PC-PLC 連線監控

一、 實驗題目舉例：

1. 撰寫一連線程式使 PLC 之 output y0~y7 可任意受控制 on-off。

2. 撰寫一連線程式使 PLC 之 output y0~y7 可以跑馬燈方式 on-off。

3. 撰寫一連線程式使 PLC 之 output y0~y7 可連續閃爍。

4. 撰寫一連線程式使 PLC 之 output y0~y7 每秒切換一顆燈。

進階實驗題目舉例

1.將 PC-PLC 連線監控與頭控程式結合之介面。
2. 將 PC-PLC 連線監控與頭控程式結合，進行環境（開關）控制。

二、 實驗目的：

本實驗介紹電腦與 PLC 間之連線監控,利用 RS232C 轉 RS422 在電腦與 PLC 之間作溝通,先以 FXGP-WIN-T 軟體編寫指令階梯圖,再以 Borland C++ Builder 軟體為操作介面，將指令傳送至 PLC，使 PLC 之 Outport 受控，進而達到連線監控之目的。

三、 實驗儀器：

PLC FX_{2N} 16MR	一台
RS232 傳輸線	一條
個人電腦（含 Borland C++ Builder 軟體）	一台
RS232 to RS422 轉接器	一件

MITSUBISHI FX_{2N} 16MR

四、實驗原理：

PC 與 PLC 之連線

本實驗介紹 PC 與 PLC 之間連線監控，可分為硬體與軟體兩方面

一、 硬體

由於 PLC 之介面規格為 RS422 之介面，因此在與電腦作連線控制時，需經由 232 轉 422 之轉換連接器，以轉換成電腦適用之介面格式。故在硬體方面，由 PLC 之輸寫孔接出一條 422 之傳輸線(25pin)，連接轉換器，轉換器另一端則接 RS232(25pin)轉 9 pin 之傳輸線，再接於 PC 之 COM port。

1. Mitsubishi FX_{2N} 簡介：

FX_{2N} 系列 PLC 是 FX 系列中最高级的模型。它擁有無以匹及的速度、高级的功能、邏輯選件以及定位控制等特点，FX2n 是從 16 到 256 輸入/輸出的多種應用的選擇方案。

2, 交流電源、24V 直流輸入類型：

模型	I/O 總數	輸入		輸出		尺寸（mm） （宽）x（厚）x（高）
		數目	類型	數目	類型	
FX_{2N}-16MR-001	16	8	漏型	8	繼電器	130x87x90

3.FX_{2N} 性能規格：

項目	規格	備註
運轉控制方法	通過儲存的程序週期運轉	

I/O 控制方法		批次處理方法（當執行 END 指令時）	I/O 指令可以刷新
運轉處理時間		基本指令：0.08 μs 應用指令：1.52 至幾百 μs 指令	
編輯語言		邏輯梯形圖和指令清單	使用步進梯形圖能生成 SFC 類型程序
程式容量		8000 步內置	使用附加寄存器盒可擴展到 16000 步
指令數目		基本順序指令：27 步進梯形指令：2 應用指令:：128	最大可用 298 條應用指令
I/O 配置		最大硬體 I/O 配置 256，依照用戶的選擇（最大軟件可設定地址輸入 256、輸出 256）	
輔助繼電器（M 線圈）	一般	500 點	M0 至 M499
	鎖定	2572 點	384 至 M3071
	特殊	256 點	M8000 至 M8255
狀態繼電器（S 線圈）	一般	490 點	S0 至 S499
	鎖定	400 點	S500 至 S899
	初始	10 點	S0 至 S9
	信號報警器	100 點	S900 至 S999
定時器（T）	100 毫秒	範圍：0 至 3276.7 秒 200 點	T0 至 T199
	10 毫秒	範圍：0 至 3276.7 秒 46 點	T200 至 T245
	1 毫秒保持型	範圍：0.001 至 3276.7 秒 4 點	T246 至 T249
	100 毫秒保持型	範圍：0 至 3276.7 秒 6 點	T250 至 T255
計數器（C）	一般 16 位	範圍：0 至 32767 數 200 點	C0 至 C199 類型：16 位上計數器
	鎖定 16 位	100 點（子系統）	C100 至 C199 類型：16 位上計數器
	一般 32 位	範圍：-2147483648 至 +32147483648 數 35 點	C200 至 C219 類型：32 位上/下計數器
	鎖定 32 位	15 點	C220 至 C234 類型：32 位上/下計數器

高速計數器（C）	單相	範圍： -2147483648+2147483648 數 一般規則：選擇組合計數頻率不大於 20KHz.計數器组合 注意所有的計數器都鎖定。	C235 至 C240　6 點
	單相 c/w 起始停止輸入		C241 至 C245　5 點
	雙相		C246 至 C250　5 點
	A/B 相		C251 至 C252　5 點
數據寄存器（D）	一般	200 點	D0 至 D199 類型：32 位元件的 16 位數據存儲寄存器
	鎖定	7800 點	D200 至 D7999 類型:32 位元件的 16 位數據存儲寄存器
	文件寄存器	7000 點	D1000 至 D7999 通過 14 塊 500 程式步的參数設置類型:16 位數據存儲寄存器
	特殊	256 點	從 D8000 至 D8255 類型：16 位數據存儲寄存器
	變址	16 點	V0 至 V7 和 Z0 至 Z7 類型：16 位數據存儲寄存器
指標（P）	用於 CALL	128 點	N0 至 P127
	用於中斷	6 輸入點、3 定時器、6 計數器	100*至 130* （上升觸發*＝1，下降觸發*＝0，**＝时间(单位：毫秒)）
嵌套層次		用於 MC 和 MRC 時 8 點	N0 至 N7
常數	十進位 K	16 位：-32768 至 32768 32 位：-2147483648 至+2147483647	
	十六進位 H	16 位：-32768 至+32768 32 位：-214783648 至+2147483647	
	浮點	32 位：$\pm 1.175 \times 10^{-33}$，$\pm 3.403 \times 10^{33}$（不能直接輸入）	

二、 軟體

1. 通訊設定

(1) 指定 PC Port Number
(2) 傳輸格式為非同步雙向 Baud Rate 9600
(3) 資料 7 位元
(4) 停止位元 1 個
(5) 檢查位元為偶位元(even parity)

2. 通訊協定格式

電腦和 FX2N 之間是一種主僕關係,也就是電腦是主端(master), FX2N 是僕端

(slave), 一切的通訊過程,首先電腦發出命令, FX2N 接到命令解讀後,會回應訊息給 PC。電腦傳給 PLC 的格式稱為命令格式,而 PLC 回給電腦的格式稱為回應格式,皆以 ASCII 碼表之。FX2N 通訊協定格式依序包含起始碼(start)、命令 (command)、位址和資料(address/data)、結束碼(terminator)、以及 SUM 檢查碼等欄所構成。

1	2	3	4	5	
起始碼	命令代號	資　料	結束碼	檢查碼 SUM	
STX	CMD	DATA	ETX	H	L

FX2N 通訊協定命令格式之各欄項意義分述如下

起始碼 : 格式的第一個字元為 STX=Chr$(02),表一命令(或回應)之起始。

命令代號 : 以命令代號"0"表 PLC 是對 FX2N 之元件群讀取資料;以"1"表要對 FX2N 之元件群寫入資料。

位址和資料欄 : 指定命令所要讀寫的元件對象。若是讀取 PLC 元件資料,則只要給起始位元位址和要讀取的元件數目。若是寫入 PLC 元件資料,則只要給定起始位元位址和寫入資料。

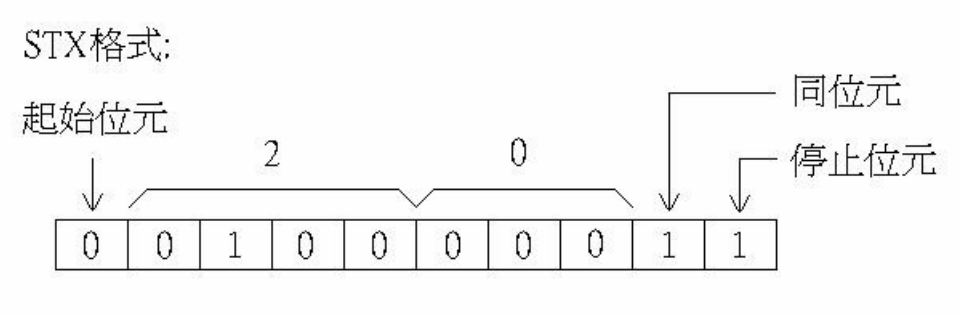
結束碼 : 格式的最後字元為 ETX=Chr$(03),表一命令(或回應)之結束。

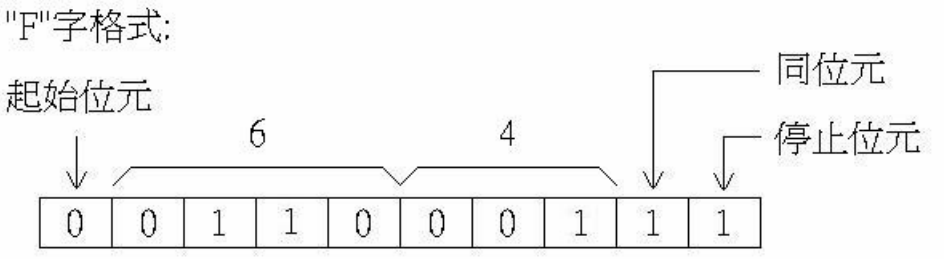
SUM 檢查碼 : 取格式 2～4 項各字元之 ASCII 碼做加運算,然後取最後兩字元即得到 SUM 碼,此碼是為了確保傳輸的可靠性而設的一個碼,其功能有如 RS232 協定的奇/偶數位元。在通訊過程中,PC 端將命令格式(1～5 項),這包括 PC 方所計算的 SUM 值,一併傳到 PLC 端。PLC 在接收到該命令字串時,會取 2～4 項複算 SUM 一次,核對此 SUM 值是否和 PC 端傳來的 SUM 值一致,如果一致,則表示傳輸正確。

命令總類	命令代號	目 標 元 件	功　能　說　明
讀取	0	X,Y,M,S,T,C,D	讀取元件群組
寫入	1	Y,M,S,T,C,D	寫入元件群組
強制 ON	7	Y,M,S,T,C,	強制單一接點為 ON
強制 OFF	8	Y,M,S,T,C,	強制單一接點為 OFF

STX,ETX 字元和其它字元構成一筆包封資料,隨著命令一起傳送,而 ENQ,ACK,NAK 則以單獨一字元表達某一種訊息,分別表達兩裝置通訊之間的請求與認知情況。

字元	ASCII	說　　明
ENQ	05H	Enquiry : 來自電腦方的請求。
ACK	06H	Acknowledge : 給 ENQ 的認知回應。
NAK	15H	Nagative Ack : 當無法認知時的回應。
STX	02H	Start of Text : 資料封包的起頭碼。
ETX	03H	End of Text : 資料封包的結束碼。
'0'	30H	
'1'	31H	
'2'	32H	





‘3’	33H	
‘4’	34H	
‘5’	35H	
‘6’	36H	
‘7’	37H	
‘8’	38H	
‘9’	39H	
‘A’	41H	
‘B’	42H	
‘C’	43H	
‘D’	44H	
‘E’	45H	
‘F’	46H	

3.SUM 檢查碼

檢查碼計算命令、位址、資料、結束碼各字元的 ASCII 碼做『和』運算,然後取其後兩位碼。

PC 讀取 PLC 資料,PC 主端將包含 SUM 的命令字串傳給 PLC,當傳至 PLC 端後,PLC 會再將接收到的命令字串複算 SUM 一次,核對是否與 PC 方算得之 SUM 相符!若是,則表示傳輸正確,若不相符,則表示傳方資料和接方資料兩者不一致,導致兩邊 SUM 值不一致,據此判定傳輸失敗。

PC 寫入 PLC 資料,PLC 回應格式 06H 代表正常,回應 15H 代表無法認知。

例：

STX	COM	16^3	16^2	16^1	16^0	16^1	16^0	ETX	161	160
02H	'0' 30H	'1' 31H	'0' 30H	'F' 46H	'6' 36H	'0' 30H	'4' 34H	03H	'7' 37H	'4' 34H

=========檢查碼計算範圍==========

======位址======= =位視組數= ==SUM==

SUM Check：30H+31H+30H+46H+36H+30H+34H+03H=174H=74H(取後兩位)

五、範例程式

本程式示範如何以 BCB 傳送指令控制 PLC 之 M 接點、

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "SPComm"
#pragma resource "*.dfm"
TForm1 *Form1;
char    writebuffer[9];
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
```

```
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Comm1ReceiveData(TObject *Sender, Pointer Buffer,
          WORD BufferLength)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StarClick(TObject *Sender)
{
Comm1->StartComm();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::StopClick(TObject *Sender)
{
Comm1->StopComm();
}




//---------------------------------------------------------------------------
void __fastcall TForm1::M0Click(TObject *Sender)
{
    //強制 M0 繼電器 ON
    writebuffer[0]=0x02;
    writebuffer[1]=0x37;
    writebuffer[2]=0x30;
    writebuffer[3]=0x30;
    writebuffer[4]=0x30;
    writebuffer[5]=0x38;
    writebuffer[6]=0x03;
    writebuffer[7]=0x30;
    writebuffer[8]=0x32;
    Comm1->WriteCommData(writebuffer,9);
}
 //---------------------------------------------------------------------------
```

```
}
void __fastcall TForm1::M1Click(TObject *Sender)
{
  //強制 M1 繼電器 ON
    writebuffer[0]=0x02;
    writebuffer[1]=0x37;
    writebuffer[2]=0x30;
    writebuffer[3]=0x31;
    writebuffer[4]=0x30;
    writebuffer[5]=0x38;
    writebuffer[6]=0x03;
    writebuffer[7]=0x30;
    writebuffer[8]=0x33;
    Comm1->WriteCommData(writebuffer,9);
}
//---------------------------------------------------------------------------

void __fastcall TForm1::M2Click(TObject *Sender)
{
    //強制 M2 繼電器 ON
    writebuffer[0]=0x02;
    writebuffer[1]=0x37;
    writebuffer[2]=0x30;
    writebuffer[3]=0x32;
    writebuffer[4]=0x30;
    writebuffer[5]=0x38;
    writebuffer[6]=0x03;
    writebuffer[7]=0x30;
    writebuffer[8]=0x34;
    Comm1->WriteCommData(writebuffer,9);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::M3Click(TObject *Sender)
{
  //強制 M3 繼電器 ON
    writebuffer[0]=0x02;
    writebuffer[1]=0x37;
    writebuffer[2]=0x30;
```

```
    writebuffer[3]=0x33;
    writebuffer[4]=0x30;
    writebuffer[5]=0x38;
    writebuffer[6]=0x03;
    writebuffer[7]=0x30;
    writebuffer[8]=0x35;
    Comm1->WriteCommData(writebuffer,9);
}
//---------------------------------------------------------------------------
```

☆階梯圖