Macro Function Guide

Macro Function

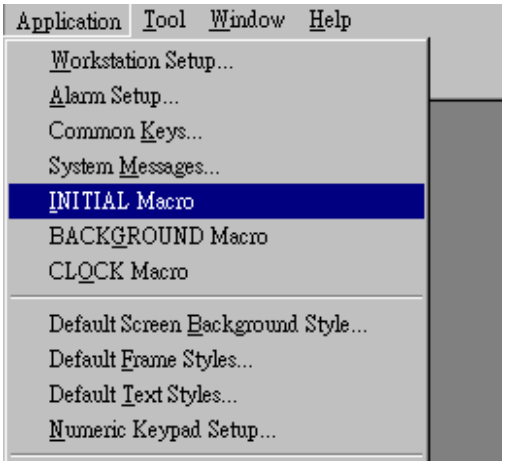
ADP3 provides a convenient macro function that executes Arithmetic, Logical, Data Transfer, Comparison, Flow Control, Data Conversion, system service instructions, etc. The workstation communicates with PLC and meanwhile link with other devices by another communication port through macro function. The macro function is the most economic way to integrate systems and it saves PLC program size by using macro instructions.

Outline of The Macro Function

The macro function provides different macro instructions according to different uses.

Application Macro

There are three different types of macro to be executed when the application is start, and while the application is running:



1). Initial Macro: An application can have one Initial Macro. It is executed only once when the application is started. The purpose of Initial Macro is to initialize data and communication settings. The start-up screen will not display until the Initial Macro is executed.

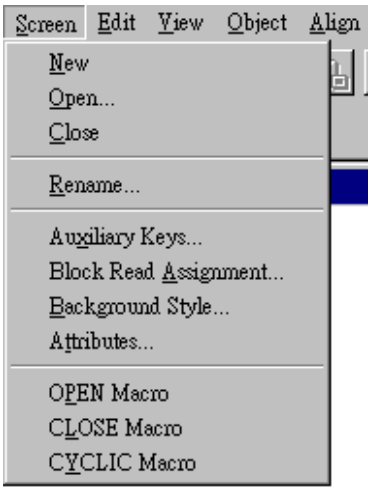
2). Background Macro: An application can have one Background Macro. The Workstation executes the Background Macro cyclically and processed at most N commands in every T milliseconds. N is 30 and T is 100 for SoftPanel. The Workstation stops executing the macro when reaches the end of macro or encounters an END command and will execute the macro from the first command in the next T milliseconds.

3). Clock Macro: An application can have one Clock Macro. It is executed every T milliseconds after the application is started. T is 100 for SoftPanel. Unlike the Background Macro, the Workstation

will not do anything else until the Clock Macro is completely executed. Therefore, you should keep the Clock Macro as small as possible so that it does not degrade the performance of the Workstation too much.

Screen Macro

The three different types of macro to be executed when the screen is opened, closed, and while the screen is open are:



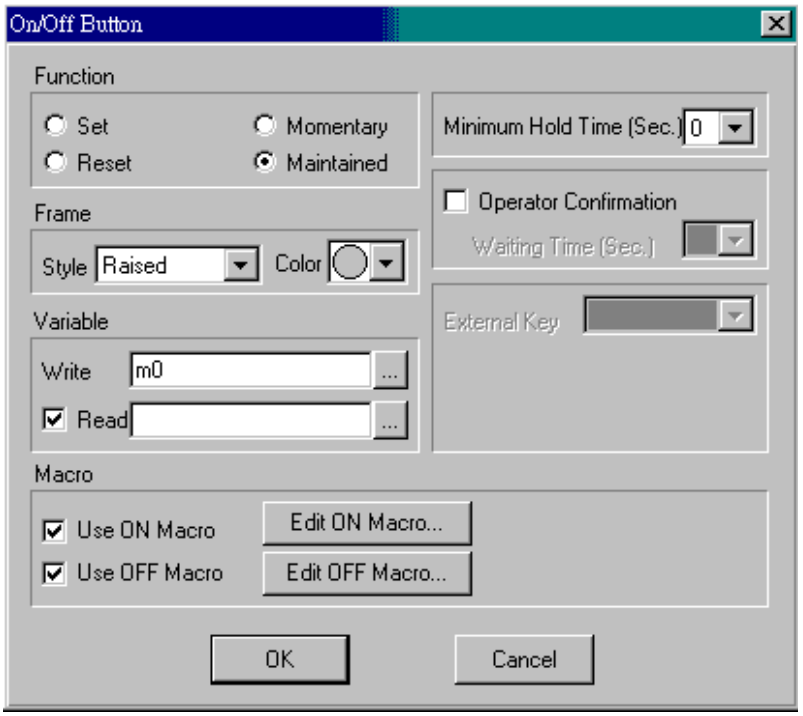
4). Open Macro: A screen can have one Open Macro. An Open Macro is executed when its owner is selected to open. A screen will not display until its Open Macro is completely executed.

5). Close Macro: A screen can have one Close Macro. A Close Macro is executed when its owner is selected to close. A screen will not be erased until its Open Macro is completely executed.

6). Cyclic Macro: A screen can have one Cyclic Macro. A Cyclic Macro will be executed when its owner is displayed. The Workstation executes a Cyclic Macro cyclically and processed at most N commands in T milliseconds. The N is 30 and the T is 100 for SoftPanel. The Workstation stops executing the macro when reaches the end of macro or encounters an END instruction and will execute the macro from the first command in the next T milliseconds.

ON/OFF Macro

There are two different modes of macro to be executed when the operator touches or releases the push-button are:



7). ON Macro: Each on/off button except Reset button can have one On Macro. An On Macro is executed when its owner is pressed to set a bit to on. The bit will not be set to on until the On Macro is completely executed. Therefore, you should keep an On Macro as small as possible in order not to delay the time to change the bit.

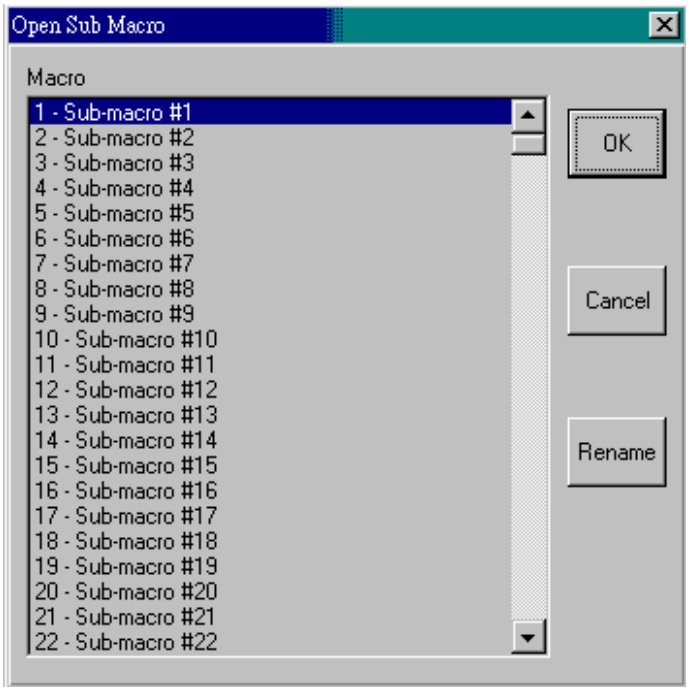
8). OFF Macro: Each on/off button except Set button can have one Off Macro. An Off Macro is executed when its owner is pressed or released to set a bit to off. The bit will not be set to off until the Off Macro is completely executed. Therefore you should keep an Off Macro as small as possible in order not to delay the time to change the bit.

Sub-Macro

The macros which are edited in "Sub-Macro" editing can be executed by CALL commands. RET command returns the sequence to the original macro command.

Up to 512 Sub-Macro can be set.

Up to 512 commands (lines) can be edit for one macro.



9). Sub-macro: A Sub-macro is a macro that your other macros can ask the Workstation to execute it by using CALL command. The last command of a Sub-macro must be a RET command. You can also use a RET command to stop the execution of a Sub-macro at any point you want. The Workstation will process the macro command following the CALL command when the invoked Sub-macro stops.

Macro Commands

Table of Macro commands

Operation	Format	A1*	A2*	A3*	Data	PLC Data
ADD	A1=A2+A3	2	2,4	2,4	DW/Signed	X
SUB	A1=A2-A3	2	2,4	2,4	DW/Signed	X
MUL	A1=A2xA3	2	2,4	2,4	DW/Signed	X
DIV	A1=A2/A3	2	2,4	2,4	DW/Signed	X
MOD	A1=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
IF==	IFA1==A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF!=	IFA1!=A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF>	IFA1>A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF>=	IFA1>=A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF<	IFA1<A2 GOTOA3	2,4	2,4	4	DW/Signed	X
IF<=	IFA1<=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	IFA1=ON GOTO A2	3	4	~	Bit	X
IF==OFF	IFA1=OFFGOTO A2	3	4	~	Bit	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

Operation	Format	A1*	A2*	A3*	Data	PLC Data
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
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
	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_BITS,N)		4			X
	SYS(WRITE_WORDS,N)		4			X
	SYS(WRITE_BIT,N)		4			X
	SYS(SUM_ADD,N)		4			X
	SYS(SUM_XOR,N)		4			X

*Available settings for A1, A2, and A3.

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

A>Arithmetic

Notes: Only internal memory can be located in those commands. @, RCPW, CB, RCPNO, *@(indirect internal memory). The data format can be word, double-word, signed binary, and unsigned binary

Operation: ADD

Format: A1 = A2 + A3

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

Operation: SUB

Format: A1 = A2 – A3

Description: Subtracts A3 from A2 and saves the result in A1.

Operation: MUL

Format: A1(Dword)=A2(word) x A3(word)
A1(Dword)=A2(Dword) x A3(Dword)

Description: Multiplies A2 by A3 and saves the product in A1. The product is a double-word number. The low word is saved in A1 and the high word is saved in the word following A1

Operation: DIV

Format: A1(word)=A2(word) / A3(word)
A1(Dword)=A2(Dword) / A3(Dword)

Description: Divides A2 by A3 and saves the quotient in A1, A3 cannot be zero.

Operation: MOD

Format: A1 = A2 % A3

Description: Divides A2 by A3 and saves the remainder in A1, A3 cannot be zero.

B>Logical

Notes: Only internal memory can be located in those commands. @, RCPW, CB, RCPNO, *@(indirect internal memory)

Operation: OR

Format: A1 = A2 | A3

Description: Performs the bit-wise OR operation of A2 and A3 and saves the result in A1.

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

Operation: AND

Format: A1 = A2 & A3

Description: Performs the bit-wise AND operation of A2 and A3 and saves the result in A1.

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

Operation: XOR

Format: A1 = A2 ^ A3

Description: Performs the bit-wise exclusive OR operation of A2 and A3 and saves the result in A1.

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

Operation: SHL

Format: A1 = A2 << A3

Description: Shifts A2 left by A3 bits and saves the result in A1. No. of shifts A3 for single-word is 0 to 15, double words is 0 to 31.

Operation: SHR

Format: A1 = A2 >> A3

Description: Shifts A2 right by A3 bits and saves the result in A1. No. of shifts A3 for single-word is 0 to 15, double words is 0 to 31.

C>
Data Transfer

Notes: Both MOV and BMOV commands can be located in the PLC memory or internal memory. @, RCPW, CB, RCPNO, *@(indirect internal memory)

Operation: MOV

Format: A1(Word)=A2(Word)
A1(Dword)=A2(Dword)
Description: Copies the value of A2 to A1.

Both A1 and A2 can be located in the PLC memory at the same time.

Operation: BMOV

Format: BMOV(A1, A2, A3)

Description: Copies a block of data starting at A2 to the memory block starting at A1. A3 specifies the number of words to be copied.

Both A1 and A2 can be located in the PLC memory, but not at the same time.

Operation: FILL

Format: FILL(A1, A2, A3)

Description: Fills a block of memory starting at A1 with the value of A2. A3 specifies the number of words to be filled.

Operation: CHR

Format: CHR(A1, "A2")

Description: Copies character string A2 to the memory block starting at A1. The data of A1 will be in ASCII code format.

D>Comparison

Notes: Only internal memory can be located in those commands. @, RCPW, CB, RCPNO, *@(indirect internal memory)

Operation: IF ==

Format: IF A1 == A2 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if A1 is equal to A2.

Operation: IF !=

Format: IF A1 != A2 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if A1 is not equal to A2.

Operation: IF >

Format: IF A1 > A2 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if A1 is greater than A2.

Operation: IF >=

Format: IF A1 >= A2 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if A1 is greater than or equal to A2.

Operation: IF <

Format: IF A1 < A2 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if A1 is less than A2.

Operation: IF <=

Format: IF A1 <= A2 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if A1 is less than or equal to A2.

Operation: IF AND == 0

Format: IF (A1 & A2) == 0 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if the result of AND operation of A1 and A2 is 0.

Operation: IF AND != 0

Format: IF (A1 & A2) != 0 THEN GOTO LABEL A3

Description: Goes to LABEL A3 if the result of AND operation of A1 and A2 is not 0.

Operation: IF == ON

Format: IF A1 == ON THEN GOTO LABEL A2

Description: If bit A1 is ON (1) then goes to LABEL A2.

Operation: IF == OFF

Format: IF A1 == OFF THEN GOTO LABEL A2

Description: If bit A1 is OFF (0) then goes to LABEL A2.

E>Flow Control

Notes: Only internal memory can be located in those commands. @, RCPW, CB, RCPNO, *@(indirect internal memory)

Operation: GOTO

Format: GOTO LABEL A1

Description: Unconditionally goes to LABEL A1.

Operation: LABEL

Format: LABEL A1

Description: Specifies a label with the label number A1. No two labels can have the same number in a macro.

Operation: CALL

Format: CALL A1

Description: Goes to the sub-macro specified by A1. The value of A1 specifies the number of invoked sub-macro. The specified sub-macro must exist or unpredictable results may occur.

There can be up to 8 nested CALL loops.

Operation: RET

Format: RET

Description: A sub-macro uses this instruction to return to the macro invoking it. Only sub-macros can have RET.

Operation: FOR

Format: FOR A1

Description: Executes the macro instructions within the FOR loop by A1 times. A FOR loop is formed by a matching pair of FOR and NEXT instructions. There can be up to 3 nested FOR loops. Ex. FOR @1…, FOR @2…, FOR @3… NEXT, NEXT, NEXT.

Operation: NEXT

Format: NEXT

Description: Indicates the end of a FOR loop.

F> Data Conversion

Notes: Only internal memory can be located in those commands. @, RCPW, CB, RCPNO, *@(indirect internal memory)

Operation: BCD

Format: A1 = BCD(A2)

Description: Converts A2 from a binary number to a BCD number and saves the result in A1.

Operation: BIN

Format: A1 = BIN(A2)

Description: Converts A2 from a BCD number to a binary number and saves the result in A1.

Operation: W2D

Format: A1 = W2D(A2)

Description: Converts A2 from a single-word number to a double-word number and saves the result in A1.

Operation: B2W

Format: A1 = B2W(A2, A3)

Description: Converts a byte array starting at A2 with the size specified by A3 to a word array. The result is saved in the memory starting at A1. The high bytes of the word array are set to 0.

Operation: W2B

Format: A1 = W2B(A2, A3)

Description: Converts a word array starting at A2 with the size specified by A3 to a byte array. The result will be saved in the memory starting at A1. The conversion will discard the high bytes of the word array.

Operation: SWAP

Format: SWAP(A1, A2)

Description: Swaps 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 words.

Operation: MAX

Format: A1 = MAX(A2, A3)

Description: Sets A1 to the larger of A2 and A3.
(Data format could be word, Dword, signed binary, unsigned binary)

Operation: MIN

Format: A1 = MIN(A2, A3)

Description: Sets A1 to the smaller of A2 and A3.
(Data format could be word, Dword, signed binary, unsigned binary)

Operation: A2H

Format: A1 = A2H(A2)

Description: Converts a 4-digit hex number in ASCII character form into a binary number. 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.

Ex. A2 as @200 and the data in @200=0039H, @201=0033H, @202=0035H, @203=0038H then after converts the result 9358H will be saved in A1. (only word format can be used).

Operation: H2A

Format: A1 = H2A(A2)

Description: Converts a 16-bit binary number into a 4-digit hex number in ASCII character form. 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.

Ex. A2 as @100 and the data in @100=1234H, then after converts the result 1234H will be saved in A1=@110, @110=0031H, @111=0032H, @112=0033H, @113=0034H. (only word format can be used).

G>Bit Setting

Notes: Both internal memory and PLC bit can be located in those commands.

Operation: SETB

Format: SETB A1

Description: Sets bit A1 to ON (1).

Operation: CLRB

Format: CLRB A1

Description: Sets bit A1 to OFF (0).

Operation: INVB

Format: INVB A1

Description: Inverses the state of bit A1.

H>others

Operation: TIMETICK

Format: A1 = TIMETICK()

Description: Gets the current system time tick and save it in A1. The system time tick has a double-word value and increases by 1 every 100 ms.

Operation: Comment

Format: #A1

Description: This is a non-executable instruction and its purpose is for making comment in macros. A1 is a line of text.

Operation: SYS

Format: SYS(A1, A2)

Description: Requests the system service specified by A1. A2 specifies a block of memory that the system uses to provide the service. The available System Services include:

SET_TIMER
STOP_TIMER
SET_COUNTER
STOP_COUNTER
WAIT_TIMER
WAIT_COUNTER
INIT_COM
GET_CHAR
GET_CHARS
PUT_CHAR
PUT_CHARS
READ_WORDS
READ_BIT
WRITE_WORDS
WRITE_BIT
SUM_ADD
SUM_XOR

Service: SET_TIMER

Format: SYS(SET_TIMER, n)

Description: Initializes and starts the specified internal timer. The use of the data block is shown below:

@n: timer number (specified by programmer)
@n+1: current timer value (changed by the timer)
@n+2: timer limit (specified by programmer)
@n+3: time-up flag (changed by the timer)
@n+4: type of operation (specified by programmer)

An internal timer will use the assigned data block as its private memory before you stop the timer, so don't use any words in the data block for other purposes. No error checking will be performed on the data you pass to the system, so make sure all data specified by you are correct.

Internal Timer: There are eight internal timers available that are numbered 0 through 7. An internal timer increases the current timer value by 1 every 100ms. When the current timer value reaches the timer limit, an internal timer will do the following according to the type of operation:

Type of operation	Doing when current timer value reaches the timer limit
0	Sets the time-up flag to 1 and resets the current timer value to 0

1	Toggles the time-up flag and resets the current timer value to 0
2 (PWS-520S only)	Does the same thing as type 0 and sets the corresponding digital output Yn to 0
3 (PWS-520S only)	Does the same thing as type 1 and toggles the corresponding digital output Yn and n =0~7.

Service: STOP_TIMER

Format: SYS(STOP_TIMER, n)

Description: Stops the internal timer specified by the timer number in @n.

Service: SET_COUNTER

Format: SYS(SET_COUNTER, n)

Description: Initializes and starts the specified internal counter. The use of the data block is shown below:

@n: counter number (specified by programmer)
@n+1: current counter value (changed by the counter)
@n+2: counter limit (specified by programmer)
@n+3: over-limit flag (changed by the counter)
@n+4: type of operation (specified by programmer)

An internal counter will use the assigned data block as its private memory before you stop the counter, so don't use any words in the data block for other purposes. No error checking will be performed on the data you pass to the system, so make sure all data specified by you are correct.

Internal Counter: Only the PWS-520S has internal counters. There are eight internal counters available with the PWS-520S and they are numbered 0 through 7. Internal counter n uses the digital input Xn as the trigger input. A pulse presenting on Xn increases the internal counter n by one. The internal counters can detect a pulse width as narrow as 25 ms. Note that the interval between two consecutive pulses must not be less than 25 ms. When the current counter value increase to the counter limit an internal counter will do the following according to the type of operation:

Type of operation	Doing when current counter value reaches the counter limit
0	Sets the over-limit flag to 1 and resets the current counter value to 0
1	Toggles the over-limit flag and resets the current counter value to 0
2 (PWS-520S only)	Does the same thing as type 0 and sets the corresponding digital output Yn to 0
3 (PWS-520S only)	Does the same thing as type 1 and toggles the corresponding digital output Yn and n =0~7.

Service: STOP_COUNTER

Format: SYS(STOP_COUNTER, n)

Description: Stops the internal counter specified by the counter number in @n.

Service: WAIT_TIMER

Format: SYS(WAIT_TIMER, n)

Description: Waits for the time-up event of the internal timer specified by the timer number in @n. The next macro instruction following this one will not be executed until the timer reaches its timer limit. Before requesting this service, the corresponding timer must be already activated by the SET_TIMER service.

Service: WAIT_COUNTER

Format: SYS(WAIT_COUNTER, n)

Description: Waits for the over-limit event of the internal counter specified by the counter number in @n. The next macro instruction following this one will not be executed until the counter reaches its counter limit. Before requesting this service, the corresponding counter must be already activated by the SET_COUNTER service.

Service: INIT_COM

Format: SYS(INIT_COM, n)

Description: Selects and initializes the COM port for the macro operations. The word @n specifies the COM port and settings of the communications according to the following rule:

1) Bit 11.8: Baud Rate
0001: 115200, 0010: 57600, 0011: 38400, 0110: 19200, 1100: 9600, others: 4800

2) Bit 7 not used

3) Bit 6.5: COM Port
00: COM1, 01: COM2, 10: COM3, 11: COM4

4) Bit 4.3: Parity
00: none, 01: odd, 11: even

5) Bit 2: Stop bits
0: 1 bit, 1:2 bits

6) Bit 1.0: Data Bits
10: 7 bits, 11:8 bits

If the service succeeds the word @n+1 will be set to 1; otherwise it will be set to 0.

Service:

Format: SYS(GET_CHAR, n)

GET_CHAR

Description: Gets a character from the COM port. The received character will be saved in the low byte of the word @n. If there is no input, the word @n will be set to -1(ffffH).

Service: GET_CHARS

Format: SYS(GET_CHARS, n)

Description: Gets a number of characters from the COM port. The word @n specifies the maximum number of characters to receive. The actual number of characters received will be saved in word @n+1. The received characters will be saved in low bytes of @n+2 and the following words.

Service: PUT_CHAR

Format: SYS(PUT_CHAR, n)

Description: Sends the character in the low byte of word @n to the COM port. If the service succeeds the word @n+1 will be set to 1; otherwise it will be set to 0.

Service: PUT_CHARS

Format: SYS(PUT_CHARS, n)

Description: Sends the characters in the low bytes of @n+2 and the following words to the COM port. The word @n specifies the number of characters to be sent. The actual number of characters sent will be saved in @n+1.

Service: SUM_ADD

Format: SYS(SUM_ADD, n)

Description: Calculates the sum of a block of words by the normal arithmetic addition.
@n: must be 0
@n+1: specifies the address of the block
@n+2: specifies the size of the block
@n+3: specifies the initial value of the sum; the sum will be saved here too after the calculation; remember to initialize this word before every calculation.

Service: SUM_XOR

Format: SYS(SUM_XOR, n)

Description: Calculates the sum of a block of words by the bit-wise logical exclusive-or operation.
@n: must be 0
@n+1: specifies the address of the block
@n+2: specifies the size of the block
@n+3: specifies the initial value of the sum; the sum will be saved here too after the calculation; remember to initialize this word before every calculation.

Service: READ_WORDS

Format: SYS(READ_WORDS, n)

Description: Reads a number of words from PLC word devices or internal memory by using general read command.
@n: PLC station number or 0 if no station number is required
@n+1: device type
@n+2: low word of the device address
@n+3: high word of the device address
@n+4: auxiliary address if required
@n+5: specifies the address of the internal memory to receive the data
@n+6: specifies the number of words to be read; the maximum is 255.
Please refer to the Specifying PLC Devices and Internal Devices for the information of device type codes, internal addresses, and internal auxiliary addresses.
n=630 indicates that 7 continuous addresses will be used from internal memory address @630 and each of them has different function as shown below;
(@630, @631, @632, @633, @634, @635, @636)

a. The data in @630 indicates the PLC station number. It should be "0" in the memory @630 when there is no PLC station number.
b. The data in @631 indicates the device type for the Workstation to communicate. Please see table 1 to get more information. For example @631=8 indicates "data register D".
c. The data in double word @633 and @632 indicate the start address of the PLC to be read. For example @632=100, @633=0 indicates the start reading address is D100.
d. The data in @634 indicates the "Auxiliary Address" for the Workstation to communicate. Please see table 1 to get more information.
e. The data in @635 indicates the address of the internal memory to receive the data.
f. The data in @636 indicates the number of words to be read; the maximum is 255.

Service: READ_BIT

Format: SYS(READ_BIT, n)

Description: Reads a PLC bit device or internal bit by using general read command.
@n: PLC station number or 0 if no station number is required
@n+1: device type
@n+2: low word of the device address
@n+3: high word of the device address
@n+4: auxiliary address if required
@n+5: specifies the address of the internal memory to receive the data; the word will be set to 1 if the bit is on or the word will be set to 0 if the bit is off.

Please refer to the Specifying PLC Devices and Internal Devices for the information of device type codes, internal addresses, and internal auxiliary addresses.

n=530 indicates that 6 continuous addresses will be used from internal memory address @530 and each of them has different function as shown below;
(@530, @531, @532, @533, @534, @535)

a. The data in @530 indicates the PLC station number. It should be "0" in the memory @530 when there is no PLC station number.
b. The data in @531 indicates the device type for the Workstation to communicate. Please see table 2 to get more information. For example @531=C0H indicates "internal relay M".
c. The data in double word @533 and @532 indicate the start address of the PLC to be read. For example @532=200, @533=0 indicates the start reading address is M200.
d. The data in @534 indicates the "Auxiliary Address" for the Workstation to communicate. Please see table 2 to get more information.
e. The data in @535 specifies the address of the internal memory to receive the data; the word will be set to 1 if the bit is on or the word will be set to 0 if the bit is off. For example @535=537, @537.0=(ON/OFF) indicates the status of M200(ON/OFF).

Service: WRITE_WORDS

Format: SYS(WRITE_WORDS, n)

Description: Writes the data of a block of internal memory to PLC word devices or internal memory by using general read command.
@n: PLC station number of the destination or 0 if no station number is required
@n+1: device type of the destination
@n+2: low word of the device address of the destination
@n+3: high word of the device address of the destination
@n+4: auxiliary address of the destination if required
@n+5: address of the source
@n+6: specifies the number of words of the data; the maximum is 255.
Please refer to the Specifying PLC Devices and Internal Devices for the information of device type codes, internal addresses, and internal auxiliary addresses.

Service: WRITE_BIT

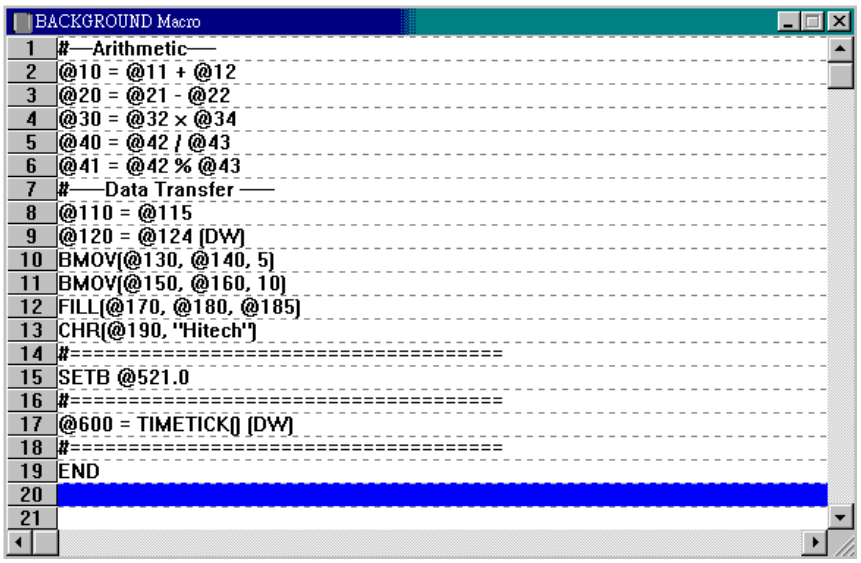
Format: SYS(WRITE_BIT, n)

Description: Sets a PLC bit device or internal bit to the state of an internal word.
@n: PLC station number of the destination or 0 if no station number is required
@n+1: device type of the destination
@n+2: low word of the device address of the destination
@n+3: high word of the device address of the destination
@n+4: auxiliary address of the destination if required
@n+5: address of the source. The state is off if the internal word is 0; otherwise the state is on.

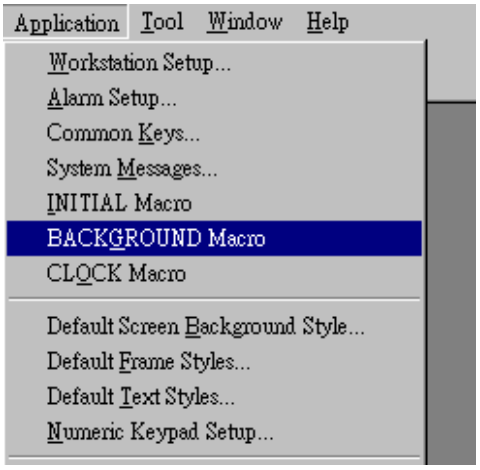
Please refer to the Specifying PLC Devices and Internal Devices for the information of device type codes, internal addresses, and internal auxiliary addresses.

Macro Commands Operate

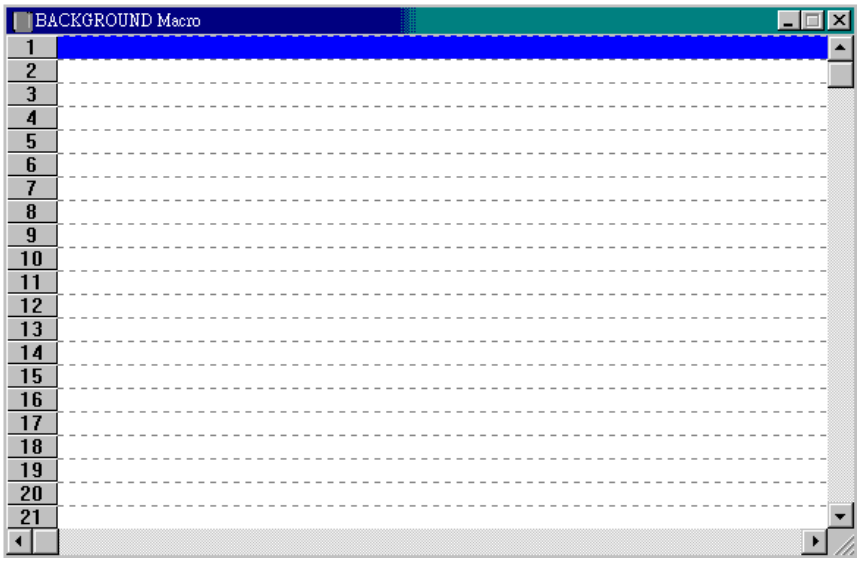
This section illustrates how to use the macro instructions. There is a background macro example below.



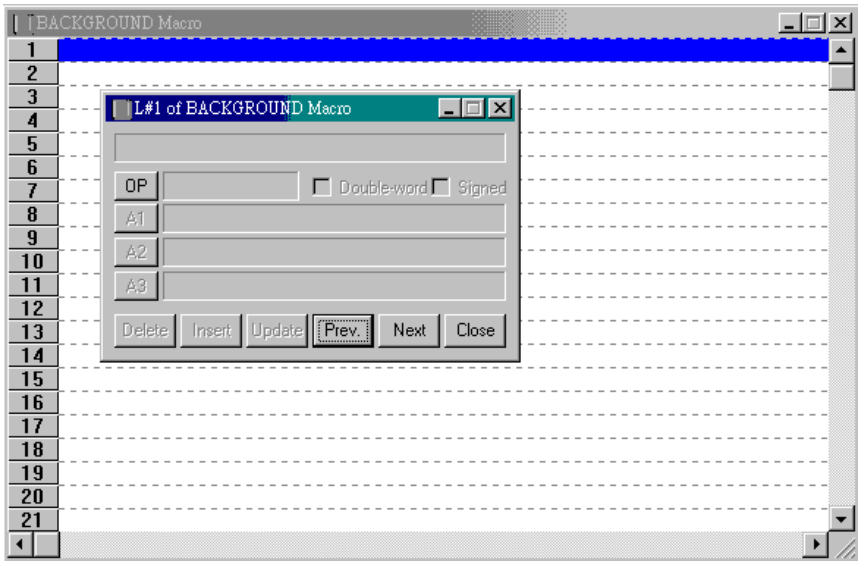
Step 1 chooses “BACKGROUND Macro” from “Application”



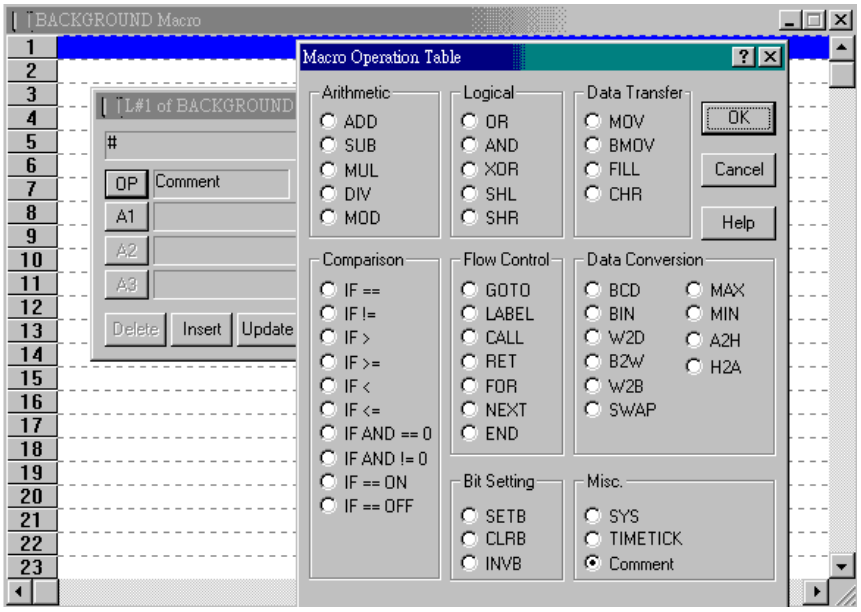
Step 2 A blank BACKGROUND Macro shows on the screen. There are up to 512 instructions.



Step 3 Edit the Macro program. Click first line to see a Macro comment dialogue box.

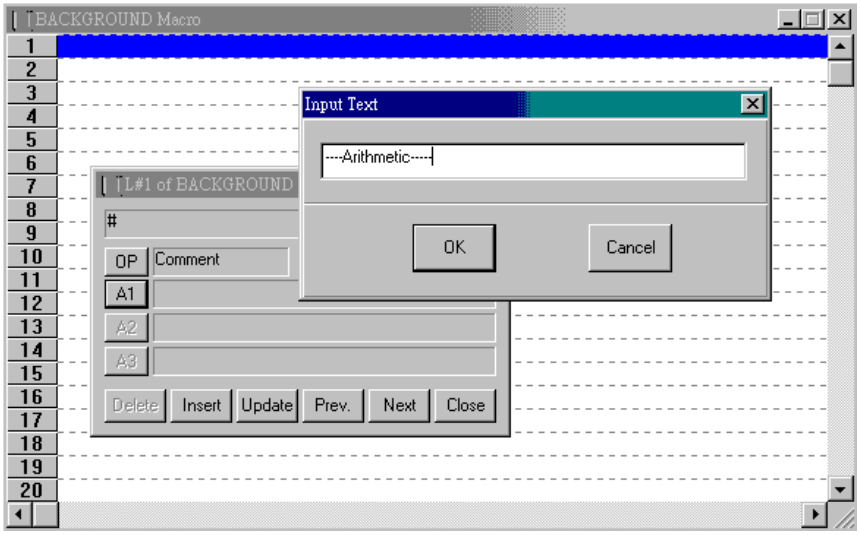


Step 4 Click “OP” to see a Macro Operation Table.

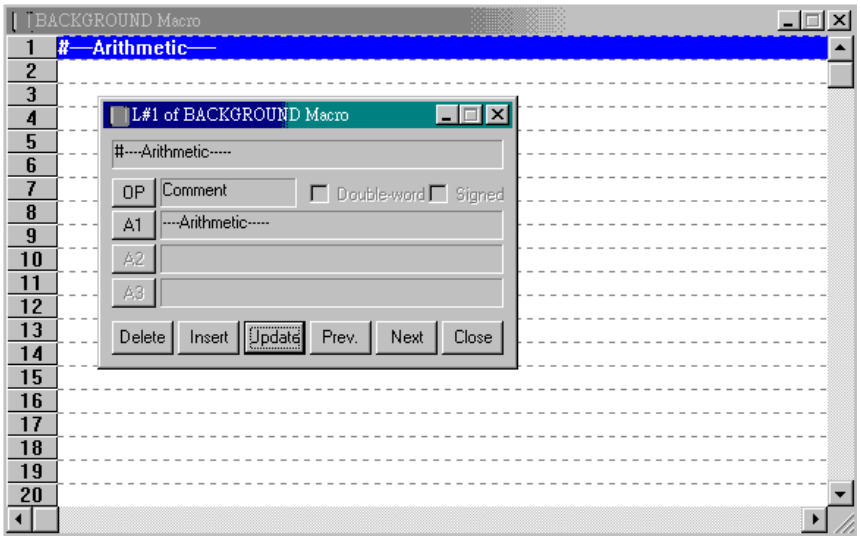


Step 5. Click “Comment” then click “O.K.”

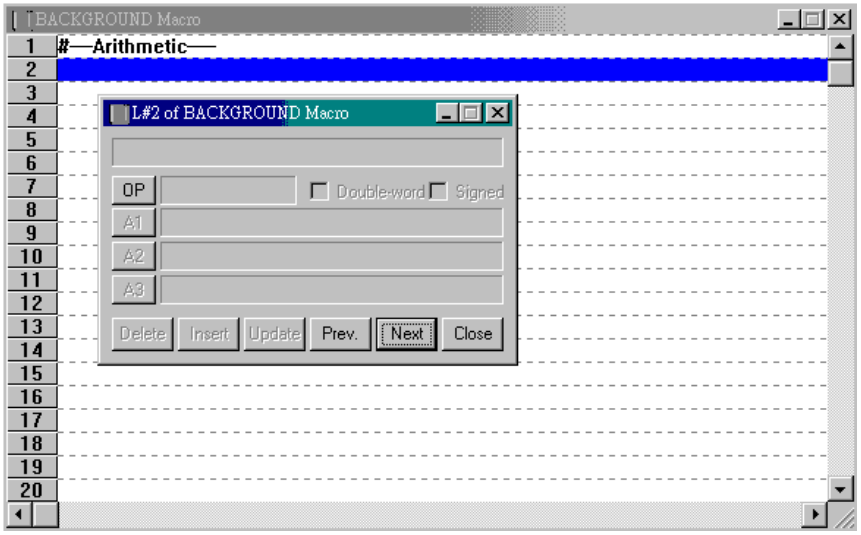
Step 6. Click “A1”, types “--Arithmetic--” in the “Input Text” box then click “O.K.”



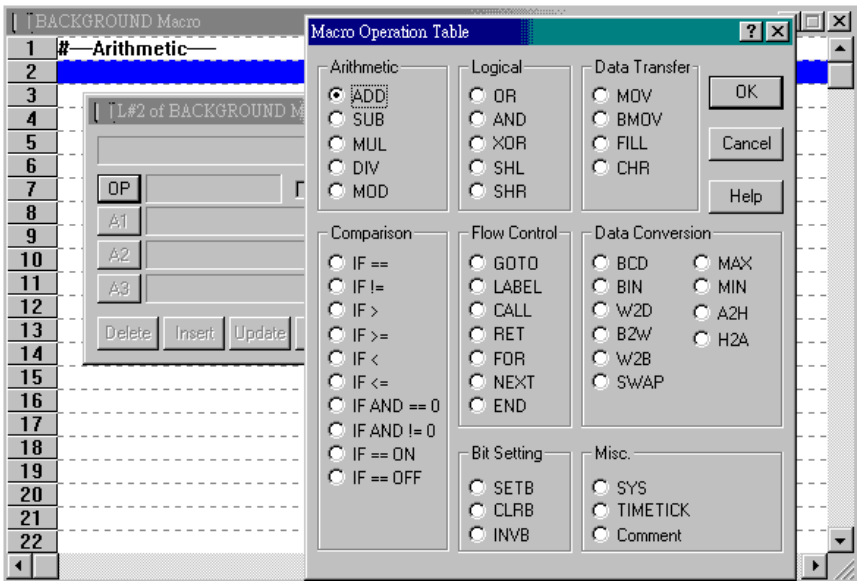
Step 7. Click “update” to complete the first instruction.



Step 8. Click “Next” to move to line 2.

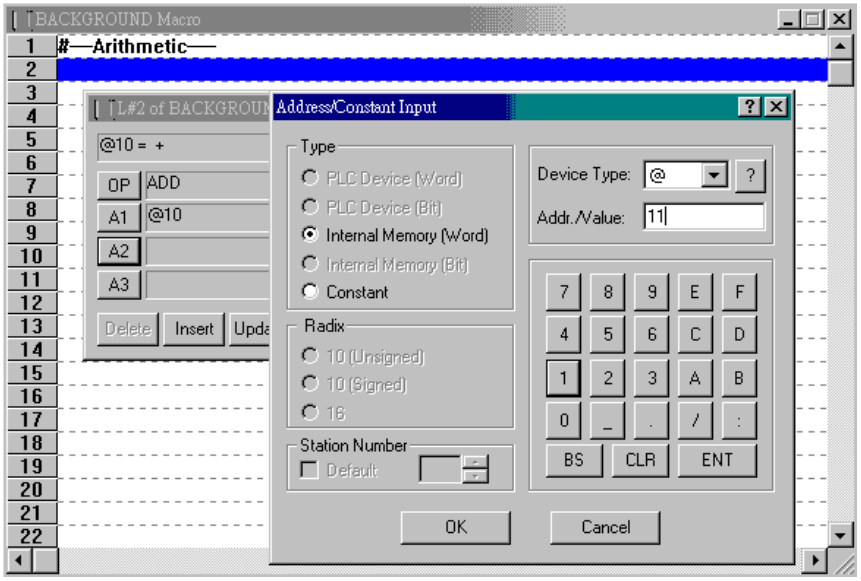


Step 9. To edit function of “ADD”.



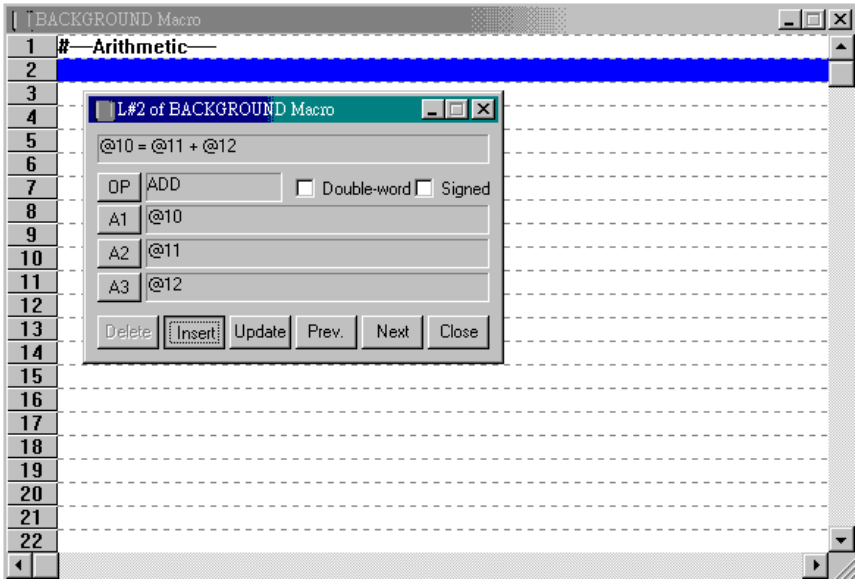
Step 10. Click “A1” to get a “ Address/Constant Input “dialogue. Input “@10” then choose “OK”.

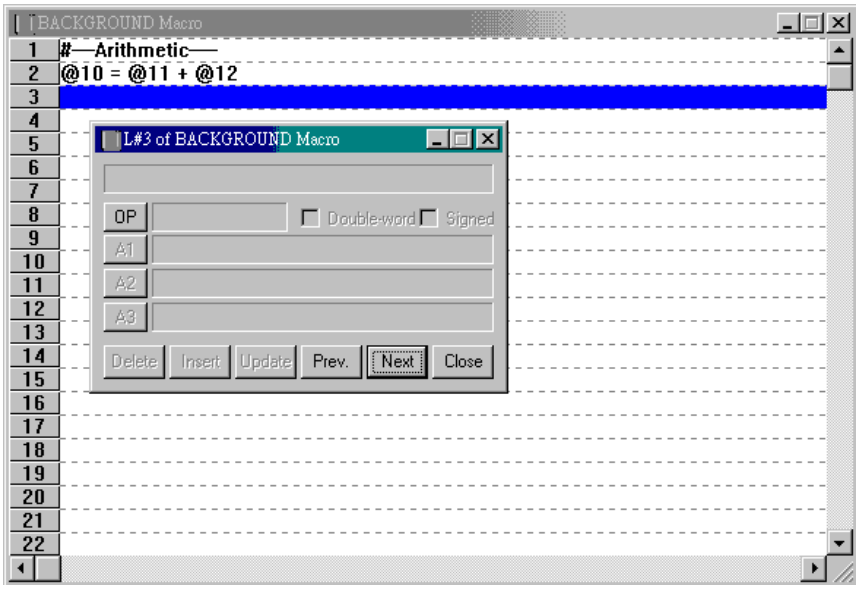
Step 11. Click “A2” to get a “ Address/Constant Input “dialogue. Input “@11” then choose “OK”.



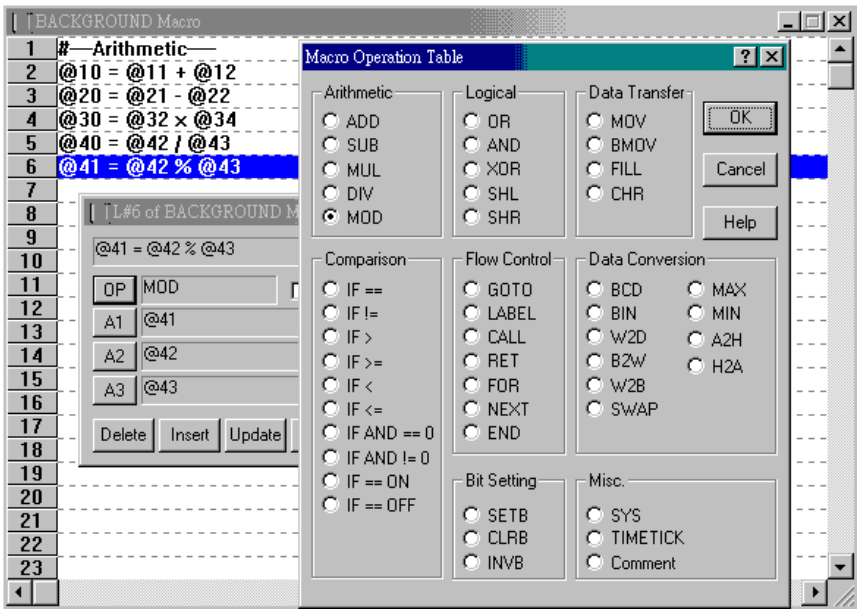
Step 12. Click “A3” to get a “ Address Constant Input “dialogue. Input “@12” then choose “OK”.

Click “insert” to complete the second instruction.





According to step 9 to step12 to edit the other instructions line 3 to line 6 for your application.



Internal Memory

The internal memory is a memory in the workstation that can be used by user. With this internal memory, PWS can operate quickly because it is not necessary to transfer to or receive data from the controller or PLC. Combining the internal memory with the Macro function gives additional high functionality for PWS-workstations.

There are four different types of internal memory.

Words

	Device Type	Size	Address	Aux. Address	R/W
RCPNO	0x80	W	0	0	R/W
RCPWn	0x81	W	0-?	0	R/W
CBn	0x82	W	0-31	0	R
@n	0x85	W	0-10239 (PWS1760)	0	R/W
@n	0x85	W	0-639 (PWS500S, 1711)	0	R/W
*@n	0x85	W	0-10239 (PWS1760)	0	R/W
*@n	0x85	W	0-639 (PWS500S, 1711)	0	R/W

Internal Memory

Bits

	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 (PWS1760)	0-15	R/W
@n.b (b=0-f)	0x86	0-639 (PWS500S, 1711)	0-15	R/W
@Xn	NA	0-7 (PWS520S)	NA	R/W
@Yn	NA	0-7 (PWS520S)	NA	R/W

Note 1: the PWS-1700, PWS-1760, PWS-3100, PWS-3160, PWS-3700, PWS-3760, and SoftPanel support these device types.

Note 2: Although the maximum you can specify is 65534, the valid maximum is determined by the following calculation:
Valid maximum = (Maximum number of recipes + 1) x Recipe size – 1

Note 3: The valid maximum is the size of Control Block minus one.

Note 4: This device type is supported by the PWS-500/520, PWS-1711 Macro, PWS-1760, PWS-3160, PWS-3760, and SoftPanel

Note 5: The maximum is 639 for PWS-500 and PWS-1711 Macro.

Note 6: This is for indirect addressing. For example, if the value of @n is m, *@n identifies @m.

Note 7: This type is digital input and is supported by PWS-520 only.

Note 8: This type is digital output and is supported by PWS-520 only.