

**NS-Series**

# **MACRO REFERENCE**

**OMRON**

## Section 1 Outline of Macro Function

*NS series Macro Reference*

## **Section 1 Outline of Macro Function**

This section describes execution conditions and programming procedure for using macro.

1-1 What is Macro?.....	1 – 2
1-2 Macro Execution Condition.....	1 – 4
1-3 Macro Programming.....	1 – 7

## 1-1 What is Macro?

Macro is the function which can be executed by users original program. User can add functions, such as arithmetic operations and distinction of conditions, which are not supported by standard functions in NS-Designer. This function allows the PT to process screen display or data, which is performed by PLC before. It is also possible to reduce a load of PLC. In this manual, the timing for executing macro is called "Macro Execution Condition". Macro can be made by roughly divided three execution conditions as shown below.

- Execution condition for the project
- Execution condition for the screen
- Execution condition for the functional objects

There is no restriction on the number of macros for 1 project/1screen.  
Up to 3000 characters can be used for one macro. Line feed is counted as two characters. There is also no restriction on the number of lines.

Example:

'Number of inputting characters 11 characters+line

b feed (2characters), including comment  
\$W0=10; 7characters+line feed (2characters)  
STRCPY(\$W10",ABCDE"); 21characters  
In this case, 43 characters are used.

## 1-2 Macro Execution Condition

Macros can be created for each project, screen, and functional objects. Also they can be created for the following execution conditions.

### Executing conditions for the project

Macro execution conditions that can be made for the project are indicated below. Select [Settings]-[Project properties]-[Macro] tab in NS-Designer, then set execution condition and record macro.

For details on registering macros, refer to NS-Designer operation manual “6-1 Registering Macros”-“6-1-1 Registering Macros to Projects”.

Execution condition	Explanation
When Loading a Project	Execute just before loading the first screen after starting up NS-Hardware
Alarm/Event ON Timing	Execute when alarm is occurred
Alarm/Event OFF Timing	Execute when alarm is cancelled

### Execution conditions for the screen

Macro execution conditions that can be made for each screen are indicated below. Select [Settings]-[Screen properties]-[Macro] tab screen in NS-Designer, then set execution condition and record macro.

For details on registering macros, refer to NS-Designer operation manual “6-1 Registering Macros”-“Registering Macros to Screens”.

Execution condition	Explanation
When Loading a Screen	Execute immediately after reading screen data to display the next
When Unloading a Screen	Execute immediately after closing the current screen

### Reference

Macros are executed by the timing as shown below.

	When loading a screen (Execute at the destination screen)	When unloading a screen (Execute at the destination screen)
User screen->User screen	Executed	Executed
User screen->Transfer screen	Not Executed	Executed
User screen->System menu	Not Executed	Executed
System menu->User screen	Executed	Not Executed
User screen->Screen Saver	Not Executed	Not Executed
Screen Saver->User screen	Not Executed	Not Executed

## Section 1 Outline of Macro Function

### NS series Macro Reference

#### Execution conditions for the functional objects

Macro execution conditions that can be made for each functional object are described in the following table.

Open property dialog for each functional object and select [Macro] tab page, then set the execution condition and create macro.

For details on registering macros, refer to NS-Designer operation manual "6-1 Macro Registration"- "Registration for the functional objects".

Execution condition	Explanation
Touch on Timing	Execute when functional object is pressed.
Touch off Timing	Execute when functional object is released
Before Inputting numeral or character string	Execute just before display tenkey pad or virtual keyboard for inputting values or strings
Before Writing numeral or character string	Execute just before notice numeral and character string to the host.
When changing numeral and character string	Execute then changing the value of address.
When Processing Display Area	Execute when display area for alarm display is pressed.
When Selecting an Alarm/Event	Execute just after select each alarm/event displayed on Alarm/Event Summary
When selecting a list	Execute just after select a list displayed on the List Selection.

#### **Note**

If the password is set for functional objects, the following macros are executed after inputting the password. If the password has been cancelled for inputting, macros will not be executed.

- Touch on/Touch off timing
- Before inputting Numeral/Character string
- When pressing Display Area
- When selecting an Alarm/Event
- When selecting a list

## Section 1 Outline of Macro Function

### NS series Macro Reference

The following conditions can be selected for functional objects.

Functional Object	Touch on Timing	Touch Off Timing	When changing Numeral/String	Before inputting Numeral/String	Before writing Numeral/String	List Selection
ON/OFF Button	OK	OK	-	-	-	-
Word Button	OK	OK	-	-	-	-
Command Button	OK	OK	-	-	-	-
Bit Lamp	-	-	OK	-	-	-
Word Lamp	-	-	OK	-	-	-
Numeral Display & Input	-	-	OK	OK	OK	-
String Display & Input	-	-	OK	OK	OK	-
Thumbwheel Switch	-	-	OK	-	OK	-
Text	-	-	-	-	-	-
List Selection	-	-	-	-	-	OK
Level Meter	-	-	-	-	-	-
Broken-line Graph	-	-	-	-	-	-
Bitmap	-	-	-	-	-	-
Analogue Meter	-	-	-	-	-	-
Video Display	-	-	-	-	-	-
Date	-	-	-	-	-	-
Time	-	-	-	-	-	-
Data Log Graph	-	-	-	-	-	-
Data Block Table	-	-	-	OK	OK	-

#### [Alarm/Event object]

Functional Object	When Pressing a Display Area	When selecting an Alarm/Event
Alarm/Event Display	OK	-
Alarm/Event Summary	-	OK

## 1-3 Macro Programming

This section describes macro creation procedures and programming terms.

### The Method for Writing a Macro

#### Delimiter of the Program

Put a semicolon (;) at the end of each program as a delimiter. However, it is not necessary for IF (), ELSEIF (), ELSE(),ENDIF.

Example;

```
$W0=2;
```

```
IF ($W0>=10)
```

```
    $W5=$W0-$W2;
```

```
ELSE
```

```
    $W5=$W0+$W2;
```

```
ENDIF
```

#### Comment

Put single quotation mark at the beginning of the sentence when you add the comment for each program. From single quotation mark (') to the end of character sting will be regarded as a comment.

Example:

```
$W0 = 100; 'Comment
```

```
'Comment
```

```
IF ($W1==200)
```

```
...
```

#### Writing Programming Terms

Both uppercase and lowercase can be used for programming macros because they are not classified.

Example:

-MovePopwDown () and MOVEPOPWDOWN() are regarded as same function.

-[Host1:DM0]and [host1:dm0] are regarded as same host address.



## Section 1 Outline of Macro Function

### NS series Macro Reference

#### Programming Terms

This section describes terms used in this function.

#### Variable

The following variables can be used in macro program.

Item	Explanation
Host address	Use functions (READCMEM and WRITECMEM) for communication when accessing to the address in the host. Enclose address in [ ] Example: READCMEM(\$W100,[HOST1:DM00000],100); 'Read HOST1:DM00000 to DM00099 to \$W100 to \$W199
PT Memory	Bit Internal memory: \$B                      \$B0 to \$B32767 (1bit per 1point) Internal Holding memory: \$HB         \$HB0 to \$HB8191 (1bit per 1point) System Memory: \$SB                     \$SB 47 (1bit per 1point) Word Internal memory: \$W                     \$W to \$W32767(16bit per 1point) Internal Holding memory: \$HW         \$HW0 to \$HW 8191 (16bit per 1point) System Memory: \$SW                     \$SW0 to \$SW36 (16bit per 1point) Example: \$W100=\$W0+1; 'Set the value \$W0 and 1 added to \$W100
Index	Index is used for processing bit and word in the PT memory. Add index to the end of the address and it will processed as [specified address + index value] There are 10 index points (I0 to I9). Set I0 to I9 for the value of \$SW27 to \$SW36. Example: \$SW27 = H20; \$W0I0 = 123; '\$W0I0 is regarded as \$W20 added \$W0 and 20 '\$W20 = 123

#### Qualifier of Variable

Qualifiers set for variables must be used as shown below.

Qualifiers are used when performing 32-bit data processing and numeral processing for bit.

Item	Explanation
Long Access (32bit) of Word (16bit)	Put "L" at the end of variable. Uses 2 words. \$W0L=1000000; 'Accesses regarding \$W0, \$W1 as 32-bit \$W100L=1000*1000; 'Accesses regarding \$W100, \$W101 as 32-bit
Numeral Access of Bit	Put ":n" at the end of variable. Specify the value of bit address (up to 32 by 4-bit unit) for "n". Exceptions: If n=16, input "W". If n=32, input "L". \$B0:4 = 3; 'Sets 3(0011) for 4-bit from \$B0 to \$B3 \$B0W = 12345; 'Sets 12345(0011000000111001) for 16-bit from \$B0 to \$B15

## Section 1 Outline of Macro Function

### NS series Macro Reference

#### Constant

Usable constants for macro program and procedure are described in the following table.

Item	Explanation
Decimal constant	-32768 to 32767 can be input when using word (16-bit) -2147483648 to 2147483647 can be input when using 2 words (32-bit)
Hexadecimal constant	H0 to HFFFF can be input when using word (16-bit) H0 to HFFFFFFFF can be input when using 2 words (32-bit)
Character String	Enclose in “ “ E.g. "ABCDE"

#### Branches

The following keywords can be used for specifying conditions.

Item	Explanation
IF ELSEIF ELSE ENDIF	Enclose conditional expressions in parentheses () after the IF and ELSEIF. Always use ENDIF at the end. Up to 8 nests can be input. There is no restrictions for inputting the number of lines under IF sentence. However, the total number of characters used in whole macro must be no more than 1024 characters. E.g. IF(\$W100 == 1)                    'if \$W100 is 1 \$W99 = 1; ELSEIF(\$W100 == 2)            'if \$W100 is 2 \$W99 = 2; ELSE                                'if \$W100 is other than 1 or 2 \$W99 = 3; ENDIF

## Section 1 Outline of Macro Function

### NS series Macro Reference

#### Conditional Expressions

Use the following conditional expressions for specifying conditions in the IF sentences. It works for all types of data (word, long access of word, bit, and numeral access of bit).

Item	Explanation
A == B	If A is equal to B, TRUE.
A > B	If A is greater than B, TRUE.
A >= B	If A is greater than or equal to B, TRUE.
A < B	If A is less than B, TRUE.
A <= B	If A is less than or equal to B, TRUE.
A <> B	If A is not equal to B, TRUE.
A != B	
A && B A AND B	If both A and B are true, TRUE. (AND)
A    B A OR B	If either of A or B is true, TRUE. (OR)

#### **Reference**

The result (A&&B, A>B) can be substituted for the variable.

E.g.\$B100=\$W0>100;

If the value of \$W is "100" or less, "0" will be substituted for \$B100. If the value of \$W is more than "100", "1" will be substituted for \$B100.

## Section 1 Outline of Macro Function

### NS series Macro Reference

#### Basic Operational Statements

The following operational statements can be used in the program.

Item	symbol	Example	meaning
substitution	=	A = B	Substitute B for A
addition	+	C = A+B	Set A+B to C
subtraction	-	C = A-B	Set A-B to C
multiplication	*	C = A*B	Set AxB to C
division	/	C = A/B	Set A/B to C
residue	%	C = A%B	Set A%B to C
OR		C = A B	Logically ORs for A and B
AND	&	C = A & B	Logically ANDs for A and B
NOT	!	C = !A	Set C for denial of A
Exclusive	^	C = A^B	Result C of logical XORs for A and B
Complement of 1	~	B = ~A	Set the 1-complement of A to B
Bit Shift (left)	<<	C = A<<B	Set the value that A is shifted B-bit to the left to C.
Bit Shift (right)	>>	C = A>>B	Se the value that A is shifted B-bit to the right to C.

#### Reference

If executing logical operation, process must be performed between the same data types (between word, bit, or long access)

E.g. \$SW0L=\$SW10L&\$W20L;'Use all long access of word'

Multiple operations can be combined.

E.g. A=(B+C)\*(D+E/2)

The priority of orders is as follows.

Item	Symbol
High	( )
↑	~
	*, /, %
	+, -
	<<, >>
	&
	^
↓	
Low	=

## Section 1 Outline of Macro Function

### NS series Macro Reference

#### Functions

The following functions are provided for macro of NS Series.

	Action	Function
Conversion between BCD and BIN	Value (BIN code)->BCD code	BCD
	BCD code->value (BIN code)	BIN
Manipulating character strings	Copy character string	STRCPY/STRCPYW
	Convert from ASCII code to Unicode	STRM2W
	Convert from Unicode to ASCII code	STRW2M
Alarm/Event summary	Clear the number of Alarm/Event occurrence	RSTALARMCNT
HMI exclusive statements	Output written value and changed value	GETNUMVAL
	Switch screen	SHOWPAGE/SHOWPAGEBCD
	Movement of object display area	MOVEPARTS
	Display message dialog box	MSGBOX
	Get displayed rectangle of the object	GETPARTS
	Move pop-up window	MOVEPOPW
	Move pop-up window up	MOVEPOPWUP
	Move pop-up window down	MOVEPOPWDOWN
	Move pop-up window left	MOVEPOPWLEFT
	Move pop-up window right	MOVEPOPWRIGHT
	Close pop-up window	CLOSEPOPW
Communications	Read data from specified address	READCMEM
	Write data to specified address	WRITECMEM
Process termination	Terminate macro program	RETURN
Set Date/Time	Change settings of internal clock of the PT	SETTIME
Reading/writing data	Read contents (values in binary) of the specified file in a memory card (CF) to PT memory.	READCF
	Save the contents of PT memory in a memory card (CF).	WRITECF
Data manipulation /conversion	Swap high order and low order of the specified address.	SWAP
	Swap high order (2byte) and low order (2byte) of the specified long word data.	SWAPL
	Copy contents of \$W in the PT memory	MEMCOPY
Input Focus Control	Set the input focus for the specified object	SETFOCUS
	Release the input focus set for the object	RELEASEFOCUS
Repeat Program	Repeat Program	FOR, NEXT
	Aborting from Program Repetition	BREAK
	Return to the top of FOR loop.	CONTINUE

## Section 1 Outline of Macro Function

### NS series Macro Reference

The timing when a function can be executed as shown below.

	Project Macro			Screen Macro		Functional object Macro							
	When Loading a Project	Alarm/Event On Timing	Alarm/Event Off Timing	When Loading a screen	When Unloading a screen	Touch On Timing	Touch Off Timing	When changing value	Before Inputting Numeral/String	When writing Numeral/String	List Selection	When pressing a Display Area	When Selecting an Alarm/Event
BCD	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
BIN	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
CLOSEPOPW		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
GETNUMVAL								*		*			
GETPARTS				OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MOVEPARTS				OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MOVEPOPW		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MOVEPOPWDOWN		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MOVEPOPWLEFT		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MOVEPOPWRIGHT		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MOVEPOPWUP		OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MSGBOX	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
READCMEM		OK	OK		OK	OK	OK	OK	OK	OK	OK	OK	OK
RETURN	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
RSTALARMCNT	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
SHOWPAGE	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
SHOWPAGEBCD	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
STRCPY(W)	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
STRM2W	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
WRITECMEM		OK	OK		OK	OK	OK	OK	OK	OK	OK	OK	OK
SETTIME	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
READCF	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
WRITECF	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
MEMCOPY	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
SWAP	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
SWAPL	OK												
SETFOCUS		OK	OK			OK	OK	OK			OK	OK	OK
RELEASEFOCUS		OK	OK			OK	OK	OK			OK	OK	OK

\*Numeral Display & Input Only

## Section 1 Outline of Macro Function

### *NS series Macro Reference*

#### Added Functions

Action	Function	Remarks
Set Date/Time	SETTIME	Added in NS-Designer Ver.3.0
Reading/writing data	READCF	Added in NS-Designer Ver.4.0 (NS Series hardware with – V1 suffix)
	WRITECF	
Data manipulation /conversion	SWAP	
	SWAPL	
	MEMCOPY	
Input Focus Control	SETFOCUS	
	RELEASEFOCUS	
Switch Screen (BCD)	SHOWPAGEBCD	Added in NS-Designer Ver.6.0.
Repeat Program	FOR, NEXT	
	BREAK	
	CONTINUE	

## **Section2 Functions**

This section describes how to use the standard functions.

2-1	Table of Function and Argument.....	2-2
2-2	Details of the Function.....	2-4



## 2-1 Table of Function and Argument

The variety of variables and values, which can be specified as an argument for macro function is described below. Alphabets such as S,D,n,x,y in the row indicated parameter used in “2-2” Details of function”-“Format”.

Function	Argument	PT Memory				Constant	String	Host Side Address	Specify Index
		\$B \$HB \$SB	Numerical access of bit	\$W \$HW \$SW	Long access of word				
BCD	S		○	○	○	○			○
BIN	S		○	○	○	○			○
CLOSEPOPW	n		○	○	○	○			○
GETNUMVAL	None								
GETPARTS	N		○	○	○	○			○
	Left, Top, Right, Bottom			○	○				○
MOVEPARTS	n		○	○	○	○			○
	X		○	○	○	○			○
	Y		○	○	○	○			○
MOVEPOPW	n		○	○	○	○			○
	x		○	○	○	○			○
	y		○	○	○	○			○
MOVEPOPWDOWN	n		○	○	○	○			○
	y		○	○	○	○			○
MOVEPOPWLEFT	n		○	○	○	○			○
	x		○	○	○	○			○
MOVEPOPWRIGHT	n		○	○	○	○			○
	y		○	○	○	○			○
MOVEPOPWUP	n		○	○	○	○			○
	y		○	○	○	○			○
MSGBOX	S1			○	○		○		○
	S2			○	○		○		○
	S3		○	○	○	○			○

Section2 Functions

NS series Macro Reference

Function	Argument	PT Memory				Constant	String	Host Side Address	Specify Index
		\$B \$HB \$SB	Numerical access of bit	\$W \$HW \$SW	Long access				
READCMEM	D	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	S	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>
	N					<input type="checkbox"/>			
RETURN	S		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
RSTALARMCNT	S		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
SHOWPAGE	S		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
SHOWPAGEBCD	S		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>
STRCPY(W)	D			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	S			<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>
STRM2W	D			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	S			<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>
STRM2M	D			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	S			<input type="checkbox"/>	<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>
WRITECMEM	D	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>			<input type="checkbox"/>	<input type="checkbox"/>
	S	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	n	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>				
SETTIME	S			<input type="checkbox"/>					
READCF	Mem			<input type="checkbox"/>					<input type="checkbox"/>
	Size			<input type="checkbox"/>					
	File			<input type="checkbox"/>			<input type="checkbox"/>		
	Dev			<input type="checkbox"/>		<input type="checkbox"/>			
WRITECF	Mem			<input type="checkbox"/>					<input type="checkbox"/>
	Size			<input type="checkbox"/>					
	File			<input type="checkbox"/>			<input type="checkbox"/>		
	Dev			<input type="checkbox"/>		<input type="checkbox"/>			
SWAP	S			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	n			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
SWAPL	S			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	n			<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>			
MEMCOPY	S			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	D			<input type="checkbox"/>	<input type="checkbox"/>				<input type="checkbox"/>
	N			<input type="checkbox"/>					<input type="checkbox"/>
SETFOCUS	N			<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	
RELEASEFOCUS	None								

## 2-2 Details of the Function

This section describes details of standard functions used in Macro programming.

### BCD Converts the Value to BCD code

Applicable versions	System Version 2 or higher
Format	BCD(S)
Function	Convert value "s" to BCD code Converting range is 0 to 99999999 If you specify the character string outside of the range, overflow occurs. "0" is set at the end of the character string.
Return Value	BCD code
Example	\$W0 = 1234; 'Set value 1234 to \$W0 \$W10 = BCD(\$W0); 'Set BCD code (H1234) to \$W10 \$W20L = 12345678; 'Set value 12345678 to \$W20 to \$W21 \$W22L = BCD(\$W20L); 'Set BCD code (H12345678) to \$W22 to W23

### BIN Converts BCD code to Numeral value

Applicable versions	System Version 2 or higher
Format	BIN(S)
Function	Convert BCD code S to numeral value Converting range is H0 to H99999999
Return Value	Numeral value
Example	\$W0 = H1234; 'Set BCD code (H1234) to \$W0 \$W10 = BIN(\$W0); 'Set 1234 to \$W10 \$W20L = H334455; 'Set BCD code (H334455) to \$W20 to \$W21BCD \$W22L = BIN(\$W20L); 'Set 334455 to \$W22 to \$W23

**BREAK**      **Aborts from program repetition**

Applicable versions	System version 6 or higher
Format	BREAK
Function	Interrupt a loop program between "FOR and NEXT".
Return Value	None
Example	If \$W100I0>30 is true, exit FOR loop.

```

$SW27=0;
FOR(10)
  $W100I0=$W50I0+10;
  IF($W100I0>30)
    BREAK;
  ENDF
  $SW27=$SW27+1;
NEXT;

```

\*Setting range for "n" is 0 to 32767. A negative number is considered as 0. \$W, \$HW and \$SW can specified as an address.

**CLOSEPOPW**      **Closes pop-up window**

Applicable versions	System version 2 or higher
Format	CLOSEPOPW(n)
Function	Close pop-up window screen page number "n" Setting range for "n" is 0 to 3999. If you set pop-up screen page number that does not exist, the process will not be performed.
Return Value	None
Example	CLOSEPOPW(15);Close pop-up screen page 15

**CONTINUE**      **Repeats program**

Applicable versions	System version 6 or higher
Format	CONTINUE
Function	During a program between “FOR to NEXT”, it will return to the top of the FOR loop and resume the FOR process.
Return Value	None
Example	If '\$W50I0>30 is true, a loop will return to the top, and resume the next repetitious program.

```

$SW27=0;
FOR(10)
  IF($W50I0>30)
    $SW27=$SW27+1;
  CONTINUE;
ENDIF
$W100I0=$W50I0+10;
$SW27=$SW27+1;
NEXT;

```

\*Setting range for “n” is 0 to 32767. A negative number is considered as 0. \$W, \$HW and \$SW can be specified as an address.

**FOR(n), NEXT n; number of iteration**

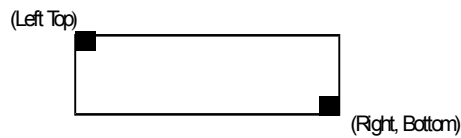
Applicable versions	System version 6 or higher
Format	FOR(n), NEXT n; a LOOP
Function	A series of statements in a computer program that are to be executed repeatedly at specified times. A program between "FOR to NEXT" cannot be nested in another "FOR to NEXT". (Single loop only)
Return Value	None
Example	<p>Execute a loop "FOR to NEXT" 10 times and substitute \$W0~\$W9 to the initial value,0.</p> <pre> \$W0=0; \$SW27=0; FOR(10)   \$W0I0=0;   \$SW27=\$SW27+1; NEXT; </pre> <p>*Setting range for "n" is 0 to 32767. A negative number is considered as 0. \$W, \$HW or \$SW can be specified as an address.</p>

**GETNUMVAL Outputs written value and changed value**

Applicable versions	System version 2 or higher
Format	GETNUMVAL( )
Function	<p>Get writing numeral value or changing numeral value for numeral display &amp; input object.</p> <p>Use this function for "Before writing numeral" or "When changing numeral" at "Macro Execution Condition" in numeral display &amp; input object.</p>
Return Value	Input numeral value
Example	<pre> \$W0=GETNUMVAL( ); 'Set value for writing numeral value to \$W0 </pre>

**GETPARTS** Gets displayed rectangle of the object

Applicable versions	System version 2 or higher
Format	GETPARTS(n, Left, Top, Right, Bottom)
Function	Get displayed rectangle of the object ID number "n". Set coordinate (Left, Top) at the upper left, (Right, Bottom) at the lower right on rectangle.



Setting range is 0 to 1023. If you set other value or ID number that does not exist, return value 1 will be returned.

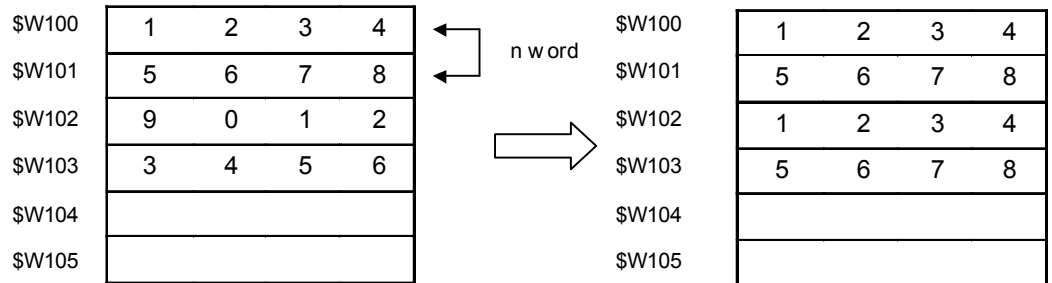
Return Value	0: Completed normally -1:Specified no object
Example	GETPARTS(1, \$W0, \$W1, \$W2, \$W3); 'Set coordinate of displayed rectangle of object ID number1 'to (\$W0, \$W1)-(\$W2, \$W3)

**MEMCOPY**      **Copies contents of \$W in the PT memory**

Applicable versions	System version 4 or higher
Format	MEMCOPY (S, D, n);
Function	<p>Copy data of \$W or \$HW in the PT memory.          S: Top address of source data.          D: Top address to which data will be copied.          n: The number of word data which will be taken from S.          Setting range is as follows:              When specifying “n” directly: 1 to 32767              When specifying “n” indirectly: \$W0 to \$W32767            \$HW0 to \$HW8191</p> <p>Note: Index can be set when using address to specify for S. Setting range is for \$W is between 0 and 32767 and for \$HW is between 0 to 8191.</p>
Return Value	None
Example	<p>MEMCOPY(\$W100, \$W102, 2);          Take 2 words from \$W 100 and copy to 102</p>

<Before executing MEMCOPY>

<After executing MEMCOPY>





**MOVEPARTS** Moves object display area

Applicable versions	System version 2 or higher
Format	MOVEPARTS (n,x,y)
Function	Move the object ID number “n” to specified coordinate (x, y). Specify coordinate upper left of the moving object for “x, y”. Setting range for “n” is 0 to 1023. If the value out side the range or ID number that does not exist is specified, return value –1 will be returned. There is no restriction on setting value for “x, y”. However, set the value for the coordinate of x and y in order that the objects are displayed inside of the screen. All objects or some objects on the screen may be deleted depending on the set value so care must be taken.
Return Value	0: Completed normally -1: Specified no object
Example	MOVEPARTS (3, 150, 200); Move the object ID number 3 to position (150,200)

**Reference**

When setting macro “MOVEPARTS” for ON/OFF button, Word button and Command button and moving these buttons or these buttons with frame, select “Touch Off Timing”. If “Touch On Timing” is selected, the status of the button will be pressed.

Video Display object cannot be moved using MOVEPARTS.

**MOVEPOPW Moves pop-up window**

Applicable versions	System version 2 or higher
Format	MOVEPOPW(n,x,y)
Function	Moves top left of the pop-up window for screen number "n" to the specified coordinates (x, y). Setting range for "n" is 0 to 3999. If the value outside of the range or screen number that does not exist is specified, return value "-1" will be returned. There is no restriction on setting the value for "x, y". However, set the value for the coordinate of x and y in order that the objects are displayed inside of the screen. Part of the screen or whole screen may be deleted depending on the set value.
Return Value	0: Completed normally -1: Specified no page
Example	\$W0=MOVEPOPW(10, 140, 160); 'Moving pop-up screen page number 10 to the specified position (140, 160), then return "0" to \$W0. 'If pop-up screen is not displayed, return "-1" to \$W0.

**MOVEPOPWDOWN Moves pop-up window down**

Applicable versions	System version 2 or higher
Format	MOVEPOPWDOWN(n, y)
Function	Move the pop-up window page number "n" to y down. Setting range for "n" is 0 to 3999. If the value outside of the range or page number which does not exist is specified, return value "-1" will be returned. There is no restriction on setting value for "y". However, set the value to the pop-up screen in order to be displayed inside of the screen. Part of the screen or whole screen may be deleted depending on the set value.
Return Value	0: Completed normally -1: Specified no page
Example	\$W0=MOVEPOPWDOWN(10, 32); 'Move the pop-up window page number 10 to 32dot down, 'then return "0"to \$W0. If pop-up screen is not displayed, 'return "-1" to #W0.

**MOVEPOPWLEFT Moves pop-up window to the left**

Applicable versions	System version 2 or higher
Format	MOVEPOPWLEFT (n, x)
Function	Move the pop-up window page number “n” to x dot left. Setting range for “x” is 0 to 3999. If the value outside of the range or page number that does not exist is specified, return value “-1” is returned. There is no restriction on setting the value for “x”. However, set the value in order to be displayed inside of the screen. It may be deleted part of the screen or whole screen depends on the value.
Return Value	0: Completed normally -1: Specified no page
Example	\$W0=MOVEPOPWLEFT (10, 32); 'Move pop-up window page number 10 to 32 dot left, then return "0" to \$W0. If pop-up screen is not displayed, return "-1" to \$W0.

**MOVEPOPWRIGHT Moves pop-up window to the right**

Applicable versions	System version 2 or higher
Format	MOVEPOPWRIGHT(n, x)
Function	Move pop-up window page “n” to x dot right. Setting range for “n” is 0 to 3999. If the value outside the range or screen number that does not exist is specified, return value “-1” is returned. There is no restriction on setting for “x”. However, set the value in order to be displayed inside of the screen. Part of the screen or whole screen may be deleted depending on the set value.
Return Value	0: Completed normally -1: Specified no page
Example	\$W0=MOVEPOPWRIGHT (10,32); 'Move the pop-up screen page 10 to 32 dot right, then return "0" to \$W0. If the pop-up screen is not displayed, return "-1" to \$W0.

**MOVEPOPWUP** Moves pop-up window up





Applicable versions	System version 2 or higher
Format	MOVEPOPWUP (n, y)
Function	Move the pop-up screen page “n” to y dot up. Setting range for “n” is 0 to 3999. If the value outside of the range or page number that does not exist is specified, return value “-1” is returned. There is no restriction on setting the value for “y”. However, set the value in order to be displayed inside of the screen. Part of the screen or whole screen may be deleted depending on the set value.
Return Value	0: Completed normally -1: Specified no page
Example	\$W0=MOVEPOPWUP (10,32); 'Move pop-up screen page 10 to 32 dot up, then return “0” to \$W0. If pop-up screen is not displayed, return “-1” to \$W0.

**MSGBOX**                      **Displays message dialog box**

Applicable versions	System version 2 or higher
---------------------	----------------------------

Format	MSGBOX (S1, S2, S3)
--------	---------------------

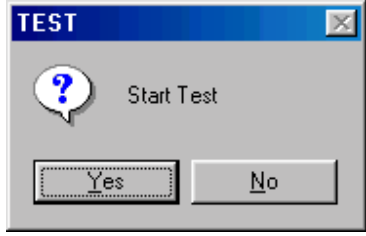
Function	<p>Display message dialog which is specified.                  S1: message string                  S2: title bar string                  S3: reply with icon type that is displayed in message dialog                  Specify type of the button.                  Rightmost 8 bits (B0-B7)</p>
----------	--

0:	 STOP Mark	1:	 QUESTION Mark
2:	 EXCLAMATION Mark	3:	 INFORMATION Mark

Leftmost 8 bits (B8-B15)  
 0:[OK] button only                      1:[OK]/[Cancel]button  
 2:[Retry]/[Cancel]button  
 3:[Yes]/[No]button  
 4:[Yes]/[No]/[Cancel]button  
 5:[Stop]/[Retry]/[Ignore]button

Return Value	0:Select [OK] button                      1:Select [Cancel]button 2:Select [Yes]button                      3:Select [No]button 4:Select [Ignore]button                      5:Select [Retry]button 6:Select [Stop]button
--------------	--

Example	<pre> \$W0=MSGBOX ("Start Test", "TEST", H31); 'H31:[Yes]/[No] button, Display QUESTION mark IF(\$W0==2)                                 'Write processing if you select "Yes" ELSE                                 'Write processing if you select "No" ENDIF                 As you set above, message dialog is displayed below             </pre>
---------	---



## Reference

Only one message box can be displayed using MSGBOX. If executing MSGBOX with displaying other message box, new message box is not displayed and "1" is returned as return value.

Example: making two bit lamps

	address	macro
Lamp 1	\$B0	\$W0=MSGBOX("message1", "title1", H31);
Lamp 2	\$B0	\$W1=MSGBOX("message2", "title2", H31);

Suppose macro of lamp1 is executed first. When changing the value of \$B0, the message box of lamp 1 is displayed. Message box of lamp 2 is not displayed and value "1" is stored in \$W1.

If Buzzer Sound is set ON or OFF at PT tab in the system menu and STOP or EXCLAMATION is specified for the icon, buzzer will be sounded when displaying the message dialog.

## READCF Reads the contents (values in binary) of the specified file in a memory card (CF) to PT memory

Applicable versions	System version 4 or higher
Format	READCF(Mem, Size, File, Dev)
Function	<p>Reads the contents (binary format) of the specified file in a Memory Card to PT memory.</p> <p>Mem: Top address of destination. (\$W, \$HW or \$SW) Index can be set. Setting range for \$W is between 0 and 32767 and for \$HW is between 0 to 8191. \$B, \$HB and \$SB cannot be specified.</p> <p>Size: Data size to be read. (unit: word) Data size can be specified using long word directly, or \$W or \$HW(uses 2 words) indirectly. Setting range for \$W is 0 to 32767 and for \$HW is between 0 and 8191. If the set value is exceeded the maximum, an access error occurs and an error message appears. If the value 0 or less is set for Size, the specified size will be read to the PT memory. If the file size is bigger than the value set for Size (Size&gt;0), it executes reading data set for Size. If the file size is smaller than the value set for Size(Size&gt;0), it executes reading actual file size only.</p> <p>File: Source file name The file name can be specified using character string directly, or \$W or \$HW (uses 2 words) indirectly. Read action is executed by word unit, however, the last 1 byte of address will not be read if file size is odd byte. (Return value will be 0). Up to 43 alphanumerical characters ("0 to 9", "A to Z", "a to z", "\$", "_") including extension can be set for the file name.</p> <p>Dev: Specify destination device. Always specify 0 since destination will be a memory card only.</p>
Return Value	<p>0: Completed normally</p> <p>-1: Failed to read data</p>
Example	<pre>1. \$W100=READCF(\$W1000,0", CF_FILE.BIN",0); 2. \$W2000L=0;    STRCOPY(\$W2002, "CF_FILE.BIN");    \$W100=READCF(\$W1000,\$W2000,\$W2002,0);</pre>

**READCMEM**      **Reads the data from specified address**

Applicable versions	System version 2 or higher
Format	READCMEM (D,S,n)
Function	Read n but/n channel from the address in the host specified with "s", and copy to "D". Maximum points for reading are indicated below.

Bit	126Bit
Word	126Channel

If value outside of the range is set for "n", communication error or macro execution error occurs.

Maxim points of reading depend on PLC type.

Return Value	None
Example	READCMEM (\$W0, [HOST1:DM0], 10) 'Read the value "\$W0 to \$W9" to "DM0 to DM9" at the host 'named Host1 in PLC.



**RELEASEFOCUS** Releases the input focus set for the object

Applicable versions	System version 5 or higher
Format	RELEASEFOCUS(); No argument is used.
Function	<p>If the input focus has been set for any of numeral display &amp; input object or string display &amp; input object in the project, this macro will release the input focus.</p> <ul style="list-style-type: none"> <li>- If an object that the input focus has been set exists on the screen currently displayed, the macro will release the input focus.</li> <li>- If the input focus is not set for any object on the screen currently displayed, this macro will NOT work.</li> <li>- If the input focus has been set for the object created in the frame page displayed as top, this macro will also release the focus.</li> <li>- If the input focus has been set for the object in the sheet, this macro will release the focus.</li> </ul> <p>RELEASEFOCUS macro will NOT work in the following execution timing.</p> <ul style="list-style-type: none"> <li>-When Loading a Project</li> <li>-When Loading a Screen</li> <li>-When Unloading a Screen</li> <li>-Before Inputting Numeral set using numeral display &amp; input objects</li> <li>-Before Writing Numeral set using numeral display &amp; input objects</li> <li>-Before Inputting String set using string display &amp; input objects</li> <li>-Before Writing String set using string display &amp; input objects</li> </ul>
Return Value	None
Example	In all cases that you want to release the input focus, set as the following example. RELEASEFOCUS();

**RETURN Terminates Macro program**

Applicable versions	System version 2 or higher
Format	RETURN(S)
Function	If the value of "S" is "0", terminate macro program and continue to process for functional object. If a value is set other than "0", terminate program and stop processing for functional object.
Return Value	None
Example	RETURN(0); 'terminate macro and continue to process RETURN(1); 'terminate macro and stop processing

**RSTALARMCNT Clear the number of occurrence of Alarm/Event**

Applicable versions	System version 2 or higher
Format	RSTALARMCNT(S)
Function	When the value of "S" is 0, clear the number of occurrence of alarm. When the value of "s" is 1, clear the number of occurrence of event.
Return Value	0: Completed normally -1: None
Example	RSTALARMCNT(0); 'clear the number of occurrence of alarm RSTALARMCNT(1); 'clear the number of occurrence of event

**SETFOCUS**      **Sets the input focus set for the object**

Applicable versions	System version 5 or higher
Format	SETFOCUS(n);
Function	<p>Set the input focus on the specified numeral display &amp; input object or string display &amp; input object.  n: Object ID number which the input focus should be set.  (0 to 32767)</p> <p>When setting the input focus for the object specified as top in the <b>Input Order List</b>, set "-1".</p> <ul style="list-style-type: none"> <li>- This macro will NOT work if the input focus has already been set for other object.</li> <li>- The input focus will NOT be set if the specified object is created in the frame page which is not displayed as top. (A dialog which indicates macro execution error will be displayed when executing this macro.)</li> <li>- If an object other than numeral display &amp; input object and string display &amp; input is specified, this macro will NOT work.</li> <li>- The input focus cannot be set for the objects created in the sheet.</li> </ul> <p>SETFOCUS macro will NOT work in the following execution timing.</p> <ul style="list-style-type: none"> <li>-When Loading a Project</li> <li>-When Loading a Screen</li> <li>-When Unloading a Screen</li> <li>-Before inputting Numeral set using numeral display &amp; input objects</li> <li>-Before Writing Numeral set using numeral display &amp; input objects</li> <li>-Before inputting String set using string display &amp; input objects</li> <li>-Before Writing String set using string display &amp; input objects</li> </ul>
Return Value	0: Completed normally -1: The specified object ID could not be found.
Example	Case that the input focus is set for object with ID number 4. SETFOCUS(4);

**SETTIME**      **Changes settings of internal clock of the PT**

Applicable versions	System version 3 or higher
Format	SETTIME(S)
Function	<p>Preset values for the specified address as S and writes them to the internal clock of PT. Setting range for S is between \$W0 and \$W32765 or between \$HW0 and \$HW8189. (See note)</p> <p>Set the value in BCD format for addresses to be written.</p> <p>Uses 3 words regarding the specified address as top.</p> <p>Note: Index can be used for specifying the address.</p>
Return Value	None
Example	<p>Case that December 31, 2002 18:59:20 is set.</p> <p>\$W 100=H5920; \$W101=H3118; \$W102=H0212; SETTIME(\$W100);</p>

**SHOWPAGE**      **Switches screen**

Applicable versions	System version 2 or higher
Format	SHOWPAGE(S)
Function	<p>Switch screen to the page that is specified in "s".</p> <p>Setting range for "s" is 0 to 3999. If the value outside of the range is set, macro execution error occurs. If the screen number that does not exist is set, reading page error occurs.</p>
Return Value	None
Example	SHOWPAGE(10); 'Switch screen to page 10

**Reference**

Macro written after SHOWPAGE is not executed. Be sure to write SHOWPAGE at the end of line.

Bad Example:

```
SHOWPAGE(10); <-Switch to page 10
$W50=100;      <-Substitute 100 to $W50 is not executed
```

Good Example:

```
$W50=100;      <-Substitute 100 to $W50 is executed
SHOWPAGE(10); <-Switch to page 10
```

**SHOWPAGEBCD** Switches screen to the screen page n.

Applicable versions	System version 6 or higher
Format	SHOWPAGEBCD(S)
Function	S: screen page number (H0 to H3999) Switch screen to the page that is specified in "S". Setting range for "S" is H0 to H3999. If either the value outside of the range is set or an invalid value is set for BCD, macro execution error occurs. If the screen number that does not exist is set, reading page error occurs.
Return Value	None
Example	Switch screen to page 10 SHOWPAGEBCD(H10);  Specify a screen page number indirectly to switch screen to page 10. \$W100=H10; SHOWPAGEBCD(\$W100);

**Reference**

Macro written after SHOWPAGEBCD is not executed. Be sure to write SHOWPAGEBCD at the end of line.

Bad Example:

```
SHOWPAGEBCD(H10); ←Switch to page 10
$W50=100;          ←Substitute 100 to $W50 is not executed
```

Good Example:

```
$W50=100;          ←Substitute 100 to $W50 is executed
SHOWPAGEBCD(H10); ←Switch to page 10
```

**STRCPY(W) Copies Character string**

Applicable versions	System version 2 or higher
Format	STRCPY (D,S) ;ASCII code STRCPYW(D,S) ;Uni code
Function	Copy character string from D to S Copy is performed including null.
Return Value	None
Example	STRCPYW (\$W0",ABC"); 'Set "ABC" to \$W0 to \$W2 \$W100=H6400;STRCPY(\$W110,\$W100); 'Set "d" to \$W110

**Reference**

'null' matches for "00" in ASCII code, and "0000" in Unicode.  
Care must be taken to set string to \$W32767 because copy is performed including null.  
When executing STRCPY(W), string data and null may not be executed \$W32767 because null is copied. (If data is exceeded \$W32767, communication error occurs).

**STRM2W Converts character string from ASCII code to Unicode**

Applicable versions	System version 2 or higher
Format	STRM2W (D,S)
Function	Convert character string specified in "S" from ASCII code to Unicode and copy to "D". Copy is performed including null.
Return Value	None
Example	STRM2W(\$W0", ABC"); 'Convert "ABC" to Unicode, and copy to \$W0 to \$W2

**STRW2M**      **Converts character string from Unicode to ASCII code**

Applicable versions	System version 2 or higher
Format	STRW2M (D,S)
Function	Convert string specified in “S” to ASCII code and copy to “D”. Copy is performed including null.
Return Value	None
Example	STRW2M(\$W0, “ABC”); ‘Convert “ABC “ to ASCII code and copy to \$W0 to \$W1.

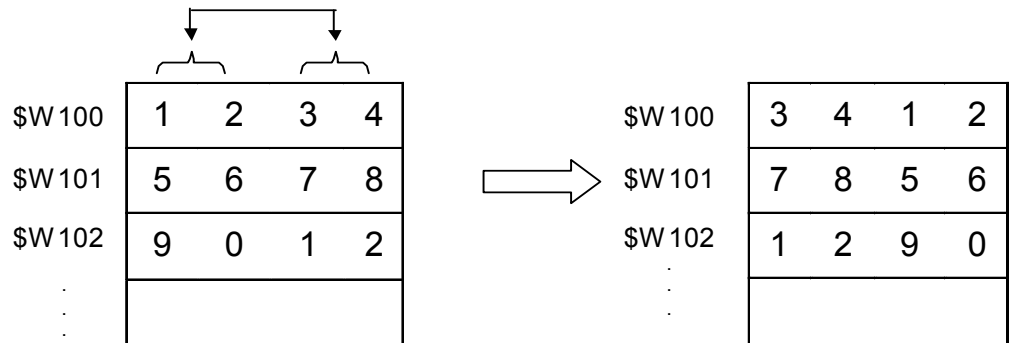
**SWAP** Swaps high order and low order of the specified address

Applicable versions	System version 4 or higher
Format	SWAP(S,n)
Function	<p>Swap high order (1 byte) and low order (1 byte) of the word data or the internal holding word which was taken n word form S.                      S: Top address (\$W or \$HW) to be swapped. (See note.)                      n: The number of words to be taken from S.                      Setting range is as follows:                      When specifying "n" directly: 1 to 32767                      When specifying "n" indirectly: \$W0 to \$W32767                      \$HW0 to \$HW8191</p> <p>Note: Index can be set when using address to specify for S. Setting range is for \$W is between 0 and 32767 and for \$HW is between 0 to 8191.</p>
Return Value	None

Example	<p>SWAP(\$W100,3);                      Swap high order and low order of the word data which was taken from 3 words from \$W100.</p>
---------	--

<Before performing SWAP>  
 Swap high order and low order

<After performing SWAP>







**WRITECF**      **Saves the contents of PT memory in a memory card (CF)**

Applicable versions	System version 3 or higher
Format	WRITECF (Mem, Size, File, Dev)
Function	<p>Saves the contents of PT memory in the specified file of Memory Card. The contents of the specified address will be written to the file in binary format.</p> <p>Mem: Top address of source data. (\$W, \$HW or \$SW) Index can be used. Setting range for \$W is between 0 and 32767, and for \$HW is between 0 and 8191. \$B, \$HB or \$SB can not be specified.</p> <p>Size: Data size to be saved in a Memory Card. (unit; word) Data size can be specified using long word directly, \$W or \$HW (uses 2 words) indirectly. Setting range for \$W is between 0 and 32767 and for \$HW is between 0 and 8191. If the set value has been exceeded the maximum, an access error will occur and an error message will appear</p> <p>File: Destination file name The file name can be specified using character string directly or using \$W or \$HW (uses 2 words) indirectly. If the specified file name for "F" already exists, the file name will be overwritten without showing a confirmation message. (Return value will be 0 (Completed normally)). Up to 43 alphanumerical characters ("0 to 9", "A to Z", "a to z", "\$", "_") including extension can be set for the file name.</p> <p>Dev: Always specify 0 since destination will be a Memory Card only.</p>
Return Value	<p>0: Completed normally</p> <p>-1: Failed to save data</p>
Example	<ol style="list-style-type: none"> <li>1. \$W100=WRITECF(\$W1000, 128, "CF_FILE.BIN", 0);</li> <li>2. \$W2000L=128; STRCPY(\$W2002, "CF_FILE.BIN"); \$W100=WRITECF(\$W1000, \$W2000, \$W2002, 0);</li> </ol>

**WRITECMEM**      **Writes the data to specified address**

Applicable versions	System version 2 or higher
Format	WRITECMEM(D,S,n)
Function	Copy data of n bit/n channel from “S” to the address in the host specified in “D”. Maximum points of writing are indicated below.

Bit	126Bit
Word	126Channel

If the value outside of range is set for “n”, communication error or macro execution error occurs. Maximum points of writing depends on PLC type.

Return Value	None
Example	WRITECMEM([HOST1: DM0], \$W0,10); “Write the value \$W0 to \$W9 at the host named HOST1 in PLC.”

## **Section3 Error Message List**

This section describes error message which is displayed in error list box when macro is added to the project, the screen and the functional objects.

3-1	Error Message List .....	3-2
-----	--------------------------	-----

### 3-1 Error Message List

Error messages are displayed in the error list box after checking the error as shown below.

Error Message	Details	Example
Format error	The program contains description besides variable name, function name, or programming terms that cannot be recognized. Check whether the input function is correct.	\$W0=ABC+100;
Variable name error	Variable name is incorrect	\$B0:3=1;
( is missing	The ((left parentheses symbol ) is missing from a function or sentence	IF\$W0==1)
No. of ( )does not agree	The number of ( ) (parentheses) in the program does not agree	IF(\$W0=1)!(\$W1=0
Position of , is incorrect	The position of the , (comma) is incorrect	IF(\$W0==1),(\$W1==0)
Function argument error	The program contains an incorrect function argument, such as word memory being set in a position that permits bit memory only. Refer to "Section 2 Explanation for the function "-"Correspondence Table of Function and Argument" and check the argument.	\$W0=BCD(\$B0);
= Command error	The program contains an incorrect substitution statement, such as 3=10, \$B0=3	\$W0="ABCDE"
End of program is incomplete	The program that was input is incomplete	\$W10=10+;
If sentence error	The program contains an incorrect IF,ELSE or ENDIF is statement	IF(\$W0==1)!(\$W1==0) \$W10=1; ELSE \$W10=10;
,or; is missing	The number of ,(comma) that divides the argument is insufficient. The program is not divided by a ;(semicolon).	\$W10=1
FOR Statement Error	The number other than between 0 to 32767 is set for n when setting FOR(n) sentence, or there is another loop structure in the FOR sentence (single loop only).	FOR(50000) FOR(3) \$W0=\$W0+1; FOR(5) \$W10=\$10+10; NEXT; NEXT;