SUB1

**Reference**

GOTO, GOSUB, Label



# COSIMIR® · MELFA-BASIC IV

## LOADSET (Load Set)

**Function:**

Designates the hand and workpiece conditions for carrying out optimum acceleration/deceleration.

**Format**

LOADSET<Hand condition No.>, <Workpiece condition No.>

**Terminology**

<Hand condition No.> Designate the hand condition (HNDDAT 1 to 8) No. for which the weight and size are designated.

<Workpiece condition No.> Designate the workpiece condition (WRKDAT 1 to 8) No. for which the weight and size are designated.

**Explanation**

(1) Set the hand conditions and workpiece conditions used for optimum acceleration / deceleration. This is used when setting the optimum acceleration / deceleration for workpiece types having different weights.

(2) The maximum load is set for the hand when the program execution starts.

(3) Set the weight, size (X, Y, Z) and center of gravity position (X, Y, Z) as the hand conditions in parameter (HNDDAT 1 to 8).

(4) Set the weight, size (X, Y, Z) and center of gravity position (X, Y, Z) as the workpiece conditions in parameter (WRKDAT 1 to 8).

(5) The hand conditions and workpiece conditions changed when this command is executed are reset to the system default value when the program is reset and when the END statement is executed.

(6) As the system default values, the hand conditions are set to the rated load, and the workpiece conditions are set to none (0kg).

(7) Refer to "OADL (Optimal Acceleration)" on page 130 for details on the optimum acceleration / deceleration.

**Reference Program**

```
5 OADL ON

10 LOADSET 1,1' Hand 1 and workpiece 1 conditions

20 MOV P1

30 MOV P2

40 LOADSET 1,2' Hand 1 and workpiece 2 conditions

50 MOV P1

60 MOV P2
```



# COSIMIR® · MELFA-BASIC IV

## MOV (Move)

**Function:**

Using joint interpolation operation, moves from the current position to the destination position.

**Format**

MOV<Target Position> [,[,<Close Distance>] [<Interpolation Type>]] [Appended conditions>]

**Terminology**

<Movement Target Position>

This is the final position for interpolation operation. Describe a movement position statement.

<Close Distance>
If this value is designated, the actual movement target position will be a position separated by the designated distance in the tool coordinate system Z axis direction (+/- direction).

<Interpolation Type>
Designate the interpolation type.

TYPE <Numeric constant 1>, <Numeric constant 2>

Numeric constant 1 ... Detour/short cut = 1/0

Numeric constant 2 ... Invalid

The default value is l, 0 (detour).

<Appended conditions>

The WTH and WTHIF statements can be used.

**Explanation**

(1) By using the WTH and WTHIF statement, the signal output timing and motion can be synchronized.

(2) The numeric constant 1 for the TYPE designates the posture interpolation amount.

(3) Detour refers to the operating exactly according to the teaching posture. Short cut operation may take place depending on the teaching posture.

(4) Short cut operation refers to posture interpolation between the start point and end point in the direction with less motion.

(5) The detour/short cut designation is significant when the posture axis has a motion range of 180 deg. or more.

(6) Even if short cut is designated, if the target position is outside the motion range, the axis may move with the detour in the reverse direction.

(7) The TYPE numeric constant 2 setting is insignificant for joint interpolation.

**Reference Program**

```
10 MOV P1 TYPE 1,0

20 MOV J1

30 MOV (PLT 1,10),100.0 WTH M_OUT(17)=1
```

40 MOV P4+P5,50.0 TYPE 0.0 WTHIF M_IN(18)=1,M_OUT(20)=1

**Reference**

Movement Commands, WTH, WTHIF, FINE, CNT, Movement Positions



# COSIMIR® · MELFA-BASIC IV

## MVC (Move C)

**Function:**

Carries out 3D circular interpolation in the order of start point, transit point 1, transit point 2 and start point.

**Format**

MVC<Start point>, <Transit point 1>, <Transit point 2> [<Additional condition>]

**Terminology**

<Start point>

The start point and end point for a circle. Describe a position operation expression or joint operation expression.

<Transit point 1>

Transit point 1 for a circular arc. Describe a position operation expression or joint operation expression.

<Transit point 2>

Transit point 2 for a circular arc. Describe a position operation expression or joint operation expression.

<Additional condition>

Describe a WTH conjunction or a WTHIF conjunction.

**Explanation**

(1) In circular interpolation motion, a circle is formed with the 3 given points, and the circumference is moved. (360 degrees)

(2) The posture during circular interpolation does not change.

(3) If the current position does not agree with the start point, the robot will automatically move to start point by linear interpolation.

(4) Once the execution has been interrupted and once jog operations have been restarted, it will start moving in relation to the position where it was interrupted, by linear interpolation.

**Reference Program**

10 MVC P1,P2,P3

20 MVC P1,J2,P3

30 MVC P1,P2,P3 WTH M_OUT(17)=1

40 MVC P3,(PLT 1,5),P4 WTHIF M_IN(20)=1,M_OUT(21)=1

**Reference**

Movement Commands, WTH, WTHIF, FINE, CNT, Movement Positions



# COSIMIR® · MELFA-BASIC IV

## MVR (Move R)

**Function:**

Carries out 3-dimensional circular interpolation movement from the start point to the end point via transit points.

**Format**

MVR<Start point>, <Transit point>, <End point> [<Interpolation Type>] [<Appended Condition>]

**Terminology**

<Start Point>

Start point for the circular arc. Describe a position operation expression or joint operation expression.

<Transit Point>

Transit point for the circular arc. Describe a position operation expression or joint operation expression.

<End Point>

End point for the circular arc. Describe a position operation expression or joint operation expression.

<Interpolation Type> Designate the interpolation type.

TYPE <Numerit constant 1>, <Numeric constant 2>

Numeric constant 1 ... Detour/short cut = 1/0

Numeric constant 2 ... 3-axis orthogonal/Equivalent rotation = 1/0

The default value is 0, 0 (detour, equivalent rotation).

<Appended Condition>

The WTH and WTHIF conjunctions.

**Explanation**

(1) In circular interpolation motion, a circle is formed with three given points, and robot moves along the circumference.

(2) The posture is interpolation from the start point to the end point; the transit point posture has no effect.

(3) If the current position and start point do not match, the robot will automatically move with linear interpolation (3-axis orthogonal interpolation) to the start point.

(4) If the execution is stopped, the robot is moved with jog, and then the execution is restarted, the robot will move with joint interpolation to the position where execution was stopped, and then the remaining circular interpolation will restart.

(5) If the start point and end point structure flags differ for an interpolation method other than 3-axis orthogonal interpolation, an error will occur at the execution.

(6) Of the three designated points, if any points coincide with the other, or if three points are on a straight line, linear interpolation will take place from the start point to the end point. An error will not occur.

(7) If 3-axis orthogonal is designated for the numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

(8) Numeric constant 2 designates the posture interpolation type. 3-axis orthogonal is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.

**Reference Program**

10 MVR P1,P2,P3

20 MVR P1,J2,P3

30 MVR P1,P2,P3 WTH M_OUT(17)=1

40 MVR P3,(PLT 1,5),P4 WTHIF M_IN(20)=1,M_OUT(21)=1

**Reference**

Movement Commands, WTH, WTHIF, FINE, CNT, Movement Positions



# COSIMIR® · MELFA-BASIC IV

## MVR2 (Move R2)

**Function:**

Carries out 3-dimensional circular interpolation motion from the start point to the end point on the arc composed of the start point, end point, and reference points.

The direction of movement is in a direction that does not pass through the reference points.

**Format**

MVR2<Start point>, <End point>, <Reference point> [<Interpolation Type>] [<Appended Condition>]

**Terminology**

<Start Point>

Start point for the circular arc. Describe a position operation expression or joint operation expression.

<End Point>

End point for the circular arc. Describe a position operation expression or joint operation expression.

<Reference Point>

Reference point for a circular arc. Describe a position operation expression or joint operation expression.

<Interpolation Type> Designate the interpolation type.

TYPE <Numerit constant 1>, <Numeric constant 2>

Numeric constant 1 ... Detour/short cut = 1/0

Numeric constant 2 ... 3-axis orthogonal/Equivalent rotation = 1/0

The default value is 0, 0 (detour, equivalent rotation).

<Appended Condition>

The WTH and WTHIF conjunctions.

**Explanation**

(1) In circular interpolation motion, a circle is formed with three given points, and robot moves along the circumference.

(2) The posture is interpolation from the start point to the end point; the transit point posture has no effect.

(3) If the current position and start point do not match, the robot will automatically move with linear interpolation (3-axis orthogonal interpolation) to the start point.

(4) If the execution is stopped, the robot is moved with jog, and then the execution is restarted, the robot will move with joint interpolation to the position where execution was stopped, and then the remaining circular interpolation will restart.

(5) The direction of movement is in a direction that does not pass through the reference points.

(6) If the start point and end point structure flags differ for an interpolation method other than 3-axis orthogonal interpolation, an error will occur at the execution.

(7) Of the three designated points, if any points coincide with the other, or if three points are on a straight line, linear interpolation will take place from the start point to the end point. An error will not occur.

(8) If 3-axis orthogonal is designated for the numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

(9) Numeric constant 2 designates the posture interpolation type. 3-axis orthogonal is used when carrying out interpolation on the (X, Y, Z, J4, J5, J6) coordinate system, and the robot is to move near a particular point.

**Reference Program**

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

20 MVR2 P1,J2,P3

30 MVR2 P1,P2,P3 WTH M_OUT(17)=1

40 MVR2 P3,(PLT 1,5),P4 WTHIF M_IN(20)=1,M_OUT(21)=1

**Reference**

Movement Commands, WTH, WTHIF, FINE, CNT, Movement Positions



# COSIMIR® · MELFA-BASIC IV

## MVR3 (Move R3)

**Function:**

Carries out 3-dimensional circular interpolation movement from the start point to the end point on the arc composed of the center point, start point and end point.

**Format**

MVR3<Start point>, <End point>, <Center point> [<Interpolation Type>] [<Appended Condition>]

**Terminology**

<Center Point>

Center point for the arc. Describe a position operation expression or joint operation expression.

<Start Point>

Start point for the arc. Describe a position operation expression or joint operation expression.

<End Point>

End point for the circular arc. Describe a position operation expression or joint operation expression.

<Interpolation Type> Designate the interpolation type.

TYPE <Numerit constant 1>, <Numeric constant 2>

Numeric constant 1 ... Detour/short cut = 1/0

Numeric constant 2 ... 3-axis orthogonal/Equivalent rotation = 1/0

The default value is 0, 0 (detour, equivalent rotation).

<Appended Condition>

The WTH and WTHIF conjunctions.

**Explanation**

(1) In circular interpolation motion, a circle is formed with three given points, and robot moves along the circumference.

(2) The posture is interpolation from the start point to the end point; the transit point posture has no effect.

(3) If the current position and start point do not match, the robot will automatically move with linear interpolation (3-axis orthogonal interpolation) to the start point.

(4) If the execution is stopped, the robot is moved with jog, and then the execution is restarted, the robot will move with joint interpolation to the position where execution was stopped, and then the remaining circular interpolation will restart.

(5) If the start point and end point structure flags differ for an interpolation method other than 3-axis orthogonal interpolation, an error will occur at the execution.

(6) Of the three designated points, if any points coincide with the other, or if three points are on a straight line, linear interpolation will take place from the start point to the end point. An error will not occur.

(7) If 3-axis orthogonal is designated for the numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

(8) The fan angle from the start point to the end point is 0 < fan angle < 180 deg.

(9) Designate the positions so that the difference from the center point to the end point and the center point to the distance is within 0.01 mm.

(10) If the three points are on the same line, or if the start point and center point, or end point and center point are the same, an error will occur.

(11) If the start point and end point are the same, or if the three points are the same, the robot will move with linear interpolation from the start point to the end point.

**Reference Program**

10 MVR3 P1,P2,P3

20 MVR3 P1,J2,P3

30 MVR3 P1,P2,P3 WTH M_OUT(17)=1

40 MVR3 P3,(PLT 1,5),P4 WTHIF M_IN(20)=1,M_OUT(21)=1

**Reference**

Movement Commands, WTH, WTHIF, FINE, CNT, Movement Positions



# COSIMIR® · MELFA-BASIC IV

## MVS (Move S)

**Function:**

Carries out linear interpolation movement from the current position to the movement target position.

**Format**

MVS<Movement Target Position> [, <Close distance>] [<Interpolation Type>] [<Appended Condition>]

**Format 2**

MVS<Separation Distance> [<Interpolation Type>]

**Terminology**

<Movement Target Position>

The final position for the linear interpolation. Describe a movement position statement

<Close Distance>

If this value is designated, the actual movement target position will be a position separated by the designated distance in the tool coordinate system Z axis direction {+/- direction).

<Interpolation Type>

Designate the interpolation type.

TYPE <Numeric constant 1>, <Numeric constant 2>

Numeric constant 1 ... Detour/short cut = 1/0

Numeric constant 2 ... 3-axis orthogonal/Equivalent rotation = 1/0

The default value is 0, 0 {detour, equivalent rotation).

<Appended conditions>

The WTH and WTHIF statements can be used.

<Separation Distance>

If this value is designated, the robot will move from the current position to a position separated by the designated amount in the tool coordinate system Z axis direction. When a positive value is set, the movement will take place in the direction that the mechanical interface {flange surface) is facing, and when a negative value is set. the movement will take place in the opposite direction {hand retract).

**Explanation**

(1) Linear interpolation motion is a type of movement where the robot moves from its current position to the movement target position so that the locus of the control points is in a straight line.

(2) The posture is interpolation from the start point to the end point.

(3) If the execution is stopped, the robot is moved with jog, and then the execution is restarted, the robot will move with joint interpolation to the position where execution was stopped, and then will move to the target position.

(4) Note that when resuming a command for the target position, such as MVS, 100, the robot will not move the remaining distance.

(5) If the start point and end point structure flags differ for an interpolation method other than 3-axis orthogonal interpolation, an error will occur at the execution.

(6) Of the three designated points, if any points coincide with the other, or if three points are on a straight line, linear interpolation will take place from the start point to the end point. An error will not occur.

(7) If 3-axis orthogonal is designated for the numeric constant 2, the numeric constant 1 will be invalidated, and the robot will move with the taught posture.

**Reference Program**

```
10 MVS PLT1,10,100.0 WTH M_OUT(17)=1
```

20 MVS P4+P5,50.0 WTHIF M_IN(18)=1,M_OUT(20)=1

30 MVS ,50

**Reference**

Movement Commands, WTH, WTHIF, FINE, CNT, Movement Positions



# COSIMIR® · MELFA-BASIC IV

## OADL (Optimal Acceleration)

**Function:**

Automatically sets the optimum acceleration/deceleration time according to the designated robot hand's load state.

**Format**

OADL<Switch>

**Terminology**

<Switch>

Set whether to turn optimization ON or OFF.

**Explanation**

(1) The robot moves with the optimum acceleration/deceleration according to the hand conditions and workpiece conditions designated with the LOADSET command.

(2) The workpiece grasp/not grasp for when the hand is opened or closed is set with parameter HNDHOLD 1 to 8.

(3) The ACCEL command is invalid when OADL is ON.

(4) OADL is set to OFF as the default.

(5) Once OADL is ON, it is valid until OADL OFF is executed or until the program END is executed.

**Reference Program**

10 OADL ON

20 MOV P1' Move with maximum load.

30 LOADSET 1,1' Set hand 1 and workpiece 1.

40 MOV P2' Move with hand 1 + workpiece 1 load.

50 HOPEN 1

60 MOV P3' Move with hand 1 load.

70 HCLOSE 1

70 HOPEN P4' Move with hand 1 + workpiece 1 load.

80 LOADSET 1,2' Set hand 1 and workpiece 2.

90 MOV P5' Move with hand 1 load.

100 HCLOSE 1

110 MOV P6' Move with hand 1 + workpiece 2 load

*When parameter HNDHOLD1 is set to 0, 1



# COSIMIR® · MELFA-BASIC IV

### ON COM GOSUB (On Communication Go Subroutine)

**Function:**

Defines the starting line of a branching subroutine when an interrupt is generated from a designated communication line.

**Format**

ON COM[(<File No.>)] GOSUB <Call destination>

**Terminology**

<File No.>

Describe a number between 1 and 3 assigned to the communication line.

<Call Destination>

Describe the line No. and label name.

**Explanation**

(1) If the file No. is omitted, 1 will be used as the file No.

(2) The file Nos. with the smallest No. have the order of priority for the interrupt.

(3) If a communication interrupt is generated while the robot is moving. the robot will stop.

(4) It is possible to use COM STOP to stop the interrupt. and prevent the robot from stopping.

(5) In the default state, the interrupt is disabled. Execute the COM ON command after this command to validate the interrupt.

**Reference Program**

```
10 OPEN "COM1:" AS #1

20 OPEN "COM3:" AS #2

30 ON COM GOSUB 300' If an interrupt is generated from

' the file No.1 communication line

' (COM1:), carry out line 300 process.

40 ON COM(2) GOSUB *RECV' If an interrupt is generated from the

' communication line No.1 communication

' line, carry out the label RECV process.

50 COM(1) ON' Enable interrupt from file No.1 communication line.

60 COM(2) ON' Enable interrupt from file No.2 communication line.

70 COM OFF' Disable interrupt from file No.1 communication line.

80 COM(2) OFF' Disable interrupt from file No.2 communication line.
```

**Reference**

COM ON, COM OFF, COM STOP, OPEN

## ON GOSUB (On Gosub)

**Function:**

Calls up the subroutine at the line No. or label corresponding to the value.

**Format**

ON<Terminology> GOSUB [<Expression>] [, [<Call destination>]] ...

**Terminology**

<Terminology>

Designate the line No. or label on the nth line to branch to with a numeric operation expression.

<Call Destination>

Describe the line No. or the label No.

**Explanation**

(1) The value of <Expression> determines which line No. or label subroutine to call. For example, if the value of <Expression> is 2, the line No. or label described for the second value is called.

(2) When a line No. or label that is called up does not exist, or when there are two definitions, an error will occur.

| Value of <Expression> | Process <Control> |
|---|---|
| Real number | Value is converted to an Integer by rounding it off, and then branching is executed. |
| When 0, or when the value exceeds the number of line Nos. or labels | Control proceeds the next line |
| Negative number or 32767 is exceeded | Execution Error |
| Line number or label is omitted | Execution Error |

**Reference Program**

100 ON M1 GOSUB 1000, *SUBPR' Call line 1000 when numeric variable MI

' value is 1, and the label *SUBPR line when 2.

**Reference**

RETURN, END, ON GOTO, GOSUB, IF THEN ELSE

# COSIMIR® · MELFA-BASIC IV

## ON GOTO (On go to)

**Function:**

Branches to the line with the line No. or label that corresponds to a designated value.

**Format**

ON<Expression> GOTO [<Branch destination>] [, [<Branch destination>]] ...

**Terminology**

<Expression>

Designate the line No. or label on the nth line to branch to with a numeric operation expression.

<Call Destination>

Describe the line No. or the label No.

**Explanation**

1. This is the GOTO version of ON GOSUB.

**Reference Program**

100 ON M1 GOTO 1000.*SUBPR' Branch to line 1000 when numeric variable M1

' value is 1; and the label *SUBPR line when 2.

**Reference**

GOTO, ON GOSUB, IF THEN ELSE



# COSIMIR® · MELFA-BASIC IV

## OPEN (Open)

**Function:**

Opens a file.

**Format**

OPEN "<File descriptor>" [FOR <Mode>] AS [#] <File No.>

**Terminology**

<File Descriptor>

Describe a file name (including communication lines).

To open a communication line, set "<Communication Line File Name>:"

When not using a communications line, set "<File Name>"

<Mode>

Designate the method to access a file.

Omitted = random mode

This can be omitted when using a communication line.

INPUT = input mode

Inputs from an existing file.

OUTPUT = output mode (new file)

Creates a new file and outputs it there.

APPEND = Output mode (existing file)

Appends output to the end of an existing file.

| File Descriptor | File name | Access method |
|---|---|---|
| File | Describe with 16 characters or less | INPUT, PRINT, APPEND |
| Communication line file | COM 1:, COM 2:, COM 3: | Omitted = random mode only |

<File No.> Describe a constant within the following range.

To interrupt from communication line: 1 to 3.

To not interrupt from communication line: 1 to 8.

**Explanation**

(1) Designate the designated file No. of the file described in <File Descriptor>, and open
Use this file No. when reading from or writing to the file.

(2) A communication line is handled as a file.

| Communication file | Hardware device name |
|---|---|
| COM 1 | Standard RS-232-C |
| COM 2 | (Reserved) |
| COM 3 | (Reserved) |

**Reference Program**

```
10 OPEN "COM1:" AS #1' Open standard RS-232-C line as file No. 1.

20 MOV P_01‘ Move to position P_01

30 PRINT #1,P_CURR' Output current position to external source.

' "(100.00,200.00,300.00,400.00)(7.)" format

40 INPUT #1,M1,M2,M3' Send from external source with "101.00,202.00,303.00" ASCI
format.

50 P_01.X=M1

60 P_01.Y=M2

70 P_01.C=M3' Copy to global data.

80 CLOSE' Close all opened files.

90 END
```

**Reference**

INPUT, PRINT, COM ON, COM OFF, COM STOP, ON COM GOSUB


# COSIMIR® · MELFA-BASIC IV

## OVRD (Override)

**Function:**

Designates the override applied on the entire program.

**Format**

OVRD <Designated override>

**Terminology**

<Designated override>

Designate the override with a real number. Unit: [%] (Recommended range: 1 to 100.0)

A numeric operation expression can also be described. If 0 or a value over 100 is set, an error will occur.

**Explanation**

(1) The OVRD command is valid regardless of the interpolation type.

(2) The actual override is as follows:

During joint interpolation: (Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Joint override (JOVRD command)).

During Iinear interpolation: (Operation panel (T/B) override setting value) x (Program override (OVRD command)) x (Linear designated speed (SPD command)).

(3) The OVRD command changes only the program override. 100% is the maximum capacity of the robot. Normally, the system default value (M_NOVRD) is set to 100%. The designated override is the system default value until the OVRD command is executed in the program.

(4) Once the OVRD command has been executed, the designated override is applied until the next OVRD command is executed, the program END is executed or until the program is reset. The value will return to the default value when the END statement is executed or the program is reset.

**Reference Program**

```
10 OVRD 50

20 MOV P1

30 MVS P2

40 OVRD M_NOVRD' Set the default value
```

**Reference**

JOVRD, SPD, Movement commands, Robot status variables

# COSIMIR® · MELFA-BASIC IV

## PLT (Pallet)

**Function:**

Calculates the position of grid in the pallet.

**Format**

PLT<Pallet No.> <Numeric operation expression>

**Terminology**

<Pallet No.>

Select a pallet No. between 1 and 8 that has already been defined with a DEF PLT command.

<Numeric Operation Expression>

Sets the grid No. for the selected pallet.

**Explanation**

(1) The position of grid of a pallet defined by the DEF PLT statement is operated.

(2) The pallet Nos. are from 1 to 8, and up to 8 can be defined at once.

(3) Note that the position of the grid may vary because of the designated direction in the pallet definition.

(4) If a grid No. is designated that exceeds the largest grid No. defined in the pallet definition statement, an error will occur during execution.

(5) When using the pallet grid point as the target position of the movement command, an error will occur if the point is not enclosed in parentheses as shown above.

**Reference Program**

```
10 M1=0

20 DEF PLT 1,P10,P11,P12,P13,8,8,1

30 M1=M1+1

40 P1=PLT 1,M1

50 MOV P1,-50
```

60 MVS P1

70 MOV (PLT 1,5) TYPE 1,1

**Reference**

DEF PLT, Movement Command



# COSIMIR® · MELFA-BASIC IV

## PRINT (Print)

**Function:**

Outputs data into a file (or output device)

**Format**

PRINT#<File No.> [, [<Expression>;] ... [<Expression> [;]]]

**Terminology**

<File No.>

Described with numbers 1 to 8.

Corresponds to the control No. assigned by the OPEN command.

<Expression>

Describes numeric operation expressions, position operation expressions and character string expressions.

**Explanation**

(1) If <Expression> is not described, then a carriage return will be output.

(2) If <File No.> is omitted, the default value of 1 will be used.

(3) Output format of data (reference)

The output space for the value for <Expression> and for the character string is in units of 10 characters. When outputting multiple values, use a comma between each <Expression> as a delimiter.

If a semicolon (;) is used at the head of each space unit, it will output after

the item that was last displayed.

(4) The carriage will always be returned after the OPEN statement.

**Example**

10 M1=123.5

20 P1=(130.5,-117.2,55.1,16.2,0.0,0.0)(1,0)

1.)

30 PRINT# 1,"OUTPUT TEST",M1,P1 is described,

OUTPUT TEST 123.5 (130.5,-117.2,55.1,16.2,0.0)(1,0) is output.

2.)

30 PRINT# 1,"OUTPUT TEST";M1' P1 is described,

OUTPUT TEST 123.5(130.5,-117.2,55.1,16.2,0.0)(1,0) is output.

If a comma or semicolon is inserted after a <Expression>, the carriage return will not be issued, and instead, printing will continue on the same line.

3)

30 PRINT# 1,"OUTPUT TEST",

40 PRINT# 1,M1,

50 PRINT# 1,P1 is described,

OUTPUT TEST 123.5(130.5,-117.2,55.1,16.2)(1,0) is output.

**Reference Program**

10 OPEN "COM1:" AS #1' Open .standard RS-232-C line as file No. 1.20
MOV P_01

20 MDATA=150' Substitute 150 for the numeric variable MDATA.

30 PRINT #1,"***PRINT TEST***"' Outputs the character string "***PRINT TEST****".

40 PRINT #1' Issue a carriage return

50 PRINT #1, "MDATA=",MDATA' Output the character string "MDATA" followed by the value of MDATA, (150).

60 PRINT #1' Issue a carriage return

40 PRINT #1,"**********"' Outputs the character string "**********".

50 END' End the program.

The output result is shown below.

***PRINT TEST***

MDATA=150

**********

**Reference**

OPEN, INPUT



# COSIMIR® · MELFA-BASIC IV

**RELM (Release Mechanism)**

**Function:**

Releases the mechanism resource.

**Format**

RELM

**Explanation**

(1) Releases the currently acquired mechanism resource.

(2) If an interrupt is applied while the mechanism is acquired and the program execution is stopped, the acquired mechanism resource will be automatically released.



# COSIMIR® · MELFA-BASIC IV

**REM (Remarks)**

**Function:**

Uses the following character strings as comments.

**Format**

REM[<Comment>]

**Terminology**

<Comment>

Describe a user-selected character string.

Descriptions can be made in the range of position lines.

**Explanation**

(1) REM can be abbreviated to be a single quotation mark (').

(2) This can also be after another command on the same line.

**Reference Program**

10 REM ***MAIN PROGRAM***

20 ' ***MAIN PROGRAM***

30 MOV P1' Move to P1



# COSIMIR® · MELFA-BASIC IV

## RETURN (Return)

**Function:**

When returning from a normal subroutine returns to the next line after the GOSUB.

When returning from an interrupt processing subroutine, returns either to the line where the interrupt was generated, or to the next line.

**Format**

When returning from a normal subroutine:

RETURN

When returning from an interrupt processing subroutine:

RETURN <Return designation No.>

**Terminology**

<Return Designation No.>

Designate the line where control will return to after an interrupt has been generated and processed.

0 ... Return control to the line where the interrupt was generated.

1 ... Return control to the next line after the line where the interrupt was issued.

**Explanation**

(1) When there is a RETURN command in a normal subroutine with a return-to designation number, and when there is a RETURN command in an interrupt-processing subroutine with no return-to destination number, an error will occur.

**Reference**

GOSUB, ON GOSUB, END



# COSIMIR® · MELFA-BASIC IV

## SELECT CASE (Select Case)

**Function:**

Executes one of multiple statement blocks according to the condition expression value.

**Format**

SELECT <Condition>
CASE <Expression>
[<Process>]
BREAK
CASE <Expression>
[<Process>]
BREAK
...

CASE <Expression>
[<Process>]
BREAK
DEFAULT
[<Process>]
BREAK
END SELECT

**Terminology**

<Condition>
Describe a numeric operation expression.

<Expression>
Describe a numeric operation expression. The type must be the same as the condition expression.

<Process>
Describe a command statement (excluding a branch condition statement or repeated control statement) provided with MELFA-BASICI V, a line No. or a label name.

**Explanation**

(1) If the condition matches one of the CASE items, the process will be executed until the next CASE, DEFAULT or ENDSELECT.

(2) If the case does not match with any of the CASE items but DEFAULT is described, that block will be executed.

(3) If there is no DEFAULT, the program will jump to the line after ENDSELECT without processing.

(4) The SELECT CASE and END SELECT statements must always correspond.

(5) If an END SELECT statement that does not correspond to SELECT CASE is executed, an execution error will occur. Note that if the END statement is executed midway, or if the last line of the program is executed, the program execution will stop at that line.

(6) Another SELECT CASE cannot be described in the SELECT CASE. However, WHILE WEND or FOR NEXT can be described.

**Reference Program**

```
10 SELECT MCNT

20 M1=10' This line is not executed

30 CASE IS <= 10' MCNT <= 10

40 MOV P1
```

```
50 BREAK
60 CASE 11' MCNT = 11
70 MOV P2
80 BREAK
90 CASE 12' MCNT = 12
100 MOV P3
110 BREAK
120 CASE 13 TO 18' 13 <= MCNT <= 18
130 MOV P4
140 DEFAULT' Other than the above
150 M_OUT(10)=1
160 BREAK
170 END SELECT
```



# COSIMIR® · MELFA-BASIC IV

**SERVO (Servo)**

**Function:**

Controls the ON and OFF of the servo.

**Format**

SERVO ON

SERVO OFF

**Explanation**

(1) The robot arm controls the servo power for all axes.

**Reference Program**

```
10 SERVO ON                    ' Servo ON
```

```
20 IF M_SVO<>0 THEN GOTO 20   ' Wait for servo ON
30 SPD M_NSPD
```



# COSIMIR® · MELFA-BASIC IV

**SKIP (Skip)**

**Function:**

Transfers control of the program to the next line.

**Format**

SKIP

**Explanation**

(1) This command can be described with the WTH or with WTHIF statements. In this case, the execution of that line is interrupted, and control is automatically transferred to the next line. Execution of skip can be seen with the M_SKIPCQ information.

**Reference Program**

```
20 MOV P1 WTHIF M_IN17=0,SKIP
' If the input signal (M_IN(17)) turns ON while moving with joint
interpolation to the position indicated with position variable P1,
stop the robot interpolation motion, and stop execution of this
command.
```

**Reference**

HLT, WTH, WTHIF



# COSIMIR® · MELFA-BASIC IV

## SPD (Speed)

**Function:**

Designates the speed for the robot's linear and circular movements.

**Format**

SPD<Designated speed>

**Terminology**

<Designated Speed>

Designate the speed as a real number. Unit: [mm/s]

**Explanation**

(1) The SPD command is valid only for the robot's linear and circular movements.

(2) The actual designated override is (Operation panel (T/B) override setting value) x (Program override (OVRDcommand)) x (Linear designated speed (SPD command)).

(3) The SPD command changes onIy the linear designated speed.

(4) When M_NSPD is designated for the designated speed, the robot will always move at the maximum possible speed, so the line speed will not be constant.

(5) The system default value is applied for the designated speed until the SPD command is executed in the program. Once the SPD command is executed, that designated speed is held until the next SPD command.

(6) The designated speed will return to the system default value when the program END statement is executed.

**Reference Program**

```
10 SPD 100

20 MVS P1

30 SPD M_NSPD' Set the default value
```

**Reference**

ACCEL, CNT, OVRD, JOVRD, Movement command, Robot status variable

# COSIMIR® · MELFA-BASIC IV

## Substitute

**Function:**

The results of an operation are substituted in a variable or array variable.

**Format 1**

<Variable Name> = <Expression 1>

**Format 2 (For pulse substitution)**

<Variable Name> = <Expression 1> DLY <Expression 2>

**Terminology**

<Variable Name>

Describe the name of the variable where the value is to be substituted.

(Refer to the syntax diagram for the types of variables.)

<Expression 1>

Substitution value. Describe an arithmetic operation expression.

<Expression 2>

Pulse timer. Describe an arithmetic operation expression.

**Explanation**

(1) When using this additionally for the pulse output, the pulse will be executed in parallel with the execution of the commands on the following lines.

(2) After the designated time has passed, the state will return to that before the pulse output.

(3) If the END command or program's last line is executed during the designated time, or if the program execution is stopped due to an emergency stop, etc., the output state will be held. (The output state during the interrupt can be selected with the system parameters if it is to be return to the state before the pulse output in the same manner as after the designated time has passed.)

**Reference Program**

10 P100=P1+P2*2

20 M_OUT(17)=1 DLY 10.0

**Reference**

DLY, Syntax diagram



# COSIMIR® · MELFA-BASIC IV

## TOOL (Tool)

**Function:**

Designates the tool conversion data.

**Format**

TOOL<Tool conversion data>

**Explanation**

(1) The system default value (P_NTOOL) is applied until the TOOL command is executed. Once the TOOL command is executed, the designated tool conversion data is applied until the next TOOL command is executed. This is operated with 6-axis three-dimension regardless of the mechanism structure.

(2) The tool conversion data changed with the TOOL command is saved in parameter MEXTL, and is saved even after the controller power is turned OFF.

**Reference Program**

10TOOL (100,100,0,0,0,0)

20MVS P1

30TOOL P_NTOOL

# COSIMIR® · MELFA-BASIC IV

## TORQ (Torque)

**Function:**

Designates the torque limit for each axis.

**Format**

TOOL<Axis no.>, <Numeric value>

**Terminology**

<Axis no.>

Designate the axis No., with a numeric constant. (1 to 8)

<Numeric value>

Designate the limit of the force generated from the axis as a percentage. (1 to 100)

**Explanation**

(1) The torque limit value of the designated axis is limited. The movement is limited so that a torque exceeding the designation is not applied. Designate a percentage in respect to the standard torque limit.

(2) This is used when the grasping force is to be limited for the servo hand, etc.

**Reference Program**

10TORQ 4,80

20MVS P1



# COSIMIR® · MELFA-BASIC IV

## WAIT (Wait)

**Function:**

Waits for the variable to reach the designated value.

**Format**

WAIT<Numeric value> = <Numeric constant>

**Terminology**

<Numeric value>

Designate an M variable.

<Numeric constant>

Designate a numeric constant.

**Explanation**

(1) This command is used as the interlock during signal input wait and during multitask execution.

**Reference Program**

10WAIT M_IN(1) = 1

20WAIT M_IN(3) = 0

30WAIT M_RUN(2) = 1

40WAIT M_01 = 100



# COSIMIR® · MELFA-BASIC IV

### WHILE WEND (While End)

**Function:**

The program between the WHILE statement and WEND statement is repeated until the loop conditions are satisfied.

**Format**

WHILE<Loop condition>
.
.
WEND

**Terminology**

<Loop Condition>

Describe a numeric operation expression. (Refer to the syntax diagram)

**Explanation**

(1) The program between the WHILE statement and WEND statement is repeated.

(2) If the result of <Expression> is true (not 0), then the control moves to the line following the WHILE statement and the process is repeated.

(3) If the result of <Expression> is false (is 0), then the control moves to the line following the WEND statement.

**Reference Program**

```
20 WHILE (M1>=-5) AND (M1<=5) ' Repeat the process while the numeric variable M1

' value is between -5 and +5, and transfer control to

' line after WEND statement if range is exceeded.

30 M1=-(M1+1)' Add 1 to M1, and reverse the sign

40 PRINT# 1, M1' Output the M1 value

50 WEND ' Return to the WHILE statement (line 20)

60 END' End the program
```



# COSIMIR® · MELFA-BASIC IV

## WTH (With)

**Function:**

A process is added to the interpolation motion.

**Format**

WTH<Process>

**Terminology**

<Process>

Describe the process to be added. The commands that can be described are as follow.

1. <Numeric type data B> <Substitution operator><Numeric type data A>

[Substitute, signal modifier command (refer to syntax diagram)]

2. HLT statement

3. SKIP statement

**Explanation**

(1) This command can only be used to describe the additional command for the movement command.

(2) An error will occur if the WTH command is used alone.

(3) The process will be executed simultaneously with the start of motion.

(4) The relationship between the interrupts regarding the priority order is shown below.

COM > ACT > WTHIF (WTH) > Pulse substitution

**Reference Program**

10 MOV P1 WTH M_OUT(17)=1 DLY M1+2

**Reference**

Movement Command, WTHIF, Substitute, Syntax Diagram



# COSIMIR® · MELFA-BASIC IV

## WTHIF (With If)

**Function:**

A process is conditionally added to the interpolation motion command.

**Format**

WTHIF<Additional command>, <Process>

**Terminology**

<Additional Command>

Describe the condition for adding the process. (Same as ACT condition expression)

<Process>

Describe the process to be added with the additional conditions are established. (Same as WTH)

The commands that can be described as a process are as follow. (Refer to syntax diagram.)

1. <Numeric type data B> <Substitution operator><Numeric type data A>

2. HLT statement

3. SKIP statement

**Explanation**

(1) This command can only be used to describe the additional command with conditions for the movement command.

(2) Monitoring of the condition will start simultaneously with the start of movement.

**Reference Program**

10 MOV P1 WTHIF M_IN(17)=1, HLT

20 MVS P2 WTHIF M_RSPD>200,M_OUT(17)=1 DLY M1+2

30 MVS P3 WTHIF M_MRATIO>15, M_OUT(1)=1

**Reference**

Movement Command, WTH, Substitute, Syntax Diagram

Substitute



# COSIMIR® · MELFA-BASIC IV

## XLOAD (X Load)

**Function:**

Select a designated program for a designated slot.

**Format**

XLOAD<Slot no.>, <Program name>

**Terminology**

<Slot no.>

Designate the slot no.

<Program name>

Designate the program name.

**Explanation**

(1) If the designated program is already selected for another slot, an error will occur at execution.

(2) If the designated program is being edited, an error will occur at execution.

(3) Designate the program name in double quotations. " ".

**Reference Program**

10XLOAD 2, "10"' Select program 10 for slot 2

20XRUN 2' Start slot 2

30WAIT M_RUN(2)=1' Wait to confirm starting of slot 2



# COSIMIR® · MELFA-BASIC IV

## XRST (X Reset)

**Function:**

The execution line of the program in the designated slot is returned to the head line.

**Format**

XRST<Slot no.>

**Terminology**

<Slot no.>

Designate the slot no.

**Explanation**

(1) This is valid only when the slot is in the stopped state.

**Reference Program**

```
10XRUN 2' Start
20WAIT M_RUN(1)=1' Wait for start to complete
...
100XSTP 2' Stop
110WAIT M_WAIT(1)=1' Wait for stop to complete
...
150XRST 2' Set program execution start line to head line.
160WAIT M_PSA(1)=1' Wait for program reset to complete
...
200XRUN 2' Restart
210WAIT M_RUN(1)=1' Wait for restart to complete
```



# COSIMIR® · MELFA-BASIC IV

**XRUN (X Run)**

**Function:**

Runs the designated program in parallel.

**Format**

XRUN<Slot no.>, <Program name> [,<Operation Mode>]

**Terminology**

<Slot no.>

Designate the slot no.

<Program name>

Designate the program name.

<Operation Mode>

0 = Continuous operation

1 = Cycle stop operation

**Explanation**

(1) If the designated slot no. Is already in use, an error will occur at execution.

(2) If XRUN is executed in the "waiting" state with the program stopped midway, continuous execution will start.

(3) Designate the program name in double quotations. " ".

(4) If the operation mode is omitted, the current operation mode will be used.

**Reference Program**

10XRUN 1,"1"' Run program 1 with slot 1.

30XRUN 3,"2",1' Restart program 2 with slot 3

' in the cycle operation mode.



# COSIMIR® · MELFA-BASIC IV

## XSTP (X Stop)

**Function:**

Operation of the program in the designated slot is stopped.

**Format**

XSTP<Slot no.>

**Terminology**

<Slot no.>

Designate the slot no.

**Explanation**

(1) If the program is already stopped, an error will not occur.

(2) XSTP can also stop the constant execution attribute program.

**Reference Program**

```
10XRUN 2
...
100XSTP 2' Stop
110WAIT M_WAI(1)=1' Wait for stop to complete
...
200XRUN 2' Restart
```