

Example of Enhance Indirect Recipe Index Register

Table 23-5-1 Example of enhance recipe index register

	*ENRCP3									
	ENRCP0	0	ENRCP1	0	ENRCP2	0	ENRCP3	65535	ENRCP65535	160

Chapter 24 Macro

This chapter mainly covers the types of macros and macro commands supported by DOPSoft and how to configure macro commands.

◆ Macro types:

Type of Macro	ON Macro
	OFF Macro
	Before Execute Macro
	After Execute Macro
	Screen Open Macro
	Screen Close Macro
	Screen Cycle Macro
	Submacro
	Initial Macro
	Background Macro
	Clock Macro

Table 24-1-1 Types of Macros

DOPSoft provides a list of macro commands for users to perform various operations. They are grouped based on their natures, including the following categories: Arithmetic, Logical, Data transfer, Data conversion, Comparison, Flow control, Bit setting, Communication and Drawing, File Access, and Other macros.

Arithmetic	▶
Logical	▶
Data transfer	▶
Data Conversion	▶
Comparison	▶
FlowControl	▶
Bit Setting	▶
COM port	▶
Drawing	▶
Others	▶

Figure 24-1-1 Categories of Macro Commands

24-1 Types of Macros

Macros consist of independent commands processing procedures written by users. Each macro supports 512 lines of commands and the table below outlines the main features of each type of macro.

Macro Type	Feature
ON Macro	<ul style="list-style-type: none"> ➤ Executed once after ON Macro is triggered. ➤ Only available for ON Button, OFF Button, Maintained Button and Momentary Button
OFF Macro	<ul style="list-style-type: none"> ➤ Executed once after OFF Macro is triggered. ➤ Only available for ON Button, OFF Button, Maintained Button and Momentary Button.
Before Execute Macro	<ul style="list-style-type: none"> ➤ After users touch the onscreen button element, this macro will be executed first before all other procedures programmed for this button element. If the state of the button is not changed by user touches, then this macro will not be executed (ex. Commands of external controllers or other macro commands). ➤ Available for all button elements and input elements.
After Execute Macro	<ul style="list-style-type: none"> ➤ After users touched the onscreen button element, this macro will be executed first before all other procedures programmed for this button element. If the state of the button is not changed by user touches, then this macro will not be executed (ex. Commands of external controllers or other macro commands). ➤ Available for all button elements and input elements.
Screen Open Macro	<ul style="list-style-type: none"> ➤ Executed only once after users open a screen.
Screen Close Macro	<ul style="list-style-type: none"> ➤ Executed only once after users close a screen.
Screen Cycle Macro	<ul style="list-style-type: none"> ➤ Executed continuously. If screen open macro is used, then it will be executed first before this macro.
Sub Macro	<ul style="list-style-type: none"> ➤ One Submacro supports a maximum of 512 submacros and 512 statements can be written within each submacro. ➤ A submacro is similar to a subprogram found in other programming languages. Users can put repeated functions or procedures into a submacro and call it when necessary.
Initial Macro	<ul style="list-style-type: none"> ➤ First executed macro after a HMI system is initialized and this macro is only executed once.
Background Macro	<ul style="list-style-type: none"> ➤ Continuously executed during HMI operations. Whether it is executed one line or several lines at a time (does not stop after the first sequence), this macro will continue to be executed and repeated from the first line after it reaches the last line.
Clock Macro	<ul style="list-style-type: none"> ➤ Continuously executed during HMI operations. This macro will finish executing all commands within the macro rather than one line or several lines at once.

Table 24-1-2 Features of Macro

24-1-1 ON Macro/OFF Macro

On button and off button will only become available after ON button, OFF button, Maintained button, or Momentary button are created.

When a user touches the onscreen button and changes the state to on, the HMI will execute the ON Macro. When the user touches the onscreen button and changes the button state to off, the HMI will execute the OFF Macro. If the button state is not changed by touching the screen button elements (ex. by commands of external controllers or other macros), then On/Off Macro will not be executed.

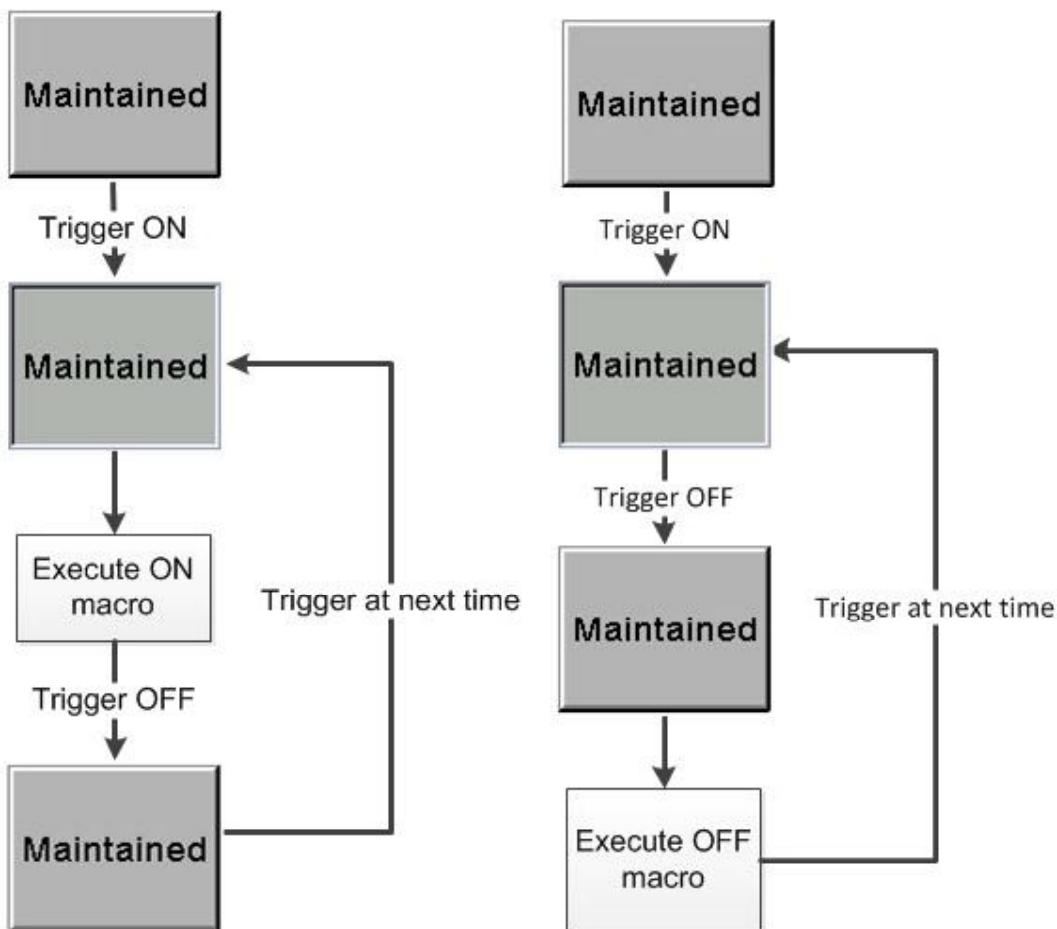


Figure 24-1-1-1 ON/OFF Macro Flowchart

24-1-2 Before Execute Macro

This macro only becomes available after onscreen elements established are button elements or input elements. When users press the onscreen button element, this macro will be executed first before all other procedures programmed for this button element. If the state of the button is not changed by user touches, then this macro will not be executed (ex. Commands of external controllers or other macro commands).

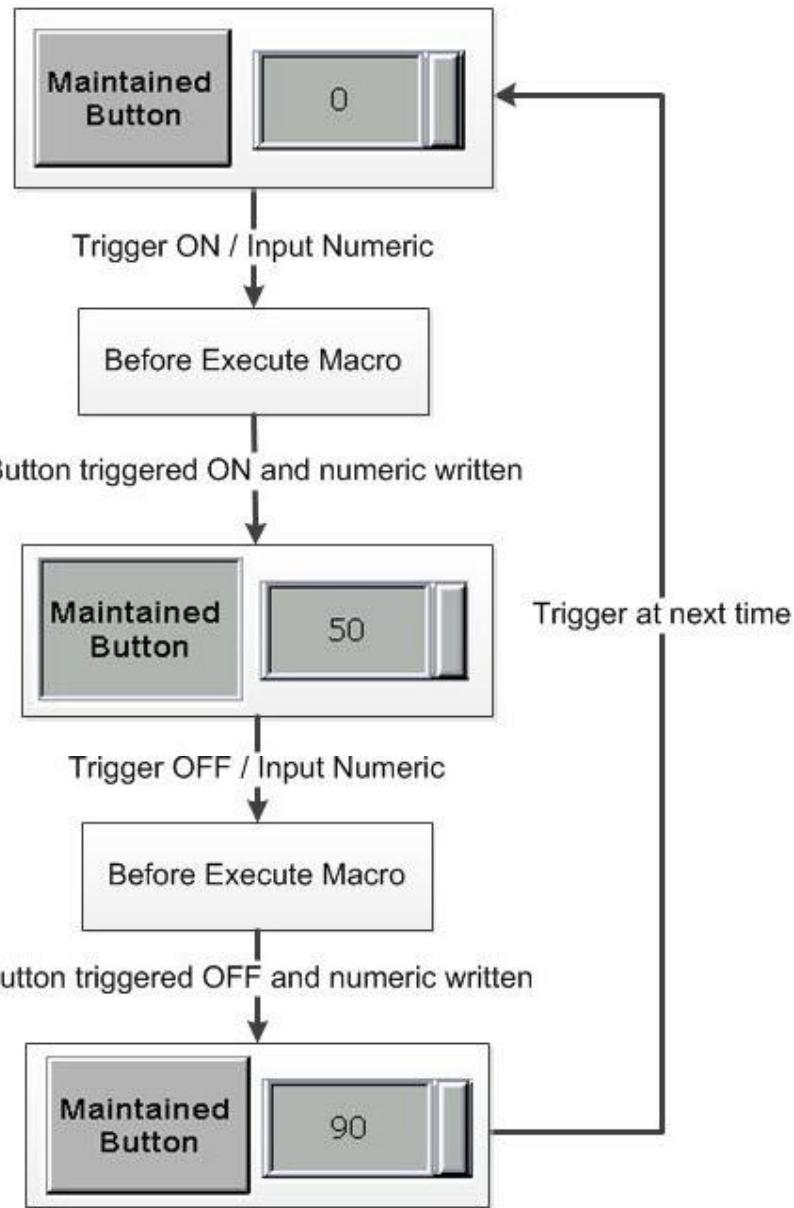


Figure 24-1-2-1 Before Execute Macro Flowchart

24-1-3 After Execute Macro

This macro only becomes available after onscreen elements established are button elements or input elements. When users press the onscreen button element, this macro will be executed first before all other procedures programmed for this button element. If the state of the button is not changed by user touches, then this macro will not be executed (ex. Commands of external controllers or other macro commands).

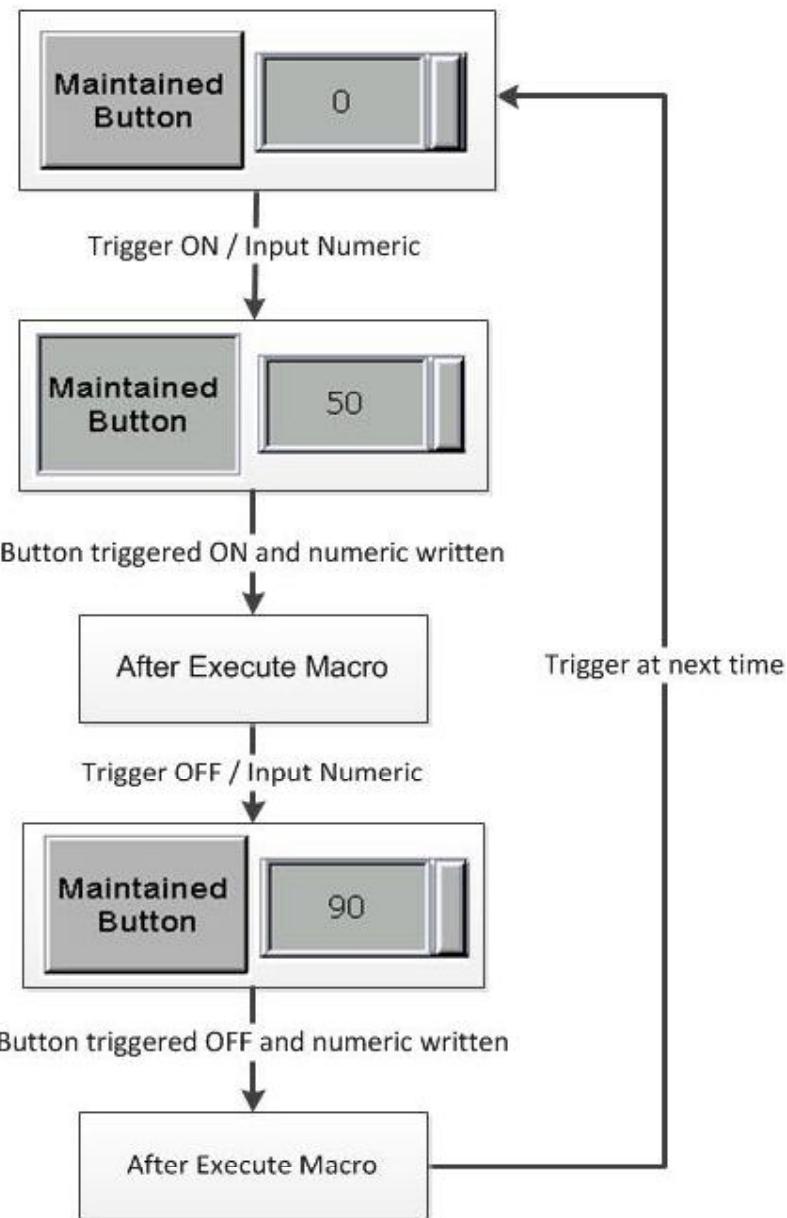


Figure 24-1-3-1 After Execute Macro Flowchart

24-1-4 Screen Open Macro

The Screen Open Macro can be edited by going into [Screen] → [Screen Open Macro].

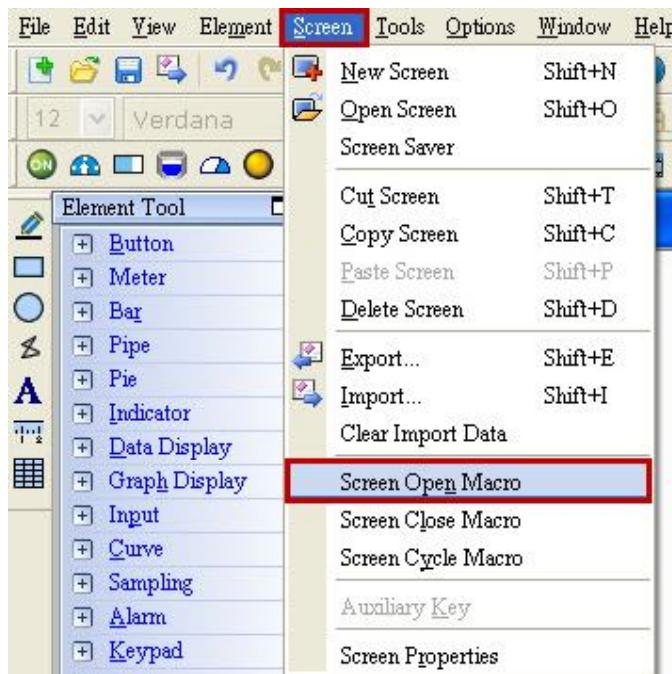


Figure 24-1-4-1 Screen Open Macro

Each screen created within DOPSoft contains a Screen Open Macro and it is executed each time a screen is opened or switched to another. Other macros or commands are executed after the Screen Open Macro.

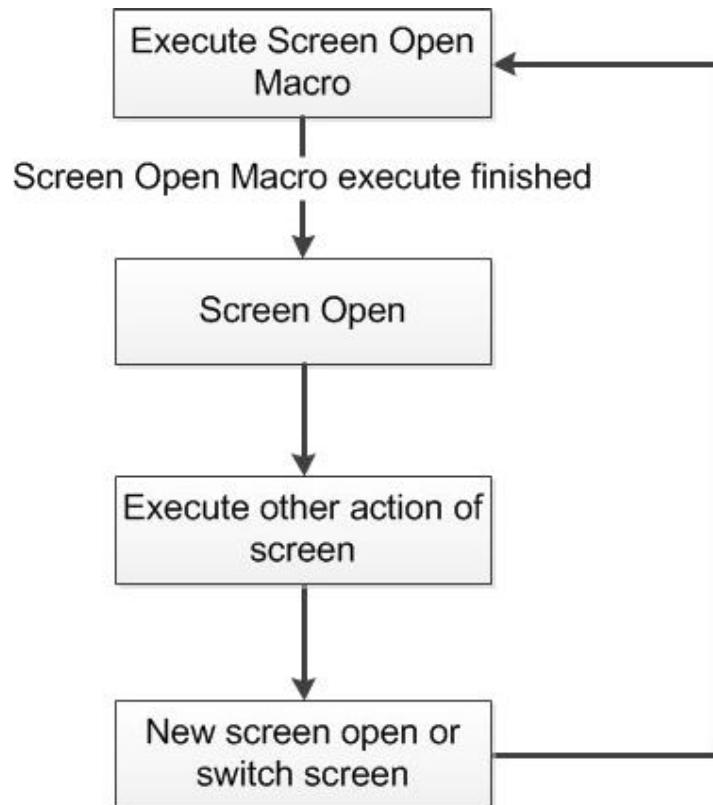


Figure 24-1-4-2 Screen Open Macro Flowchart

24-1-5 Screen Close Macro

The Screen Close Macro can be edited by going into [Screen] → [Screen Close Macro].

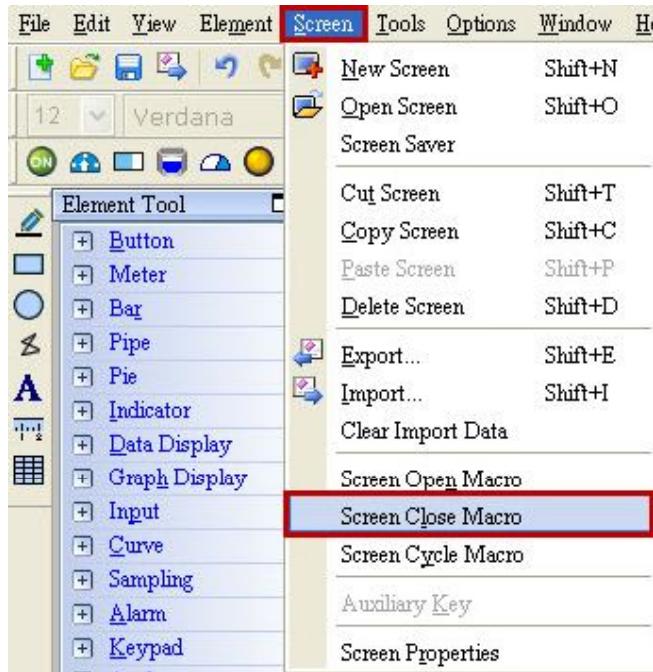


Figure 24-1-5-1 Screen Close Macro

Each screen created within DOPSoft contains a Screen Close Macro and it is executed each time a screen is closed or switched to another. Other macros or commands are executed after the Screen Close Macro.

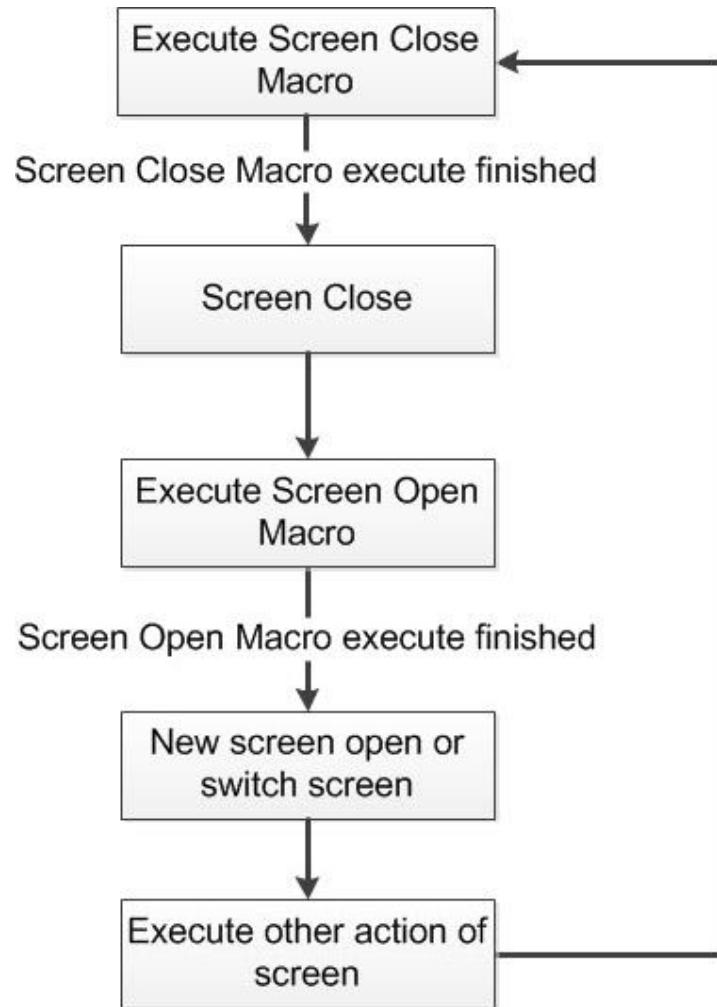


Figure 24-1-5-2 Screen Close Macro Flowchart

24-1-6 Screen Cycle Macro

The Screen Cycle Macro can be edited by going into [Screen] → [Screen Cycle Macro].

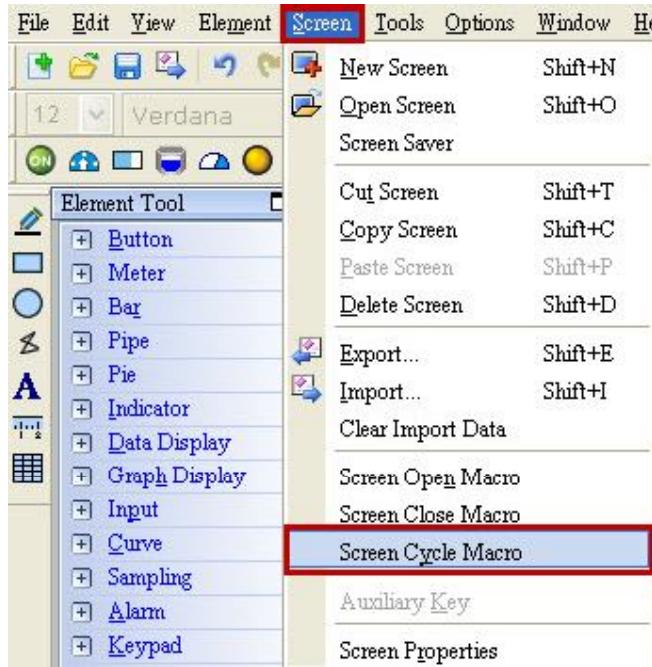


Figure 24-1-6-1 Screen Cycle Macro

Each screen created within DOPSoft contains a Screen Cycle Macro and it is executed, based on the preset delay duration, after the Screen Open Macro. Users can double click on the screen to set the macro cycle delay in the screen properties dialog window. The delay duration refers to how long it will takes before executing the Screen Cycle Macro again. The default unit is 100ms.

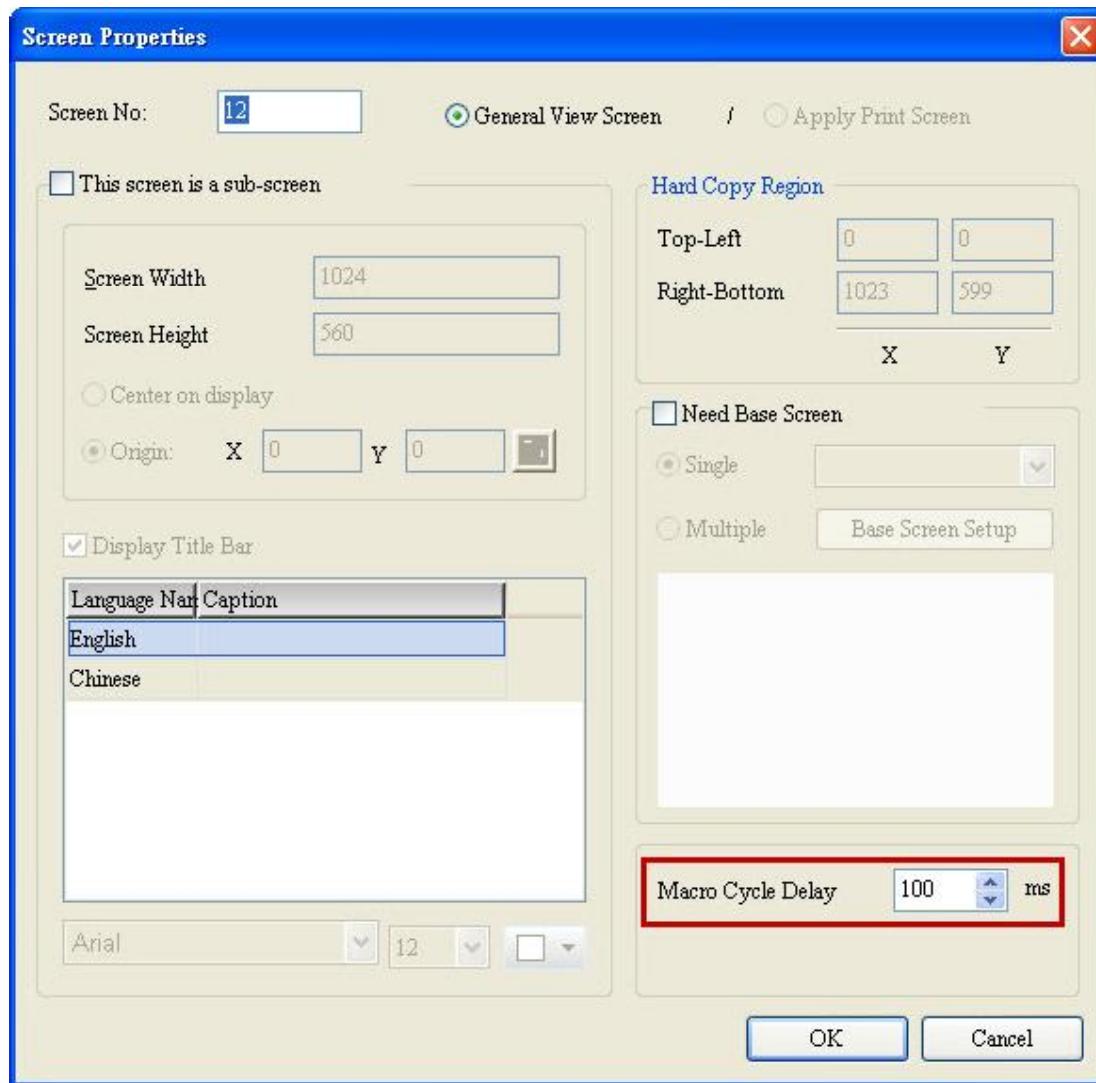


Figure 24-1-6-1 Setting of Macro Cycle Delay

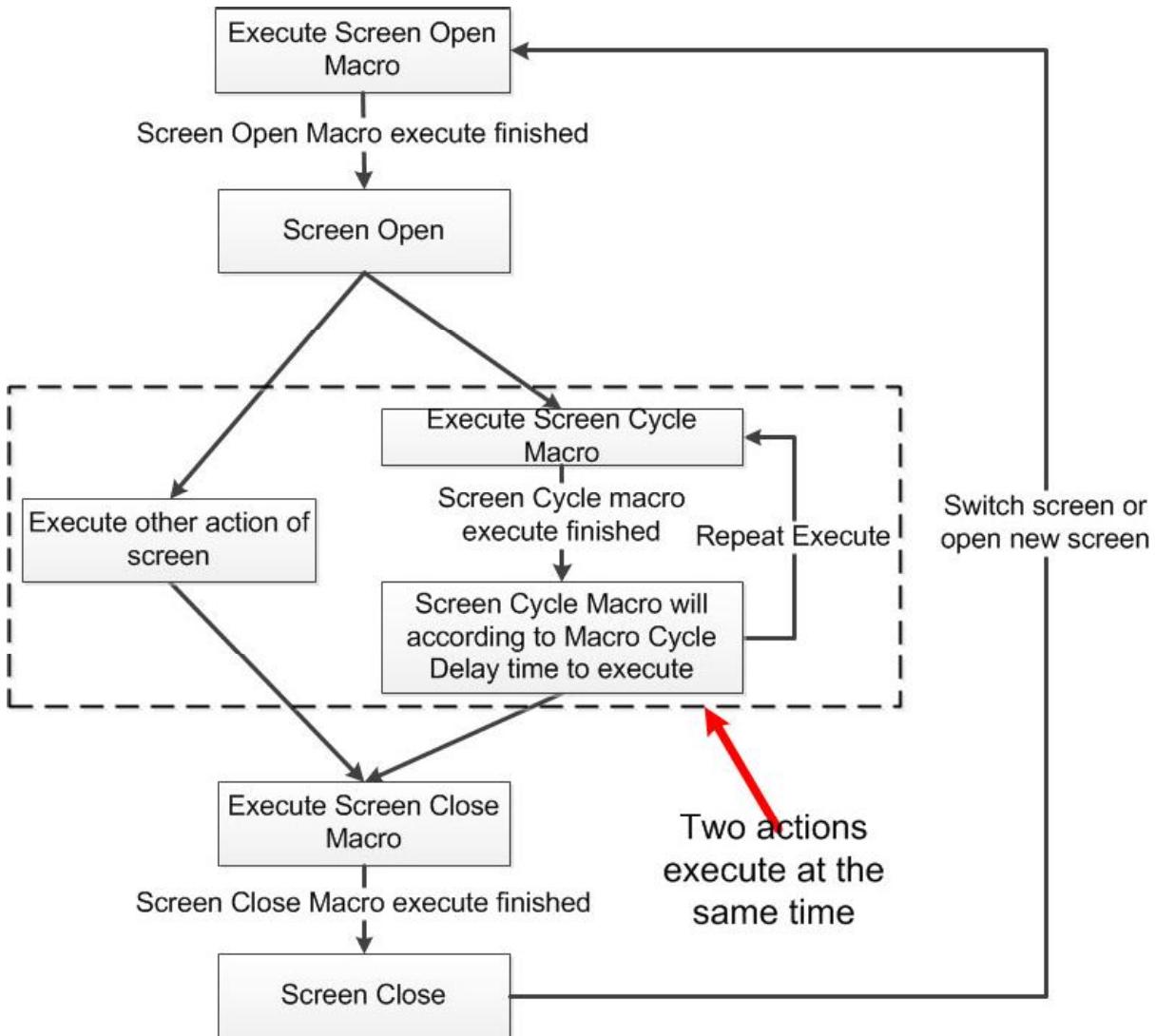


Figure 24-1-6-2 Screen Cycle Macro Flowchart

24-1-7 Submacro

The Submacro can be edited by going into [Options] → [Submacro].

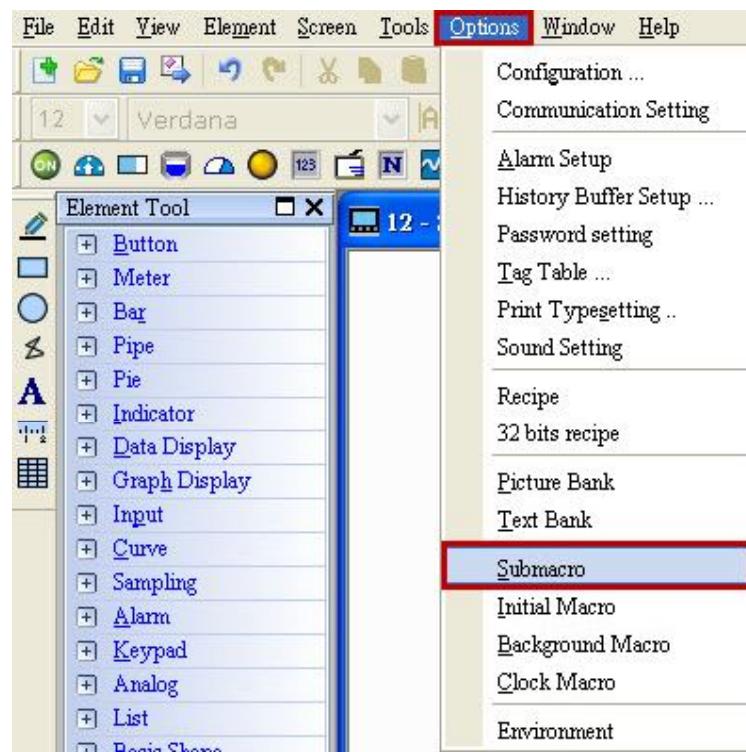


Figure 24-1-7-1 Submacro

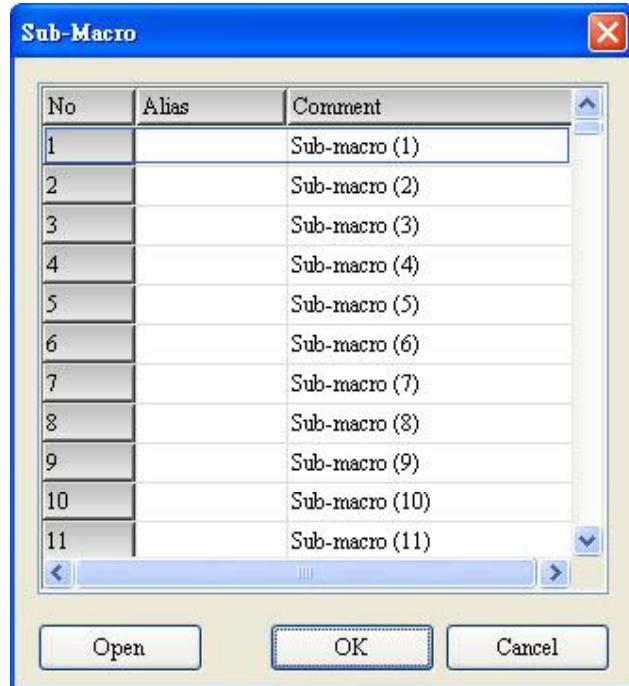


Figure 24-1-7-2 Submacro Configuration Dialog Window

One Submacro supports a maximum of 512 submacros (identified by their number from 1 ~ 512). A submacro is similar to a subprogram found in other programming languages. Users can put repeated functions or procedures into a submacro and call it when necessary. This not only saves time to write repeated macro codes but is also easier to maintain.

Note: only six submacros can be used within one submacro.

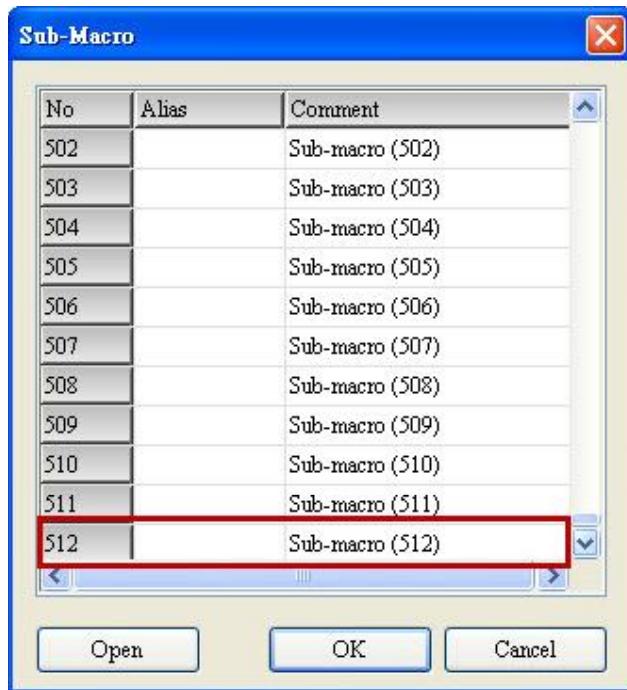


Figure 24-1-7-3 Submacro Screenshot I

Users can call a submacro by its NO or its alias given by users. The name of a submacro can be either numerical value, English, or Chinese characters, and a maximum of 64 characters are allowed for the name.



Figure 24-1-7-4 Submacro Screenshot II



Figure 24-1-7-5 Submacro Screenshot III

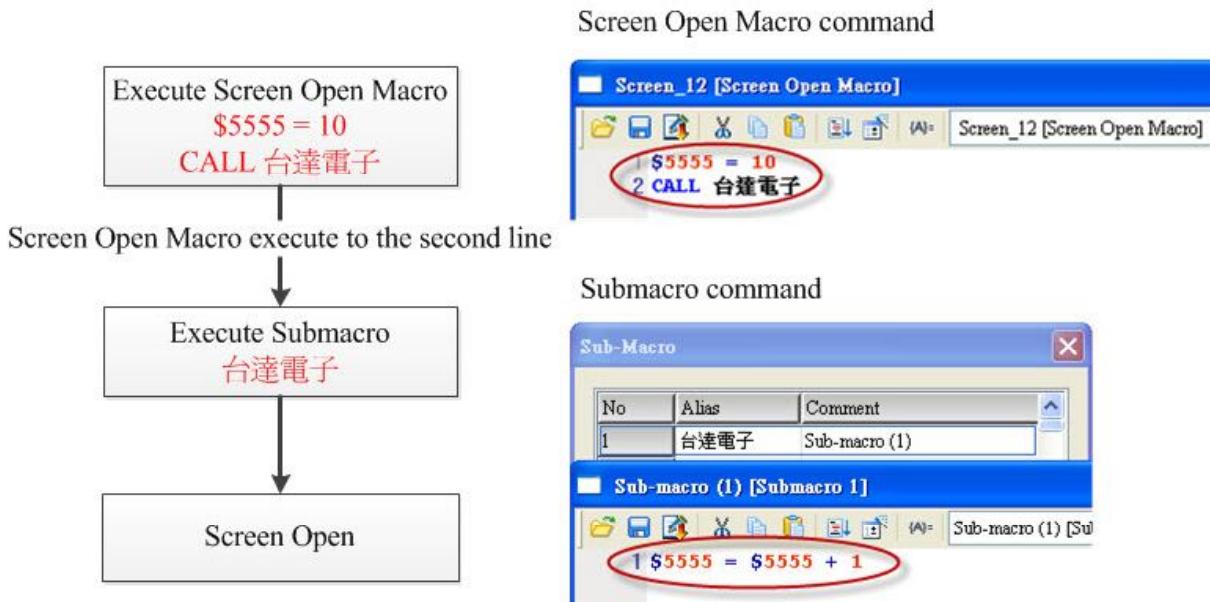


Figure 24-1-7-6 Submacro Flowchart

Submacro also provides the function of password protection. Users can encrypt each submacro.

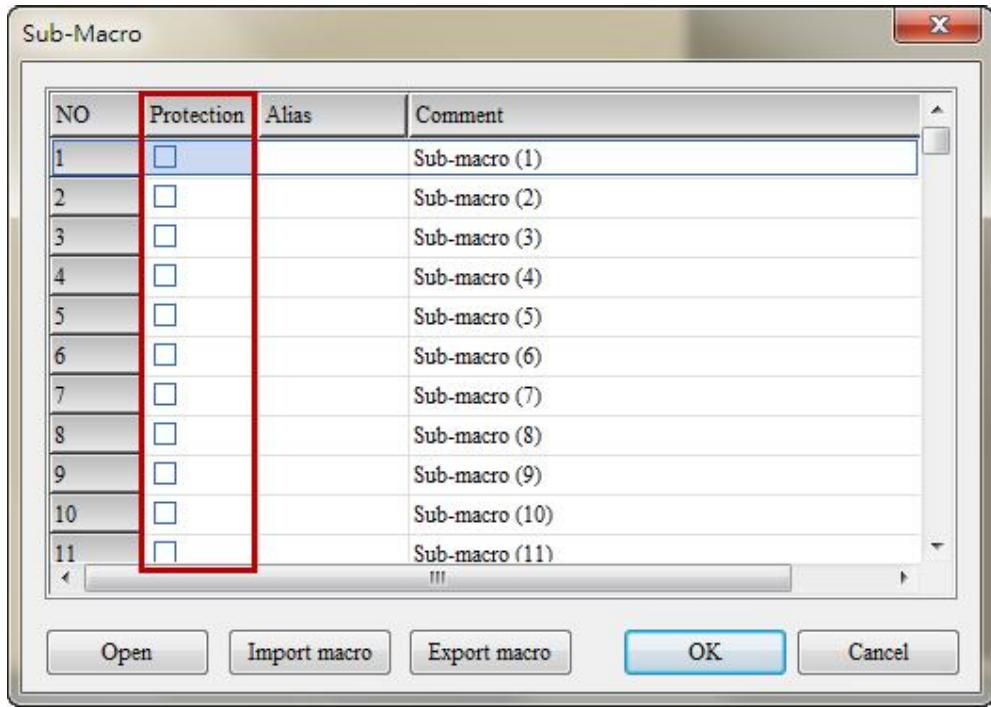


Figure 24-1-7-7 Protection from sub-macro

When check “protection” function, users will be asked to enter the password.

If sub-macro (1) has been encrypted, users have to enter the password to decrypt it and edit sub-macro (1).

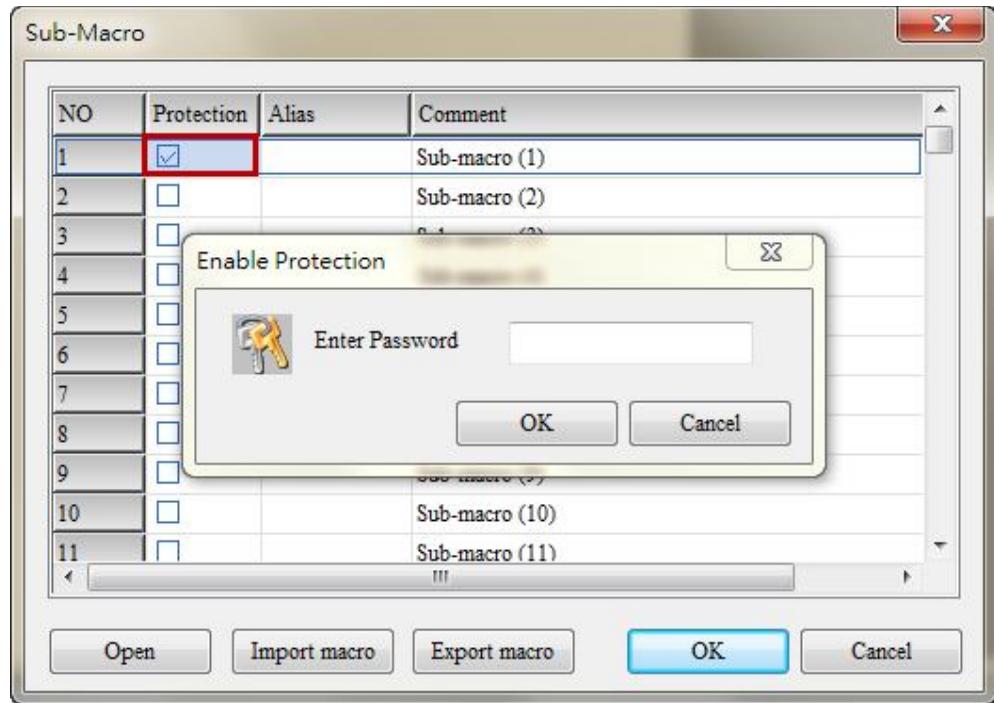


Figure 24-1-7-8 Sub-macro encryption

To disable the protection function, users will be asked to enter the password that just set for sub-macro (1).

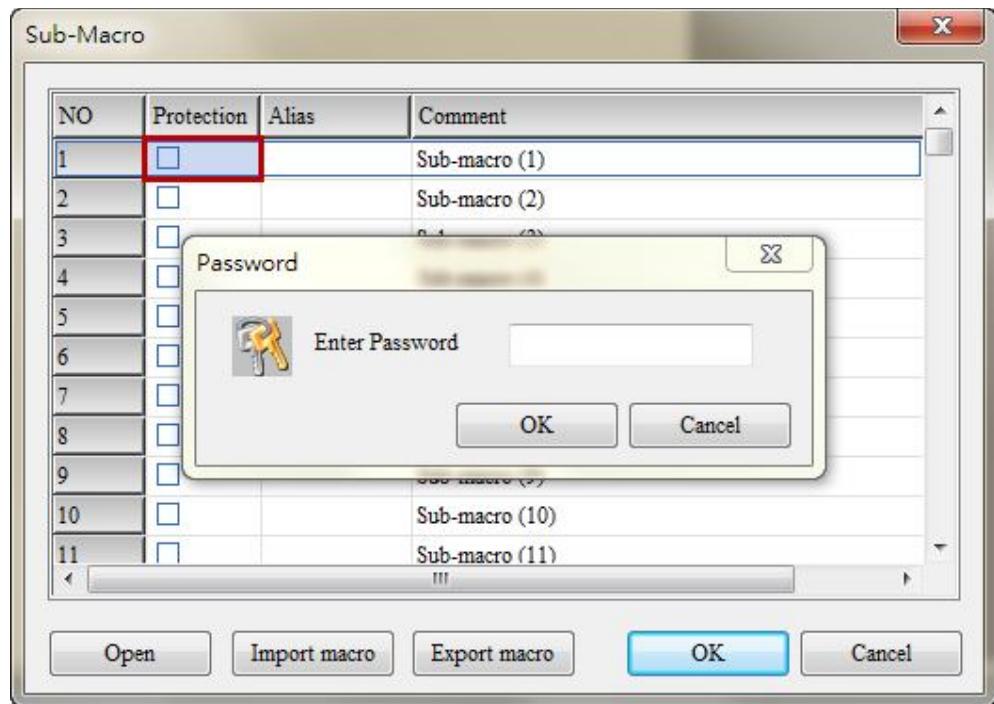
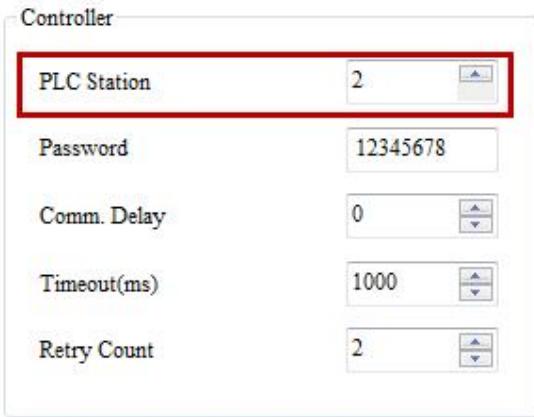
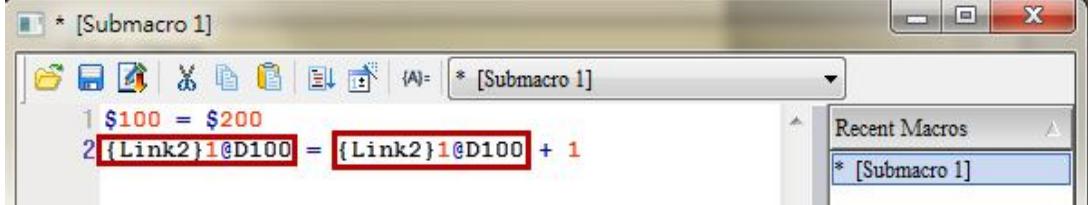
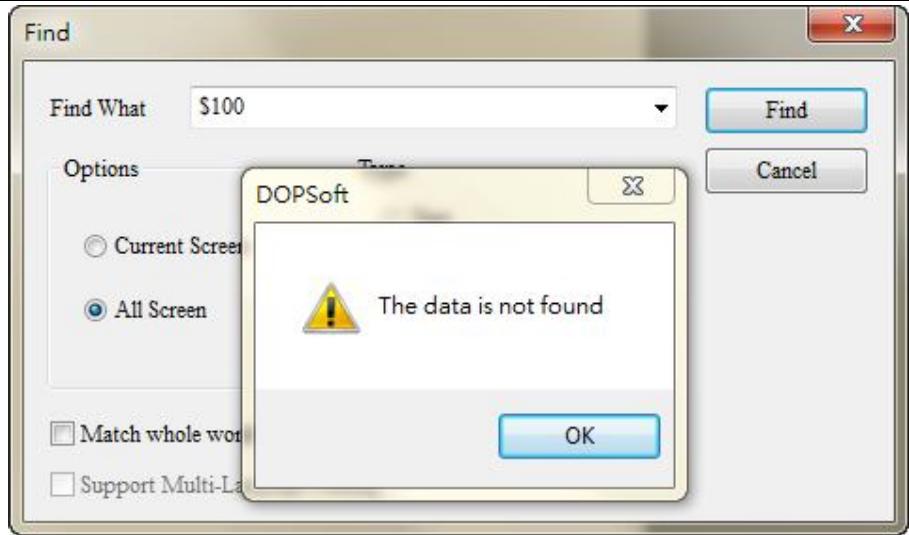
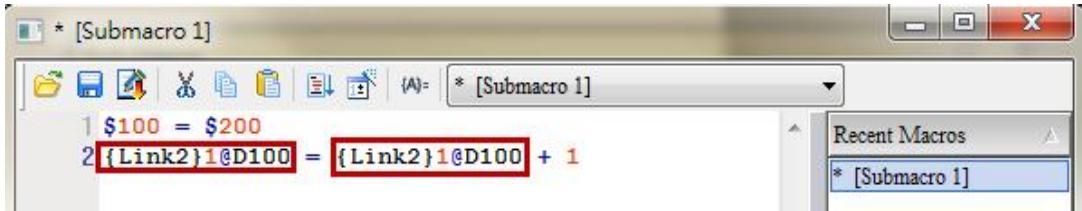
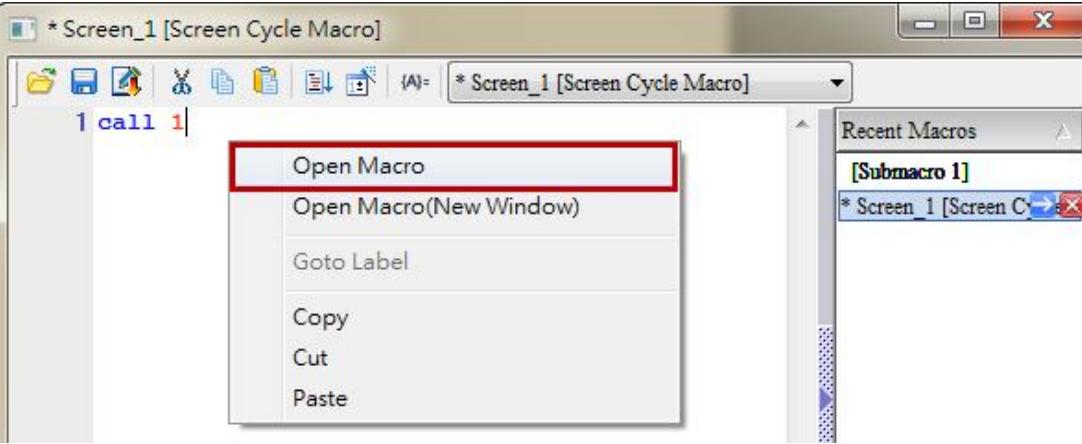
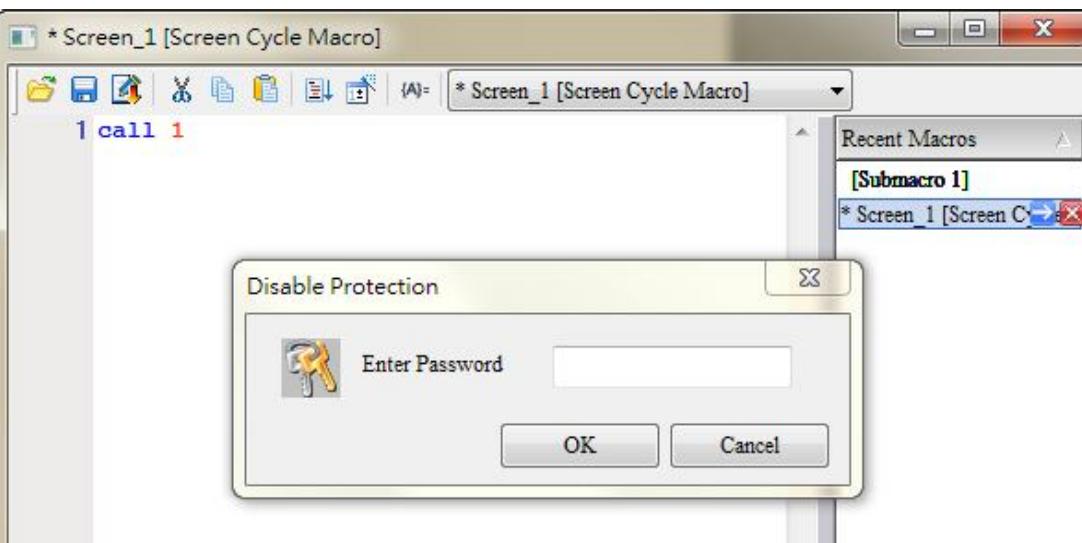
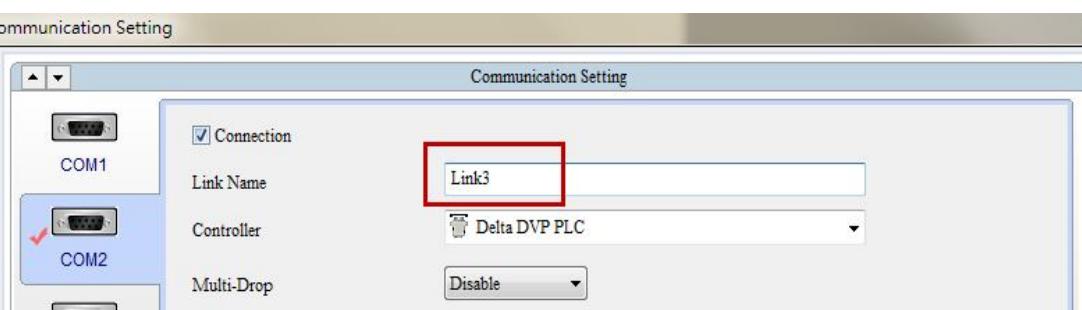


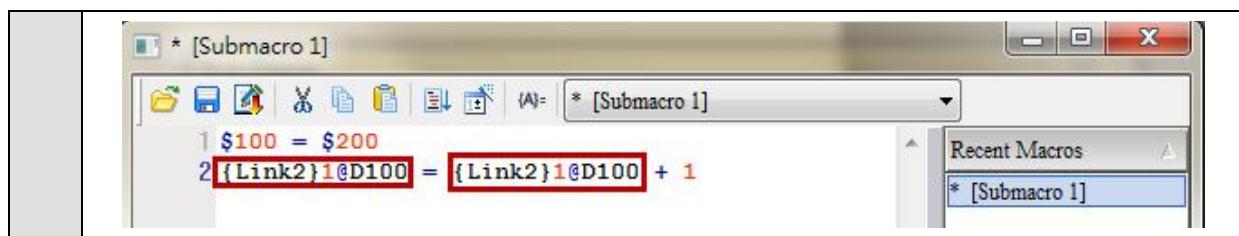
Figure 24-1-7-9 Sub-macro decryption

When the sub-macro is set with protection, functions that would be influenced are shown as below:

	<p>If the protected sub-macro is set with station number for communication, it will be failed to change the station number.</p> <ul style="list-style-type: none"> ➤ If the station number of the protected sub-macro is 1, change the default station number of PLC to 2, and then check the protected sub-macro. You will find that the station number has to be 1, which is unchangeable.
(1)	 
(2)	<p>If \$100 is set in the protected macro, \$100 cannot be found when searching the address.</p> 

	<p>If \$100 is set in the protected macro, \$100 cannot be replaced by \$500 when executing Replace function.</p>
(3)	
	<p>If the station number of the protected sub-macro is 1, it cannot be changed to 2 when executing the function of Replace Station Value.</p>
(4)	

	
	<p>Assuming that Call command is included in macro, right click to open sub-macro window. If it is protected, users will be asked to enter the password.</p> 
(5)	
(6)	<p>Change the Link Name to Link3. In protected sub-macro, Link2 cannot be changed to Link3.</p> 



24-1-8 Initial Macro

The Initial Macro can be edited by going into [Options] → [Initial Macro].

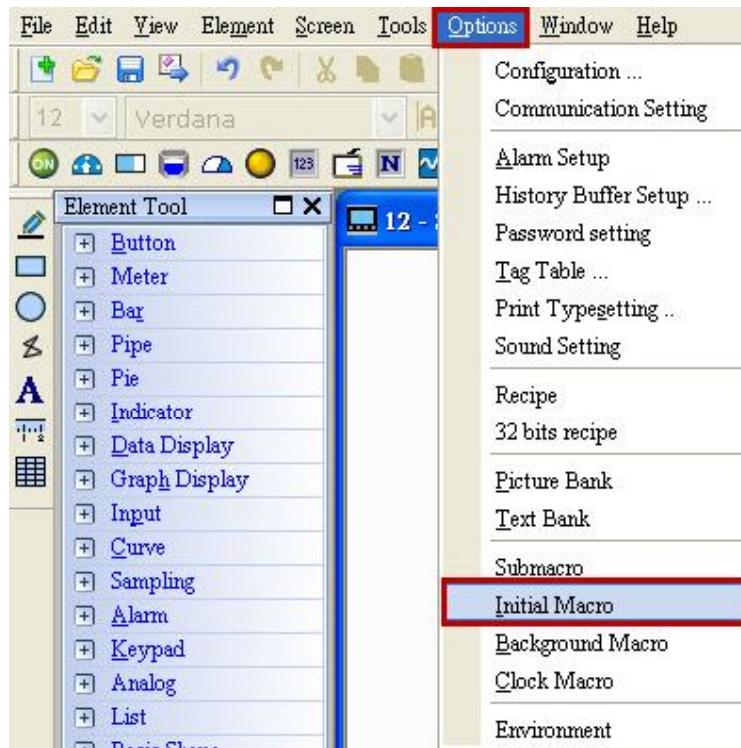


Figure 24-1-8-1 Initial Macro

The Initial Macro is the first macro to be executed right after the HMI starts up, and hence, users can write common HMI initial values into this macro.

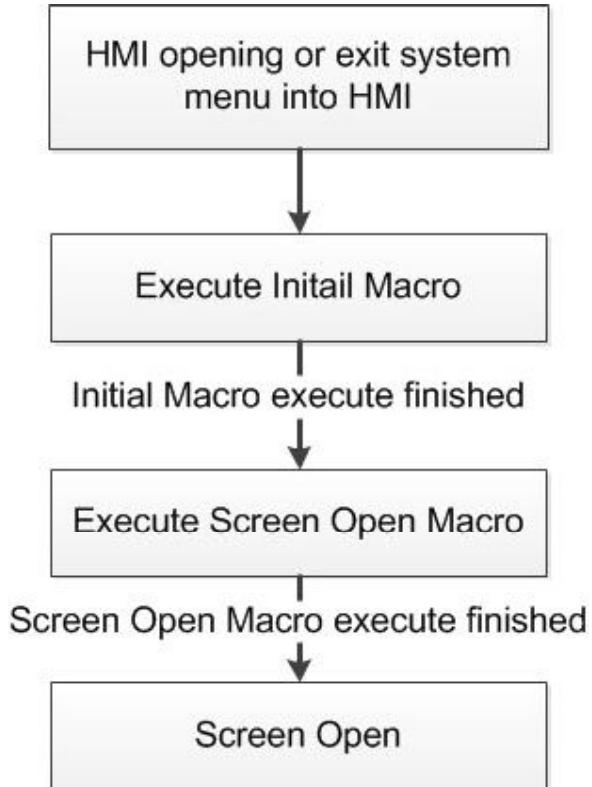


Figure 24-1-8-2 Initial Macro Flowchart

24-1-9 Background Macro

The Background Macro can be edited by going into [Options] → [Background Macro].

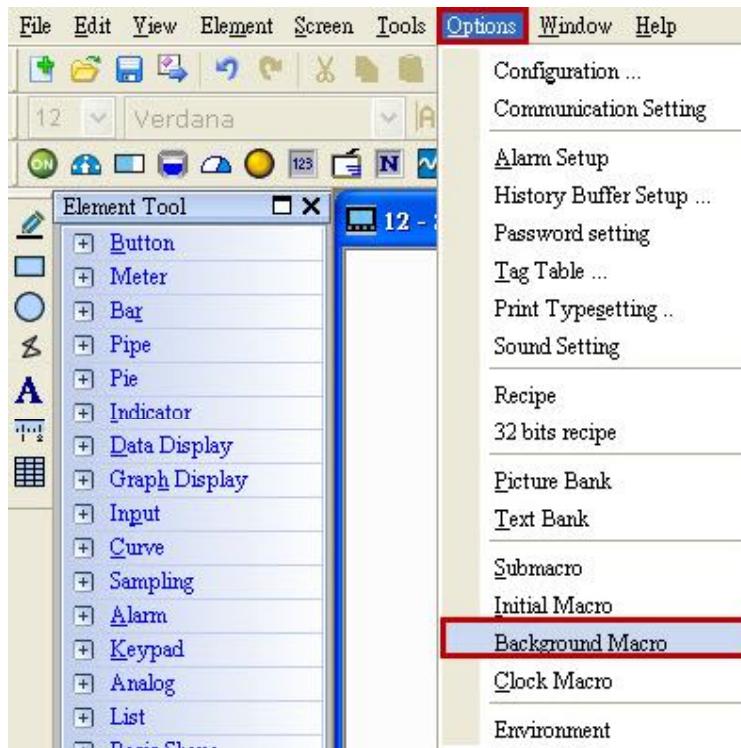


Figure 24-1-9-1 Background Macro

The Background Macro will be continuously executed during HMI operations. Whether it is executed one line or several lines at a time (does not stop after the first sequence), this macro will continue to be executed and repeat from the first line after it reaches the last line. To define the number of lines to be executed each time, please go to [Options] → [Configuration...] to set the [Background Macro Update Cycle] and the maximum number of lines for each update cycle is 512.

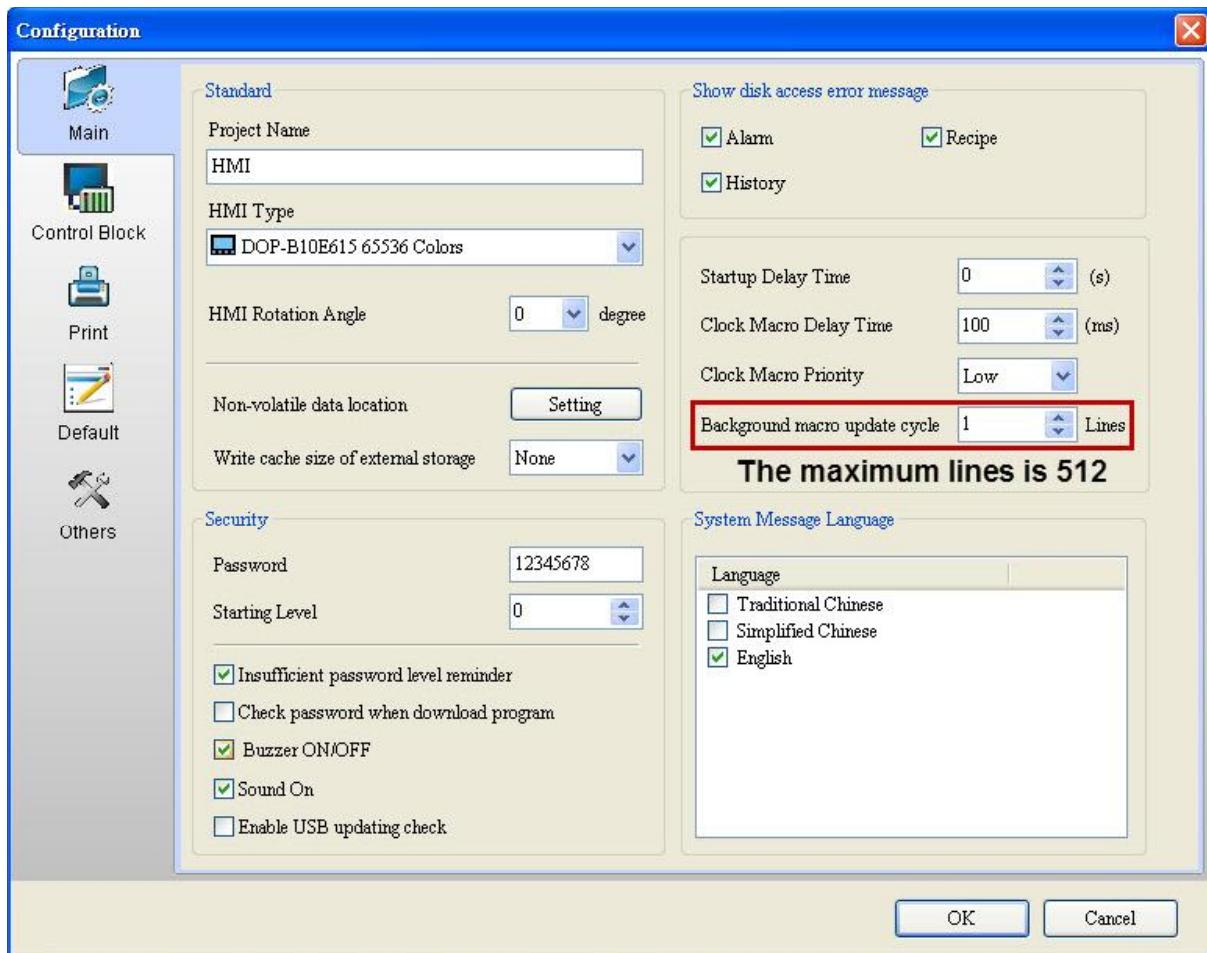


Figure 24-1-9-2 Background Macro Update Cycle

Suppose there are 10 elements created onscreen and 6 macro commands are written within the Background Macro. If the background macro update cycle is set to 3, then the flow of the process is shown below:

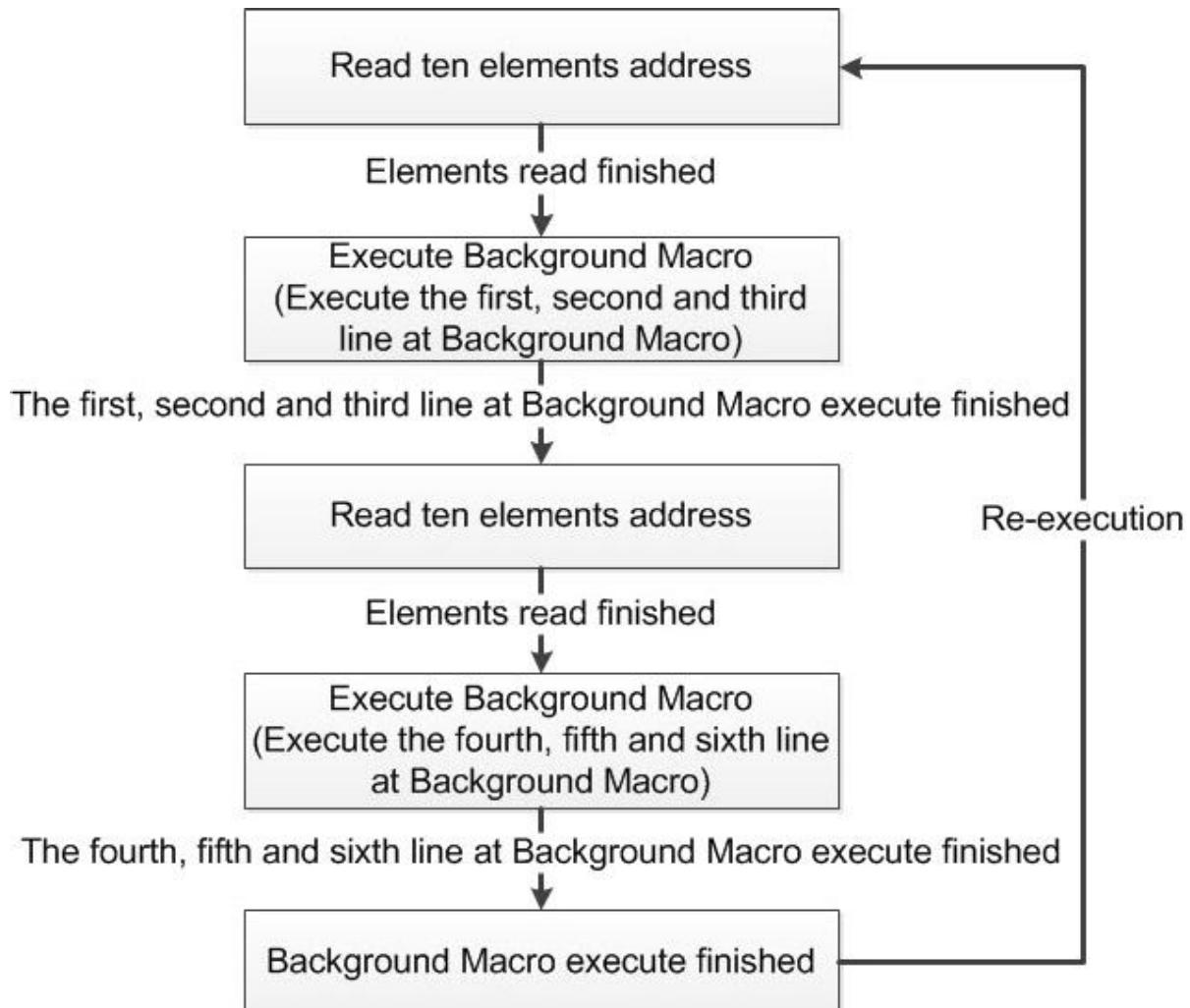


Figure 24-1-9-3 Background Macro Flowchart

24-1-10 Clock Macro

The Clock Macro can be edited by going into [Options] → [Clock Macro].

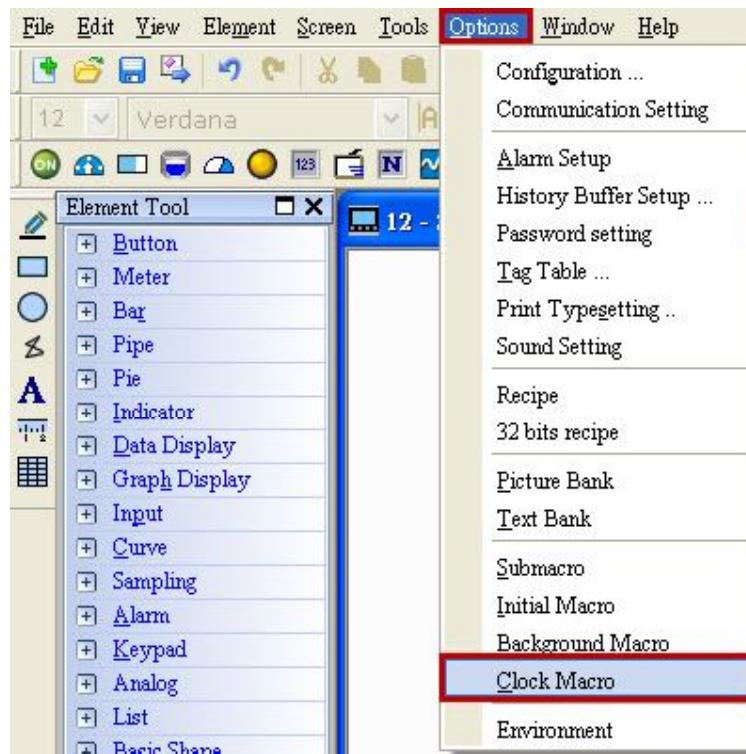


Figure 24-1-10-1 Clock Macro

The Clock Macro will be continuously executed during HMI operations. Unlike the Background Macro, the Clock Macro will finish executing all commands within the macro rather than one line or several lines at once. Similar to the Screen Cycle Macro, the Clock Macro also repeats its executions based on the Clock Macro Delay Time. Users can configure the delay time by going to [Options] → [Configuration...] to set the [Clock Macro Delay Time]. So that each time the Clock Macro is executed, it will wait until the macro delay time is completely elapsed. The default Clock Macro Delay Time is 100ms and the maximum time length is 65535ms.

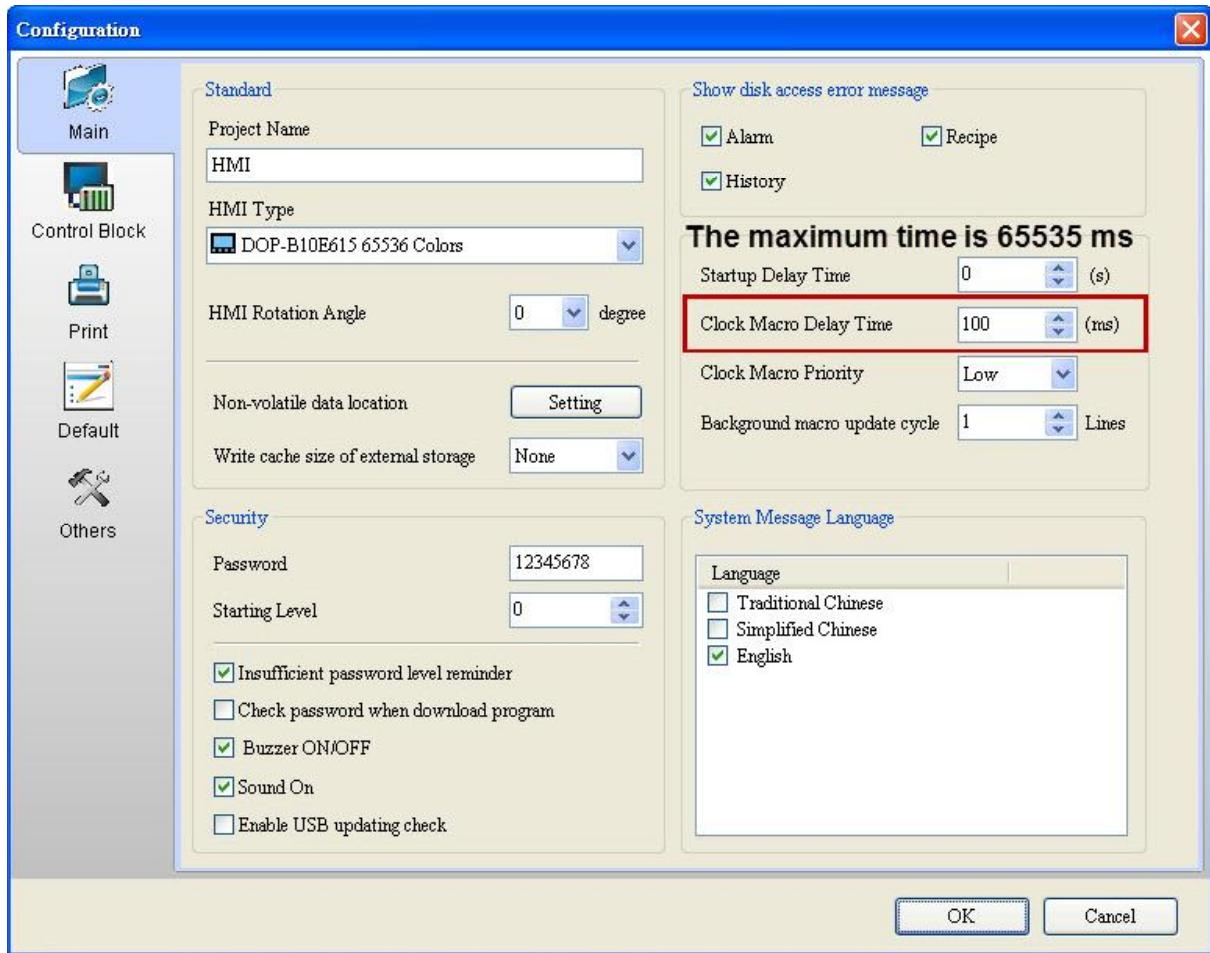


Figure 24-1-10-2 Clock Macro Delay Time

Three levels of priority (high, medium, and low) are also available for users to configure the priority of the Clock Macro. The order of priority can ensure the accurate delay time of Clock Macro.

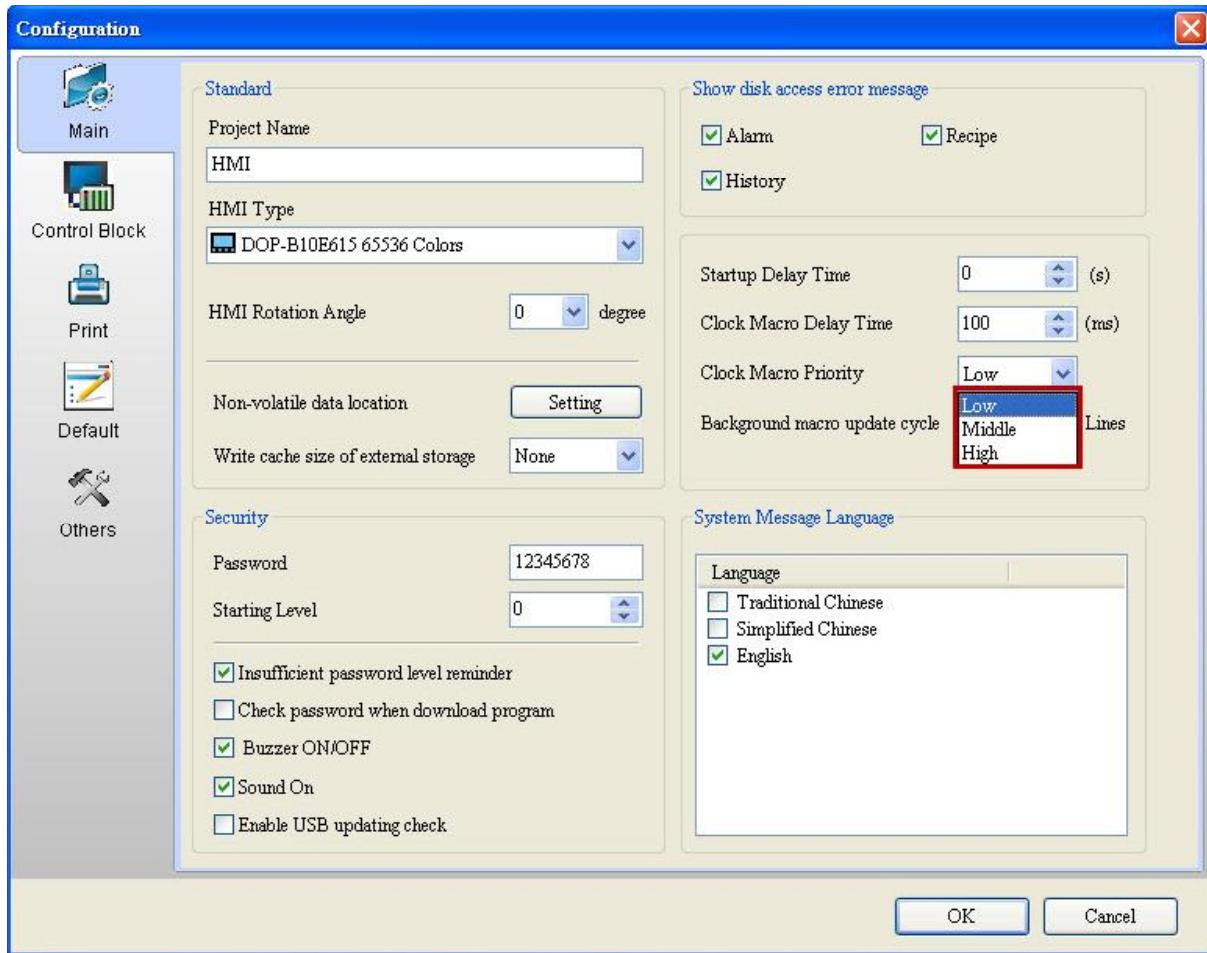


Figure 24-1-10-3 Clock Macro Priority

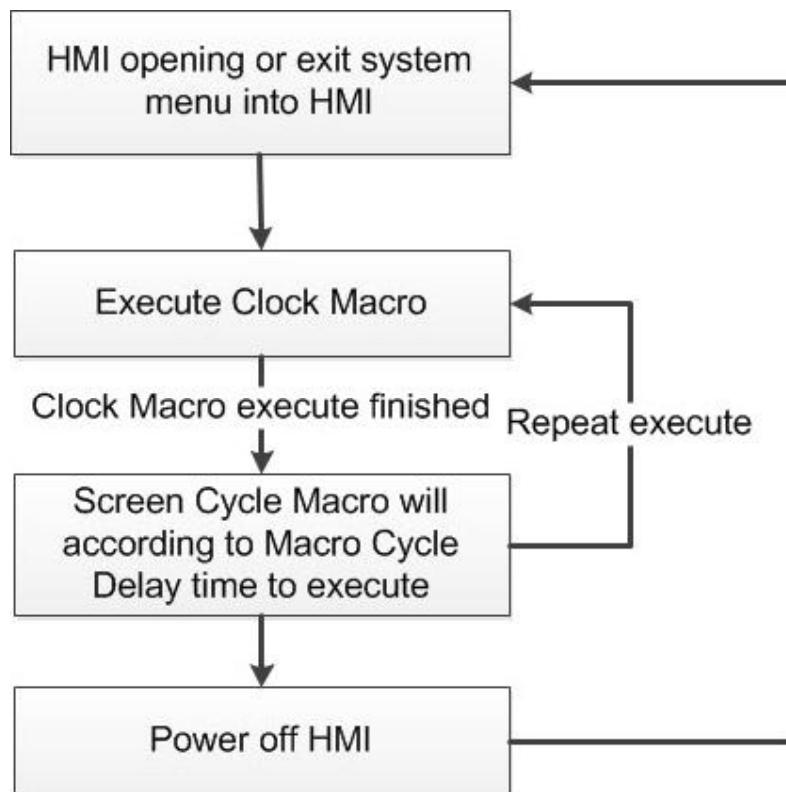


Figure 24-1-10-4 Clock Macro Flowchart

24-2 Macro Editing Window

After choosing a desired Macro, users can start to edit a Macro. Please note that each macro is capable of handling 512 lines of commands and only 640 bytes of characters are allowed on each line. Only 10 most recently edited macros will be displayed in the box on the top right side of the editing area. If there are more than 10 macros opened recently, then the first macro will be closed and the new Macro will be listed instead. Suppose the first Macro has been modified before it is closed, then a dialog box will pop up to ask the user to save the update of the macro before adding in the new macro.

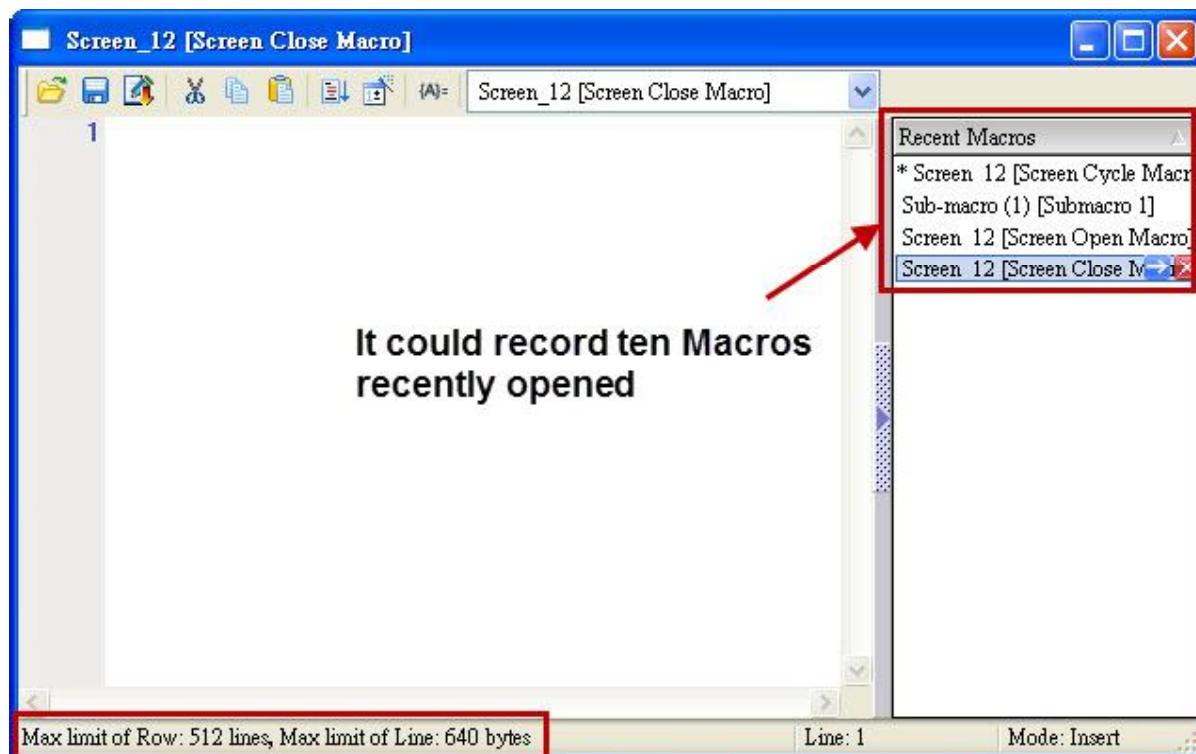


Figure 24-2-1 Macro Editing Window

The toolbar within the editing window is also available for users to design and edit macro commands.

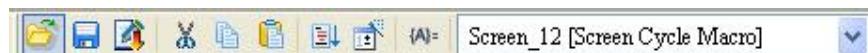
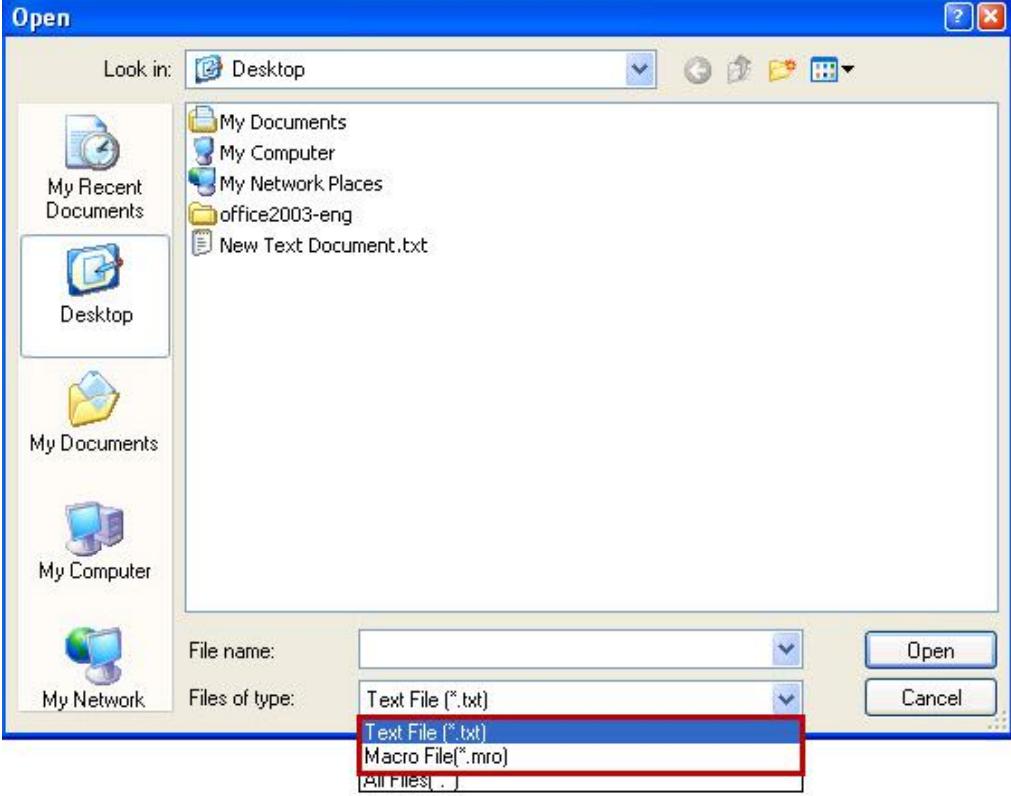
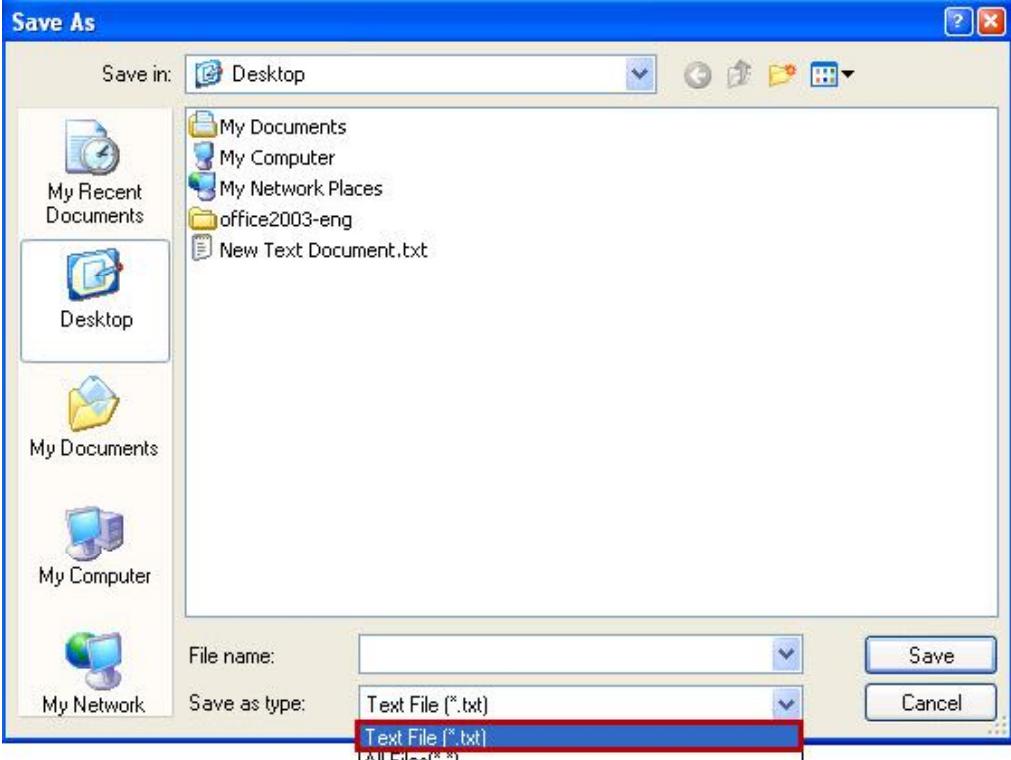


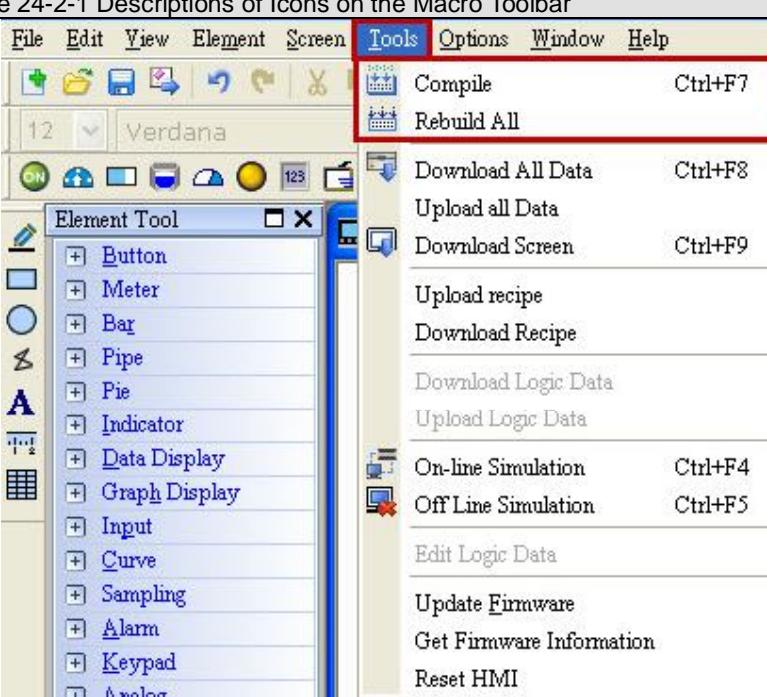
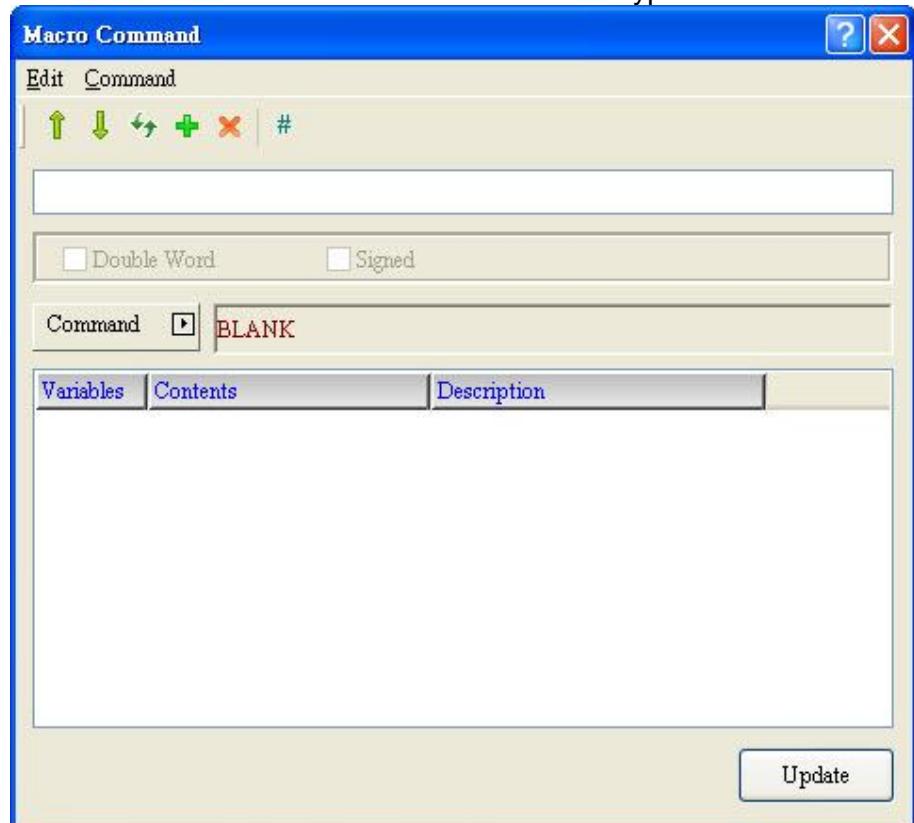
Figure 24-2-2 Macro Toolbar

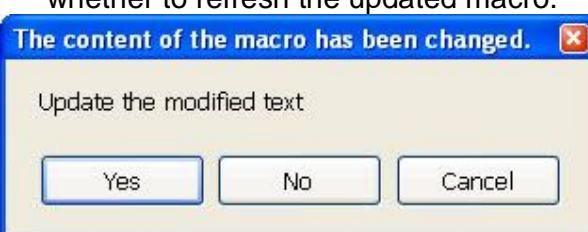
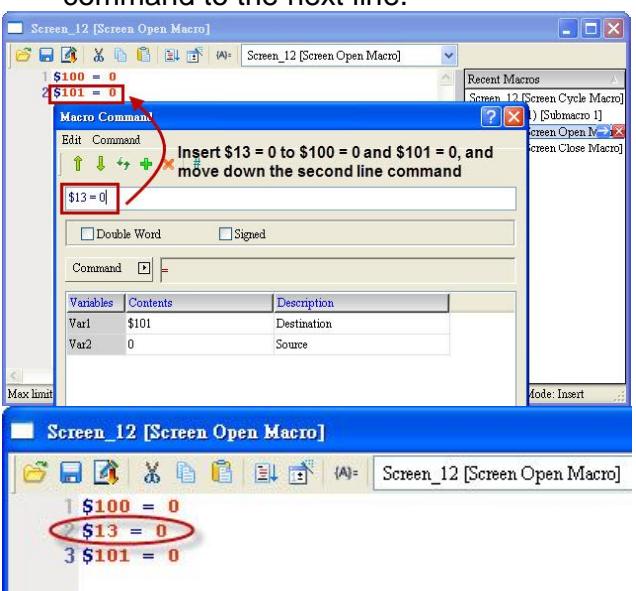
Functions of each icon on the Macro Toolbar are explained in the following table.

Functions of Macro Toolbar Icons		
Table 24-2-1 Descriptions of Icons on the Macro Toolbar		
Icon	Function	Content Description
	Open	➤ Open is equivalent to import. Two file formats are available: txt and mro. Users can choose to directly import previously edited macros to save time.

<h3 style="text-align: center;">Functions of Macro Toolbar Icons</h3> <p style="text-align: center;">Table 24-2-1 Descriptions of Icons on the Macro Toolbar</p>		
		
		<ul style="list-style-type: none"> ➤ Save is equivalent to Export, but only the txt file format is supported. Users can save the edited macro as a backup or for use of other screens. 
	Save	<ul style="list-style-type: none"> ➤ This function will refresh the updated macro and check if the syntaxes
	Update	<ul style="list-style-type: none"> ➤ This function will refresh the updated macro and check if the syntaxes

Functions of Macro Toolbar Icons Table 24-2-1 Descriptions of Icons on the Macro Toolbar		
		<p>are correct. If this button is not executed before exiting the current Macro editing window (click the exit button), the users will be prompted that the macro being edited has been updated.</p> 
		<ul style="list-style-type: none"> ➤ Syntax check will be performed after clicking the update button. If there is a syntax error, then the following error message will pop up. 
  	Cut Copy Paste	<ul style="list-style-type: none"> ➤ Cut, Copy and Paste are used the same way as Microsoft Office and users can also choose to perform these operations through hotkeys (Cut: Ctrl + X; Copy: Ctrl + C; Paste: Ctrl + V).
	Syntax Check	<ul style="list-style-type: none"> ➤ Syntax Check will check if macro commands are written correctly. If there is a syntax error, then the following error message will pop up. 
<p>NOTE: Syntax Check is not equivalent to compiling macros. Please consider the compile function to compile.</p>		

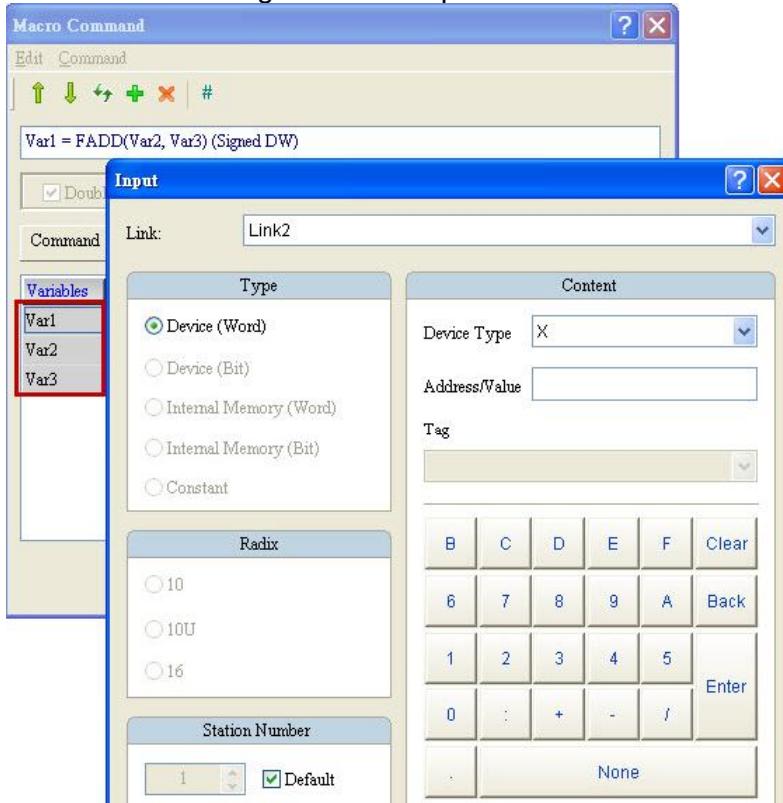
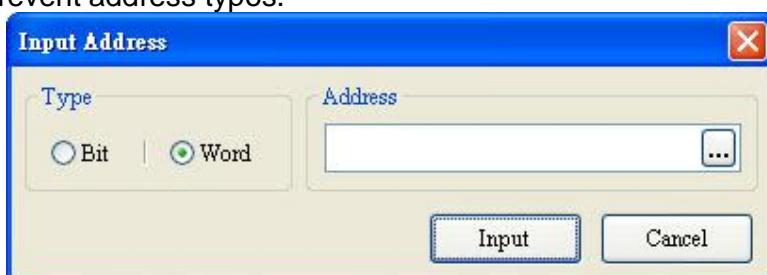
<h3 style="text-align: center;">Functions of Macro Toolbar Icons</h3> <p style="text-align: center;">Table 24-2-1 Descriptions of Icons on the Macro Toolbar</p> 		
 Macro Wizard	<ul style="list-style-type: none"> ➤ The purpose of the wizard is for users to more easily edit and type in Macro commands and this will also eliminate typos. 	<p style="margin-left: 20px;">Edit Previous ➤ The previous icon will move the mouse cursor to the last line and the next icon will</p>
		24-34 Revision March, 2014

Functions of Macro Toolbar Icons				
Table 24-2-1 Descriptions of Icons on the Macro Toolbar				
		Next 		<p>move the mouse cursor to the next line.</p> <ul style="list-style-type: none"> ➤ Via these two icons, users can choose which line within the macro to move to.
		Update 		<ul style="list-style-type: none"> ➤ This function will refresh the updated macro and check if the syntaxes are correct. If syntax errors are detected, then the error messages shown below will pop up for user notification.  <ul style="list-style-type: none"> ➤ If this icon is not executed before exiting the current Macro editing window (click the exit icon), the users will be prompted whether to refresh the updated macro. 
		Insert 		<ul style="list-style-type: none"> ➤ Insert a new macro command. This icon will replace the macro command where the cursor is located with the newly inserted command and move the current macro command to the next line. 

Functions of Macro Toolbar Icons

Table 24-2-1 Descriptions of Icons on the Macro Toolbar

			Delete 	<ul style="list-style-type: none"> ➤ Delete the macro command where the mouse cursor is located. If the deleted command is not on the last line within the macro, then the lines below the deleted commands will move up. <p>NOTE : If no macro command exists where the mouse cursor is located, then nothing will happen.</p>								
			Comment 	<ul style="list-style-type: none"> ➤ Comment helps users to manage macros, improve program readability and simplify code maintenance tasks. Users only need to insert the symbol # or click # button within the macro wizard ([Edit] → [Comment]) to comment on the program. <p>NOTE: Comments within the macro will not be executed.</p>								
		Commands		<ul style="list-style-type: none"> ➤ These are macro functions and they are grouped into the following categories: <ul style="list-style-type: none"> Arithmetic ► Logical ► Data transfer ► Data Conversion ► Comparison ► FlowControl ► Bit Setting ► COM port ► Drawing ► Others ► ➤ For detailed introductions on the commands, please refer to 24-3 Macro Commands. 								
		Double Word		<ul style="list-style-type: none"> ➤ 32 bit or signed number operations are supported by macro commands. If Signed is checked, then the command will be executed as a signed number, or else the command will be executed as an unsigned number. If Double Word is checked, then the command will be executed as a 32 bit command, or else the command will be executed as a Double Word command, or 16 bit command. 								
		Signed		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px; width: 50%;">Unsigned</td><td style="padding: 5px; width: 50%;">Unsigned number</td></tr> <tr> <td style="padding: 5px;">Signed</td><td style="padding: 5px;">Signed number</td></tr> <tr> <td style="padding: 5px;">WORD</td><td style="padding: 5px;">16 bit data</td></tr> <tr> <td style="padding: 5px;">DW (DOUBLE WORD, DWORD)</td><td style="padding: 5px;">32 bit data</td></tr> </table>	Unsigned	Unsigned number	Signed	Signed number	WORD	16 bit data	DW (DOUBLE WORD, DWORD)	32 bit data
Unsigned	Unsigned number											
Signed	Signed number											
WORD	16 bit data											
DW (DOUBLE WORD, DWORD)	32 bit data											
				<p>NOTE:</p>								

Functions of Macro Toolbar Icons Table 24-2-1 Descriptions of Icons on the Macro Toolbar			
			For the Double Word data type, then each memory address will take two registers.
	Command		<ul style="list-style-type: none"> ➤ Command is equivalent to the [Command] icon in the Macro Wizard dialog window and they both are used for users to choose their desired macro command. ➤ For detailed explanations please refer to 24-3 Macro Commands.
	Variable		<ul style="list-style-type: none"> ➤ This function provide variables used in macro commands and users can directly click the variable button to configure relevant parameters. 
	Input Address		<ul style="list-style-type: none"> ➤ Users can enter memory addresses used in the macro via this function to prevent address typos. 

24-3 Macro Commands

Based on the nature of macro commands, they are divided into the following ten categories: Arithmetic, Logical, Data transfer, Data conversion, Comparison, Flow control, Bit setting, Communication and Drawing and Other. If select HMC and PS models, users can choose the macro command of industrial application (E-Cam) which is shown as below.

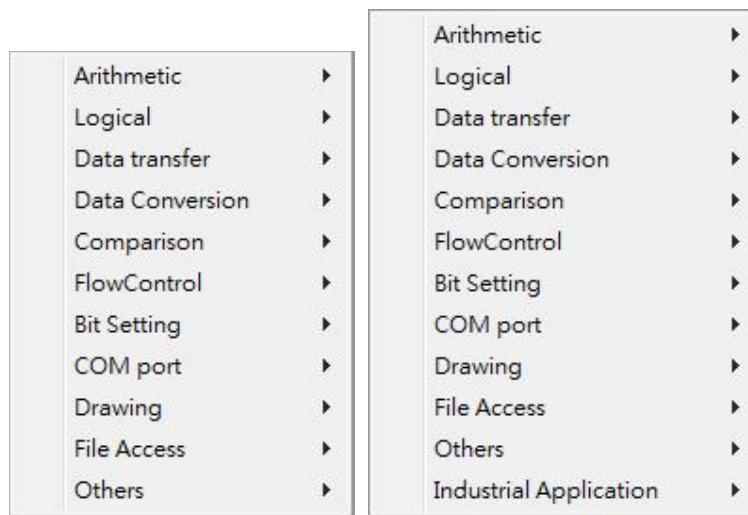


Figure 24-3-1 Type of Macro Commands

All supported macro commands are listed below:

Category	Function	Expression	Description
Arithmetic	+	Var1 = Var2 + Var3	Addition
	-	Var1 = Var2 - Var3	Subtraction
	*	Var1 = Var2 * Var3	Multiplication
	/	Var1 = Var2 / Var3	Division
	%	Var1 = Var2 % Var3	Get Reminder
	MUL64	Var1 = MUL64(Var2, Var3) (Signed DW)	64 bit Multiplication
	ADDSUMW	Var1 = ADDSUMW(Var2, Var3)	Repeated Addition
	FADD	Var1 = FADD(Var2, Var3) (Signed DW)	Floating Point Addition
	FSUB	Var1 = FSUB(Var2, Var3) (Signed DW)	Floating Point Subtraction
	FMUL	Var1 = FMUL(Var2, Var3) (Signed DW)	Floating Point Multiplication
	FDIV	Var1 = FDIV(Var2, Var3) (Signed DW)	Floating Point Division
	FMOD	Var1 = FMOD(Var2, Var3) (Signed DW)	Floating Point Reminder
	SIN	Var1 = SIN(Var2) (Signed DW)	Sine Function
	COS	Var1 = COS(Var2) (Signed DW)	Cosine Function
	TAN	Var1 = TAN(Var2) (Signed DW)	Tangent Function
	COT	Var1 = COT(Var2) (Signed DW)	Cotangent Function
	SEC	Var1 = SEC(Var2) (Signed DW)	Secant Function
	CSC	Var1 = CSC(Var2) (Signed DW)	Cosecant Function

Category	Function	Expression	Description
Logical		Var1 = Var2 Var3	Bitwise OR Operation
	&&	Var1 = Var2 && Var3	Bitwise AND Operation
	^	Var1 = Var2 ^ Var3	Bitwise XOR Operation
	NOT	Var1 = NOT Var2	Bitwise NOT Operation
	<<	Var1 = Var2 << Var3	SHL(Bitwise Left-shift Operation)
	>>	Var1 = Var2 >> Var3	SHR(Bitwise Right-shift Operation)
Data transfer	MOV	Var1 = Var2	Data Moving Operand
	BMOV	BMOV(Var1, Var2, Var3)	Move in Block
	ArrayCopy	Var1 = ArrayCopy(Var2, Var3, Var4, Var5, Var6)	Copy Array
	FILL	FILL(Var1, Var2, Var3)	Fill in the Block
	FILLASC	FILLASC(Var1, " ")	String to ASCII Conversion
	FMOV	Var1 = FMOV(Var2) (Signed DW)	Move floating point data
Data conversion	BCD	Var1 = BCD(Var2)	Decimal to BCD Conversion
	BIN	Var1 = BIN(Var2)	BCD to Decimal Conversion
	TODWORD	Var1 = TODWORD(Var2)	WORD to Double WORD Conversion
	TOWORD	Var1 = TOWORD(Var2, Var3)	BYTE to Word Conversion
	TOBYTE	Var1 = TOBYTE(Var2, Var3)	Word to Byte Conversion
	SWAP	SWAP(Var1, Var2, Var3)	Swap between highbit and lowbit of WORD
	XCHG	XCHG(Var1, Var2, Var3)	Data Exchange
	MAX	Var1 = MAX(Var2, Var3)	Get Maximum value
	MIN	Var1 = MIN(Var2, Var3)	Get Minimum value
	TOHEX	Var1 = TOHEX(Var2)	Convert 4 ASCII characters to a four digit integer in hexadecimal format
	TOASC	Var1 = TOASC(Var2)	Convert hexadecimal integers into 4 Words ASCII characters
	FCNV	Var1 = FCNV(Var2) (Signed DW)	Conversion of

Category	Function	Expression		Description
	ICNV			integer into floating point value
		Var1 = ICNV(Var2) (Signed DW)		Conversion from integer to floating point value
	SPRINTF	Var1 = SPRINTF(Var2, "%u", Var4)		Format String
Comparison	IF...THEN GOTO	IF ==	IF Var1 == Var2 THEN GOTO LABEL Var3	If Goto a certain label identifier and continue subsequent executions
		IF !=	IF Var1 != Var2 THEN GOTO LABEL Var3	
		IF >	IF Var1 > Var2 THEN GOTO LABEL Var3	
		IF >=	IF Var1 >= Var2 THEN GOTO LABEL Var3	
		IF <	IF Var1 < Var2 THEN GOTO LABEL Var3	
		IF <=	IF Var1 <= Var2 THEN GOTO LABEL Var3	
		IF AND == 0	IF (Var1 && Var2) == 0 THEN GOTO LABEL Var3	
		IF AND != 0	IF (Var1 && Var2) != 0 THEN GOTO LABEL Var3	
		IF == ON	IF Var1 == ON THEN GOTO LABEL Var2	
		IF == OFF	IF Var1 == OFF THEN GOTO LABEL Var2	
		IFB == ON	IFB Var1 == ON THEN GOTO LABEL Var2	
		IFB == OFF	IFB Var1 == OFF THEN GOTO LABEL Var2	
	IF...THEN CALL	IF == CALL	IF Var1 == Var2 THEN CALL Var3	Macro If... Then Call a Submacro
		IF != CALL	IF Var1 != Var2 THEN CALL Var3	
		IF > CALL	IF Var1 > Var2 THEN CALL Var3	

Category	Function	Expression		Description
IF...	IF... ELSEIF... ELSE	IF >= CALL	IF Var1 >= Var2 THEN CALL Var3	Logical Comparison
		IF < CALL	IF Var1 < Var2 THEN CALL Var3	
		IF <= CALL	IF Var1 <= Var2 THEN CALL Var3	
		IF AND == 0 CALL	IF (Var1 && Var2) == 0 THEN CALL Var3	
		IF AND != 0 CALL	IF (Var1 && Var2) != 0 THEN CALL Var3	
		IF == ON CALL	IF Var1 == ON THEN CALL Var2	
		IF == OFF CALL	IF Var1 == OFF THEN CALL Var2	
		IF ==	IF Var1 == Var2	
		IF !=	IF Var1 != Var2	
		IF >	IF Var1 > Var2	
ELSEIF...	ELSEIF... ELSE	IF >=	IF Var1 >= Var2	Logical Comparison
		IF <	IF Var1 < Var2	
		IF <=	IF Var1 <= Var2	
		IF AND == 0	IF (Var1 && Var2) == 0	
		IF AND != 0	IF (Var1 && Var2) != 0	
		IF == ON	IF Var1 == ON	
		IF == OFF	IF Var1 == OFF	
		ELSEIF ==	ELSEIF Var1 == Var2	
		ELSEIF !=	ELSEIF Var1 != Var2	
		ELSEIF >	ELSEIF Var1 > Var2	
ELSE	ELSE	ELSEIF >=	ELSEIF Var1 >= Var2	Logical Comparison
		ELSEIF <	ELSEIF Var1 < Var2	
		ELSEIF <=	ELSEIF Var1 <= Var2	
		ELSEIF AND == 0	ELSEIF (Var1 && Var2) == 0	
		ELSEIF AND != 0	ELSEIF (Var1 && Var2) != 0	
		ELSEIF == ON	ELSEIF Var1 == ON	
		ELSEIF == OFF	ELSEIF Var1 == OFF	
		ELSE	ELSE	

Category	Function	Expression	Description
	ENDIF	ENDIF	Logical Comparison
	FCMP	Var1 = FCMP(Var2, Var3) (Signed DW)	Comparison of Floating Point Data
Flow control	GOTO	GOTO LABEL Var1	Label Identifier for the current process to unconditionally jump to
	LABEL	LABEL Var1	Label Identifier
	CALL	CALL Var1	Call Submacro
	RET	RET	Exit Submacro
	FOR	FOR Var1	Loop
	NEXT	NEXT	
	END	END	End Macro
Bit setting	BITON	BITON Var1	Set Bits to On
	BITOFF	BITOFF Var1	Set Bits Off
	BITNOT	BITNOT Var1	Set Bits to Inverse State (ON→OFF; OFF→ON)
	GETB	Var1 = GETB Var2	Acquire Bit State
Communication	INITCOM	Var1 = INITCOM(Var2, Var3, Var4, Var5, Var6, Var7, Var8)	COM Port Initialization
	ADDSUM	Var1 = ADDSUM(Var2, Var3)	Checksum Calculation through Addition
	XORSUM	Var1 = XORSUM(Var2, Var3)	Checksum Calculation through XOR Operation
	PUTCHARS	Var1 = PUTCHARS(Var2, Var3, Var4)	Output Character by Com Port
	GETCHARS	Var1 = GETCHARS(Var2, Var3, Var4)	Character Acquisition through Com Port
	SELECTCOM	SELECTCOM(Var1)	Com Port Selection
	CLEARCOMBUFF ER	CLEARCOMBUFFER(Var1, Var2)	Com Port Buffer Clearance
	CHRCHKSUM	Var1 = CHRCHKSUM("Var2", Var3, Var4)	Calculation of String Length and Checksum
	LOCKCOM	Var1 = LOCKCOM(Var2, Var3)	Lock Com Port
	UNLOCKCOM	UNLOCKCOM(Var1)	Unlock Com Port
	STATIONON	STATIONON(Var1, Var2)	Set Station On
	STATIONOFF	STATIONOFF(Var1, Var2)	Set Station Off
	IPON	Var1 = IPON(Var2, Var3, Var4, Var5, Var6)	Enable IP Address
	IPOFF	Var1 = IPOFF(Var2, Var3, Var4, Var5, Var6)	Disable IP Address
Drawing	RECTANGLE	RECTANGLE(Var1)	Draw Rectangle
	LINE	LINE(Var1)	Draw Line

Category	Function	Expression	Description
	POINT	POINT(Var1)	Draw Point
	CIRCLE	CIRCLE(Var1)	Draw Ellipse
File Access	FileSlotRead	Var1 = FileSlotRead(Var2, Var3, Var4, Var5)	Read Files
	FileSlotWrite	Var1 = FileSlotWrite(Var2, Var3, Var4, Var5)	Write Files
	FileSlotRemove	Var1 = FileSlotRemove(Var2)	Remove Files
	FileSlotGetLength	Var1 = FileSlotGetLength(Var2, Var3)	Accessing File Length
	FileSlotExport	Var1 = FileSlotEXPORT(Var2, Var3, Var4, Var5)	Export Files
	FileSlotImport	Var1 = FileSlotIMPORT(Var2, Var3, Var4, Var5)	Import Files
Other	Time Tick	Var1 = TIMETICK	Acquire System up duration from System Startup to Present
	GETLASTERROR	Var1 = GETLASTERROR	Get Last Error Value
	Comment	#	Make Comment
	Delay	Delay(Var1)	System Delay
	GETSYSTEMTIME	Var1 = GETSYSTEMTIME	Acquire System Time
	SETSYSTEMTIME	SETSYSTEMTIME(Var1)	Set System Time
	GETHISTORY	Var1 = GETHISTORY (Var2, Var3, Var4, Var5, Var6)	Acquire Historical Log
	EXPORT	EXPORT(Var1)	Export Report to an External Device
	EXRCP16	Var1 = EXRCP16(Var2, Var3)	Export 16 bit Equation
	IMRCP16	Var1 = IMRCP16(Var2, Var3)	Import 16 bit Equation
	EXRCP32	Var1 = EXRCP32(Var2, Var3)	Export 32 bit Equation
	IMRCP32	Var1 = IMRCP32(Var2, Var3)	Import 32 bit Equation
	EXENRCP	Var1 = EXENRCP(Var2, Var3)	Export Enhance Recipe Equation
	IMENRCP	Var1 = IMENRCP(Var2, Var3)	Import Enhance Recipe Equation
	EXHISTORY	Var1 = EXHISTORY(Var2, Var3, Var4)	Export History Data
	EXALARM	Var1 = EXALARM(Var2, Var3)	Export Alarm Information
	DISKFORMAT	Var1 = DISKFORMAT(Var2)	Format Disk
	BMP_CAPTURE	Var1 = BMP_CAPTURE(Var2)	Screen Capture
	PLC_DOWNLOAD	Var1 = PLC_DOWNLOAD(Var2,	Via HMI to

Chapter 24 Macro

Category	Function	Expression	Description
		Var3, Var4, Var5, Var6)	download DVP or ISP file to PLC
	GetCircleCenter	Var1 = GetCircleCenter(Var2, Var3)	Calculate the Coordinate of Circle Center
Industrial Application	ECAM	Var1 = ECAM(Var2, Var3, Var4, Var5) (DW)	E-Cam Table

24-3-1 Arithmetic Operation

Arithmetic operations are divided into integer operations and floating point operations and relevant macro usages are detailed below.

+
-
*
/
%
MUL64
ADDSUMW
<hr/>
FADD
FSUB
FMUL
FDIV
FMOD
<hr/>
SIN
COS
TAN
COT
SEC
CSC

Figure 24-3-1-1 Arithmetic Operation

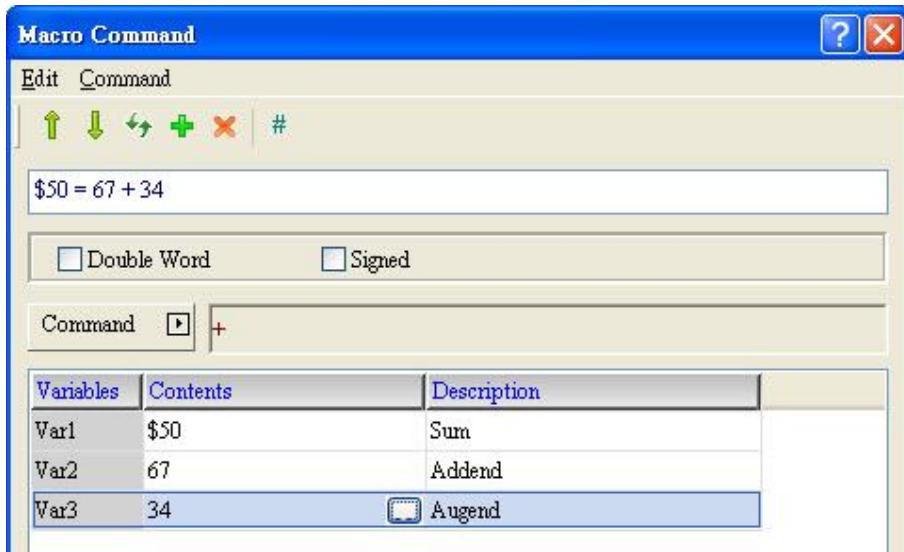
■ + (Addition)

Expression	What Variables Represent		NOTE
Var 1 = Var 2 + Var 3 (W) Var 1 = Var 2 + Var 3 (DW) Var 1 = Var 2 + Var 3 (Signed W) Var 1 = Var 2 + Var 3 (Signed DW)	Var 1	Sum	W: Word DW: Double Word Signed: Signed Number
	Var 2	Addend	
	Var 3	Augend	
	Expression Explanation		
Add Var 2 to Var 3 and save the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address and Var 2 and Var 3 are both constants.



- Use addition command for $\$50 = 67 + 34$, it will perform $67 + 34$ and put results to $\$50$. Therefore $\$50$ will display 101.

■ - (Subtraction)

Expression	What Variables Represent		NOTE	
Var 1 = Var 2 - Var 3 (W) Var 1 = Var 2 - Var 3 (DW) Var 1 = Var 2 - Var 3 (Signed W) Var 1 = Var 2 - Var 3 (Signed DW)	Var 1	Difference	W: Word DW: Double Word Signed: Signed number	
	Var 2	Subtrahend		
	Var 3	Minuend		
	Expression Explanation			
	Subtract Var 2 from Var 3 and save the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address and Var 2 and Var 3 are both constants.

Macro Command

Edit Command

↑ ↓ ← → + × #

\$50 = 67 - 34

Double Word Signed

Command [-]

Variables	Contents	Description
Var1	\$50	Difference
Var2	67	Subtrahend
Var3	34	Minuend

- Use subtraction command for \$50 = 67 - 34, it will perform 67 - 34 and put results to \$50. Therefore \$50 will display 33.

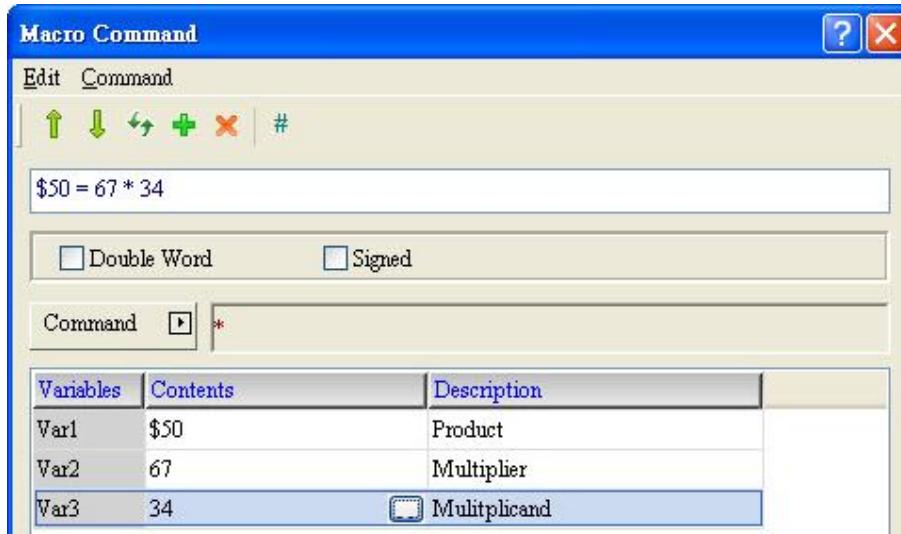
■ * (Multiplication)

Expression	What Variables Represent		NOTE	
Var 1 = Var 2 * Var 3 (W) Var 1 = Var 2 * Var 3 (DW) Var 1 = Var 2 * Var 3 (Signed W) Var 1 = Var 2 * Var 3 (Signed DW)	Var 1	Product	W : Word DW : Double Word Signed : Signed number	
	Var 2	Multiplier		
	Var 3	Multiplicand		
	Expression Explanation			
	Multiply Var 2 and Var 3 and save the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address and Var 2 and Var 3 are both constants.



- Use multiplication command for $\$50 = 67 * 34$, it will perform $67 * 34$ and put results to \$50. Therefore \$50 will display 2278.

■ / (Division)

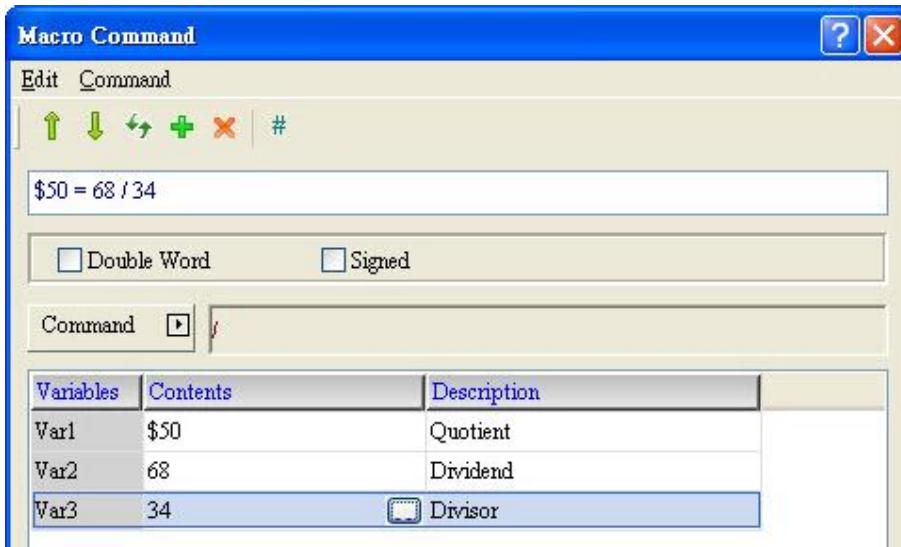
Expression	What Variables Represent		NOTE	
Var 1 = Var 2 / Var 3 (W) Var 1 = Var 2 / Var 3 (DW) Var 1 = Var 2 / Var 3 (Signed W) Var 1 = Var 2 / Var 3 (Signed DW)	Var 1	Quotient	W: Word DW: Double Word Signed: Signed number	
	Var 2	Dividend		
	Var 3	Divisor		
	Expression Explanation			
	Divide Var 2 by Var 3, and save the result (Quotient) in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Note: Var 3 can not be 0

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.



- Use division command for $\$50 = 68 / 34$, it will perform $68 / 34$ and put results to \$50. Therefore \$50 will display 2.

■ % (Get Reminder)

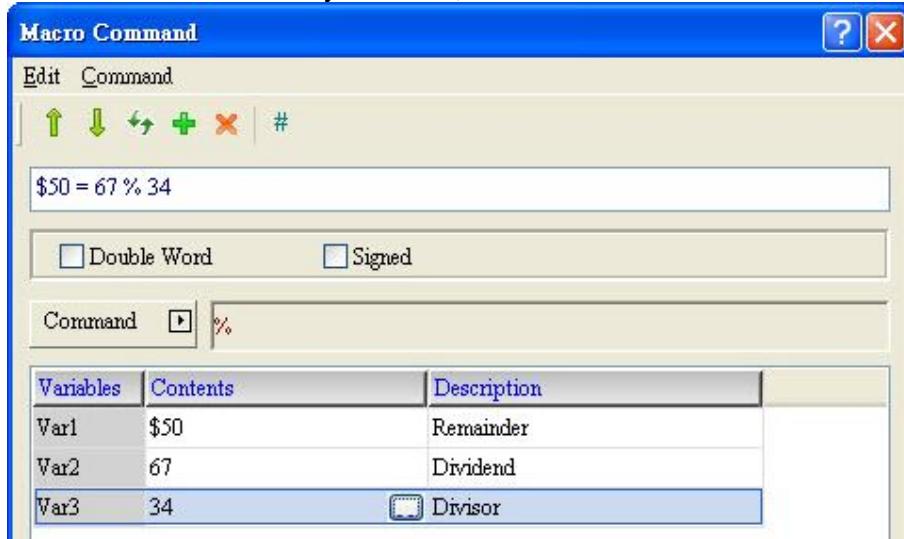
Expression	What Variables Represent		NOTE
Var 1 = Var 2 % Var 3 (W) Var 1 = Var 2 % Var 3 (DW) Var 1 = Var 2 % Var 3 (Signed W) Var 1 = Var 2 % Var 3 (Signed DW)	Var 1	Reminder	W: Word DW: Double Word Signed: Signed number
	Var 2	Dividend	
	Var 3	Divisor	
	Expression Explanation		
Divide Var 2 by Var 3, and save the result (remainder) in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Note: Var 3 can not be 0

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.



- Use get remainder command for $\$50 = 67 \% 34$, it will perform $67 \% 34$ and put results to \$50. Therefore \$50 will display 33.

■ MUL64 (64 bit Multiplication)

Expression	What Variables Represent		NOTE	
Var1 = MUL64(Var2, Var3) (W) Var1 = MUL64(Var2, Var3) (DW) Var1 = MUL64(Var2, Var3) (Signed W) Var1 = MUL64(Var2, Var3) (Signed DW)	Var 1	Product	W : Word DW : Double Word Signed : Signed number	
	Var 2	Multiplier		
	Var 3	Multiplicand		
	Expression Explanation			
	Multiply Var 2 and Var 3, and save the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.
- If checked Double Word, please also set data type of element for Double Word.



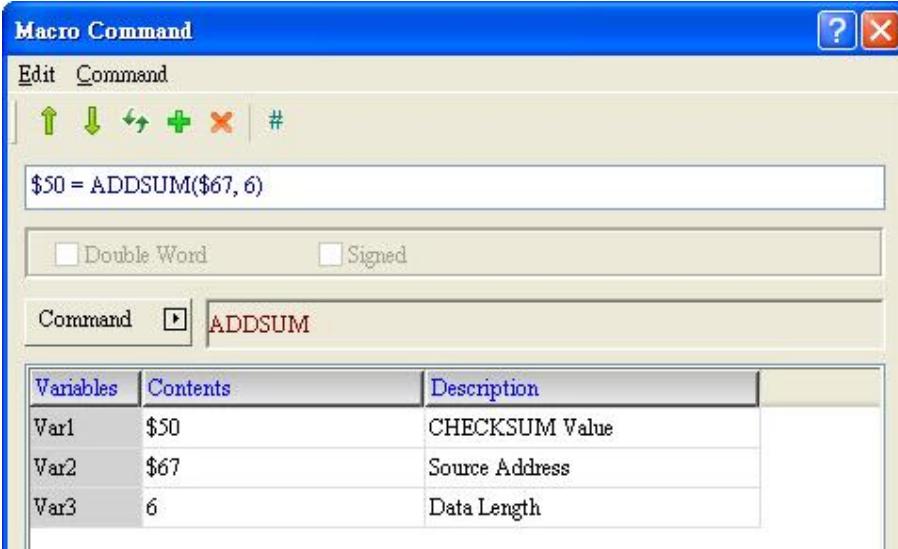
- Use MUL64 command for \$50 =MUL64(67 , 34), it will perform MUL64(67 , 34) and put results to \$50. Therefore \$50 will display 2278.

■ ADDSUMW (Repeated Addition)

Expression	What Variables Represent		NOTE	
Var1 = ADDSUMW(Var2, Var3) (W) Var1 = ADDSUMW(Var2, Var3) (DW)	Var 1	Result of repeated addition	W : Word DW : Double Word	
	Var 2	Starting address		
	Var 3	Length of repeated addition to be performed since the first item		
	Expression Explanation			
Repeatedly add from Var 2 to later variables (until length specified in Var3), and save the result in Var 1.				
Note: Repeated addition is performed with a rate of 2 bits from the starting address if selecting Double Word; repeated addition is performed with a rate of 1 bit from the starting address if selecting Word.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example
<ul style="list-style-type: none"> ➤ Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant. Please tick the Double Word box. ➤ Add up the values contained within the memory address of \$0 forward for a length of 5 with an interval of 2 in between, and variables added: \$0, \$2, \$4, \$6 and \$8.



The screenshot shows the 'Macro Command' dialog box. The command line at the top contains '\$50 = ADDSUM(\$67, 6)'. Below it, there are two checkboxes: 'Double Word' and 'Signed'. The 'Command' dropdown is set to 'ADDSUM'. A table below lists variables, their contents, and descriptions:

Variables	Contents	Description
Var1	\$50	CHECKSUM Value
Var2	\$67	Source Address
Var3	6	Data Length

➤ Input value for \$0 = 1, \$2 = 2, \$4 = 3, \$6 = 4 and \$8 = 5, and then perform accumulation to \$50. Therefore \$50 will display 15.

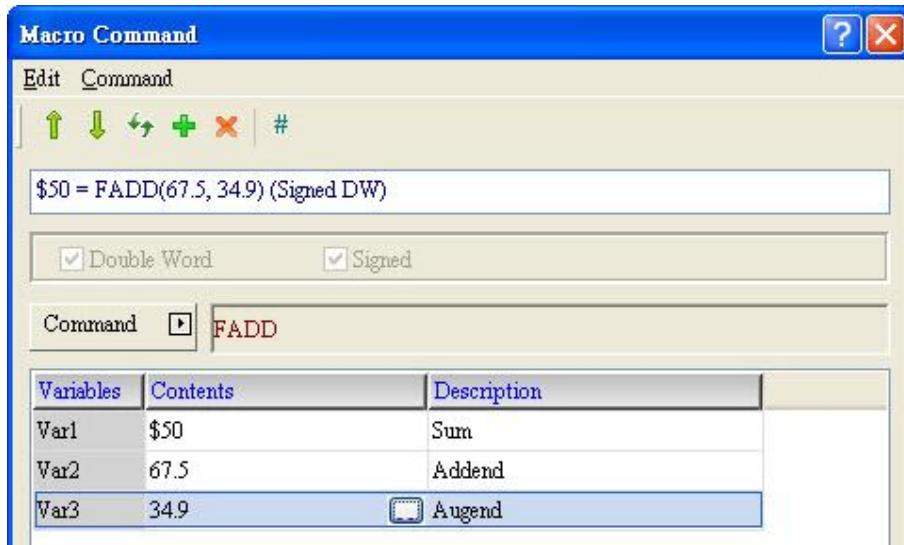
■ FADD (Floating Point Addition)

Expression	What Variables Represent		NOTE	
Var1 = FADD(Var2, Var3) (Signed DW)	Var 1	Sum	DW : Double Word Signed : Signed number	
	Var 2	Addend		
	Var 3	Augend		
	Expression Explanation			
	Multiply Var 2 and Var 3, and save and the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.



- Use FADD command for \$50 = FADD(67.5 , 34.9), it will perform FADD(67.5 , 34.9) and put results to \$50. Therefore \$50 will display 102.4.

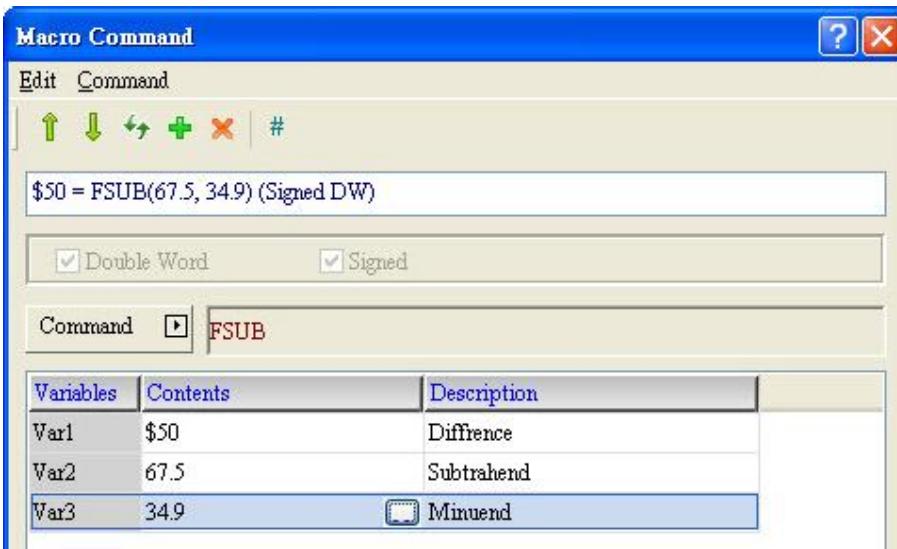
■ FSUB (Floating Point Subtraction)

Expression	What Variables Represent		NOTE	
Var1 = FSUB(Var2, Var3) (Signed DW)	Var 1	Difference	DW : Double Word Signed : Signed number	
	Var 2	Subtrahend		
	Var 3	Minuend		
	Expression Explanation			
	Subtract Var 2 from Var 3 and save the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are constants.



- Use FSUB command for \$50 = FSUB(67.5 , 34.9), it will perform FSUB(67.5 , 34.9) and put results to \$50. Therefore \$50 will display 32.6.

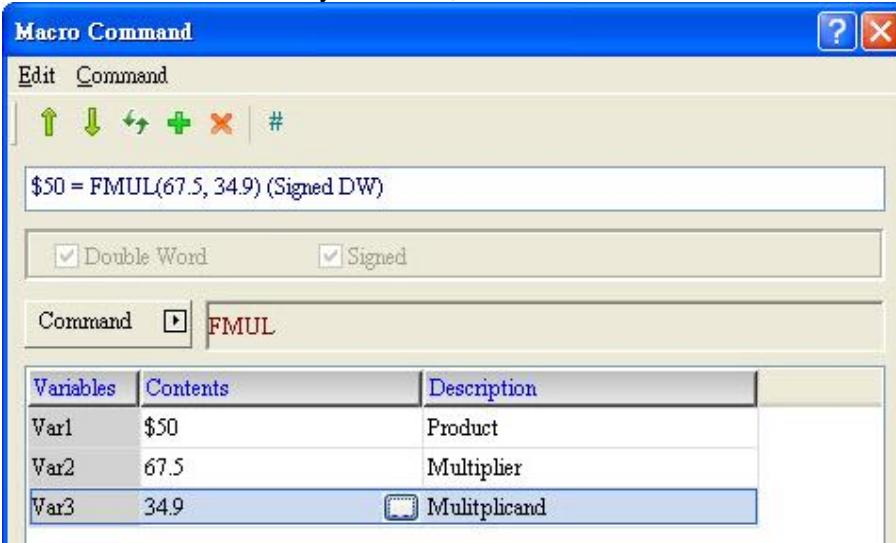
■ FMUL (Floating Point Multiplication)

Expression	What Variables Represent		NOTE	
Var1 = FMUL(Var2, Var3) (Signed DW)	Var 1	Product	DW : Double Word Signed : Signed number	
	Var 2	Multiplier		
	Var 3	Multiplicand		
	Expression Explanation			
	Multiply Var 2 and Var 3 and save the result in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address, Var 2 and Var 3 are both constants.



- Use FMUL command for \$50 = FMUL(67.5 , 34.9), it will perform FMUL(67.5 , 34.9) and put results to \$50. Therefore \$50 will display 2455.75.

■ FDIV (Floating Point Division)

Expression	What Variables Represent		NOTE
Var1 = FDIV(Var2, Var3) (Signed DW)	Var 1	Quotient	DW : Double Word Signed : Signed number
	Var 2	Dividend	
	Var 3	Divisor	
	Expression Explanation		
Divide Var 2 by Var 3, and save the result (Quotient) in Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is the internal memory address, and Var 2 and Var 3 are both constants.

Macro Command

Edit Command

↑ ↓ ← → + × #

\$50 = FDIV(67.5, 34.9) (Signed DW)

Double Word Signed

Command: FDIV

Variables	Contents	Description
Var1	\$50	Quotient
Var2	67.5	Dividend
Var3	34.9	Divisor

- Use FDIV command for \$50 = FDIV(67.5 , 34.9), it will perform FDIV(67.5 , 34.9) and put results to \$50. Therefore \$50 will display 1.934.

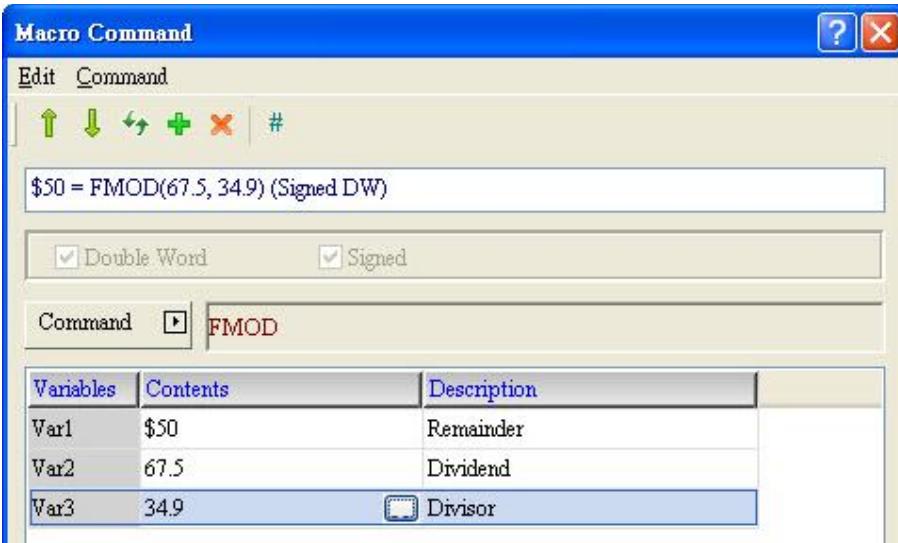
■ FMOD (Floating Point Reminder)

Expression	What Variables Represent		NOTE
Var1 = FMOD(Var2, Var3) (Signed DW)	Var 1	Reminder	DW : Double Word Signed : Signed number
	Var 2	Dividend	
	Var 3	Divisor	
	Expression Explanation		
	Divide Var 2 by Var 3, and save the result (remainder) in Var 1.		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are constants.



The dialog box shows the following settings:

- Macro Command window title.
- Toolbar with icons for Edit, Command, and navigation.
- Text input field: \$50 = FMOD(67.5, 34.9) (Signed DW).
- Checkboxes: Double Word (checked), Signed (checked).
- Command dropdown: FMOD.
- Variables table:

Variables	Contents	Description
Var1	\$50	Reminder
Var2	67.5	Dividend
Var3	34.9	Divisor

- Use FMOD command for \$50 = FMOD(67.5 , 34.9), it will perform FMOD(67.5 , 34.9) and put results to \$50. Therefore \$50 will display 32.6.

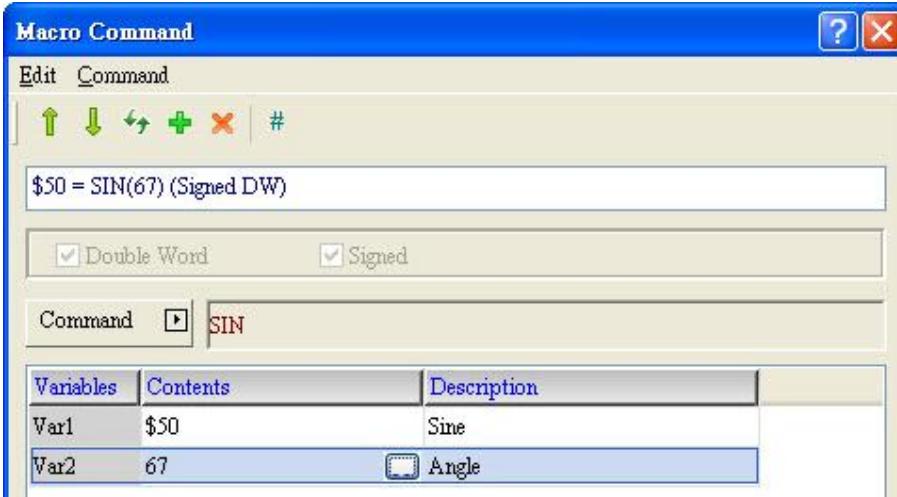
■ SIN (Sine Function)

Expression	What Variables Represent		NOTE				
Var1 = SIN(Var2) (Signed DW)	Var 1	Sine	DW : Double Word Signed : Signed number				
	Var 2	angle (in radians)					
	Expression Explanation						
	Perform the sine function operation on V2, and store the remainder in V1.						
*Data type of Var 1 must be Floating Point.							
*Data type of Var 2 must be Signed Decimal and can not have decimal places.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 is an internal memory address and Var 2 is a constant.



The screenshot shows the 'Macro Command' dialog box. The command entered is '\$50 = SIN(67) (Signed DW)'. The 'Double Word' and 'Signed' checkboxes are checked. In the 'Command' field, 'SIN' is selected. The 'Variables' table below shows 'Var1' assigned to '\$50' with a 'Sine' description, and 'Var2' assigned to '67' with an 'Angle' description.

- Use SIN command for \$50 = SIN(67), it will perform SIN(67) and put results to \$50. Therefore \$50 will display 0.921. °

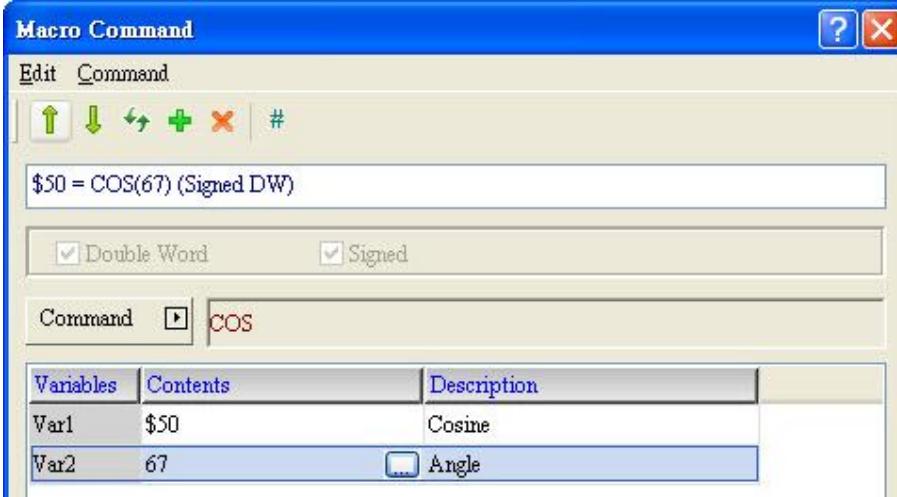
■ COS (Cosine Function)

Expression	What Variables Represent		NOTE				
Var1 = COS(Var2) (Signed DW)	Var 1	Cosine	DW : Double Word Signed : Signed number				
	Var 2	Angle (in radians)					
	Expression Explanation						
	Perform the cosine function operation on V2, and store the remainder in V1.						
*Data type of Var 1 must be Floating Point.							
*Data type of Var 2 must be Signed Decimal and can not have decimal places.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.



The screenshot shows the 'Macro Command' dialog box. In the 'Command' field, 'cos' is selected. Below it, 'Double Word' and 'Signed' checkboxes are checked. The 'Variables' tab of the table shows two rows: 'Var1' with value '\$50' and 'Description' 'Cosine', and 'Var2' with value '67' and 'Description' 'Angle'. The 'Contents' tab is also visible.

- Use COS command for $\$50 = \text{COS}(67)$, it will perform $\text{COS}(67)$ and put results to $\$50$. Therefore $\$50$ will display 0.391.

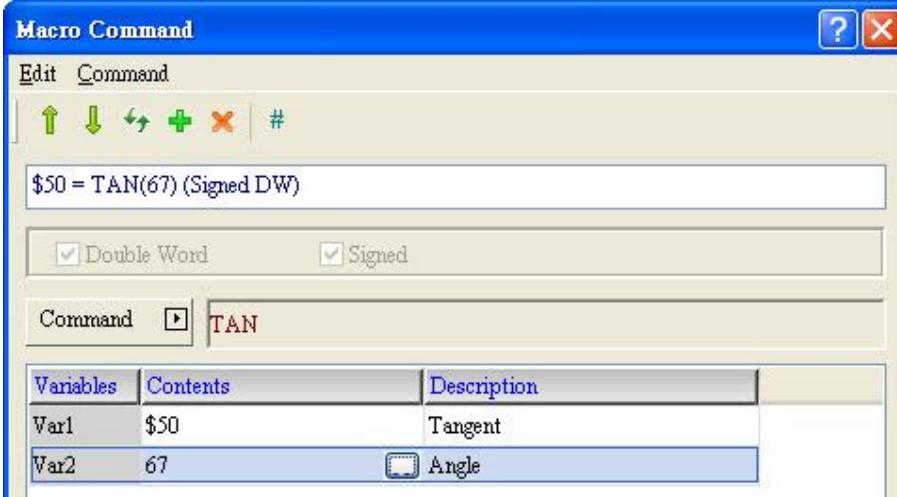
■ TAN (Tangent Function)

Expression	What Variables Represent		NOTE				
Var1 = TAN(Var2) (Signed DW)	Var 1	Tangent	DW : Double Word Signed : Signed number				
	Var 2	angle (in radians)					
	Expression Explanation						
	Perform the tangent function operation on V2, and store the remainder in V1.						
*Data type of Var 1 must be Floating Point.							
*Data type of Var 2 must be Signed Decimal and can not have decimal places.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.



The screenshot shows the 'Macro Command' dialog box. In the 'Command' field, 'TAN' is selected. Below it, 'Double Word' is checked under 'Type'. In the 'Variables' table, 'Var1' is assigned to '\$50' with a 'Tangent' description, and 'Var2' is assigned to '67' with an 'Angle' description.

- Use TAN command for \$50 = TAN(67), it will perform TAN(67) and put results to \$50. Therefore \$50 will display 2.356.

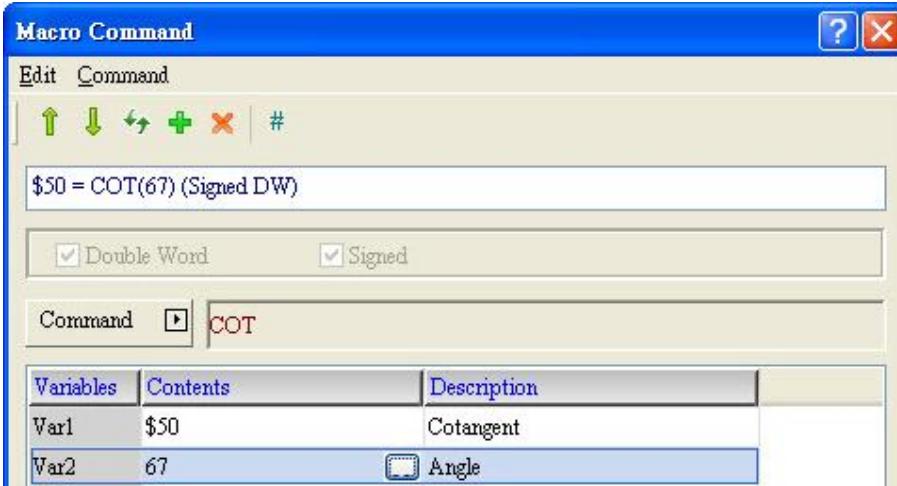
■ COT (Cotangent Function)

Expression	What Variables Represent		NOTE				
Var1 = COT(Var2) (Signed DW)	Var 1	Cotangent	DW : Double Word Signed : Signed number				
	Var 2	angle (in radians)					
	Expression Explanation						
	Perform the cotangent function operation on V2, and store the remainder in V1.						
*Data type of Var 1 must be Floating Point.							
*Data type of Var 2 must be Signed Decimal and can not have decimal places.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.



The screenshot shows the 'Macro Command' dialog box. The command entered is '\$50 = COT(67) (Signed DW)'. Under 'Command', 'COT' is selected. In the 'Variables' table, 'Var1' is assigned '\$50' with a 'Cotangent' description, and 'Var2' is assigned '67' with an 'Angle' description. Checkboxes for 'Double Word' and 'Signed' are checked.

- Use COT command for \$50 = COT(67), it will perform COT(67) and put results to \$50. Therefore \$50 will display 0.424.

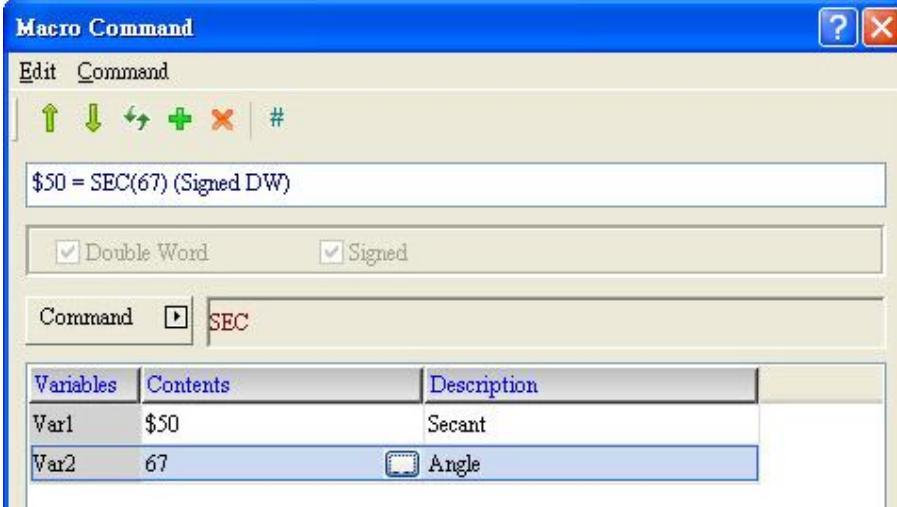
■ SEC (Secant Function)

Expression	What Variables Represent		NOTE				
Var1 = SEC(Var2) (Signed DW)	Var 1	Secant	DW : Double Word Signed : Signed number				
	Var 2	angle (in radians)					
	Expression Explanation						
	Perform the secant function operation on V2, and store the remainder in V1.						
*Data type of Var 1 must be Floating Point.							
*Data type of Var 2 must be Signed Decimal and can not have decimal places.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.



The screenshot shows the 'Macro Command' dialog box. In the 'Command' field, 'SEC' is selected. Below it, 'Double Word' and 'Signed' are checked. The expression '\$50 = SEC(67) (Signed DW)' is entered in the main input field. A table at the bottom lists variables: Var1 is \$50 (Secant), and Var2 is 67 (Angle).

- Use SEC command for $\$50 = \text{SEC}(67)$, it will perform SEC(67) and put results to \$50. Therefore \$50 will display 2.559.

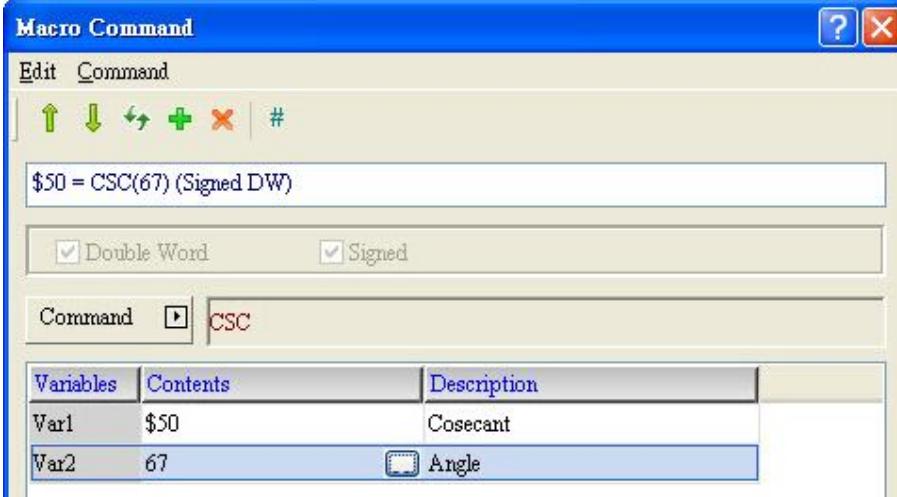
■ CSC (Cosecant Function)

Expression	What Variables Represent		NOTE				
Var1 = CSC(Var2) (Signed DW)	Var 1	Cosecant	DW : Double Word Signed : Signed number				
	Var 2	angle (in radians)					
	Expression Explanation						
	Perform the cosecant function operation on V2, and store the remainder in V1.						
*Data type of Var 1 must be Floating Point.							
*Data type of Var 2 must be Signed Decimal and can not have decimal places.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.



The screenshot shows the 'Macro Command' dialog box. The 'Command' field contains '\$50 = CSC(67) (Signed DW)'. Below it, under 'Double Word' and 'Signed' checkboxes, are checked. The 'Command' dropdown shows 'CSC'. In the bottom table, 'Var1' is mapped to '\$50' with a 'Cosecant' description, and 'Var2' is mapped to '67' with an 'Angle' description.

- Use CSC command for \$50 = CSC(67), it will perform CSC(67) and put results to \$50. Therefore \$50 will display 1.086.

24-3-2 Logical Operation

Logical operations include six operators and relevant macro usages are detailed below.



Figure 24-3-2-1 Logical Operations

■ | (Bitwise OR Operation)

Bitwise OR Operation characteristic	
Operator	Result
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	1

Expression	What Variables Represent		NOTE
Var 1 = Var 2 Var 3 (W) Var 1 = Var 2 Var 3 (DW)	Var 1	Result of OR operation	W : Word DW : Double Word
	Var 2	Logical operand	
	Var 3	Logical operand	
	Expression Explanation		
Perform the Bitwise OR operation on Var 2 and Var 3 and save the result Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example
➤ Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.

Macro Command

Edit Command

\$50 = 2 | 3

Double Word Signed

Command: |

Variables	Contents	Description
Var1	\$50	Result
Var2	2	Logical Variable
Var3	3	Logical Variable

➤ This command will transfer values for Var 2 and Var 3 to binary like 2 = 0010, 3 = 0011, and then bitwise OR operator with 0010 and 0011. After bitwise OR operator to get the result is 0011 and equals to 3.

Binary Value	
0010	2
0011	3
<hr/>	
0011	3

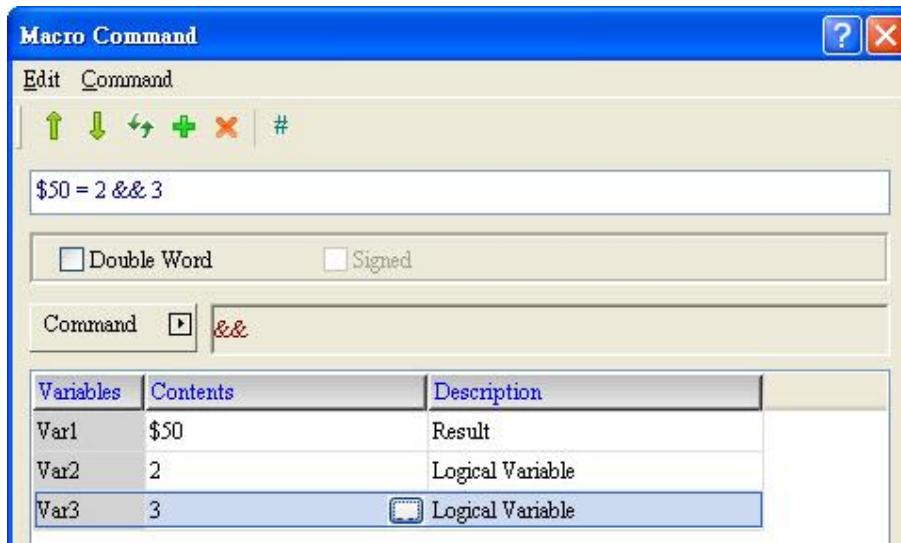
■ && (Bitwise AND Operation)

Bitwise AND Operation characteristic	
Operator	Result
0 OR 0	0
0 OR 1	0
1 OR 0	0
1 OR 1	1

Expression	What Variables Represent		NOTE
Var 1 = Var 2 && Var 3 (W) Var 1 = Var 2 && Var 3 (DW)	Var 1	Result of Bitwise AND operation	W : Word DW : Double Word
	Var 2	Logical Operand	
	Var 3	Logical Operand	
	Expression Explanation		
Perform the Bitwise AND operation on Var 2 and Var 3 and save the result Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example	
➤ Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.	



- This command will transfer values for Var 2 and Var 3 to binary like 2 = 0010, 3 = 0011, and then bitwise AND operator with 0010 and 0011. After bitwise AND operator to get the result is 0010 and equals to 2.

Binary Value

0010	2
0011	3

Binary Operator

0010	2
-------------	---

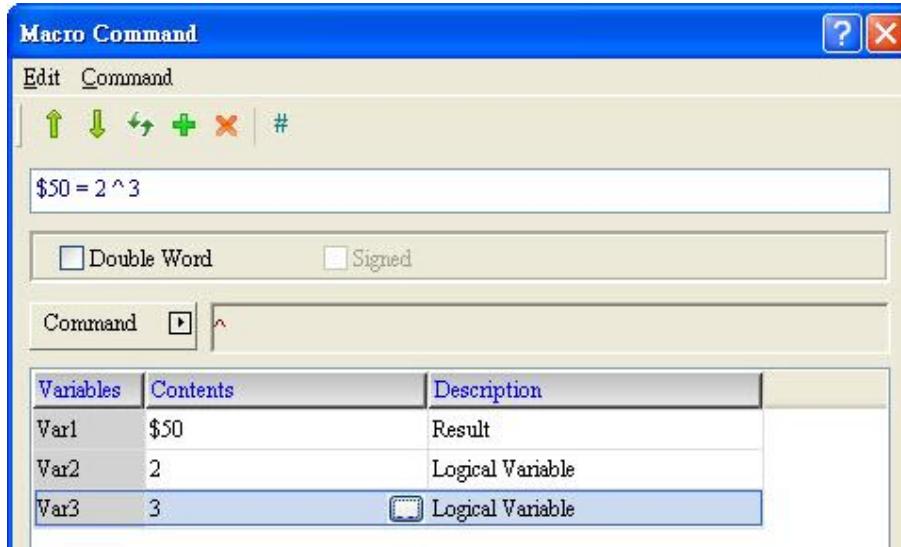
■ ^ (Bitwise XOR Operation)

Bitwise XOR Operation characteristic	
Operator	Result
0 OR 0	0
0 OR 1	1
1 OR 0	1
1 OR 1	0

Expression	What Variables Represent		NOTE
Var 1 = Var 2 ^ Var 3 (W) Var 1 = Var 2 ^ Var 3 (DW)	Var 1	Result of Bitwise XOR operation	W : Word DW : Double Word
	Var 2	Logical Operand	
	Var 3	Logical Operand	
	Expression Explanation		
Perform the Bitwise XOR operation on Var 2 and Var 3 and save the result Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

- Example**
- Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.



- This command will transfer values for Var 2 and Var 3 to binary like 2 = 0010, 3 = 0011, and then bitwise XOR operator with 0010 and 0011. After bitwise XOR operator to get the result is 0001 and equals to 1.

	Binary	Value
2	0010	2
3	0011	3
	0001	1

■ NOT (Bitwise NOT Operation)

Bitwise NOT Operation characteristic	
Operator	Result
NOT 0	1
NOT 1	0

Expression	What Variables Represent	NOTE
Var 1 = NOT Var 2 (W) Var 1 = NOT Var 2 (DW) Var 1 = NOT Var 2 (Signed W) Var 1 = NOT Var 2 (Signed DW)	Var 1 Result of Bitwise NOT Operation	W : Word DW : Double Word Signed : Signed number
	Var 2 Logical Operand	
	Expression Explanation	
	Perform the Bitwise NOT operation on Var 2 and save the result in Var 1.	

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.

Macro Command

Edit Command

↑ ↓ ↶ ↷ #

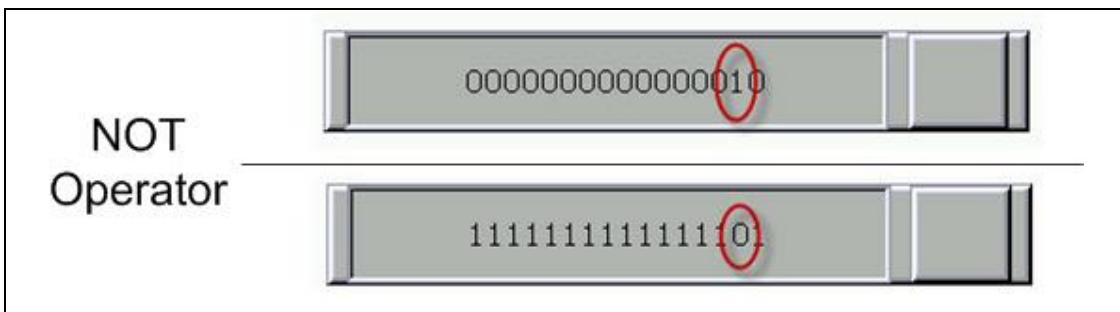
\$50 = NOT 2

Double Word Signed

Command: NOT

Variables	Contents	Description
Var1	\$50	Result
Var2	2	Logical Variable

- This command will transfer values for Var 2 to binary like $2 = 00000000000000010$, and then bitwise NOT operator with 00000000000000010 . After bitwise NOT operator to get the result is 111111111111101 .



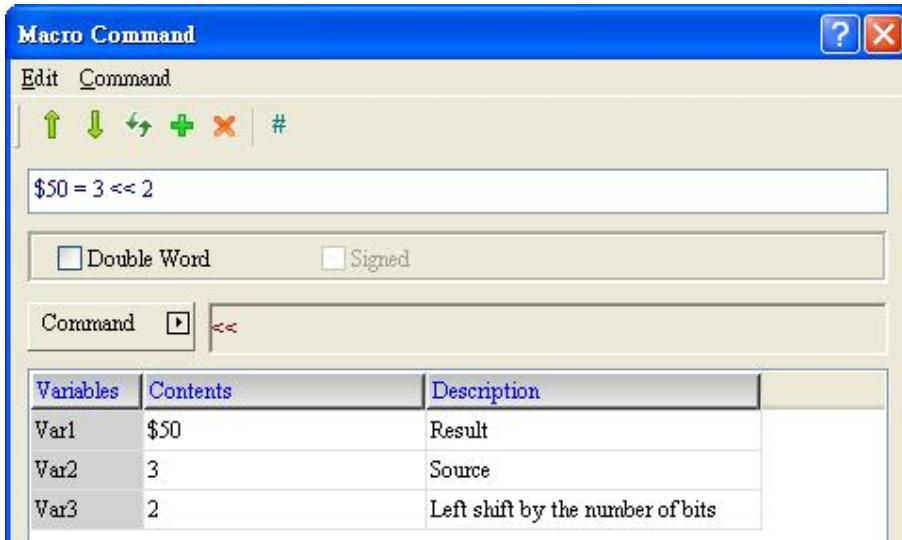
■ << (SHL Bitwise Left-shift Operation)

Expression	What Variables Represent		NOTE	
Var1 = Var2 << Var3 (W) Var1 = Var2 << Var3 (DW)	Var 1	Result of left-shifted value	W : Word DW : Double Word	
	Var 2	Source Address for the left-shift operation		
	Var 3	Numbers of bits to shift		
	Expression Explanation			
	Left-shift Var 2 by Var 3 bits and save the result in Var 1.			

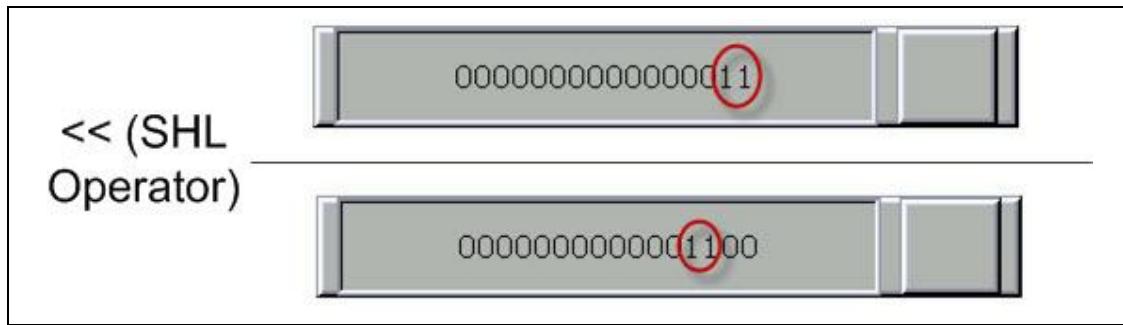
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



- This command will transfer values for Var 2 to binary like $3 = 0000000000000011$, and then bitwise left-shift two bits with 0000000000000011 . After bitwise left-shift operator to get the result is 0000000000001100 .



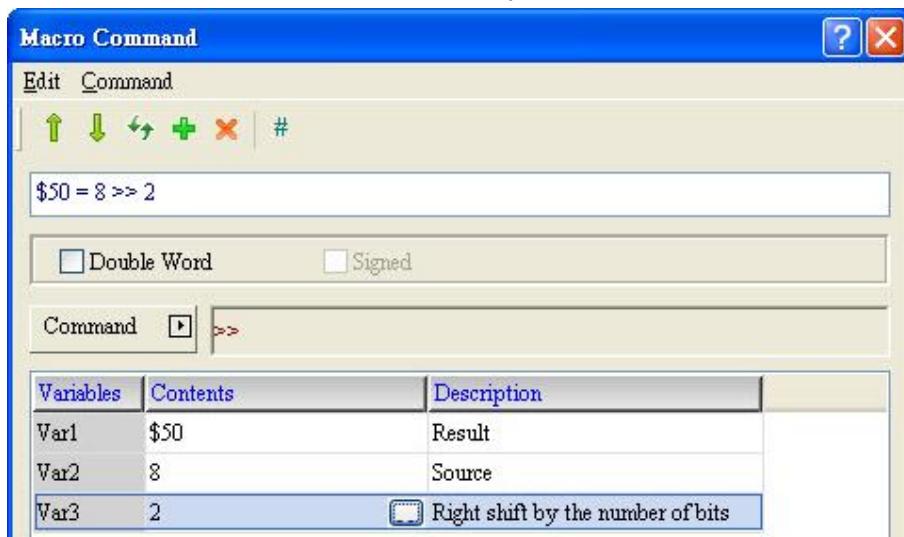
■ >> (SHR Bitwise Right-shift Operation)

Expression	What Variables Represent		NOTE
Var1 = Var2 >> Var3 (W) Var1 = Var2 >> Var3 (DW)	Var 1	Result of right-shifted value	W : Word DW : Double Word
	Var 2	Source Address for the right-shift operation	
	Var 3	Number of bits to shift	
	Expression Explanation		
Right-shift Var 2 by Var 3 bits and save the result in Var 1.			

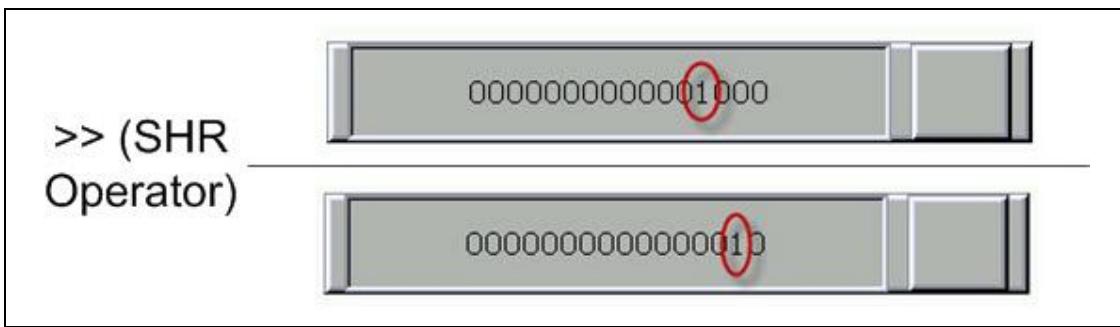
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



- This command will transfer values for Var 2 to binary like $8 = 0000000000001000$, and then bitwise right-shift two bits with 0000000000001000 . After bitwise right-shift operator to get the result is 0000000000000010 .



24-3-3 Data Transfer

There are six commands for data transfer and they are detailed below:

MOV
BMOV
ArrayCopy
FILL
FILLASC
FMOV

Figure 24-3-3-1 Data Transfer

■ MOV (Data Moving Operand)

Expression	Context		NOTE	
Var 1 = Var 2 (W) Var 1 = Var 2 (DW) Var 1 = Var 2 (Signed W) Var 1 = Var 2 (Signed DW)	Var 1	Destination Address	W : Word DW : Double Word Signed : Signed number	
	Var 2	Source Data		
	Expression Explanation			
	Copy source data contained in Var 2 to Var 1, and data contained in Var 2 will not change.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎

Example
<ul style="list-style-type: none"> ➤ Var 1 is an internal memory address, and Var 2 is a constant. <p>The Macro Command dialog box shows the following settings: - Command: \$50 = \$67 - Destination: Var1 (\$50) - Source: Var2 (\$67) - Options: Double Word (unchecked), Signed (unchecked)</p>
<ul style="list-style-type: none"> ➤ Use MOV command for \$50 = \$67, it will perform move with input value at \$67 to \$50. If input value is 34 at \$67, therefore \$50 will display 34.

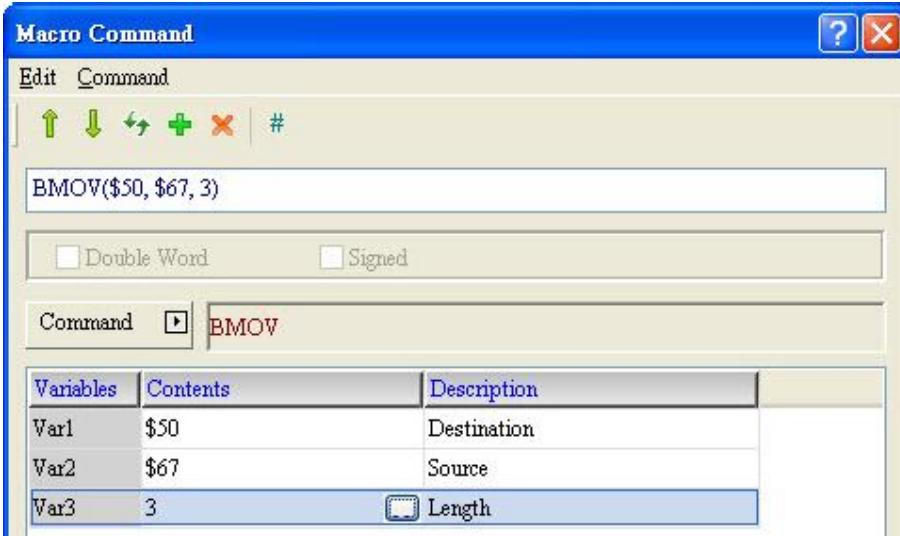
■ BMOV (Move in Block)

Expression	What Variables Represent		NOTE	
BMOV(Var1, Var2, Var3) (W)	Var 1	Destination Address	W : Word Bulk-copy the entire block with the range specified in Var 3 from Var 2 to Var 1.	
	Var 2	Source Data Address		
	Var 3	Word Data Length		
	Expression Explanation			
	Bulk-copy the entire block with the range specified in Var 3 from Var 2 to Var 1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	
Var 3	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



- It will perform move 3 data length from \$67 to \$50. Therefore when input value at \$67, \$68, \$69 will move data to \$50, \$51, \$52.



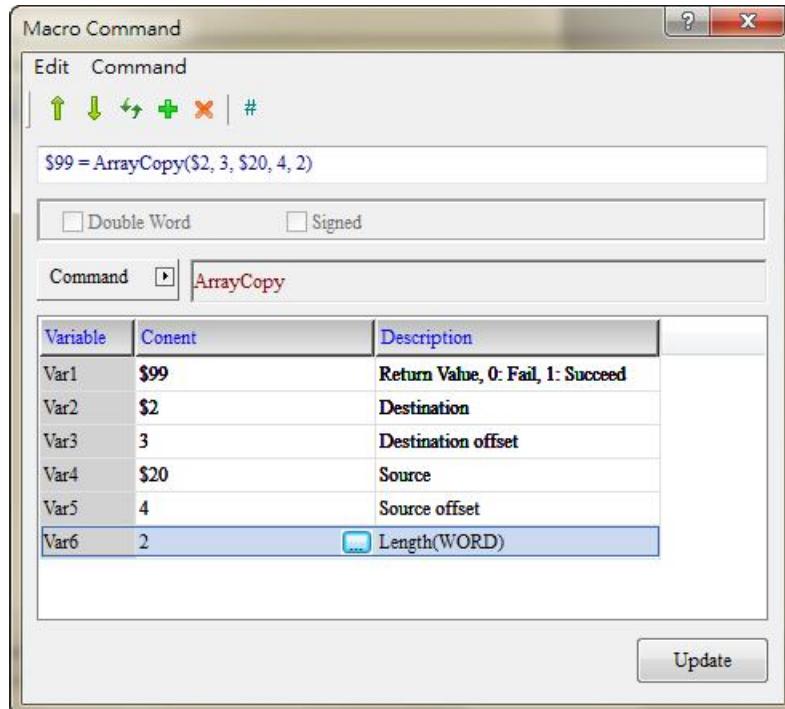
■ ArrayCopy (Copy Array)

Expression	What Variables Represent			NOTE			
Var1 = ArrayCopy(Var2, Var3, Var4, Var5, Var6)	Var 1	Returned Value					
		Failure	0				
		Success	1				
	Var 2	Target Address					
	Var 3	Target Offset Amount					
	Var 4	Source Address					
	Var 5	Source Offset Amount					
	Var 6	Moving Length (unit: WORD)					
	Expression Explanation						
Copy one section of consecutive address to another section.							

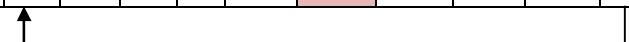
Variable	Memory Usage			
	Internal Memory	PLC Register	String	Constant
Var 1	◎			
Var 2	◎	◎		
Var 3	◎	◎		◎
Var 4	◎	◎		
Var 5	◎	◎		◎
Var 6	◎	◎		◎

Example

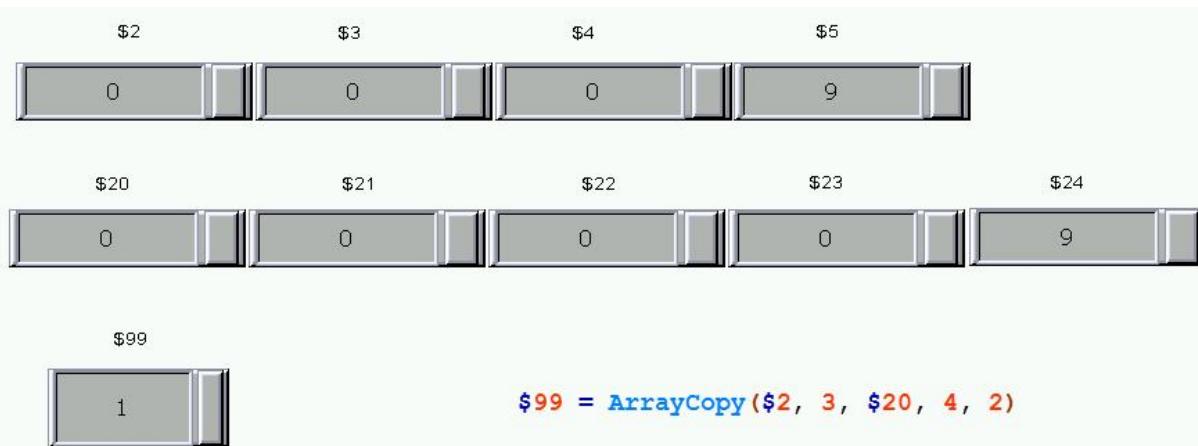
- \$99 = ArrayCopy(\$2, 3, \$20, 4, 2)



Address	\$0	\$1	\$2	\$3	\$4	\$5	\$6	\$7	...	\$19	\$20	\$21	\$22	\$24	\$25	\$26	...
Offset			+0	+1	+2	+3				+0	+1	+1	+2	+3	+4		...



Copy the offset amount (4, the representative address is \$24) specified by source address (\$20) to the offset amount (3, the representative address is \$5) specified by target address \$2. See the figure below:



■ FILL (Fill in the Block)

Expression	What Variables Represent		NOTE
FILL(Var1, Var2, Var3) (W) FILL(Var1, Var2, Var3) (Signed W)	Var 1	Destination Address of the Source Data	W : Word Copy the value contained in Var 2 to a number of variables starting from Var 1. (the number of variable is specified in Var 3)
	Var 2	Source value	
	Var 3	Length	
	Expression Explanation		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 are both constants.

The screenshot shows the 'Macro Command' dialog box. The 'Command' field contains 'FILL'. Below it, the 'Variables' table is populated with:

Variables	Contents	Description
Var1	\$50	Source Address
Var2	\$67	Content
Var3	3	Length

- Input value at \$67 and will fill the value for 3 data length from \$50, \$51 and \$52.

FILL (\$50, \$67, 3)

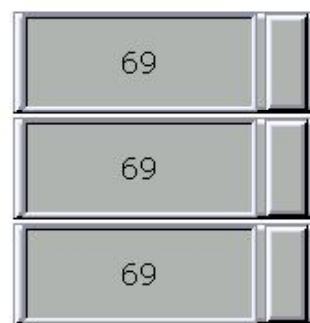
\$67



\$50

\$51

\$52



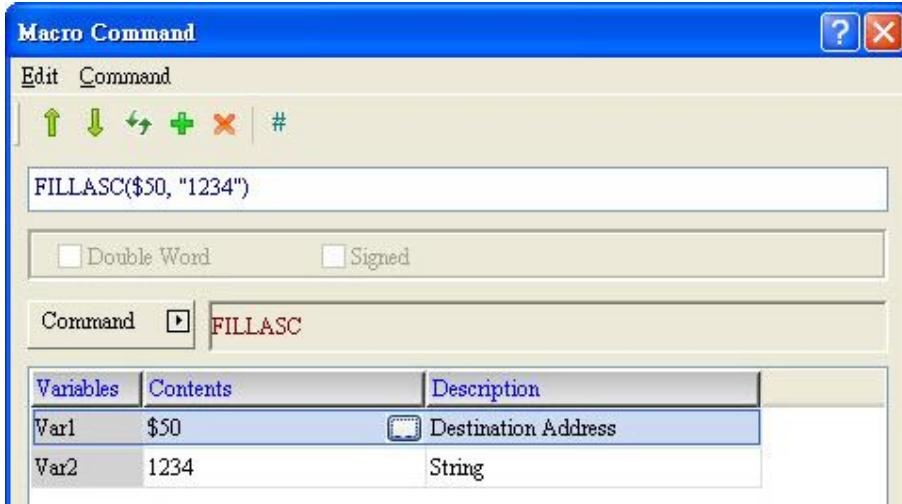
■ FILLASC (String to ASCII Conversion)

Expression	What Variables Represent		NOTE	
FILLASC(Var1, "Var2") (W)	Var 1	Destination address for the string	W : Word	
	Var 2	String		
	Expression Explanation			
	Convert the Var2 text string into corresponding ASCII values and save them in Var1.			

Variable	Memory Usage			
	Internal Memory	PLC Register	Constant	String
Var 1	◎	◎		
Var 2				◎

Example

- Var 1 is an internal memory address, and Var 2 is a text string.



- The result after conversion: \$50 = 3241H, \$51 = 3433H.

■ FMOV (Move floating point data)

Expression	What Variables Represent		NOTE
Var1 = FMOV(Var2) (Signed DW)	Var 1	Destination Address	DW : Double Word Signed : Signed number
	Var 2	Source data address	
	Expression Explanation		
Copy floating point source data contained in Var 2 to Var 1, and the floating point contained in Var 2 will not change.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎

Example

- Var 1 is an internal memory address and Var 2 is a constant.

Macro Command

Edit Command

↑ ↓ ↶ ↷ + × #

\$50 = FMOV(67.5) (Signed DW)

Double Word Signed

Command

Variables	Contents	Description
Var1	\$50	Destination
Var2	67.5	Source

- Put 67.5 floating data to \$50, therefore \$50 will display 67.5.

24-3-4 Data Conversion

Data conversion operations are performed with commands for data type conversion, maximum and minimal values and data swap, and their usages are detailed below:

BCD	XCHG
BIN	MAX
TODWORD	MIN
TOWORD	TOHEX
TOBYTE	TOASC
SWAP	
	FCNV
	ICNV
	SPRINTF

Figure 24-3-4-1 Data Conversion

■ BCD (Decimal to BCD Conversion)

Expression	What Variables Represent		NOTE
Var1 = BCD(Var2) (W)	Var 1	BCD type data	
Var1 = BCD(Var2) (DW)	Var 2	Decimal type data	
Expression Explanation			
Convert decimal data stored in Var2 into BCD data and save the converted data in Var1.			W : Word DW : Double Word

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		

Example
➤ Var 1 and Var 2 are internal memory addresses.

Macro Command

Edit Command

\$50 = BCD(\$67)

Double Word Signed

Command: BCD

Variables	Contents	Description
Var1	\$50	BCD Value
Var2	\$67	Decimal Value

➤ Transfer Unsigned Decimal format with \$67 to BCD format with \$50.

\$50
BCD format

\$67
Unsigned Decimal
format

The diagram illustrates the conversion process. On the left, under 'Unsigned Decimal format', the value '\$67' is shown. On the right, under 'BCD format', the value '\$50' is shown. Both values are displayed in a light gray rectangular box with a small handle at the bottom right corner, representing memory locations or registers.

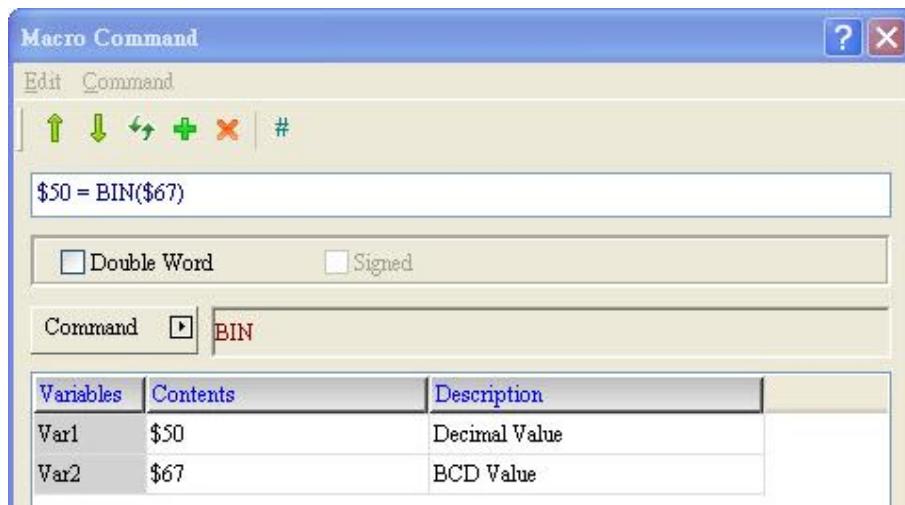
■ BIN (BCD to Decimal Conversion)

Expression	What Variables Represent		NOTE
Var1 = BIN(Var2) (W) Var1 = BIN(Var2) (DW)	Var 1	Decimal type data	W : Word DW : Double Word
	Var 2	BCD type data	
	Expression Explanation		
Convert BCD data stored in Var2 into decimal data and save the converted data in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		

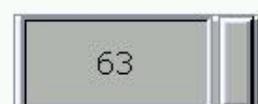
Example

- Var 1 and Var 2 are internal memory addresses.



- Transfer BCD format with \$67 to Unsigned Decimal format with \$50.

\$50
Unsigned Decimal
format



\$67
BCD format



■ TODWORD (WORD to Double WORD Conversion)

Expression	What Variables Represent		NOTE	
Var1 = TODWORD(Var2) (W) Var1 = TODWORD(Var2) (Signed W)	Var 1	Double Word type data	W : Word	
	Var 2	Word type data		
	Expression Explanation			
	Convert Word data stored in Var2 into Double Word data and save the converted data in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		

Example

- Var 1 and Var 2 are internal memory addresses.



- Convert the Word data originally stored in \$67 into Double Word data and store it in \$50. Since the converted data is a Double Word data, it actually takes up two addresses from \$50 to \$51.

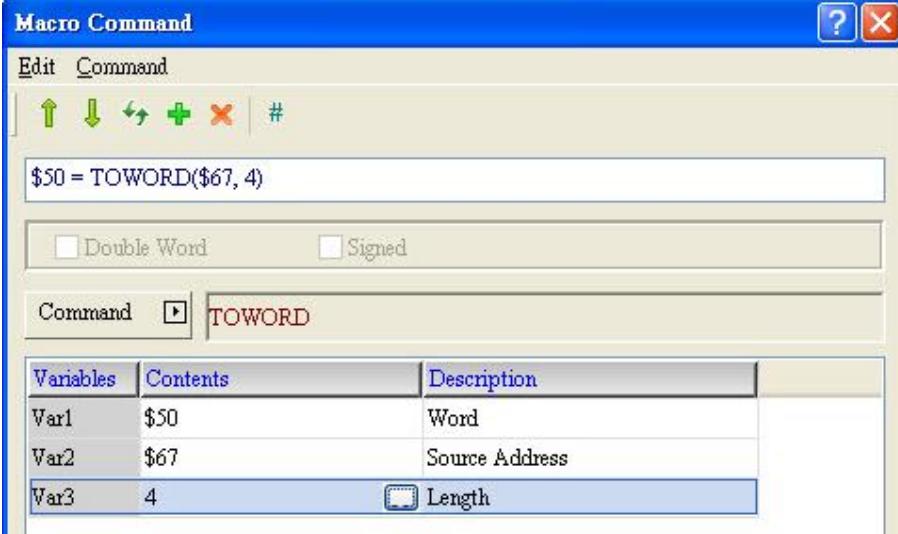
■ TOWORD (BYTE to Word Conversion)

Expression	What Variables Represent		NOTE	
Var1 = TOWORD(Var2, Var3) (W)	Var 1	Word type data	W : Word Convert BYTE data (number of byte is Var3) from Var2 to WORD value and store the result in Var1. The high byte will be filled with 0.	
	Var 2	Starting Address of the Source Data		
	Var 3	Length		
	Expression Explanation			
	Convert BYTE data (number of byte is Var3) from Var2 to WORD value and store the result in Var1. The high byte will be filled with 0.			
* As V2 is in word data type, each word stored in Var 2 can be converted in to two words. * Please note that the first and last byte of the converted WORD will be exchanged.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.



The screenshot shows the Macro Command dialog box. The command is set to TOWORD. The variables are listed as follows:

Variables	Contents	Description
Var1	\$50	Word
Var2	\$67	Source Address
Var3	4	Length

- Convert the 4 consecutive bytes in \$67 into the word type and save them into \$50.
- Both \$50 and \$67 are set to the Hex data type.
- If \$67 = AB67H and \$68 = 9A62H, then after conversion using the TOWORD command, then the result is: \$50 = 67H, \$51 = ABH, \$52 = 62H and \$53 = 9AH.

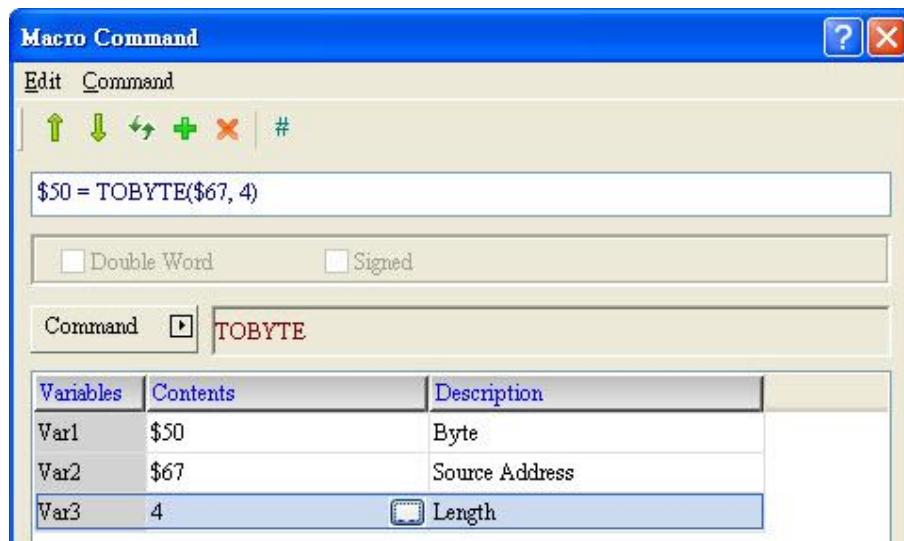
■ TOBYTE (Word to Byte Conversion)

Expression	What Variables Represent		NOTE				
Var1 = TOBYTE(Var2, Var3) (W)	Var 1	BYTE type data	W : Word Var1 = TOBYTE(Var2, Var3) (W)				
	Var 2	Starting Address of the Source Data					
	Var 3	Length					
	Expression Explanation						
Convert Word data (number of word is Var3) from low-byte of Var2 to Byte data (discard high-byte of Var2) and store the result in Var1.							
* Please note that the first and last byte of the converted WORD will be exchanged.							

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.



- Convert the 4 consecutive words in \$67 into the byte type and save them into \$50.
- Both \$50 and \$67 are set to the Hex data type.
- If \$67 = 12H, \$68 = 76H, \$69 = 24H and \$70 = ABH, then after conversion using the TOBYTE command, the result is: \$50 = 7612H, \$51 = AB24H.

■ SWAP (Swap between highbit and lowbit of WORD)

Expression	What Variables Represent		NOTE
SWAP(Var1, Var2, Var3) (W)	Var 1	Starting Address of the Destination Data	W : Word Var 1, Var 2, Var 3 are Internal Memory Addresses.
	Var 2	Starting Address of the Source Data	
	Var 3	Length	
	Expression Explanation		
Swap between the high-byte and low-byte of Var 2 until the Var2+Var3 item, and store the result in the starting position of Var1 forward.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.

The screenshot shows the 'Macro Command' dialog box. The 'Command' field contains 'SWAP(\$50, \$67, 2)'. Below it, there are checkboxes for 'Double Word' and 'Signed'. The 'Command' dropdown is set to 'SWAP'. The 'Variables' table below lists three variables: Var1 (\$50), Var2 (\$67), and Var3 (2). The 'Description' column indicates Var1 is the Destination, Var2 is the Source, and Var3 is the Length.

Variables	Contents	Description
Var1	\$50	Destination
Var2	\$67	Source
Var3	2	Length

- Swap between the high-byte and low-byte of value stored in \$67 and save the swapped result in \$50.
- Both \$50 and \$67 are set to the Hex data type.
- If 67 = 5612H and 68 = B724H, then after executing the SWAP command to swap the high-byte and low-byte of values stored within \$67 and save them into \$50 and \$51, the result will be \$50=1256H, \$51 = 24B7H.

■ XCHG (Data Exchange)

Expression	What Variables Represent		NOTE
XCHG(Var1, Var2, Var3) (W) XCHG(Var1, Var2, Var3) (DW)	Var 1	Starting Address of the Destination Data	W : Word DW : Double Word
	Var 2	Starting Address of the Source Data	
	Var 3	Length	
	Expression Explanation		
Exchange between values stored in Var 2 and Var 1 based on the length specified in Var 3.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.

The screenshot shows the 'Macro Command' dialog box. The command is set to 'XCHG(\$50, \$67, 2)'. The 'Command' dropdown is set to 'XCHG'. The 'Variables' table below lists:

Variables	Contents	Description
Var1	\$50	Destination
Var2	\$67	Source
Var3	2	Length

- Exchange between 2 bytes of data stored in \$67 and \$50.
- Both \$50 and \$67 are Hex data type.
- If \$67 = 1244H, \$68 = 5678H, \$50 = ABCDH and \$51 = EFDCH, then after executing XCHG to exchange between values stored in \$67 and \$68, and \$50 and \$51, then the result will be \$67 = ABCDH, \$68 = EFDCH, \$50 = 1244H and \$51 = 5678H.

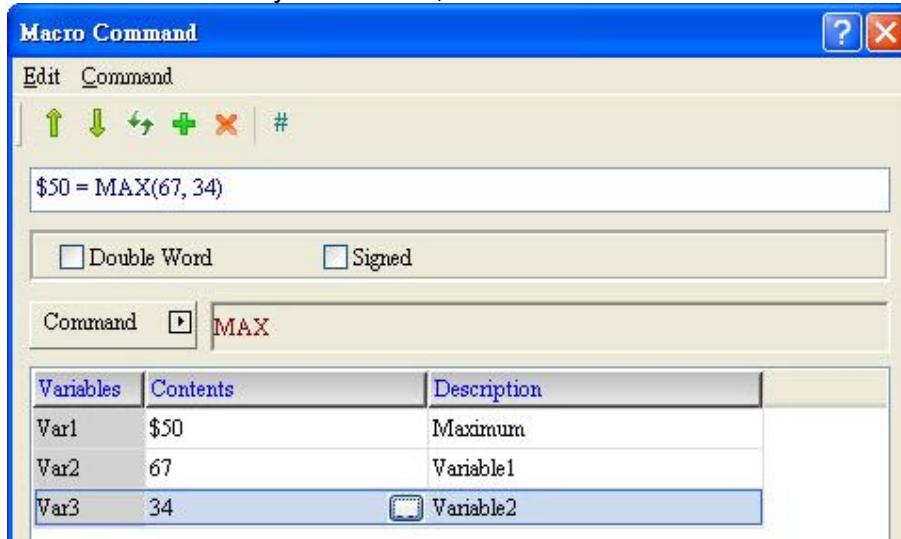
■ MAX (Get Maximum Value)

Expression	What Variables Represent		NOTE	
Var1 = MAX(Var2, Var3) (W) Var1 = MAX(Var2, Var3) (DW) Var1 = MAX(Var2, Var3) (Signed W) Var1 = MAX(Var2, Var3) (Signed DW)	Var 1	Maximum value	W : Word DW : Double Word	
	Var 2	Variable 1		
	Var 3	Variable 2		
	Expression Explanation			
	Get the maximum value between Var2 and Var3 and store the result in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎
Var 3	◎		◎

Example

- Var 1 is an internal memory addresses, and Var 2 and Var 3 are constants.



- Maximize (67, 34) and put the maximum value to \$50. Therefore \$50 will display 67.

■ MIN (Get Minimum Value)

Expression	What Variables Represent		NOTE	
Var1 = MIN(Var2, Var3) (W) Var1 = MIN(Var2, Var3) (DW) Var1 = MIN(Var2, Var3) (Signed W) Var1 = MIN(Var2, Var3) (Signed DW)	Var 1	Minimum Value	W : Word DW : Double Word Get the minimum value between Var2 and Var3 and store the result in Var1.	
	Var 2	Variable1		
	Var 3	Variable2		
	Expression Explanation			
	Get the minimum value between Var2 and Var3 and store the result in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎
Var 3	◎		◎

Example

- Var 1 is an internal memory address and Var 2 and Var 3 are both constants.

Macro Command

Edit	Command	?	X												
↑ ↓ ↶ ↷ ✚ ✖ #															
\$50 = MIN(67, 34)		<input type="checkbox"/> Double Word	<input type="checkbox"/> Signed												
Command MIN															
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Variables</th> <th style="width: 35%;">Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$50</td> <td>Minimum</td> </tr> <tr> <td>Var2</td> <td>67</td> <td>Variable1</td> </tr> <tr> <td>Var3</td> <td>34</td> <td>Variable2</td> </tr> </tbody> </table>				Variables	Contents	Description	Var1	\$50	Minimum	Var2	67	Variable1	Var3	34	Variable2
Variables	Contents	Description													
Var1	\$50	Minimum													
Var2	67	Variable1													
Var3	34	Variable2													

- Minimize (67, 34) and put the minimum value to \$50. Therefore \$50 will display 34.

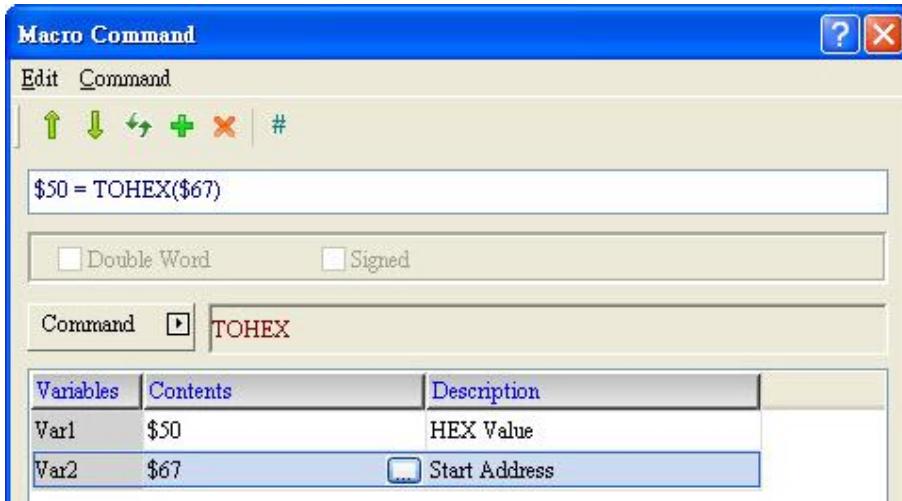
■ TOHEX (Conversion of 4 ASCII characters to four digit HEX integer)

Expression	What Variables Represent		NOTE
Var1 = TOHEX(Var2) (W)	Var 1	Hexadecimal value	W : Word
	Var 2	Starting ASCII Address	
	Expression Explanation		
Convert the ASCII characters of V2 (4 WORDS) to the HEX value and store the result in V1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		

Example

- Var 1 and Var 2 are all internal memory addresses.



- Convert the ASCII characters of \$67 (4 WORDS) to the HEX value and store the result in \$50.
- Both \$50 and \$67 are set to the Hex data type.
- If \$67 = 31H, \$68 = 32H, \$69 = 33H and \$70 = 34H, then after using the TOHEX command to convert the ASCII characters stored in these four variables into HEX data type and store them in \$50, then the result is: \$50 = 1244H.

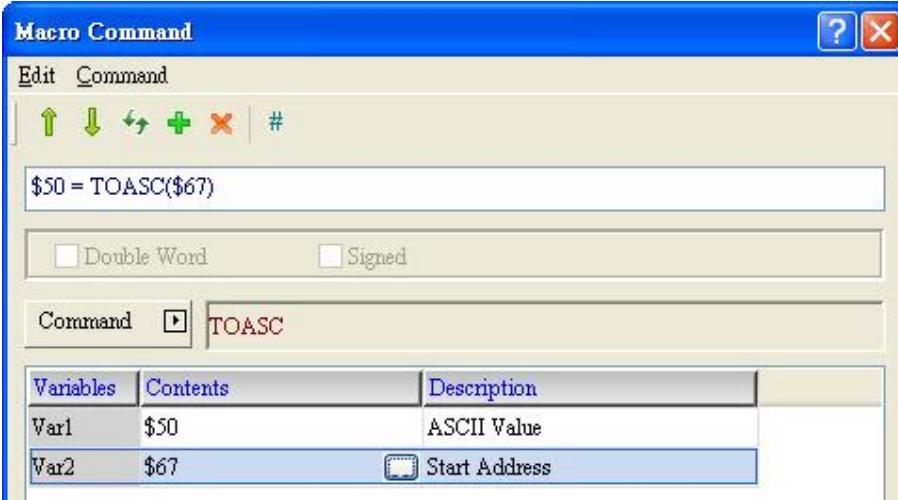
■ TOASC (Conversion of HEX integers into ASCII characters)

Expression	What Variables Represent		NOTE
Var1 = TOASC(Var2) (W)	Var 1	ASCII Value	W : Word
	Var 2	Starting hexadecimal address	
	Expression Explanation		
Convert Var2 in HEX format to the ASCII (4 WORDS) characters and store the result in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		

Example

- Var 1 and Var 2 are internal memory addresses



The screenshot shows the 'Macro Command' dialog box. In the 'Command' field, 'TOASC' is selected. Below it, 'Variables' is selected in the 'Command' dropdown. The 'Variables' table shows two entries: Var1 (\$50) with a description 'ASCII Value' and Var2 (\$67) with a description 'Start Address'. Above the table, there are checkboxes for 'Double Word' and 'Signed', both of which are unchecked.

- Convert the HEX values of \$67 into ASCII characters (4 WORDS) and store the result in \$50.
- Both \$50 and \$67 are set to the Hex data type.
- If \$67 = 1244H, then after using the TOASC command to convert the HEX value stored in \$67 into ASCII characters and stored in these four variables into \$50, \$51, \$52 and \$53, then the result is: \$50 = 31H, \$51 = 32H, \$52 = 33H, \$53 = 34H.

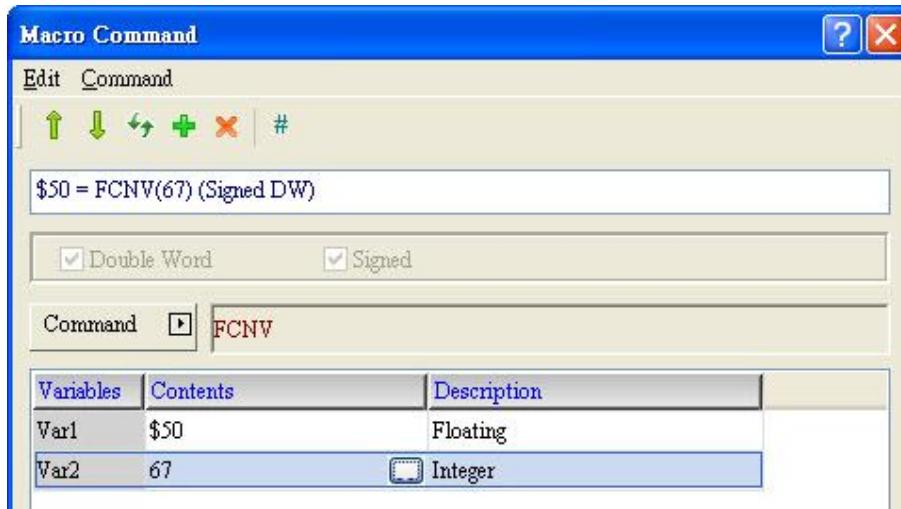
■ FCNV (Conversion of integer into floating point value)

Expression	What Variables Represent		NOTE
Var1 = FCNV(Var2) (Signed DW)	Var 1	Floating point value	DW : Double Word
	Var 2	Integer value	
Expression Explanation			Signed : Signed number
Convert the value in Var2 (integer) to floating point value and store in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 and Var 2 are internal memory addresses.



- Convert \$67 (integer value) into the floating point value and save it in \$50.
- \$50 is set to floating point data type and the integer to Double Word data type.
- The result is \$50 = 67.0.

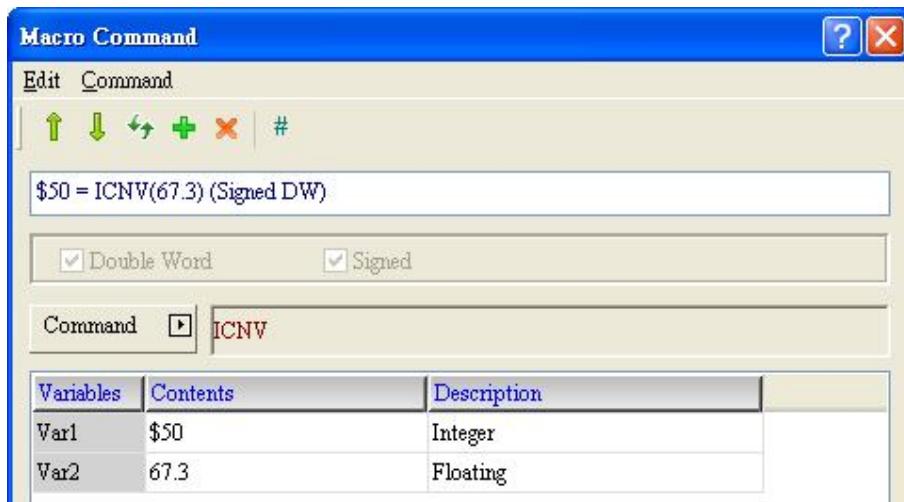
■ ICNV (Conversion from integer to floating point value)

Expression	What Variables Represent		NOTE
Var1 = ICNV(Var2) (Signed DW)	Var 1	Integer value	DW : Double Word Signed : Signed number
	Var 2	Floating point value	
	Expression Explanation		
Convert the value in Var2 (floating point value) into an integer and store it in Var1.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎

Example

- Var 1 and Var 2 are internal memory addresses.



- Convert 67.3 (floating point value) into the an integer and save it in \$50.
- \$50 is set to Unsigned Decimal and the integer to Word data type.
- The result is \$50 = 67.

■ SPRINTF (Format String)

Expression	What Variables Represent			NOTE			
Var1 = SPRINTF (Var2, "%u", Var4) (DW) Var1 = SPRINTF (Var2, "%u", Var4, ..., Var24) (DW) (Note 2)	Var 1	Returned Value		DW = Double Word			
		Failure	0				
		Success	1				
	Var 2	Fill in the target address of string					
	Var 3	String Content (Note 1)					
	Var 4	Value 1					
					
	Var 24	Value 20					
	Expression Explanation						
Fill in the string content then the target address according to Var 2							

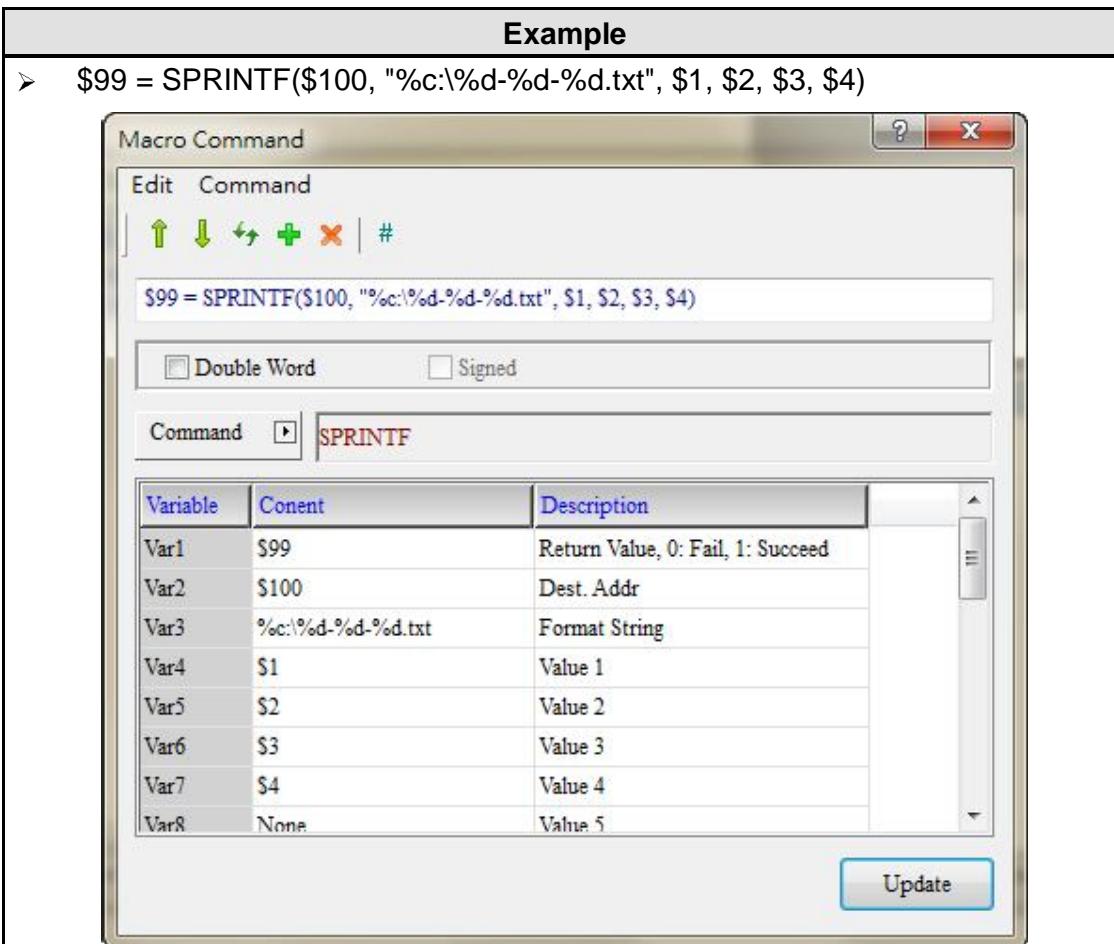
Note 1: The supported string formats are shown as below:

- %d : signed integer
- %u : unsigned integer
- %c : ASCII character
- %x : value in hexadecimal

Note 2: It supports up to 20 variables

Note 3: The fill in number should be identical with the value number.

Variable	Memory Usage			
	Internal Memory	PLC Register	String	Constant
Var 1	◎			
Var 2	◎	◎		
Var 3			◎	
Var 4	◎	◎		◎
...
Var 24	◎	◎		◎



Create numeric entry element: \$1, \$2, \$3, \$4. Then, enter the value which shown as below:

\$1 = 68 (represents D in ASCII code)

\$2 = 2012

\$3 = 12

\$4 = 21

Create character entry element and set its address to \$100. After execution, the screen will show as below:



24-3-5 Comparison

Comparison is performed with such commands as If...Then Goto, If...Then Call, If and Elseif and they are detailed below.

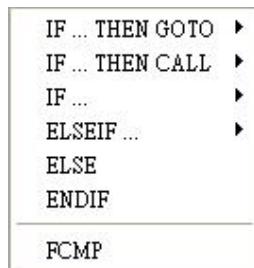


Figure 24-3-5-1 Comparison

NOTE :

- ✓ IF...structures only support 7 levels.

The screenshot shows a software window titled "Screen_12 [Screen Cycle Macro]". The code editor displays the following macro code:

```
1 IF $10 == 10
2 IF $20 == 20
3 IF $30 == 30
4 IF $40 == 40
5 IF $50 == 50
6 IF $60 == 60
7 IF $70 == 70
8 ENDIF
9 ENDIF
10 ENDIF
11 ENDIF
12 ENDIF
13 ENDIF
14 ENDIF
```

A red annotation on the right side of the code states: "Only support the seventh level for IF structure".

- IF...THEN GOTO (If Goto a certain label identifier and continue subsequent executions)

IF =	IF AND = 0
IF !=	IF AND != 0
IF >	IF = ON
IF >=	IF = OFF
IF <	IFB = ON
IF <=	IFB = OFF

- There are twelve commands in the category of If...Then Goto Macro, and they are introduced below.

(1) IF ==		
Expression	What Variables Represent	NOTE
IF Var1 == Var2 THEN GOTO LABEL Var3 (W) IF Var1 == Var2 THEN GOTO LABEL Var3 (DW) IF Var1 == Var2 THEN GOTO LABEL Var3 (Signed W) IF Var1 == Var2 THEN GOTO LABEL Var3 (Signed DW)	Var 1	condition1
	Var 2	condition2
	Var 3	Label identifier
	Expression Explanation	
	If condition1 equals condition2 then GOTO and execute Label Var 3.	W : Word DW : Double Word Signed : Signed number

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

Macro Command

Edit Command

| ↑ ↓ ← → + X | #

IF \$50 == \$67 THEN GOTO LABEL 1

Double Word Signed

Command : IF ==

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Label Name

Screen_12 [Screen Cycle Macro]

```

1 IF $50 == $67 THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200
    
```

- If \$50 equals \$67 (value comparison), then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(2) IF !=		
Expression	What Variables Represent	NOTE
IF Var1 != Var2 THEN GOTO LABEL Var3 (W)	Var 1 condition1	
IF Var1 != Var2 THEN GOTO LABEL Var3 (DW)	Var 2 condition2	
IF Var1 != Var2 THEN GOTO LABEL Var3 (Signed W)	Var 3 Label identifier	
Expression Explanation		
IF Var1 != Var2 THEN GOTO LABEL Var3 (Signed DW)	If condition1 does not equal to condition2 then GOTO and execute Label Var 3.	W : Word DW : Double Word Signed : Signed number

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



```
Screen_12 [Screen Cycle Macro]
1 IF $50 != $67 THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200
```

The screenshot shows the 'Screen_12 [Screen Cycle Macro]' window. It displays the following macro code:

```

Screen_12 [Screen Cycle Macro]
1 IF $50 != $67 THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200

```

- If \$50 does not equal to \$67 (value comparison), then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(3) IF >			
Expression	What Variables Represent		NOTE
IF Var1 > Var2 THEN GOTO LABEL Var3 (W) IF Var1 > Var2 THEN GOTO LABEL Var3 (DW) IF Var1 > Var2 THEN GOTO LABEL Var3 (Signed W) IF Var1 > Var2 THEN GOTO LABEL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number
	Var 2	condition2	
	Var 3	Label identifier	
	Expression Explanation		
	If condition1 is larger than condition2 then GOTO and execute Label Var 3.		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example													
➤	Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.												
 <p>The dialog box shows the following details:</p> <ul style="list-style-type: none"> Macro Command window title. Toolbar with icons for up, down, left, right, plus, minus, and hash. Text input field: IF \$50 > \$67 THEN GOTO LABEL 1 Checkboxes: Double Word (unchecked), Signed (unchecked). Command dropdown: IF > (selected). Variables table: <table border="1"> <thead> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$50</td> <td>Condition1</td> </tr> <tr> <td>Var2</td> <td>\$67</td> <td>Condition2</td> </tr> <tr> <td>Var3</td> <td>1</td> <td>Label Name</td> </tr> </tbody> </table> 		Variables	Contents	Description	Var1	\$50	Condition1	Var2	\$67	Condition2	Var3	1	Label Name
Variables	Contents	Description											
Var1	\$50	Condition1											
Var2	\$67	Condition2											
Var3	1	Label Name											

➤ If \$50 is larger than \$67 (value comparison), then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(4) IF >=		
Expression	What Variables Represent	NOTE
IF Var1 >= Var2 THEN GOTO LABEL Var3 (W)	Var 1 condition1	
IF Var1 >= Var2 THEN GOTO LABEL Var3 (DW)	Var 2 condition2	
IF Var1 >= Var2 THEN GOTO LABEL Var3 (Signed W)	Var 3 Label identifier	
Expression Explanation		
IF Var1 >= Var2 THEN GOTO LABEL Var3 (Signed DW)	If condition1 is larger than or equals to condition2 then GOTO and execute Label Var 3.	W : Word DW : Double Word Signed : Signed number

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

Macro Command

Edit Command

| + X | # |

IF \$50 >= \$67 THEN GOTO LABEL 1

Double Word Signed

Command IF >=

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Label Name

Screen_12 [Screen Cycle Macro]

```
1 IF $50 >= $67 THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200
```

- If \$50 is larger than or equals to \$67 (value comparison), then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(5) IF <		
Expression	What Variables Represent	NOTE
IF Var1 < Var2 THEN GOTO LABEL Var3 (W)	Var 1 condition1	W : Word DW : Double Word Signed : Signed number
IF Var1 < Var2 THEN GOTO LABEL Var3 (DW)	Var 2 condition2	
IF Var1 < Var2 THEN GOTO LABEL Var3 (Signed W)	Var 3 Label identifier	
IF Var1 < Var2 THEN GOTO LABEL Var3 (Signed DW)	Expression Explanation	
	If condition1 is smaller than condition2 then GOTO and execute Label Var 3.	

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



Screen_12 [Screen Cycle Macro]

```

1 IF $50 < $67 THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200

```

- If \$50 is smaller than \$67 (value comparison), then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(6) IF <=				
Expression	What Variables Represent		NOTE	
IF Var1 <= Var2 THEN GOTO LABEL Var3 (W) IF Var1 <= Var2 THEN GOTO LABEL Var3 (DW) IF Var1 <= Var2 THEN GOTO LABEL Var3 (Signed W) IF Var1 <= Var2 THEN GOTO LABEL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
	Var 2	condition2		
	Var 3	Label identifier		
	Expression Explanation			
	If condition1 is smaller than or equals to condition2 then GOTO and execute Label Var 3.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

Macro Command

Edit	Command	? X												
↑ ↓ ← → + × #														
IF \$50 <= \$67 THEN GOTO LABEL 1														
<input type="checkbox"/> Double Word <input type="checkbox"/> Signed														
Command IF <=														
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;">Variables</th> <th style="width: 40%;">Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$50</td> <td>Condition1</td> </tr> <tr> <td>Var2</td> <td>\$67</td> <td>Condition2</td> </tr> <tr> <td>Var3</td> <td>1</td> <td>Label Name</td> </tr> </tbody> </table>			Variables	Contents	Description	Var1	\$50	Condition1	Var2	\$67	Condition2	Var3	1	Label Name
Variables	Contents	Description												
Var1	\$50	Condition1												
Var2	\$67	Condition2												
Var3	1	Label Name												

➤ If \$50 is smaller or equals to \$67 (value comparison), then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(7) IF AND == 0

Expression	What Variables Represent		NOTE
IF (Var1 && Var2) == 0 THEN GOTO LABEL Var3 (W) IF (Var1 && Var2) == 0 THEN GOTO LABEL Var3 (DW)	Var 1	condition1	W : Word DW : Double Word
	Var 2	condition2	
	Var 3	Label identifier	
	Expression Explanation		
If the result of Bitwise AND Operation between condition1 equals condition2 is 0, then GOTO and execute Label Var 3.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example
<p>➤ Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.</p>

Macro Command

IF (\$50 && \$67) == 0 THEN GOTO LABEL 1

Double Word Signed

Command: IF AND == 0

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Label Name

Screen_12 [Screen Cycle Macro]

```

1 IF ($50 && $67) == 0 THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200
    
```

- If the result of Bitwise AND operation between \$50 and \$67 is 0, then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(8) IF AND != 0			
Expression	What Variables Represent		NOTE
IF (Var1 && Var2) != 0 THEN GOTO LABEL Var3 (W)	Var 1	condition1	
IF (Var1 && Var2) != 0 THEN GOTO LABEL Var3 (DW)	Var 2	condition2	
	Var 3	Label identifier	
Expression Explanation			
		If the result of Bitwise AND Operation between condition1 equals condition2 does not equal to 0, then GOTO and execute Label Var 3.	
		W : Word DW : Double Word	

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

- If the result of Bitwise AND operation between \$50 and \$67 does not equal to 0, then execute LABEL1 ($\$100 = 200$); otherwise, execute $\$100 = \$100 + 1$.

(9) IF == ON		
Expression	What Variables Represent	NOTE
IF Var1 == ON THEN GOTO LABEL Var2 (W)	Var 1	condition1
	Var 2	Label identifier
	Expression Explanation	W : Word
	If condition 1 is ON, then GOTO and execute LABEL Var 2.	

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		
Var 2			◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.



The Macro Command dialog box shows the following configuration:

- Command: IF == ON
- Variables table:

Variables	Contents	Description
Var1	\$50.0	Condition1
Var2	1	Label Name



The Screen_12 [Screen Cycle Macro] window displays the following macro code:

```

1 IF $50.0 == ON THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200
    
```

- If \$50.0 is ON, then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(10) IF == OFF			
Expression	What Variables Represent		NOTE
IF Var1 == OFF THEN GOTO LABEL Var2 (W)	Var 1	Condition1	
	Var 2	Label identifier	
Expression Explanation		W : Word	
		If condition 1 is OFF, then GOTO and execute LABEL Var 2.	

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		
Var 2			◎

Example

- Var 1 is an internal memory address, and Var 2 is a constant.

Macro Command

Edit Command

| ↑ ↓ ← → + X | #

IF \$50.0 == OFF THEN GOTO LABEL 1

Double Word Signed

Command : IF == OFF

Variables	Contents	Description
Var1	\$50.0	Condition1
Var2	1	Label Name

Screen_12 [Screen Cycle Macro]

File Edit View Tools Help (A) = Screen_12 [Screen Cycle Macro]

```

1 IF $50.0 == OFF THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200

```

- If \$50.0 is OFF, then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(11) IFB == ON		
Expression	What Variables Represent	NOTE
IFB Var1 == ON THEN GOTO LABEL Var2 (W)	Var 1	W : Word
	Var 2	
	Expression Explanation	
	If condition 1 is ON, then GOTO and execute LABEL Var 2.	

* in the command IFB == ON, the Bit address of Var 1 can support the external PLC Register.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)	◎ (Can only be Bit)	
Var 2			◎

Example

➤ Var 1 is a PLC Register address, and Var 2 is a constant.

Macro Command

Edit Command

| + - # |

IFB {Link2}1@M0 == ON THEN GOTO LABEL 1

Double Word Signed

Command IFB == ON

Variables	Contents	Description
Var1	{Link2}1@M0	Condition1
Var2	1	Label Name

Screen_12 [Screen Cycle Macro]

File Edit View Insert Tools Options Help Screen_12 [Screen Cycle Macro]

```

1 IFB {Link2}1@M0 == ON THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200

```

➤ If M0 is ON, then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(12) IFB == OFF		
Expression	What Variables Represent	NOTE
IFB Var1 == OFF THEN GOTO LABEL Var2 (W)	Var 1 Condition1 Var 2 Label identifier	
	Expression Explanation	
	If condition 1 is OFF, then GOTO and execute LABEL Var 2.	W : Word
* in the command IFB == ON, the Bit address of Var 1 can support the external PLC Register.		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)	◎ (Can only be Bit)	
Var 2			◎

Example

- Var 1 is a PLC Register address, and Var 2 is a constant.

Macro Command

Edit Command

IFB {Link2}1@M0 == OFF THEN GOTO LABEL 1

Double Word Signed

Command: IFB == OFF

Variables	Contents	Description
Var1	{Link2}1@M0	Condition1
Var2	1	Label Name

Screen_12 [Screen Cycle Macro]

```

1 IFB {Link2}1@M0 == OFF THEN GOTO LABEL 1
2 $100 = $100 + 1
3 END
4 LABEL 1
5 $100 = 200
    
```

- If M0 is OFF, then execute LABEL1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

■ IF...THEN CALL (If... Then Call a Submacro)

IF = CALL
IF != CALL
IF > CALL
IF >= CALL
IF < CALL
IF <= CALL
IF AND = 0 CALL
IF AND != 0 CALL
IF = ON CALL
IF = OFF CALL

- There are 10 commands in the category of If...Then Call Macro, and they are introduced below.

(1) IF ==				
Expression	What Variables Represent		NOTE	
IF Var1 == Var2 THEN CALL Var3 (W) IF Var1 == Var2 THEN CALL Var3 (DW) IF Var1 == Var2 THEN CALL Var3 (Signed W) IF Var1 == Var2 THEN CALL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
	Var 2	condition2		
	Var 3	Label identifier		
	Expression Explanation			
	If condition 1 equals to condition 2, then call Var 3 (a submacro label).			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.

The figure consists of three vertically stacked windows from a software application:

- Macro Command Window:** Shows a command line with "IF \$50 == \$67 THEN CALL 1". Below it are checkboxes for "Double Word" and "Signed". A table below lists variables: Var1 (\$50), Var2 (\$67), and Var3 (1). The "Command" dropdown shows "IF == CALL".
- Sub-Macro Window:** Shows a table with one entry: No 1, Alias Sub-macro (1), Comment Sub-macro (1). Below is a code editor with two lines: 1 \$100 = 200 and 2 END.
- Screen_12 [Screen Cycle Macro] Window:** Shows a code editor with two lines: 1 IF \$50 == \$67 THEN CALL 1 and 2 \$100 = \$100 + 1.

- If \$50.0 equals to \$67 (value comparison), then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(2) IF !=			
Expression	What Variables Represent		NOTE
IF Var1 != Var2 THEN CALL Var3 (W) IF Var1 != Var2 THEN CALL Var3 (DW) IF Var1 != Var2 THEN CALL Var3 (Signed W) IF Var1 != Var2 THEN CALL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number
	Var 2	condition2	
	Var 3	Label identifier	
	Expression Explanation		
If condition 1 does not equal to condition 2, then call Var 3 (a submacro label).			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

➤ Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.

Macro Command

Edit Command

IF \$50 != \$67 THEN CALL 1

Double Word Signed

Command: **IF != CALL**

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Call Sub-Macro

Sub-Macro

No	Alias	Comment
1		Sub-macro (1)

Sub-macro (1) [Submacro 1]

```

1 $100 = 200
2 END
  
```

➤ If \$50.0 does not equal to \$67 (value comparison), then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(3) IF >

Expression	What Variables Represent		NOTE	
IF Var1 > Var2 THEN CALL Var3 (W) IF Var1 > Var2 THEN CALL Var3 (DW) IF Var1 > Var2 THEN CALL Var3 (Signed W) IF Var1 > Var2 THEN CALL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
	Var 2	condition2		
	Var 3	Label identifier		
	Expression Explanation			
	If condition 1 is larger than condition 2, then call Var 3 (a submacro label).			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.



➤ If \$50.0 is larger than \$67 (value comparison), then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(4) IF >=				
Expression	What Variables Represent		NOTE	
IF Var1 >= Var2 THEN CALL Var3 (W) IF Var1 >= Var2 THEN CALL Var3 (DW) IF Var1 >= Var2 THEN CALL Var3 (Signed W) IF Var1 >= Var2 THEN CALL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
	Var 2	condition2		
	Var 3	Label identifier		
	Expression Explanation			
	If condition is larger or equals to condition 2, then call Var 3 (a submacro label).			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.

The screenshot shows three windows related to macro definitions:

- Macro Command** window:
 - Text input: IF \$50 >= \$67 THEN CALL 1
 - Checkboxes: Double Word, Signed
 - Command dropdown: IF >= CALL
 - Table:

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Call Sub-Macro
- Sub-Macro** window:
 - Table:

No	Alias	Comment
1		Sub-macro (1)
 - Code area:


```
Sub-macro (1) [Submacro 1]
1 $100 = 200
2 END
```
- Screen_12 [Screen Cycle Macro]** window:
 - Code area:

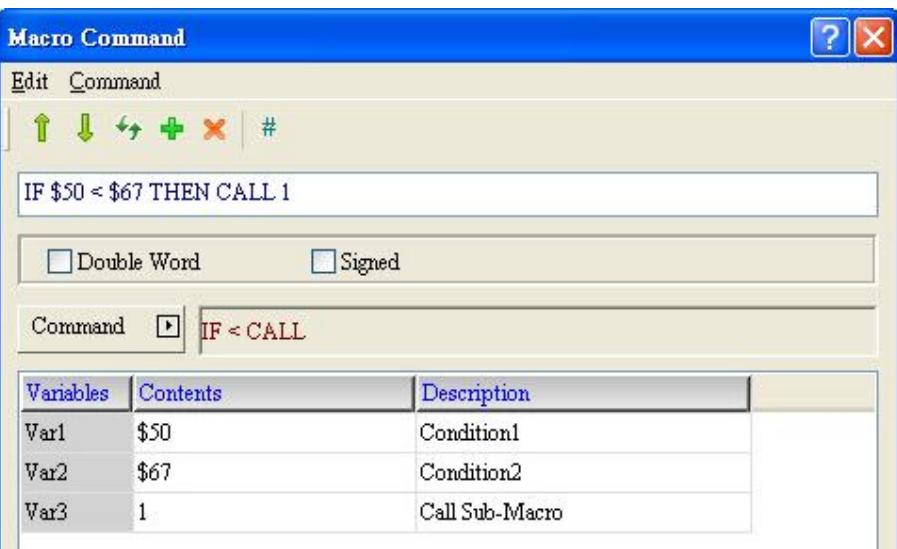

```
IF $50 >= $67 THEN CALL 1
$100 = $100 + 1
```

- If \$50 is larger than or equals to \$67 (value comparison), then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(5) IF <				
Expression		What Variables Represent	NOTE	
IF Var1 < Var2 THEN CALL Var3 (W) IF Var1 < Var2 THEN CALL Var3 (DW) IF Var1 < Var2 THEN CALL Var3 (Signed W) IF Var1 < Var2 THEN CALL Var3 (Signed DW)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
	Var 2	condition2		
	Var 3	Label identifier		
	Expression Explanation			
	If condition 1 is smaller than condition 2, then call Var 3 (a submacro label).			
Variable	Memory Usage			
	Internal Memory	PLC Register	Constant	
Var 1	◎		◎	
Var 2	◎		◎	
Var 3			◎	

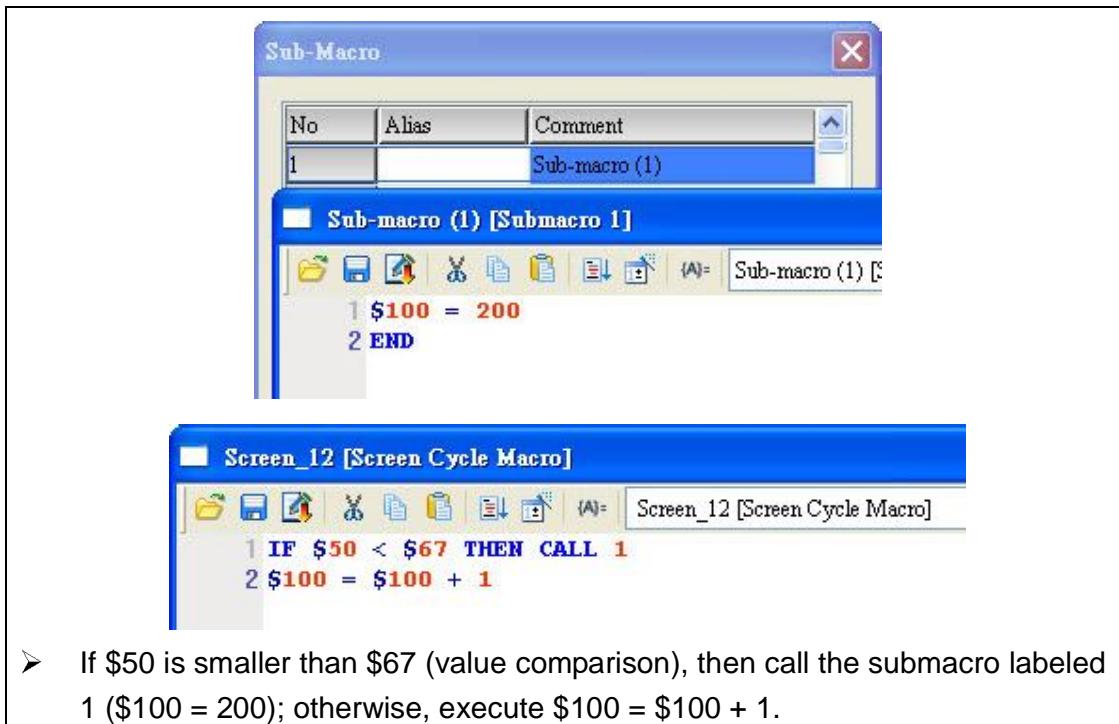
Example

- Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.



The screenshot shows the 'Macro Command' dialog box. The command entered is 'IF \$50 < \$67 THEN CALL 1'. Under 'Command', 'IF < CALL' is selected. In the 'Variables' table:

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Call Sub-Macro



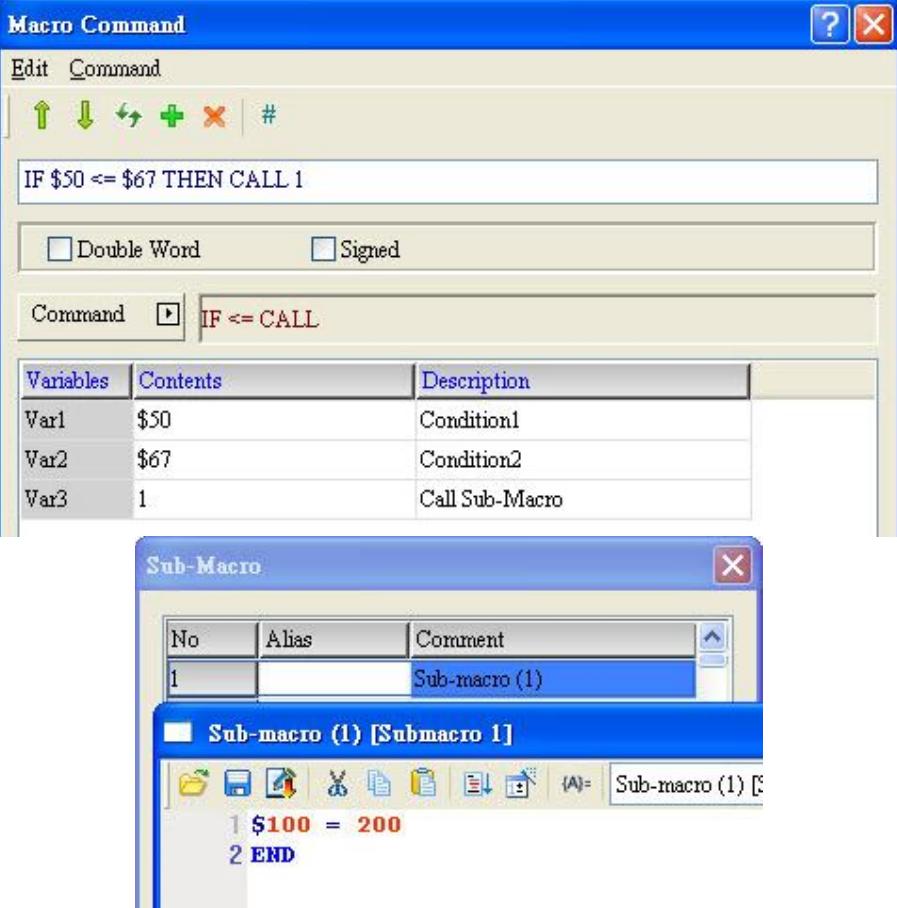
- If \$50 is smaller than \$67 (value comparison), then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(6) IF <=			
Expression	What Variables Represent		NOTE
IF Var1 <= Var2 THEN CALL Var3 (W)	Var 1	condition1	
IF Var1 <= Var2 THEN CALL Var3 (DW)	Var 2	condition2	
IF Var1 <= Var2 THEN CALL Var3 (Signed W)	Var 3	Label identifier	
IF Var1 <= Var2 THEN CALL Var3 (Signed DW)	Expression Explanation		
	If condition 1 is smaller than or equals to condition 2, then call Var 3 (a submacro label).		W : Word DW : Double Word Signed : Signed number

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

➤ Var 1 and Var 2 are internal memory addresses, and Var 3 is a constant.



The Macro Command dialog shows the command:

```
IF $50 <= $67 THEN CALL 1
```

With options for Double Word and Signed. The Command field is set to IF <= CALL.

The Sub-Macro dialog shows a table with one entry:

No	Alias	Comment
1		Sub-macro (1)

The macro body contains:

```
1 $100 = 200
2 END
```

Screen_12 [Screen Cycle Macro]

1 IF \$50 <= \$67 THEN CALL 1
2 \$100 = \$100 + 1

➤ If \$50 is smaller than or equals to \$67 (value comparison), then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(7) IF AND == 0

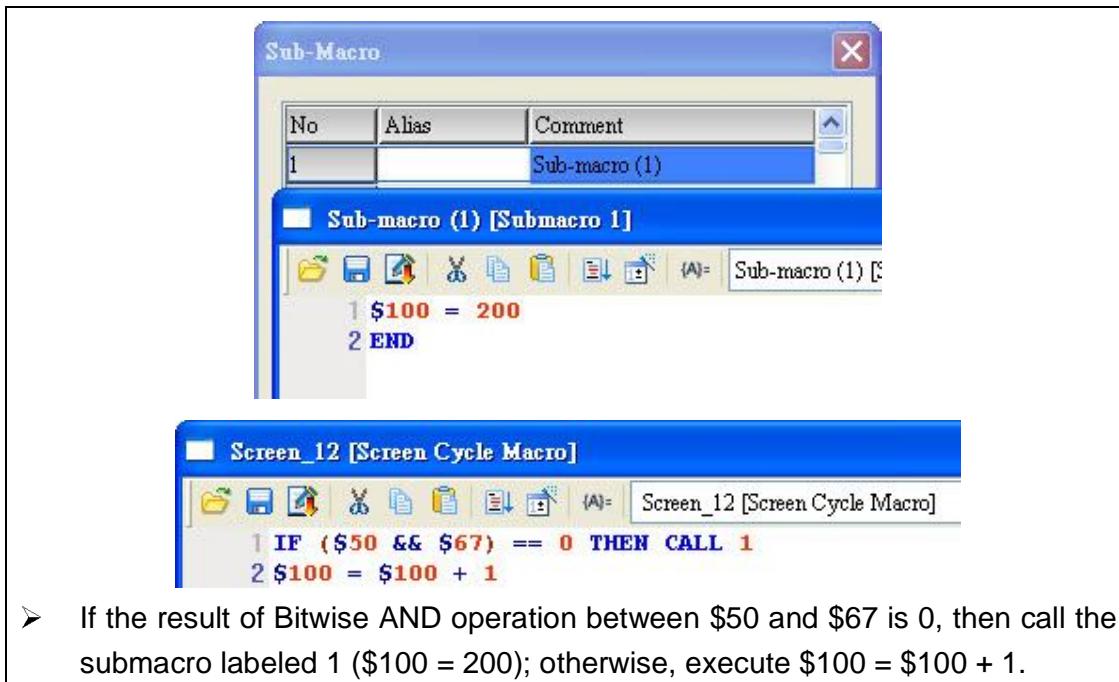
Expression	What Variables Represent		NOTE
IF (Var1 && Var2) == 0 THEN CALL Var3 (W) IF (Var1 && Var2) == 0 THEN CALL Var3 (DW)	Var 1	condition1	W : Word DW : Double Word Word
	Var 2	condition2	
	Var 3	Label identifier	
	Expression Explanation		
	If the result of Bitwise AND Operation between condition1 and condition2 is 0, then call the submacro labeled with Var 3.		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.





(8) IF AND != 0				
Expression	What Variables Represent		NOTE	
IF (Var1 && Var2) != 0 THEN CALL Var3 (W) IF (Var1 && Var2) != 0 THEN CALL Var3 (DW)	Var 1	condition1	W : Word DW : Double Word	
	Var 2	condition2		
	Var 3	Label identifier		
	Expression Explanation			
	If the result of Bitwise AND Operation between condition1 and condition2 is not 0, then call the submacro labeled with Var 3.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

Macro Command

IF (\$50 && \$67) != 0 THEN CALL 1

Double Word Signed

Command: IF AND != 0 CALL

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2
Var3	1	Call Sub-Macro

Sub-Macro

No Alias Comment

1	Sub-macro (1)
---	---------------

Sub-macro (1) [Submacro 1]

```
1 $100 = 200
2 END
```

Screen_12 [Screen Cycle Macro]

1 IF (\$50 && \$67) != 0 THEN CALL 1

2 \$100 = \$100 + 1

- If the result of Bitwise AND operation between \$50 and \$67 is not 0, then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

(9) IF == ON				
Expression	What Variables Represent		NOTE	
IF Var1 == ON THEN CALL Var2 (W)	Var 1	condition1	W : Word If condition1 is ON, then call the submacro labeled with Var 2.	
	Var 2	Label identifier		
	Expression Explanation			
	If condition1 is ON, then call the submacro labeled with Var 2.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		
Var 2			◎

Example

➤ Var 1 is the internal memory address, and Var 2 is a constant.



The Macro Command dialog box shows the following settings:

- Command: IF \$50.0 == ON THEN CALL 1
- Type: Double Word
- Condition: IF == ON
- Variables:

Variables	Contents	Description
Var1	\$50.0	Condition1
Var2	1	Call Sub-Macro



The Sub-Macro dialog box shows the following entry:

No	Alias	Comment
1		Sub-macro (1)

Below the table, it says "Sub-macro (1) [Submacro 1]" and shows the following code:

```

1 $100 = 200
2 END
  
```

Screen_12 [Screen Cycle Macro]

1 IF \$50.0 == ON THEN CALL 1
2 \$100 = \$100 + 1

➤ If \$50.0 is ON, then call the submacro labeled 1 (\$100 = 200); otherwise, execute \$100 = \$100 + 1.

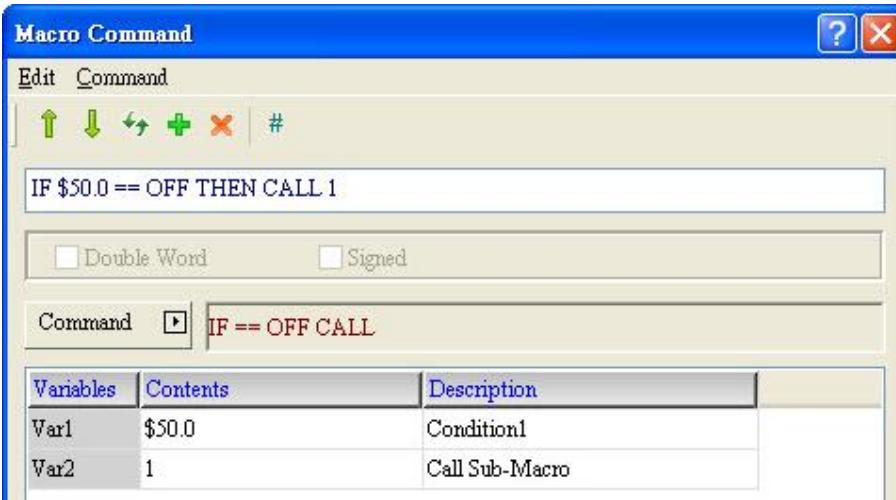
(10) IF == OFF

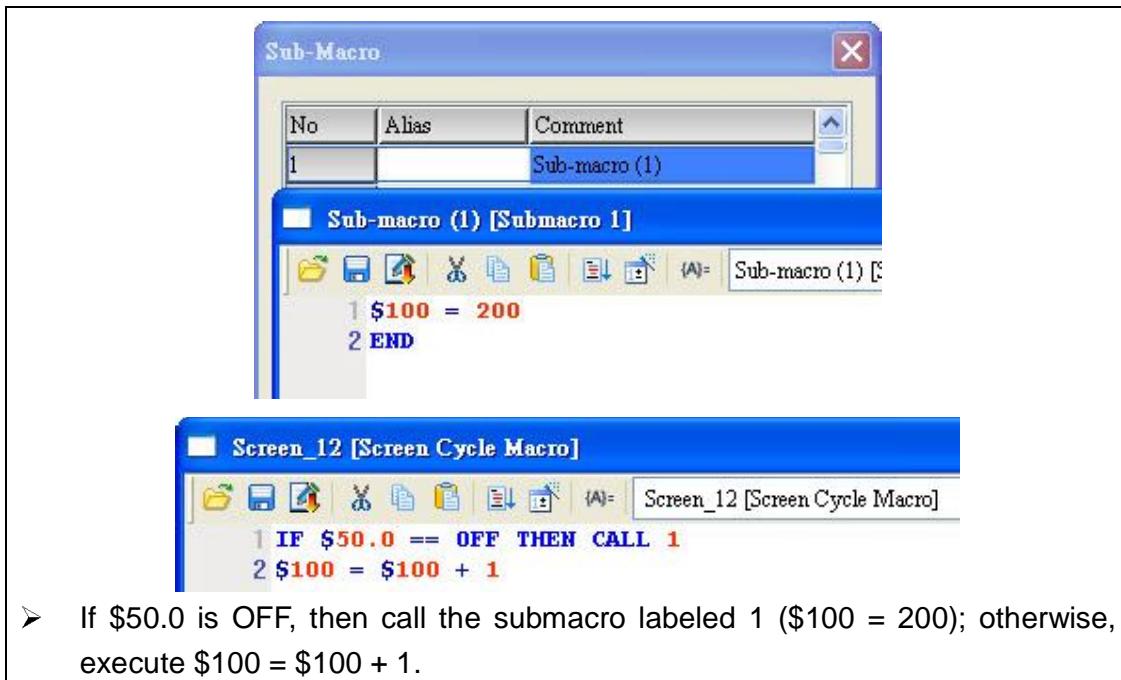
Expression	What Variables Represent		NOTE	
IF Var1 == OFF THEN CALL Var2 (W)	Var 1	condition1	W : Word	
	Var 2	Label identifier		
	Expression Explanation			
	If condition1 is OFF, then call the submacro labeled with Var 2.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be bit)		
Var 2			◎

Example

- Var 1 is the internal memory address, and Var 2 is a constant.





- If \$50.0 is OFF, then call the submacro labeled 1 (\$100 = 200); otherwise, execute $\$100 = \$100 + 1$.

■ IF...(If...)

IF =
IF !=
IF >
IF >=
IF <
IF <=
IF AND =0
IF AND !=0
IF =ON
IF =OFF

- There are 10 commands in the category of If... Macro, and they are introduced below.

(1) IF ==				
Expression	What Variables Represent		NOTE	
IF Var1 == Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
IF Var1 == Var2 (DW)	Var 2	condition2		
IF Var1 == Var2 (Signed W)	Expression Explanation			
IF Var1 == Var2 (Signed DW)	If condition1 equals to condition2, then execute...			

*ENDIF is required at end of the IF Macro, or error messages will pop up during compile.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example
<ul style="list-style-type: none"> ➤ Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

Macro Command

IF \$50 == \$67

Double Word Signed

Command : IF ==

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

Screen_12 [Screen Cycle Macro]

```

1 IF $50 == $67
2 $100 = $100 + 1
3 ENDIF

```

- If \$50 equals to \$67 (value comparison), then execute \$100 = \$100+1; if \$50 is larger or smaller than \$67, then do not execute \$100 = \$100 + 1.

(2) IF !=

Expression	What Variables Represent		NOTE
IF Var1 != Var2 (W)	Var 1	condition1	W : Word
IF Var1 != Var2 (DW)	Var 2	condition2	DW : Double
IF Var1 != Var2 (Signed W)	Expression Explanation		
IF Var1 != Var2 (Signed DW)	If condition1 does not equal to condition2, then execute...		

*ENDIF is required at end of the IF Macro, or error messages will pop up during compile.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.

Macro Command

Edit Command

IF \$50 != \$67

Double Word Signed

Command: IF !=

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

Screen_12 [Screen Cycle Macro]

```

1 IF $50 != $67
2 $100 = $100 + 1
3 ENDIF

```

- If \$50 does not equal to \$67 (value comparison), then execute \$100 = \$100+1; otherwise, do not execute \$100 = \$100 + 1.

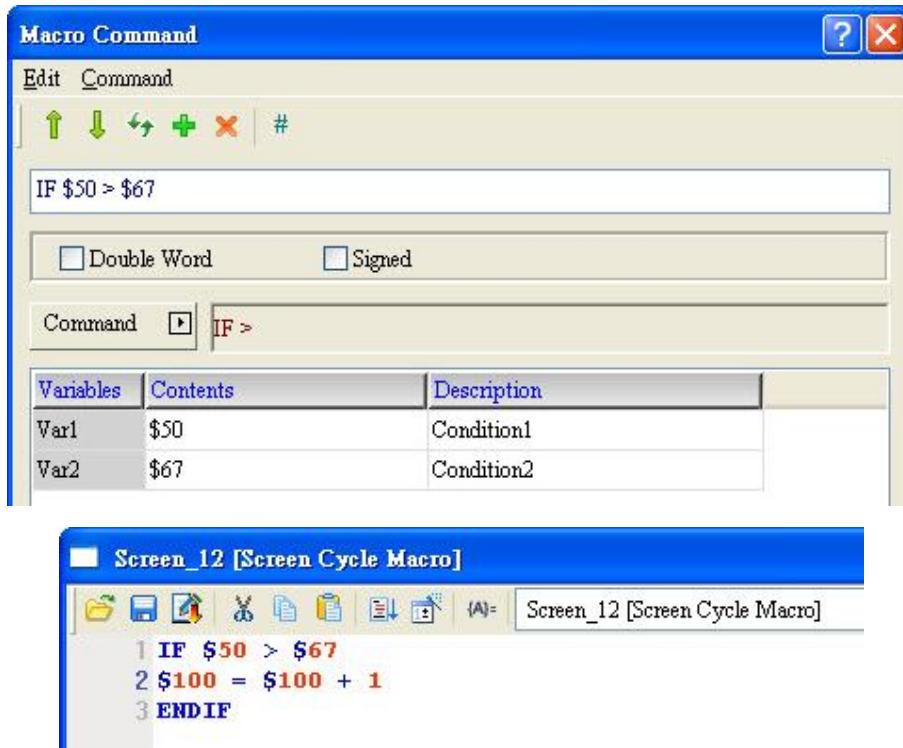
(3) IF >				
Expression	What Variables Represent		NOTE	
IF Var1 > Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
IF Var1 > Var2 (DW)	Var 2	condition2		
IF Var1 > Var2 (Signed W)	Expression Explanation			
IF Var1 > Var2 (Signed DW)	If condition1 is larger than condition2, then execute...			

*ENDIF is required at end of the IF Macro, or error messages will pop up during compile.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



```

Macro Command
Edit Command
IF $50 > $67
Double Word Signed
Command IF >
Variables Contents Description
Var1 $50 Condition1
Var2 $67 Condition2

Screen_12 [Screen Cycle Macro]
(A) Screen_12 [Screen Cycle Macro]
1 IF $50 > $67
2 $100 = $100 + 1
3 ENDIF

```

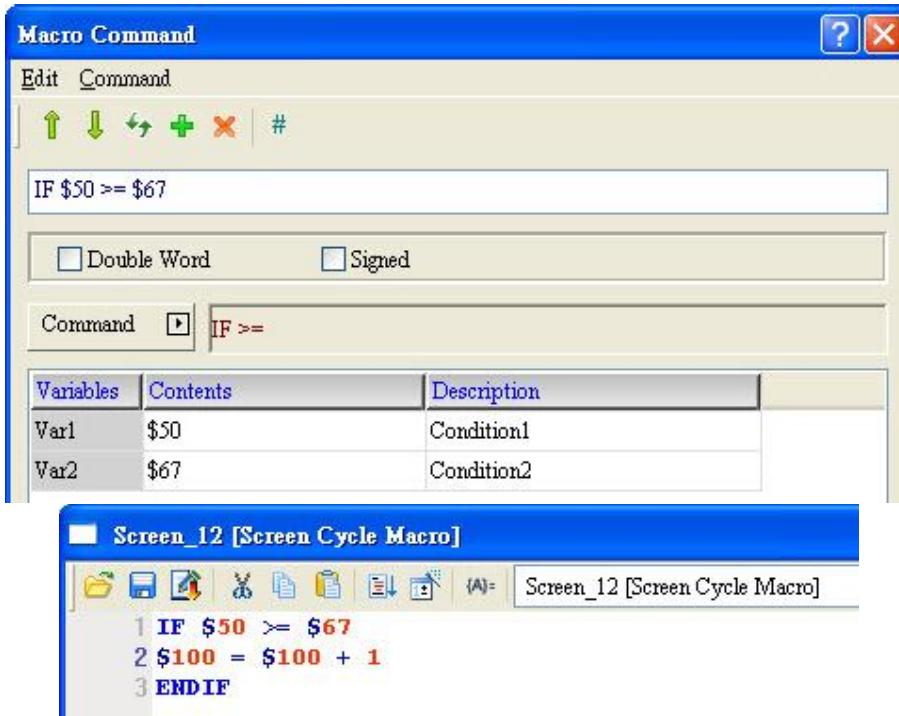
- If \$50 is larger than \$67 (value comparison), then execute $\$100 = \$100 + 1$; otherwise, do not execute $\$100 = \$100 + 1$.

(4) IF >=				
Expression	What Variables Represent		NOTE	
IF Var1 >= Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
IF Var1 >= Var2 (DW)	Var 2	condition2		
IF Var1 >= Var2 (Signed W)	Expression Explanation			
IF Var1 >= Var2 (Signed DW)	If condition1 is larger than or equals to condition2, then execute...			
*ENDIF is required at end of the IF Macro, or error messages will pop up during compile.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.



- If \$50 is larger than or equal to \$67 (value comparison), then execute $\$100 = \$100 + 1$; otherwise, do not execute $\$100 = \$100 + 1$.

(5) IF <

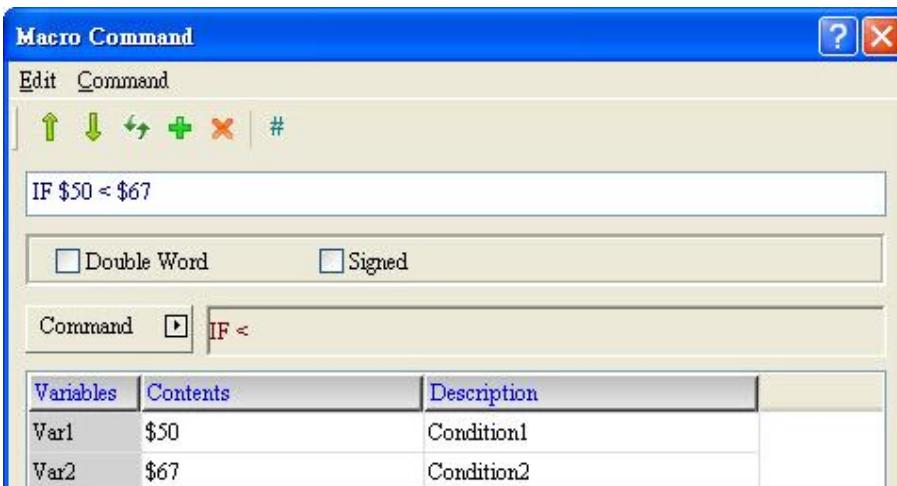
Expression	What Variables Represent		NOTE	
IF Var1 < Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
IF Var1 < Var2 (DW)	Var 2	condition2		
IF Var1 < Var2 (Signed W)	Expression Explanation			
IF Var1 < Var2 (Signed DW)	If condition1 is smaller than condition2, then execute...			

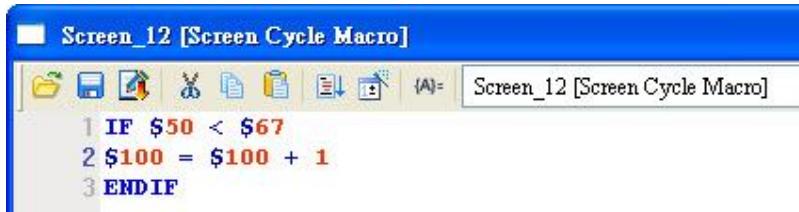
*ENDIF is required at end of the IF Macro, or error messages will pop up during compile.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.





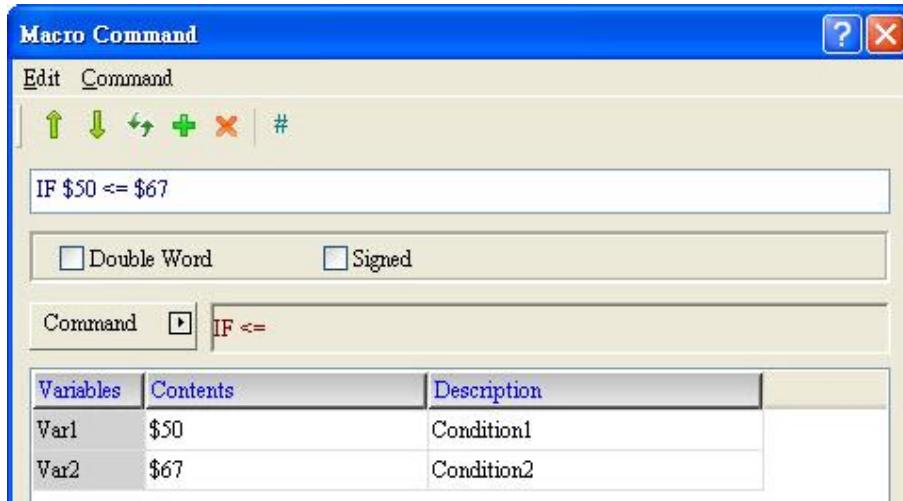
- If \$50 is smaller than \$67 (value comparison), then execute $\$100 = \$100 + 1$; otherwise, do not execute $\$100 = \$100 + 1$.

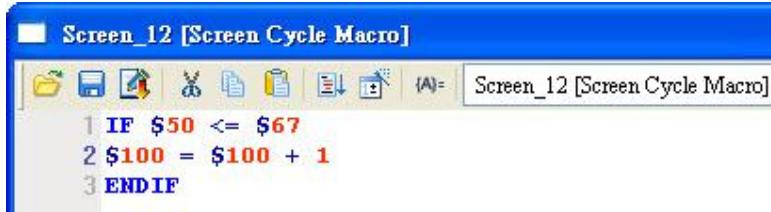
(6) IF <=				
Expression	What Variables Represent		NOTE	
IF Var1 <= Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
IF Var1 <= Var2 (DW)	Var 2	condition2		
IF Var1 <= Var2 (Signed W)	Expression Explanation			
IF Var1 <= Var2 (Signed DW)	If condition1 is smaller than or equals to condition2, then execute...			
*ENDIF is required at end of the IF Macro, or error messages will pop up during compile.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎
Var 3			◎

Example

- Var 1 and Var 2 are both internal memory addresses, and Var 3 is a constant.





- If \$50 is smaller than or equals to \$67 (value comparison), then execute \$100 = \$100+1; otherwise, do not execute \$100 = \$100 + 1.

(7) IF AND == 0				
Expression	What Variables Represent		NOTE	
IF (Var1 && Var2) == 0 (W) IF (Var1 && Var2) == 0 (DW)	Var 1	condition1	W : Word DW : Double Word	
	Var 2	condition2		
	Expression Explanation			
	If the result of Bitwise AND Operation between condition1 and condition2 is 0, then execute ...			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.

Macro Command

Edit Command

↑ ↓ ↺ ↻

#

```
IF ($50 && $67) == 0
```

Double Word Signed

Command
 IF AND == 0

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

Screen_12 [Screen Cycle Macro]

Screen_12 [Screen Cycle Macro]


```

1 IF ($50 && $67) == 0
2 $100 = $100 + 1
3 ENDIF
    
```

- If the result of Bitwise AND Operation between \$50 and \$67 is 0, then execute \$100 = \$100+1; otherwise, do not execute \$100 = \$100 + 1.

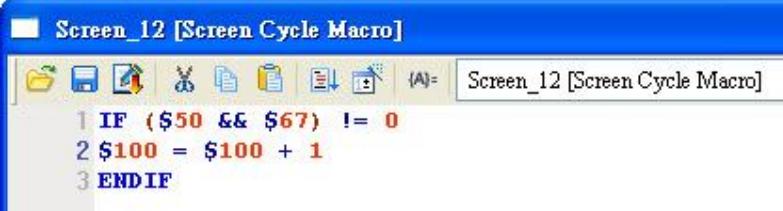
(8) IF AND != 0				
Expression	What Variables Represent		NOTE	
IF (Var1 && Var2) != 0 (W) IF (Var1 && Var2) != 0 (DW)	Var 1	condition1	W : Word DW : Double Word	
	Var 2	condition2		
	Expression Explanation			
	If the result of Bitwise AND Operation between condition1 and condition2 is not 0, then execute ...			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.





- If the result of Bitwise AND Operation between \$50 and \$67 is not 0, then execute $\$100 = \$100 + 1$; otherwise, do not execute $\$100 = \$100 + 1$.

(9) IF == ON				
Expression	What Variables Represent		NOTE	
IF Var1 == ON (W)	Var 1	condition1	W : Word	
	Expression Explanation			
	If condition1 is ON, then execute...			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		

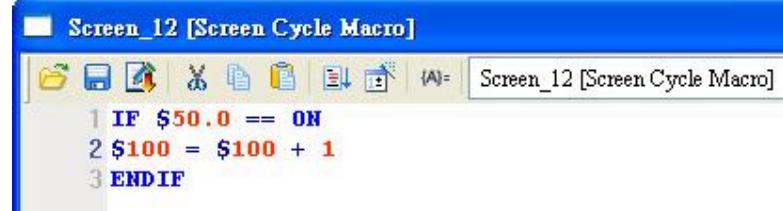
Example

- Var 1 is an internal memory address.



The Macro Command dialog box shows the following configuration:

- Command:** IF \$50.0 == ON
- Type:** Double Word (unchecked)
- Signed:** Signed (unchecked)
- Condition:** IF == ON
- Variables:** Var1 (\$50.0) is listed with Description: Condition.



The macro code in the Screen_12 [Screen Cycle Macro] window is:

```

1 IF $50.0 == ON
2 $100 = $100 + 1
3 ENDIF

```

- If \$50.0 is ON, then execute $\$100 = \$100 + 1$; otherwise, do not execute $\$100 = \$100 + 1$.

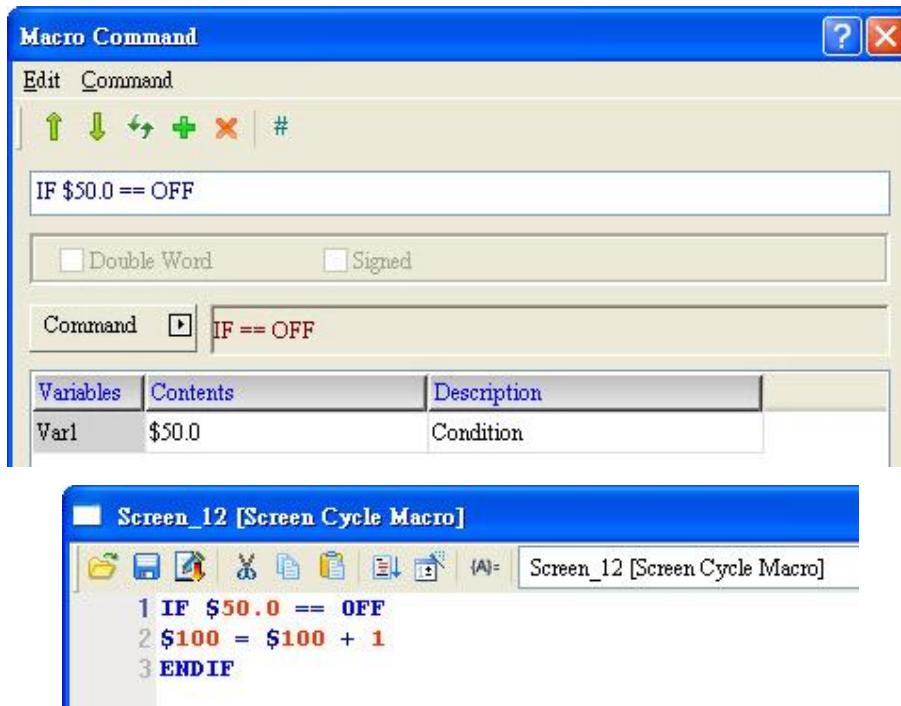
(10) IF == OFF

Expression	What Variables Represent		NOTE	
IF Var1 == OFF (W)	Var 1	condition1	W : Word	
	Expression Explanation			
	If condition1 is OFF, then execute...			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		

Example

- Var 1 is an internal memory address.



- If `$50.0` is OFF, then execute $\$100 = \$100 + 1$; otherwise, do not execute $\$100 = \$100 + 1$.

■ ELSEIF... (Else...)

ELSEIF ==
ELSEIF !=
ELSEIF >
ELSEIF >=
ELSEIF <
ELSEIF <=
ELSEIF AND == 0
ELSEIF AND != 0
ELSEIF = ON
ELSEIF = OFF

- There are 10 commands in the category of ELSEIF ...Macro, and they are introduced below:

(1) ELSEIF ==

Expression	What Variables Represent		NOTE	
ELSEIF Var1 == Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
ELSEIF Var1 == Var2 (DW)	Var 2	condition2		
ELSEIF Var1 == Var2 (Signed W)	Expression Explanation			
ELSEIF Var1 == Var2 (Signed DW)	Or else, if condition1 equals to condition2, then execute...			
*IF...ENDIF is required to pair up with ELSEIF Macro commands, or error messages will pop up during compile.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.

The Macro Command dialog box shows the following settings:

- Command:** ELSEIF \$50 == \$67
- Type:** Double Word (unchecked), Signed (unchecked)
- Variables:**

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

The Screen_12 [Screen Cycle Macro] window displays the following assembly-like code:

```

1 IF $50 != $67
2 $200 = $200 + 1
3 ELSEIF $50 == $67
4 $100 = $100 + 1
5 ENDIF

```

➤ If \$50 does not equal to \$67 (value comparison), then execute \$200 = \$200+1; otherwise, then execute \$100 = \$100 + 1.

(2) ELSEIF !=

Expression	What Variables Represent		NOTE
ELSEIF Var1 != Var2 (W)	Var 1	condition1	W : Word
ELSEIF Var1 != Var2 (DW)	Var 2	condition2	DW : Double
ELSEIF Var1 != Var2 (Signed W)	Expression Explanation		Word
ELSEIF Var1 != Var2 (Signed DW)	Or else, if condition1 does not equal to condition2, then execute...		Signed : Signed number

*IF...ENDIF is required to pair up with the ELSEIF Macro commands, or error messages will pop up during compile.

Memory Usage			
Variable	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.

Macro Command

Edit Command

ELSEIF \$50 != \$67

Double Word Signed

Command: ELSEIF !=

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

Screen_12 [Screen Cycle Macro]

```

1 IF $50 == $67
2 $200 = $200 + 1
3 ELSEIF $50 != $67
4 $100 = $100 + 1
5 ENDIF

```

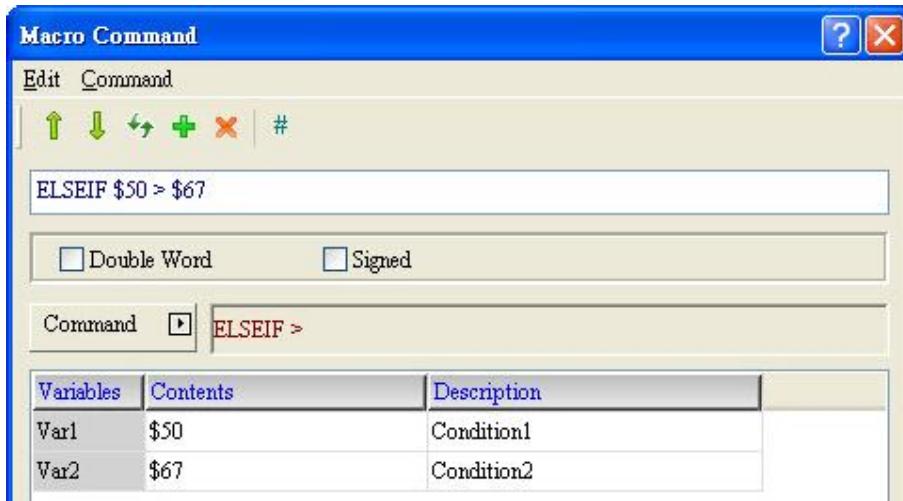
- If \$50 equals to \$67 (value comparison), then execute \$200 = \$200+1; otherwise, then execute \$100 = \$100 + 1.

(3) ELSEIF >				
Expression	What Variables Represent		NOTE	
ELSEIF Var1 > Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
ELSEIF Var1 > Var2 (DW)	Var 2	condition2		
ELSEIF Var1 > Var2 (Signed W)	Expression Explanation			
ELSEIF Var1 > Var2 (Signed DW)	Or else, if condition1 is larger than condition2, then execute...			
*IF...ENDIF is required to pair up with the ELSEIF Macro commands, or error messages will pop up during compile.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.



Screen_12 [Screen Cycle Macro]

```

1 IF $50 == $67
2 $200 = $200 + 1
3 ELSEIF $50 > $67
4 $100 = $100 + 1
5 ENDIF

```

- If \$50 equals to \$67 (value comparison), then execute \$200 = \$200+1; or else if \$50 is larger than \$67, then execute \$100 = \$100 + 1.

(4) ELSEIF >=

Expression	What Variables Represent		NOTE
ELSEIF Var1 >= Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number
ELSEIF Var1 >= Var2 (DW)	Var 2	condition2	
Expression Explanation			
Or else, if condition1 is larger than or equals to condition2, then execute...			
*IF...ENDIF is required to pair up with the ELSEIF Macro commands, or error messages will pop up during compile.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.

Macro Command

Edit Command

| | #

```
ELSEIF $50 >= $67
```

Double Word Signed

Command ELSEIF >=

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

Screen_12 [Screen Cycle Macro]

(A) = Screen_12 [Screen Cycle Macro]

```

1 IF $50 < $67
2 $200 = $200 + 1
3 ELSEIF $50 >= $67
4 $100 = $100 + 1
5 ENDIF

```

- If \$50 is smaller than \$67 (value comparison), then execute \$200 = \$200+1; otherwise, execute \$100 = \$100 + 1.

(5) ELSEIF <				
Expression	What Variables Represent		NOTE	
ELSEIF Var1 < Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
ELSEIF Var1 < Var2 (DW)	Var 2	condition2		
ELSEIF Var1 < Var2 (Signed W)	Expression Explanation			
ELSEIF Var1 < Var2 (Signed DW)	Or else, if condition1 is smaller than condition2, then execute...			

*IF...ENDIF is required to pair up with the ELSEIF Macro commands, or error messages will pop up during compile.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

```

1 IF $50 >= $67
2 $200 = $200 + 1
3 ELSEIF $50 < $67
4 $100 = $100 + 1
5 ENDIF
    
```

- If \$50 is larger than or equals to \$67 (value comparison), then execute \$200 = \$200+1; or else if \$50 is larger than \$67, then execute \$100 = \$100 + 1.

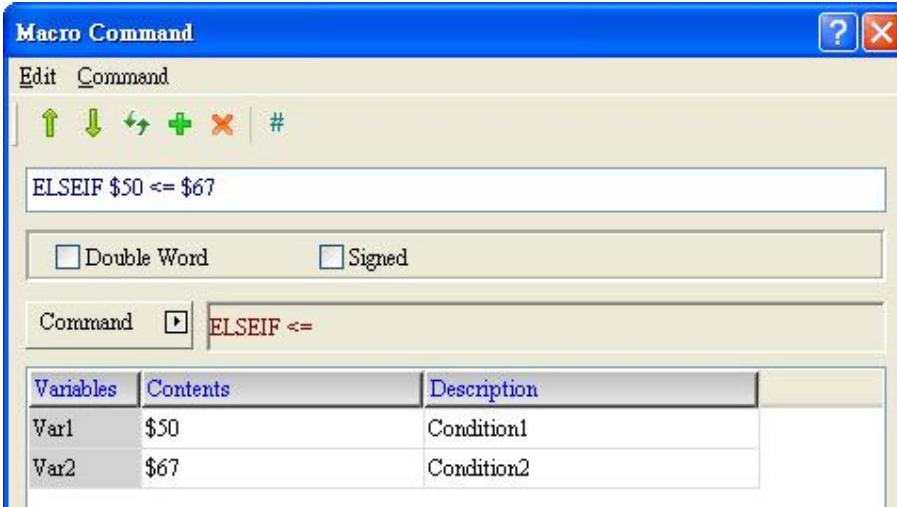
(6) ELSEIF <=				
Expression	What Variables Represent		NOTE	
ELSEIF Var1 <= Var2 (W)	Var 1	condition1	W : Word DW : Double Word Signed : Signed number	
ELSEIF Var1 <= Var2 (DW)	Var 2	condition2		
ELSEIF Var1 <= Var2 (Signed W)	Expression Explanation			
ELSEIF Var1 <= Var2 (Signed DW)	Or else, if condition1 is smaller than or equals to condition2, then execute...			

*IF...ENDIF is required to pair up with the ELSEIF Macro commands, or error messages will pop up during compile.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

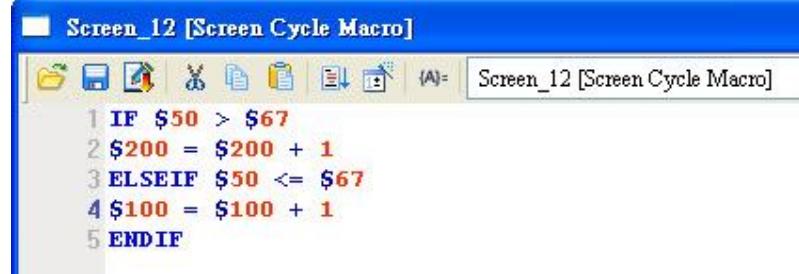
- Var 1 and Var 2 are both internal memory addresses.



The Macro Command dialog box shows the following configuration:

- Command: ELSEIF \$50 <= \$67
- Type: Double Word (unchecked)
- Signed: Signed (unchecked)
- Variables:

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2



The screen shows the following macro code:

```

1 IF $50 > $67
2 $200 = $200 + 1
3 ELSEIF $50 <= $67
4 $100 = $100 + 1
5 ENDIF

```

- If \$50 is larger than \$67 (value comparison), then execute \$200 = \$200+1; otherwise, execute \$100 = \$100 + 1.

(7) ELSEIF AND == 0		
Expression	What Variables Represent	NOTE
ELSEIF (Var1 && Var2) == 0 (W) ELSEIF (Var1 && Var2) == 0 (DW)	Var 1 condition1 Var 2 condition2	
Expression Explanation		
Or else, if the result of Bitwise AND Operation between condition 1 and condition 2 is 0, then execute...		W : Word DW : Double Word

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.

Macro Command

Edit Command

ELSEIF (\$50 && \$67) == 0

Double Word Signed

Command: ELSEIF AND == 0

Variables	Contents	Description
Var1	\$50	Condition1
Var2	\$67	Condition2

Screen_12 [Screen Cycle Macro]

```

1 IF ($50 && $67) != 0
2 $200 = $200 + 1
3 ELSEIF ($50 && $67) == 0
4 $100 = $100 + 1
5 ENDIF
    
```

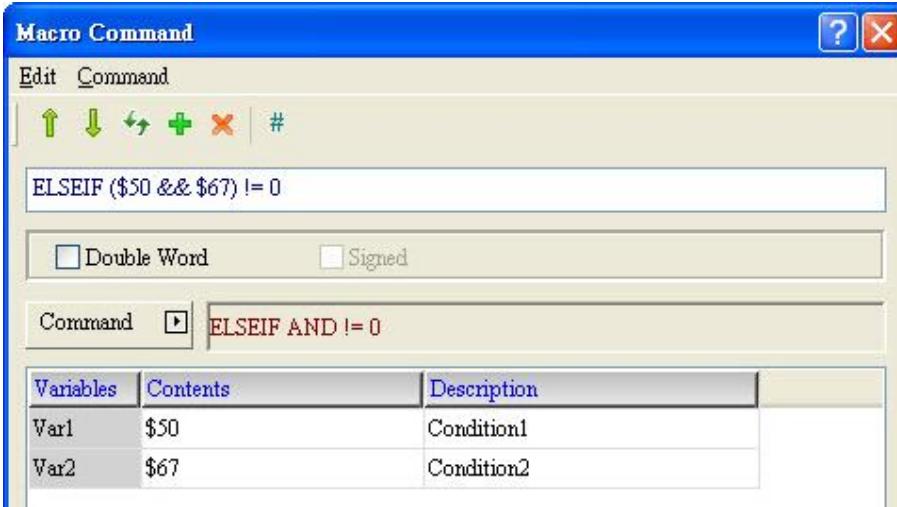
- If the result of Bitwise AND Operation between \$50 and \$67 is not 0, then execute \$200 = \$200+1; otherwise, execute \$100 = \$100 + 1.

(8) ELSEIF AND != 0		
Expression	What Variables Represent	NOTE
ELSEIF (Var1 && Var2) != 0 (W) ELSEIF (Var1 && Var2) != 0 (DW)	Var 1 condition1 Var 2 condition2 Expression Explanation Or else, if the result of Bitwise AND Operation between condition 1 and condition 2 is not 0, then execute...	W : Word DW : Double Word

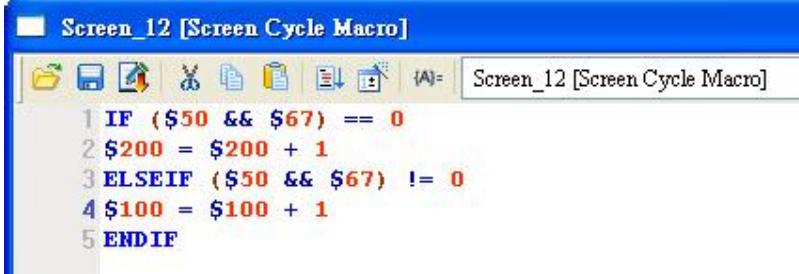
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example

- Var 1 and Var 2 are both internal memory addresses.



The Macro Command dialog box shows the command `ELSEIF ($50 && $67) != 0`. Below it, under "Command", is `ELSEIF AND != 0`. A table below lists variables and their values: Var1 is \$50 (Condition1) and Var2 is \$67 (Condition2).



The Screen_12 [Screen Cycle Macro] window displays the following macro code:

```

1 IF ($50 && $67) == 0
2 $200 = $200 + 1
3 ELSEIF ($50 && $67) != 0
4 $100 = $100 + 1
5 ENDIF

```

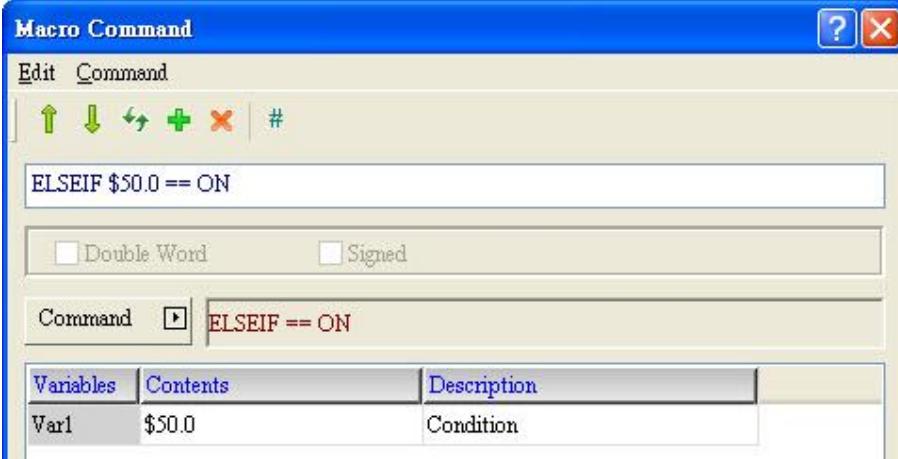
- If the result of Bitwise AND Operation between \$50 and \$67 is 0, then execute $\$200 = \$200 + 1$; otherwise, execute $\$100 = \$100 + 1$.

(9) ELSEIF == ON				
Expression	What Variables Represent		NOTE	
ELSEIF Var1 == ON (W)	Var 1	condition1	W : Word	
	Expression Explanation			
	Or else, if condition1 is ON, then execute...			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		

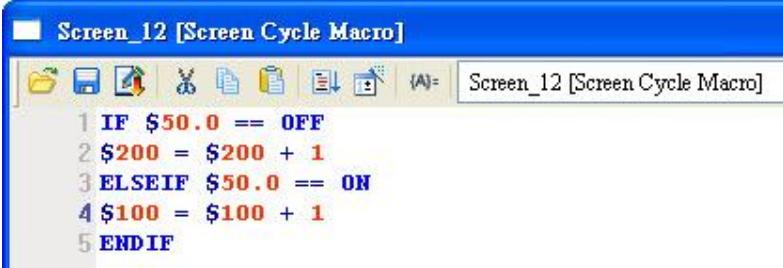
Example

- Var 1 is the internal memory address.



The Macro Command dialog box shows the following details:

- Command: ELSEIF \$50.0 == ON
- Type: Double Word
- Signed: Unsigned
- Variables Tab (selected): Var1 \$50.0 Condition



The Screen_12 [Screen Cycle Macro] window displays the following macro code:

```

1 IF $50.0 == OFF
2 $200 = $200 + 1
3 ELSEIF $50.0 == ON
4 $100 = $100 + 1
5 ENDIF

```

- If \$50.0 is OFF, then execute \$200 = \$200 + 1; if \$50.0 is ON, then execute \$100 = \$100 + 1.

(10) ELSEIF == OFF

Expression	What Variables Represent		NOTE
ELSEIF Var1 == OFF (W)	Var 1	condition1	
Expression Explanation			
Or else, if condition1 is OFF, then execute...			W : Word

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)		

Example

- Var 1 is the internal memory address.

Macro Command

Edit Command

ELSEIF \$50.0 == OFF

Double Word Signed

Command: ELSEIF == OFF

Variables	Contents	Description
Var1	\$50.0	Condition

Screen_12 [Screen Cycle Macro]

```

1 IF $50.0 == ON
2 $200 = $200 + 1
3 ELSEIF $50.0 == OFF
4 $100 = $100 + 1
5 ENDIF
    
```

- If \$50.0 is ON, then execute \$200 = \$200 + 1; if \$50.0 is OFF, then execute \$100 = \$100 + 1.

■ ELSE

The Else command is used to execute other procedures when either the If or Elseif statement is not true. Else needs to pair up with If and Endif, or error messages will pop up during compile.

Screen_12 [Screen Cycle Macro]

```

1 IF $50 == $67
2 $200 = $200 + 1
3 ELSEIF $50 > $67
4 $100 = $100 + 1
5 ELSE ←
6 $300 = $300 + 1
7 ENDIF
    
```

When IF and ELSEIF condition are not hold, then execute ELSE command

■ ENDIF

ENDIF is primarily used to pair up with commands such as If..., Else, and Elseif...

```

Screen_12 [Screen Cycle Macro]
1 IF $50 == $67
2 $200 = $200 + 1
3 ELSEIF $50 > $67
4 $100 = $100 + 1
5 ELSE
6 $300 = $300 + 1
7 ENDIF ←

```

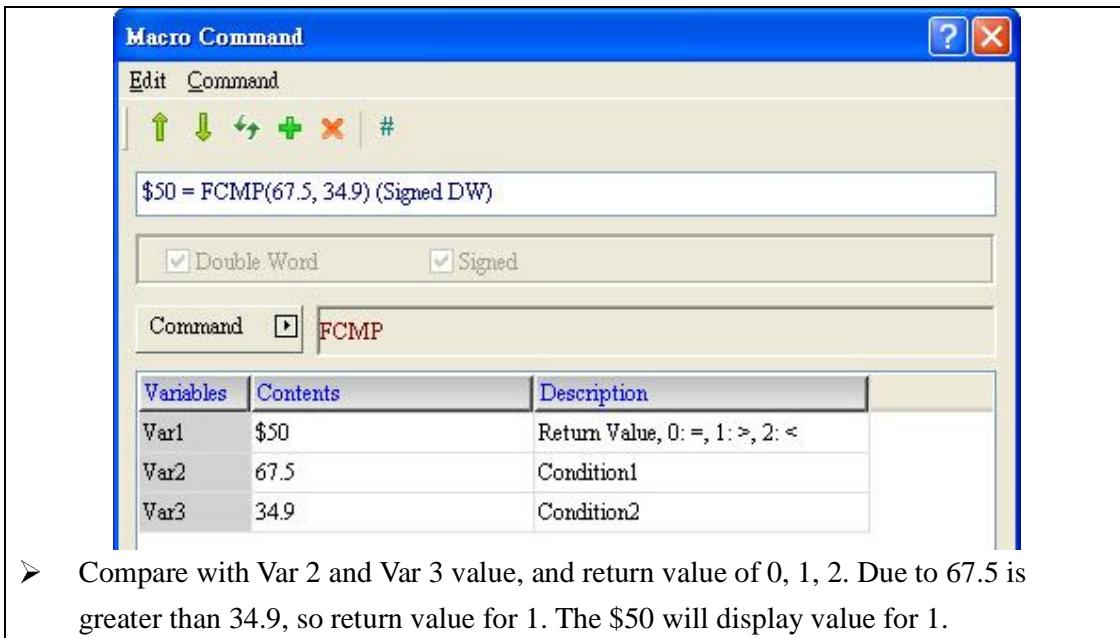
ENDIF have to use in company with IF, ELSEIF and ELSE

■ FCMP (Comparison of Floating Point Data)

Expression	What Variables Represent			NOTE
Var1 = FCMP(Var2, Var3) (Signed DW)	Var 1	Returned the compared result.	=	0
		=	>	1
		<		2
		Var 2		condition1
	Var 3			condition2
	Expression Explanation			
	Compare Var 2 and Var 3 and save the result in Var 1.			DW : Double Word Signed : Signed number

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎
Var 3	◎		◎

Example
<ul style="list-style-type: none"> ➤ Var 1 is an internal memory address, and Var 2 and Var 3 are constants.



24-3-6 Flow Control

There are seven commands for flow control: GOTO, LABEL, CALL, RET, FOR, NEXT and END and with them, users can control the sequence of a program being executed. Their usages are detailed below.

```

GOTO
LABEL
CALL
RET
FOR
NEXT
END

```

Figure 24-3-6-1 Flow Control

- GOTO LABEL (label identifier for the current process to unconditionally jump to)

Expression	What Variables Represent		NOTE	
GOTO LABEL Var1 (W)	Var 1	Label identifier for the current process to jump to	W : Word	
	Expression Explanation			
	A designated identifier for the current process to directly jump to.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1			◎

Example
<ul style="list-style-type: none"> ➤ Var 1 can only be Constant.  <p>The screenshot shows the 'Macro Command' dialog box. In the 'Command' field, 'GOTO' is selected. Below it, 'Variables' is chosen in the 'Command' dropdown. The 'Variables' table shows 'Var1' with value '1'. The 'Description' column for 'Var1' is 'Goto Label'.</p>

Screen_12 [Screen Cycle Macro]

```

1 GOTO LABEL 1
2 LABEL 1
3 $100 = $100 + 1

```

➤ Directly jump to Label 1 and the expression marked by Label 1 is $\$100 = \$100 + 1$.

■ LABEL (Label Identifier)

Expression	What Variables Represent		NOTE	
LABEL Var1 (W)	Var 1	Label identifier for the current process to jump to	W : Word	
	Expression Explanation			
	A designated identifier for the current process to directly jump to.			

* Each label has to have its unique label name and no identical label name is allowed for another label within the same macro.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1			◎

Example

➤ Var 1 can only be Constant.

Macro Command

Variables	Contents	Description
Var1	1	Label Name

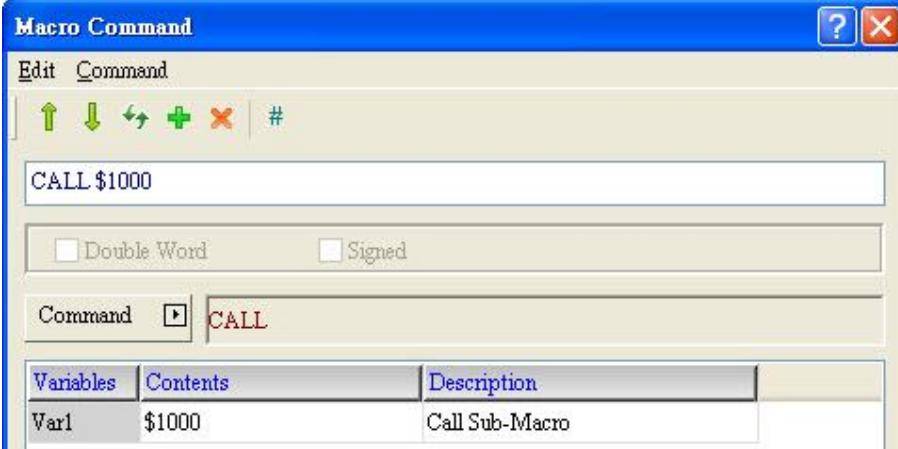
■ CALL (Call Submacro)

Expression	What Variables Represent		NOTE	
CALL Var1 (W)	Var 1	Submacro ID(1~512)	W : Word A designated identifier for the current process to directly jump to.	
	Expression Explanation			
	A designated identifier for the current process to directly jump to.			

* Var 1 supports both Chinese and English names, but in such case, please manually enter the alias of the macro. The macro wizard only supports a submacro ID.



Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎

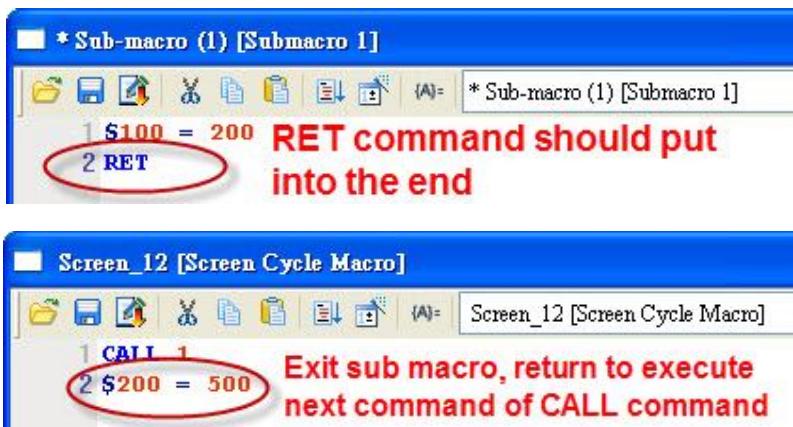
Example
<ul style="list-style-type: none"> ➤ Var 1 is an internal memory address. <div style="border: 1px solid #ccc; padding: 10px;">  </div> <ul style="list-style-type: none"> ➤ Users can enter submacro ID via \$1000, or an internal memory address, to execute commands within a macro.

■ RET (Exit Submacro)

Expression	Expression Explanation	NOTE
RET	Exit a submacro and return back to the next line right after a submacro.	RET needs to be at the last line of a submacro and needs to pair up with the Call command.
<pre> graph TD CALL[CALL 1] --> Execute[Execute Sub Macro NO. 1] Execute --> RET[RET] RET --> Return[Return CALL command and execute next command of CALL command] </pre>		

Example

- RET needs to be at end of a macro



■ FOR, NEXT (Loop)

Expression	What Variables Represent		NOTE	
FOR Var1 (W)	Var 1	Number of loops	W : Word	
	Expression Explanation			
	Statement to run the loop "Var 1" times			
Expression	Expression Explanation		NOTE	
NEXT	Need to pair up with "For"			

*multiple layers are possible (up to a maximum of three loops).

```

1 FOR 4
2 $50 = $50 + 1
3 FOR 4
4 $50 = $50 + 1
5 FOR 4
6 $50 = $50 + 1
7 FOR 4
8 $50 = $50 + 1
9 FOR 4
10 $50 = $50 + 1
11 FOR 4
12 $50 = $50 + 1
13 FOR 4
14 $50 = $50 + 1
15 FOR 4
16 $50 = $50 + 1
17 FOR 4
18 $50 = $50 + 1
19 FOR 4
20 $50 = $50 + 1
21 next
22 next
23 next
24 next
25 next
26 next
27 next FOR ... NEXT command only
28 next support ten levels structure
29 next
30 next

```

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎

Example

- Var 1 is a constant.

Macro Command

Edit Command

| + X | #

FOR 4

Double Word Signed

Command FOR

Variables	Contents	Description
Var1	4	Loop Counter

Screen_12 [Screen Cycle Macro]

[] (A): Screen_12 [Screen Cycle Macro]

```

1 FOR 4
2 $100 = $100 + 1
3 NEXT

```

- For 4 means to execute the statement “\$100 = \$100 + 1” four times and hence, the result is 4.

■ END (End Macro)

Expression	Expression Explanation	NOTE
END	End a macro.	If END command is used in a submacro, it indicates the program will not execute the next statement within the macro that the submacro belongs to.
<pre> graph TD S1[Statement 1] --> S2[Statement 2] S2 --> S3[Statement 3] S2 -- feedback --> S1 </pre> <p>Will not continue to execute statement 3</p>		

Example

- Below END command will not execute.

Screen_12 [Screen Cycle Macro]

```
1 $300 = $300 + 1
2 $301 = $300 % 10
3
4 IF $350 == 3
5   $312 = 2
6   $313 = 0
7 ELSEIF $350 == 2
8   $312 = 0
9   $313 = 2
10 ENDIF
11 END ← It didn't execute below the END command
12 IF $301 == 0
13   IF $311 == 2
14     $310 = $310 - 5
15   ENDIF
16   IF $312 == 2
17     $320 = $320 + 5
18   ENDIF
19   IF $313 == 2
20     $330 = $330 + 5
21   ENDIF
22   IF $345 == 1
23     $340 = $340 + 10
24     $342 = $342 - 10
25 ELSEIF $345 == 2
26   $340 = $340 - 10
27   $342 = $342 + 10
28 ENDIF
29 ENDIF
```

- If the END command is at the end of a submacro, it indicates the program will not execute the next statement within the macro the submacro belongs to.

A screenshot of the AutoCAD ribbon interface. The 'Sub-macro (1) [Submacro 1]' tab is highlighted in blue. Below the ribbon, the status bar displays the text '1 \$100 = 200' and '2 END'. A red oval is drawn around the '2 END' text.

Screen_12 [Screen Cycle Macro]

(A): Screen_12 [Screen Cycle Macro]

1 CALL 1

2 \$100 = 500 This command cannot be executed

24-3-7 Bit Settings

Four commands are available for Bit settings. With these commands, users can set the ON/OFF or inverse status of a bit, or display the value of a bit. These commands are detailed below.

BITON
BITOFF
BITNOT
GETB

Figure 24-3-7-1 Bit Settings

■ BITON (Set Bits to ON)

Expression	What Variables Represent		NOTE
BITON Var1 (W)	Var 1	Set the state of the bit	
	Expression Explanation		
	Set the n^{th} bit to ON (N as denoted by Var 1).		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)	◎ (Can only be Bit)	

Example

- Var 1 is an internal memory address.

Macro Command

Edit Command

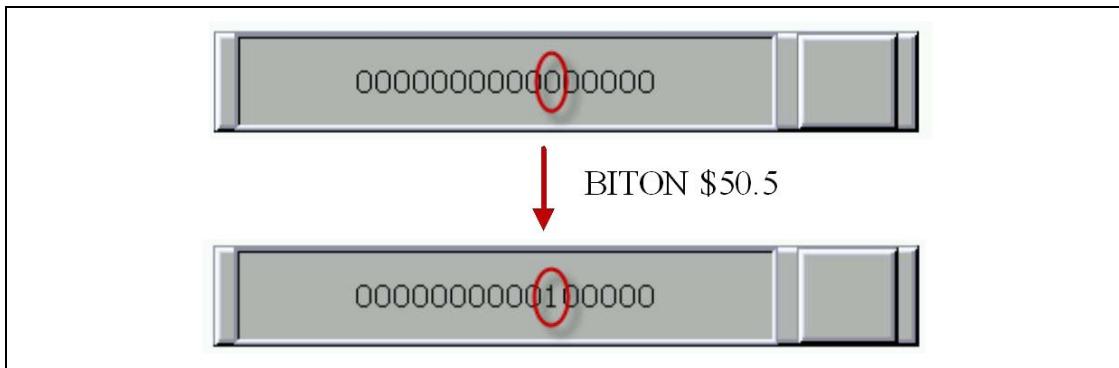
BITON \$50.5

Double Word Signed

Command: BITON

Variables	Contents	Description
Var1	\$50.5	Bit On Addresss

- Set \$50 to be the variable to set the bit and data type to be binary. When executing BITON \$50.5, then the fifth bit will be set to ON.



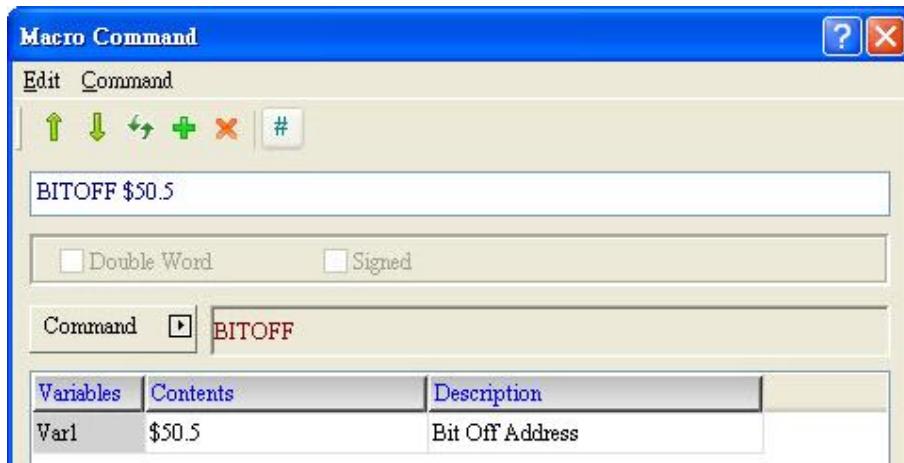
■ BITOFF (Set Bits to OFF)

Expression	What Variables Represent		NOTE	
BITOFF Var1 (W)	Var 1	Set the state of the bit	W : Word	
	Expression Explanation			
	Set the n^{th} bit to OFF (N as denoted by Var 1).			

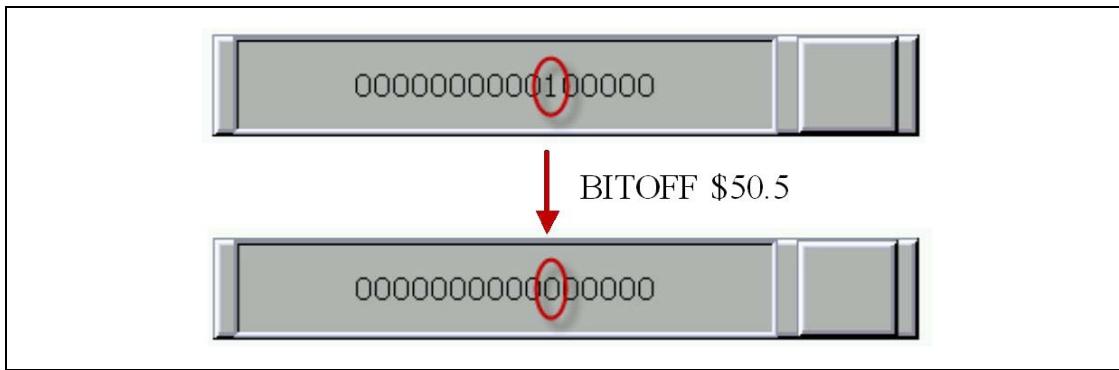
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be Bit)	◎ (Can only be Bit)	

Example

- Var 1 is an internal memory address.



- Set \$50 to be the variable to set the bit and data type to be binary. When executing BITOFF \$50.5, then the fifth bit will be set to OFF.



■ BITNOT (Set the bit to inverse state, ON→OFF, OFF→ON)

Expression	What Variables Represent		NOTE
BITNOT Var1 (W)	Var 1	Set the state of the bit	
	Expression Explanation		
	Set the n^{th} bit to its inverse state: ON→OFF or OFF→ON (N as denoted by Var 1).		W : Word

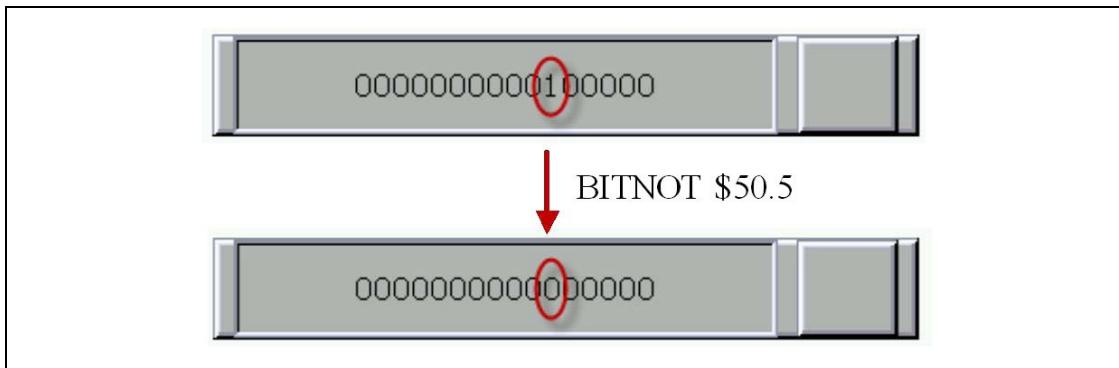
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be bit)	◎ (Can only be bit)	

Example

- Var 1 is an internal memory address.



- Set \$50 to be the variable to set the bit and data type to be binary. When executing BITNOT \$50.5, then the fifth bit will be set from ON to OFF.



■ GETB (Acquire Bit State)

Expression	What Variables Represent		NOTE	
(Var1) = GETB (Var2) (W)	Var 1	Set the state of the bit	W : Word Get n th bit value and store it in Var 1(N as denoted by Var 2).	
	Expression Explanation			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎ (Can only be bit)	◎ (Can only be bit)	
Var 2	◎ (Can only be bit)	◎ (Can only be bit)	

Example									
<ul style="list-style-type: none"> Var 1 and Var 2 are both internal memory addresses. <div style="border: 1px solid #ccc; padding: 10px;"> <p>Macro Command</p> <p>Edit Command</p> <p>\$50.0 = GETB \$67.0</p> <p><input type="checkbox"/> Double Word <input type="checkbox"/> Signed</p> <p>Command: GETB</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$50.0</td> <td>Destination Bit</td> </tr> <tr> <td>Var2</td> <td>\$67.0</td> <td>Source Bit</td> </tr> </tbody> </table> </div> <ul style="list-style-type: none"> Set \$50 to be the variable to set the bit and data type to be binary. Set \$50.0 and \$67.0 to be the ON button. When executing \$50.0 = GETB 	Variables	Contents	Description	Var1	\$50.0	Destination Bit	Var2	\$67.0	Source Bit
Variables	Contents	Description							
Var1	\$50.0	Destination Bit							
Var2	\$67.0	Source Bit							

\$67.0 and push the \$67.0 button, then \$50.0 will be triggered to be ON.

$\$50.0 = \text{GETB}(\$67.0)$



24-3-8 COM Port

COM Port macros are used to control com ports and they are detailed below.

INITCOM
ADDSUM
XORSUM
PUTCHARS
GETCHARS
SELECTCOM
CLEARCOMBUFFER
CHRCHKSUM
LOCKCOM
UNLOCKCOM
STATIONON
STATIONOFF
IPON
IPOFF

Figure 24-3-8-1 COM Port

■ INITCOM (COM Port Initialization)

Expression	What Variables Represent		NOTE	
Var1 = INITCOM(Var2, Var3, Var4, Var5, Var6, Var7, Var8) (W)	Var 1	Returned result	W : Word	
		0 : Failure		
		1 : Successful		
	Var 2	Com Port		
	Var 3	Interface		
	Var 4	Data Bit		
	Var 5	Parity		
	Var 6	Stop Bit		
	Var 7	Baud Rate		
	Var 8	Flow Control		
Expression Explanation				
		Initialize com ports, open com ports, set the protocol (Var 2 ~ Var 8) and return the result of initialization back to Var 1.		

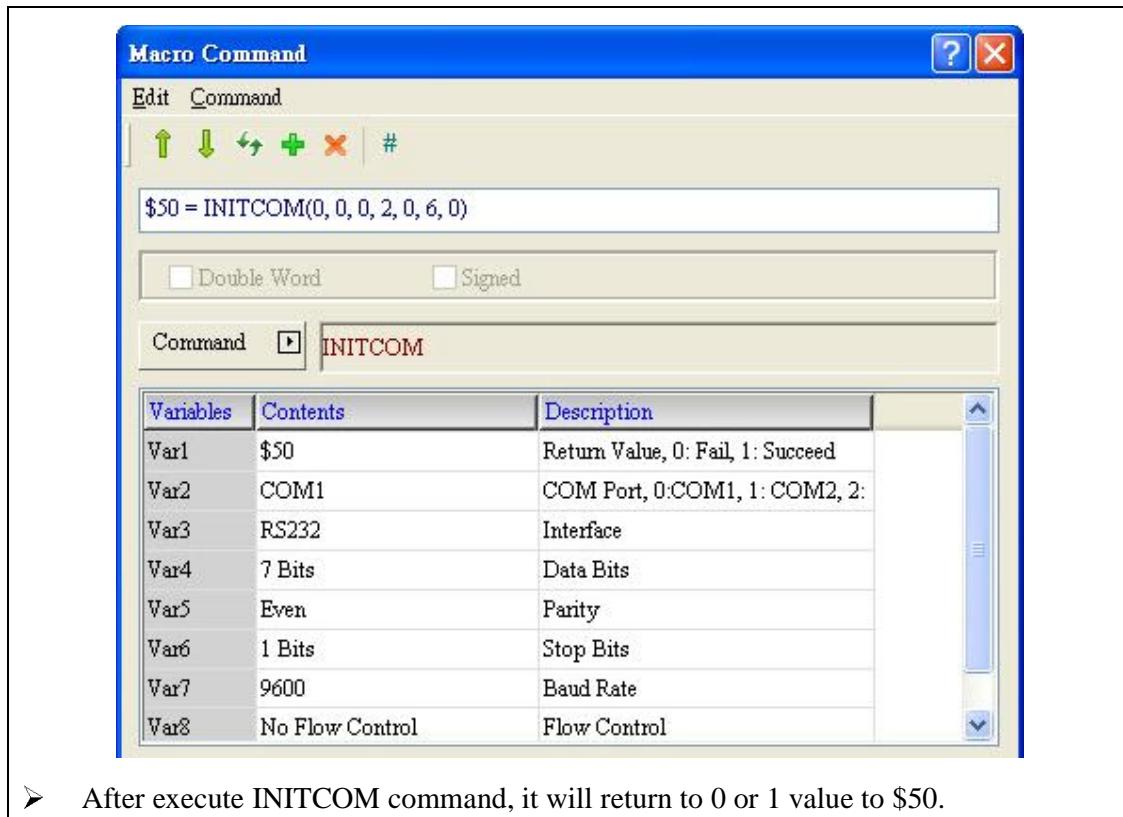
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Parameter Setup			
Variable	Parameter	Parameter Details	Corresponding Codes
Var 2	Com Port	COM 1	0
		COM 2	1
		COM 3	2
Var 3	Interface	RS242	0
		RS422	1
		RS485	2
Var 4	Data Bit	7 Bits	0
		8 Bits	1
Var 5	Parity	None	0
		Old	1
		Even	2
Var 6	Stop Bit	1 Bits	0
		2 Bits	1
Var 7	Baud Rate	300	0
		600	1
		900	2
		1200	3
		2400	4
		4800	5
		9600	6
		14400	7
		19200	8
		28800	9
		38400	10
		57600	11
		115200	12

Var 8	Flow Control	No Flow Control	0
		CTS RTS Flow Control	1
		DTR DSR Flow Control	2
		Xon Xoff Flow Control	3

Notes about Flow Control		
No Flow Control		Flow control function is disabled.
Flow Control		The transmission speed and communication validity are enhanced during communication due to new transmission technology, such as compress immediately, debug,...etc. But the new technology also makes the transmission speed between HMI and PC will be longer than the actual transmission speed. Therefore, ensure the data security and transmit complete data between computer and HMI, when transmitting data through serial communication port, the flow control is necessary.
Flow Control	CTS / RTS Flow Control	Flow control for hardware. It uses handshaking signal to control receiving and sending data. The control is achieved via internal modem or external modem that connects to HMI by connecting cable.
	DSR / DTR Flow Control	It is flow control for hardware also. It is used when PC and HMI is connected by cable directly.
	Xon / Xoff Flow Control	It is flow control for software. It is only used for 2400bps modem. The control method is to generate control code by software and add it in the transmission data.

Example
➤ Var 1 is an internal memory address



■ ADDSUM (Checksum Calculation through Addition)

Expression	What Variables Represent		NOTE
Var1 = ADDSUM(Var2, Var3) (W)	Var 1	Checksum	W : Word
	Var 2	Starting address of the source data	
	Var 3	Data length	
	Expression Explanation		
Calculate the checksum using addition. Var1 stores the calculated checksum value, Var2 stores the starting address for data to be calculated and Var3 stores the length of data.			W : Word

*the checksum value calculated through ADDSUM is based on BYTE. If the data length is 6, it must be divided by 2 for the correct length is 3.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎

Example
<ul style="list-style-type: none"> Var 1 and var 2 are internal memory addresses, and Var 3 is a constant. <p>The screenshot shows the Macro Command dialog box. The expression input field contains '\$50 = ADDSUM(\$67, 6)'. Below the input field are checkboxes for 'Double Word' and 'Signed'. The 'Command' dropdown is set to 'ADDSUM'. At the bottom, there is a table titled 'Variables' with three rows: Var1 (\$50), Var2 (\$67), and Var3 (6).</p> <ul style="list-style-type: none"> After addition, the three consecutive bytes ($6 / 2 = 3$), starting from the address stored in \$67, will be stored in \$50. The expression means $\\$67 + \\$68 + \\$69 = \\50.

■ XORSUM (Checksum Calculation through XOR Operation)

Expression	What Variables Represent		NOTE
Var1 = XORSUM(Var2, Var3) (W)	Var 1	Checksum	W : Word
	Var 2	Starting Address of the Source Data	
	Var 3	Data Length	
Expression Explanation			
Calculate the checksum using XOR operations. Var1 stores the calculated checksum, Var2 stores the starting address for data to be calculated and Var3 stores the length of the data.			
*the checksum value calculated through XORSUM is based on BYTE. If the data length is 6, it must be divided by 2 for the correct length is 3.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎

Example
<ul style="list-style-type: none"> Var 1 and var 2 are internal memory addresses, and Var 3 is a constant. <p>The screenshot shows the Macro Command dialog box. The command is set to 'XORSUM'. The variables are listed in the 'Variables' column: Var1 (\$50) is described as 'Checksum Value'; Var2 (\$67) is described as 'Source Address'; and Var3 (6) is described as 'Data Length'. There are also 'Double Word' and 'Signed' checkboxes.</p> <ul style="list-style-type: none"> After XOR operations, the three consecutive bytes ($6 / 2 = 3$), starting from the address stored in \$67, will be stored in \$50. The expression means $\\$67 + \\$68 + \\$69 = \\50.

■ PUTCHARS (Output Character by Com Port)

Expression	What Variables Represent			NOTE				
Var1 = PUTCHARS(Var2, Var3, Var4) (W)	Var 1	Returned result		W : Word				
		Failure	0					
		Successful	1					
	Var 2	Starting address of the source data						
	Var 3	Data length						
	Var 4	Duration of data transmission						
Expression Explanation								
Send in data (including starting address stored in Var 2, data length specified in Var 3, and required transmission duration saved in Var 4) via selected communication ports and save the returned value in Var 1.								
*PUTCHARS must be paired up with INITCOM and SELECTCOM. *Var 3 is Byte format.								

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎
Var 4	◎		◎

Example																
<ul style="list-style-type: none"> ➤ Var 1 and var 2 are internal memory addresses, and Var 3 and Var 4 are constants. 																
<p>The dialog box shows the command \$50 = PUTCHARS(\$67, 12, 500). The command is PUTCHARS. The variables are listed in the table below:</p> <table border="1"> <thead> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$50</td> <td>Return Value, 0: Fail, 1: Succeed</td> </tr> <tr> <td>Var2</td> <td>\$67</td> <td>Source Address</td> </tr> <tr> <td>Var3</td> <td>12</td> <td>Data Length</td> </tr> <tr> <td>Var4</td> <td>500</td> <td>Communication Time</td> </tr> </tbody> </table>		Variables	Contents	Description	Var1	\$50	Return Value, 0: Fail, 1: Succeed	Var2	\$67	Source Address	Var3	12	Data Length	Var4	500	Communication Time
Variables	Contents	Description														
Var1	\$50	Return Value, 0: Fail, 1: Succeed														
Var2	\$67	Source Address														
Var3	12	Data Length														
Var4	500	Communication Time														

```

1 $1 = INITCOM 0, 0, 1, 2, 0, 6, 0
2 SELECTCOM(0)
3 FILASC($67, ":FEE0020")
4 $71 = 0D30H
5 $72 = 000AH
6 $50 = PUTCHARS($67, 12, 500)
    
```

➤ Save inputted characters from \$67 to \$50 along with relevant parameters (12 consecutive characters; transmission duration: 500; and returned value: 0 or 1).

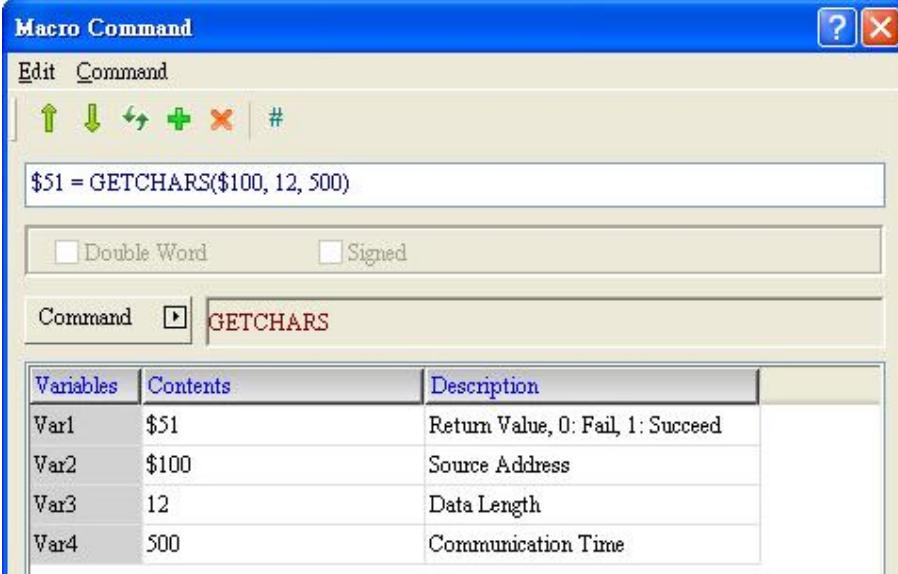
■ GETCHARS (Character Acquisition through Com Port)

Expression	What Variables Represent			NOTE				
Var1 = GETCHARS(Var2, Var3, Var4) (W)	Var 1	Returned result		W : Word				
		Failure	0					
		Successful	1					
	Var 2	Starting address of the source data						
	Var 3	Data length						
	Var 4	Duration of data transmission						
Expression Explanation								
Send in data (including starting address stored in Var 2, data length specified in Var 3, and required transmission duration saved in Var 4) via selected communication ports and save the returned value in Var 1.								
*GETCHARS much be paired up with INITCOM and SELECTCOM.								
*Var 3 is Byte format.								

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		◎
Var 4	◎		◎

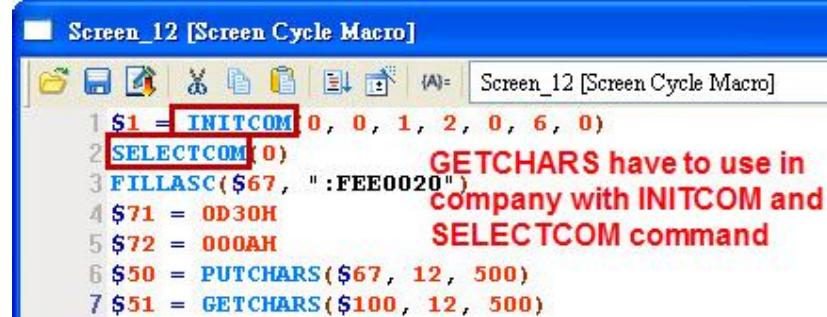
Example

- Var 1 and var 2 are internal memory addresses, and Var 3 and Var 4 are constants.



The screenshot shows the Macro Command dialog box. The command entered is `$51 = GETCHARS($100, 12, 500)`. The variables section shows:

Variables	Contents	Description
Var1	\$51	Return Value, 0: Fail, 1: Succeed
Var2	\$100	Source Address
Var3	12	Data Length
Var4	500	Communication Time



The screenshot shows the Screen_12 [Screen Cycle Macro] window. The assembly code is:

```

1 $1 = INITCOM 0, 0, 1, 2, 0, 6, 0
2 SELECTCOM 0)           GETCHARS have to use in
3 FILLASC($67, " :FEE0020")
4 $71 = 0D30H             company with INITCOM and
5 $72 = 000AH              SELECTCOM command
6 $50 = PUTCHARS($67, 12, 500)
7 $51 = GETCHARS($100, 12, 500)

```

Comments:

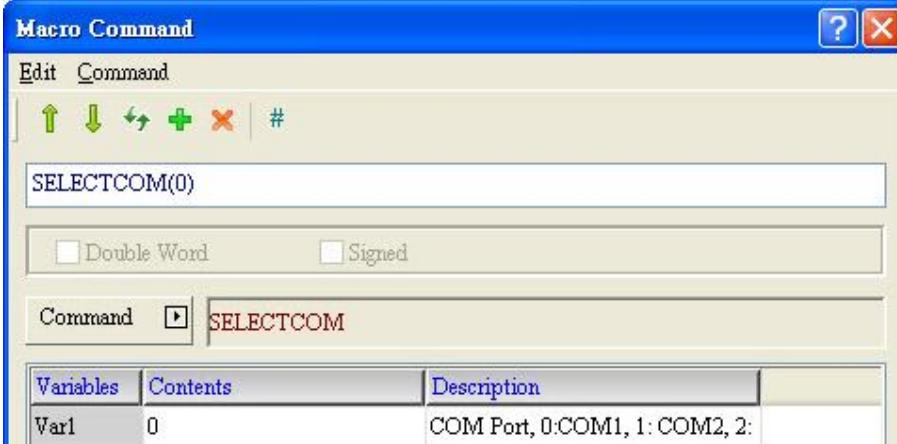
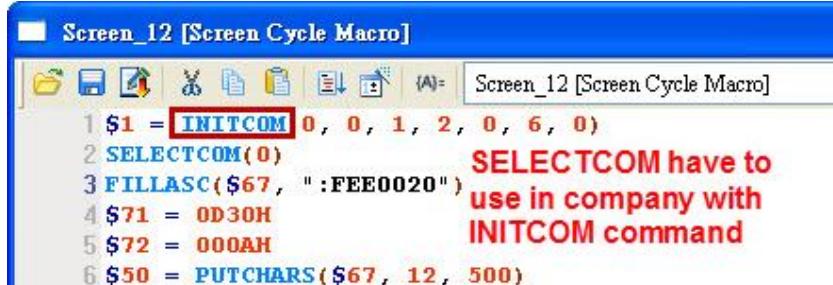
- Line 2: `SELECTCOM 0)` **GETCHARS have to use in company with INITCOM and SELECTCOM command**

- Save received characters from \$100 to \$51 along with relevant parameters (12 consecutive characters; transmission duration: 500; and returned value: 0 or 1).

■ SELECTCOM (Com Port Selection)

Expression	What Variables Represent			NOTE
SELECTCOM(Var1) (W)	Var 1	COM 1	0	W : Word Send in acquired characters (including starting address stored in Var 2, data length specified in Var 3, and required transmission duration saved in Var 4) via selected communication ports and save the returned value in Var 1.
		COM 2	1	
		COM 3	2	
	Expression Explanation			
*SELECTCOM much be paired up with INITCOM. *The designated com port can not be the same as the com port used by the system. All communication commands will be processed via the COM port the user selects after executing this command. Therefore, the Selectcom command of a particular macro does not support other macros and there will be no interference between different macros.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1			◎

Example
<ul style="list-style-type: none"> ➤ Var 1 can only be Constant.  <p>The Macro Command dialog box shows the command <code>SELECTCOM(0)</code>. The 'Variables' tab displays <code>Var1 = 0</code>, which is highlighted in red. The 'Description' column indicates it is a constant.</p>  <p>The Screen_12 [Screen Cycle Macro] window contains the following code:</p> <pre> 1 \$1 = INITCOM 0, 0, 1, 2, 0, 6, 0 2 SELECTCOM(0) 3 FILASC(\$67, ":FEE0020") 4 \$71 = 0D30H 5 \$72 = 000AH 6 \$50 = PUTCHARS(\$67, 12, 500) </pre> <p>A red annotation on the right side of the screen states: SELECTCOM have to use in company with INITCOM command.</p>

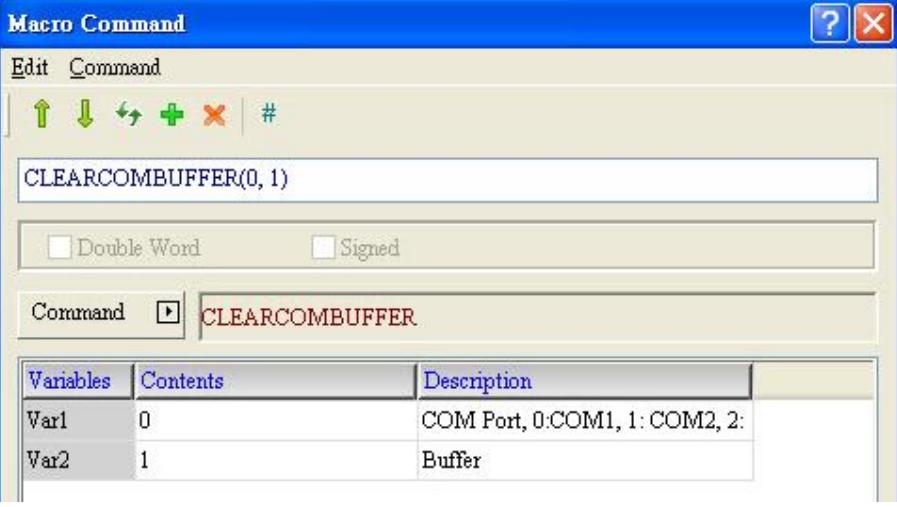
■ CLEARCOMBUFFER (Com Port Buffer Clearance)

Expression	What Variables Represent			NOTE	
CLEARCOMBUFFER(Var1, Var2) (W)	Var 1	COM 1	0	W : Word Var 1 : COM Port Number Var 2 : Buffer	
		COM 2	1		
		COM 3	2		
	Var 2	Receiving buffer	0		
		Transmitting buffer	1		
Expression Explanation					
Clear the buffer for the N th Com Port (N as denoted by Var 1).					

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1			◎
Var 2			◎

Example

➤ Var 1 and var 2 can only be Constant.



The screenshot shows the Macro Command dialog box. The command entered is `CLEARCOMBUFFER(0, 1)`. The variables section shows `Var1` with value `0` and `Var2` with value `1`. The description for Var1 is `COM Port, 0:COM1, 1:COM2, 2:` and for Var2 is `Buffer`.

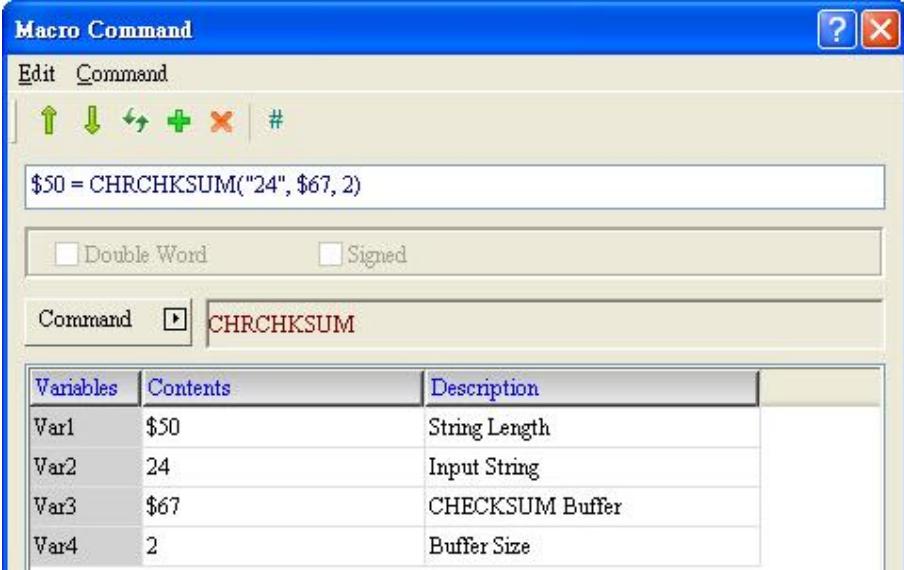
■ CHRCHKSUM (Calculation of String Length and Checksum)

Expression	What Variables Represent				NOTE							
Var1 = CHRCHKSUM("Var2", Var3, Var4) (W)	Var 1	String length		W : Word	W : Word							
	Var 2	Inputted string										
	Var 3	Memory address to store string										
	Var 4	Format to display the checksum	1 Byte	1								
			2 Bytes (Word)	2								
	Expression Explanation											
Calculate the string length and checksum and save them in Var 1												
* the string length for string stored in Var 1 will be different based on the format set in Var 4. * if the inputted string is "345" and Var 4 is set to 2, then the Var 1 string length will be 5; or else if Var 4 is set to 1, then the Var 1 string length will be 4. (based on BYTE)												

Variable	Memory Usage			
	Internal Memory	PLC Register	String	Constant
Var 1	◎			
Var 2			◎	
Var 3	◎			
Var 4				◎ (Can only be 1 and 2)

Example

➤ **Example 1:**



\$50 = CHRCHECKSUM ("24" , \$67, 2)

Variables	Contents	Description
Var1	\$50	String Length
Var2	24	Input String
Var3	\$67	CHECKSUM Buffer
Var4	2	Buffer Size

String length Address to store string length Checksum of the stored string length

\$50 \$67 \$68

4 = { 0X34 | 0X32 } { 0X66 | 0X00 }

**0X34
+ 0X32
0X66**

Low Byte High Byte

2 means the checksum takes 2 bytes to display

➤ **Example 2:**

Macro Command

<input type="button" value="Edit"/>	<input type="button" value="Command"/>	<input type="button" value="?"/>	<input type="button" value="X"/>															
<input type="button" value="↑"/> <input type="button" value="↓"/> <input type="button" value="↶"/> <input type="button" value="↷"/> <input type="button" value="+"/> <input type="button" value="X"/> #																		
\$50 = CHRCHKSUM("2425", \$67, 1)																		
<input type="checkbox"/> Double Word <input type="checkbox"/> Signed																		
Command <input type="button" value="CHRCHECKSUM"/>																		
<table border="1"> <thead> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$50</td> <td>String Length</td> </tr> <tr> <td>Var2</td> <td>2425</td> <td>Input String</td> </tr> <tr> <td>Var3</td> <td>\$67</td> <td>CHECKSUM Buffer</td> </tr> <tr> <td>Var4</td> <td>1</td> <td>Buffer Size</td> </tr> </tbody> </table>				Variables	Contents	Description	Var1	\$50	String Length	Var2	2425	Input String	Var3	\$67	CHECKSUM Buffer	Var4	1	Buffer Size
Variables	Contents	Description																
Var1	\$50	String Length																
Var2	2425	Input String																
Var3	\$67	CHECKSUM Buffer																
Var4	1	Buffer Size																

\$50 = CHRCHECKSUM ("2425" , \$67, 1)

String length **Address to store string length** **Checksum of the stored string length**

\$50 **\$67** **\$68** **\$69**

$$\boxed{5} = \left\{ \boxed{0X34 \mid 0X32} \mid \boxed{0X35 \mid 0X32} \right\} \left\{ \boxed{0XCD \mid \boxed{}} \right\}$$

↓ ↓ ↓

0X34 0X35 0XCD
 0X32 + 0X32 |
 0X35
 + 0X32
 0XCD

Low Byte High Byte

1 means the checksum takes 1 byte to display

■ LOCKCOM / UNLOCKCOM (Lock Com Port/Unlock Com Port)

Expression	What Variables Represent			NOTE				
Var1 = LOCKCOM(Var2, Var3) (W)	Var 1	Returned result		W : Word Var 1: Returned result Var 2: COM Port Var 3: Time out value				
		Failure	0					
		Successful	1					
	Var 2	COM 1	0					
		COM 2	1					
		COM 3	2					
	Var 3	Time out value						
	Expression Explanation							
	Lock Com Port							
	Var 1	COM 1	0					
		COM 2	1					
		COM 3	2					
UNLOCKCOM(Var1) (W)	Expression Explanation							
	Unlock Com Port							

* If Lockcom is set to continuously wait without limit (or Var 3 = 0), it indicates that the Lockcom will be executed twice within the same macro and this will cause the HMI to become unresponsive.

* When the communication commands are used in different macros, such as Screen Cycle Macro, Clock Macro, Background Macro, Run Pre-action/Post-action Macro (ON/OFF Macro) and Screen Open/Close Macro, the different macros may have interferences and cause error results. The solution to this issue is to add the LOCKCOM and UNLOCKCOM commands before and after communication commands, and this is to ensure that communication will not be interrupted for other purposes and preserve integrity of transmitted messages.

Memory Usage (LOCKCOM)			
Variable	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2			◎
Var 3			◎

Memory Usage (UNLOCKCOM)			
Variable	Internal Memory	PLC Register	Constant
Var 1			◎

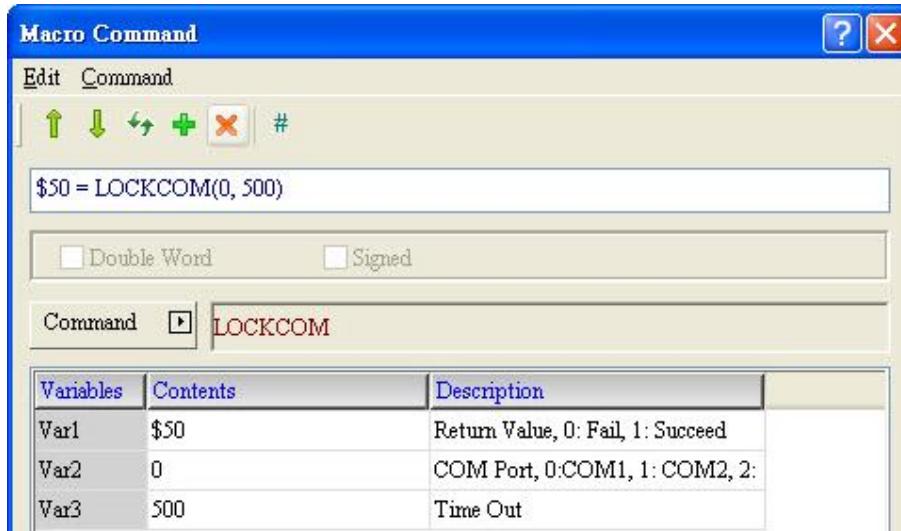
Followings are the examples describe the situations when properly applying, improperly applying and misapplying LOCKCOM / UNLOCKCOM macro command.

Example of Applying LOCKCOM / UNLOCKCOM (correctly use)		
Background Macro	On Macro	Screen Cycle Macro
Assuming that communication commands are executed in three macros, if LOCKCOM (0, 500) is executed in Background macro, it means COM 1 is locked at the moment. Since COM1 has been locked by Background macro, On Macro and Screen Cycle macro will stop when executing to LOCKCOM (0, 500). LOCKCOM (0, 500) in On Macro or Xscreen Cycle macro can be executed until COM 1 is released (when Background macro executes UNLOCKCOM(0)). This could avoid data transmission and receiving error.		
Example of Applying LOCKCOM / UNLOCKCOM (improperly use)		
On Macro	Screen Cycle Macro	
\$51 = GETCHARS(\$67, 3, 300)	\$50 = LOCKCOM(0,500) \$51 = PUTCHARS(\$67, 3, 300) UNLOCKCOM(0)	
Assuming that communication commands are executed in two macros, if LOCKCOM (0, 500) is executed in Screen Cycle Macro, it means COM 1 is locked at the moment. However, On Macro is not locked by LOCKCOM, which means it still can execute GETCHARS command. It does not have to wait until UNLOCKCOM command is executed by Screen Cycle Macro. This would result in data transmission and receiving error. Please avoid the situation that mentioned above.		
Example of Applying LOCKCOM / UNLOCKCOM (misapplying)		
Background Macro	On Macro	
\$50 = LOCKCOM(0, 500) \$51 = PUTCHARS(\$67, 3, 300)	UNLOCKCOM(0)	

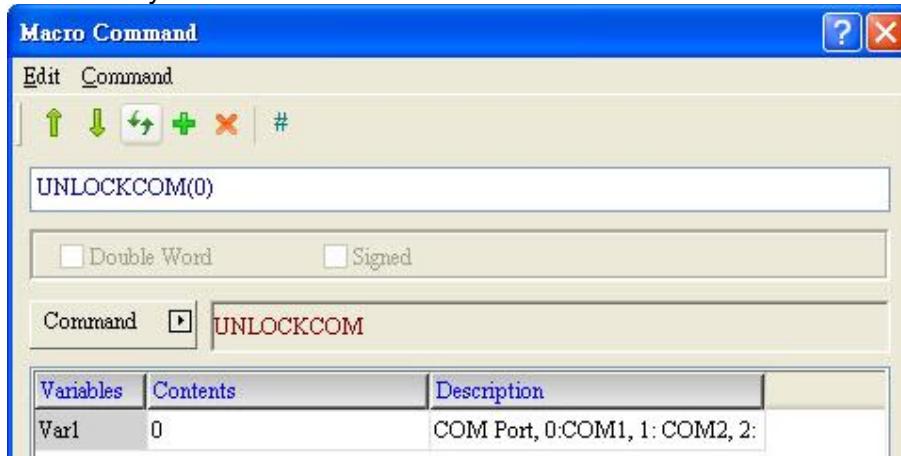
Assuming that COM port is locked in Background Macro and is used to transmit data, but COM Port cannot be unlocked in On Macro. This means commands to lock and unlock COM Port cannot be separated.

Example

- Var 1 is an internal memory address, and Var 2 and Var 3 can only be Constant



- Var 1 can only be constant.



```

Screen_12 [Screen Cycle Macro]
1 $50 = LOCKCOM(0, 500)
2 $51 = PUTCHARS($67, 12, 500)
3 $52 = GETCHARS($100, 12, 500)
4 UNLOCKCOM(0)

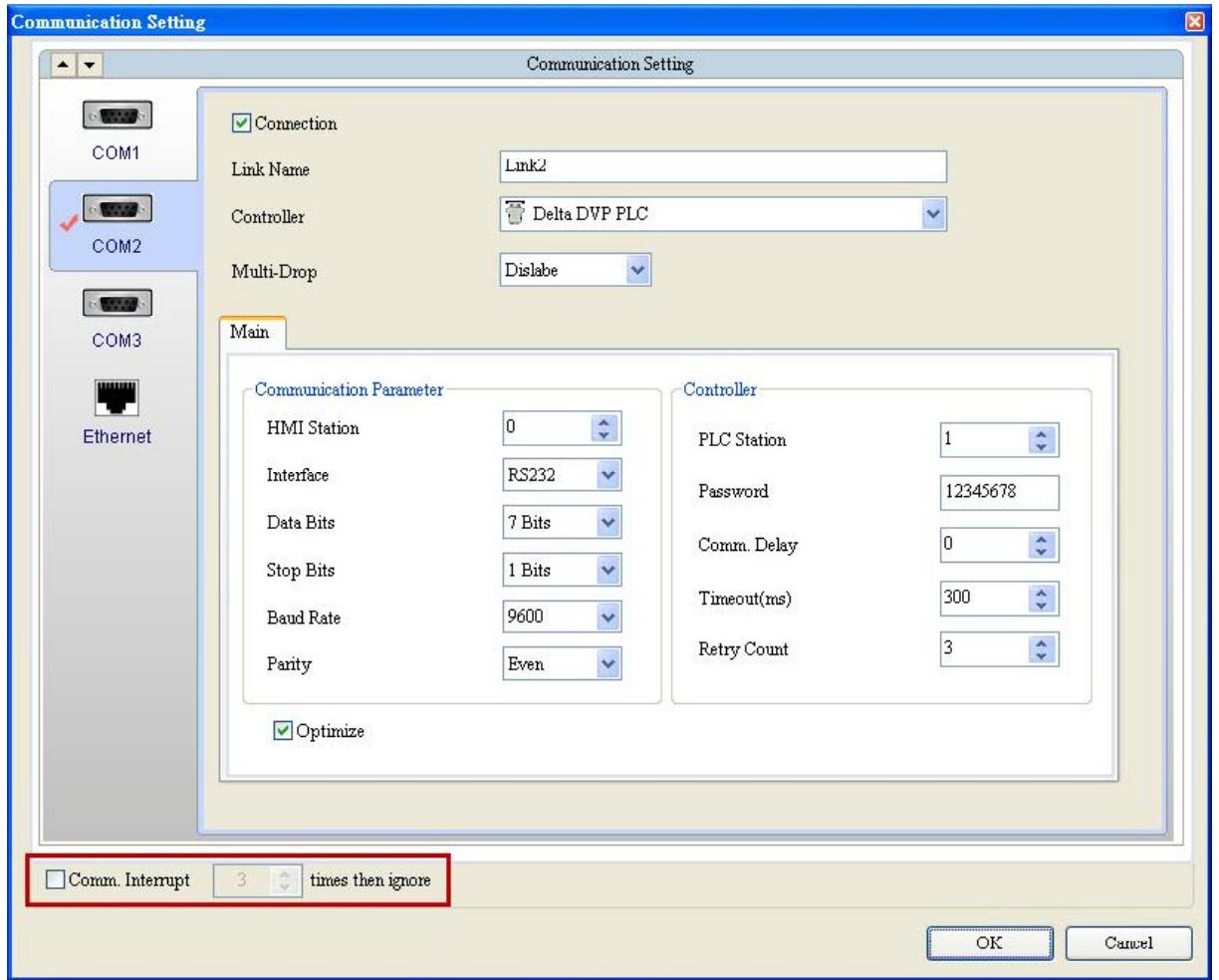
```

■ STATIONON (Set Station On)

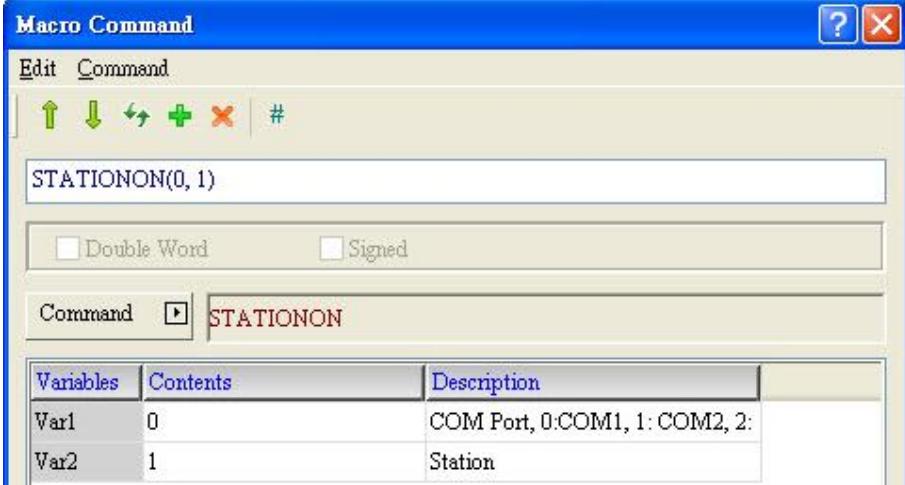
Expression	What Variables Represent			NOTE
STATIONON(Var1, Var2) (W)	Var 1	COM 1	0	W : Word
		COM 2	1	

	COM 3	2	
	Var 2	Station ID	
Expression Explanation			
Enable the N th Com Port of K th station and so that HMI can communicate with the controller of the K th station (N th : denoted in Var1; K th : denoted in Var2)			

* The STATIONON macro command cannot be used when the “Comm. Interrupt XXX times then ignore” box is ticked [Options] → [Communication Setting].



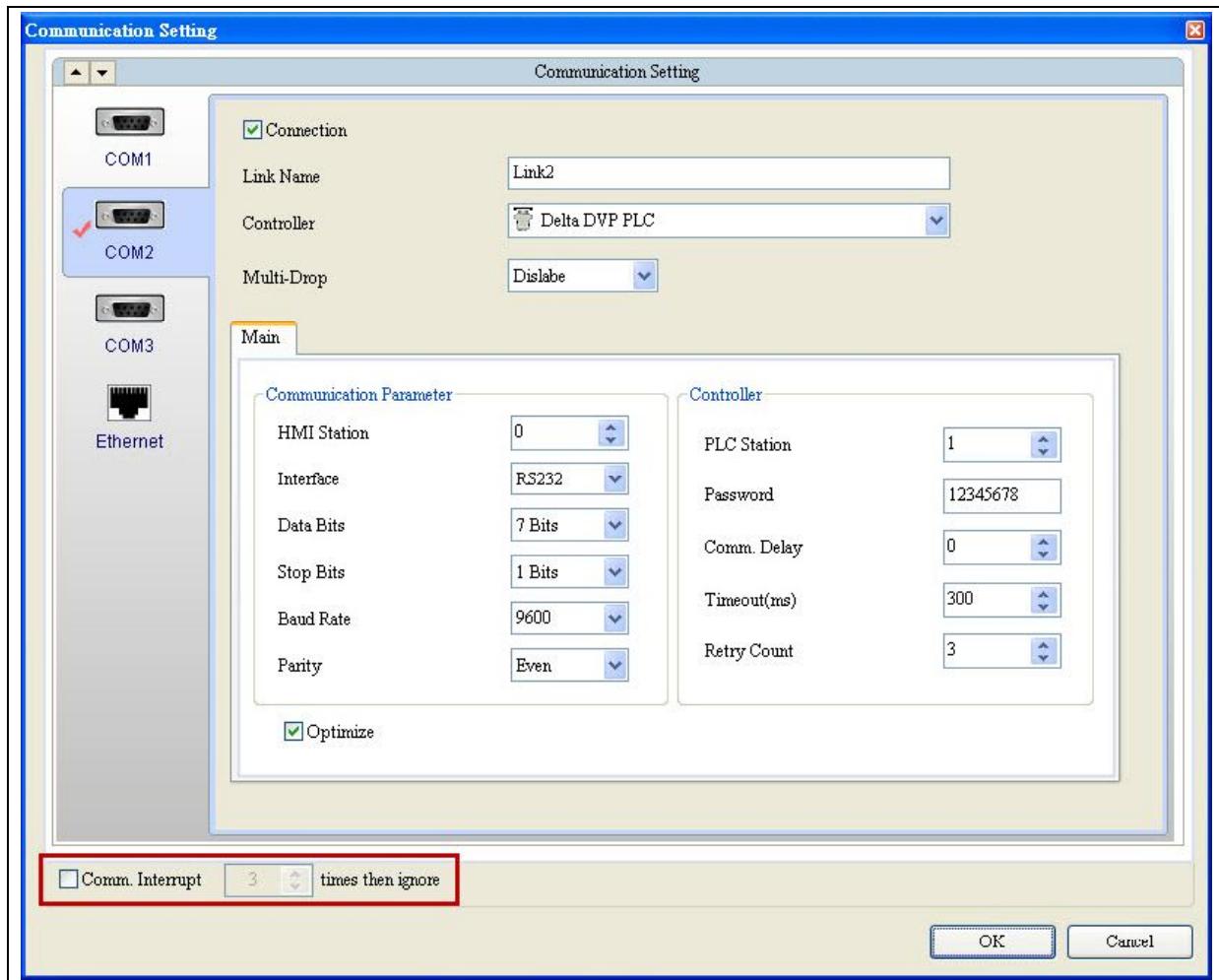
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

Example
<ul style="list-style-type: none"> ➤ Var 1 and Var 2 are constants and open Com 1 of Station #1.  <p>The screenshot shows the Macro Command dialog box. The command field contains <code>STATIONON(0, 1)</code>. Below it, under "Double Word" and "Signed" checkboxes, is the command name <code>STATIONON</code>. A table below shows variables <code>Var1</code> and <code>Var2</code> with values 0 and 1 respectively, and descriptions "COM Port, 0:COM1, 1:COM2, 2:" and "Station".</p>

■ STATIONOFF (Set Station Off)

Expression	What Variables Represent			NOTE	
STATIONOFF(Var1, Var2) (W)	Var 1	COM 1	0	W : Word Disable the N th Com Port of K th station and so that HMI cannot communicate with the controller of the K th station (N th : denoted in Var1; K th : denoted in Var2).	
	Var 1	COM 2	1		
	Var 1	COM 3	2		
	Var 2	Station ID			
	Expression Explanation				

* The STATIONOFF macro command cannot be used when the “Comm. Interrupt XXX times then ignore” box is ticked [Options] → [Communication Setting].



Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎
Var 2	◎		◎

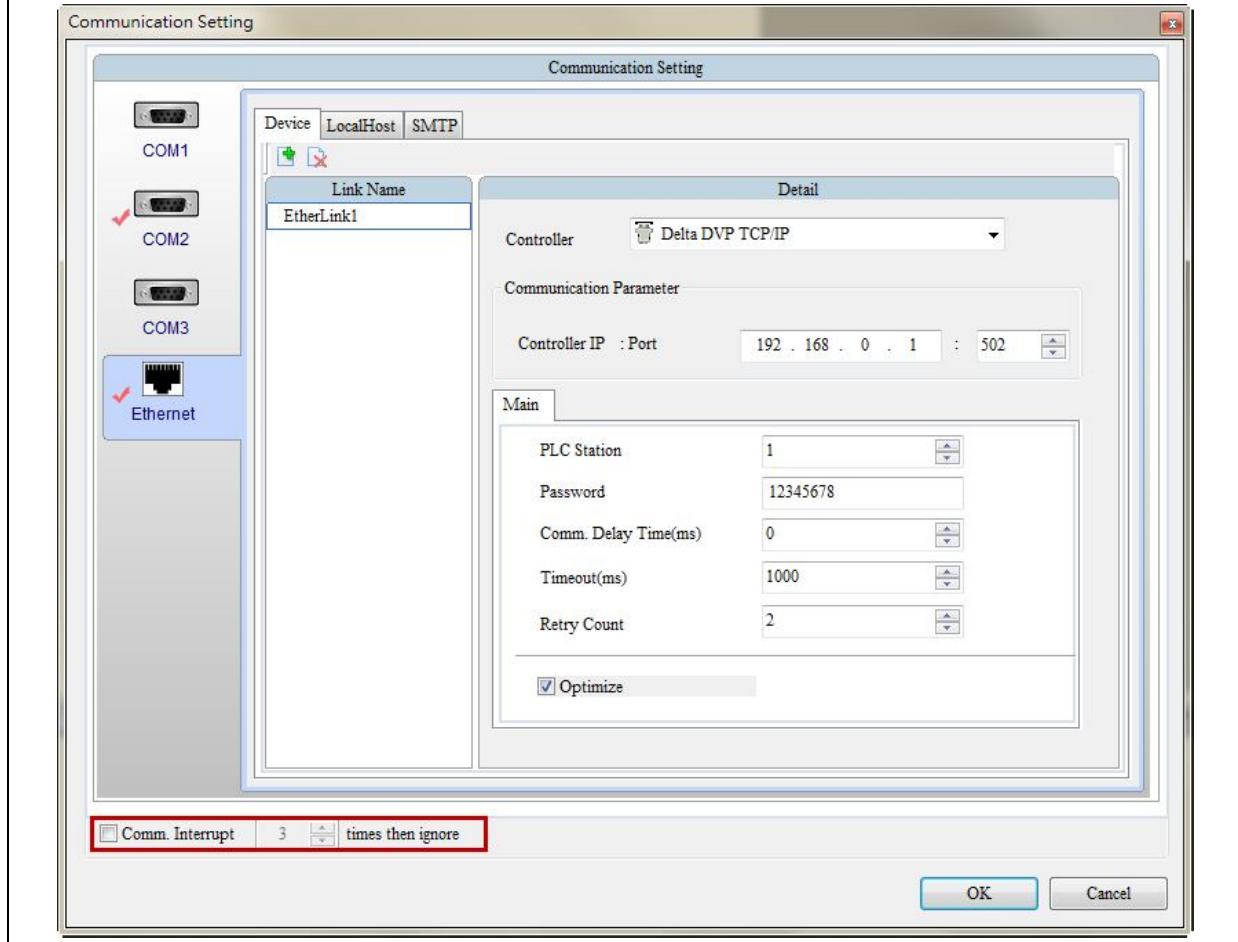
Example
<ul style="list-style-type: none"> ➤ Var 1 and Var 2 are constants and open Com 1 of Station #1. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p style="margin: 0;">Screen_12 [Screen Cycle Macro]</p> <p style="margin: 0; font-size: small;">(A)= Screen_12 [Screen Cycle Macro]</p> <p style="margin: 0; font-size: x-small;">1 STATIONOFF(0, 1)</p> </div>

■ IPON (IP Address Activate)

Expression	What Variables Represent			NOTE				
Var1 = IPON(Var2,Var3,Var4, Var5,Var6) (W)	Var 1	Returned Value		W : Word Var 2 IP1 Var 3 IP2 Var 4 IP3 Var 5 IP4 Var 6 Port				
		Failure	0					
		Success	1					
	Var 2	IP1						
	Var 3	IP2						
	Var 4	IP3						
	Var 5	IP4						
	Var 6	Port						
	Expression Explanation							
	Activate IP Var2.Var3.Var4.Var5 and Port Var6. Then, HMI can communicate with the controller.							

Note 1 : Var6 cannot be used. When it is not using, all Port under this IP will be activated.

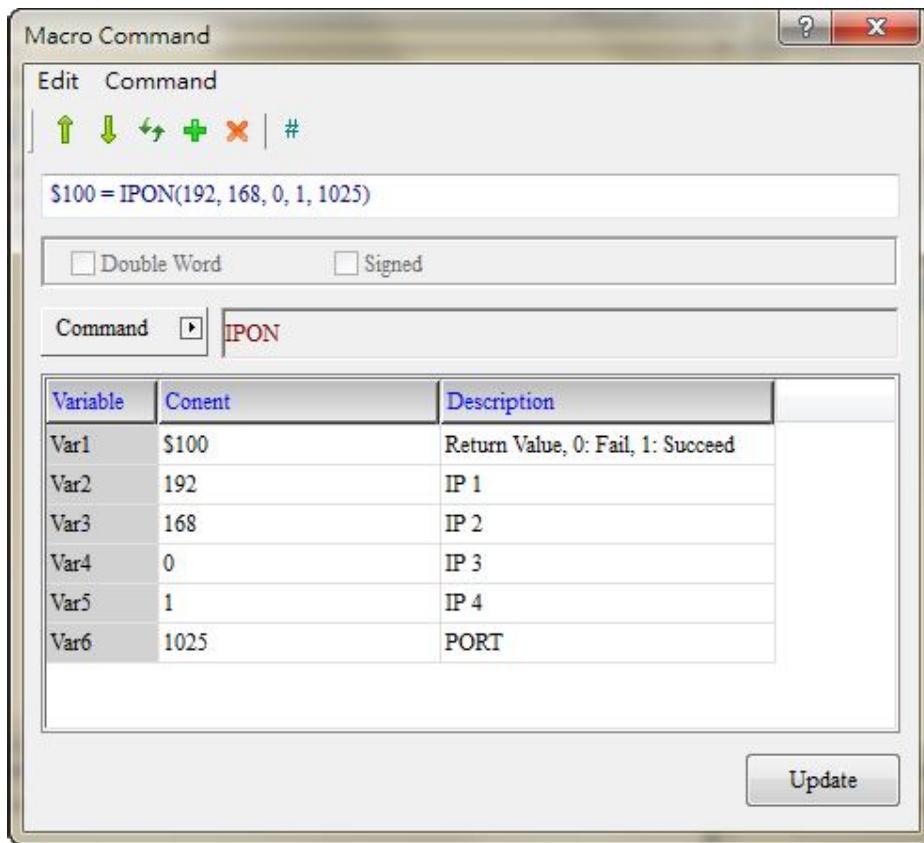
Note 2 : IPON macro cannot be used with Communication Interrupt under 【Option】→【Communication Setting】.



Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	◎
Var 4	◎	◎	◎
Var 5	◎	◎	◎
Var 6	◎	◎	◎

Example

- Var 1 is internal register while Var 2 ~ Var 6 are constants. Activate IP 192.168.0.1 Port : 1025.



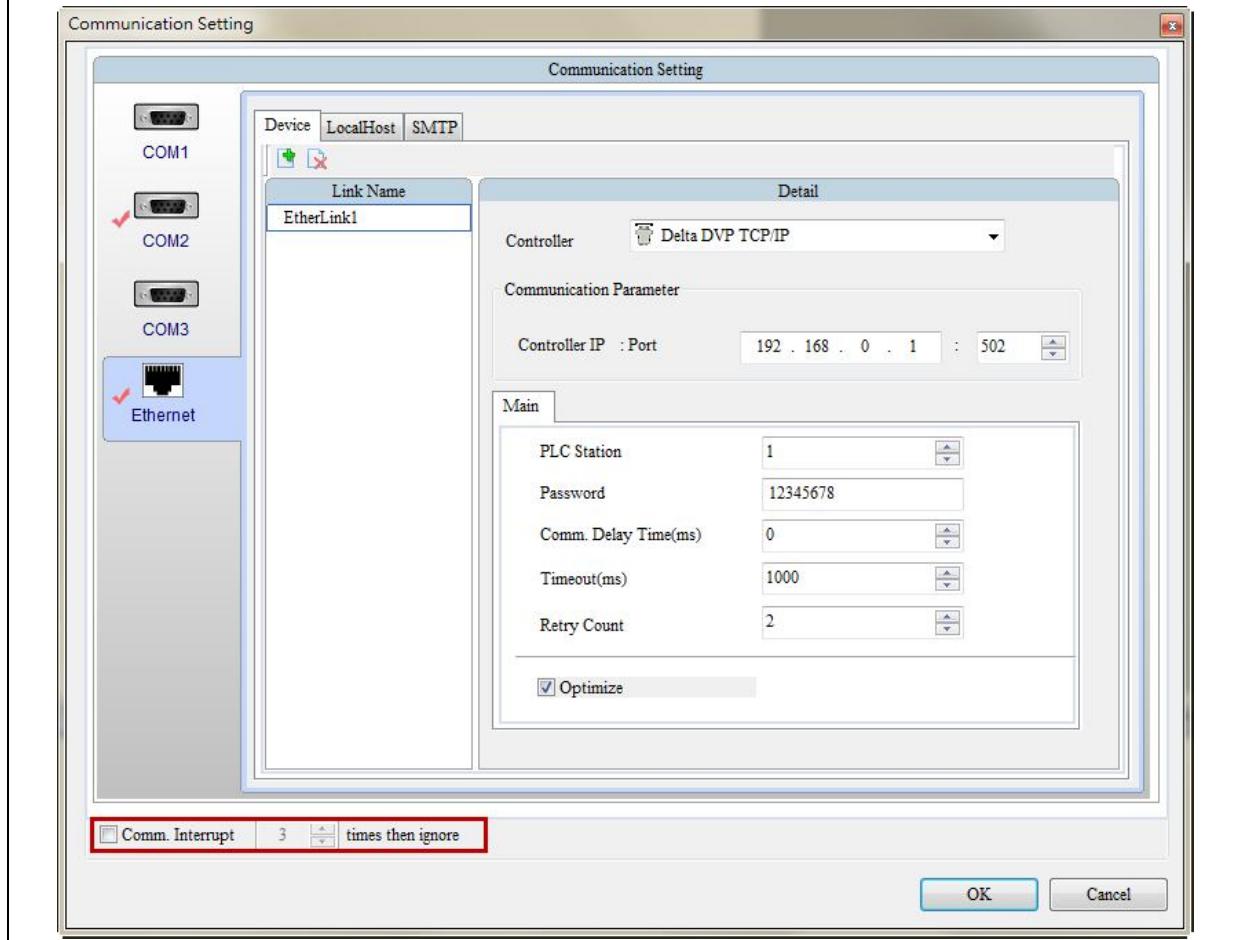
- If not using Var6, then **\$100 = IPON (192, 168, 0, 1)**. To enable all port in IP 192.168.0.1.

■ IPOFF (Disable IP address)

Expression	What Variables Represent			NOTE				
Var1 = IPOFF(Var2,Var3,Var4, Var5,Var6) (W)	Var 1	Returned Value		W : Word				
		Failure	0					
		Success	1					
	Var 2	IP1						
	Var 3	IP2						
	Var 4	IP3						
	Var 5	IP4						
	Var 6	Port						
	Expression Explanation							
	Close IP Var2.Var3.Var4.Var5 and Port Var6. Then, HMI will be unable to communicate with the controller of that station.							

Note 1: Var6 is not necessary. When it is not using, all Ports under the IP will be closed.

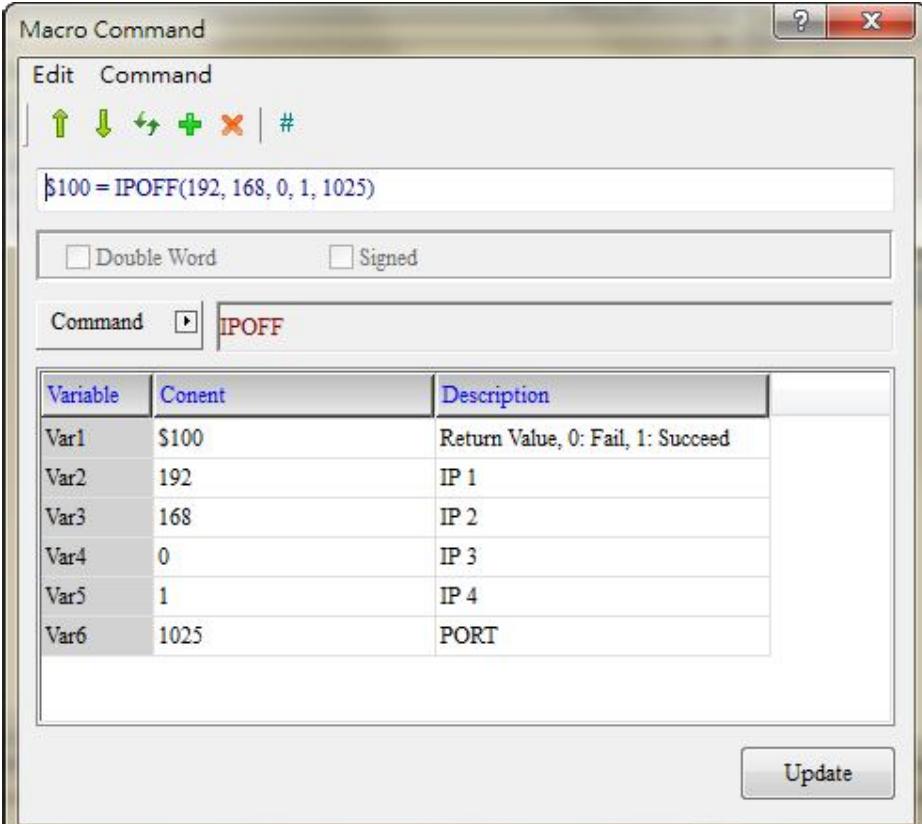
Note 2: IPON macro cannot be used with Communication Interrupt under 【Option】→【Communication Setting】.



Variables	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	◎
Var 4	◎	◎	◎
Var 5	◎	◎	◎
Var 6	◎	◎	◎

Example Description

➤ Var 1 is internal register. Var 2 ~ Var6 are constants. Close IP 192.168.0.1 Port : 1025.



The screenshot shows the 'Macro Command' dialog box. In the 'Edit' tab, the command `$100 = IPOFF(192, 168, 0, 1, 1025)` is entered. Below it, there are checkboxes for 'Double Word' and 'Signed'. The 'Command' dropdown is set to 'IPOFF'. A table below lists variables and their values:

Variable	Content	Description
Var1	\$100	Return Value, 0: Fail, 1: Succeed
Var2	192	IP 1
Var3	168	IP 2
Var4	0	IP 3
Var5	1	IP 4
Var6	1025	PORT

Update

➤ If not use Var6, `$100 = IPOFF(192, 168, 0, 1)`, close all ports in IP 192.168.0.1.

24-3-9 Drawing

DMI supports a list of drawing commands, including Rectangle, Line, Point and Circle for users to draw different graphics. They are detailed below.

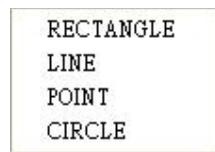


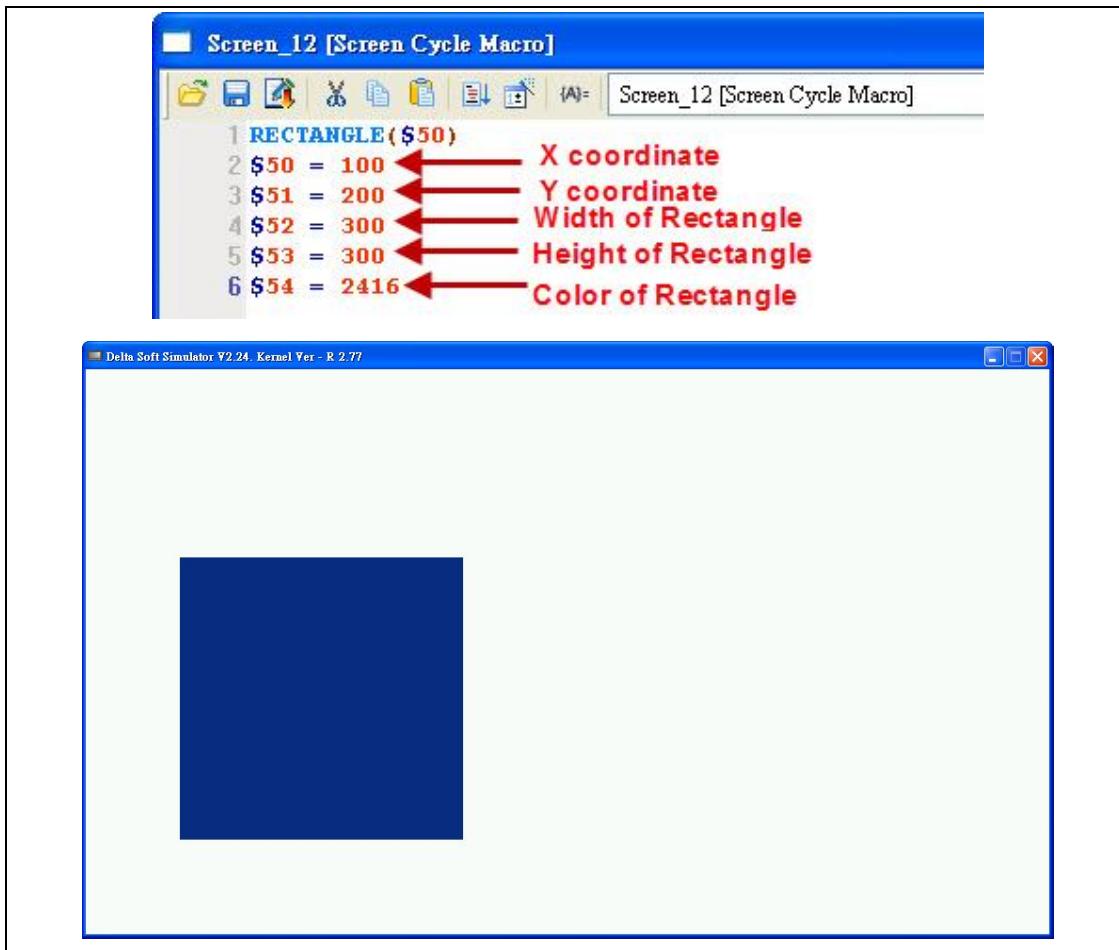
Figure 24-3-9-1 Drawing

■ RECTANGLE (Draw Rectangle)

Expression	What Variables Represent		NOTE
RECTANGLE(Var1) (W)	Var 1	upper-left X-coordinate	W : Word Continuous addresses to draw a rectangle.
	Var 1 + 1	upper-left Y-coordinate	
	Var 1 + 2	width of the rectangle	
	Var 1 + 3	height of the rectangle	
	Var 1 + 4	color of the rectangle	
	Expression Explanation		

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example						
<ul style="list-style-type: none"> ➤ Var 1 is an internal memory address. <div style="border: 1px solid black; padding: 10px; margin-top: 10px;"> <p>Macro Command</p> <p>Edit Command</p> <p>RECTANGLE(\$50)</p> <p><input type="checkbox"/> Double Word <input type="checkbox"/> Signed</p> <p>Command: RECTANGLE</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> <tr> <td>Var1</td> <td>\$50</td> <td>(X coordinate, Y coordinate, Width,</td> </tr> </table> </div>	Variables	Contents	Description	Var1	\$50	(X coordinate, Y coordinate, Width,
Variables	Contents	Description				
Var1	\$50	(X coordinate, Y coordinate, Width,				



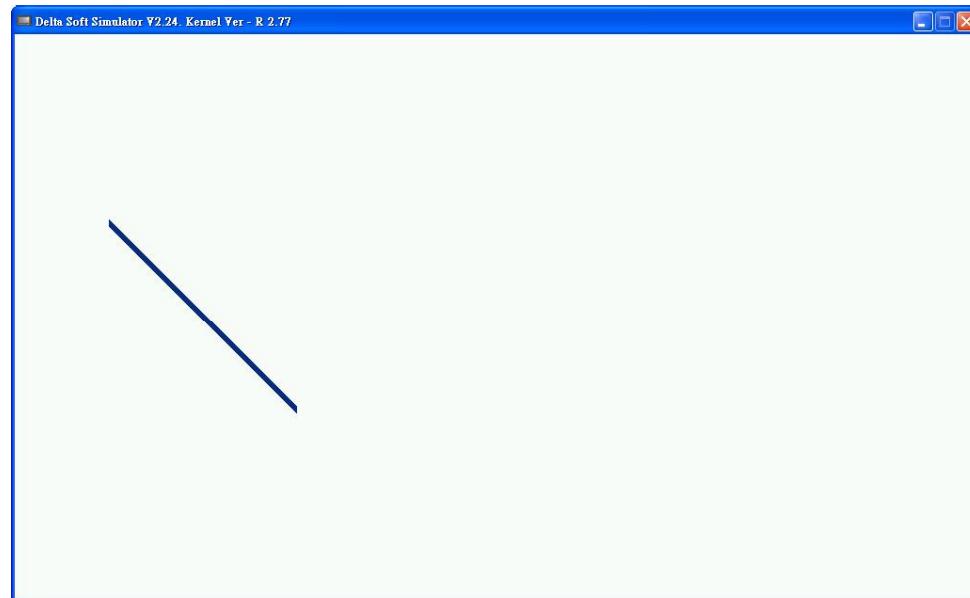
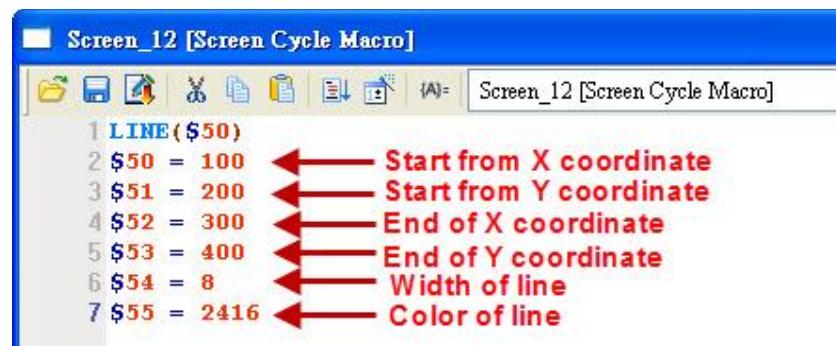
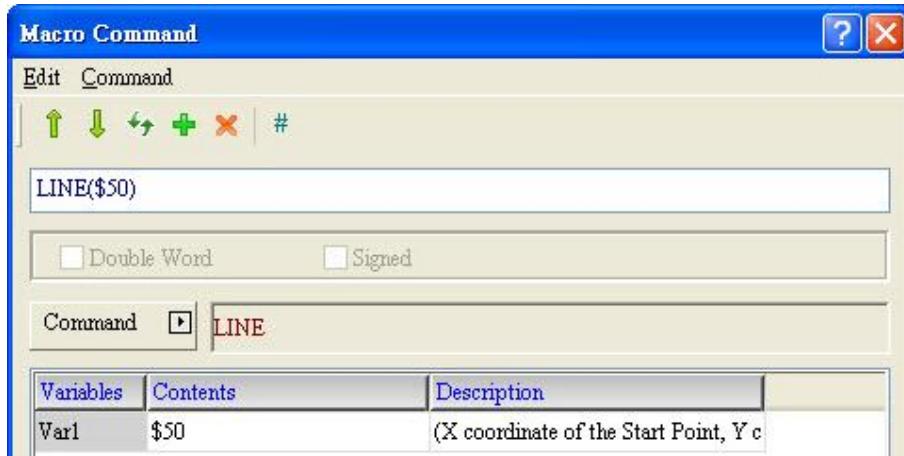
■ LINE (Draw Line)

Expression	What Variables Represent		NOTE	
LINE(Var1) (W)	Var 1	Starting X-coordinate	W : Word Continuous addresses to draw a line.	
	Var 1 + 1	Starting Y-coordinate		
	Var 1 + 2	Ending X-coordinate		
	Var 1 + 3	Ending Y-coordinate		
	Var 1 + 4	Width of the line		
	Var 1 + 5	Color of the line		
	Expression Explanation			
	Continuous addresses to draw a line.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example

- Var 1 is an internal memory address.

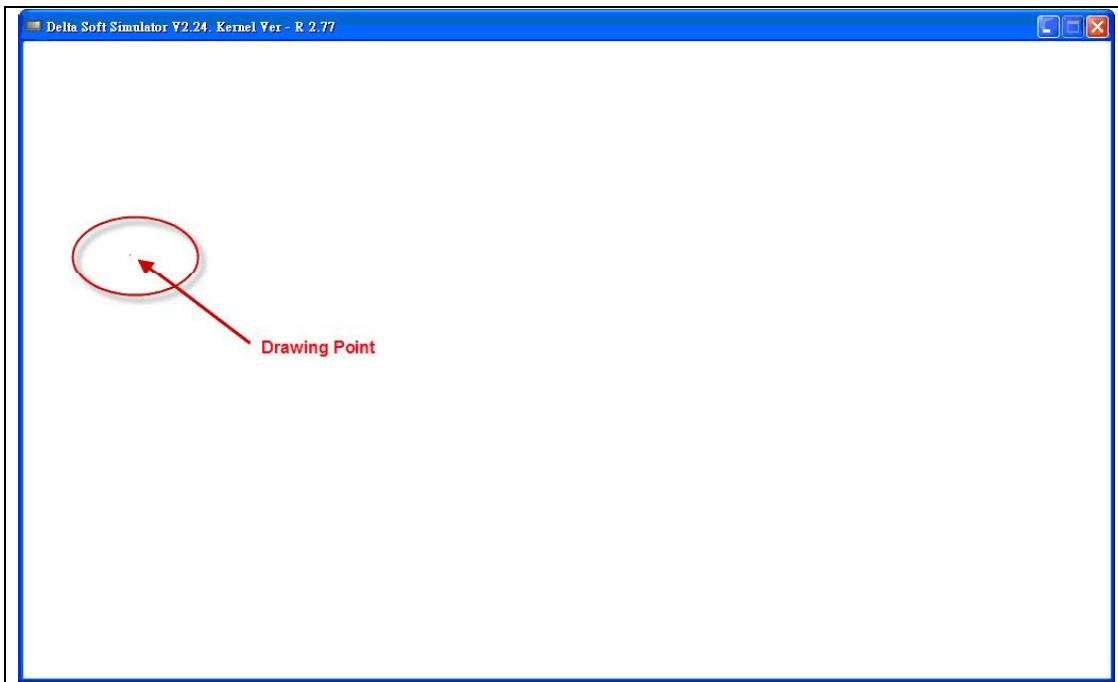


■ POINT (Draw Point)

Expression	What Variables Represent		NOTE	
POINT(Var1) (W)	Var 1	X-coordinate	W : Word Continuous addresses to draw a point.	
	Var 1 + 1	Y-coordinate		
	Var 1 + 2	Color of the point		
	Expression Explanation			
	Continuous addresses to draw a point.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example
<p>➤ Var 1 is an internal memory address.</p> <p>The Macro Command dialog box shows the command <code>POINT(\$50)</code>. The <code>Command</code> field is set to <code>POINT</code>. The <code>Variables</code> table shows <code>Var1</code> assigned to <code>\$50</code>, with a description of <code>(X coordinate, Y coordinate, Color)</code>.</p> <p>The Screen_12 [Screen Cycle Macro] window displays the following macro code:</p> <pre> 1 POINT(\$50) 2 \$50 = 100 ← X coordinate 3 \$51 = 200 ← Y coordinate 4 \$52 = 63488 ← Color of Point </pre> <p>Red arrows point from the labels X coordinate, Y coordinate, and Color of Point to the respective variable assignments in the code.</p>

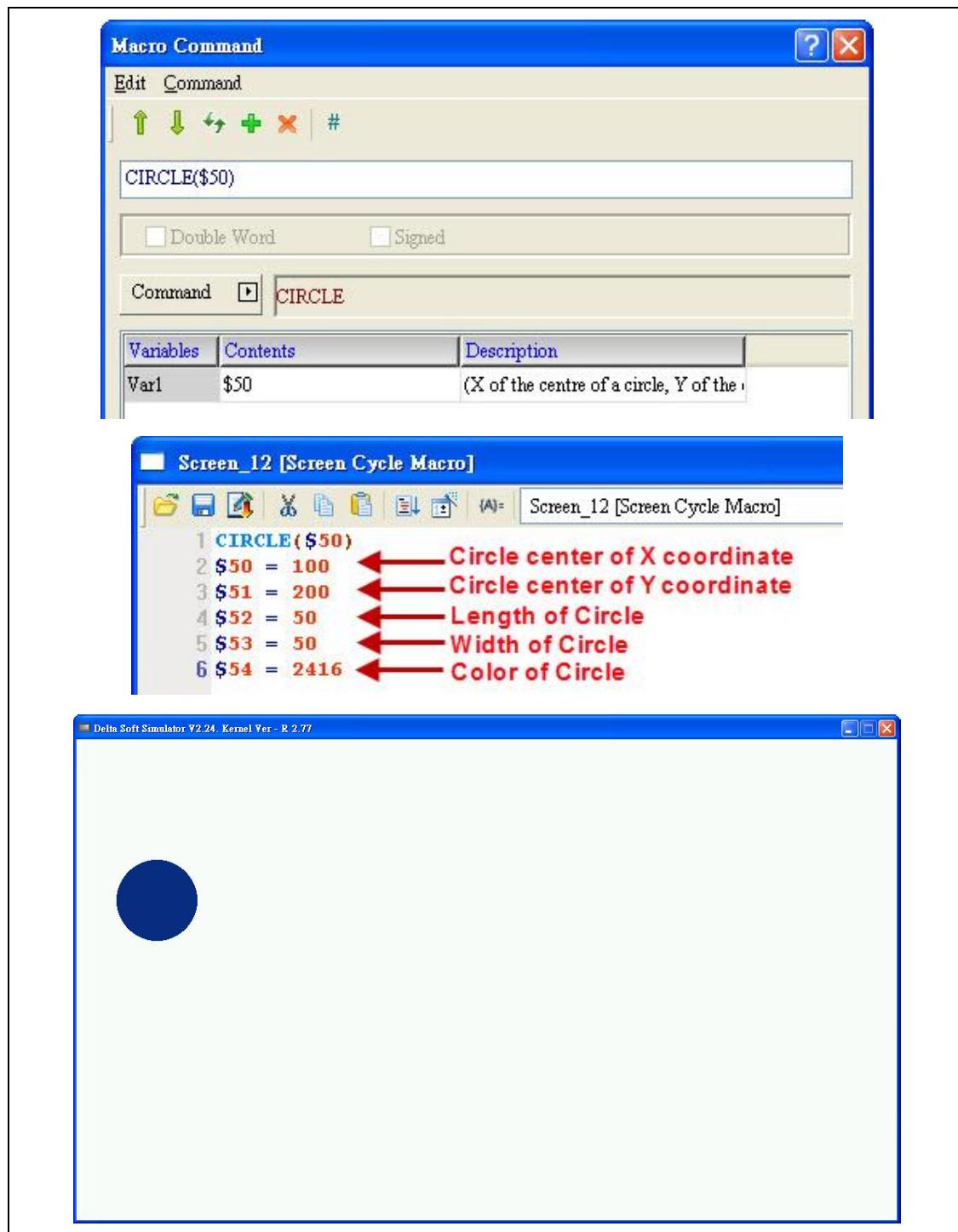


■ CIRCLE (Draw Ellipse)

Expression	What Variables Represent		NOTE	
CIRCLE(Var1) (W)	Var 1	X-coordinate of the center of the ellipse	W : Word Var 1 + 1 Var 1 + 2 Var 1 + 3 Var 1 + 4 Expression Explanation Continuous addresses to draw a point.	
	Var 1 + 1	Y-coordinate of the center of the ellipse		
	Var 1 + 2	length of the ellipse		
	Var 1 + 3	width of the ellipse		
	Var 1 + 4	color of the ellipse		
	Expression Explanation			
	Continuous addresses to draw a point.			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example
➤ Var 1 is an internal memory address.



24-3-10 File Access

The related commands of File Access include FileSlotRead, FileSlotWrite, FileSlotRemove, FileSlotGetLength, FileSlotExport and FileSlotImport. Please refer to the followings for further descriptions.

FileSlotRead
FileSlotWrite
FileSlotRemove
FileSlotGetLength
FileSlotExport
FileSlotImport

■ FileSlotRead (Read the File)

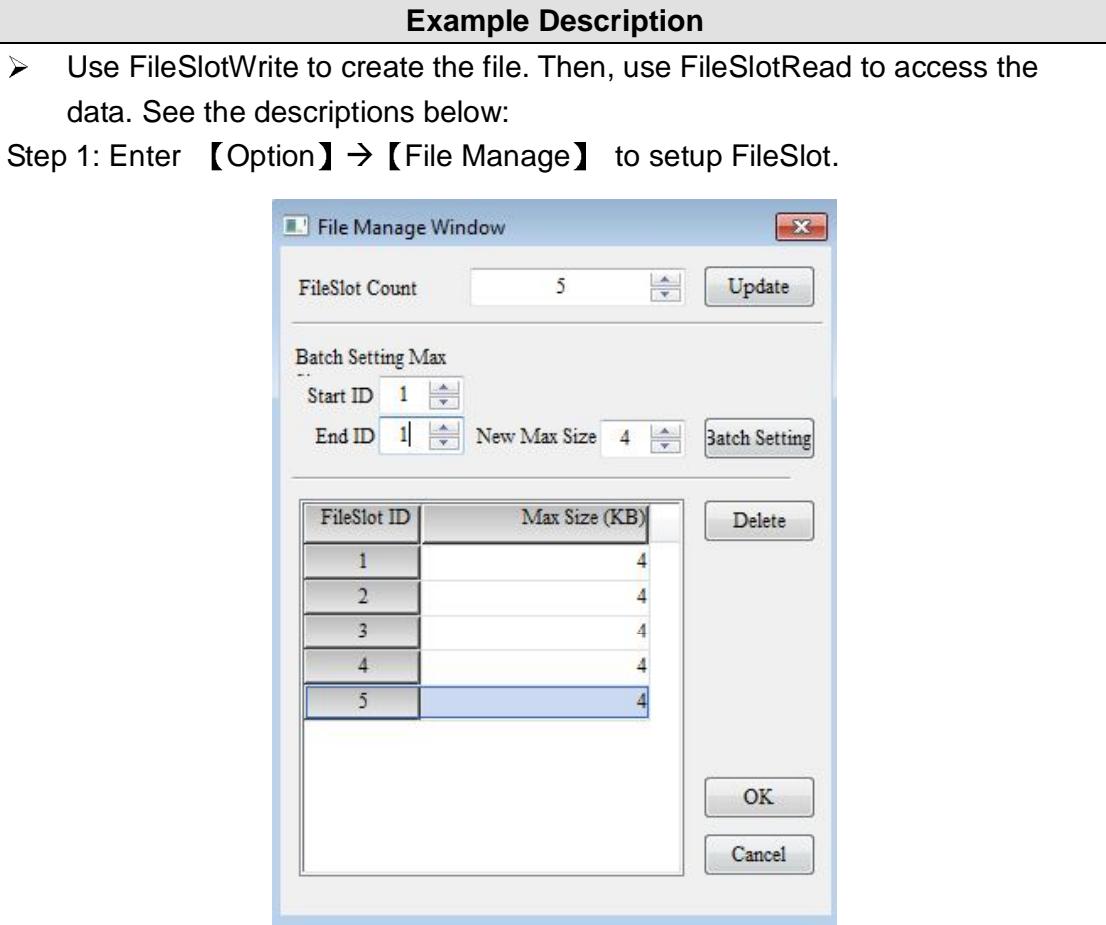
Expression	What Variables Represent			NOTE				
Var1 = FileSlotRead (Var2, Var3, Var4, Var5) (W)	Var 1	Returned Value		W : Word				
		Failure	0					
		Success	1					
	Var 2	FileSlot ID						
	Var 3	Target Address						
	Var 4	Start address of FileSlot content (DW)						
	Var 5	Word data length						
Expression Explanation								
Read FileSlot content of Var 2, whose starting address is specified by Var 4 and data length specified by Var 5 to the target address which is specified by Var 3. Then, return the value to Var 1.								
Note: If the specified FileSlot file does not exist, please use FileSlotWrite command to create the file first.								

Variables	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	
Var 4	◎	◎	◎
Var 5	◎	◎	◎

■ FileSlotWrite (Write the File)

Expression	What Variables Represent			NOTE				
Var1 = FileSlotWrite (Var2, Var3, Var4, Var5) (W)	Var 1	Returned Value		W : Word				
		Failure	0					
		Success	1					
	Var 2	FileSlot ID						
	Var 3	Source Address						
	Var 4	Start position of FileSlot content (DW)						
	Var 5	Word data length						
	Expression Explanation							
	Read Var5 Word starting from Var3 and regard Var4 as start address to write FileSlot of Var2. Then, return the result to Var1.							

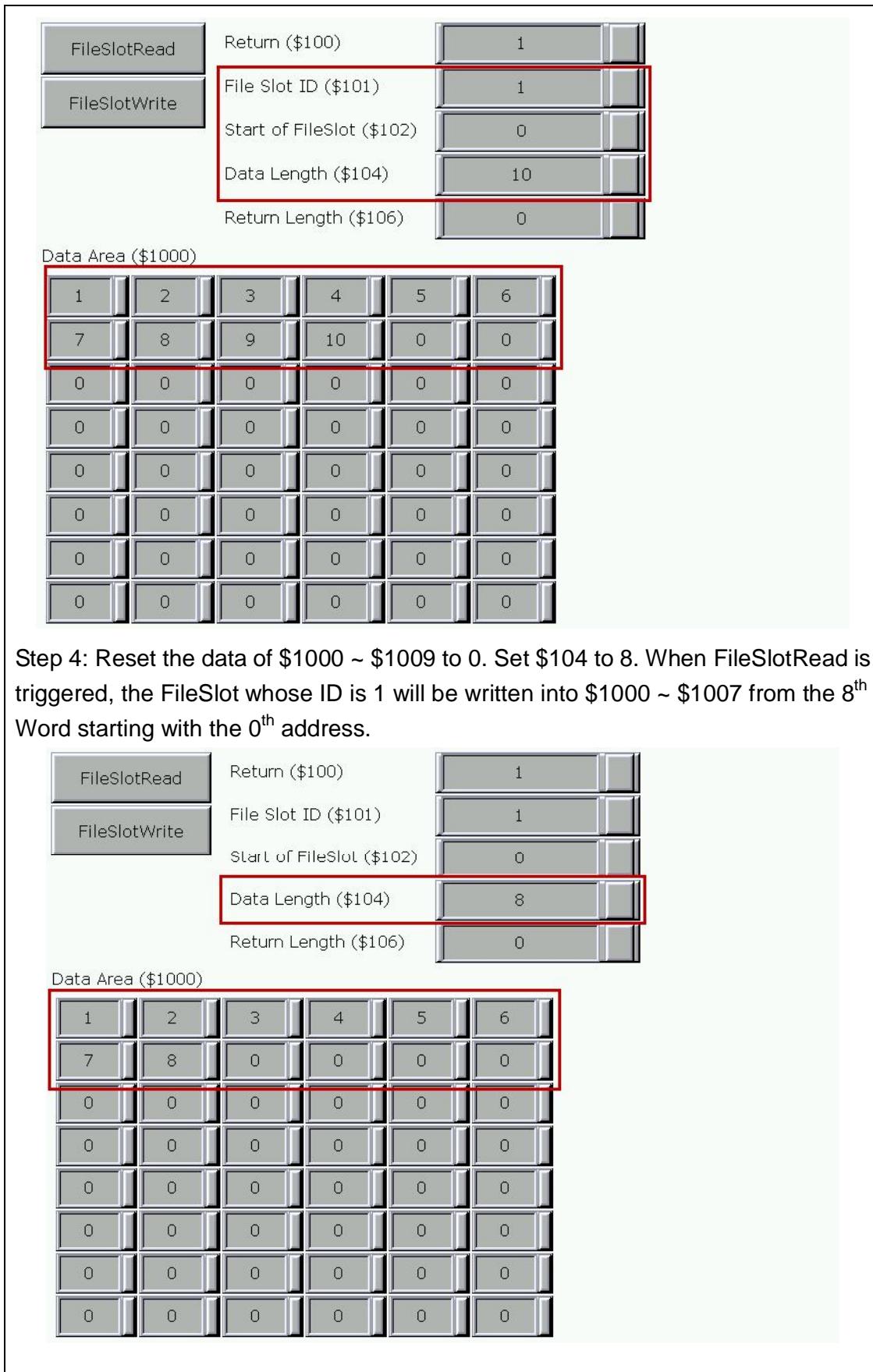
Variables	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	
Var 4	◎	◎	◎
Var 5	◎	◎	◎



Step 2: Create two momentary buttons and compile ON macro.



Step 3: Compile the screen and download it to HMI. Regard \$1000 as start address to edit 10 Words. Set \$101 to 1, \$102 to 0 and \$104 to 10. When FileSlotWrite is triggered, data of \$1000 ~ \$1009 will be written into the FileSlot whose ID is 1 and start to write from address 0.



■ FileSlotRemove (Remove the File)

Expression	What Variables Represent			NOTE	
Var1 = FileSlotRemove (Var2) (W)	Var 1	Returned Value		W : Word	
		Failure	0		
		Success	1		
	Var 2	FileSlot ID			
	Expression Explanation				
Remove FileSlot of VAr2 and return the result to Var1.					

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎

Example Description

- Var 1 is internal memory and Var 2 is the constant. Remove FileSlot whose ID is 1 (Var2) and save the returned value in \$100 (Var1).

Variable	Content	Description
Var1	\$100	Return Value, 0: Fail, 1: Succeed
Var2	1	FileSlot ID

■ FileSlotGetLength (Read the file length)

Expression	What Variables Represent			NOTE	
Var1 = FileSlotGetLength (Var2, Var3) (W)	Var 1	Returned Value		W : Word	
		Failure	0		
		Success	1		
	Var 2	FileSlot ID			
	Var 3	Returned Value of FileSlot Length (DW)			
	Expression Explanation				
Save the FileSlot length of Var 2 in Var 3 and return the result to Var 1.					

Note: The unit of accessing length is BYTE.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	

Example Description		
<ul style="list-style-type: none"> Var 1 and Var 3 are internal memory and Var 2 is constant. Acquire the length of FileSlot whose ID is 1 (Var2) and save it in \$106. Then, save the returned value in \$100 (Var 1). If the FileSlot length is 10 Words, the value returned to \$106 is 20 (BYTE). 		
<p>The screenshot shows the 'Macro Command' dialog box. The command line is set to '\$100 = FileSlotGetLength(1, \$106)'. Below the command line, there are checkboxes for 'Double Word' and 'Signed'. Under the 'Command' tab, 'FileSlotGetLength' is selected. In the bottom table, 'Var1' is assigned '\$100' with the description 'Return Value, 0: Fail, 1: Succeed'. 'Var2' is assigned '1' with the description 'FileSlot ID'. 'Var3' is assigned '\$106' with the description 'Read by BYTE, Address length is D'. At the bottom right of the dialog is a 'Update' button.</p>		

■ FileSlotExport (Export the File)

Expression	What Variables Represent			NOTE				
Var1 = FileSlotExport (Var2, Var3, Var4, Var5) (W)	Var 1	Returned Value		W : Word				
		Failure	0					
		Success	1					
	Var 2	FileSlot ID						
	Var 3	File Export Device	USB Disk					
			SD Card					
	Var 4	Exported File Name						
	Var 5	Length of Exported File Name						
Expression Explanation								
Export Var 2 FileSlot to external storage device, Var3 and its file name is Var 4. Then, return the result to Var1.								

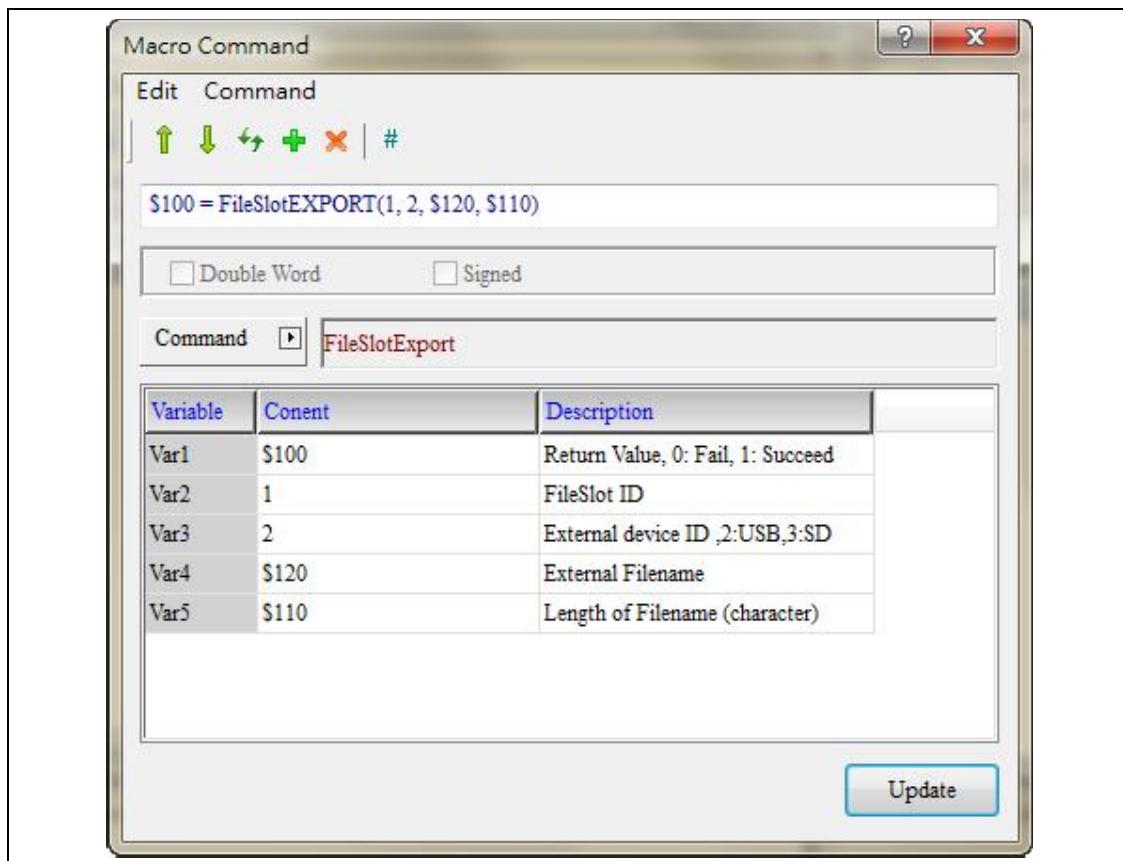
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	◎
Var 4	◎	◎	
Var 5	◎	◎	◎

- | Example Description |
|---|
| <p>➤ Var 1, Var 4 and Var 5 are internal memory while Var 2 and Var3 are constants. Export FileSlot ID 1 (Var 2) to USB Disk (Var 3), the length of file name is 2 (Var 5) and its file name is Slot (Var 4). Then, save the returned value in \$100 (Var 1).</p> |

FileName Length (\$110)



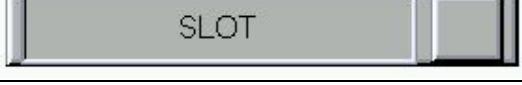
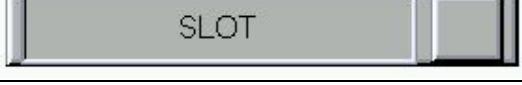
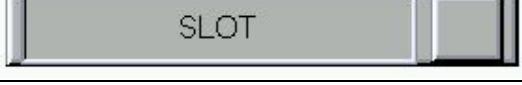
FileName (\$120)

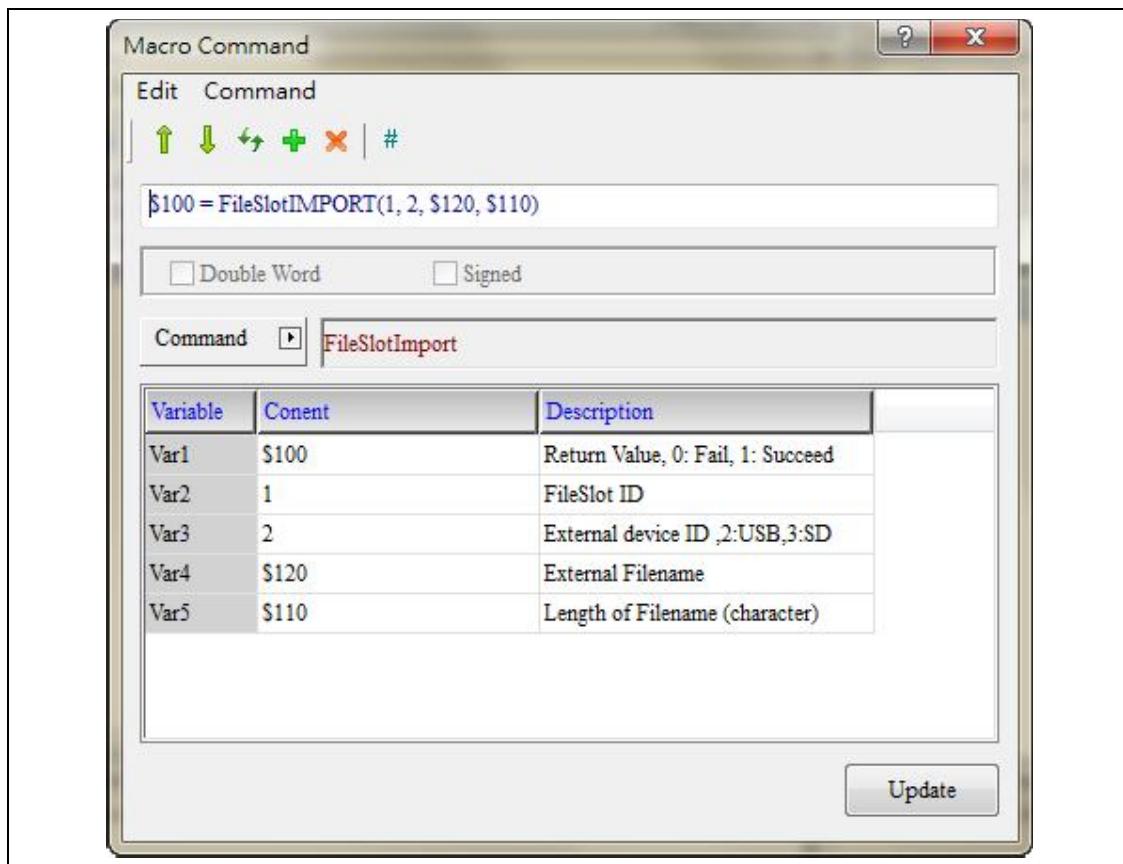


■ FileSlotImport (Import the File)

Expression	What Variables Represent			NOTE			
Var1 = FileSlotImport (Var2, Var3, Var4, Var5) (W)	Var 1	Returned Value		W : Word			
		Failure	0				
		Success	1				
	Var 2	FileSlot ID					
	Var 3	File Import Device	USB Disk	2			
			SD Card	3			
	Var 4	Imported File Name					
Var 5		Length of Imported File Name					
Expression Explanation							
Import the file names Var4 from external storage device Var3 to FileSlot of Var2. Then, return the result to Var1.							

Variables	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎	◎	◎
Var 3	◎	◎	◎
Var 4	◎	◎	
Var 5	◎	◎	◎

Example Description				
<p>➤ Var 1, Var 4 and Var 5 are internal memory while Var 2 and Var3 are constants. Import the file named Slot (Var4) with file name length as 2 (Var5) from USB Disks (Var 3) to FileSlot ID 1 (Var 2). Then, save the returned value in \$100 (Var 1).</p> <table style="margin-left: 20px;"> <tr> <td>FileName Length (\$110)</td> <td></td> </tr> <tr> <td>FileName (\$120)</td> <td></td> </tr> </table>	FileName Length (\$110)		FileName (\$120)	
FileName Length (\$110)				
FileName (\$120)				



24-3-11 Others

Others macro commands include TIME TICK, Comment, Delay, GETSYSTEMTIME、SETSYSTEMTIME, EXPORT, EXRCP, IMRCP, EXENRCP, IMENRCP. Users can acquire system setup time, import and export equations with these commands and they are detailed below.

Time Tick
GETLASTERROR
Comment
Delay
GETSYSTEMTIME
SETSYSTEMTIME
GETHISTORY
EXPORT
EXRCP16
IMRCP16
EXRCP32
IMRCP32
EXENRCP
IMENRCP
EXHISTORY
EXALARM
DISKFORMAT
BMPCAPTURE
PLCDOWNLOAD

Figure 24-3-11-1 Others

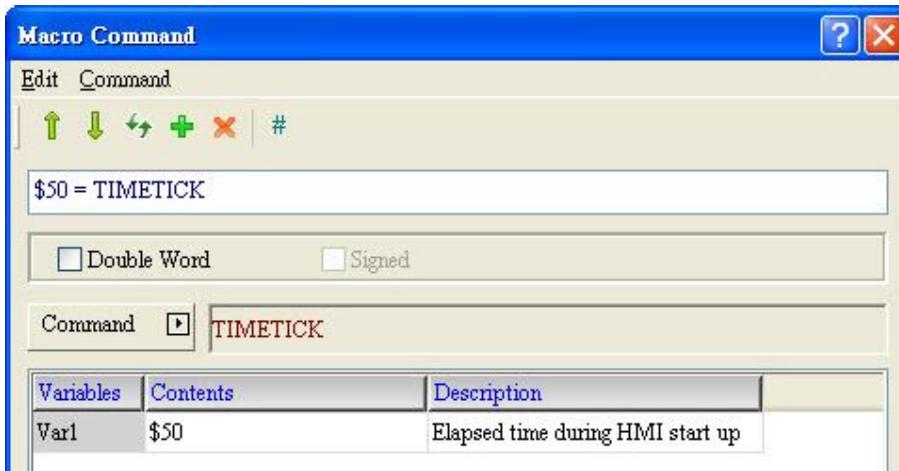
■ Time Tick (Acquire System up duration from System Startup to Present)

Expression	What Variables Represent		NOTE
Var1 = TIMETICK (W) Var1 = TIMETICK (DW)	Var 1	TIMETICK from system startup to present	W : Word DW : Double Word
Expression Explanation			
Calculate system up duration from system startup time to present and save it in Var 1 (in ms).			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example

- Var 1 is an internal memory address and save the system up duration in \$50.



■ GETLASTERROR (Get Last Error Value)

Expression	What Variables Represent		NOTE
Var1 = GETLASTERROR (W) Var1 = GETLASTERROR (DW) Var1 = GETLASTERROR (Signed W) Var1 = GETLASTERROR (Signed DW)	Var 1	Last error value	W : Word DW : Double Word Signed : Signed number
		1 : Successful	
		Negative value: error (for details on the negative value please refer to 24-4 Macro Error Codes)	
		Expression Explanation	
		Acquire the last error value and save the result in Var1.	

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example

- Var 1 is an internal memory address and save the last error value in \$50.



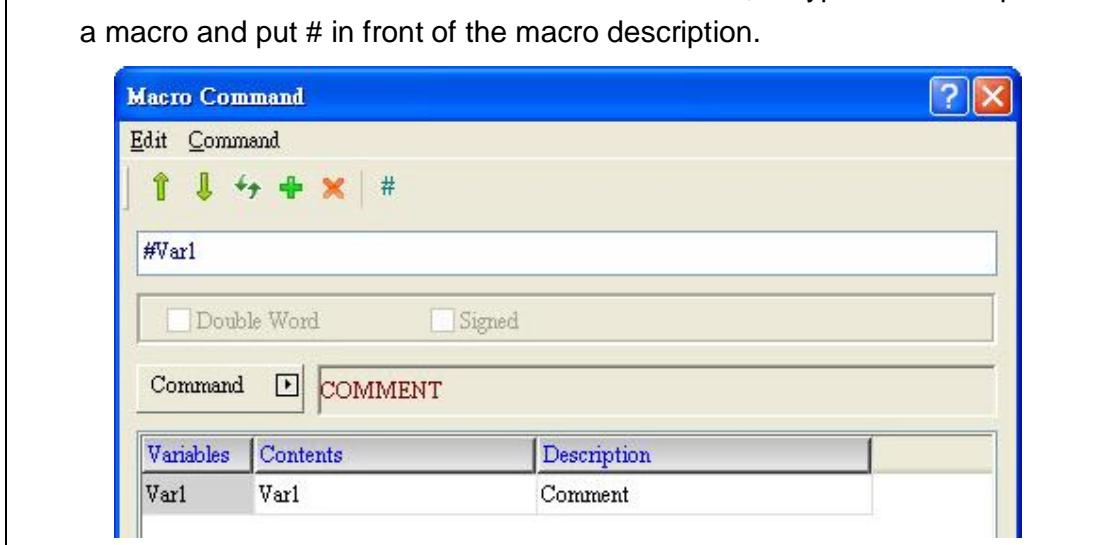
■ COMMENT (Make Comment)

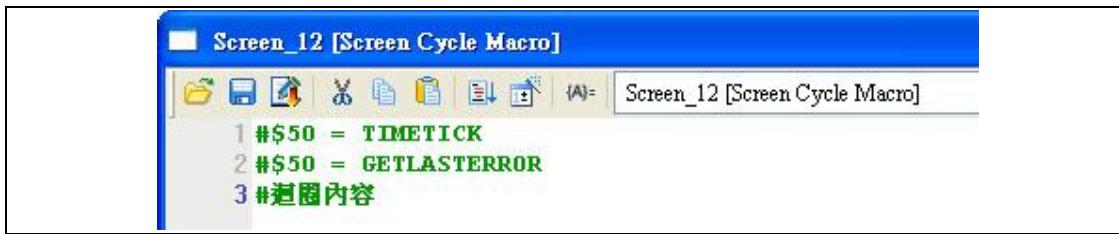
Expression	What Variables Represent		NOTE	
#Var1 (W)	Var 1	Description of the macro	W : Word	
	Expression Explanation			
	Mark Var 1 as a comment			

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example

- Put # in front of a statement to make it a comment; or type in a description for a macro and put # in front of the macro description.





■ Delay (System Delay)

Expression	What Variables Represent		NOTE	
Delay(Var1) (W)	Var 1	Delay time length	W : Word	
	Expression Explanation			
	Set the system to delay for a duration(in ms) specified in Var1 before executing the next command			
<ul style="list-style-type: none"> * Since the HMI is a multiplexer system, a default system delay may occur. If this command is set, then the delay duration will become longer due to multiplexing operations, and the condition that setting the time forward will not happen. * Please note that a long delay duration may cause the HMI to respond slowly. * When executing this command, the HMI will suspend all current operations and resume them after the delay duration is over. 				

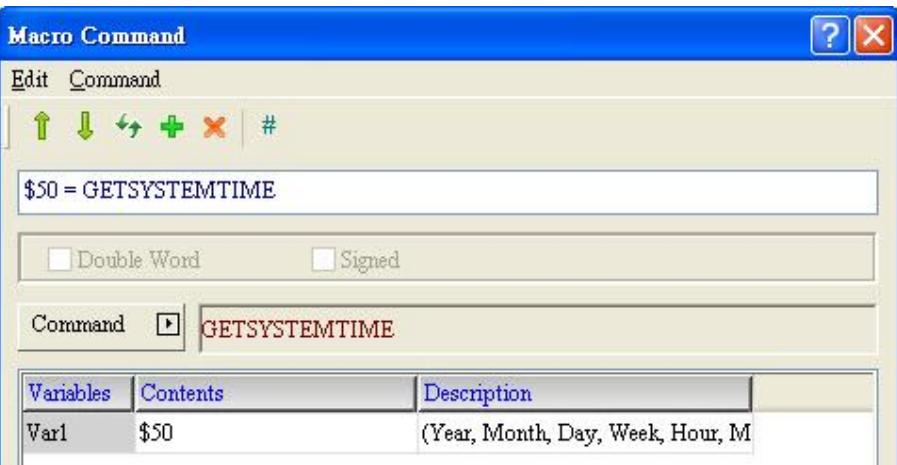
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎

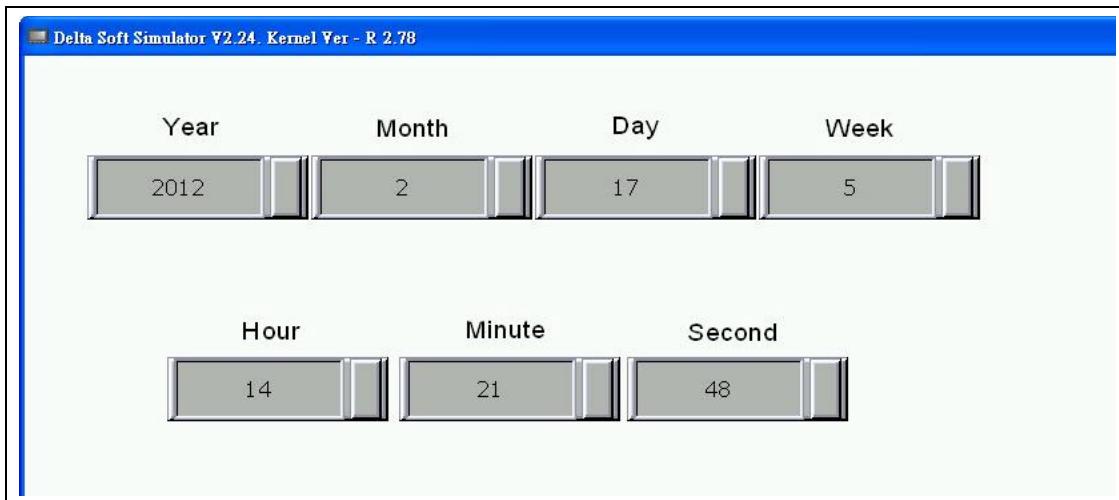
Example						
<p>➤ Var 1 is a constant and set the delay to 500 ms.</p> <p>Macro Command</p> <p>Edit Command</p> <p>Delay(500)</p> <p><input type="checkbox"/> Double Word <input type="checkbox"/> Signed</p> <p>Command: Delay</p> <table border="1"> <thead> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>500</td> <td>Delay Time</td> </tr> </tbody> </table>	Variables	Contents	Description	Var1	500	Delay Time
Variables	Contents	Description				
Var1	500	Delay Time				

■ GETSYSTEMTIME (Acquire System Time)

Expression	What Variables Represent		NOTE
Var1 = GETSYSTEMTIME (W)	Var 1	Year	W : Word
	Var 1 + 1	Month	
	Var 1 + 2	Day	
	Var 1 + 3	Week	
	Var 1 + 4	Hour	
	Var 1 + 5	Minute	
	Var 1 + 6	Second	
	Expression Explanation		
Acquire system time with 7 address (in Words) from Var 1 to Var 7.			

Memory Usage			
Variable	Internal Memory	PLC Register	Constant
Var 1	◎		

Example
<ul style="list-style-type: none"> ➤ Var 1 is an internal memory address and save the current system time to \$50 ~ \$57.  <p>The screenshot shows the 'Macro Command' dialog box. The 'Edit Command' field contains '\$50 = GETSYSTEMTIME'. Below it, there are checkboxes for 'Double Word' and 'Signed'. The 'Command' dropdown is set to 'GETSYSTEMTIME'. At the bottom, a table shows the variables: Var1 is mapped to \$50, with a description '(Year, Month, Day, Week, Hour, M)'.</p>



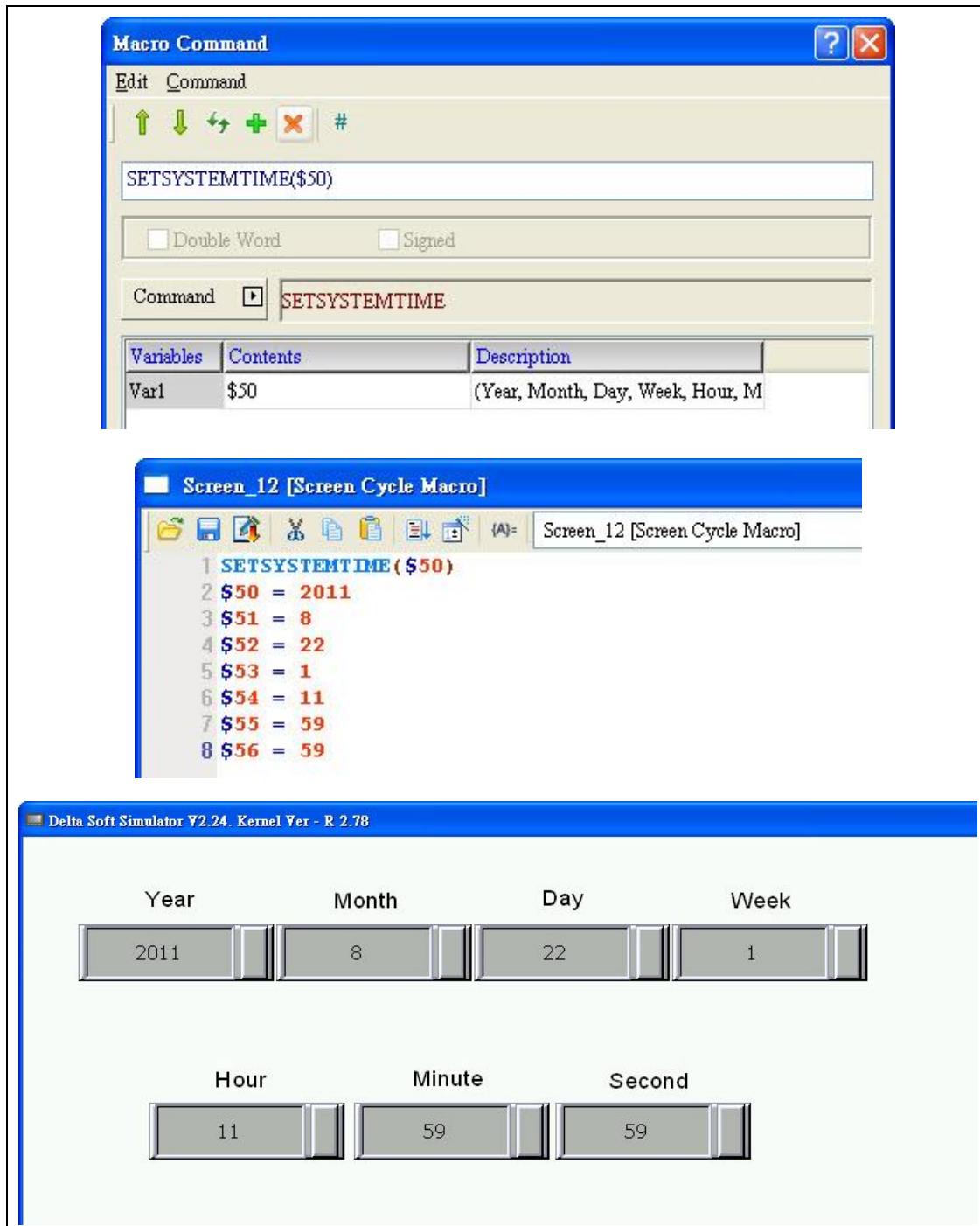
■ SETSYSTEMTIME (Set System Time)

Expression	What Variables Represent		NOTE	
Var1 = SETSYSTEMTIME (W)	Var 1	Year	W : Word	
	Var 1 + 1	Month		
	Var 1 + 2	Day		
	Var 1 + 3	Week		
	Var 1 + 4	Hour		
	Var 1 + 5	Minute		
	Var 1 + 6	Second		
Expression Explanation				
Set system time with 7 address (in Words) from Var 1 to Var 7.				

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		

Example

- Var 1 is an internal memory address. Set and save the current system time to \$50 ~ \$57.

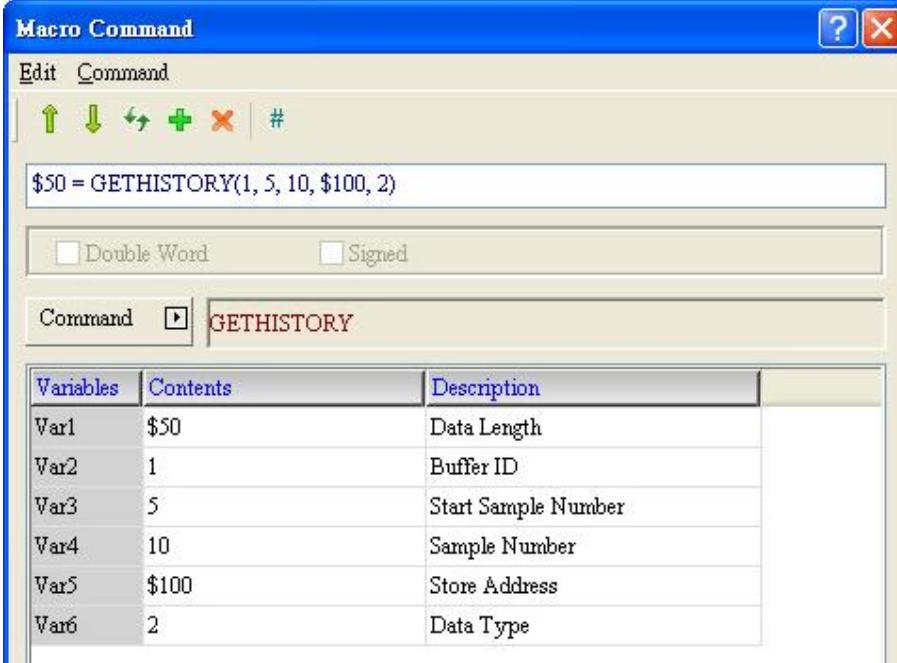


■ GETHISTORY (Acquire Historical Log)

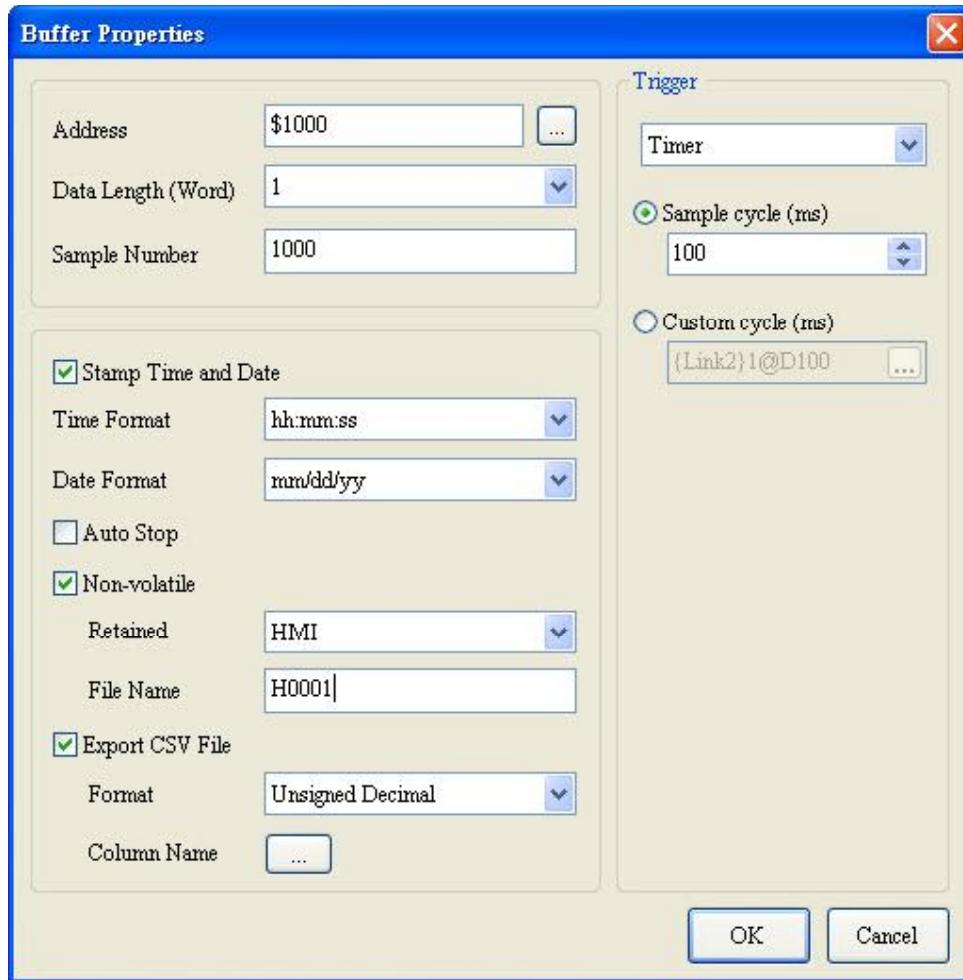
Expression	What Variables Represent			NOTE		
Var1 = GETHISTORY(Var2, Var3, Var4, Var5, Var6) (W)	Var 1	Store data length		W : Word Var 1, Var 3 and Var 4 are Double Word.		
	Var 2	Store ID for history buffer				
	Var 3	Store starting address for sampling				
	Var 4	Store the number of record accessing point				
	Var 5	Store the data storage address				
	Var 6	Data	0			
		Time	1			
		Data and Time	2			
Expression Explanation						
Acquire historical log.						
* it is recommended to set Var 1, Var 3 and Var 4 as Double Word. If using consecutive addresses in Word, then data may be overwritten and result may be affected.						

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		◎
Var 3	◎		◎
Var 4	◎		◎
Var 5	◎	◎	
Var 6	◎		◎

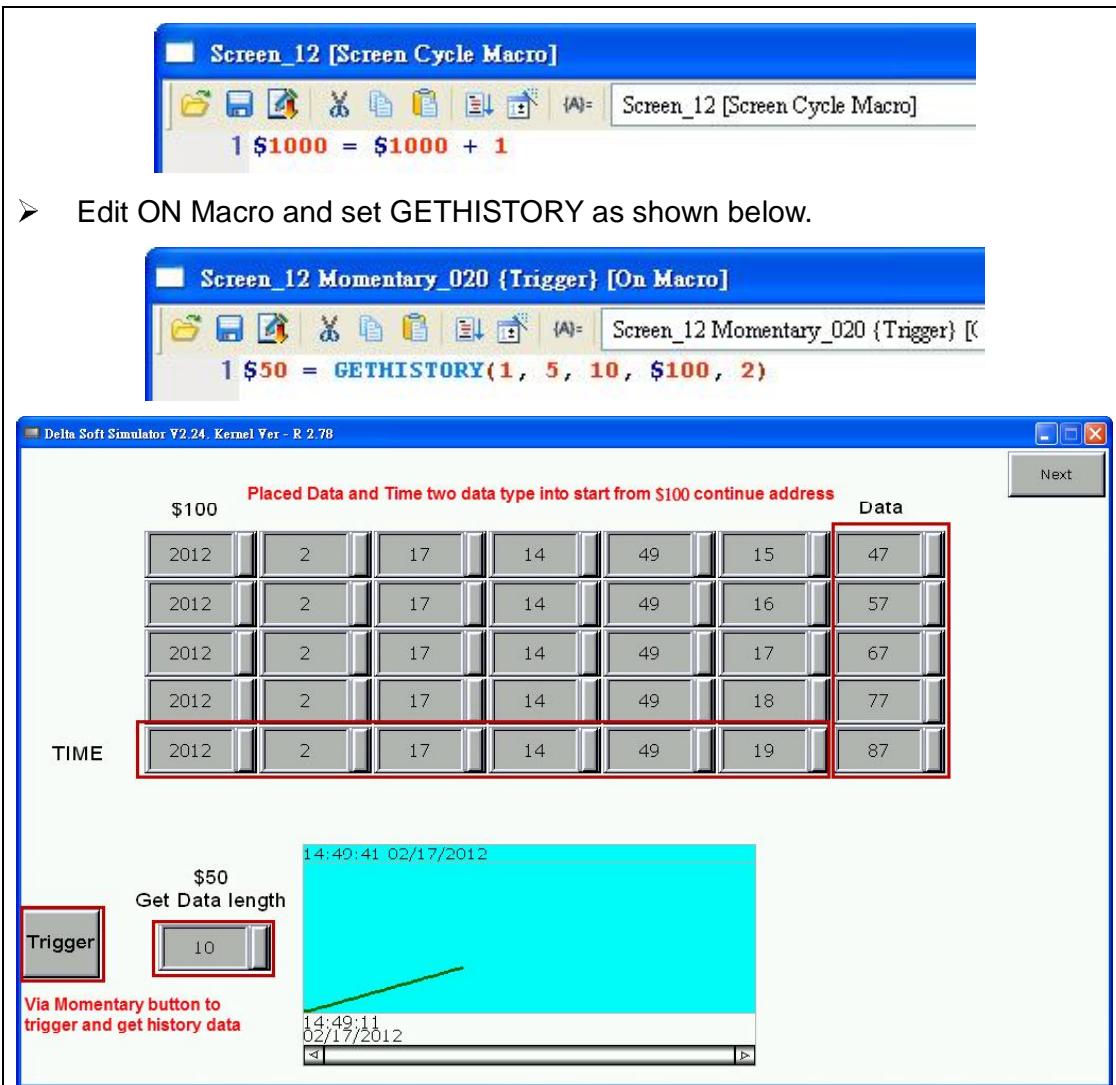
Example
<ul style="list-style-type: none"> ➤ Var 1 and Var 5 are an internal memory addresses, and Var 2, Var3, Var 4 and Var 6 are constants. Set the ID for the history buffer as 1(Var 2), sample 10 records (Var 3) starting from the 5th record (Var 2), Set the data type to 2 (Var 6), including time and data, and save it to consecutive addresses of \$100 (var 5) and save the data length into \$50 (Var 1).



- Set \$1000 as the record accessing address for sampling history buffers.



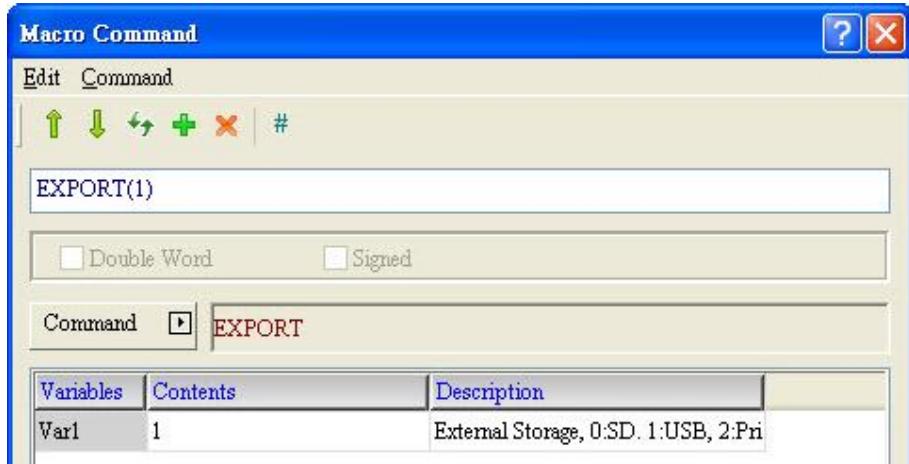
- Edit Screen Cycle Macro to repeatedly adding history records to \$1000.



■ EXPORT (Export Report to an External Device)

Expression	What Variables Represent				NOTE		
EXPORT(Var1) (W)	Var 1	Device to export a report	SD Card	0	W : Word		
			USB Disk	1			
			Printer	2			
	Expression Explanation						
	Delay for a duration specified in Var 1 to resume the next command (in ms).						

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		◎

Example
<ul style="list-style-type: none"> ➤ Var 1 is a constant. Export the report to a USB Disk.  <p>The screenshot shows the Macro Command dialog box. The command is set to EXPORT(1). The variable Var1 is selected and has a value of 1. The description for Var1 is "External Storage, 0:SD, 1:USB, 2:PRI".</p>

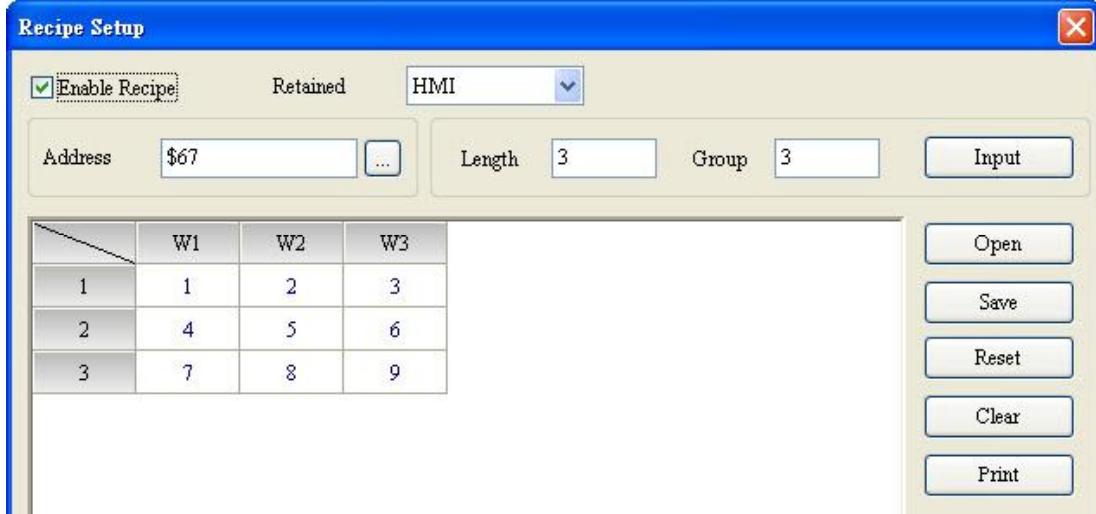
■ EXRCP16/EXRCP32 (Export 16 bit Equation/32 bit Equation)

Expression	What Variables Represent				NOTE						
Var1 = EXRCP16(Var2, Var3) (W) Var1 = EXRCP32(Var2, Var3) (W)	Var 1	Returned Value			W : Word						
		Failure	0								
		Successful	1								
	Var 2	File name of the exported 16 bit equation									
		Var 3	Storage device the equation is exported to	SD Card	2						
	Expression Explanation										
Export and save 16 bit (or 32 bit) equations to Var 3 and save the returned result to Var 1.											
Note: The exported file is saving to root in the storage device.											

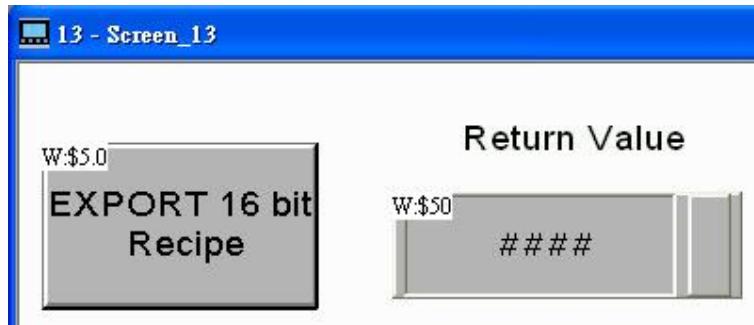
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	
Var 3	◎	◎	◎

Example

- The below example illustrates how to export a 16 bit equation and for a 32 bit equation, the exporting steps are the same.
 - Export a 16 bit equation to a USB Disk and file name is tina.
- Step 1: go to equation setup dialog box to set up an equation [Options] → [Equation].



Step 2: create a maintained button (\$5.0) and variable (\$50).

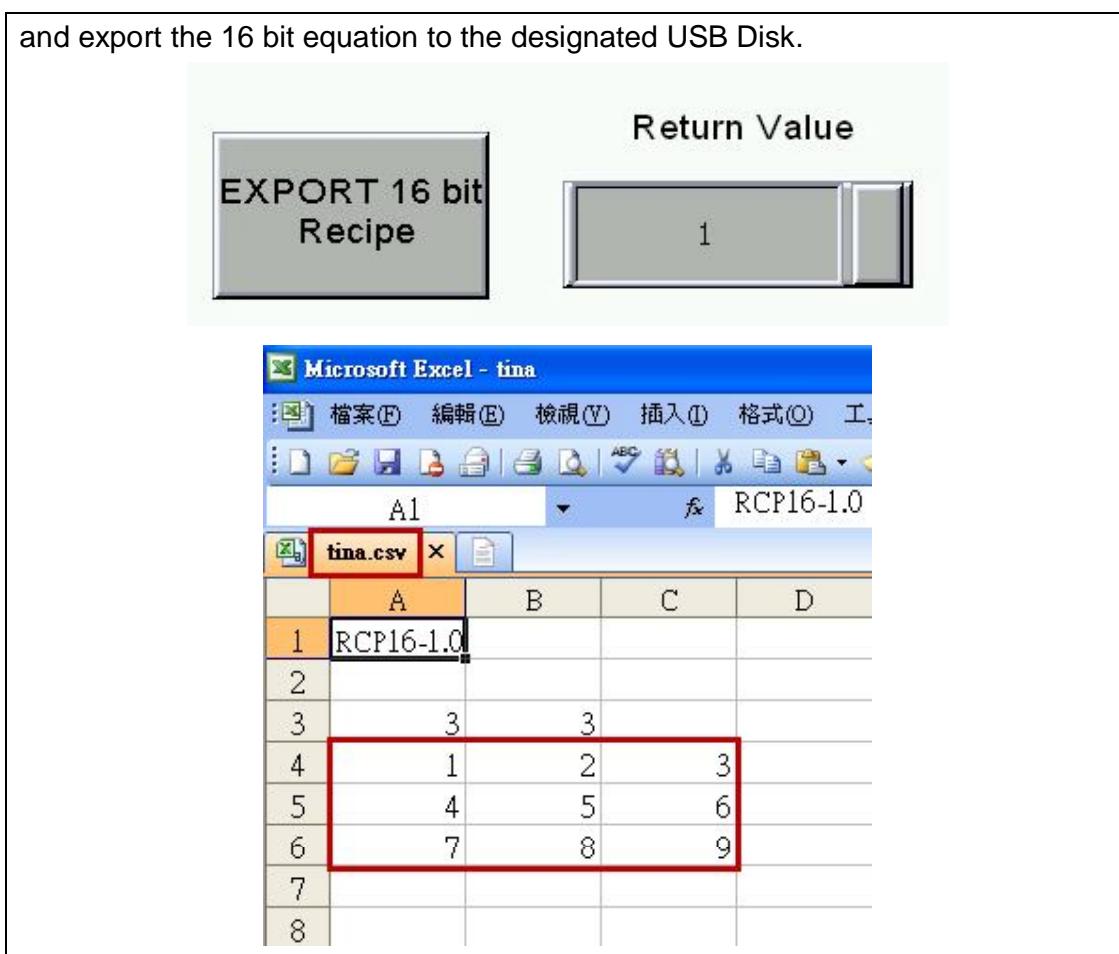


Step 3: enter into the button and create an ON Macro shown below. Put the tina string into \$100 and export the data to a USB Disk, using the command EXRCP16, and name the file tina.



Step 4: edit the elements within the editing area and download the equation to the HMI. Trigger the \$5.0 button and then \$50 will display 1, indicating it is successful

and export the 16 bit equation to the designated USB Disk.



■ IMRCP16/IMRCP32 (Import 16 bit Equation/32 bit Equation)

Expression	What Variables Represent			NOTE	
Var1 = IMRCP16(Var2, Var3) (W) Var1 = IMRCP32(Var2, Var3) (W)	Var 1	Returned Value		W : Word	
		Failure	0		
		Successful	1		
	Var 2	File name of the imported 16 bit equation			
		Storage device the equation is imported from	SD Card		
	Var 3		USB Disk		
	Expression Explanation				
From Var 3 to import 16 bit (or 32 bit) equations and returned result to Var 1.					

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	

Var 2	◎	◎	
Var 3	◎	◎	◎

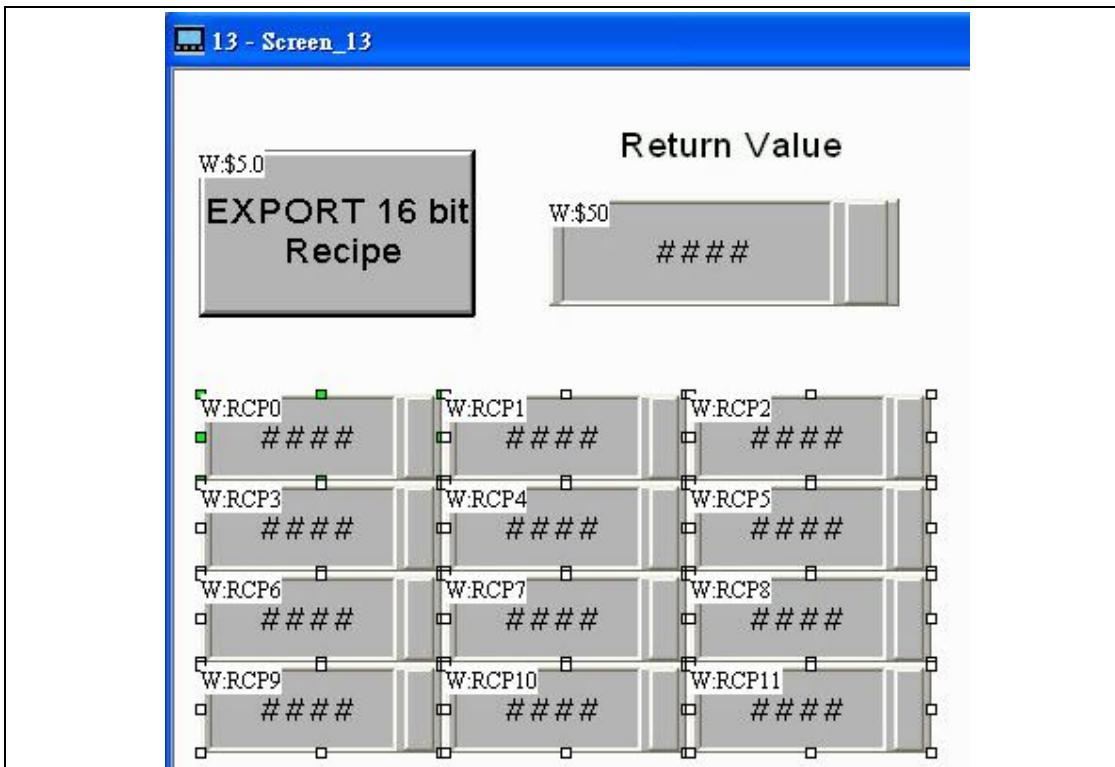
Example

- The below example illustrates how to import a 16 bit equation, and for a 32 bit equation, the importing steps are the same.
- Import a 16 bit equation to a USB Disk and file name is quaque.

Step 1: Modify and save the equation to the USB Disk.

	A	B	C	D	
1	RCP16-1.0				
2					
3	3	3			
4	2	4	6		
5	8	10	12		
6	14	16	18		
7					
8					

Step 2: create a maintained button (\$5.0), variable (\$50), and addresses for storing equations RCP0~RCP11.



Step 3: enter into the button and create an ON Macro shown below. Put the quaqua string into \$100 and import the data to a USB Disk, using the command IMRCP16.

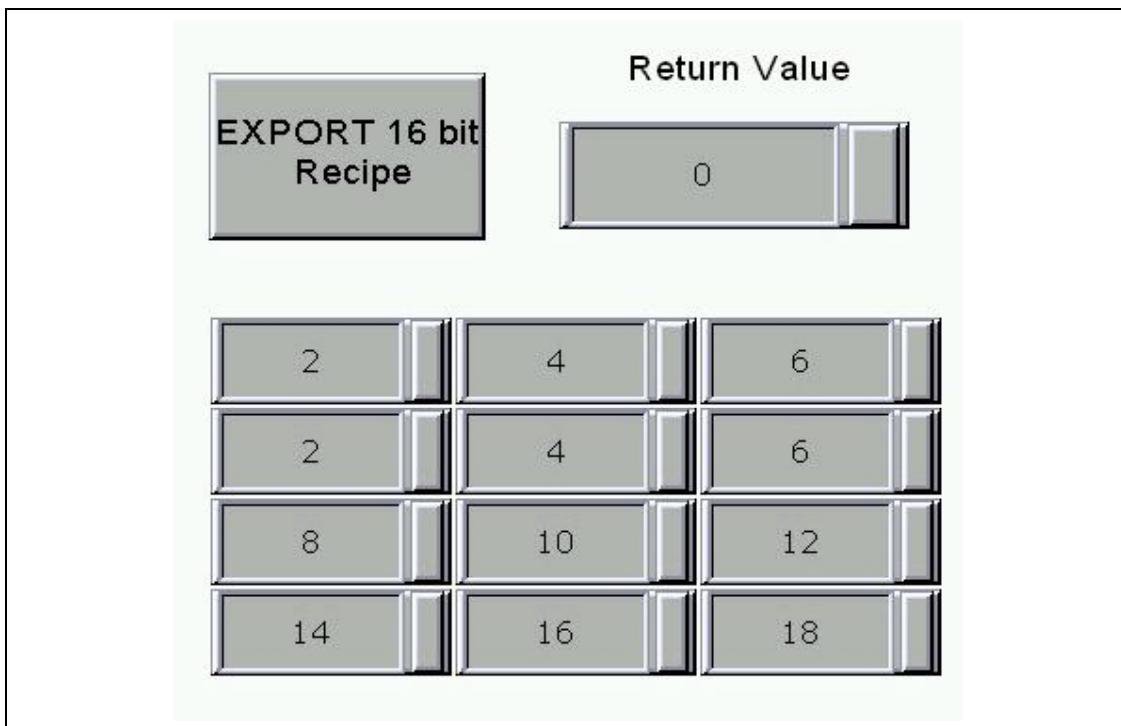
Screen_13 Momentary_002 {EXPORT 16 bit Recipe} [On Macro]

```

1 FILLASC($100, "quaqua")
2 $50 = IMRCP16($100, 2)

```

Step 4: edit the elements within the editing area and download the equation to the HMI. Trigger the \$5.0 button and then \$50 will display 1, indicating it is successful and import the 16 bit equation to the HMI. The equation data will replaced by the quaqua equation just imported.



■ EXENRCP (Export Enhance Recipe Equation)

Expression	What Variables Represent			NOTE
Var1 = EXENRCP(Var2, Var3) (W)	Var 1	Returned Value		W : Word W : Word
		Failure	0	
		Successful	1	
	Var 2	File name of the exported enhance recipe equation		
	Var 3	Storage device the equation is exported to	USB Disk	2
			SD Card	3
Expression Explanation Export and save 16 bit (or 32 bit) equations to Var 3 and save the returned result to Var 1.				

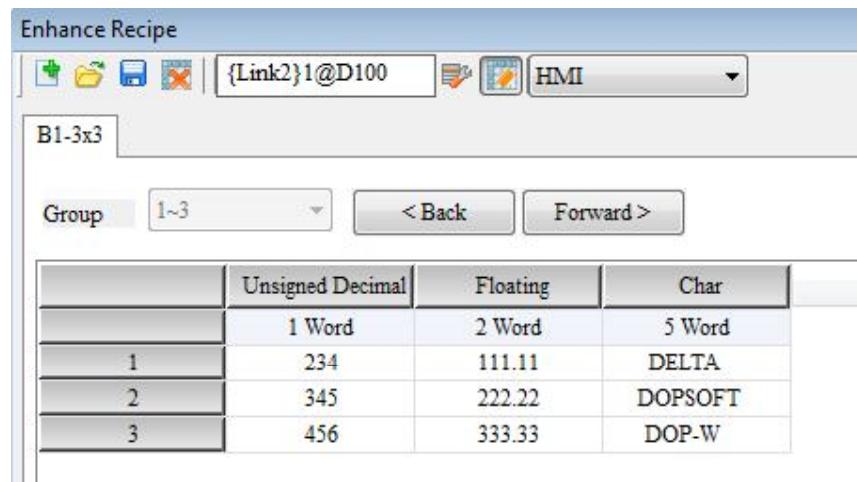
Note: The exported file is saving to root in the storage device.

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	
Var 3	◎	◎	◎

Example

- Export the enhance recipe data to USB Disk and name it as tina.

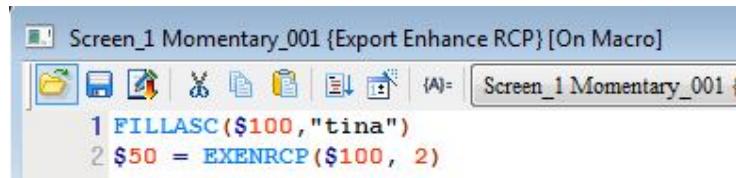
Step 1: Enter 【Option】 → 【Enhance Recipe】 to setup enhance recipe data.



Step 2: Create momentary button (\$5.0) and numeric entry element (\$50).



Step 3: Go to Screen of Momentary Button to edit On macro. Please see as below. Place “tina” to \$100. Then, export the enhance recipe to USB Disk whose file name is tina.



Step 4: Compile the screen and download the recipe data to HMI. When \$5.0 button is triggered, \$50 will display 1, which means the enhance recipe is exported successfully. Then, export the enhance recipe data to USB Disk.



	A	B	C	D	E
1	ENRCP-1.0				
2	3	3			
3	2	1	0	0	
4	5	2	3	2	
5	8	5	0	0	
6	234	111.11	DELTA		
7	345	222.22	DOPSOFT		
8	456	333.33	DOP-W		
9					

Note: Please refer to the followings for the definition of each field.

Format	Definition of DOPSoft
BCD	0
Signed Decimal	1
Unsign Decimal	2
Hexdecimal	3
Binary	4
Floating	5
Char	8

Descriptions of each cell in exported CSV file are shown as below:

A-1 : Version	ENRCP-1.0
A-2 : File Number	3
B-2 : Group Number	3
A-3 : Numeric Format of the 1 st Field	2 (Unsigned Decimal)
B-3 : Numeric Length of the 1 st Field	1 (Word)
C-3 : Integer Digits of the 1 st Field	0
D-3 : Fractional Digits of the 1 st Field	0
A-4 : Numeric Format of the 2 nd Field	5 (Floating)
B-4 : Numeric Length of the 2 nd Field	2 (Word)
C-4 : Integer Digits of the 2 nd Field	3
D-4 : Fractional Digits of the 2 nd Field	2
A-5 : Numeric Format of the 3 rd Field	8 (Char)
B-5 : Numeric Length of the 2 nd Field	5 (Word)
C-5 : Integer Digits of the 3 rd Field	0
D-5 : Fractional Digits of the 3 rd Field	0

■ IMENRCP (Import Enhance Recipe Equation)

Expression	What Variables Represent				NOTE			
Var1 = IMENRCP(Var2, Var3) (W)	Var 1	Returned Value			W : Word			
		Failure	0					
		Successful	1					
	Var 2	File name of the imported enhance recipe equation						
	Var 3	Storage device the equation is imported from	USB Disk	2				
			SD Card	3				
Expression Explanation								
From Var 3 to import enhance recipe equations and returned result to Var 1.								

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	
Var 3	◎	◎	◎

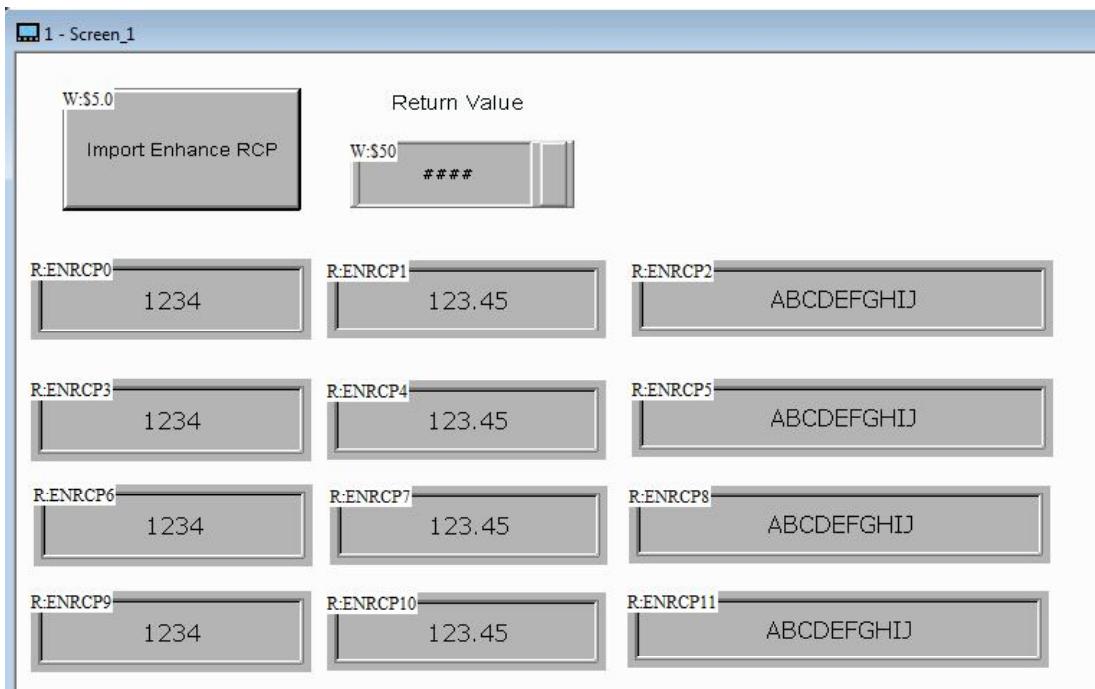
Example

- Import the enhance recipe data from USB Disk to HMI and its file name is quaqua.

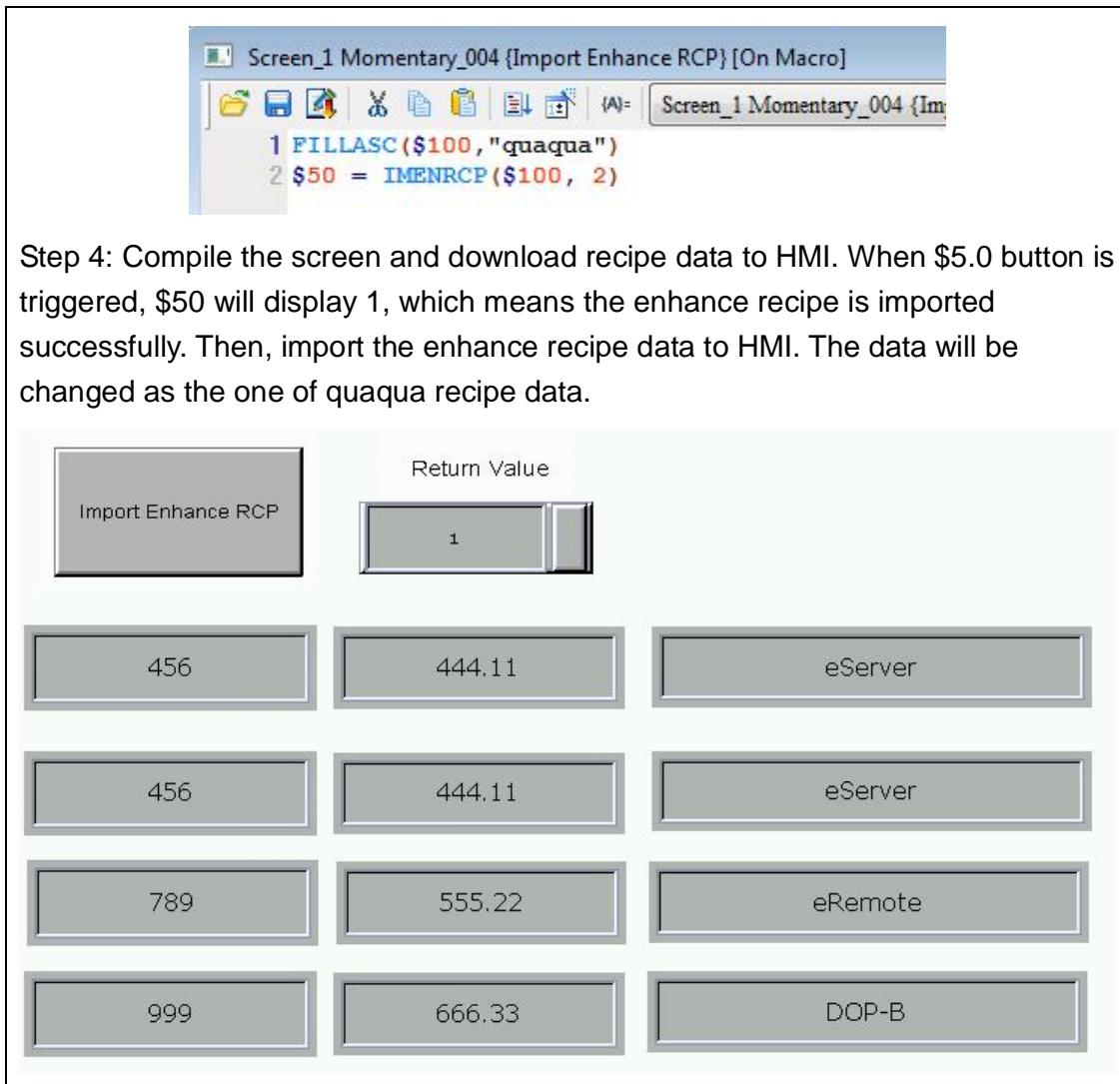
Step 1: Refer to the following figure that uses EXCEL to create recipe file. Its file name is quaqua.csv and is saved in USB Disk.

	A	B	C	D	E
1	ENRCP-1.0				
2	3	3			
3	2	1	0	0	
4	5	2	3	2	
5	8	5	0	0	
6	456	444.11 eServer			
7	789	555.22 eRemote			
8	999	666.33 DOP-B			

Step 2: Create momentary button (\$5.0), numeric entry element (\$50) and recipe address RCP0 ~ RCP11.



Step 3: Enter Screen of Momentary Button to edit On macro. Please see as below. Place quaqua to \$100 and import the enhance recipe data from USB Disk.



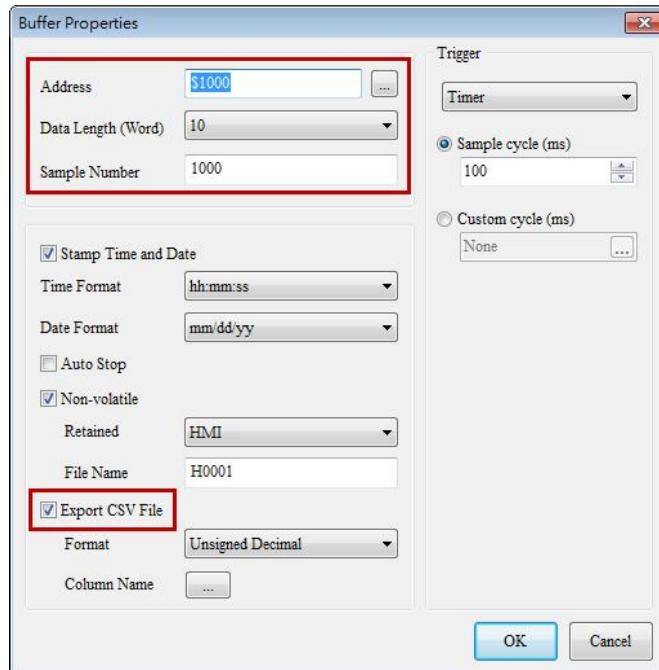
■ EXHISTORY (Export Historical Data)

Expression	What Variables Represent			NOTE				
Var1 = EXHISTORY(Var2, Var3, Var4)	Var 1	Returned Value						
		Failure	0					
		Success	1					
	Var 2	Historical Buffer Area ID						
	Var 3	Exported File Name						
	Var 4 External Storage Device	USB Disk	2					
		SD Card	3					
	Expression Explanation							
Export the historical data to external storage device.								
<p>Note 1: In buffer area property setting, 『output to CSV file』 has to be checked.</p> <p>Note 2: When the buffer area ID is 0, it means all historical buffer areas are exported. If 3 historical buffer areas are opened, 3 files will be exported and the file name will be 『Exported File 1.csv』, 『 Exported File 2.csv』 and 『 Exported File 3.csv』, respectively. If the area ID is not 0, it means the exported historical buffer area ID is specified. When it exports one file, the file name will be 『Exported File.csv』.</p> <p>Note 3: When the file is exported, 『.csv』 will be added as the filename extension automatically. Characters such as 『\』, 『/』, 『:』, 『*』, 『?』, 『"』, 『<』, 『>』 and 『 』 cannot be included in file name.</p> <p>If x00 is included in the string, the string will be judged as finish.</p>								

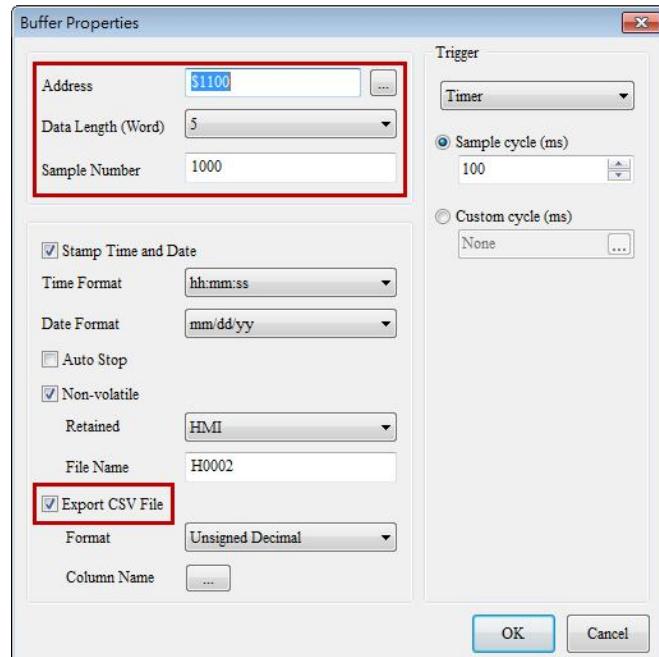
Variables	Memory Usage			
	Internal Memory	PLC Register	String	Constant
Var 1	◎	◎		
Var 2	◎	◎		◎
Var 3	◎	◎		
Var 4	◎	◎		◎

Example Description

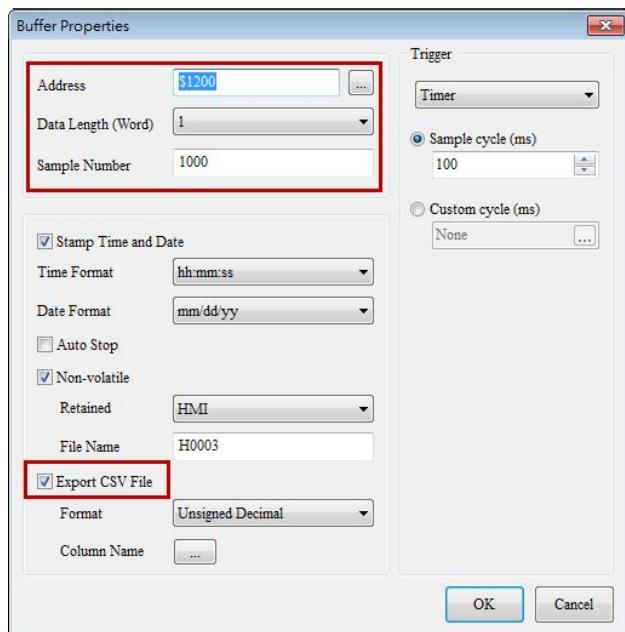
- Step 1: Setup three historical buffer areas. See the setting below:
(1) Historical Buffer Area 1: Set Address to \$ 1000, Data Length to 10 and check “Export CSV file”.



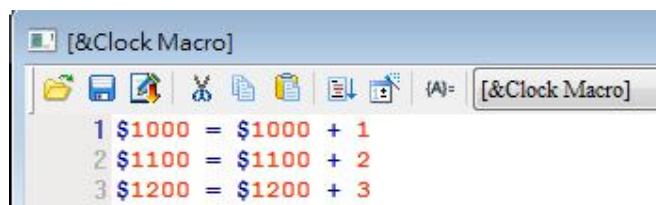
- (2) Historical Buffer Area 2: Set Address to \$ 1100, Data Length to 5 and check “Export CSV file”.



- (3) Historical Buffer Area 3: Set Address to \$ 1200, Data Length to 1 and check “Export CSV file”.



Step 2: Add historical data in Clock Macro.



Step 3: Create character entry element and set its address as \$580, string length as 5.



Step 4: Create 3 numeric entry elements and set their address as \$500, \$591 and \$501.



Step 5: Create maintained button element, set the address to \$100.0 and add On Macro.



The screenshot shows a software interface for editing macros. The title bar says "Screen_1 Maintained_009 {EXHISTORY} [On Macro]". The menu bar includes options like File, Edit, View, Tools, Help, and a dropdown "(A)..." which is set to "Screen_1 Maintained_009 {EXHISTORY} ▾". The main area contains the following macro code:

```
1 FILL($510, 0, 30)
2 $570 = TOWORD($580, 5)
3 $200 = GETSYSTEMTIME
4 $50 = SPRINTF($510, "%c%c%c_%d%d%d_History", $570, $571, $572, $200, $201, $202)
5
6 $500 = EXHISTORY($501, $510, $591)
```

Description of macro command:

Line 1: Clear \$510 ~ \$539

Line 2: Convert \$580 (String of Equipment Name) to word by the unit of byte.

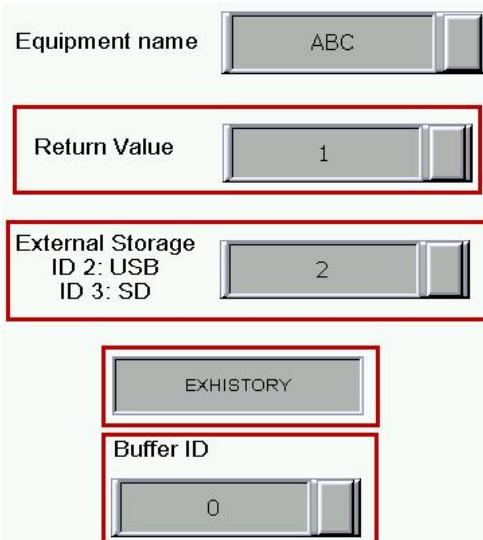
Line 3: Move the system time to \$200 ~ \$206 (year, month, day, week, hour, minute, second).

Line 4: Add『_year, month, day』 and 『_History』 to 『three words of equipment name』 to form a serial string and specify \$510 as start address.

Line 6: Export the historical data to the specified external device and file name.

Step 6: Download the editing screen to HMI and insert USB Disk to the port of HMI.

Step 7: Enter the Equipment name as 『ABC』 and choose 2 for External Storage Device, which means USB is applied. Set buffer area ID to 0 (export all) then press 『EXHISTORY』. When it is completed, the returned value becomes 1.



Step 8: Remove USB. And the exported files are shown as below:



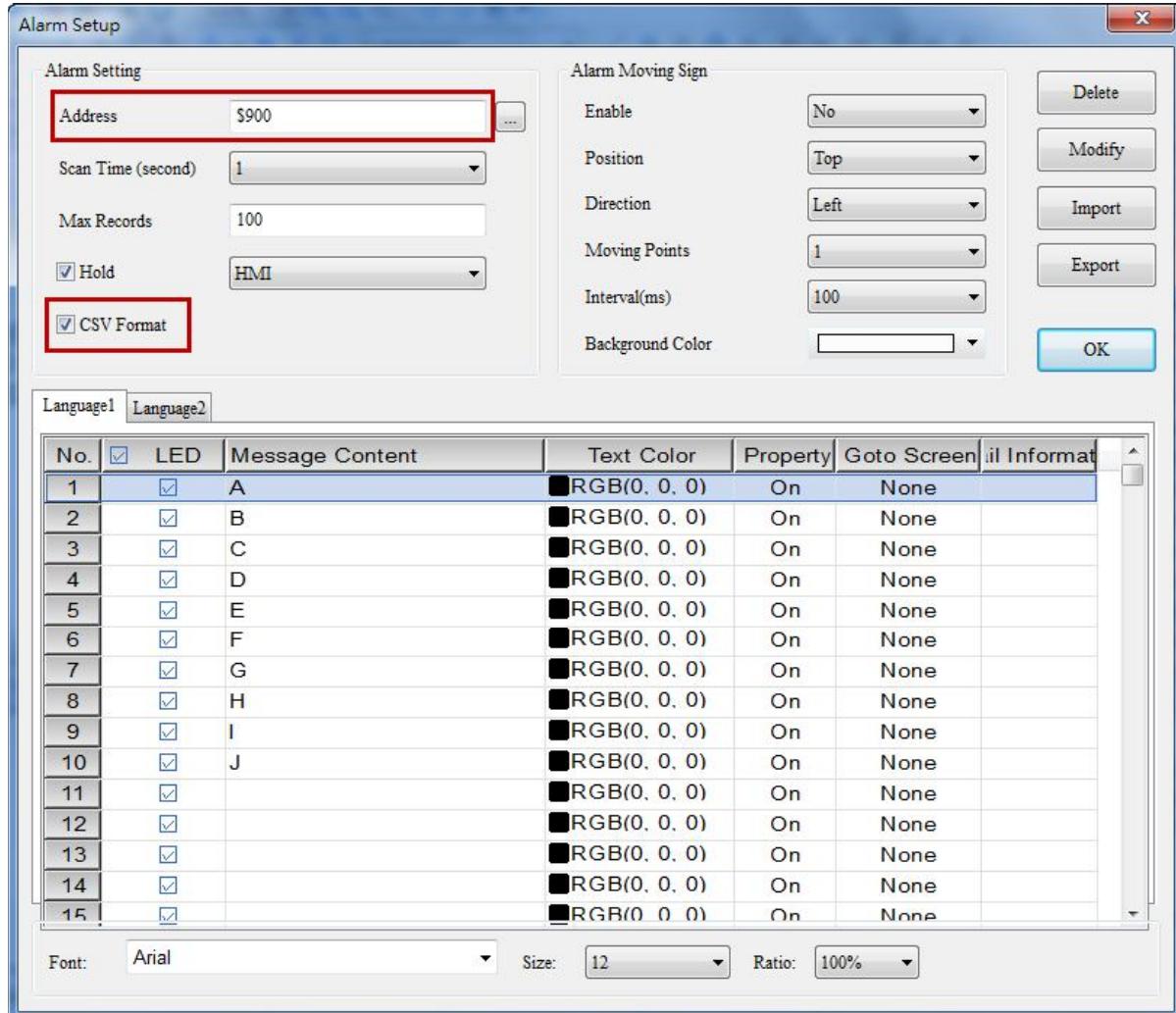
■ EXALARM (Export the Alarm)

Expression	What Variable Represent			NOTE		
Var1 = EXALARM(Var2, Var3)	Var 1	Returned Value				
		Failure	0			
		Success	1			
	Var 2	Exported File Name				
	Var 3	External Storage Device	USB Disk	2		
			SD Card	3		
Expression Explanation						
Export the alarm information to external storage device.						
Note 1: In Alarm setting, please check 『output to CSV file』 . Note 2: When the file is exported, 『.csv』 will be added as the filename extension automatically. Characters such as 『\』, 『/』, 『:』, 『*』, 『?』, 『"』, 『<』, 『>』 and 『 』 cannot be included in file name. If x00 is included in the string, the string will be judged as finish.						

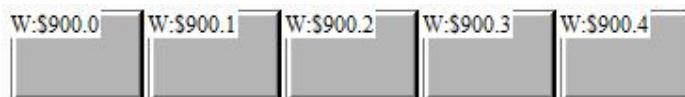
Variables	Memory Usage			
	Internal Memory	PLC Register	String	Constant
Var 1	◎	◎		
Var 2	◎	◎		
Var 3	◎	◎		◎

Example Description

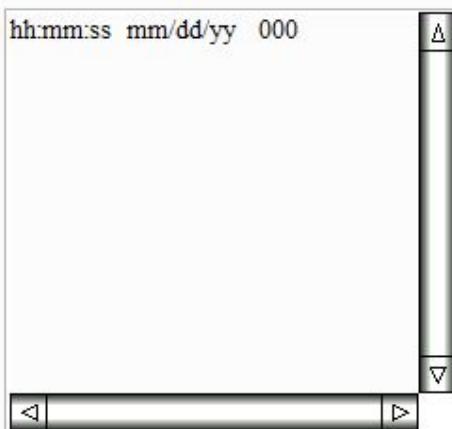
- Step 1: The setting of Alarm is shown as below. The Address is set as \$900. Please check "CSV Format".



- Step 2: Create maintained element for triggering alarms. The address is set as \$900.0, \$900.1, \$900.2, \$900.3 and \$900.4 in sequence.



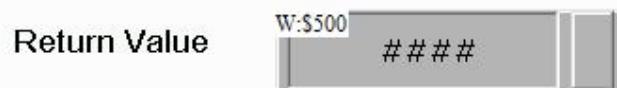
- Step 3: Create Historical Alarm Table to show the current historical alarms.



Step 4: Create character entry element and set its address as \$580 and string length as 5.



Step 5: Create two numeric entry elements and set their address as \$500 and \$591, respectively.



Step 6: Create maintained button element and set its address as \$100.1. Then, add On Macro.



Screen_1 Maintained_016 {EXALARM} [On Macro]

```
1 FILL($510, 0, 30)
2 $570 = TOWORD($580, 5)
3 $200 = GETSYSTEMTIME
4 $50 = SPRINTF($510, "%c%c%c_%d%d%d_Alarm", $570, $571, $572, $200, $201, $202)
5
6 $500 = EXALARM($510, $591)
```

See below for the descriptions of macro commands:

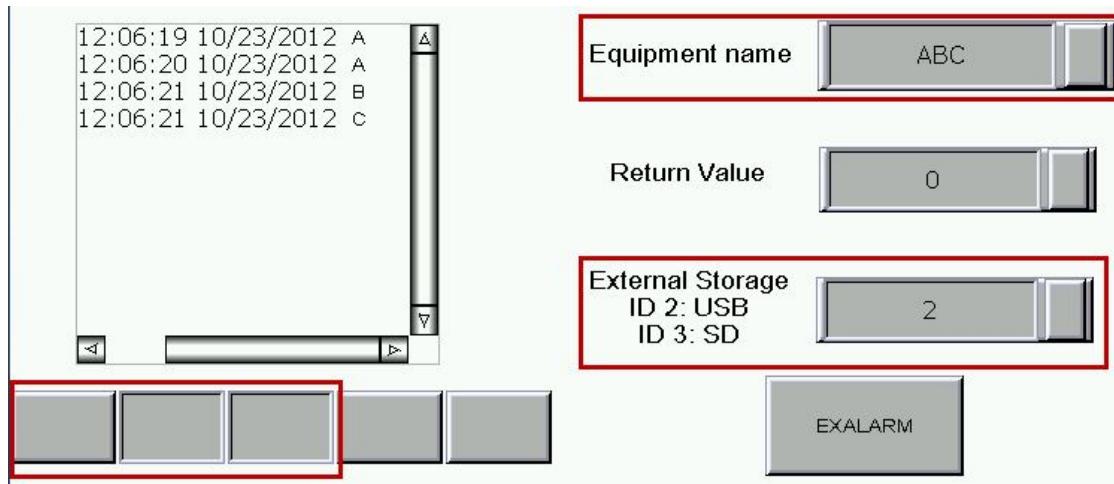
Line 1: Clear \$510 ~ \$539.

Line 2: Convert \$580 (String of Equipment Name) to word by the unit of byte.

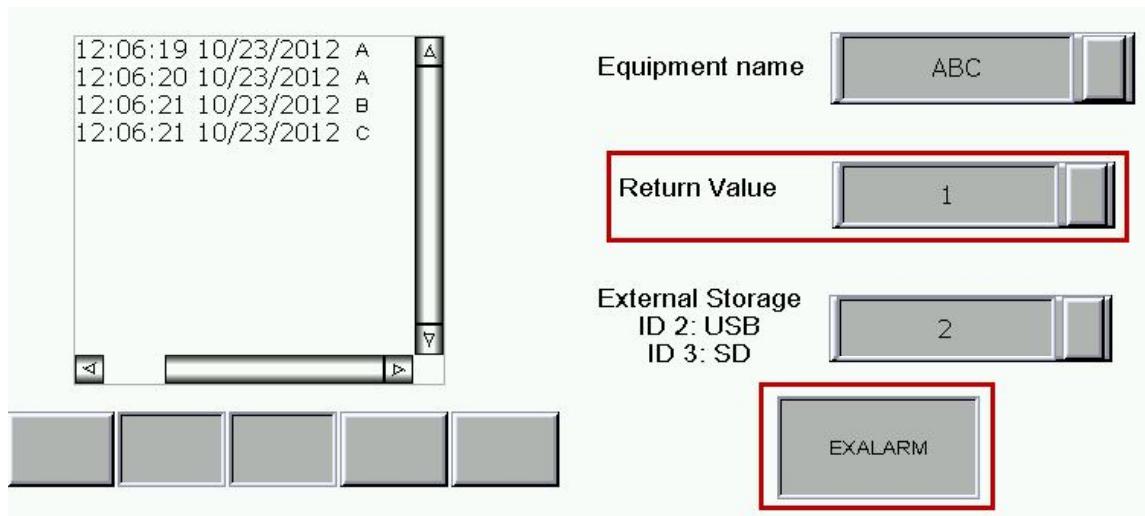
- Line 3: Move the system time to \$200 ~ \$206 (year, month, day, week, hour, minute, second).
 Line 4: Add『_year, month, day』 and『_Alarm』to『three words of equipment name』to form a serial string and specify \$510 as start address.
 Line 6: Export the historical alarm to the specified external device and file name.

Step 7: Download the editing screen to HMI and insert USB Disk to the port of HMI.

Step 8: Trigger the alarm and enter『ABC』as the equipment name on the screen. Select『2』for external storage device, which means USB is applying.



Step 9: Press『EXALARM』button. When it is complete, the returned value will return to 1.



Step 10: Remove USB. And the file exported from USB will be shown as below:



■ DISKFORMAT (Format Disk)

Expression	What Variables Represent				NOTE			
Var1 =DISKFORMAT(Var2) (W)	Var 1	Returned Value	Successful	0	W : Word Var 1 : Internal Memory Address Var 2 : Constant			
			Failure	1				
	Var 2	External storage device	SD Card	1				
			USB Disk	2				
Expression Explanation								
Select the desired device to be formatted as specified in Var 2 and save the returned value in Var 1.								

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎

Example

- Var 1 is an internal memory address and Var 2 is a constant. Format the USB Disk and save the returned value in \$100.

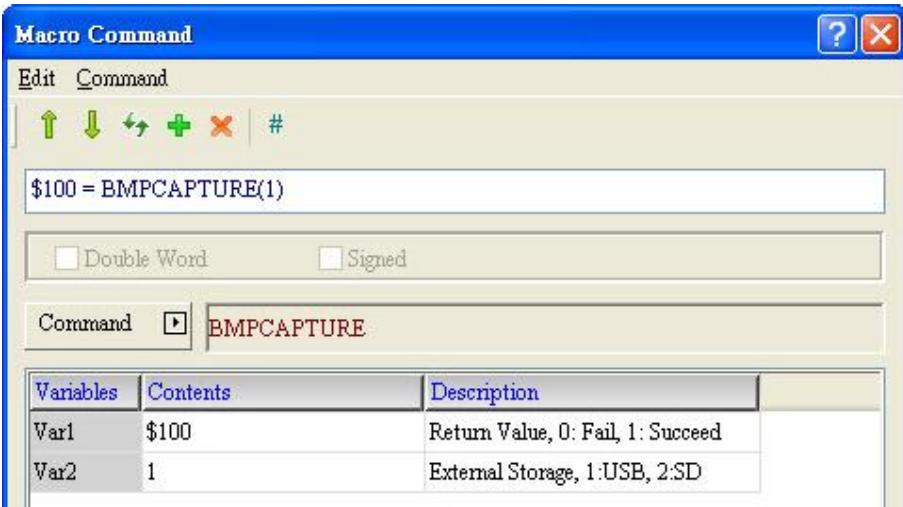
The screenshot shows the 'Macro Command' dialog box. The command input field contains '\$100 = DISKFORMAT(1)'. Below the input field are two checkboxes: 'Double Word' (unchecked) and 'Signed' (unchecked). Under the 'Command' label, 'DISKFORMAT' is selected. At the bottom, there is a table with three columns: 'Variables', 'Contents', and 'Description'. The table contains two rows:

Variables	Contents	Description
Var1	\$100	Return Value, 0: Fail, 1: Succeed
Var2	1	External Storage, 1:USB, 2:SD

■ BMPCAPTURE (Screen Capture)

Expression	What Variables Represent				NOTE	
Var1 = BMPCAPTURE(Var2) (W)	Var 1	Returned Value	Successful	0	W : Word Save the snapshot into the device specified in Var 2 and the returned value to Var 1.	
			Failure	1		
	Var 2	External storage device	SD Card	1		
			USB Disk	2		
	Expression Explanation					
Save the snapshot into the device specified in Var 2 and the returned value to Var 1.						
<ul style="list-style-type: none"> * Use BMPCAPTURE to export file is save as .bmp format. * Export path is under the root and save folder as named for currently year, month and date, save file as named for Hour, minute and second. 						

Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎

Example									
<p>➤ Var 1 is an internal memory address and Var 2 is a constant. Save the snapshot to the USB Disk and the returned value in \$100.</p>  <p>The Macro Command dialog box shows the following details:</p> <ul style="list-style-type: none"> Command: \$100 = BMPCAPTURE(1) Variables: <table border="1"> <thead> <tr> <th>Variables</th> <th>Contents</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Var1</td> <td>\$100</td> <td>Return Value, 0: Fail, 1: Succeed</td> </tr> <tr> <td>Var2</td> <td>1</td> <td>External Storage, 1:USB, 2:SD</td> </tr> </tbody> </table> 	Variables	Contents	Description	Var1	\$100	Return Value, 0: Fail, 1: Succeed	Var2	1	External Storage, 1:USB, 2:SD
Variables	Contents	Description							
Var1	\$100	Return Value, 0: Fail, 1: Succeed							
Var2	1	External Storage, 1:USB, 2:SD							

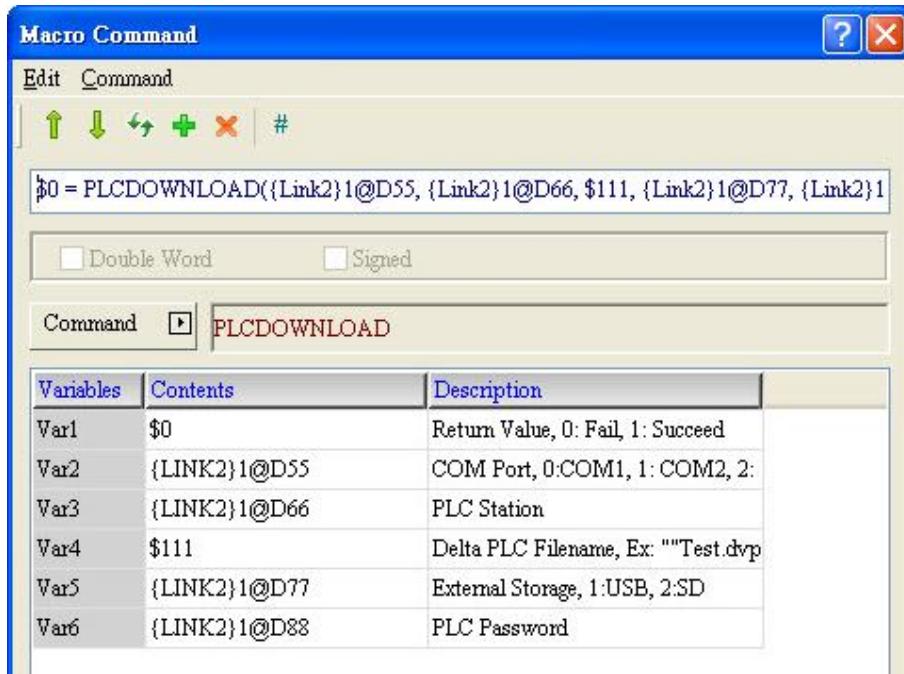
■ PLCDOWNLOAD

Command	What Variables Represent				NOTE					
Var1 = PLCDOWNLOAD(Var2, Var3, Var4, Var5, Var6) (W)	Var 1	Return Value	Failed	0	W : Word					
			Success	1						
	Var 2	COM Port	COM1	0						
			COM2	1						
			COM3	2						
	Var 3	PLC Station number								
	Var 4	DELTA PLC file name For example like delta.dvp, delta.isp								
	Var 5	External	SD Card	1						
		Storage	USB Disk	2						
	Var 6	PLC Password								
Expression Explanation										
Download PLC file to PLC										
<ul style="list-style-type: none"> * Only support Delta PLC. * File format support .dvp and .isp. * Please use Character entry element for PLC Password with Var 6. 										

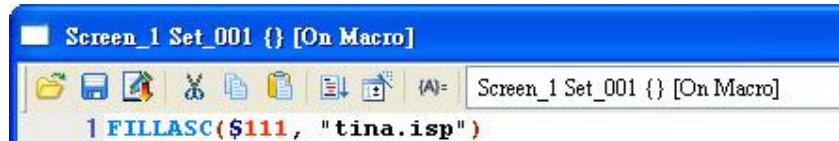
Variable	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎	◎	
Var 2	◎	◎	◎
Var 3	◎	◎	◎
Var 4	◎	◎	
Var 5	◎	◎	◎
Var 6	◎	◎	

Example

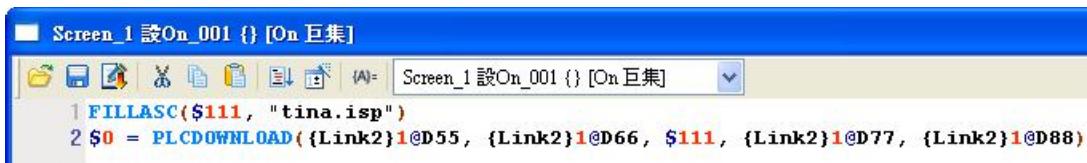
- Please save downloaded ISP file to USB Disk or SD card then return value to \$0.



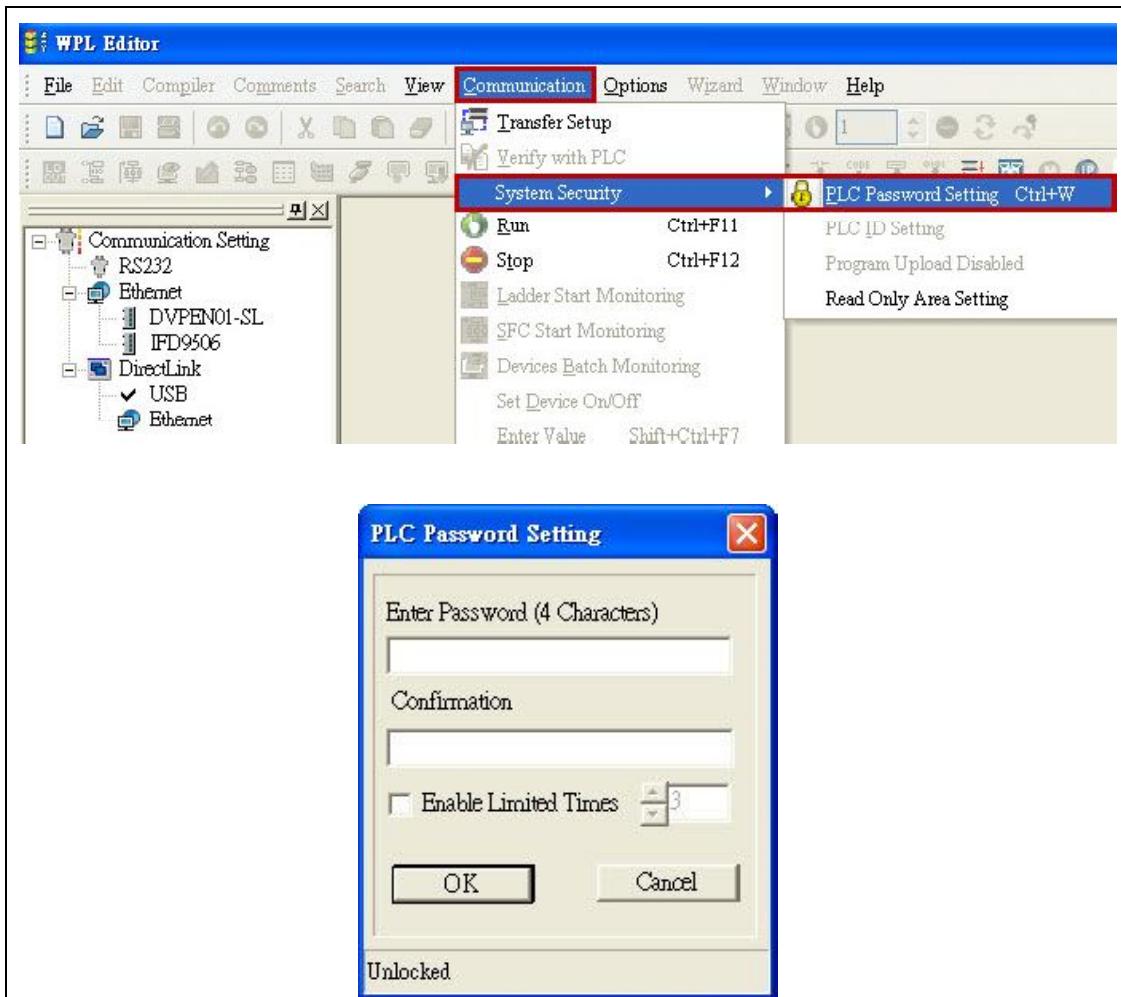
- Var 4 means PLC file name. So it has to use another command "FILLASC" and set file name string to some address.



- Then use the address \$111 on Var 4 of PLCDOWNLOAD macro command.



- Var 6 means PLC Password, and it has to set password by WPL or ISP software. After setting, it could use Character Entry element to input password and download PLC file to PLC.
 - WPL Password setting



- ISP Password setting





■ GetCircleCenter (Calculate the Coordinate of Circle Center)

Expression	What Variable Represent			NOTE			
Var1 = GetCircleCenter (Var2, "Var3")	Var 1	Returned Value		DW = Double Word			
		Failure	0				
		Success	1				
	Var 2	Enter the calculated three coordinates (Note 1)					
	Var 3	Circle center coordinate that is complete calculation (Note 2)					
	Expression Explanation						
Enter the three coordinates to calculate circle center coordinate (Note 3)							

Note 1: Enter the calculated three coordinates:

Set 3 points as P1(x1, y1), P2(x2, y2) and P3(x3, y3). Each point has the length of DW.

Assume that Var2 is \$100, then:

x1: LOW WORD is saved in \$100 and HIGH WORD is saved in \$101.

y1: LOW WORD is saved in \$102 and HIGH WORD is saved in \$103

x2: LOW WORD is saved in \$104 and HIGH WORD is saved in \$105

y2: LOW WORD is saved in \$106 and HIGH WORD is saved in \$107

x3: LOW WORD is saved in \$108 and HIGH WORD is saved in \$109

y3: LOW WORD is saved in \$110 and HIGH WORD is saved in \$111

Note 2: Circle center coordinate that is complete calculation:

Assume the coordinate of circle center is P4 (x4, y4) and each point has the length of DW. If

Var3 is \$200, then:

x4: LOW WORD is saved in \$200 and HIGH WORD is saved in \$201

y4: LOW WORD is saved in \$202 and HIGH WORD is saved in \$203

Note 3 : Calculation formula:

$$x = \Delta x / \Delta$$

$$y = \Delta y / \Delta$$

Among them, $\Delta = 2(xa - xb) * (yc - yb) - 2(ya - yb) * (xc - xb)$

$$\Delta x = (yc - yb) * (xa^2 + ya^2 - xb^2 - yb^2) - (ya - yb) * (xc^2 + yc^2 - xb^2 - yb^2)$$

$$\Delta y = (xa - xb) * (xc^2 + yc^2 - xb^2 - yb^2) - (xc - xb) * (xa^2 + ya^2 - xb^2 - yb^2)$$

Variables	Memory Usage		
	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2	◎		
Var 3	◎		

Example Description

➤ **Example:**
 $\$1 = \text{GetCircleCenter} (\$100, \$200)$

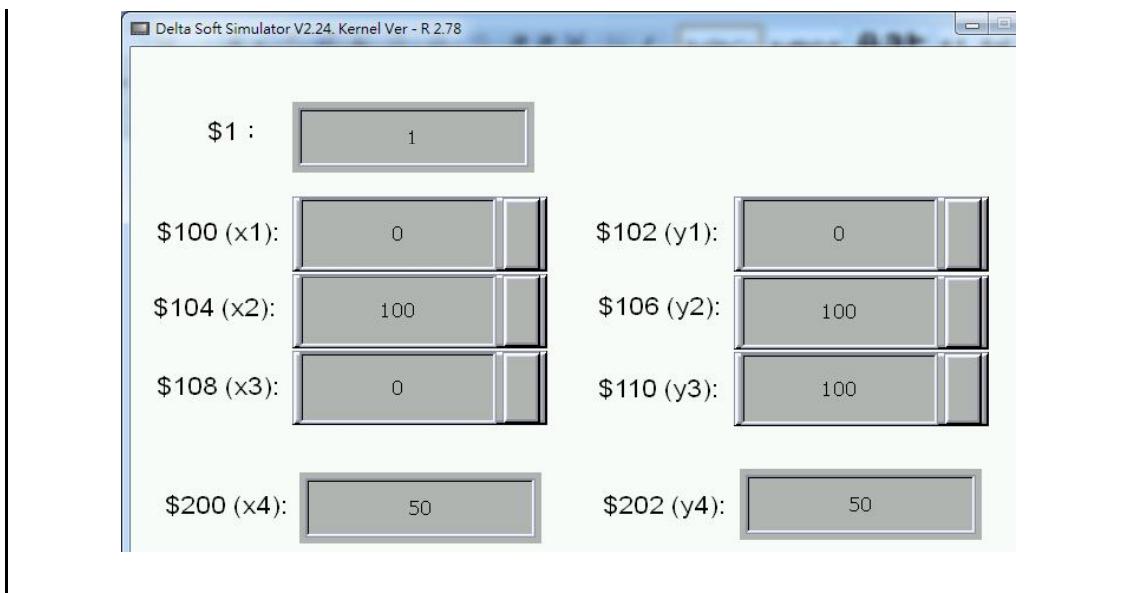
變數	內容	描述
Var1	\$1	回傳值 0:失敗, 1:成功
Var2	\$100	輸入三點坐標(每一點為DW長度)
Var3	\$200	輸出圓心坐標(每一點為DW長度)

Step 1: Create \$1 numeric entry element and the unit is Word.
Step 2: Create \$100, \$102, \$104, \$106, \$108, \$110 numeric entry element and the unit is Double Word. Please enter the following value, respectively.

$\$100 = 0$
 $\$102 = 0$
 $\$104 = 100$
 $\$106 = 100$
 $\$108 = 0$
 $\$110 = 100$

Step 3: Create \$200 and \$202 numeric entry element. The unit is Double Word.
After Macro is executed, the screen will be shown as below:

Chapter 24 Macro



24-3-12 Industrial Application

Industrial application provides ECAM macro to create E-Cam curve. Please refer to the followings for further description:

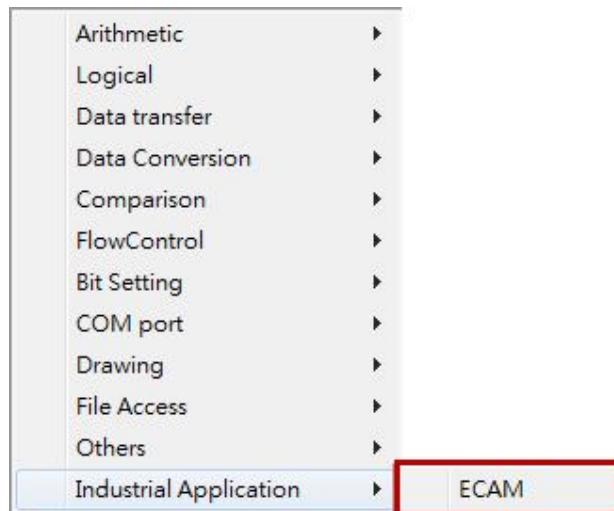


Figure 27-2-1 E-CAM Macro

■ ECAM

Expression	What Variable Represent		NOTE
Var1 = ECAM(Var2, Var3, Var4, Var5) (DW)	Var 1	Returned Value	DW : Double Word
	Var 2	Address of E-Cam curve algorithm mode	
	Var 3	Start address of E-Cam parameter input	
	Var 4	Start address of E-Cam curve output	
	Var 5	Address of E-Cam curve output	

Memory Usage			
Variables	Internal Memory	PLC Register	Constant
Var 1	◎		
Var 2			◎
Var 3	◎		
Var 4	◎		
Var 5	◎		

Detailed Command Description

Var1: Returned value

Everytime when curve calculation is complete, x1234000 will increase 1.

In curve calculation, Var1 will be cleared to 0.

If the curve cannot be calculated, it will return the error code. See the descriptions

below:

Error Code	Causes
0	No error
1	E-Cam table type error
2	Command type error
3	Input parameter error
4	E-Cam area number exceeds the range
5	Space in data array is not enough (ASDA-A2 series provides 800DW)
6	Input degree exceeds the range, reduce the degree of synchronous area or S-curve area
7	Cutting length exceeds the suggested range. Please enter the recommended cutting length (L) or speed compensation rate.
8	Speed compensation exceeds the range
9	E-gear ratio exceeds the specified range
10	The printing range is larger than roller circumference
11	Initial speed is < 0
12	Time procedure is in error
13	The cutter diameter is too small
14	Operation of cutter may be in error

Var2: Address of E-Cam curve algorithm mode

Use this address to select E-Cam curve algorithm mode ◦

Input Value	E-Cam Curve Creation Method
1	Rotary shear - Adjustable sealing zone
2	Printer machine
3	Rotary shear – cos compensation

Var3: Start address of E-Cam parameter input

Different E-cam curve creation methods requires different number of parameters. Var3 is used to setup the start address of E-Cam parameter. Please refer to the following table for further information:

E-Cam curve creation method	Input address	Parameters
Rotary shear - Adjustable sealing zone	n	E-Cam area number, P5_82_MIN ~ P5_82_MAX
	n+2	Deceleration ratio: Numerator (nGA)
	n+4	Deceleration ratio: Denominator (nGB)
	n+6	Knife number (nKnife)
	n+8	Knife diameter (d1) Unit: mm x 100

	n+10	Cutting length (L) Unit: mm x 100
	n+12	Speed compensation (dVcp) -50.00% ~ 50.00% x 100
	n+14	Angle in acceleration area (ns2) Unit: degree
	n+16	Angle in synchronous area (ns3) Unit: degree
	n+18	Angle in S-Curve area (nsS) Unit: degree
	n+20	E-gear ratio (numerator) (P1-44)
	n+22	E-gear ratio (denominator) (P1-45)
Printer Machine	n	E-Cam area number, P5_82_MIN ~ P5_82_MAX
	n+2	Deceleration ratio: Numerator (nGA)
	n+4	Deceleration ratio: Denominator (nGB)
	n+6	Printing range (dPL) Unit: mm x 100
	n+8	Blank range (dBLL) Unit: mm x 100
	n+10	Roller diameter (dd1) Unit: mm x 100
	n+12	Roller diameter of master (dd2) Unit: mm x 100
	n+14	Angle in waiting area (Deg1) Unit: degree
	n+16	Angle in S-Curve area (DegS) Unit: degree
	n+18	Angle in synchronous area (DegA) Unit: degree
	n+20	E-gear ratio (numerator) (P1-44)
	n+22	E-gear ratio (denominator) (P1-45)
Rotary shear – cos compensation	n	E-Cam area number, P5_82_MIN ~ P5_82_MAX
	n+2	Deceleration ratio: Numerator (nGA)
	n+4	Knife number
	n+6	Knife diameter (mm) x 100
	n+8	Product cut length (mm) x 100
	n+10	Speed compensation (master axis) Unit: %
	n+12	Lead constant speed area Unit: count number
	n+14	Cut thickness D(mm) x 100
	n+16	Cut height H(mm) x 100
	n+18	E-gear ratio (numerator) (P1-44)
	n+20	E-gear ratio (denominator) (P1-45)

	n+22	Speed ratio (conveyor 2: conveyor 1)
	n+24	S (curve level): 1 ~ 4 levels

Var4: Start address of E-cam curve output

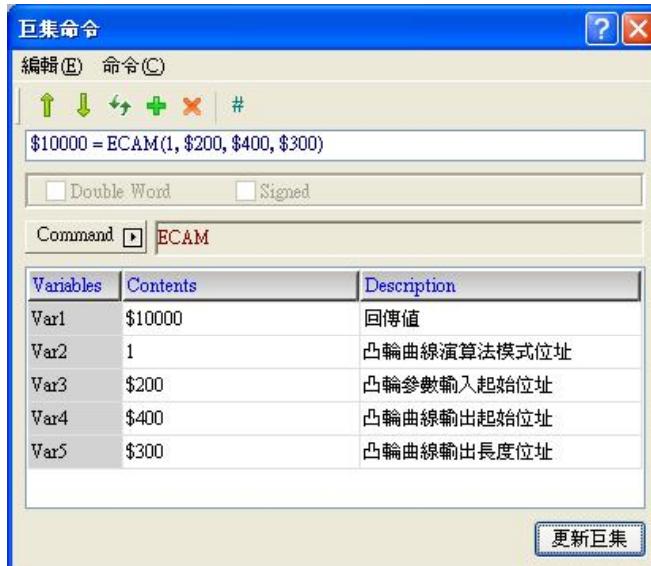
After calculation, the E-cam curve outputs to the start address of Var4. Please refer to Var5 for its output length.

Var5: Address of E-cam curve output length

After calculation, if the E-cam output length is 100, it will take 200-word long address.
(For E-cam funciton, the unit is Double Word.)

NOTE: Please be aware of address overlap when applying E-Cam funciton. E-Cam curve of ASDA-A2 can create up to 721 areas. Thus, the E-cam value will take 1422 words at most. If Var4 (start adress of E-Cam curve output) is set to \$0, it is suggested to reserve \$0~\$1441 for E-cam use only.

- | Example Description |
|--|
| <p>1. Create a momentary button and set its address as \$0.0. Write \$10000 = ECAM(1, \$200, \$400, \$300) in On Macro.</p> |



Var2 = 1. It indicates that this macro command calculate the E-Cam curve for rotary shear-adjustable sealing zone.

Parameters that is required by macro algorithm starting from \$200 use 12 Double Words (= 24 Words) and take \$200-\$223 as the memory address.

\$200	E-Cam area number, P5_82_MIN ~ P5_82_MAX
\$202	Reduction ratio (numerator) (nGA)
\$204	Reduction ratio (denominator) (nGB)
\$206	Knife number (nKnife)
\$208	Knife diameter (d1) Unit: mm x 100
\$210	Cutting length (L) Unit: mm x 100
\$212	Speed compensation, (dVcp) -50.00% ~ 50.00% x 100
\$214	Angle in acceleration area (ns2) Unit: degree
\$216	Angle in synchronous area (ns3) Unit: degree
\$218	Angle in S-Curve area (nsS) Unit: degree
\$220	E-gear ratio (numerator) (P1-44)
\$222	E-gear ratio (denominator) (P1-45)

After calculation, E-Cam parameters will be outputted to the address starting from \$400 to \$1841 at most.

(In A2 system, one E-cam curve can be divided up to 721 parts. One part takes

one double word. Thus, it will take $721 \times 2 = 1442$ Words. And its memory address is from \$400 to \$1841.)

The actual curve will be outputted to \$300.

2. Create one E-cam curve element on the screen. See below for its address setting:

E-CAM Curve Element	E-CAM Macro	Address
Read Buffer Address	Var4	\$400
Read Size Address	Var5	\$300
Read Start Address	Var1	\$10000

3. Create one E-cam sketch element. See below for its address setting:

E-CAM Sketch Element	E-CAM Macro	Address
Read Buffer Address	Var4	\$400
Read Size Address	Var5	\$300
Read Start Address	Var1	\$10000

4. Create 12 numeric entry elements on the screen. Unit is Double Word. See below for its address setting:

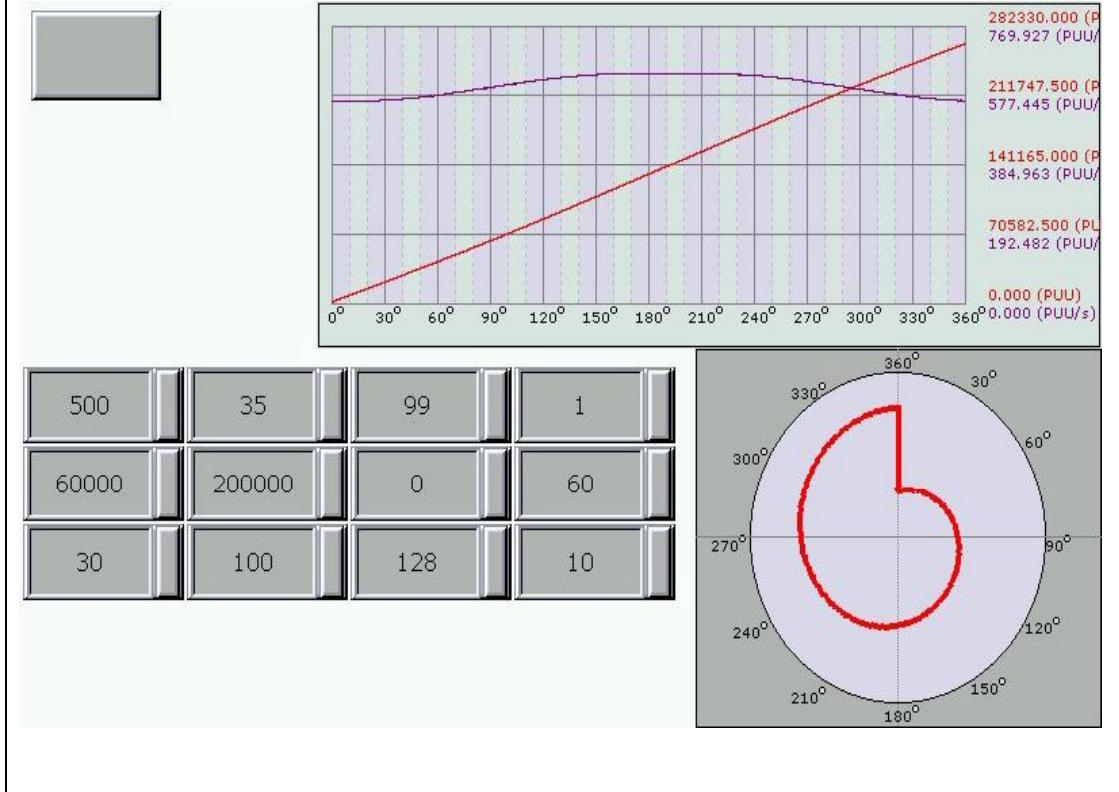
\$200	E-Cam area number, P5_82_MIN ~ P5_82_MAX
\$202	Reduction ratio (numerator) (nGA)
\$204	Reduction ratio (denominator) (nGB)
\$206	Knife number (nKnife)
\$208	Knife diameter (d1) Unit: mm x 100
\$210	Cutting length (L) Unit: mm x 100
\$212	Speed compensation, (dVcp) -50.00% ~ 50.00% x 100
\$214	Angle in acceleration area (ns2) Unit: degree
\$216	Angle in synchronous area (ns3) Unit: degree
\$218	Angle in S-Curve area (nsS) Unit: degree
\$220	E-gear ratio (numerator) (P1-44)
\$222	E-gear ratio (denominator) (P1-45)

5. When downloading to HMI, please fill in the following value in sequence:

\$200	500
\$202	35
\$204	99
\$206	1
\$208	60000
\$210	200000

	\$212	0
	\$214	60
	\$216	30
	\$218	100
	\$220	128
	\$222	10

6. Press momentary button \$0.0. The E-cam curve will be generated as below:



24-4 Macro Error Messages

Users may accidentally type in incorrect syntaxes or typos within macros. To help users quickly locate error Macro codes, the HMI system will display error codes in the output field during the coding process and also prompt error messages during complies to remind users about these errors.

➤ Error Message During Editing

Number	Number name	Trouble shooting
-100	LABEL cannot be found	There is no such LABEL required by the GOTO command
-101	Recursion occurred	This error message indicates that recursion has occurred, and errors of this sort mostly happen in submacros. The reason is that a submacro is called by itself within the same submacro, either directly or indirectly. Basically, the recursion technique can not be used within a submacro, but if this is unavoidable, please consider Goto or For (infinite loop) instead.
-102	More than 10 nested FOR used	This error message is to remind users not to use more than 10 nested FOR commands. The purpose is to prevent excessive uses of the FOR command and avoid memory insufficiency. If necessary, please consider GOTO or IF instead.
-103	Submacro does not exist	This error message indicates that the submacro called upon does not exist. As an example, the code "CALL 5" is intended to call a submacro 5, but the user somehow does not write submacro 5 in the program. To avoid unpredictable consequences due to this kind of mistakes (typos or forgot to add the corresponding sub-macro), this error message is hence displayed to remind the users.
-104	Number of NEXT is less than the number of FOR	Next and For are paired operands and must be used together. This error message indicates that the number of NEXT and FOR

		does not match. The program can not proceed to the next For if there is one NEXT missing.
-105	Number of FOR is less than the number of NEXT	Next and For are paired operands and must be used together. This error message indicates that the number of NEXT and FOR does not match. The program can not proceed to the next For if there is one extra NEXT.
-106	Repeated LABEL	This error message indicates that there is one duplicated LABEL within the same macro. This will generate unpredictable consequences since there are two sequences for the same GOTO. To avoid this kind of mistakes, this error message is hence displayed.
-107	There is RET in Macro	This error message indicates that is a RET command in the macro. The RET command is reserved for the submacro to return back to the macro. Should there be a need to end a macro, please consider END instead.

➤ HMI Macro Error Messages

Users can read error messages via macro commands. However, if a correct command is executed before the error message is read, then the previous error message will be overwritten (changed). Macro error messages will not be changed, however, while a different macro is being executed.

Number	Number name	Trouble shooting
-10	GOTO error	This message means that there is a GOTO error for the macro that is currently being executed.
-11	Stack overflow	This message indicates that the stack for the macro currently executed is full. This may be caused by using too many sub-macros or executing multiple macros at the same time. This message is a mechanism avoiding memory insufficiency.
-12	Empty Submacro	This message indicates that calling a sub-macro has failed. Since the Call

		command identifies a submacro through its ID stored in an internal memory address, if that address does contain the corresponding ID for the submacro, and then there will be no submacro to be upon.
-13	Data Read Error	This message indicates an error has occurred during the data reading process. Although there is a possibility the error is due to incorrect data stored in a specific internal memory address, most of the time it is the problem of an external controller that has caused this read error.
-14	Data Write Error	This message indicates an error has occurred during the data writing process. Although there is a possibility the error is due to incorrect data stored in a specific internal memory address, most of the time it is the problem of an external controller that has caused this write error.
-15	Divisor is 0	This error message indicates that the divisor is identified as 0 during the division or reminder operations.
-16	Data process error with BCD format	This error message indicates when execute BCD macro command have some error with data process.
-17	Data process error with convert ASCII to HEX format	This error message indicates when execute TOHEX macro command have some error with covert ASCII to HEX format.
-18	NEXT OFFSET error	This error message indicates when macro have data error will cause execute next command error.
-19	Character command error	This error message indicates when execute FILLASC will have error.
-20	Data process error with BIN format	This error message indicates when execute BIN macro command have some error with data process.
-21	Sub macro data error	This error message indicates when macro have data error will cause call sub macro error.

-22	FOR Loop have OFFSET error	This error message indicates when macro have data error will cause execute FOR macro error.
-24	INITIAL ERROR	This error message indicates when execute INITCOM command will have error.
-24	Memory allocation error	This error message indicates HMI memory is not enough to execute macro commands.
-25	COM Port error	This error message indicates COM Port has error and will cause execute related about COM Port macro failed.
-26	Print Port error	This error message indicates select incorrect Print Port when execute print action.
-27	Read value error	This error message indicates over range when read macro parameter.
-28	IF ELSE ENDIF error	This error message indicates when execute IF ELSE ENDIF macro command have error.
-29	Pen width setting error	This error message indicates set incorrect pen width of draw macro.
-30	History data error	This error message indicates when execute GETHISTORY macro command have error.
-31	Export error	This error message indicates when execute EXPORT macro command have error.
-32	Disk reading error	This error message indicates external or internal storage have error will cause execute related about EXPORT and DISKFORMAT macro command error.
-33	Print error	This error message indicates when execute macro command to print have error.
-34	IF ELSE ENDIF stack over flow	This error message indicates stack over flow when execute IF ELSE ENDIF macro command.
-35	Password error	This error message indicates input password error when execute related password confirm macro.
-36	Password lock error	This error message indicates password exceed input limitation when execute related password confirm macro.
-37	ID password identify error	This error message indicates ID password error when execute related ID password

		confirm macro.
-38	Syntax error	This error message indicates after download PLC program have syntax error.
-39	Connection failed or no response	This error message indicates when download PLC program and detect connection failed or no response.

Explanation about PLC file error includes DVP and ISP file format.		
-40	File name not support	This error message indicates not support file name when execute macro to open PLC file.
-41	Version not support	This error message indicates not support version when execute macro to open PLC file.
-42	Open file error	This error message indicates open file failed when execute macro to open PLC file.
-43	File Handle error	This error message indicates file point error when execute macro to open PLC file.
-44	File reading error	This error message indicates cannot reading when execute macro to open PLC file.
-45	File Seek error	This error message indicates cannot move file content when execute macro to open PLC file.
-46	File writing error	This error message indicates cannot writing when execute macro to open PLC file.
-47	File remove error	This error message indicates remove file failed when execute macro to remove file.
-48	File Rename error	This error message indicates file rename failed when execute macro to rename file.
-49	File length error	This error message indicates file length error when execute macro.
-50	File data error	This error message indicates file data error when execute macro.