使用 PLC(GPU228, S7-226 仿製品)

- 工具和文件
  - https://60.249.144.60:5001 路徑：**WTGroup_Data/information/PLC**
- 軟體
  - STEP-7MicroWIN V4.0 SP9完整版 .zip (安裝過程有錯誤，目前堪用，還沒發現問題)
  - 【工贝电子】GB PPI-CH340编程电缆驱动 .rar (usb-ppi 線的驅動程式)
    - 必須將 windows 10作業系統的區域設為簡體中文後才能順利安裝
  - PCPPIcable_FixInterfaceNotFound.zip (萬一連線硬體失敗的話，可以嘗試這個更新)
- 程式
 - serial_communication_20210114.mwp (以官方手冊的 tx/rx 範例程式為基礎來發展)

目前實現的 TX/RX 程式

- 以 S7-200 Programmable Controller System Manual 文件提供的 TX/RX 範例程式為基礎來發展

- 程式的行為
  - 原則上 PLC 會透過 PORT 0 回傳相等數量的字元給 PC 端，除了內容有些變化
    - 第二和第三個 byte 會被換成板子上類比輸入 0 量到的電壓值
  - 若 PC 端送給 PLC 的第一個字元是 1(對等 ASCII code 的值為 0x31， 0x 代表 16 進位)
    - PLC 板子上的 Q0.1 數位輸出開關會變成 ON。 (若不是 1 的任何字元， Q0.1 會 OFF)
  - I0.0(數位輸入) 控制 Q0.0(數位輸出)
    - 兩者呈現反相關係 (若其中一個為 0，另一個一定為 1，反之亦然)
  - 類比輸入 0(AI0) 量到的值會原封不動傳給類比輸出 0(AQ0)，而 AQ1 固定輸出 5V

Basic Elements of a Program

A program block is composed of executable code and comments. The executable code consists of a main program and any subroutines or interrupt routines. The code is compiled and downloaded to the S7-200; the program comments are not. You can use the organizational elements (main program, subroutines, and interrupt routines) to structure your control program.

The following example shows a program that includes a subroutine and an interrupt routine. This sample program uses a timed interrupt for reading the value of an analog input every 100 ms.
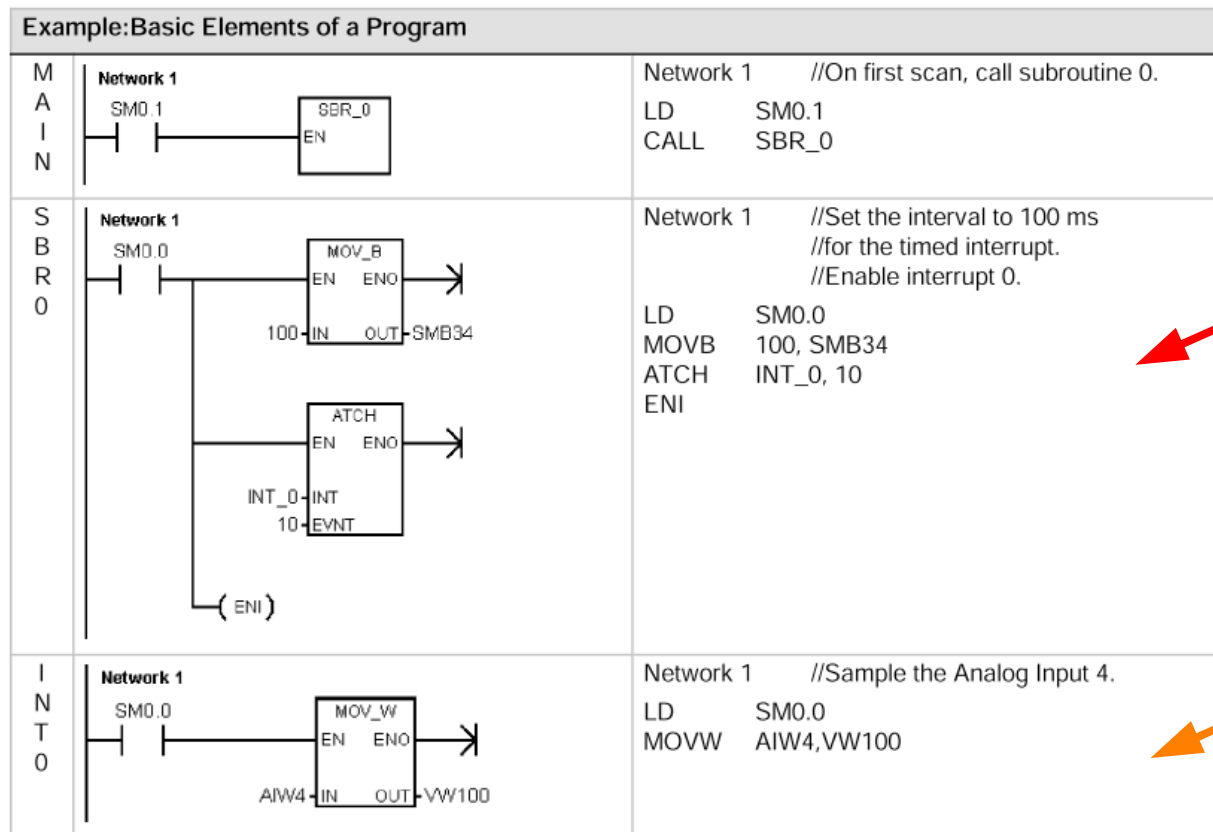
程式的輪廓

將 Q0.2 輸出開關每 250mS 打開或關閉

Example:Basic Elements of a Program
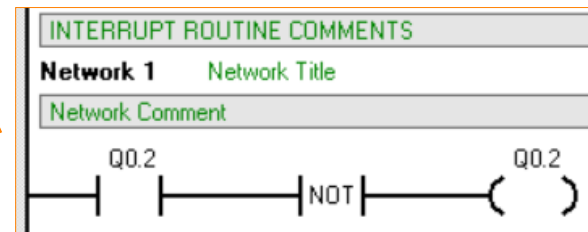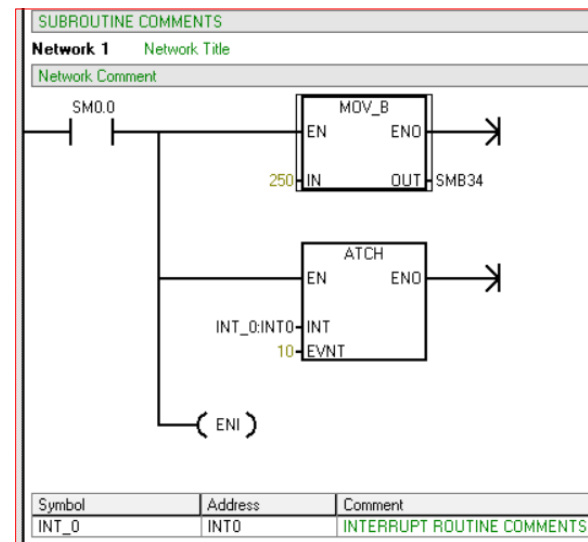
LAD(目前寫 code 首選) 和 FBD

Features of the LAD Editor

The LAD editor displays the program as a graphical representation similar to electrical wiring diagrams. Ladder programs allow the program to emulate the flow of electric current from a power source through a series of logical input conditions that in turn enable logical output conditions. A LAD program includes a left power rail that is energized. Contacts that are closed allow energy to flow through them to the next element, and contacts that are open block that energy flow.

The logic is separated into networks. The program is executed one network at a time, from left to right and then top to bottom as dictated by the program. Figure 5-3 shows an example of a LAD program. The various instructions are represented by graphic symbols and include three basic forms.

Contacts represent logic input conditions such as switches, buttons, or internal conditions.

Coils usually represent logic output results such as lamps, motor starters, interposing relays, or internal output conditions.

Boxes represent additional instructions, such as timers, counters, or math instructions.

Network 1
Network 2

Figure 5-3   Sample LAD Program

Consider these main points when you select the LAD editor:

易學，類似傳統的電器線路圖

- ❑ Ladder logic is easy for beginning programmers to use.
- ❑ Graphical representation is easy to understand and is popular around the world.
- ❑ The LAD editor can be used with both the SIMATIC and IEC 1131-3 instruction sets.
- ❑ You can always use the STL editor to display a program created with the SIMATIC LAD editor.

Features of the FBD Editor

The FBD editor displays the program as a graphical representation that resembles common logic gate diagrams. There are no contacts and coils as found in the LAD editor, but there are equivalent instructions that appear as box instructions.

X
FBD

Figure 5-4 shows an example of an FBD program.

FBD does not use the concept of left and right power rails; therefore, the term "power flow" is used to express the analogous concept of control flow through the FBD logic blocks.
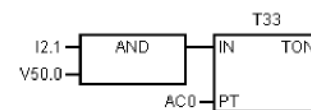
Figure 5-4   Sample FBD Program

The logic "1" path through FBD elements is called power flow. The origin of a power flow input and the destination of a power flow output can be assigned directly to an operand.

The program logic is derived from the connections between these box instructions. That is, the output from one instruction (such as an AND box) can be used to enable another instruction (such as a timer) to create the necessary control logic. This connection concept allows you to solve a wide variety of logic problems.

Consider these main points when you select the FBD editor:

- ❑ The graphical logic gate style of representation is good for following program flow.
- ❑ The FBD editor can be used with both the SIMATIC and IEC 1131-3 instruction sets.
- ❑ You can always use the STL editor to display a program created with the SIMATIC FBD editor.
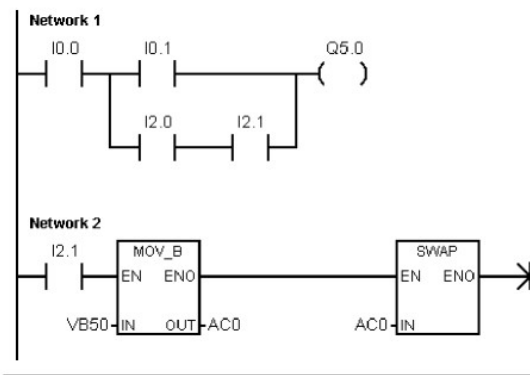
STL 可以實現另外 2 種不易做到的

Features of the STL Editor

The STL editor displays the program as a text-based language. The STL editor allows you to create control programs by entering the instruction mnemonics. The STL editor also allows you to create programs that you could not otherwise create with the LAD or FBD editors. This is because you are programming in the native language of the S7-200, rather than in a graphical editor where some restrictions must be applied in order to draw the diagrams correctly. As shown in Figure 5-2, this text-based concept is very similar to assembly language programming.

The S7-200 executes each instruction in the order dictated by the program, from top to bottom, and then restarts at the top.

寫法非常類似原生微處理器 (ARM?) 的組合語言

```
LD      I0.0        //Read one input
A       I0.1        //AND with another input
=       Q1.0        //Write value to output 1
```

STL uses a logic stack to resolve the control logic. You insert the STL instructions for handling the stack operations.

Figure 5-2   Sample STL Program

Consider these main points when you select the STL editor:

❑ STL is most appropriate for experienced programmers. 不容易學會

❑ STL sometimes allows you to solve problems that you cannot solve very easily with the LAD or FBD editor.

❑ You can only use the STL editor with the SIMATIC instruction set.

❑ While you can always use the STL editor to view or edit a program that was created with the LAD or FBD editors, the reverse is not always true. You cannot always use the LAD or FBD editors to display a program that was written with the STL editor.

Example: Transmit and Receive Instructions

MAIN

Network 1

```
SM0.1
|       |----[ MOV_B ]
        EN    ENO
 16#09-IN   OUT-SMB30

        [ MOV_B ]
        EN    ENO
 16#B0-IN   OUT-SMB87

        [ MOV_B ]
        EN    ENO
 16#0A-IN   OUT-SMB89

        [ MOV_W ]
        EN    ENO
   +5-IN   OUT-SMW90

        [ MOV_B ]
        EN    ENO
  100-IN   OUT-SMB94

        [ ATCH ]
        EN    ENO
 INT_0-INT
    23-EVNT

        [ ATCH ]
        EN    ENO
 INT_2-INT
     9-EVNT

       -( ENI )

        [ RCV ]
        EN    ENO
VB100-TBL
    0-PORT
```

```
Network 1   //This program receives a string of characters
            //until a line feed character is received.
            //The message is then transmitted back
            //to the sender.

LD      SM0.1       //On the first scan:
MOVB    16#09, SMB30 //1.  Initialize Freeport:
            //     - Select 9600 baud.
            //     - Select 8 data bits.
            //     - Select no parity.

MOVB    16#B0, SMB87 //2.  Initialize RCV message control byte:
            //     - RCV enabled.
            //     - Detect end of message character.
            //     - Detect idle line condition
            //       as the message start condition.

MOVB    16#0A, SMB89 //3.  Set end of message character
            //     to hex 0A (line feed).

MOVW    +5, SMW90   //4.  Set idle line timeout
            //     to 5 ms.

MOVB    100, SMB94   //5.  Set maximum number of characters
            //     to 100.

ATCH    INT_0, 23    //6.  Attach interrupt 0
            //     to the Receive Complete event.

ATCH    INT_2, 9     //7.  Attach interrupt 2
            //     to the Transmit Complete event.

ENI         //8.  Enable user interrupts.

RCV     VB100, 0     //9.  Enable receive box with
            //     buffer at VB100.
```

這列用 LAD 表示

另外這一列是 STL 對照

**Serial Port 範例程式: *Chapter 6* S7-200 Instruction Set
(S7-200 Programmable Controller System Manual)**

Example: Transmit and Receive Instructions, continued

INT0

Network 1

```
SMB86
|==B |----[ MOV_B ]
16#20    EN    ENO
     10-IN   OUT-SMB34

         [ ATCH ]
         EN    ENO
  INT_1-INT
     10-EVNT

       -( RETI )

  |NOT|----[ RCV ]
            EN    ENO
   VB100-TBL
       0-PORT
```

```
Network 1   //Receive complete interrupt routine:
            //1.  If receive status shows receive of
            //    end character, then attach a
            //    10 ms timer to trigger a transmit and return.
            //2.  If the receive completed for any other reason,
            //    then start a new receive.

LDB=    SMB86, 16#20
MOVB    10, SMB34
ATCH    INT_1, 10
CRETI
NOT
RCV     VB100, 0
```

INT1

Network 1

```
SM0.0
|       |----[ DTCH ]
             EN    ENO
         10-EVNT

             [ XMT ]
             EN    ENO
     VB100-TBL
         0-PORT
```

```
Network 1   //10-ms Timer interrupt:
            //1.  Detach timer interrupt.
            //2.  Transmit message back to user on port.

LD      SM0.0
DTCH    10
XMT     VB100, 0
```

INT2

Network 1

```
SM0.0
|       |----[ RCV ]
             EN    ENO
     VB100-TBL
         0-PORT
```

```
Network 1   //Transmit Complete interrupt:
            //Enable another receive.

LD      SM0.0
RCV     VB100, 0
```