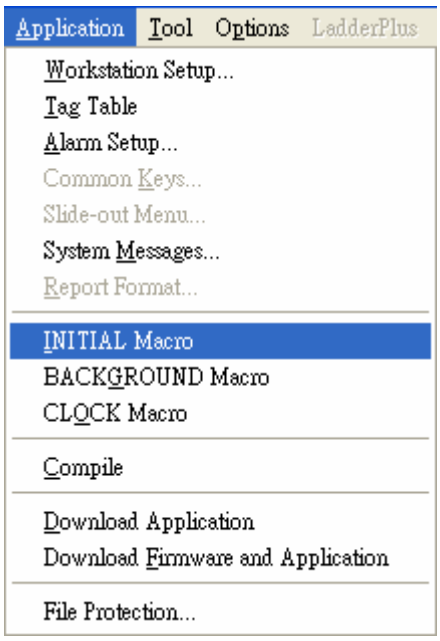
8. Macro

8.1. Macro Function

ADP offers user a convenient and powerful Macro application; Macro enables the Workstation to execute a number of tasks: Arithmetic, Logic, Flow Control, Data Transfer, Comparison, Conversion and system service instructions, etc. Using Macro can also significantly save the program size and optimize the efficiency in PLC. In Macro, not only can the Workstation communicate to the PLC but also connect to other devices. Thus Macro provides an efficient integration system as well as an economical structure in hardware application.

8.2. Classification in Macro

Macro offers user a number of functions in different situations and applications. A user can define the application according to the needs in the corresponding Marco window. PWS HMI will execute the macro commands according to the different modes. Macro is categorized into the following commands:



A. Application Macro commands：There are three types of macro commands in the [Application] menu.

1. INITIAL Macro: The purposes of INITIAL Macro are data initialization and communication parameters declaration. This command is executed only once when an application is started and the start-up screen is not appeared until this command is executed. There is one INITIAL Macro in one application.

2. BACKGROUND Macro ： When the HMI runs the application, the macro commands will execute cyclically. The maximum macro commands are 30 rows. The macro commands will execute whatever the current screen is. The common use for BACKGROUND Macro is communication control and PLC sampling data convert….etc.

3. CLOCK Macro：When the HMI runs this application, the entire macro commands will be executed every 500ms.

The common use for Clock Macro is display control, PLC bit monitor, timer control, data timer convert…etc.

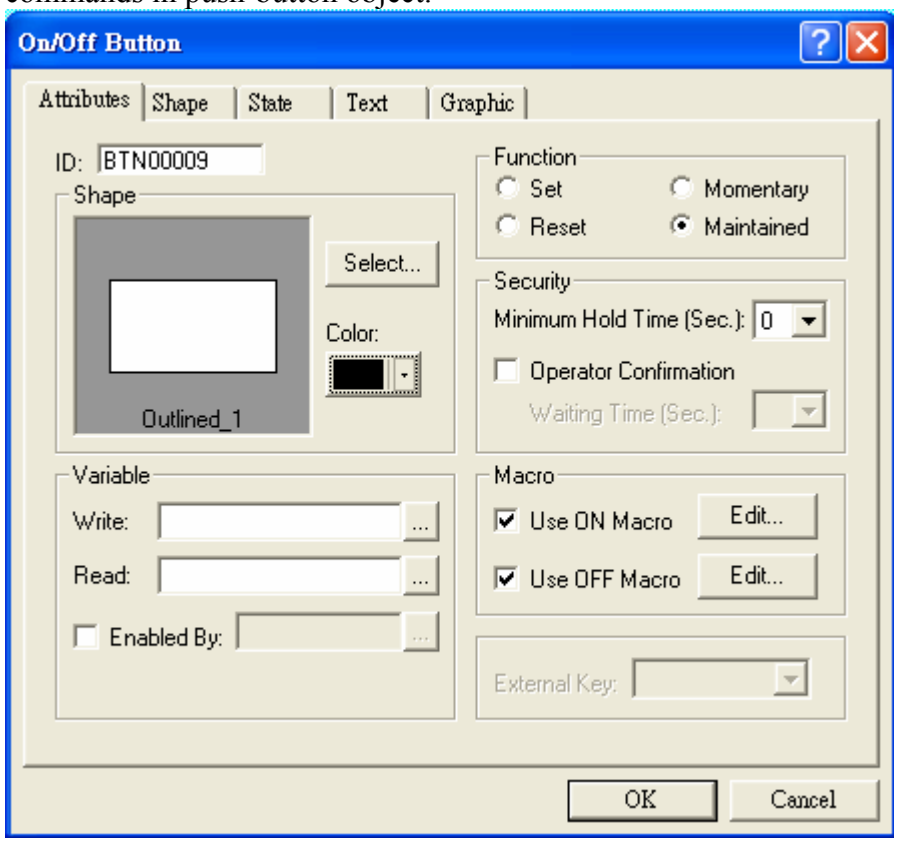
B. Screen Macro Commands：There are three macro commands in the [Screen] menu.

1. OPEN Macro：OPEN Macro is executed when the screen is commanded to be opened. The common use for OPEN Macro is screen initialization, display control, internal register or bit initialization….etc.

2. CLOSE Macro：CLOSE Macro is executed when the screen is commanded to be closed. The CLOSE Macro will execute its command once.

3. CYCLIC Macro：CYCLIC Macro is executed cyclically when the screen is display. The HMI will execute the BACKGROUND Macro and CLOCK Macro periodically.

C. ON/OFF Macro Commands：There are two ON/OFF macro commands in push-button object.

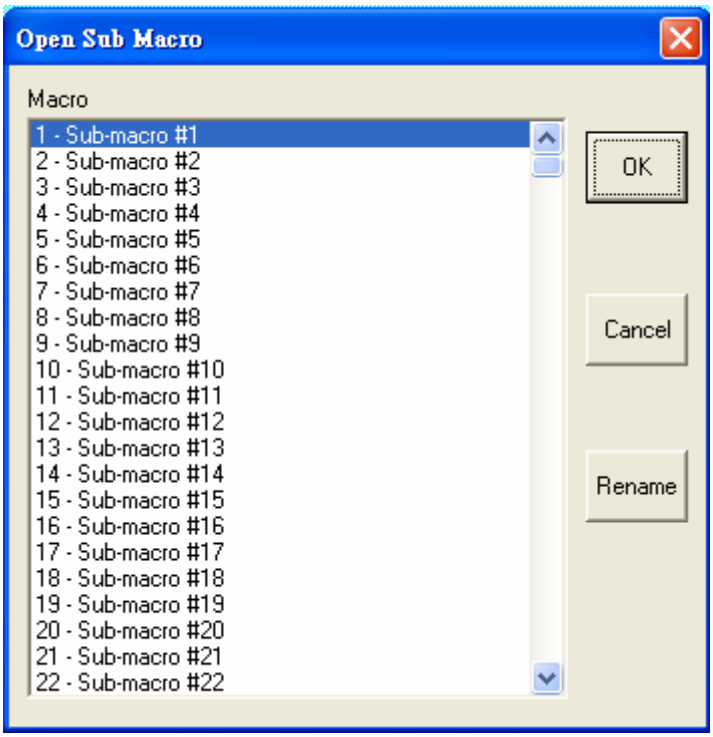


1. ON Macro：ON Macro is executed when the button is clicked and set a bit to be ON. The common use for ON Macro is the action of push-button, the control of chain process or the effect of displaying the initial screen and PLC register and bit initilialization…etc.

2. OFF Macro：OFF Macro is executed when the button is clicked and released a bit to be OFF. Then the HMI will execute the Macro commands once. The common use for HMI is the action of push-button, the control of sequence process, the effect of displaying the close screen…etc.

D. Sub-Macro Commands：A user can select [Sub-Macro] from the [Object] menu.

1. Sub-Macro：It is the sub-command of Macro. Sub-Macro is executed by the HMI with CALL command. The common use for Sub-Macro is to edit and save some basic functions or arithmetic commands for Macro.


Open Sub Macro
Macro
1 - Sub-macro #1
2 - Sub-macro #2
3 - Sub-macro #3
4 - Sub-macro #4
5 - Sub-macro #5
6 - Sub-macro #6
7 - Sub-macro #7
8 - Sub-macro #8
9 - Sub-macro #9
10 - Sub-macro #10
11 - Sub-macro #11
12 - Sub-macro #12
13 - Sub-macro #13
14 - Sub-macro #14
15 - Sub-macro #15
16 - Sub-macro #16
17 - Sub-macro #17
18 - Sub-macro #18
19 - Sub-macro #19
20 - Sub-macro #20
21 - Sub-macro #21
22 - Sub-macro #22
OK
Cancel
Rename

8.3. Macro Commands

The following details the macro commands and the formats. For the setup, please refer to the next section.

Operation	Format	A1*	A2*	A3*	Data Format	PLC Data
ADD	A1=ADD(A2,A3)	2	2,4	2,4	DW/Signed	X
SUB	A1=SUB(A2,A3)	2	2,4	2,4	DW/Signed	X
MUL	A1=MUL(A2,A3)	2	2,4	2,4	DW/Signed	X
DIV	A1=DIV(A2,A3)	2	2,4	2,4	DW/Signed	X
MOD	A1=MOD(A2,A3)	2	2,4	2,4	DW/Signed	X
OR	A1=A2 | A3	2	2,4	2,4	DW	X
AND	A1=A2&A3	2	2,4	2,4	DW	X
XOR	A1=A2^A3	2	2,4	2,4	DW	X
SHL	A1=A2<<A3	2	2,4	2,4	DW	X
SHR	A1=A2>>A3	2	2,4	2,4	DW	X
MOV	A1= A2	0,2	0,2,4	~	DW	O
BMOV	BMOV(A1,A2,A3)	0,2	0,2,	2,4		O
FILL	FILL(A1,A2,A3)	2	2,4	2,4		X
CHR	CHR(A1,"A2")	2	5	~		X
GETX	A1=@X	2,4				X
SETY	@Y=A1	2,4				X
X2Y	X2Y(A1,A2)	2,4	2,4			X
IF==	IF A1==A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF!=	IF A1!=A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF>	IF A1>A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF>=	IF A1>=A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF<	IF A1<A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF<=	IF A1<=A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF AND ==0	IF A1 AND A2==0 THEN GOTO A3	2,4	2,4	4	DW	X
IF AND !=0	IF A1 AND A2!=0 THEN GOTO A3	2,4	2,4	4	DW	X
IF==ON	IF A1=ON GOTO A2	3	4	~	Bit	X
IF==OFF	IF A1=OFFGOTO A2	3	4	~	Bit	X

* The useable range of memory will be indetified according to the commands. The number in the table represent :

0=PLC Device (Word), 1=PLC Device (Bit),
2=Internal Memory(Word), 3=Internal Memory(Bit),
4=Constant, 5=ASCII Character.

Operation	Format	A1*	A2*	A3*	Data Format	PLC Data

IF-THEN	IF condition ♦ THEN
	DO
	ENDIF
		2,4	2,4	~	Condition♦	X

IF-THEN-ELSE	IF condition ♦ THEN
	DO
	ELSE DO
	ENDIF
		2,4	2,4	~	Condition♦	X

Nest IF-THEN-ELSE	IF condition ♦ THEN
	DO
	IF-THEN-ELSE
	ELSE DO
	IF-THEN-ELSE
	ENDIF
		2,4	2,4	~	Condition♦	X

ELIF	IF condition 1♦ THEN
	DO
	ELIF condition 2♦
	THEN DO
	ELIF condition 3♦
	THEN DO
	ENDIF
		2,4	2,4	~	Condition♦	X

GOTO	Goto label A1	4	~	~		X
LABEL	Label A1	4	~	~		X
CALL	Call A1	2,4	~	~		X
RET	Return	~	~	~		X
FOR	For A1	2,4	~	~		X
NEXT	Next	~	~	~		X
SETB	Bit setting A1	1,3	~	~	Bit	O
CLRB	Bit resetting A1	1,3	~	~	Bit	O
INVB	Bit inversion A1	1,3	~	~	Bit	O
BCD	A1=BCD(A2)	2	2	~	DW	X
BIN	A1=BIN(A2)	2	2	~	DW	X
W2D	A1=W2D(A2)	2	2	~	Signed	X
B2W	A1=B2W(A2,A3)	2	2	2,4		X
W2B	A1=W2B(A2,A3)	2	2	2,4		X
SWAP	SWAP(A1,A2)	2	2,4	~		X
MAX	A1=MAX(A2,A3)	2	2,4	2,4	DW/Signed	X
MIN	A1=MIN(A2,A3)	2	2,4	2,4	DW/Signed	X

* The useable range of memory will be indetified according to the commands. The number in the table represent :

0=PLC Device (Word), 1=PLC Device (Bit),
2=Internal Memory(Word), 3=Internal Memory(Bit),
4=Constant, 5=ASCII Character.

♦ Condition includes A1==A2, A1!=A2, A1>A2, A1>=A2, A1<A2, A1<=A2, (A1&A2)==0, (A1&A2)!=0, A1==ON或A1==OFF。A1 and A2 are only for internal memory and constant.

A2H	A1=A2H(A2)	2	2			X
H2A	A1=H2A(A2)	2	2			X
TIMETICK	A1= TIMETICK	2	~	~	DW	X
COMMENT	#A1="Chars"	5	~	~		X
SYS	SYS(A1,A2)					X
Operation	**Format**	**A1***	**A2***	**A3***	**Data Format**	**PLC Data**
	SYS(SET_TIMER,N)		4			X
	SYS(STOP_TIMER,N)		4			X
	SYS(SET_COUNTER,N)		4			X
	SYS(STOP_COUNTER,N)		4			X
	SYS(WAIT_TIMER,N)		4			X
	SYS(WAIT_COUNTER,N)		4			X
	SYS(INI_COM,N)		4			X
	SYS(GET_CHAR,N)		4			X
	SYS(GET_CHARS,N)		4			X
	SYS(PUT_CHAR,N)		4			X
	SYS(PUT_CHARS,N)		4			X
	SYS(READ_WORDS,N)		4			X
	SYS(READ_位元 S,N)		4			X
	SYS(WRITE_WORDS,N)		4			X
	SYS(WRITE_位元,N)		4			X
	SYS(SUM_ADD,N)		4			X
	SYS(SUM_XOR,N)		4			X

Arithmetic

[Notes]: Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory). The data format is Word, Double-Word, Signed binary and Unsigned binary.
Format: A1 = A2 + A3

Description: Adds A2 and A3 and saves the result in A1.

- **ADD** → Format: A1 = A2+A3. Adds A2 and A3 and saves the result in A1.

* The useable range of memory will be indetified according to the commands. The number in the table represent :

0=PLC Device(Word), 1=PLC Device(Bit),
2=Internal Memory(Word), 3=Internal Memory(Bit),
4=Constant, 5=ASCII Character.

- **SUB** → Format: A1 = A2-A3. Subtracts A3 from A2 and saves the result in A1.
- **MUL** → Format: A1 = A2×A3.
- **DIV** → Format: A1 = A2/A3. A1 is quotient and A3 cannot be zero.
- **MOD** → Format: A1 = A2%A3. A1 is remainder and A3 cannot be zero.

Logical

Notes: Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory). The data format is Word, Double-Word…etc. (no Signed binary, floating point number arithmetic).

OR → Format: A1=A2 | A3. ◦ The truth table as left-side：
Performs the bit-wise "OR" operation of A2 (word) and A3 (word) and saves the result in A1 (word).

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

- Or the bit-wise "OR" operation of A2 (Dword) and A3 (Dword) and saves the result in A1 (Dword).

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

- AND → Format: A1 = A2& A3. The truth table as left-side：
Performs the bit-wise "AND" operation of A2 (word) and A3 (word) and saves the result in A1 (word).
- Or performs the bit-wise "AND" operation of A2 (Dword) and A3 (Dword) and saves the result in A1 (Dword).
- **XOR** → Format: A1=A2 ^ A3. The truth table as left-side：
Performs the bit-wise exclusive **OR** operation of A2 (word) and A3 (word) and saves the result in A1 (word).
- Or performs the bit-wise exclusive **OR** operation of A2 (Dword) and A3 (Dword) and saves the result in A1 (Dword).
- **SHL** → Format: A1=A2 << A3. Shifts A2 (word) left by A3 bits and saves the result in A1 (word). left shift command fills 0 into bit 0 and the last bit will shift out. If the displacement(A3) is greater than 16 ,then 16 will be the most shiftable amount.
- Or shift A2 (Dword) left by A3 bits and saves the result in A1 (Dword). Left shift command fills 0 into bit 0 and the last bit

will shift out. If the displacement(A3) is greater than 32 ,then 32 will be the most shiftable amount.

- **SHR** → Format: A1=A2 >> A3. Shifts A2 (word) right by A3 bits and saves the result in A1 (word). Right shift command fills 0 into bit 15 and the last bit will shift out. If the displacement(A3) is greater than 16 ,then 16 will be the most shiftable amount.

- Or shifts A2 (Dword) right by A3 bits and saves the result in A1 (Dword). Right shift command fills 0 into bit 31 and the last bit will shift out. If the displacement(A3) is greater than 32 ,then 32 will be the most shiftable amount.

Data transfer

[Notes]: Both MOV and BMOV command can be located in the PLC memory or internal memory. These include @, RCPW, CB, RCPNO and *@ (indirect internal memory). The data format of this command is Word.

- **MOV** →Format: A1 (Word) =A2 (Word), A1 (Dword) = A2 (Dword). The **MOV** command is to copy the value of A2 to A1 and the value of A2 is fixed. If A1 is located in PLC, it represents the A2 data in HMI internal register will write in PLC. If A2 is located in PLC, it represents the A2 data will be read and shift to HMI internal register A1.

- **BMOV** → Format: BMOV (A1, A2, A3).Copy a block of data starting at A2 to the memory block starting at A1. A3 specifies the number of Words to be copied. The data format is Word. The **BMOV** command is to copy a block of A3 starting at A2 to the A3 block starting at A1, and the A2 data is fixed. The number of A3 must be between 2 and 524. Format: BMOV (A1, A2, A3)

- **FILL** → Format: FILL (A1, A2, A3). Fill a block of memory starting at A1 with the value of A2. A3 specifies the number of words to be filled. The data format is Word. The **Fill** command is to fill A2 data into a block of A3 starting at A1, A2 data is fixed. The number of A3 must be between 2 and 524.

- **CHR** → Format: CHR (A1, "A2"). Copy the character string A2 to A1. The data of A1 is in ASCII format.

- **GETX** → Format: **A1 = @X**. Convert @X input signal to A1. The A1 data is numeric value. This command is only applicable on PWS520S model.

- **SETY** → Format: **@Y = A1**. Covert A1 to output signal. The A1 data is numeric value. This command is only applicable on PWS520S model.

- **X2Y** → Format: **X2Y (A1, A2)**. Convert input to output. This command is only applicable on PWS520S model.

If A2 value is not zero, the specified internal bit A1 is enabled. When A2 value is zero, the action of the specified internal bit A1 is disabled. There are eight internal bits (from 0 to 7) in PWS-520S. A internal bit can assign the corresponding input point and output point which the sensitive time needs to greater than 20 msec. When cancel the action internal point, the relative digital output point will be reset.

Comparison

[Notes]: Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory).

- IF ==→ Format: IF A1 == A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is equal to A2.

- IF!= → Format: IF A1!=A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is not equal to A2.

- IF > → Format: IF A1>A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is greater than A2.

- IF >= → Format: IF A1>=A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is greater than or equal to A2.

- IF < → Format: IF A1<A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is less than A2.

- IF <= → Format: IF A1<=A2 THEN GOTO LABEL A3. Goes to LABEL A3 if A1 is less than or equal to A2.

- IF AND == 0 → Format: IF (A1&A2) == 0 THEN GOTO LABEL A3. Goes to LABEL A3 if the result of AND operation of A1 and A2 is 0.

- IF AND! = 0 → Format: IF (A1&A2)! = 0 THEN GOTO LABEL A3. Goes to LABEL A3 if the result of AND operation of A1 and A2 is not 0.
- IF == ON → Format: IF A1==ON THEN GOTO LABEL A2. If bit A1 is ON (1) then goes to LABEL A2.
- IF ==OFF → Format: IF A1==OFF THEN GOTO LABEL A2. If bit A1 is OFF (0) then goes to LABEL A2.
- IF condition THEN DO Macro ENDIF → If the condition is true, the HMI will execute the Macro. See below: :

1	IF @100 == 50 THEN DO
2	SETB Y0
3	SETB Y1
4	ENDIF
5	END

1. If @100 is equal to 50, then the following Macro will be executed.
2. Force PLC Output Y0 to be 1(ON).
3. Force PLC Output Y1 to be 1(ON).
4. End of the IF condition.
5. End of the Macro.

- IF condition THEN DO Macro-A ELSE DO Macro-B ENDIF → If the condition is true, the HMI will execute Macro-A. If the condition is not true, the HMI will execute Macro-B. See below:

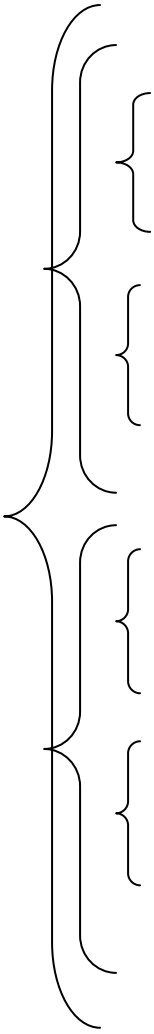
1	IF @100 == 50 THEN DO
2	SETB Y0
3	SETB Y1
4	ELSE DO
5	CLRB Y0
6	CLRB Y1
7	ENDIF
8	END

1. If @100 is equal to 50, then the following Macro will be executed.
2. Force PLC Output Y0 to be 1(ON).
3. Force PLC Output Y1 to be 1(ON).
4. If @100 is not equal to 50, the following Macro will be executed.
5. Force PLC Output Y0 to be 0(OFF).
6. Force PLC Output Y1 to be 0(OFF).
7. End of the If condition.
8. End of the Macro.

- IF condition THEN DO

  IF-THEN-ELSE-A

  ELSE DO



```
1  IF @100 > 50 THEN DO
2  IF @100 > 80 THEN DO
3  SETB Y3
4  CLRB Y0
5  CLRB Y1
6  CLRB Y2
7  ELSE DO
8  SETB Y2
9  CLRB Y0
10 CLRB Y1
11 CLRB Y3
12 ENDIF
13 ELSE DO
14 IF @100 > 30 THEN DO
15 SETB Y1
16 CLRB Y0
17 CLRB Y3
18 CLRB Y2
19 ELSE DO
20 SETB Y0
21 CLRB Y3
22 CLRB Y1
23 CLRB Y2
24 ENDIF
25 ENDIF
26 END
```

1. If @100 is greater than 50, then the following Macro will be executed.
2. If @100 is greater than 50 and @100 is greater than 80, then the following Macro will be executed.
3. Force PLC Output Y3 to be 1(ON).
4. Force PLC Output Y0 to be 0(OFF).
5. Force PLC OutputY1 to be 0(OFF).
6. Force PLC OutputY2 to be 0(OFF).
7. If @100 is greater than 50 but @100 is less than 80, then the following Macro will be executed.
8. Force PLC OutputY2 to be 1(ON).
9. Force PLC OutputY0 to be 0(OFF).
10. Force PLC OutputY1 to be 0(OFF).
11. Force PLC OutputY3 to be 0(OFF).
12. End of the No. 2 IF condition.
13. If @100 is not greater than 50, then the following Macro will be executed.
14. If @100 is not greater than 50 and @100 is less than 30, then the following Macro will be executed.
15. Force PLC OutputY1 to be 1(ON).
16. Force PLC OutputY0 to be 0(OFF).
17. Force PLC OutputY3 to be 0(OFF).
18. Force PLC OutputY2 to be 0(OFF).
19. If @100 is not greater than 50 and @100 is not greater than 30, then the following Macro will be executed.
20. Force PLC OutputY0 to be 1(ON).
21. Force PLC OutputY3 to be 0(OFF).
22. Force PLC OutputY1 to be 0(OFF).
23. Force PLC OutputY2 to be 0(OFF).
24. End of No.14 IF condition.
25. End of No. 1 IF condition.
26. End of Macro.

條件成立，則執行另一個IF-THEN

If N-ELSE-A is not true, IF-THEN-ELSE-B will be executed. See below:

♦ IF condition-1 THEN DO

Macro-A

ELIF condition-2 THEN DO

Macro-B

ELIF condition-3 THEN DO

Macro-C

ENDIF

→ If condition-1 is true, Macro-A will be executed; if condition-1 is not true but condition-2 is true, Macro-B will be executed; if condition-1 and condition-2 are not true but condition-3 is true, Macro-C will be executed. See below:

1	IF @100 > 10 THEN DO
2	CALL 10
3	ELIF @100 > 5 THEN DO
4	CALL 5
5	ELIF @100 > 1 THEN DO
6	CALL 1
7	ENDIF
8	END

1. If @100 is greater than 10, then the following Macro will be executed.
2. Call Sub-macro 10.
3. If @100 is not greater than 10 but @100is greater than 5, the following Macro will be executed.
4. Call sub-Macro 5.
5. If @100 is not greater than 10 but @100 is not greater than 5 and @100 is greater than 1, then the following Macro will be executed.
6. Call sub-Macro 1.
7. End of If condition.
8. End of Macro.

Flow Control

[Notes]: Only internal memory can be used in these commands.

- GOTO → Format: GOTO LABEL A1. Goes to LABLE A1 unconditionally. The GOTO command will cause a branch to the specified label (Label A1). And LABEL A1 must be in the Macro.

- LABEL → Format: LABEL A1. Note that no two labels are allowed to have the same number in one Macro but the same number in Macros is acceptable.

- CALL → Call Sub-macro. Format: CALL A1. Call command can assign the control to sub-macro. The common use of sub-macro is to execute some specific functions, passing the parameter table, complex instruction set…etc. Note that the specified sub-macro must exist and return by RET command where the end of the sub-macro. Then the next macro will be executed. The number of sub-macro is from 001 to 512, and allowed to name. Besides, sub-macro is assigned to CALL another sub-macro.

- RET → Return to Macro. RET command only exists in Sub-Macro but Call exist in Macro. Each RET command must have a corresponding CALL command.

- FOR..NEXT → Loop, "FOR" is the start of a loop and "NEXT" is the end of a loop. Note that the maximum number of FOR loop is 3, e.g. FOR A1. .NEXT. FOR loop is formed by the pair of FOR and NEXT commands, and executes the macro instructions within the FOR loop by A1 times. A1 can be variable and constant. When A1 is 0, the Macro will skip the FOR loop and execute the next line of code after the Next command. When A1 is greater than 0, the Macro will execute A1 continuously until the end of the FOR loop. And a user can change the A1 value within the FOR commands. Note that if A1 is too great, CPU will overload and malfunction.

- FOR/NEXT loop command can execute the program repeatedly. Each FOR command must have one corresponding NEXT command. One is allowed to have up 3 nested FOR loops, e.g. FOR @1…, FOR @2…, FOR@3...NEXT,NEXT,NEXT.

- END → End the Macro. END command represents the end of the Macro. The Macro will not execute the next line of code after the END command and start to the first line of code next time.

Note: END command represents the end of the MACRO, it is invalid in SUB-MACRO. SUN-MACRO must use RET command; otherwise, the program will cause errors.

Data Conversion

[Notes]: Only internal memory can be used in these commands. The internal memory includes @, RCPW, CB, RCPNO and *@ (indirect internal memory).

- BCD → Convert BIN to BCD. Format: A1 = BCD (A2). This command is used to convert A2 (Integer, Word or Dword) from a binary number to a BCD number and saves the result in A1. The valid integer of A2 is between 0 and 9999 (Word) or 0 and 99999999 (Dword).

- BIN → Convert BCD to BIN. Format: A1 = BIN (A2). This command is to convert A2 from a BCD number (Word or Dword) to a binary number and saves the result in A1 (Integer,Word or Dword). The valid BCD number is between 0 and 9999 (Word), or 0 and 99999999 (Dword).

- W2D → Convert WORD to DOUBLE WORD. Format: A1 = W2D (A2). The W2D command is to convert A2 from a WORD number (Integer) to a DOUBLE WORD (Integer) and saves the result in A1 (Dword, signed or unsigned). The valid integer of A2 is between 0 and 65535 (Word,unsigned) or -32768 and 32767 (Word,signed). This function can extend the size of 16 bits signed integer(Word) to 32 bits integer(DWord).

- B2W → Convert BYTE to WORD. Format: A1 = B2W (A2, A3). The byte array is starting at A2 with the size A3 and the result is saved in the memory starting at A1 (WORD). The high bytes of the word array are set to be 0.

- W2B → Convert WORD to BYTE. Format: A1 = W2B (A2, A3). The word array is starting at A2 with the size of A3. The result is saved in the memory starting at A1. The conversion will discard the high bytes of the A2 word array.

- SWAP → Swap the Bytes, Format: SWAP (A1, A2). The SWAP command is to swap the low byte and high byte of each word of a memory block starting at A1. A2 specifies the size of the memory block in word. After executing, the Al data will be changed.

- MAX → Maximum. Format: A1 = MAX (A2, A3). Sets A1 to the larger one in A2 and A3. (The data format can be in word, dword, signed binary or unsigned binary.)

- MIN → Minimum. Format: A1 = MAX (A2, A3). Sets A1 to the smaller one in A2 and A3. (The data format can be word, dword, signed binary or unsigned binary.)

- A2H → Convert 4-digit hex number in ASCII character form into a binary number. Format: A1 = A2H (A2). The character of the fourth digit is in word A2 and the characters of the other digits are in the words following A2 in sequence. The result will be saved in A1. For example, suppose A2 as @200 and the data in @210=9538H. After the conversion, the result will be saved in A1=@210 and it is @200=0039H, @201=0033H, @202=0035H and @203=0038H. (The data format is only in word.)

- H2A → Convert a 16-bit binary number into a 4-digit hex number in ASCII character form. Format: A1 = A2H (A2). The number to be converted is in A2. The character of the fourth digit will be saved in A1 and the characters of the other digits will be saved in the words following A1 in sequence. For example, suppose A2 as @100 and the data in @100=1234H. After the conversion, the result will be saved in A1=@110 and it is @110=0031H, @111=0032H, @112=0033H and @113=0034H. (The data format is only in word.)

Bit Setting

Both internal memory and PLC bit can be used including @nnn.b，RCPWnnn.b in these commands.

- SETB →Set bit to be ON. Format: SETB A1。

- CLRB →Set bit to be OFF. Format: CLRB A1。

- INVB →Inverse the bit state. Format: INVB A1.

Others

There are three special commands to use.

TIMETICK → Get the current system time tick (CPU internal clock time). Format: A1= TIMETICK (). The system time tick is increased by 1 in every 100 ms.

- COMMENT → This is a non-executable instruction and its purpose is to make comments in macro.
- SYS → There are a number of system services which can be used in **SYS** command. please refer to the complete detail in the following:

A) SET_TIMER →Specify the internal timer. Format: SYS(SET_TIMER,N) ◦

@N : Time number. The number of N is between 0 and 7.

@N+1: Current Timer Value.

@N+2: Timer Limit.

@N+3:Time-up Flag.

@N+4: Type of Operation as below:

0 Timer will stop when reach the default setting, the flag will set to be 1.

1 Timer resets to be 0 automatically when the flag is changed to be 0 or 1. When the flag is 1, the timer resets to be 0 automatically. When the flag is 0, the timer resets to be 0 automatically.

2 (PWS-520S only) Does the operations in Type 0 and sets the corresponding digital output Yn, where n = 0~7.

3 (PWS-520S only) Does the operations in Type 1 and toggles the corresponding digital output Yn; where n=0~7.

B) STOP_TIMER →Stops the internal timer . Format: SYS (STOP_TIMER,N).

C) SET_COUNTER → Set the internal counter, it is only applicable on PWS520S.

@N: Counter number. The number of N is between 0 and 7.

@N+1: Current Counter Value .

@N+2: Counter Limit.

@N+3: Over-Limit Flag.

@N+4: Type of Operation as below:

0 Counter will stop when reach the default setting, the flag will set to be 1.

1 Counter resets to be 0 automatically when the flag is

changed to be 0 or 1. When the flag is 1, the counter resets to be 0 automatically. When the flag is 0, the counter resets to be 0 automatically.

2 (PWS-520S only) Does the operations in Type 0 and sets the corresponding digital output Yn, where n = 0~7.

3 (PWS-520S only) Does the operations in Type 1 and toggles the corresponding digital output Yn; where n=0~7.

D) STOP_COUNTER → Stop the internal counter. Format: SYS(STOP_COUNTER,N).

E) WAIT_TIMER→ Wait for the time-up event in the internal timer. Format: SYS (WAIT_TIMER,N). The macro instruction following this command will not be executed until the timer reaches to the Timer Limit. Remember that the corresponding timer must be activated by the SET_TIMER service before requesting this service.

F) WAIT_COUNTER→ Wait for the over-limit event in the internal counter, it is only allocable on PWS520S. Format: (WAIT_COUNTER,N). The macro instruction following this command will not be executed until the counter reaches to the Counter Limit. Remember that the corresponding counter must be activated by the SET_COUNTER service before requesting this service.

G) INIT_COM → Select and initialize a COM port. Format: SYS (INIT_COM,N). The word @n specifies the communication setting of the COM port. The format of the setting is shown in the following:

Bit 1,Bit 0 →DATA Bit S 10：7 Bit S,11：8 Bit S.

Bit 2 →STOP Bit S 0：1 位元,1：2 Bit S.

Bit 4, Bit 3 →PARITY.> 00：NONE, 01：ODD, 11：EVEN.

Bit 6, Bit 5 →COM PORT > 00：COM1, 01：COM2, 10：COM3, 11：COM4.

Bit 7 → Not used.

Bit11,Bit 10,Bit 9,Bit 8 →0001 ： 115200, 0010 ： 57600, 0011 ： 38400, 0110：19200, 1100：9600, Others：4800.

Bit 15 → Computer Protocol Driver ; 0: Disable, 1: Enable (This command is only applicable on PWS1711-Macro, PWS1711-Color, PWS1760, PWS3260, PWS3760)

If this service is successful, then the word @n+1 will be set to be 1; otherwise, it will be set to be 0.

There are some models (e.g. PWS1711-Macro and PWS1711-Color) providing Computer Protocol slave driver for the second com port; This function provides the communication between PC/another PWS and PWS1711-Macro and PWS1711-Color in the second com port. The HMI can communicate with PLC over the first com port and the connection steps are the same as usual steps. The PC can read from the internal registers of PWS1711 @0-@639 over the second port (The data of W0-W639 is correspondent with @0-@639). For PWS1762, PWS3160, PWS3760, the PC can read from the internal registers @0-@2047 (The data of W0-W2047 is correspondent with @0-@2047). To communicate with SoftPanel or another PWS, the controller/PLC must be Computer (as slave) and the PLC station must set as the PWS station and the related format must be the same as the format of INIT_COM.

When use the Computer Protocol driver, this function is inapplicable on GET_CHAR, GET_CHARS, PUT_CHAR, and PUT_CHARS.

H) GET_CHAR → Gets a character from the COM port. Format: SYS (GETCHAR,N). The character will be saved in the low byte of the word @n. If there is no input, the word @n will be set to be -1(ffffH).

I)GET_CHARS → Gets a number of characters from the COM port. Format: SYS (GETCHARS,N). The word @n specifies the maximum number of characters to receive. The actual number of characters received is then saved in word @n+1. The characters received will be saved in the low bytes of the word @n+2, @n+3, @n+4 and so on.

J) PUT_CHAR → Sends a character in the low byte of word @n to the COM port. Format: SYS (PUTCHAR,N). If this service is successful, then the word @n+1 will be set to be 1; otherwise, it will be set to be other value.

K) PUT_CHARS →Sends the characters in the low bytes of the words starting from @n+2 to the COM port. Format: SYS (PUTCHARS,N). Also, the word @n specifies the number of characters to be sent and the actual number of characters sent is then saved in the word @n+1.

L) SUM_ADD → Calculates the sum of a block of words by normal arithmetic addition. Format: SYS (SUM_ADD,N). The output data will save in "@N+3", this feature offers more convenient application

for Macro. For example, SYS (SUM_ADD,30) represents N = 30, this command will calculate the sum of @30,@31,@32,@33 internal registers.

@N=30 represents the pointer parameter, and the internal value of @30 must be 0.

@N+1(@31) represents the starting address of the block.

@N+2(@32) represents the size of WORDS block.

@N+3(@33) represents the initial value of the summand and the sum will save in this address automatically. A user must set the summand before execution. Most communication protocols regulate the initial value of summand = 00H or FFH, please refer to initial value assigned by vendor.

M) SUM_XOR → Calculate the sum of a block of words by the bit-wise logical exclusive-or operation and save the result in the specified address. Format: SYS(SUM_XOR,N). The output data will save in "@N+3", this feature is convenient for Macro communication application. For example, SYS(SUM_XOR,50) represents N = 50, this command will calculate the sum of @50,@51,@52,@53 internal registers. To execute this command requires the internal value of @50, @51, @52, and @53.

@N=50 represents PLC station number, and the internal value of @50must be 0 if no PLC station is required.

@N+1(@51) represents the starting address of the block.

@N+2(@52) represents the size of WORDS block.

@N+3(@53) represents the initial value of the summand and the sum will save in this address automatically. A user must set the summand before execution. Most communication protocols regulate the initial value of summand = 00H or FFH, please refer to initial value assigned by vendor.

N) READ_WORDS → Read a number of words from PLC word devices or internal memory and save the result in the specified address. Format: SYS (READ_WORDS,N). The data will save in "@+5". This command is powerful for the communication with the any PLC registers and can be used as PLC data setting and monitor. For example, SYS(READ_WORDS,80) represents N = 80. To execute this command requires the internal value of @80, @81, @82, @83, @84, @85, @86.

@N (@80) represents PLC station number, and the internal value of @80 must be 0 if no PLC station is required.

@N+1(@81) represents the device type setting. For the device type of PLC, please refer to Chapter 9. for the complete details.

@N+2(@82) represent the low word of the device address.

@N+3(@83) represent high word of the device address.

@N+4(@84) represent auxiliary address if required or set to be 0.

@N+5(@85) represent the address of the internal memory to receive the data and the size of data is specified by N+6(@86).

@N+6(@86) represent the number of words to be read.

O) READ_Bit → Read a PLC bit device or internal bit and save the data in the specified address. Format: SYS (READ_Bit,N). The data will save in "@+5". This command is powerful for the communication with the any PLC bit-state and can be used as PLC data setting and monitor. For example, SYS(READ_bit,80) represents N = 80. . To execute this command requires the internal value of @80, @81, @82, @83, @84, and @85.

@N(@80) represents PLC station number, and the internal value of @80 must be 0 if no PLC station is required.

@N+1(@81) represents the device type. For the device type of PLCs, please refer to Chapter 9. for the complete details.

@N+2(@82) represents low word of the device address.

@N+3(@83) represents high word of the device address.

@N+4(@84) represents auxiliary address if required or set 0.

@N+5(@85) represents the address of the internal memory to receive the data. N+5(@85) DATA = 1 if the bit is ON; DATA = 0 if the bit is OFF.

P) WRITE_WORDS → Writes a block of data in internal memory to PLC word devices or internal memory. Format: SYS(WRITE_WORDS,N). The data will save in "@N+5". This command is powerful for the random modification of the any PLC data and can be used as PLC data setting and monitor. For example, SYS(WRITE_WORDS,90) represents N = 90. To execute this command requires the internal value of @90, @91, @92, @93, @94, @95 and @96.

@N(@90) represents PLC station number, and the internal value of @90 must be 0 if no PLC station is required.

@N+1(@91) represents the device type. For the device type of PLCs, please refer to Chapter 9. for the complete details.

@N+2(@92) represents low word of the device address.

@N+3(@93) represents high word of the device address.

@N+4(@94) represents auxiliary address if required or set 0.

@N+5(@95) represents the source address and the size of a continuous block of data is assigned by N+6 (@96).

@N+6(@96) represents the number of words of the data.

Q) WRITE_Bit → Set a PLC bit device or internal bit to the state of an internal word. Format: SYS(WRITE_Bit,N). The source address is "@+5". This command is powerful for the random modification of the any PLC data and can be used as PLC data setting and monitor. For example, SYS(WRITE_Bit,90) represents N = 90. To execute this command requires the internal value of @90, @91, @92, @93, @94, and @95.

@N(@90) represents PLC station number, and the internal value of @90 must be 0 if no PLC station is required.

@N+1(@91) represents the device type. For the device type of PLCs, please refer to Chapter 9. for the complete details.

@N+2(@92) represents low word of the device address.

@N+3(@93) represents high word of the device address.

@N+4(@94) represents auxiliary address if required or set 0.

@N+5(@95) represents the address of the internal memory to receive the data. N+5(@95) DATA = 1 if the bit is ON; DATA = 0 if the bit is OFF.

8.4. Cautions

The last line of code must be RET command, otherwise it will cause error when COMPILE.

Except for Sub-Macro, the END Macro represents the end of the Macro.

CPU will execute other program after the execution of INITIAL MACRO, CLOCK MACRO, ON/OFF MACRO, OPEN MACRO, and CLOSE MACRO.

For BACKGROUND MACRO, CYCLIC MACRO, Sub-MACRO, CPU executes the 30 rows of command once; then the CPU will execute other program. The CPU will execute the 30 rows of command which follows the last executed command until the next cycle.

To use the Macro communication function, a user must define the related communication format for INICOM. This command is only used once, so it is usually edited in INITIAL MACRO.

8.5. Internal Memory

The HMI provides some internal registers for read/write. With these internal registers, a user can operate the Macro more efficiently and conveniently. The internal registers can not only enhance Macro with infinite functions but also hold a lot of arithmetic data source and result. Note that this system provides the internal registers divided into RAM and ROM.

We will details the four types of internal memory in the following:

WORDS	Device Type	Size	Address	Aux. Address	R/W
RCPNO	0x80	W	0 (only one word)	0	R/W
RCPWn	0x81	W	0-?	0	R/W
CBn	0x82	W	0-31	0	R
@n	0x85	W	0-10239 (for PWS3261)	0	R/W
@n	0x85	W	8191 (for PWS6000 series)	0	R/W

位元	Device Type	Address	Aux. Address	R/W
CBn.b (b=0-f)	0x83	0-31	0-15	R
RCPWn.b (b=0-f)	0x84	0-?	0-15	R/W
@n.b (b=0-f)	0x86	0-10239 (for PWS3261)	0-15	R/W
@n.b (b=0-f)	0x86	0-8191 (for PWS6000 series)	0-15	R/W

1. RCPNO.

2. The n value of RCPWn is based on the size of recipe and the maximum number. The data register can be used as bit.

3. The n value of CBn is based on the size of the control block. The current size is 2~32. This data register can be used as bit.

4. @n: Internal Register. The size of n is based on the HMI model. The size of PWS500S, PWS1711 is 640 WORDS (n=0~639), and the size of PWS1760, PWS3760, SoftPanel is 10240 WORDS(n=0~10239). This data register can be used by bit.