

variables, function,  
and function block

BY: YUT

## content

- Data types
- **Function Block (FB)**
- **Function (FUN)**
- Exercise

# Data types

## basic data types

資料型態	目錄	大小	輸入	輸出	內部
BOOL	位元資料 BIT data	1 位元	OK	OK	OK
INT	整數 integer	16 位元	OK	OK	OK
UNIT	無正負號整數 non-positive/non-negative integer	16 位元	OK	OK	OK
DINT	二位整數 two-digit integer	32 位元	OK	OK	OK
UDINT	無正負號雙重整數 non-positive/non-negative double integer	32 位元	OK	OK	OK
LINT	長(4 個字)整數 long integer	64 位元	OK	OK	OK
ULINT	無正負號長整數(4 個字) non-positive/non-negative long integer	64 位元	OK	OK	OK
WORD	16 位元資料 16 bits data	16 位元	OK	OK	OK
DWORD	32 位元資料 32 bits data	32 位元	OK	OK	OK
LWORD	64 位元資料 64 bits data	64 位元	OK	OK	OK
REAL	實數 real numbers	32 位元	OK	OK	OK
LREAL	長實數 long integer	64 位元	OK	OK	OK
TIMER	計時器(請參閱備註 1.) timer (see note 1)	旗標：1 位元 PV：16 位元	不支援	不支援	OK
COUNTER	計數器(請參閱備註 2.) support	旗標：1 位元 PV：16 位元	不支援	不支援	OK

counter(see note 2)  
support

Does not

# Data type

## advanced data type—arrays

The array data structure(**Array**) is a data structure composed of a collection of elements of the same type, and a continuous memory is allocated for storage. The storage address corresponding to the element can be calculated using the index of the element.




data declaration type  ARRAY[0..?] OF **X** data type

size of the array

Example:

There is a variable named APPLE. This variable is used to record the weight of 10 apples, so we set the data of this variable to

ARRAY[0..9] OF INT

APPLE[5] = 100 According to the rule on the left, if you want to know

sixth apple weight the ninth apple =APPLE [X]

data type

advanced data type- arrays

exercise

Exercise: new\_Controller\_0 - Sysmac Studio (64bit)

1. Section0 - Program

2. 變數

3. Added a new internal variable named Light

The data type is ARRAY[0..9] OF BOOL

4. Internal variable declaration table:

名稱	數據類型	初始值	分配到	保持	常數	註解
Light	ARRAY[0..9] OF BOOL			<input type="checkbox"/>	<input type="checkbox"/>	
PB1	BOOL			<input type="checkbox"/>	<input type="checkbox"/>	

4. Ladder logic diagram showing PB1 connected to Light[0], Light[2], Light[4], and Light[6].

data type

advanced data type— arrays exercise

1.simulation->operation

2.inspection->monitor the inspection

3.enter the inspected variables

The screenshot shows the Sysmac Studio interface. The top menu bar includes '檢視(V)' (View) and '模擬(S)' (Simulate). The main workspace displays a ladder logic diagram with a network containing a pulse generator 'PB1' connected to an array of outputs 'Light[0]' through 'Light[6]'. The bottom panel shows the '監視(專案)' (Monitor Project) window, which lists variables and their current values.

名稱	數據類型	初始值	分配到	保持	常數	註解
Light	ARRAY[0..9] OF BOOL			<input type="checkbox"/>	<input type="checkbox"/>	
PB1	BOOL			<input type="checkbox"/>	<input type="checkbox"/>	

總上值	修改	註解	數據類型	分配到	顯示格式
True	TRUE FALSE		BOOL		Boolean
True	TRUE FALSE		ARRAY[0..9] OF B		
True	TRUE FALSE		BOOL		Boolean
True	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean
True	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean
True	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean
True	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean
False	TRUE FALSE		BOOL		Boolean

# data type

## advanced data type—arrays exercise

A union is a value with multiple types or formats. Some programming languages can have a special data type of "union" to represent the above variables.

The length of a union is equal to the length of its longest internal member, and they all share the same memory.

Example:

Set a WORD and an array Boolean in the union

array Boolean value is 1000 then WORD would be 8

on the contrary

WORD is 10 the array boolean value is 10000

# data type

## advanced data type— struct

**struct** refers to a [data structure](#) is a kind of Composite data type in C language

A struct is also a collection of elements, which are called members of the structure, and these members can be of different types. Members are generally accessed by name.

example:

A sandwich is composed of toast, ham, lettuce and eggs. If today we set the sandwich as a structure data type, and you want to find the toast in a sandwich, this variable is **sandwich.toast**



# data type

## advanced data type-enumerations

Enumeration is a basic data type in C language, which can make data more concise and easier to read.

If a variable can have several possible values, it can be defined as an enumeration data type. The so-called "enumeration" is to list each value of the variable. These values are called "enumeration elements". These enumeration elements are named using text strings when they are defined.

### Example

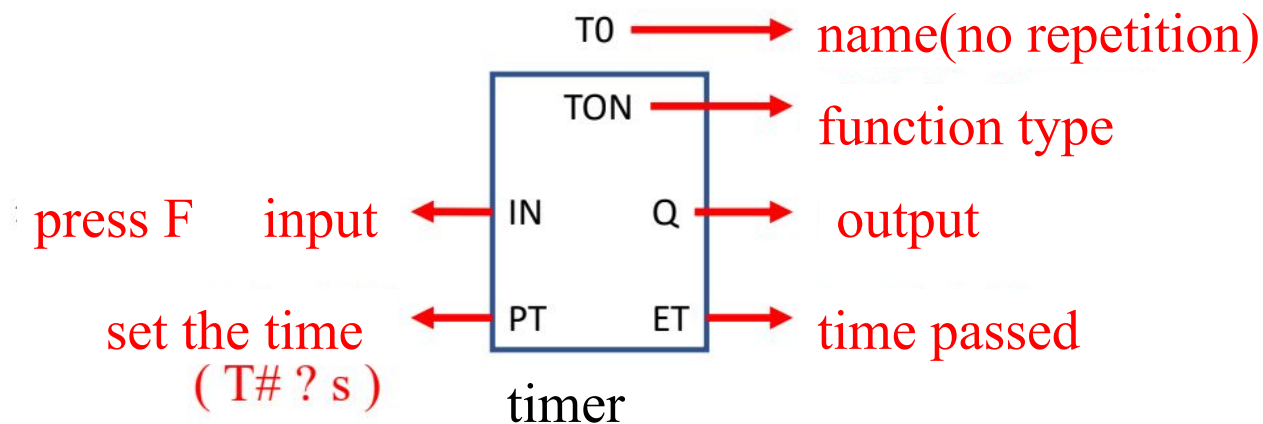
Let the rainbow variable be an enumeration, in which the enumeration elements are set to red=1, orange=2,..., purple=7  
If this variable (rainbow) = 4, it will ~~display~~ rainbow  
#green according to the set enumeration element

# Function Block (FB) **\*If you don't understand a function or function block, press F1 to query it.**

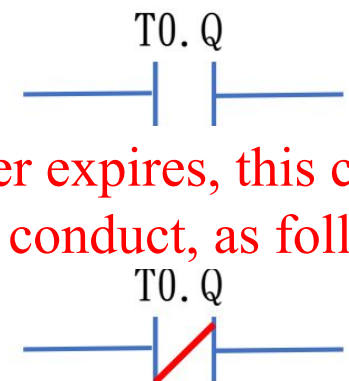
Instructions:

1. You need a **name**
2. If you use function blocks with the same function, you need to **set different names** for the function blocks.
3. There is a memory function that will take up the memory location inside the controller.
4. Belongs to struct
5. Common function blocks are timer and counter

**\*Because it is a struct, if you want to use this timer output signal alone, as follows:**



**When the timer expires, this contact will close and conduct, as follows:**



# Function Block (FB) **\*\*If you don't understand a function or function block, press F1 to query it.**

## TON exercise

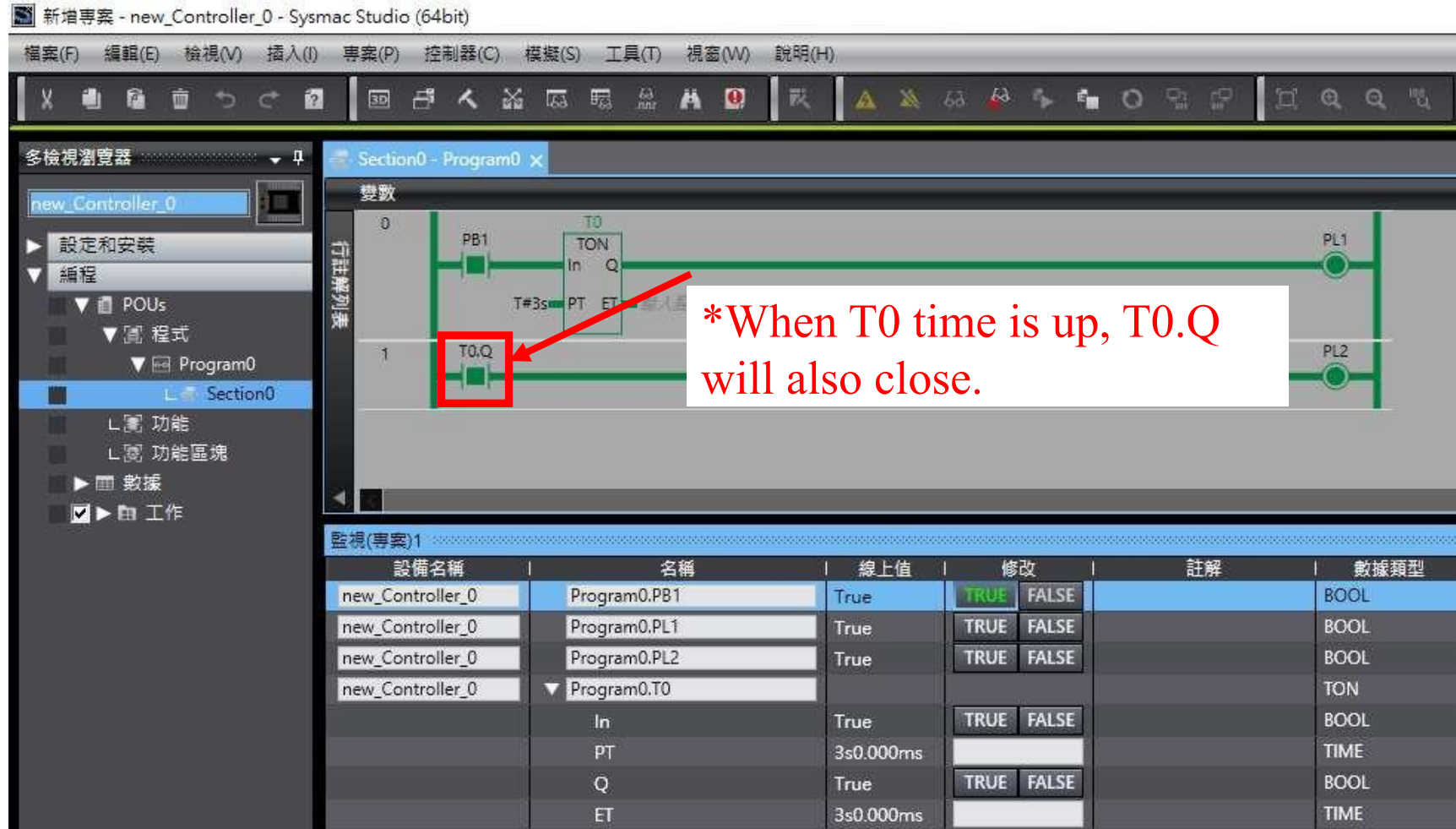
The screenshot displays the Sysmac Studio (64bit) interface. The title bar reads '新增專案 - new\_Controller\_0 - Sysmac Studio (64bit)'. The menu bar includes '檔案(F)', '編輯(E)', '檢視(V)', '插入(I)', '專案(P)', '控制器(C)', '模擬(S)', '工具(T)', '視窗(W)', and '說明(H)'. The left sidebar shows a project tree with 'new\_Controller\_0' expanded, containing '設定和安裝', '編程', 'POUs', '程式', 'Program0', 'Section0', '功能', '功能區塊', '數據', and '工作'. The main workspace shows a ladder logic diagram for 'Section0 - Program0'. A red box highlights a TON function block labeled 'T0'. The block has inputs 'In' and 'PT' (set to 'T#3s'), and outputs 'Q' and 'ET'. The output 'Q' is connected to a coil labeled 'PL2'. A red box also highlights the '內部' (Internal) tab in the variable declaration table. A text box overlay on the right states: '2. After the input is completed, you can see the variable named T0 in the internal variable table, and the data type is TON.' Another text box overlay on the right states: '1. F->enter TON->Name the FB T0'.

2. After the input is completed, you can see the variable named T0 in the internal variable table, and the data type is TON.

1. F->enter TON->Name the FB T0

Function Block (FB) **\*If you don't understand a function or function block, press F1 to query it.**

TON exercise



The screenshot displays the Sysmac Studio interface for a new project named 'new\_Controller\_0'. The main workspace shows a ladder logic diagram for 'Section0 - Program0'. It features a normally open contact labeled 'PB1' connected to the 'In' input of a TON (Timer On Delay) function block labeled 'T0'. The timer's preset time 'PT' is set to 'T#3s'. The output 'Q' of the TON block is connected to a coil labeled 'T0.Q', which is highlighted with a red box. A red arrow points from a text box to this box, stating: '\*When T0 time is up, T0.Q will also close.' The diagram also shows two output coils, 'PL1' and 'PL2', connected to the same power rail as the 'T0.Q' coil.

Below the diagram is a monitoring table titled '監視(專案)1' (Monitor (Project) 1). The table has columns for '設備名稱' (Equipment Name), '名稱' (Name), '線上值' (Online Value), '修改' (Modify), '註解' (Comment), and '數據類型' (Data Type). The table lists the following variables and their current states:

設備名稱	名稱	線上值	修改	註解	數據類型
new_Controller_0	Program0.PB1	True	TRUE FALSE		BOOL
new_Controller_0	Program0.PL1	True	TRUE FALSE		BOOL
new_Controller_0	Program0.PL2	True	TRUE FALSE		BOOL
new_Controller_0	▼ Program0.T0				TON
	In	True	TRUE FALSE		BOOL
	PT	3s0.000ms			TIME
	Q	True	TRUE FALSE		BOOL
	ET	3s0.000ms			TIME

Function Block (FB) **\*If you don't understand a function or function block, \**

TON exercise **press F1 to query it.**



**\*The upper and lower have the same function, just different ways of writing.**

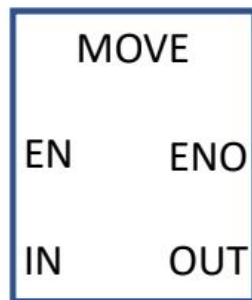
**\*If you don't understand a function or function block,  
press F1 to query it.**

## Function (FUN)

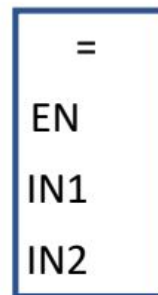
Instructions:

1. Unlike function blocks, **no name is required.**
2. There is no memory function and it will not take up the memory inside the controller.
3. Common functions include assignment, comparison, increment, four arithmetic operations, etc.

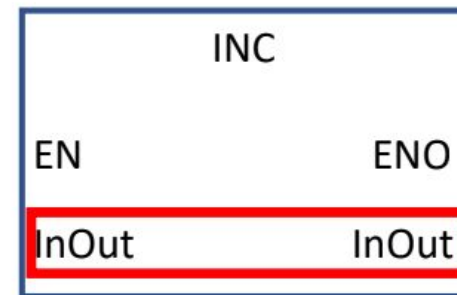
鍵盤打I



move/assignment



comparison



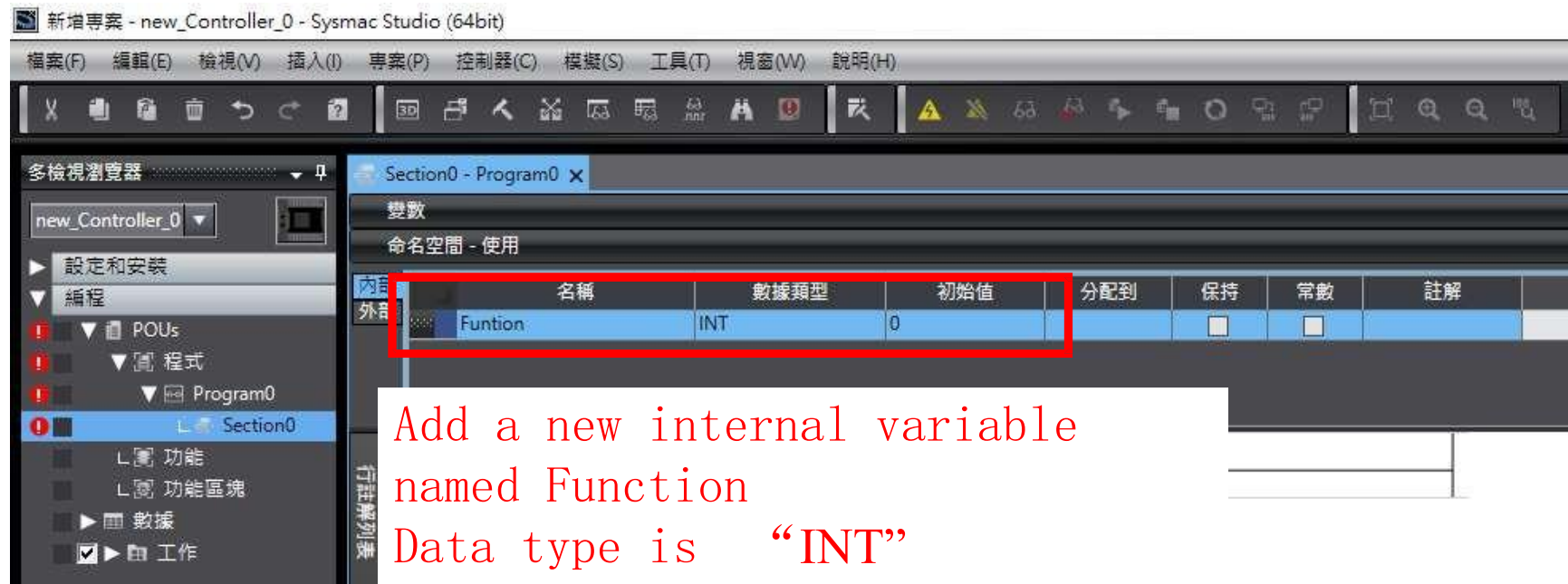
increment

can be input  
and output  
\*can be  
overwritten

# Function (FUN)

assignment,  
comparison exercise

**\*If you don't understand a function or  
function block, press F1 to query it.**



Add a new internal variable  
named Function

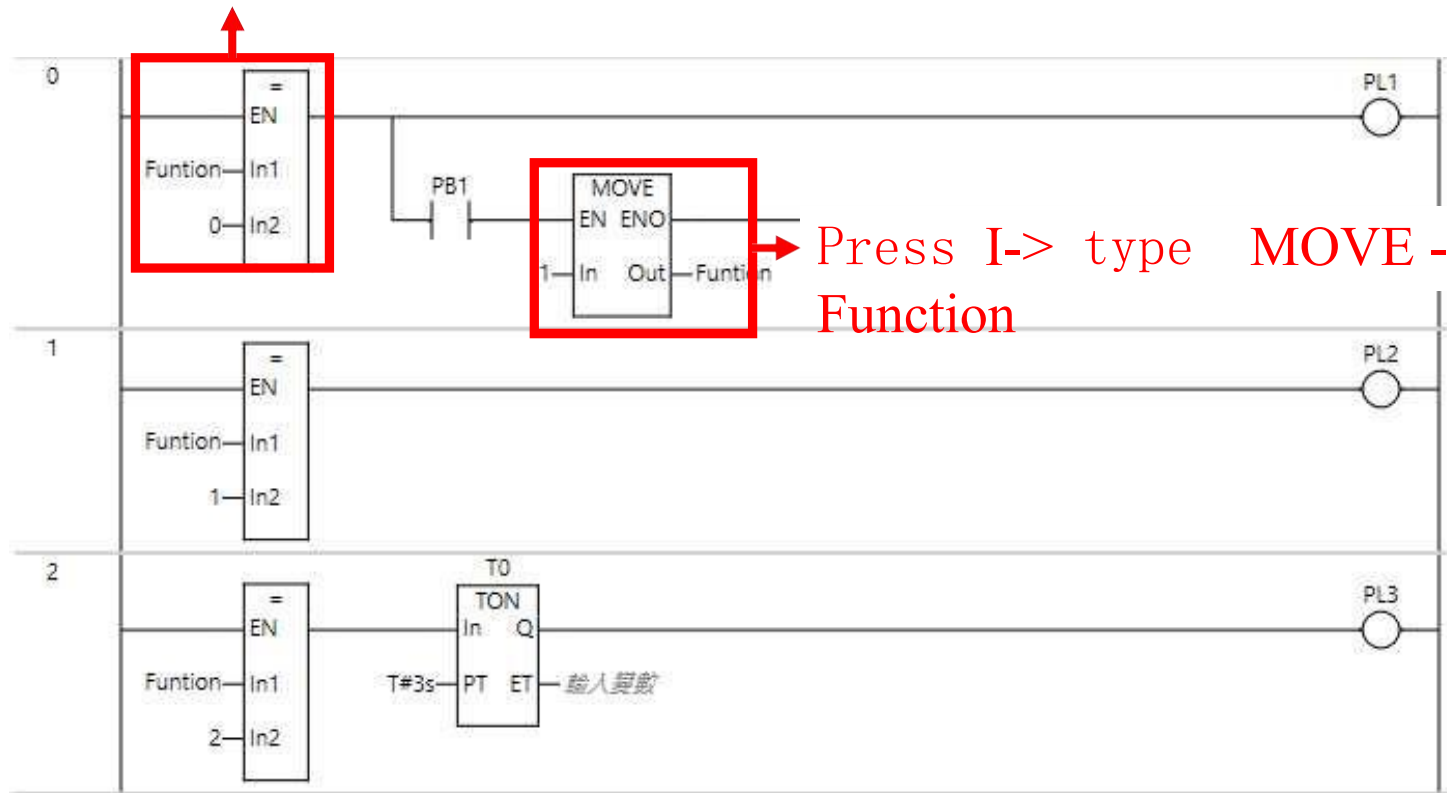
Data type is “INT”  
initial value is “0”

**\*If you don't understand a function or function block,  
press F1 to query it.**

## Function (FUN)

assignment, comparison exercise

Press I->input=->IN1 type Function、IN2 type 0





# Function (FUN)

assignment,  
comparison

**\*If you don't understand a function or function block,  
press F1 to query it.**

The screenshot displays the Sysmac Studio (64bit) interface. The main workspace shows a ladder logic program with three rungs. Rung 0 contains a function block 'Function' with inputs 'In1' and 'In2', and output 'Function(0)'. Rung 1 contains a function block 'Function' with inputs 'In1' and 'In2', and output 'Function(0)'. Rung 2 contains a function block 'Function' with input 'In1', and output 'Function(0)'. The program is titled 'Section0 - Program0'. A white box with the text 'Enter watch list' is overlaid on the program.

Below the program, the '監視(專案)1' (Watch List) is displayed. It contains a table with the following data:

設備名稱	名稱	線上值	修改	註解	數據類型
new_Controller_0	Program0.Function	0	0		INT
new_Controller_0	Program0.PB1	False	TRUE FALSE		BOOL
new_Controller_0	Program0.PL1	True	TRUE FALSE		BOOL
new_Controller_0	Program0.PL2	False	TRUE FALSE		BOOL
new_Controller_0	Program0.PL3	False	TRUE FALSE		BOOL

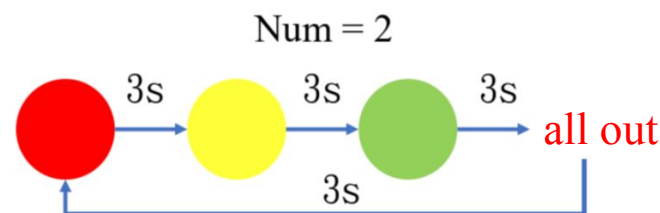
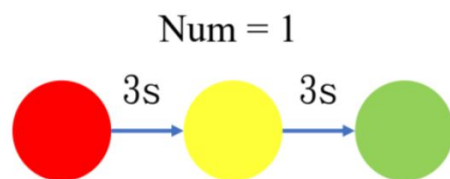
## exercise 1

1. Set an internal variable named "Number", the data type is "INT", and the initial value is "0"
2. When Number = 1, PL1 lights up
3. When Number = 2, press and hold PB1, PL2 will light up, release PB1, PL2 will turn off (inching)
4. When Number = 3, press and hold PB2, PL3 will light up, release PB2, PL3 will light up (self-holding)
5. When Number = 4, press and hold PB3. After 3 seconds, PL4 will light up. Release PB3 and PL4 will turn off.

## Exercise 2 (3-color lights)

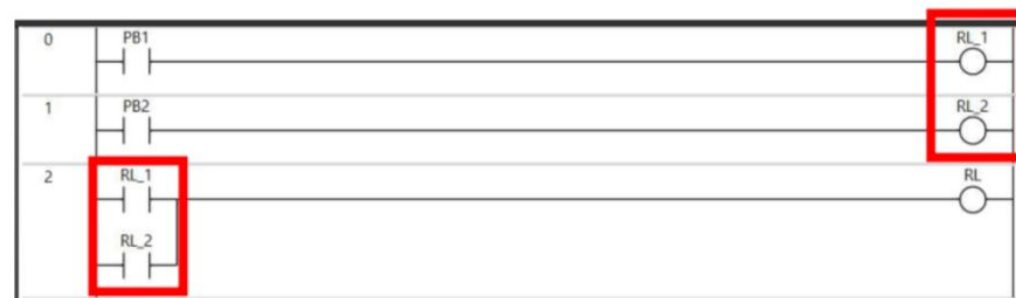
1. Set an internal variable named "Num", the data type is "INT", and the initial value is "0"
2. When Num = 1, the red light (RL) is on. After 3 seconds, the yellow light (YL) is on. After another 3 seconds, the green light (GL) is on.
3. When Num = 2, the red light turns on. After 3 seconds, the yellow light turns on. After another 3 seconds, the green light turns on.

After another 3 seconds, all the red, yellow and green lights will go out, and then repeat the above cycle after another 3 seconds.



how to avoid output  $\longrightarrow$  driving point impedance

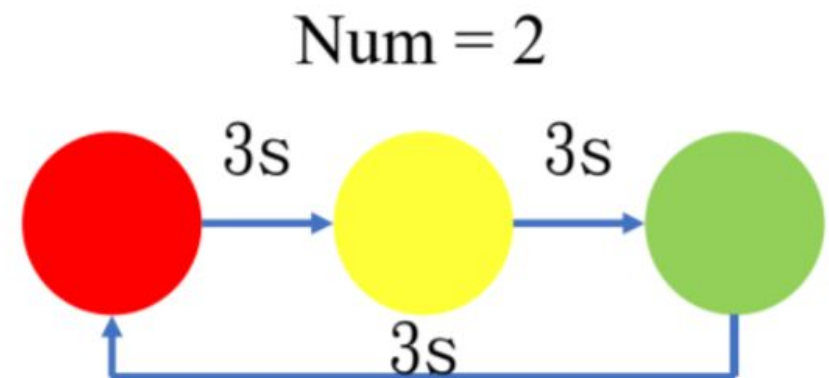
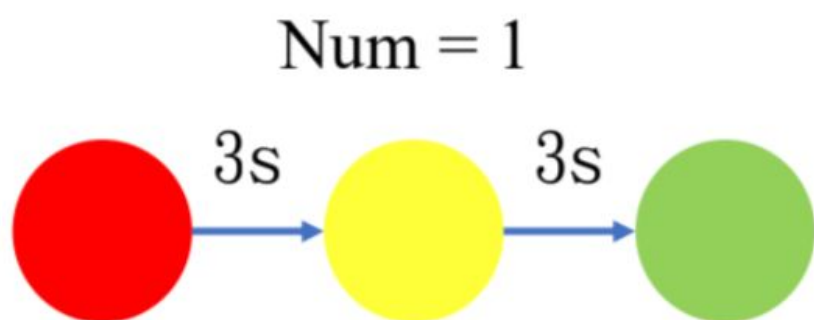
$\longrightarrow$  2.Set & Reset



driving point impedance

## Exercise 3 (3-color lights 2)

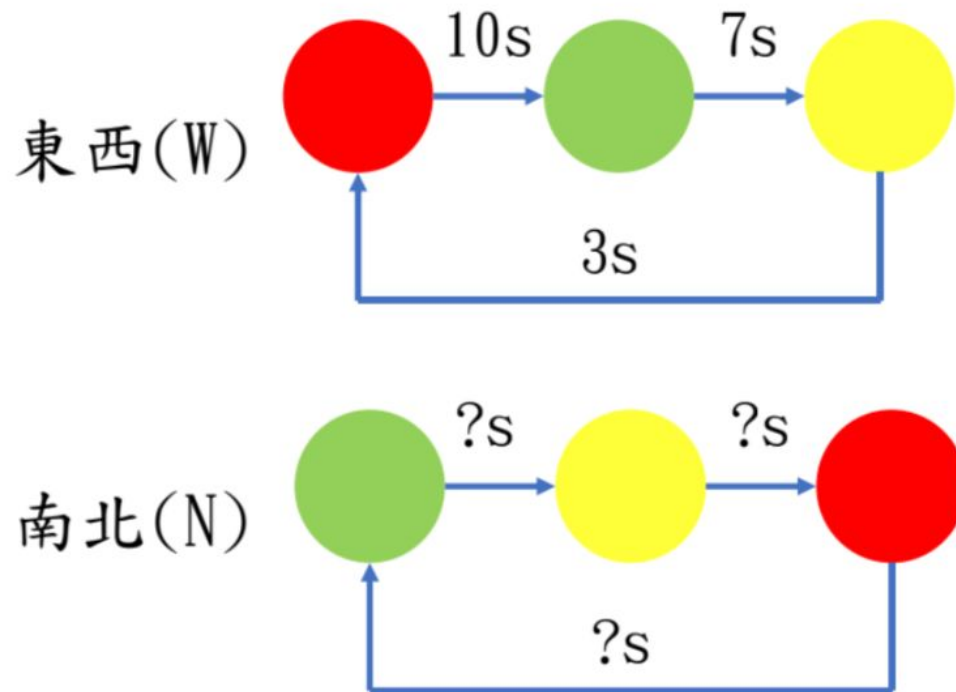
1. Set an internal variable named "Num", the data type is "INT", and the initial value is "0"
2. When Num = 1, the red light (RL) is on. After 3 seconds, the red light is off and the yellow light (YL) is on. After another 3 seconds, the yellow light is off and the green light (GL) is on.
3. When Num = 2, the red light is on. After 3 seconds, the red light is off and the yellow light is on. After another 3 seconds, the yellow light is off and the green light is on. After another 3 seconds, the green light goes out and the red light is on and then repeat the above cycle.



## Exercise 4 (3-color lights 3-Two-way traffic light)

1. Set an internal variable named "Traffic\_Light", the data type is "INT", and the initial value is "0".
2. Set internal variables named "WE\_Light, NS\_Light" with data type "ARRAY[1..3] OF BOOL"
3. When Traffic\_Light = 1, start executing two-way traffic lights

two-way traffic lights



\*naming  
variable  
horizontal  
red: WE\_Light[1]  
yellow: WE\_Light[2]  
green: WE\_Light[3]  
vertical  
red: NS\_Light[1]  
yellow: NS\_Light[2]  
green: NS\_Light[3]

## exercise 5

switch button mode

variable instruction:

1. Set the internal variable named "Mode", the data type is "INT", and the initial value is "0"
2. Set internal variables named "COS1, COS2\_L, COS2\_R" with data type "BOOL"
3. Set the internal variable named "Light" with the data type "ARRAY[0..3] OF BOOL"
4. Assuming that COS1 is a selector switch, the rules are as follows: left switch is contact a, and right switch is contact b.
5. Assuming that COS2 is a three-stage switch, the rules are as follows: COS2\_L is on when switching to the left, COS2\_L and COS2\_R are not conducting when in the middle position, and COS2\_R is on when switching to the right.

function instruction:

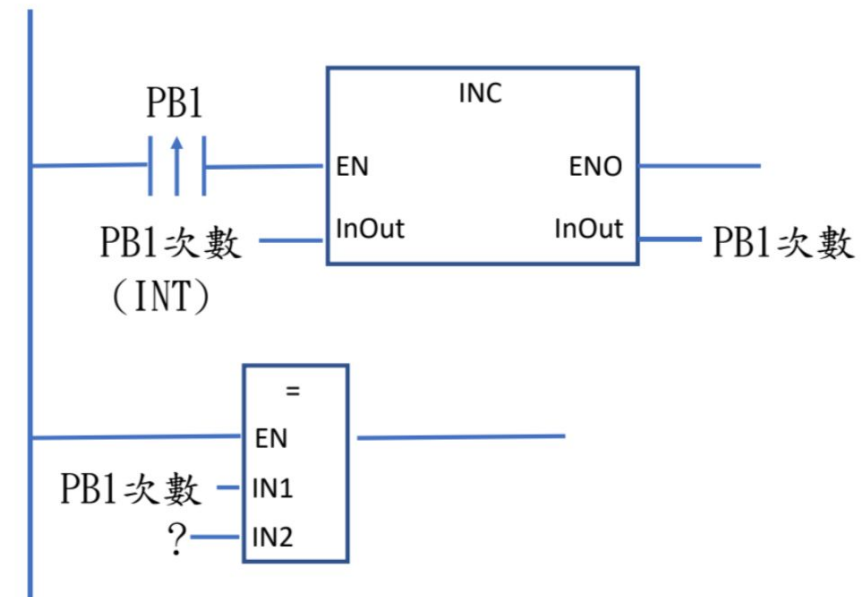
1. When COS1 cuts right, Mode will be assigned the value 1
2. When COS1 cuts left and COS2 cuts left, Mode will be assigned a value of 10
3. When COS1 cuts left and COS2 cuts mid, Mode will be assigned a value of 20
4. When COS1 cuts left and COS2 cuts right, Mode will be assigned a value of 30
5. When the value in Mode is as follows, its status is as follows:
  - Mode=1 Light[0] bright
  - Mode=10 Light[1] turns on, Light[2] turns on after 2s, Light[3] turns on after 2s
  - Mode=20 Light[1], Light[3] flash alternately for 1s
  - Mode=30 Light[1]~Light[3] form a 1s marquee

# exercise 6

## button counting

1. PL4 lights up after powering on
2. Press and hold PB1 for the first time, PL1 will light up and PL4 will turn off.  
Release PB1 for the first time, PL1 goes off, PL4 lights up
3. Press and hold PB1 for the second time, PL2 will light up and PL4 will turn off.  
Release PB1 for the second time, PL2 goes off, PL4 lights up
4. Press and hold PB1 for the third time, PL3 will light up and PL4 will turn off.  
Release PB1 for the third time, PL3 goes off, PL4 lights up

\*提示



**\*When the PB1 count is 3, remember to reset the PB1 count to 0**