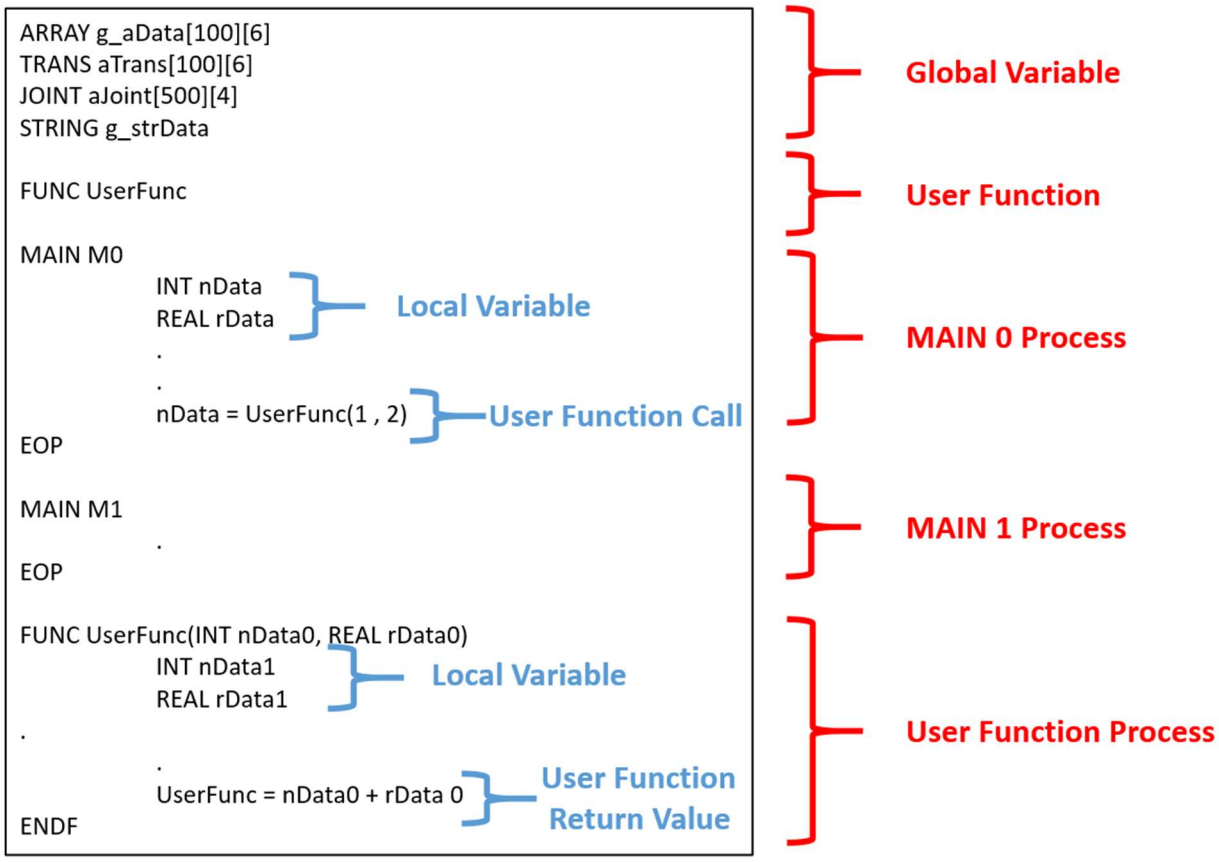
VAS CORPORATION

Robot Script Manual

CONTENT

XMC-ROBOT Script Structure

XMC-ROBOT Script Structure is shown below.



Script Command Quick List

Motion Instruction

Absolute Movement

Command	Description
LMOVE	Linear interpolation from the current position to the target position.
AMOVE	Arc move from the current position to the target point by pasing waypoint.
CMOVE	Circular interpolation passing 3 points from the current position by passing 2 way-points.
JMOVE	Joint interpolation from the current position to the target position
HOME	Joint interpolation from the current position to setting Home position

Relative Movement

Command	Description
INCJ	Incremental Joint Axis move.
INCL	Incremental Base Axis move.
INCT	Incremental Tool Axis move.

Conveyor Tracking

Command	Description
TKOBJWAIT	Wait until the object comes in the range of working.
TKMOVE	Move linear interpolated motion tracking the conveyor.
TKOBJCOUNT	Return the number of objects in the queue.
TKAPPROACH	Approach the object along the Z axis of conveyor frame while tracking motion
TKDEPART	Depart the object along the Z axis of conveyor frame while tracking motion

Motion stop

Command	Description
MSSTOP	It decelerates and stops motion being driven.
MESTOP	Suddenly stop the motion being driven.

Control Statement

Task

Command	Description
MAIN	Set task start.
EOP	Set task termination.
EXIT	Set task stop.

Task Control

Command	Description
TRUN	Run the task.
TSTOP	Stop the task.

User function

Command	Description
FUNC	Set user function startup.
ENDF	Set user function exit.
EXITF	Set user function stop.

Label/Branch

Command	Description
LABEL	Set the branch point position.
GOTO	Move program counter to a label.

Control instruction

Command	Description
IF THEN	Determines true or false of the conditional expression, and then execute the next instruction.
ELSE	Executes the instruction when the conditional expression is false.
ENDIF	Set the end of IF statement.
SWITCH	Determines the next instruction to perform.
CASE	Adds a branch of the next instruction for the SWITCH instruction.
DEFAULT	Determines a location to execute the instruction when the instruction does not corresponded to the any branches of the SWITCH.
ENDSW	Sets the end of multiple branches.

Step repeat

Command	Description
FOR TO STEP	Executes the repetition of the series of the instructions for specified number of times within a range.
NEXT	Returns to next step repetition time.

Conditional repeat

Command	Description
WHILE	Determines true or false of the added conditional expression, and execute the series of instructions within a range when it is true.
ENDWL	Sets the ends of WHILE instruction.
DO	Starts to execute LOOP statement.
LOOP	Ends LOOP statement.

Setting

Variable

Command	Description
INT	Declares an integer variable.
REAL	Declares a real variable.
ARRAY	Declares an array variable.
STRING	Declares a string variable.
TRANS	Declares a point array in World Coordinate.
JOINT	Declares a point array in Joint Coordinate.
POINT	Defines value of a point.

Speed

Command	Description
SPEED	Set percentage (%) of speed.
ACCEL	Set percentage (%) of acceleration.
DECEL	Set percentage (%) of deceleration.

Delay

Command	Description
DELAY	Program execution is delayed by the set time (ms).

Check Status

Axis Position

Command	Description
CMDPOS	Returns the command position of axis.
ACTPOS	Returns the actual position of axis.

Digital In/Out

Digital Input

Command	Description
WAITDIN	Waits until the signal conditions meet.
DIN	Reads the value of digital signal

Digital Output

Command	Description
DOUT	Turns ON/OFF a digital output signal.
DOUTGROUP	Sets value for a digital output signal group.
PULSEDOUT	Creates a pulse on a digital output signal.
DELAYDOUT	Delays digital output signal.
RESETDOUT	Resets all digital output signals.

Operator

Array operator

Command	Description
[	Array operator
]	Array operator

Binary operator

Command	Description
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Modulus operator

Preferred operator

Command	Description
(	Left bracket
)	Right bracket

Assignment operator

Command	Description
=	Asigns the right value to the left variable.

Logical operator

Command	Description
&&	AND operator.
||	OR operator.
!	NOT operator.

Comparision operator

Command	Description
>	Greater than comparision operator
<	Less then comparision operator
>=	Greater equal than comparision operator
<=	Less equal than comparision operator
!=	Difference comparision operator
==	Equal comparision operator

Bitwise operator

Command	Description
&	Bitwise AND operator
|	Bitwise OR operator
^	Bitwise XOR operator
~	Bitwise NOT operator
<<	Left shift operator
>>	Right shift operator

Script Command Function List

Motion Instruction

This section introduces the script commands required to move motion.

Absolute movement

Command	Description
LMOVE	Linear interpolation from current position to target point.
AMOVE	From the current position, moves in a circular arch to the target point through the transit point.
CMOVE	From current position, circular interpolation to current position by passing through point 1 and point 2.
JMOVE	Joint interpolation from current position to target position.
HOME	Joint interpolation from current position to preset Home position.

Relative movement

Command	Description
INCJ	Incremental Joint Axis movement.
INCL	Incremental Base Axis movement.
INCT	Incremental Tool Axis movement.

Conveyor Tracking

Command	Description
TKOBJWAIT	Wait until the object comes in the range of working.
TKMOVE	Move linear interpolated motion tracking the conveyor.
TKOBJCOUNT	Return the number of objects in the queue.

Motion stop

Command	Description
MSSTOP	It decelerates and stops motion being driven.
MESTOP	Suddenly stop the motion being driven.

JMOVE

Function

Moves to the teaching position by joint interpolation.

Format

JMOVE #P[No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]#P[No.]	Teaching position in joint space.

Example

```
MAIN  M0
SPEED 50
JMOVE #P[0]
EOP
```

Move from the manipulator's waiting position to Step 1. Move by joint interpolation at a speed of 50%.
The position in Step 1 is registered to the #P variable no. 0.
The path during movement is not specified. Be careful of interference.

LMOVE

Function

Moves to the teaching position by linear interpolation.

Format

LMOVE P[No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]P[No.]	Teaching position in World Space.

Example

```
MAIN M0
SPEED 35
JMOVE #P[0]   // Step 1
JMOVE #P[1]   // Step 2
SPEED 50
LMOVE P[0]    // Step 3
EOP
```

Manipulator moves from Step 2 to Step 3 by the linear interpolation at a speed of 50% of Max Linear Speed.

AMOVE

Function

Moves to the teaching position through the teaching mid-position by arch interpolation. In the arch motion, manipulator's orientation is not changed.

Format

AMOVE P[Mid-position No.], P[Destination Position No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]P[Mid-position No.]	Teaching mid-position in World Space.
[>>]P[Destination position No.]	Teaching destination position in World Space.

Example

```
MAIN M0
LMOVE P[0]
SPEED 20
AMOVE P[1], P[2]
EOP
```

Moves to teaching position P[0] in Step 1.

Implements arch interpolation from the current position with the teaching points P[1], P[2] at a speed of 20% of Max Linear Speed.

CMOVE

Function

Moves to the teaching position by circular interpolation. The path passes through 2 teaching mid-positions then returns to the current position. In the circular motion, the manipulator's orientation is not changed.

Format

CMOVE P[Mid-position 1 No.], P[Mid-position 2 No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]P[Mid-position 1 No.]	Teaching mid-position 1 in World Space.
[>>]P[Mid-position 2 No.]	Teaching mid-position 2 in World Space.

Example

```
MAIN M0
LMOVE P[0]
SPEED 45
CMOVE P[1], P[2]
EOP
```

Manipulator moves to P[0] by the linear interpolation in Step 1.
From the current position, implements a circular interpolation passes through 2 teaching mid-positions P[1] and P[2] then returns at a speed of 45% of Max Linear Speed in Step 2.

HOME

Function

Moves to the set home position by joint interpolation.

Format

HOME HomeNo.

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Home No.	Home position No. (There are 2 home positions in setting with ID number are: 1 and 2)

Example

```
MAIN M0
JMOVE #P[0]
HOME 1
EOP
```

Manipulator moves to #P[0] in Step 1.
Returns to the first Home position by joint interpolation in Step 2.

INCJ

Function

Moves to the teaching position in relative mode by joint interpolation.

Format

INCJ #P[No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]#P[No.]	The teaching position.

Example

```
MAIN M0
SPEED 10
INCJ #P[0]   // #P[0] = [5, -10, 20, 12]
EOP
```

Manipulator moves to position #P[0] with relative mode.
Axis 1 increases 5 units.
Axis 2 decreases 10 units.
Axis 3 increases 20 units.
Axis 4 increases 12 units.

INCL

Function

Moves to the teaching position in relative mode with Base Coordinate by linear interpolation.

Format

INCL P[No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]P[No.]	The teaching position in relative coordinate.

Example

```
MAIN M0
SPEED 10
INCL P[1]
EOP
```

Manipulator moves to position P[1] in relative mode with Base Coordinate.

INCT

Function

Moves to the teaching position in relative mode with Tool Coordinate by linear interpolation.

Format

INCT P[No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]P[No.]	The teaching position in relative coordinate.

Example

```
MAIN M0
SPEED 10
INCT P[1]
EOP
```

Manipulator moves to position P[1] in relative mode with Base Coordinate.

TKOBJWAIT

Function

Waits unitl the object comes into the range of robot working.

Format

TKOBJWAIT ConvId

Parameters

[in/out] Name	[Init Value] Explanation
[>>]ConvId	ID of the actived conveyor

Example

```
MAIN  M0
INT ConvId
```

```
ConvId = 0
JMOVE #P[0]              // Step 1
TKOBJWAIT ConvId         // Step 2
TKMOVE ConvId, P[1]      // Step 3
EOP
```

Manipulator moves to wait position in Step 1.

Program waits until object comes into synchronization range in Step 2

Manipulator executes tracking motion follow object in Step 3.

TKMOVE

Function

Implements linear interpolation motion tracking the conveyor.

Format

TKMOVE ConvId, P[No.]

Parameters

[in/out] Name	[Init Value] Explanation
[>>]ConvId	ID of the actived conveyor.
[>>]P[No.]	The teaching position in the conveyor frame.

Example

```
MAIN  M0
INT ConvId
ConvId = 0
JMOVE #P[0]
TKOBJWAIT ConvId
 TKMOVE ConvId, P[1]
EOP
```

Manipulator moves to position P[1] in the conveyor frame to track object's motion.

TKAPPROACH

Function

Approaches along Z axis of conveyor frame while tracking the conveyor's motion.

Format

TKAPPROACH ConvId, Distance

Parameters

[in/out] Name	[Init Value] Explanation
[>>]ConvId	ID of the actived conveyor.
[>>]Distance	Distance value manipulator approaches along Z axis of conveyor frame.

Example

```
MAIN  M0
INT ConvId
ConvId = 0
JMOVE #P[0]
TKOBJWAIT ConvId
 TKMOVE ConvId, P[1]
 TKAPPROACH ConvId, -10
```

```
EOP
```

Manipulator approaches along Z axis of conveyor frame 10 mm while tracking conveyor's motion.

TKDEPART

Function

Departs along Z axis of conveyor frame while tracking the conveyor's motion.

Format

TKDEPART ConvId, Distance

Parameters

[in/out] Name	[Init Value] Explanation
[>>]ConvId	ID of the actived conveyor.
[>>]Distance	Distance value manipulator departs along Z axis of conveyor frame.

Example

```
MAIN  M0
INT ConvId
ConvId = 0
JMOVE #P[0]
TKOBJWAIT ConvId
 TKMOVE ConvId, P[1]
 TKAPPROACH ConvId, -10
 DOUT 0, 1
 TKDEPART ConvID, -10
EOP
```

Manipulator departs along Z axis of conveyor frame 10 mm while tracking conveyor's motion.

TKOBJCOUNT

Function

Returns the number of objects in the object queue.

Format

TKOBJCOUNT ConvID

Parameters

[in/out] Name	[Init Value] Explanation
[>>]ConvID	ID of the active conveyor.

Example

```
MAIN  M0
INT ConvId, Nobj
Convid = 1
Nobj = TKMOVE ConvId
IF Nobj > 0 THEN
DOUT 0, 1
ENDIF
EOP
```

Checks existing of object in object queue to determine execute other tasks.

MSSTOP

Function

Slow down the motion drive and stops.

Format

MSSTOP

Parameters

[in/out] Name	[Init Value] Explanation

Example

```
MAIN  M0
 LMOVE P[0]
IF DIN(0) == 1 THEN
MSSTOP
ENDIF
EOP
```

MESTOP

Purpose

Quickly stops all motions.

Format

MESTOP

Parameters

[in/out] Name	[Init Value] Explanation

Description

Quickly stops all motions.

Example

```
MAIN  M0
LMOVE P[0]
MESTOP
EOP
```

Control Statement

This section is to provide the commands required to control script.

Task

Command	Description
MAIN	Configure the task to start.
EOP	Configure the task to terminate.
EXIT	Configure the task to stop.

Task control

Command	Description
TRUN	Run the task.
TSTOP	Stop the task.

User Function

Command	Description
FUNC	Configure the user function to start.
ENDF	Configure the user function to terminate.
EXITF	Configure the user function to stop.

Label/Branch

Command	Description
LABEL	Configure the location for braching point.
GOTO	Branch the branching points.

Conditional Branching

Command	Description
IF THEN	Execute if the condition in the conditional branch(IF statement) is true.
ELSE	Execute if the condition in the conditional branch(IF statement) is false.
ENDIF	Configure the conditional branch (IF statement) to terminate.
SWITCH	Configure the branch value of multiple branches (SWITCH statement).
CASE	Configure the comparisions of multiple branches (SWITCH statement).
DEFAULT	Configure the basic execute statement of multiple branches (SWITCH statement).
ENDSW	Configure the multiple branches (SWITCH statement) to terminate.

Step Repeat

Command	Description
FOR TO STEP	Configure step repeat (FOR statement) to execute.
NEXT	Configure the step repeat (FOR statement) to return.

Conditional Repeat

Command	Description

WHILE	Start executing the conditional repeat (WHILE statement).
ENDWL	Terminate executing the conditional repeat (WHILE statement).
DO	Start executing the conditional repeat (LOOP statement).
LOOP	Terminate executing the conditional repeat (LOOP statement).

MAIN, EOP, EXIT

Function

Assigns a start, ends and terminates the tasks.

Format

MAIN M0
Execute Statement
EXIT
EOP

MAIN M1
Execute Statement
EXIT
EOP

MAIN M2
Execute Statement
EXIT
EOP

MAIN M3
Execute Statement
EXIT
EOP

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Execute Statement	Programming scripts in the tasks.

MAIN and EOP must be used together to creat the main program block. Up to 4 block (tasks) can be prepared in a program. 4 main statements are executed at a same time and command can be executed independently for each main statement.

EOP (End Of Program) refers to the end of program block and if this command is faced when executing the program, the task is to terminate. At the EOP following line, another main statement or required sub-execute statement block to be concurrently executed can be continuously prepared.

(Up to 4 MAIN statements can be created and used. 4 main statements are executed at a time and 1 to 3 main statement excluding MAIN 0 can be controlled by task command)

Example

```
MAIN M0
WHILE 1
LMOVE P[0]
 LMOVE P[1]
 ENDWL
 EOP

 MAIN M1
 IF DIN(0) == 1 THEN
 DOUT 0, 1
 ENDIF
 EOP
```

TRUN, TSTOP

Function

Control the MAIN 1-3 task.

Format

TRUN TaskNo.
TSTOP TaskNo.

Parameters

[in/out] Name	[Init Value] Explanation
[>>]TaskNo.	ID task. (1-3)

Example

```
MAIN M0
TRUN 1
DELAY 5000
TSTOP 1
DELAY 100
EOP

MAIN M1
WHILE 1
I = I + 1
ENDWL
EOP
```

The task 1 runs immediately after start the task 0, then stops after 5s.

FUNC, ENDF, EXITF

Function

Defines the functions performing a specific role, call and use it whenever necessary.

Format

FUNC FunctionName(Factor, , ..)
Exectute Statement
EXITF
Execute Statement
ENDF

Parameters

[in/out] Name	[Init Value] Explanation
[>>]FunctionName	Name of the function assigned by user.
[>>]Factor	Factor to be used when call a function.
[>>]Execute Statement	Programming script in the function.

Users are eligible to define the function to use it in the MAIN statement, etc. The function can be defined anywhere before or after the MAIN statement. The command to inform the end of function is ENDF, whereas EXITF command can be used to termniate the function in the middle of function execution. The factors in the parenthesis of function can be set varied. The function can be called from the function to use it as a recursive call. Howerver, the function call requires massive memory, therefore, calls more then 128-staged call cannot be assured.

FunctionName = Return value (REAL, INT type are only available for return value). Data type of return value is automatically assigned.

Example

```
MAIN M0
INT RTN
RTN = SUM(1, 10)
EOP

FUNC SUM(INT a, INT b)
INT i, sum
sum = 0
FOR i = a TO b STEP 1
sum = sum + i
NEXT
AA = sum
ENDF
```

Call SUM function with 2 parameters: 1, 10.

Value 55 is stored in RTN variable.

LABEL, GOTO

Function

Assigns the branching position or branches to branching point.

Format

LABEL
GOTO Label

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Label	Branching string composed of alphabetical character and numbers

It will be jumped to a position where LABEL Point command is located once meeting a GOTO Point command and from that position, the command is executed in order. Branching includes MAIN and only valid in the function. The label cannot be repeated even if function is distinguished.

Example

```
MAIN M0
SPEED 10
LABEL   L1
LMOVE P[1]
LMOVE P[2]
GOTO   L1
EOP
```

Once meeting GOTO L1 command, program jumps to position where LABEL L1 command is located.

IF THEN, ELSE, ENDIF

Function

Determines the conditions to execute the block.

Format

IF Condition THEN
Execute the block if the condition is true.
ELSE
Execute the block if the condition is false.
ENDIF

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Condition	Logical operation or comparision operation.

IF THEN and ENDIF must be used together. The statements before THEN are to be executed if the result of conditional operation is true (or values other than 0), whereas the statements following ELSE will be executed if it false (or 0). ELSE statement can be used or skipped upon the neccessity. IF block can be used again in the IF block to use repeat IF block. Here, IF and ENDIF should be logically paired.

Example

```
MAIN M0
SPEED 10
IF DIN(0) == 1 THEN
LMOVE P[1]
ELSE
```

```
LMOVE P[2]
ENDIF
EOP
```

Manipulator moves to position P[1] in case of digital input signal no. 0 equal to 1 and moves to position P[2] otherwise.

SWITCH, CASE, DEFAULT, ENDSW

Function

Branches into many cases upon the given conditions.

Format

```
SWITCH Condition
CASE 1
        Execute the block
CASE 2
        Execute the block
CASE 3
        Execute the block
    .
    .
    .
DEFAULT
        Execute the block
ENDSW
```

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Condition	Variable or formula becomes the conditional base
[>>]Case 1 ~	Values consistent with the given conditions are to be assigned
[>>]Excute the block	Execute block suitable to case 1~ condition is executed.

It must be started form SWITCH and ended with ENDSW. For eample, the formula should be assigned for results to the product integer and string. Next to CASE<the constant responding to conditional value can be used, and various values can be listed by dividing them by “,” (ex: CASE 1,2,3 or CASE “A”, “B”, “C”). However, the parameters cannot be used in CASE statement. The block is executed when the conditions following the SWITCH and the values following the CASE are consistent each other. If the consistent CASE cannot be found, the default execute block will be executed. Here, the default can be skipped and if failed to a consistent CASE if skipped, the SWITCH statement will be completed without executing an EXECUTE block.

Example

```
MAIN M0
INT I
I = DIN(1)
SWITCH I
CASE 0
LMOVE P[0]
CASE 1
LMOVE P[1]
ENDSW
EOP
```

Manipulator moves to position P[0] in case of digital input no.1 equal to 0 and moves to position P[1] in case of digital input no.1 equal to 1.

FOR TO STEP, NEXT

Function

Increases the given parameters by the defined increment at the default to repeatly execute the block until reaching the exit value

Format

FOR Parameter = Initial Value TO Exit Value STEP increment (If increment is 1, STEP can be skipped)
Block
NEXT

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Parameter	Use the name of parameter to be declared with integer type
[>>]Initial value	Integer value to be set on variable right before repeatly executing a FOR block
[>>]Exit value	Integer value to limit the number of repeated execution of FOR block
[>>]Increment	Integer value used to increase the parameter at a constant level

Repeatedly execute FOR~NEXT block until satisfying the exit value after configuring the initial value on the integer parameter. This may be existed in FOR~NEXT statement or another FOR~NEXT statement. If increment is 1, the STEP comand can be skipped; therefore, the FOR statement can be composed with “FOR~TO, NEXT” only.

Example

```
MAIN M0
INT A, TOTAL
TOTAL = 0
FOR A = 1 TO 10 STEP 2
TOTAL = TOTAL + A
NEXT
EOP
```

Executes a loop 10 times to calculate the sum of the odd number between 1 and 10; result is stored in TOTAL variable.

WHILE, ENDWL

Function

Executes the block repeatly while condition is satisfied.

Format

WHILE Condition
Block
ENDWL

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Condition	Logical operation or comparision operation.

WHILE statement repeatedly executes the WHILE block while the conditions are satisfied (or values other than 0). WHILE and ENDWL should be always used together. If the results of condition are always true, it will be infinitely repeatedly executed.

Example

```
MAIN M0
SPEED 10
WHILE DIN(0) == 1
LMOVE P[0]
LMOVE P[1]
ENDWL
EOP
```

Manipulator moves to position P[0] and P[1] repeatly in case of digital input no.0 equal to 1.

DO, LOOP

Function

Executes the block repeatly while conditions is satisfied.

Format

DO
Block
LOOP Conditions

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Conditions	Logical operation or comparision operation.

DO LOOP statement repeatedly executes the DO LOOP block (or values other than 0). DO and LOOP should be always used together. If the results of condition are always true, it will be infinitely repeatedly executed.

Example

```
MAIN M0
SPEED 10
DO
LMOVE P[0]
LMOVE P[1]
LOOP DIN(1) == 1
EOP
```

Manipulators moves to position P[0] and P[1] once time and check digital input no.1. In case of digital input no.1 equal to 1, the loop executes repeatly, and exits the loop otherwise.

Configuration

This section is to provide the commands required to control the script.

Variable

Command	Description
INT	Declare Integer variable.
REAL	Declare Real Type variable.
ARRAY	Declare Array Type variable.
STRING	Declare Character variable
TRANS	Declare Point Array in World Coordinate variable
JOINT	Declare Point Array in Joint Coordinate variable
POINT	Define a point.

Speed

Command	Description
SPEED	Configure the percentage (%) of driving speed.
ACCEL	Configure the percentage (%) of acceleration.
DECEL	Configure the percentage (%) of deceleration.

Execution Delay

Command	Description
DELAY	Delay the program execution by setting time (ms).

INT, REAL

Function

Declares real type, integer variable.

Format

INT Variable name, Variable name, Variable name,…
REAL Variable name, Variable name, Variable name,…

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Variable name	Name of the variable to be used.

The first character of variable should be English. However, a character “P" cannot be solely used due to overlapping with position variable. These parameters should be declared the external upper of MAIN block (member variable) or beginning part (local variable) of MAIN block, and also can be declared within the function. The variables declared as INT are integer parameter (range: -2.14 * 108 ~ 2.14 * 108), whereas the variables declared as REAL are real type parameter (range: ±10-38 ~1038).

Example

```
MAIN M0
INT I
REAL r
I = 0
r = 0.0
EOP
```

ARRAY

Function

Declares Array type variable

Format

ARRAY VariableName[row size][column size]
ARRAY VariableName[row size][column size], VariableName[row size][column size],…

Parameters

[in/out] Name	[Init Value] Explanation
[>>]VariableName	The name of the variable.

The first character of parameter should be English. However, a character “P" cannot be solely used due to overlapping with teaching data.

Example

```
MAIN M0
ARRAY A[2][4]
A[0] = [100, 200, 10, 100]
EOP
```

STRING

Function

Declare string variable.

Format

STRING VariableName, VariableName, VariableName,…

Parameters

[in/out] Name	[Init Value] Explanation
[>>]VariableName	The name of variable.

The first letter of a variable must be in English. However, the 'P' character alone cannot be used because it overlaps with the Position variable. These variables must be declared at the top of the MAIN block (global variable) or at the beginning of the MAIN block (region variable), and can also be declared within the function. Unlike regular numeric data, these string variables should be handled using string-only functions.

Example

```
STRING strA, strB, strC
MAIN M0
strA = “This is”
strB = “a String”
strC = “\32”
strA = SCAT(strA, strC)
strA = SCAT(strA, strB)
EOP
```

“This is a String” is stored in strA variable.

SPEED

Function

Sets the motion speed for the next motion command.

Format

SPEED Velocity

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Velocity	Set the movement speed of the manipulator. (1% ≤ speed ≤ 100%) Robot Speed = Configured Speed * Speed/100

Example

```
MAIN M0
WHILE 1
SPEED 10
LMOVE P[0]
SPEED 50
LMOVE P[2]
ENDWL
EOP
```

Manipulators moves to position P[0] at a speed of 10% of Max Linear Velocity and to position P[2] at a speed of 50% of Max Linear Velocity.

ACCEL, DECEL

Function

Sets the acceleration and deceleration for the motion.

Format

ACCEL Value
DECEL Value

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Value	(1% ≤ Value ≤ 100%)

Example

```
MAIN M0
ACCEL 50
DECEL 50
LMOVE P[0]
EOP
```

DELAY

Function

Deplays the program execution in the set time.

Format

DELAY Time

Parameters

[in/out] Name	[Init Value] Explanation
[>>]Time	Set the time to delay (1 = 1ms) (1 ≤ Time)

Example

```
MAIN M0
SPEED 10
LMOVE P[0]
DELAY 50
LMOVE P[1]
EOP
```

Manipulator moves to position P[0] then halts 50 ms before moving to position P[1].

Status Check

This section introduces the commands you need to control the Script.

Robot status

Command	Description
CMDPOS	Returns the command position of a robot axis.
ACTPOS	Returns the actual position of a robot axis.

Task Status

Command	Description
TSTATUS	Returns the task state.

CMDPOS

Function

Returns the command position of a robot axis.

Format

CMDPOS nAxis

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nAxis	Axis No.

Example

```
MAIN M0
INT nAxis
REAL rCmdPos
nAxis = 0
rCmdPos = CMDPOS nAxis
EOP
```

ACTPOS

Function

Returns the actual position of a robot axis.

Format

ACTPOS nAxis

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nAxis	Axis No.

Example

```
MAIN M0
INT nAxis
REAL rActPos
nAxis = 0
rActPos = ACTPOS nAxis
EOP
```

TSTATUS

Function

Checks the status of the task.

Format

TSTATUS TaskNo

Parameters

[in/out] Name	[Init Value] Explanation
[>>]TaskNo	ID Task.

Example

```
MAIN M0
INT A
A = TSTATUS 2
EOP
```

In/Out

Digital Input

Command	Description
WAITDIN	Waits until the signal conditions meet.
DIN	Reads the value of digital signal

Digital Ouput

Command	Description
DOUT	Sets value for the digital output signal.
DOUTGROUP	Sets value for the digital output signal group.
PULSEDOUT	Creates a pulse on a digital output signal.
DELAYDOUT	Delays digital output signal.
RESETDOUT	Reset all digital output signal.

WAITDIN

Function

Waits the condition of digital input signal meets.

Format

WAITDIN SignalNo, Condition

Parameters

[in/out] Name	[Init Value] Explanation
[>>]SignalNo	The signal No.
[>>]Condition	Condition of digital input signal (0/1).

Example

```
MAIN M0
INT I
I = 0
WAITDIN 0, 1
// Do task
I = I + 1
EOP
```

Program waits until digital input signal no.0 is equal to 1.

DIN

Function

Returns the status of digital input signal.

Format

DIN SignalNo

Parameters

[in/out] Name	[Init Value] Explanation
[>>]SignalNo	The signal no..

Example

```
MAIN M0
IF DIN 0 == 1 THEN
LMOVE P[0]
ELSE
LMOVE P[1]
ENDIF
EOP
```

Manipulator moves to position P[0] in case of digital input no.0 is equal to 1. Otherwise, moves to position P[1].

DOUT

Function

Turns on/off the digital output signal.

Format

DOUT SignalNo, Value

Parameters

[in/out] Name	[Init Value] Explanation
[>>]SignalNo	The signal no.
[>>]Value	On = 1; Off = 0

Example

```
MAIN M0
INT I
FOR I = 0 TO 5 STEP 1
DOUT I, 1
NEXT
EOP
```

Turns ON digital output signal from no.0 to no.5 consecutively.

DOUTGROUP

Function

Sets value for the digital output signal group.

Format

DOUTGROUP GroupNo., Value

Parameters

[in/out] Name	[Init Value] Explanation
[>>]GroupNo.	Group no.(1-8)
[>>]Value	Value is to set for the digital output signal group. (0-255)

Example

```
MAIN M0
DOUTGROUP 1, 240
EOP
```

Sets value 240 to the digital output signal no.1.
240 = b11110000
Bit 0-3 of group 1 are turned OFF.
Bit 4-7 of group 1 are turned ON.

PULSEDOUT

Function

Generates a pulse on digital output signal.

Format

PULSEDOUT SignalNo, Time

Parameters

[in/out] Name	[Init Value] Explanation
[>>]SignalNo	The signal no.
[>>]Time	The active time of pulse (ms).

Example

```
MAIN M0
PULSEDOUT 0, 500
EOP
```

Generates a 500 ms pulse-width on digital output no.0.

DELAYDOUT

Function

Delays turning ON a digital output signal.

Format

DELAYDOUT SignalNo, Time

Parameters

[in/out] Name	[Init Value] Explanation
[>>]SignalNo	The signal no.
[>>]Time	The delay duration(ms).

Example

```
MAIN M0
DELAYDOUT 0, 500
EOP
```

Delays 500ms before turning ON digital output no.0.

RESETDOUT

Function

Resets all digital output signal.

Format

RESETDOUT

Parameters

[in/out] Name	[Init Value] Explanation

Example

```
MAIN M0
RESETDOUT
EOP
```

Operation

Array Operator

Command	Description
[	Left bracket.
]	Right bracket.

Binary Operator

Command	Description
+	Addition operator
-	Subtraction operator
*	Multiplication operator
/	Division operator
%	Remaining operator

Priority Operator

Command	Description
(	Left bracket
)	Right bracket

Substitute Operator

Command	Description
=	Substitute the right value to the left variable.

Logical Operator

Command	Description
&&	Logical AND operator
||	Logical OR operator
!	Logical NOT operator

Comparision operator

Command	Description
<	Less than comparision operator
>	Greater than comparision operator
<=	Less than or equal than comparision operator
>=	Greater than or equal than comparision operator
!=	Not equal comparision operator
==	Equal comparision operator

Bitwise Operator

Command	Description
&	Bitwise AND

|	Bitwise OR
^	Bitwise XOR
~	Binary One's Complement operator
<<	Left shift operator
>>	Right shift operator

[,]

Function

Array operator.

Format

[ ]

Parameters

[in/out] Name	[Init Value] Explanation

Example

```
ARRAY A[10][4]
MAIN M0
A[0] = [1.0, 100.0, 20.0, 0.0]
 EOP
```

+

Function

Addition operator.

Format

A + B

Example

```
MAIN M0
INT nData1, nData2
REAL rData3
nData1 = 5
nData2 = 10
rData3 = nData1 + nData2
 EOP
```

15 is stored in rData3 variable.

-

Function

Subtraction operator

Format

A – B

Example

```
MAIN M0
INT nData1, nData2
REAL rData3
nData1 = 5
nData2 = 10
rData3 = nData1 - nData2
EOP
```

-5 is stored in rData3 variable.

*

Function

Multiplication operator

Format

A * B

Example

```
MAIN M0
INT nData1, nData2
REAL rData3
nData1 = 5
nData2 = 10
rData3 = nData1 * nData2
EOP
```

50 is stored in rData3 variable.

/

Function

Division operator

Format

A / B

Example

```
MAIN M0
INT nData1, nData2
REAL rData3
nData1 = 5
nData2 = 10
rData3 = nData1 + nData2
EOP
```

0.5 is stored in rData3 variable.

%

Function

Remaining operator

Format

A % B

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = nData1 % 3
EOP
```

2 is stored in nData2 variable.

(,)

Function

Specifies the blocks to be prioritized in the operation expression.

Format

( )

Example

```
MAIN M0
INT nData1, nData2
REAL rData3
nData1 = 5
nData2 = 10

// Do not use priority operators 5*10+10*10 = 150
rData3 = nData1 * nData2 + nData2 * nData2

// Use priority operators 5*(10+10)*10 = 1000
rData3 = nData1 * (nData2 + nData2) * nData2
EOP
```

=

Function

Assignment operator

Format

A = B

Parameters

[in/out] Name	[Init Value] Explanation
[>>]A	Variable to store the value.
[>>]B	Value or variable to be saved.

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = 10
EOP
```

&&

Function

Logical AND operator

Format

A && B

X	Y	X&&Y
0	0	0
0	1	0
1	0	0
1	1	1

Example

```
MAIN M0
INT nData1, nData2, nData3
nData1 = 0
nData2 = 1
nData3 = nData1 && nData2
EOP
```

0 is stored in nData3 variable.

||

Function

Logical OR operator

Format

A || B

X	Y	X||Y
0	0	0
0	1	1
1	0	1
1	1	1

Example

```
MAIN M0
INT nData1, nData2, nData3
nData1 = 0
nData2 = 1
nData3 = nData1 || nData2
EOP
```

1 is stored in nData3 variable.

!

Function

Logical NOT operator

Format

!

X	!X
0	1
1	0

Example

```
MAIN M0
INT nData1, nData2
nData1 = 0
nData2 = !nData1
EOP
```

1 is stored in nData2 variable.

>

Function

Greater comparision.

Format

>

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = 10
IF nData1 > nData2 THEN
DOUT 0,1
ENDIF
EOP
```

<

Function

Less comparision.

Format

<

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = 10
IF nData1 < nData2 THEN
DOUT 0,1
ENDIF
EOP
```

>=

Function

Greater than or equal comparision.

Format

>=

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = 10
IF nData1 >= nData2 THEN
DOUT 0,0
ENDIF
EOP
```

<=

Function

Less than or equal comparision.

Format

<=

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = 10
IF nData1 <= nData2 THEN
LMOVE P[0]
ELSE
LMOVE P[1]
ENDIF
EOP
```

!=

Function

Difference operator.

Format

!=

Example

```
MAIN M0
INT nData1, nData2, nData3
nData1 = 5
nData2 = 10
IF nData1 != nData2 THEN
PULSEDOUT 0, 500
ENDIF
EOP
```

==

Function

Equal comparision.

Format

==

Example

```
MAIN M0
INT nData1, nData2
nData1 = 5
nData2 = 10
IF nData1 == nData2 THEN
LMOVE P[2]
ENDIF
EOP
```

&

Function

Bitwise AND operator.

Format

&

X	Y	X & Y
0	0	0
0	1	0
1	0	0
1	1	1

Example

```
MAIN M0
 INT nData1, nData2, nData3

 // 35 = 100011
 nData1 = 35

 // 53 = 110101
 nData2 = 53

 // Bitwise AND 100011&110101 = 100001(33)
 nData3 = nData1 & nData2

 EOP
```

33 is stored in nData3 variable.

|

Function

Bitwise OR operator.

Format

|

X	Y	X | Y
0	0	0
0	1	1
1	0	1
1	1	1

Example

```
 MAIN M0
INT nData1, nData2, nData3

// 35 = 100011
nData1 = 35

// 53 = 110101
nData2 = 53
```

```
// Bitwise OR 100011|110101 = 110111(55)
nData3 = nData1 | nData2

EOP
```

55 is stored in nData3 variable.

^

Function

Bitwise XOR operator.

Format

^

X	Y	X ^ Y
0	0	0
0	1	1
1	0	1
1	1	0

Example

```
MAIN M0
INT nData1, nData2, nData3

// 35 = 100011
nData1 = 35

// 53 = 110101
nData2 = 53

// Bitwise XOR 100011^110101 = 010110(22)
nData3 = nData1 ^ nData2

EOP
```

22 is stored in nData3 variable.

~

Function

Bitwise NOT operator.

Format

~

X	~X
0	1
1	0

Example

```
MAIN M0
INT nData1, nData2, nData3

// 35 = 100011
nData1 = 35

// 53 = 110101
nData2 = 53

// Bitwise NOT ~100011 = 11111111111111111111111111011100(-36)
nData3 = ~nData1

EOP
```

-36 is stored in nData3 variable.

<<

Function

Left shift operator.

Format

<<

Example

```
MAIN M0
INT nData1, nData2

// 35 = 100011
nData1 = 35

// Left shit 100011<<< 3 = 0001100011000 (280)
nData2= nData1 << 3

EOP
```

280 is stored in nData2 variable.

>>

Function

Shift right operator.

Format

>>

Example

```
MAIN M0
INT nData1, nData2

// 35 = 100011
nData1 = 35

// Right shift 100011>>3 = 000100(4)
nData2 = nData1 >> 3
EOP
```

4 is stored in nData2 variable.

ABS

Function

Returns absolute value of a number.

Format

ABS(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rReturn
rReturn = ABS(-88.3)
EOP
```

88.3 is stored in rReturn variable.

DEG

Function

Returns degree conversion of a number.

Format

DEG(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rDeg
rDeg = DEG(3.141592)
EOP
```

180 is stored in rDeg variable.

RAD

Function

Returns radian conversion of a number.

Format

RAD(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rRad
rRad = RAD(180)
EOP
```

3.14159265359 is stored in rRad variable.

ROUND

Function

Returns rounding conversion of a number.

Format

ROUND(REAL rData, INT nNumber)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.
[>>]nNumber	Number of digits.

Example

```
MAIN M0
REAL r
r = ROUND(-12.453647, 3)
EOP
```

-12.454 is stored in r variable.

FLOOR

Function

Returns rounding down conversion of a number.

Format

FLOOR(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rRet
rRet = FLOOR(3.141592)
EOP
```

3.0 is stored in rRet variable.

CEIL

Function

Returns rounding up conversion of a number.

Format

CEIL(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rRet
rRet = CEIL(3.1415923)
EOP
```

4.0 is stored in rRet variable.

SQRT

Function

Returns square root transformation of a number.

Format

SQRT(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rRet
rRet = SQRT(2)
EOP
```

1.41421356237 is stored in rRet variable.

POW

Function

Returns result of the first argument raised to the power of the second argument.

Format

POW(REAL rData, INT nCount)

Parameters

[in/out] Name	[Init Value] Explanation

[>>]rData	Base value.
[>>]nCount	Exponent value.

Example

```
MAIN M0
REAL rRet
rRet = POW(4, 3)
EOP
```

64.0 is stored in rRet variable.

EXP

Function

Returns the base-e exponential function.

Format

EXP(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Value of the exponent.

Example

```
MAIN M0
REAL rRet
rRet = EXP(10)
EOP
```

LOG

Function

Returns the result of natural logarithm conversion.

Format

LOG(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rRet
rRet = LOG(10)
EOP
```

2.30258509299 is stored in rRet variable.

LOG10

Function

Returns the result of common logarithm conversion.

Format

LOG10(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion value.

Example

```
MAIN M0
REAL rRet
rRet = LOG10(10)
EOP
```

1.0 is stored in rRet variable.

MIN

Function

Returns the maximin number.

Format

MIN(REAL rData1, REAL rData2)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData1	Value 1.
[>>]rData2	Value 2.

Example

```
MAIN M0
REAL rRet
rRet = MIN(-5, 10)
EOP
```

-5 is stored in rRet value.

MAX

Function

Return the maximum number.

Format

MAX(REAL rData1, REAL rData2)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData1	Value 1.
[>>]rData2	Value 2.

Example

```
MAIN M0
REAL rRet
rRet = MAX(-5, 10)
EOP
```

10 is stored in rRet variable.

AVG

Function

Returns the average of two numbers.

Format

AVG(REAL rData1, REAL rData2)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData1	Value 1.
[>>]rData2	Value 2.

Example

```
MAIN M0
REAL rReturn
rReturn = AVG(2, 88)
EOP
```

45 is stored in rReturn variable.

SIN

Function

Returns the result of sine function.

Format

SIN(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion Value.

Example

```
MAIN M0
REAL rReturn
rReturn = RAD(30)
rReturn = SIN(rReturn)
EOP
```

0.5 is stored in rReturn variable.

ASIN

Function

Returns the result of arc sine function.

Format

ASIN(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion Value.

Example

```
MAIN M0
REAL rReturn
rReturn = ASIN(0.5)
EOP
```

0.52359877559 is stored in rReturn variable.

COS

Function

Returns the result of cosine function.

Format

COS(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion Value.

Example

```
MAIN M0
REAL rReturn
rReturn = RAD(30)
rReturn = COS(rReturn)
EOP
```

0.866025403 is stored in rReturn variable.

ACOS

Function

Returns the result of arc cosine function.

Format

ACOS(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion Value.

Example

```
MAIN M0
REAL rReturn
rReturn = ACOS(0.5)
EOP
```

1.04719755119 is stored in rReturn variable.

TAN

Function

Returns the result of tangent funciton.

Format

TAN(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion Value.

Example

```
MAIN M0
REAL rReturn
rReturn = RAD(45)
rReturn = TAN(rReturn)
EOP
```

1.0 is stored in rReturn variable.

ATAN

Function

Retuns the result of arc tangent function.

Format

ATAN(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion Value.

Example

```
MAIN M0
REAL rReturn
rReturn = ATAN(1)
EOP
```

0.7853981633 is stored in rReturn variable.

ATAN2

Function

Returns the result of arc tangent with two parameters function.

Format

ATAN2(REAL rData1, REAL rData2)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData1	Value representing the proportion of the y-coordinate.
[>>]rData2	Value representing the proportion of the x-coordinate.

Example

```
MAIN M0
REAL rReturn
rReturn = ATAN2(-1, 1)
EOP
```

-0.7853981633 is stored in rReturn variable.

SBLOCK

Function

Returns the desired block by dividing the input string into blocks based on spaces.

Format

SBLOCK(STRING strData, INT nBlockNo)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.
[>>]nBlockNo	Block number.

Example

```
STRING strData1, strData2
MAIN M1
    strData1 = "VAS Corp. VN"
    strData2 = SBLOCK(strData1, 1)
EOP
```

“VAS” is stored in strData2 variable.

STOI

Function

Converts string to integer number.

Format

STOI(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string with the represent of an integer number.

Example

```
MAIN M1
INT nReturn
nReturn = STOI("1205")
EOP
```

1205 is stored in nReturn variable.

STOF

Function

Converts string to real number.

Format

STOF(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string with the represent of an real number.

Example

```
MAIN M1
REAL rReturn
rReturn = STOF("12.05")
EOP
```

12.05 is stored in rReturn variable.

ITOS

Function

Converts an integer number to a string.

Format

ITOS(INT nData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nData	Conversion integer number.

Example

```
STRING strData
MAIN M0
strData = ITOS(12345)
EOP
```

“12345” is stored in strData variable.

FTOS

Function

Converts a real number to a string.

Format

FTOS(REAL rData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]rData	Conversion real number.

Example

```
STRING strData
MAIN M0
strData = ITOS(12.345)
EOP
```

“12.345” is stored in strData variable.

SCMP

Function

Compares two strings. Returns 1 if strings are same and returns 0 if strings are different.

Format

SCMP(STRING strData1, STRING strData2)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData1	Input string 1.
[>>]strData2	Input string 2.

Example

```
MAIN M0
INT nReturn
nReturn = SCMP(“ABC”, “ABC”)
EOP
```

1 is stored in nReturn variable.

SLEFT

Function

Copies substring in the input string with number of characters from the left.

Format

SLEFT(STRING strData, INT nCount)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.
[>>]nCount	Number of characters.

Example

```
STRING strData1, strData2
MAIN M0
strData1 = “12 34 5”
strData2 = SLEFT(strData1, 3)
EOP
```

“34 5” is stored in strData2 variable.

SMID

Function

Returns the string from the starting character index to the ending character index .

Format

SMID(STRING strData, INT nStartCount, INT nEndCount)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.
[>>]nStartCount	Starting character index
[>>]nEndCount	Ending character index

Example

```
STRING strData1, strData2
MAIN M0
strData1 = “VAS Corp.”
strData2 = SMID(strData1, 3, 6)
EOP
```

“S Co” is stored in strData2 variable.

SRIGHT

Function

Copies substring in the input string with number of characters from the right.

Format

SRIGHT(STRING strData, INT nCount)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.
[>>]nCount	Number of characters.

Example

```
STRING strData1, strData2
MAIN M0
strData1 = “VAS Corp.”
strData2 = SRIGHT(strData1, 5)
EOP
```

“Corp.” is stored in strData2 variable.

SLEN

Function

Returns the length of the input string.

Format

SLEN(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Example

```
MAIN M1
INT nReturn
nReturn = SLEN("VAS Corp.")
EOP
```

9 is stored in nReturn variable.

SCAT

Function

Concatenates strings.

Format

SCAT(STRING strDestination, STRING strSource)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strDestination	String contains the concatenated resulting string.
[>>]strSource	String to be appended.

Example

```
STRING strData1, strData2, strData3
MAIN M0
strData1 = "VAS "
strData2 = "Corporation"
strData3 = SCAT(strData1, strData2)
EOP
```

"VAS Corporation" is stored in strData3 variable.

SLTRIM

Function

Removes left space string.

Format

SLTRIM(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Example

```
STRING strData1, strData2
MAIN M0
strData1 = “ 12 34 5”
strData2 = SLTRIM(strData1)
EOP
```

“12 34 5” is stored in strData2.

SRTRIM

Function

Removes right space string.

Format

SRTRIM(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Example

```
STRING strData1, strData2
MAIN M0
strData1 = “ 12 34 5   ”
strData2 = SRTRIM(strData1)
EOP
```

“ 12 34 5” is stored in strData2 variable.

STRIM

Function

Removes left space string and right space string.

Format

STRIM(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Example

```
STRING strData1, strData2
MAIN M0
strData1 = “ 12 34 5   ”
strData2 = STRIM(strData1)
EOP
```

"12 34 5" is stored in strData2 variable.

STOKEN

Function

Splits string into tokens.

Format

STOKEN(STRING strData, STRING strDelimiter, INT nCount)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.
[>>]strDelimiter	The delimiter characters.
[>>]nCount	Number of separations.

Example

```
STRING strData1, strData2
MAIN M0
strData1 = "12<34<5"
strData2 = STOKEN(strData1, ">", 1)
EOP
```

"12" is stored in strData2 variable.

ITOHEX

Function

Converts an integer to hexa string.

Format

ITOHEX(INT nData, INT nNum)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nData	Input data.
[>>]nNum	Number of characters.

Example

```
STRING strData
MAIN M0
strData = ITOHEX(1234, 5)
EOP
```

"004D2" is stored in strData variable.

ITOOCT

Function

Converts an integer number to octal string.

Format

ITOOCT(INT nData, INT nNum)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nData	Input data.
[>>]nNum	Number of characters.

Example

```
STRING strData
MAIN M0
strData = ITOOCT(1234, 5)
EOP
```

“02322” is stored in strData variable.

ITOBIN

Function

Converts an integer number to binary string.

Format

ITOBIN(INT nData, INT nNum)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nData	Input data.
[>>]nNum	Number of characters.

Example

```
STRING strData
MAIN M0
strData= ITOBIN(1234, 16)
EOP
```

“010011010010” is stored in strData variable.

HEXTOI

Function

Converts a hexa string to an integer number.

Format

HEXTOI(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string

Example

```
STRING strData
MAIN M1
strData = HEXTOI(“1F0”)
EOP
```

“000111110000” is stored in strData variable.

OCTTOI

Function

Converts a octal string to an integer number.

Format

OCTTOI(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Example

```
MAIN M0
INT nReturn
nReturn = OCTTOI("2336")
EOP
```

1182 is stored in nReturn variable.

BINTOI

Function

Converts a binary string to an integer number.

Format

BINTOI(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Description

Returns an integer number by converting the binary input string.

Example

```
MAIN M0
INT nReturn
nReturn = BINTOI("10110111")
EOP
```

183 is stored in nReturn variable.

STOASCII

Function

Converts a string to an ASCII character.

Format

STOASCII(STRING strData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]strData	Input string.

Example

```
MAIN M0
INT nReturn
nReturn = STOASCII(“A”)
EOP
```

65 is stored in nReturn variable.

ITOCHAR

Function

Converts an integer number to a character.

Format

ITOCHAR(INT nData)

Parameters

[in/out] Name	[Init Value] Explanation
[>>]nData	Input Data

Description

Returns character corresponding with the input number.

Example

```
STRING strData
MAIN M0
strData = ITOCHAR(75)
EOP
```

“K” is stored in strData variable.

Revision History

Revision	Date	Description	Author
RevA.001	25.01.2021	Initial version	TD.LB
RevB.001	30.06.2021		TD.LB
RevC.001	09.10.2021		TD.LB
RevD.001	21.06.2022		TD.LB
RevE.001	14.12.2022		TD.LB