GIAO TIẾP CỔNG NỐI TIẾP RS232

Tham khảo các bài viết trên internet

Chuẩn giao tiếp RS232 là một trong những kỹ thuật được sử dụng rộng rãi hiện nay để nối ghép các thiết bị ngoại vi với máy tính. Nó là một chuẩn giao tiếp nối tiếp dùng định dạng không đồng bộ, kết nối nhiều nhất là hai thiết bị , chiều dài kết nối lớn nhất cho phép để đảm bảo dữ liệu là 15m, tốc độ 20kbit/s (Ngày nay có thể cao hơn)

Các đặc tính kỹ thuật của chuẩn RS-232 theo tiêu chuẩn TIA/EIA-232-F như sau:

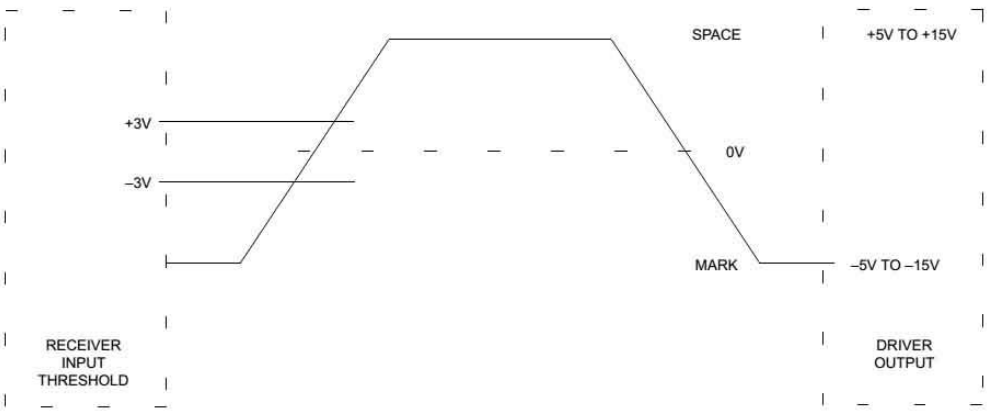
Chiều dài cable cực đại	15m (50 Feet)
Tốc độ dữ liệu cực đại	20 Kbps
Điện áp ngõ ra cực đại	± 25V
Điện áp ngõ ra có tải	± 5V đến ± 15V
Trở kháng tải	3K đến 7K
Điện áp ngõ vào	± 15V
Độ nhạy ngõ vào	± 3V
Trở kháng ngõ vào	3K đến 7K

Các tốc độ truyền dữ liệu thông dụng trong cổng nối tiếp là: 1200 bps, 4800 bps, 9600 bps và 19200 bps.

Các mức điện áp của đường truyền:
Mức điện áp của tiêu chuẩn RS232(chuẩn thường được dùng bây giờ) được mô tả như sau:

Mức logic 0: +3V, +12V (SPACE)

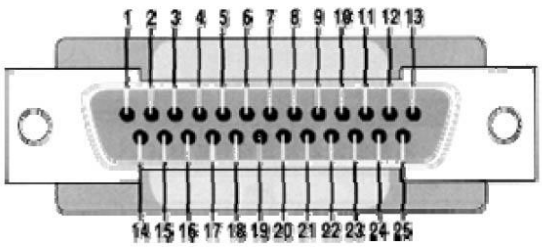
Mức logic 1: -12V, -3V. (MARK)
Các mức điện áp trong phạm vi từ -3V đến 3V là trạng thái chuyển tuyến. Chính vì từ -3V tới 3V là phạm vi không được định nghĩa, trong trường hợp thay đổi giá trị logic từ thấp lên cao hoặc từ cao xuống thấp, một tín hiệu phải vượt qua quãng quá độ trong một thời gian ngắn hợp lý. Điều này dẫn tới việc phải hạn chế về điện dung của các thiết bị tham gia và của cả đường truyền. Tốc độ truyền dẫn tối đa phụ thuộc vào chiều dài của dây dẫn.Đa số các hệ thống hiện nay chỉ hỗ trợ với tốc độ 19,2kbit/s.

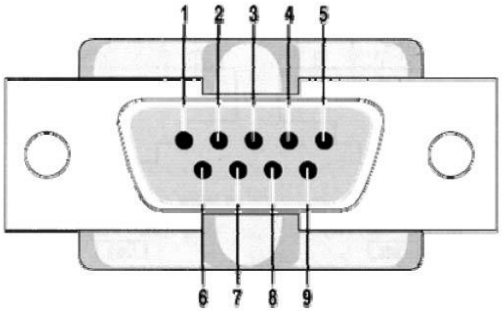


Sơ đồ chân cổng kết nối

Các máy tính thường có một hoặc hai cổng nối tiếp theo chuẩn RS232 được gọi là cổng COM. Chúng được dùng để ghép nối cho chuột, modem, thiết bị đo lường…Trên main máy tính có loại 9 chân hoặc loại 25 chân tùy vào đời máy và main của máy tính.

❖ Sơ đồ chân:





Hình 4.2 – Sơ đồ chân cổng nối tiếp

Cổng COM có hai dạng: đầu nối DB25 (25 chân) và đầu nối DB9 (9 chân) mô tả như
hình 4.2. Ý nghĩa của các chân mô tả như sau:

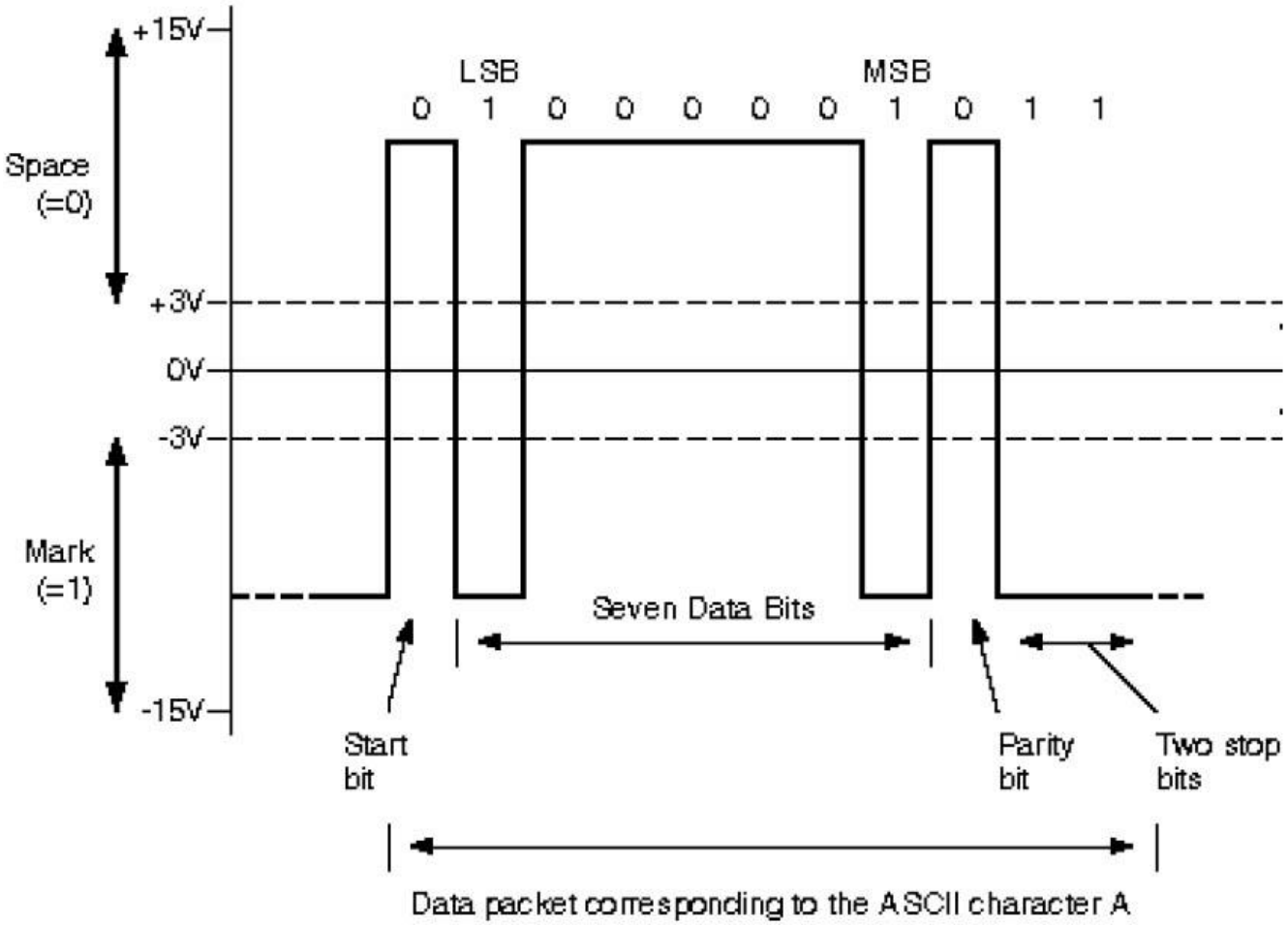
D25	**D9**	**Tín hiệu**	**Hướng truyền**	**Mô tả**
1	-	-	-	Protected ground: nối đất bảo vệ
2	3	TxD	DTE -> DCE	Transmitted data: dữ liệu truyền
3	2	RxD	DCE -> DTE	Received data: dữ liệu nhận
4	7	RTS	DTE -> DCE	Request to send: DTE yêu cầu truyền dữ liệu
5	8	CTS	DCE -> DTE	Clear to send: DCE sẵn sàng nhận dữ liệu
6	6	DSR	DCE -> DTE	Data set ready: DCE sẵn sàng làm việc
7	5	GND	-	Ground: nối đất (0V)
8	1	DCD	DCE->DTE	Data carier detect: DCE phát hiện sóng mang
20	4	DTR	DTE->DCE	Data terminal ready: DTE sẵn sàng làm việc
22	9	RI	DCE->DTE	Ring indicator: báo chuông
23	-	DSRD	DCE->DTE	Data signal rate detector: dò tốc độ truyền
24	-	TSET	DTE->DCE	Transmit Signal Element Timing: tín hiệu định thời truyền đi từ DTE

15	-	TSET	DCE->DTE	Transmitter Signal Element Timing: tín hiệu định thời truyền từ DCE để truyền dữ liệu
17	-	RSET	DCE->DTE	Receiver Signal Element Timing: tín hiệu định thời truyền từ DCE để truyền dữ liệu
18	-	LL		Local Loopback: kiểm tra cổng
21	-	RL	DCE->DTE	Remote Loopback: Tạo ra bởi DCE khi tín hiệu nhận từ DCE lỗi
14	-	STxD	DTE->DCE	Secondary Transmitted Data
16	-	SRxD	DCE->DTE	Secondary Received Data
19	-	SRTS	DTE->DCE	Secondary Request To Send
13	-	SCTS	DCE->DTE	Secondary Clear To Send
12	-	SDSRD	DCE->DTE	Secondary Received Line Signal Detector
25	-	TM		Test Mode
9	-			Dành riêng cho chế độ test
10	-			Dành riêng cho chế độ test
11				Không dùng

Truyền dữ liệu:

Định dạng của khung truyền dữ liệu theo chuẩn RS-232 như sau:

	Start	D0	D1	D2	D3	D4	D5	D6	D7	P	Stop
	0										1



Dạng tín hiệu truyền mô tả như sau (truyền ký tự A):

Truyền dữ liệu qua cổng nối tiếp RS232 được thực hiện không đồng bộ. Do vậy nên tại một thời điểm chỉ có một bit được truyền. Bộ truyền gửi một bit bắt đầu (bit start) để thông báo cho bộ nhận biết một ký tự sẽ được gửi đến trong lần truyền bit tiếp theo. Bit này luôn bắt đầu bằng mức 0. Tiếp theo đó là các bit dữ liệu (bit data) được gửi dưới dạng mã ASCII (có thể là 5,6,7, hay 8 bit dữ liệu) sau đó là một Parity bit (kiểm tra bit chẵn, lẻ hay không) và cuối cùng là bit stop (còn gọi là bit dừng) có thể là 1 hay 2 bit Stop.

Tốc độ baud.

Đây là một tham số đặc trưng của RS232. Tham số này chính là đặc trưng cho quá trình truyền dữ liệu qua cổng nối tiếp RS232 là tốc độ truyền nhận dữ liệu hay còn gọi là tốc độ bit. Tốc độ bit được định nghĩa là số bit

truyền được trong thời gian 1 giây. Tốc độ bit này phải được thiết lập ở bên phát và bên nhận đều phải có tốc độ như nhau (tốc độ giữa vi điều khiển và máy tính phải chung nhau một tốc độ truyền bit).
Ngoài tốc độ bit còn một tham số để mô tả tốc độ truyền là tốc độ baud. Tốc độ baud liên quan đến tốc độ mà phân tử mã hóa dữ liệu được sử dụng để diễn tả bit được truyền, còn tốc độ bit thì phản ánh tốc độ mà phân tử mã hóa dữ liệu được sử dụng để diễn tả bit được truyền. Vì một phần tử báo hiệu sự mã hóa một bit nên khi đó hai tốc độ bit và tốc độ baud là phải đồng nhất.

Một số tốc độ baud thường dùng: 50, 75, 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200, 28800, 38400, 56000, 115200. Trong thiết bị thường dùng tốc độ baud là 19200.

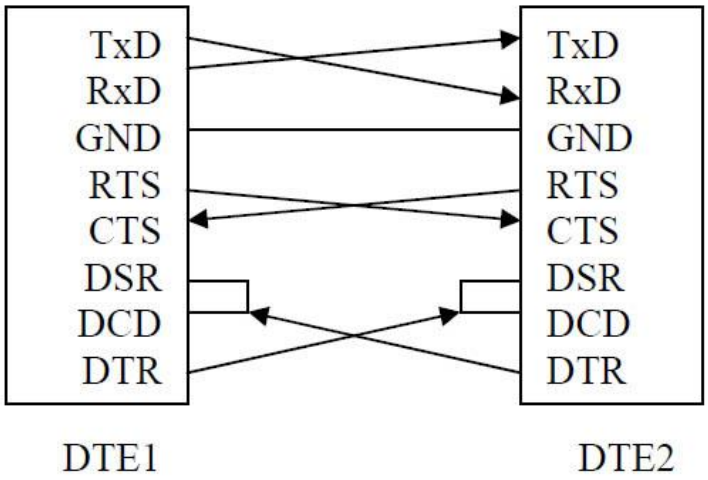
Bit chẵn lẻ hay Parity bit.
Đây là bit kiểm tra lỗi trên đường truyền. Thực chất của quá trình kiểm tra lỗi khi truyền dữ liệu là bổ sung thêm dữ liệu được truyền để tìm ra hoặc sửa một số lỗi trong quá trình truyền. Do đó trong chuẩn RS232 sử dụng một kỹ thuật kiểm tra chẵn lẻ. Một bit chẵn lẻ được bổ sung vào dữ liệu được truyền để thấy số lượng các bit "1" được gửi trong một khung truyền là chẵn hay lẻ. Một Parity bit chỉ có thể tìm ra một số lẻ các lỗi như là 1, 3, 5, 7, 9… Nếu như một bit mắc lỗi thì bit Parity bit sẽ trùng giá trị với trường hợp không mắc lỗi vì thế không phát hiện ra lỗi. Do đó trong kỹ thuật mã hóa lỗi này không được sử dụng trong trường hợp có khả năng một vài bit bị mắc lỗi.

2. Truyền thông giữa hai nút:

Các sơ đồ khi kết nối dùng cổng nối tiếp:

Hình 4.3 – Kết nối đơn giản trong truyền thông nối tiếp

Khi thực hiện kết nối như trên, quá trình truyền phải bảo đảm tốc độ ở đầu phát và thu giống nhau. Khi có dữ liệu đến DTE, dữ liệu này sẽ được đưa vào bộ đệm và tạo ngắt. Ngoài ra, khi thực hiện kết nối giữa hai DTE, ta còn dùng sơ đồ sau:



Hình 4.4 – Kết nối trong truyền thông nối tiếp dùng tín hiệu bắt tay

Khi DTE1 cần truyền dữ liệu thì cho DTR tích cực -> tác động lên DSR của DTE2 cho biết sẵn sàng nhận dữ liệu và cho biết đã nhận được sóng mang của MODEM (ảo). Sau đó, DTE1 tích cực chân RTS để tác động đến chân CTS của DTE2 cho biết DTE1 có thể nhận dữ liệu. Khi thực hiện kết nối giữa DTE và DCE, do tốc độ truyền khác nhau nên phải thực hiện điều khiển lưu lượng. Quá trinh điều khiển này có thể thực hiện bằng phần mềm hay phần cứng. Quá trình điều khiển bằng phần mềm thực hiện bằng hai ký tự Xon và Xoff. Ký tự Xon được DCE gởi đi khi rảnh (có thể nhận dữ liệu). Nếu DCE bận thì sẽ gởi ký tự Xoff. Quá trình điều khiển bằng phần cứng dùng hai chân RTS và CTS. Nếu DTE muốn truyền dữ liệu thì sẽ gởi RTS để yêu cầu truyền, DCE nếu có khả năng nhận dữ liệu (đang rảnh) thì gởi lại CTS.

3. Truy xuất trực tiếp thông qua cổng:

Các cổng nối tiếp trong máy tính được đánh số là COM1, COM2, COM3, COM4 với các địa chỉ như sau:

Tên	Địa chỉ	Ngắt	Vị trí chứa địa chỉ

COM1	3F8h	4	0000h:0400h
COM2	2F8h	3	0000h:0402h
COM3	3F8h	4	0000h:0404h
COM4	2F8h	3	0000h:0406h

Giao tiếp nối tiếp trong máy tính sử dụng vi mạch UART với các thanh ghi cho trong bảng sau:

Offset	**DLAB**	**R/W**	**Tên**	**Chức Năng**
0	0	W	THR	Transmitter Holding Register (đệm truyền)
	0	R	RBR	Receiver Buffer Register (đệm thu)
	1	R/W	BRDL	Baud Rate Divisor Latch (số chia byte thấp)
1	0	R/W	IER	Interrupt Enable Register (cho phép ngắt)
	1	R/W	BRDH	Số chia byte cao
2		R	IIR	Interrupt Identification Register (nhận dạng ngắt)
		W	FCR	FIFO Control Register
3		R/W	LCR	Line Control Register (điều khiển đường dây)
4		R/W	MCR	Modem Control Register (điều khiển MODEM)
5		R	LSR	Line Status Register (trạng thái đường dây)
6		R	MSR	Modem Status Register (trạng thái MODEM)
7		R/W		Scratch Register (thanh ghi tạm)

Các thanh ghi này có thể truy xuất trực tiếp kết hợp với địa chỉ cổng (ví dụ như thanh ghi cho phép ngắt của COM1 có địa chỉ là BA_{COM1} + 1 = 3F9h).

IIR (Interrupt Identification):

IIR xác định mức ưu tiên và nguồn gốc của yêu cầu ngắt mà UART đang chờ phục vụ. Khi cần xử lý ngắt, CPU thực hiện đọc các bit tương ứng để xác định nguồn gốc của ngắt. Định dạng của IIR như sau:

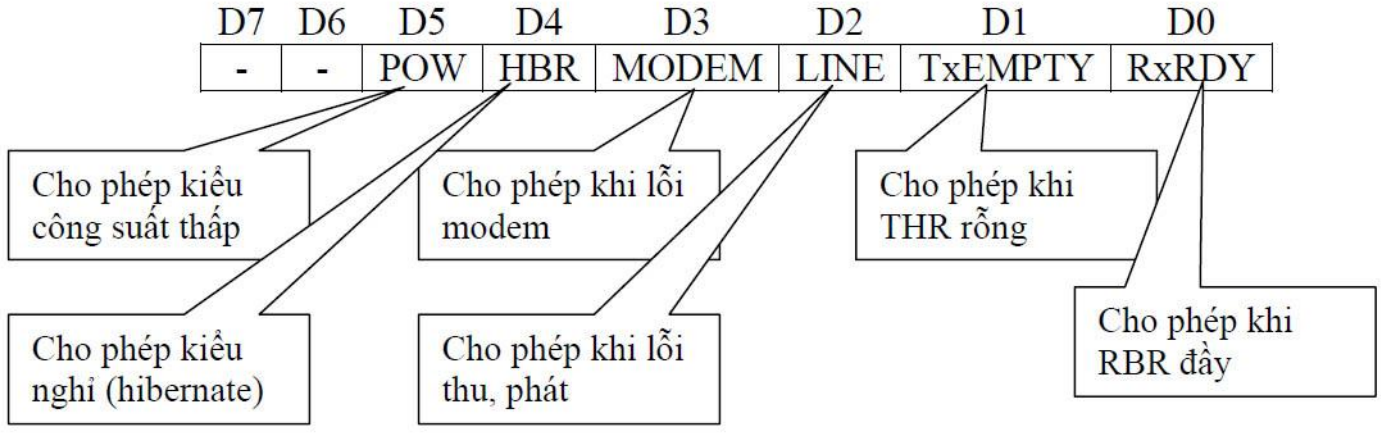
D7 D6	D5	D4	D3	D2 D1	D0
00: không có FIFO 11: cho phép FIFO	Cho phép FIFO 64 byte (trong 16750)	-	1: ngắt time-out (trong 16550)	Xác định nguồn gốc ngắt	0: có ngắt 1: không ngắt

D2	D1	Ưu tiên	Tên	Nguồn	D2 – D0 bị xoá khi
0	0	4	Đường truyền	Lỗi khung, thu đè, lỗi parity, gián đoạn khi thu	Đọc LSR
0	1	3	Đệm thu	Đệm thu đầy	Đọc RBR
1	0	2	Đệm phát	Đệm phát rỗng	Đọc IIR, ghi THR
1	1	1	Modem	CTS, DSR, RI, RLSD	Đọc MSR

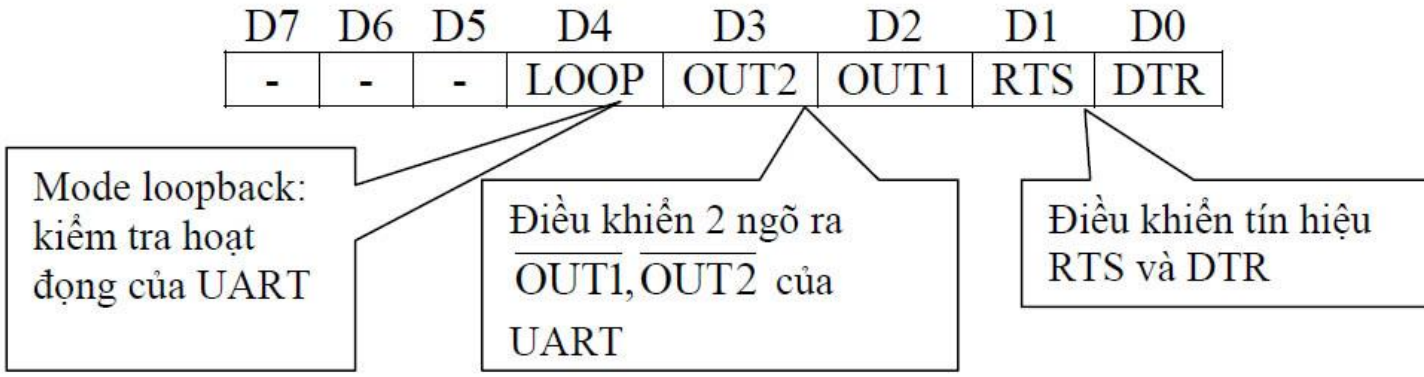
(mức 1 ưu tiên cao nhất)

IER (Interrupt Enable Register):

IER cho phép hay cấm các nguyên nhân ngắt khác nhau (1: cho phép, 0: cầm ngắt)



MCR (Modem Control Register):



LSR (Line Status Register):

D7	D6	D5	D4	D3	D2	D1	D0
FIE	TSRE	THRE	BI	FE	PE	OE	RxDR

FIE: FIFO Error – sai trong FIFO
TSRE: Transmitter Shift Register Empty – thanh ghi dịch rỗng (=1 khi đã phát 1 ký tự và bị xoá khi có 1 ký tự chuyển đến từ THR.
THRE: Transmitter Holding Register Empty (=1 khi có 1 ký tự đã chuyển từ THR –TSR và bị xoá khi CPU đưa ký tự tới THR).
BI: Break Interrupt (=1 khicó sự gián đoạn khi truyền, nghĩa là tồn tại mức logic 0 trong khoảng thời gian dài hơn khoảng thời gian truyền 1 byte và bị xoá khi CPU đọc LSR)
FE: Frame Error (=1 khi có lỗi khung truyền và bị xoá khi CPU đọc LSR)
PE: Parity Error (=1 khi có lỗi parity và bị xoá khi CPU đọc LSR)
OE: Overrun Error (=1 khi có lỗi thu đè, nghĩa là CPU không đọc kịp dữ liệu làm cho quá trình ghi chồng lên RBR xảy ra và bị xoá khi CPU đọc LSR)
RxDR: Receiver Data Ready (=1 khi đã nhận 1 ký tự và đưa vào RBR và bị xoá khi CPU đọc RBR).

LCR (Line Control Register):

D7	D6	D5	D4	D3	D2	D1	D0
DLAB	SBCB	PS2	PS1	PS0	STB	WLS1	WLS0

DLAB (Divisor Latch Access Bit) = 0: truy xuất RBR, THR, IER, = 1 cho phép đặt bộ chia tần trong UART để cho phép đạt tốc độ truyền mong muốn. UART dùng dao động thạch anh với tần số 1.8432 MHz đưa qua bộ chia 16 thành tần số 115,200 Hz. Khi đó, tuỳ theo giá trị trong BRDL và BRDH, ta sẽ có tốc độ mong muốn.
Ví dụ như đường truyền có tốc độ truyền 2,400 bps có giá trị chia 115,200 / 2,400 = 48d = 0030h -> BRDL = 30h, BRDH = 00h.
Một số giá trị thông dụng xác định tốc độ truyền cho như sau:

Tốc độ (bps)	**BRDH**	**BRDL**
1,200	00h	60h
2,400	00h	30h
4,800	00h	18h
9,600	00h	0Ch
19,200	00h	06h
38,400	00h	03h
57,600	00h	02h
115,200	00h	01h

SBCB (Set Break Control Bit) =1: cho phép truyền tín hiệu Break (=0) trong khoảng
thời gian lớn hơn một khung
PS (Parity Select):

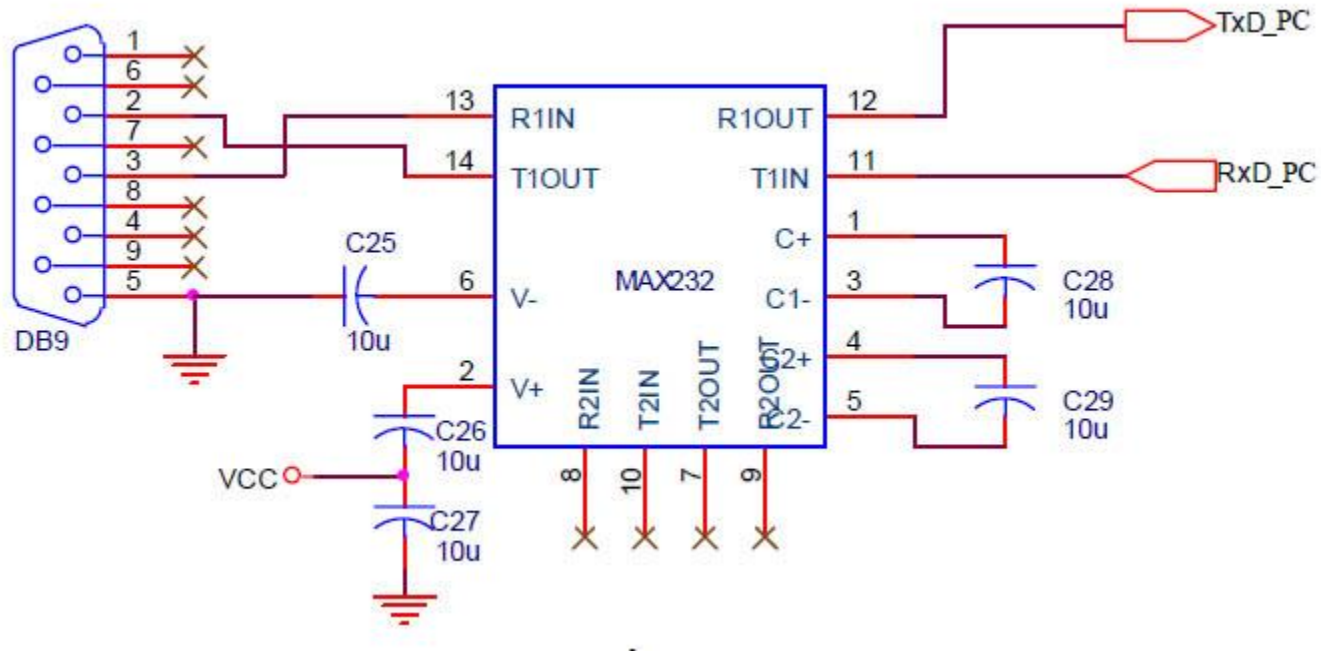
PS2	**PS1**	**PS0**	**Mô tả**
X	X	0	Không kiểm tra
0	0	1	Kiểm tra lẻ
0	1	1	Kiểm tra chẳn
1	0	1	Parity là mark
1	1	1	Parity là space

STB (Stop Bit) = 0: 1 bit stop, =1: 1.5 bit stop (khi dùng 5 bit dữ liệu) hay 2 bit stop
(khi dùng 6, 7, 8 bit dữ liệu).
WLS (Word Length Select):

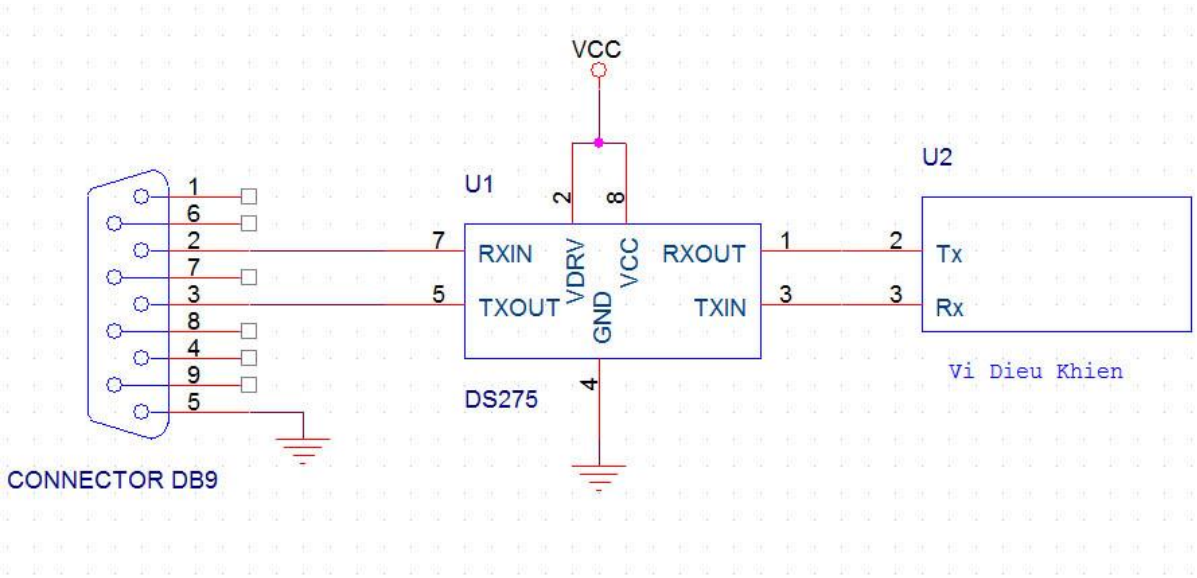
WLS1	WLS0	Độ dài dữ liệu
0	0	5 bit
0	1	6 bit
1	0	7 bit
1	1	8 bit

Tín hiệu truyền theo chuẩn RS-232 của EIA (Electronics Industry Associations). Chuẩn RS-232 quy định mức logic 1 ứng với điện áp từ -3V đến -15V (mark), mức logic 0 ứng với điện áp từ 3V đến 15V (space). Nhưng vi điều khiển sử dụng mức logic TTL với mức 0 là (0 - 0.8V) và mức 1 là (2.2 - 5V), vì vậy để giao tiếp giữa vi điều khiển và máy tính thông qua cổng com ta cần có mạch chuyển đổi giữa hai tín hiệu này

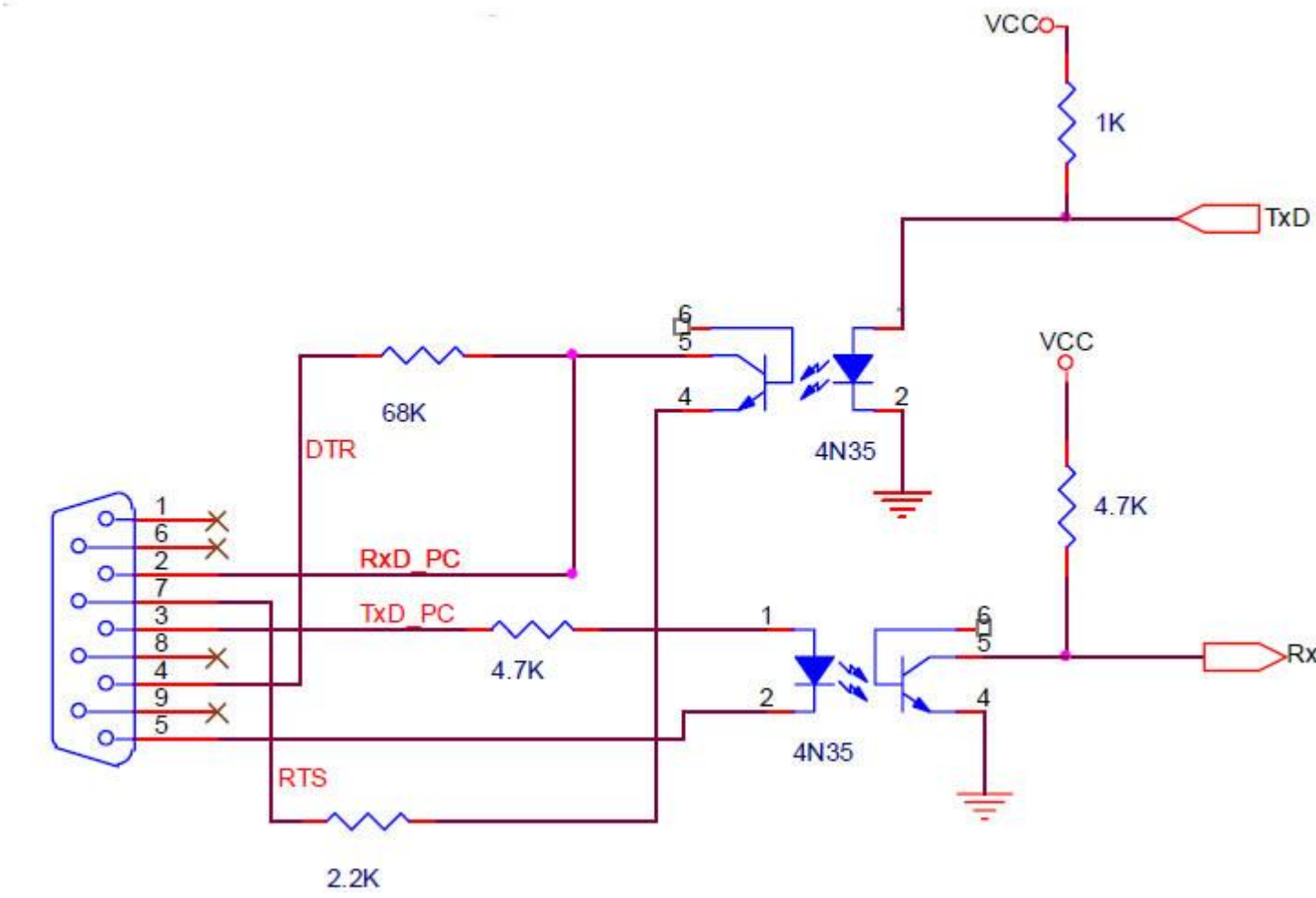
Các vi mạch thường sử dụng là MAX232 của Maxim hay DS275 của Dallas. Mạch chuyển mức logic mô tả như sau:



Hình 1: Mạch chuyển mức logic TTL RS232 sử dụng Max 232



Hình 2 : Mạch chuyển mức logic TTL RS232 sử dụng DS275



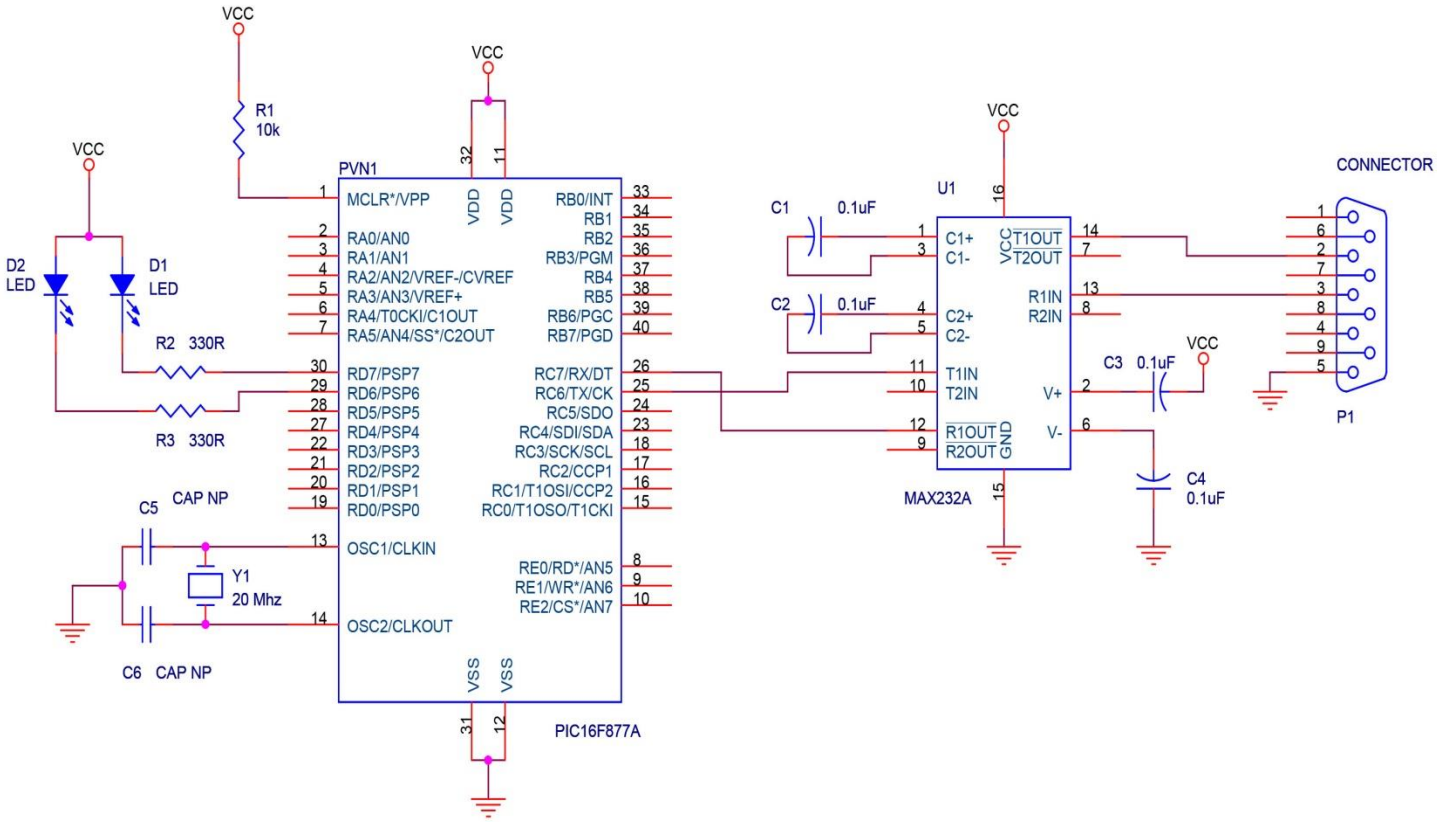
Giao tiếp giữa vi điều khiển PIC với máy tính thông qua cổng COM (RS232)

Ví dụ 1:

Trong ví dụ này ta viết chương trình điều khiển hai led (D1 và D2) kết nối với chân RD7 và RD6 (Port D) bằng máy tính thông qua cổng COM. Vi điều

khiển trong ví dụ này là 16f877A, thạch anh tạo dao động cho PIC là 20Mhz

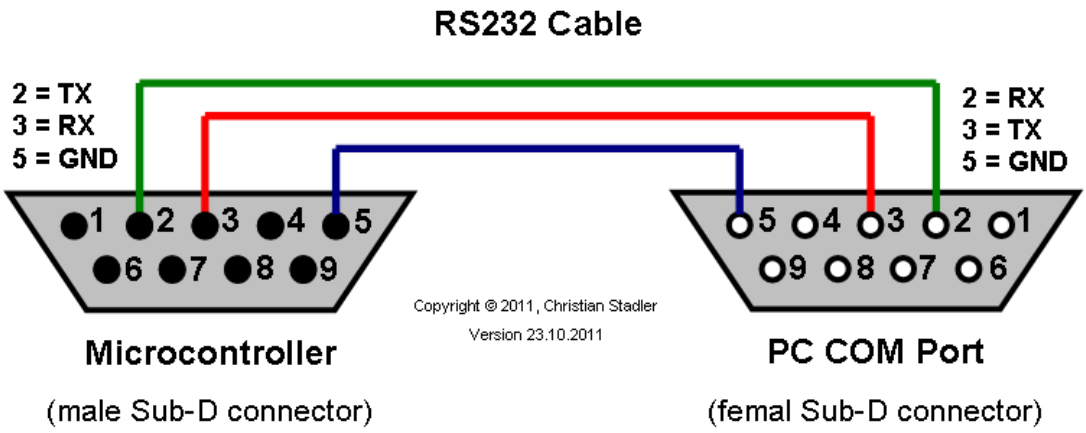
1. Phần cứng giao tiếp:



Hình sơ đồ phần cứng của ví dụ 1

Hình cáp RS232

Để kết nối từ board mạch đến cổng COM trên máy tính chúng ta cần cáp RS232, nếu ta không có dây cáp sẵn ta có thể đấu dây như hình bên dưới



Hình cách đấu dây cáp

2. Phần mềm:

1. Truyền thông với RS232 với trình biên dịch CCS

Trình biên dịch CCS cung cấp cho chúng ta các cách thức rất đơn giản để giao tiếp thông qua RS232. Nó ẩn tất cả các thiết lập thanh ghi. Chúng ta chỉ cần cung cấp một vài thông tin, những thứ còn lại được thực hiện bởi chương trình biên dịch

```
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

- Chỉ thị tiền xử lý *#use delay(clock=20000000)* cung cấp cho trình biên dịch thông tin về tốc độ thạch anh mà vi điều khiển PIC được chạy, do ví dụ này sử dụng thạch anh 20Mhz, nên ta khai báo Clock=20000000

- Chỉ thị *#use rs232* cung cấp cho trình biên dịch thông tin về các tham số cấu hình RS232 sẽ sử dụng để giao tiếp:

- baud=9600 : khai báo tốc độ baud sử dụng là 9600

- parity = N : khai báo có sử dụng bit kiểm tra chẵn lẻ hay không, ở đây ta không kiểm tra nên thiết lập parity = N

- xmit =PIN_C6 : Chân truyền dữ liệu là PIN C6

- rcv=PIN_C7 : Chân nhận dữ liệu là PIN C7

- bits=8 : Số bit dữ liệu là 8 bít

Các hàm hỗ trợ truyền dữ liệu:

```
int value = 1;
putc('A');                              /* gửi một ký tự thông qua RS232 */
puts("Test-String");                    /* gửi một chuỗi thông qua RS232 */
printf("Transmit a value: %d", value);  /* gửi một chuỗi theo định dạng qua
RS232 */
```

Các hàm hỗ trợ nhận dữ liệu:

```
char ch;
```

```
char string[32];
ch = getc();                          /* nhận 1 ký tự đơn */
gets(string);                         /* nhận 1 chuỗi  */
```

Ngắt nhận của RS232

```
#INT_RDA          /*Ngắt khi nhận dữ liệu*/

void Ngat_Nhan

{

//Hàm xữ lý sự kiện ngắt nhận

//Tại đây ta có thể xữ lý dữ liệu vừa nhận được

}
```

Dưới đây là chương trình viết cho PIC

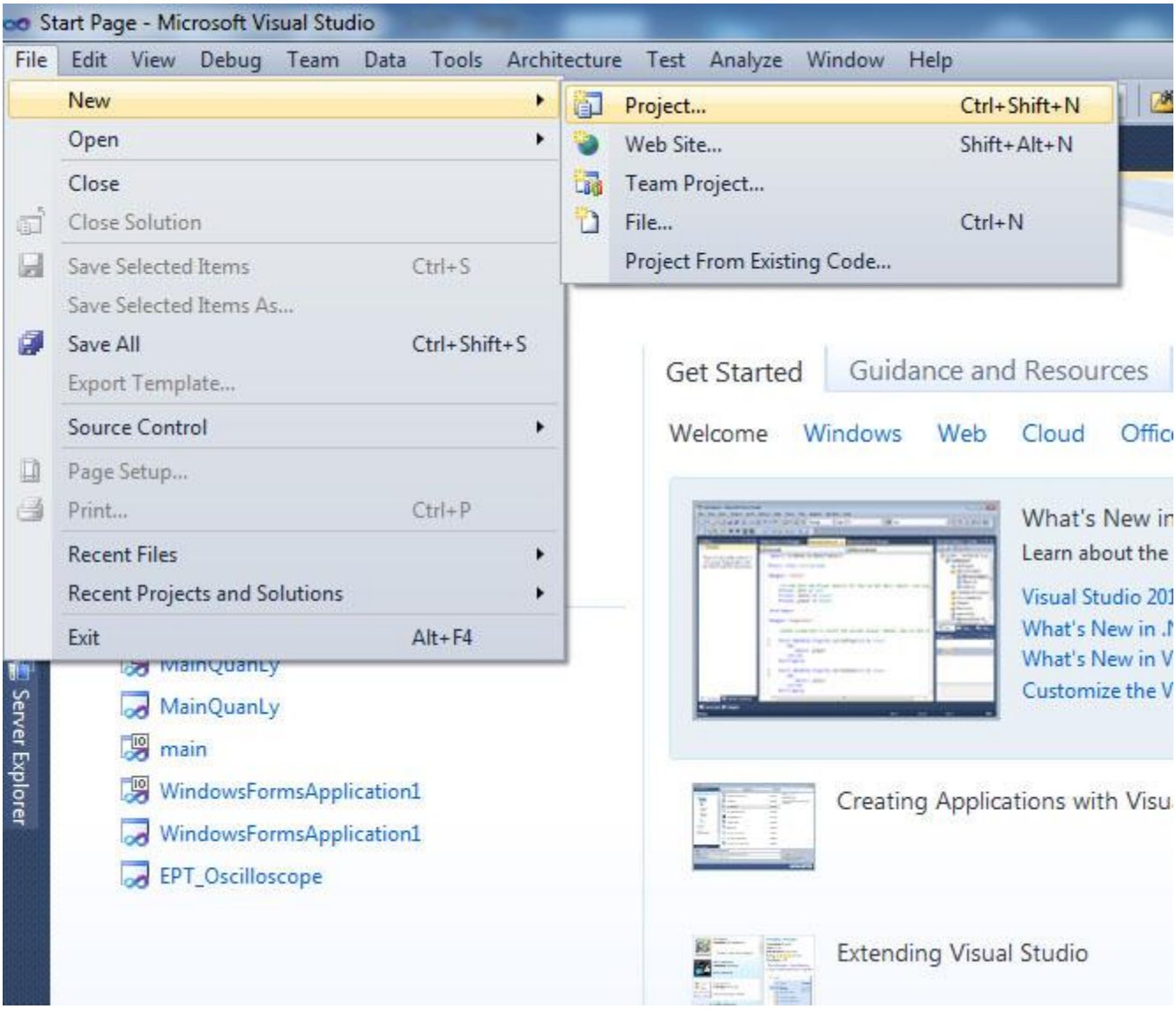
```
#include <16F877a.h>
#fuses HS,NOWDT,NOPROTECT,BROWNOUT,PUT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

//============================
void main()
{
char c;

Set_tris_D(0x00); // thiết lập cổng I/O của port D là cổng xuất
Output_D(0xFF); // tắt tất cả các LED kết nối đến port D
while(1)
  {
    c = getc();
    if(c=='0')
   {
   output_d(0b11111111);
   }
   if(c=='1')
   {
   output_d(0b11111110);
   }
   if(c=='2')
   {
   output_d(0b11111101);
   }
   if(c=='3')
   {
   output_d(0b11111100);
   }
```

```
 }
}
```

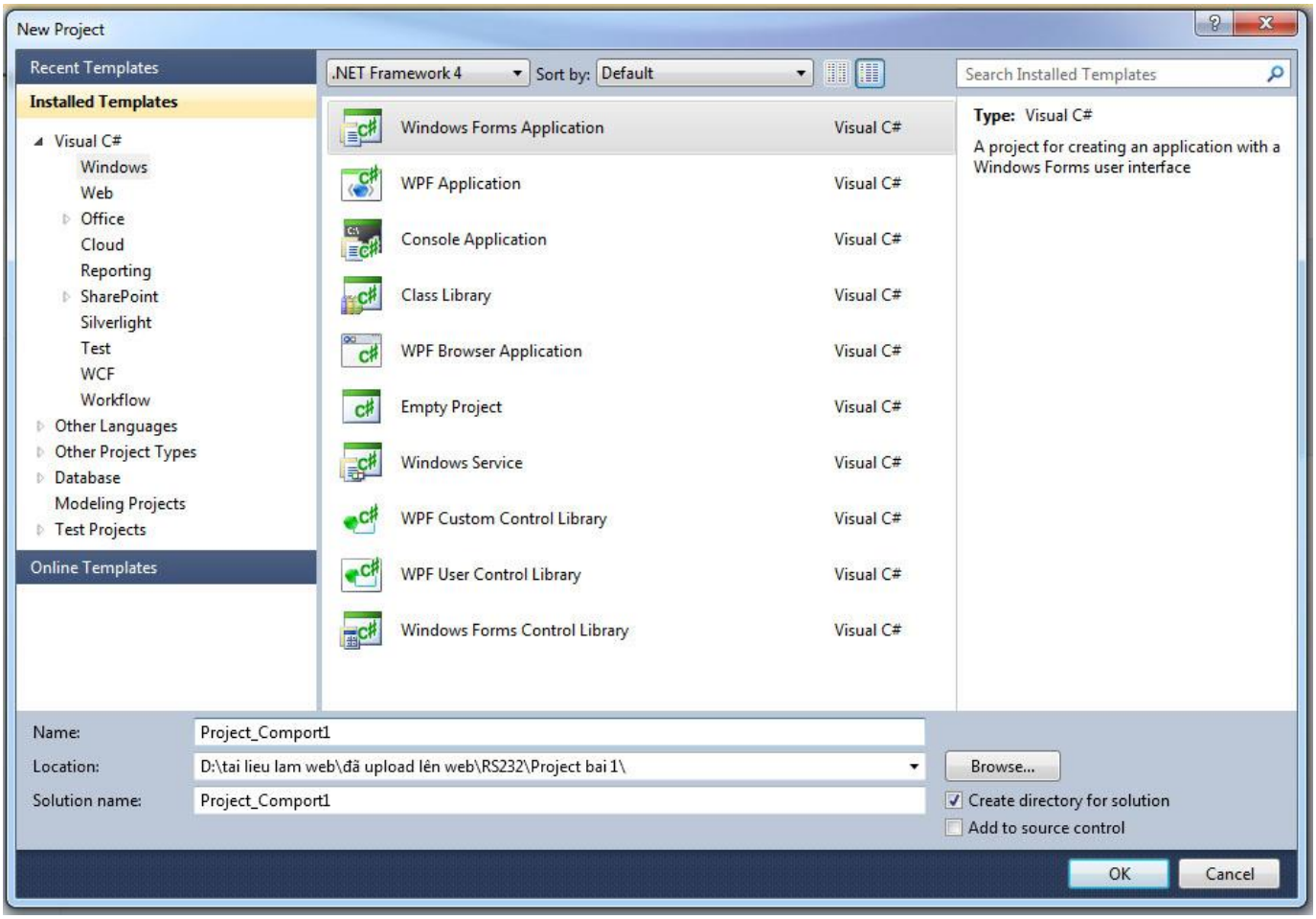
Sử dụng C# để viết chương trình giao tiếp giữa máy tính với vi điều khiển PIC

Trong bài này ta sử dụng C# trong MICROSOFT VISUAL STUDIO 2010 để viết để viết chương trình giao tiếp giữa vi điều khiển với máy tính qua cổng COM.

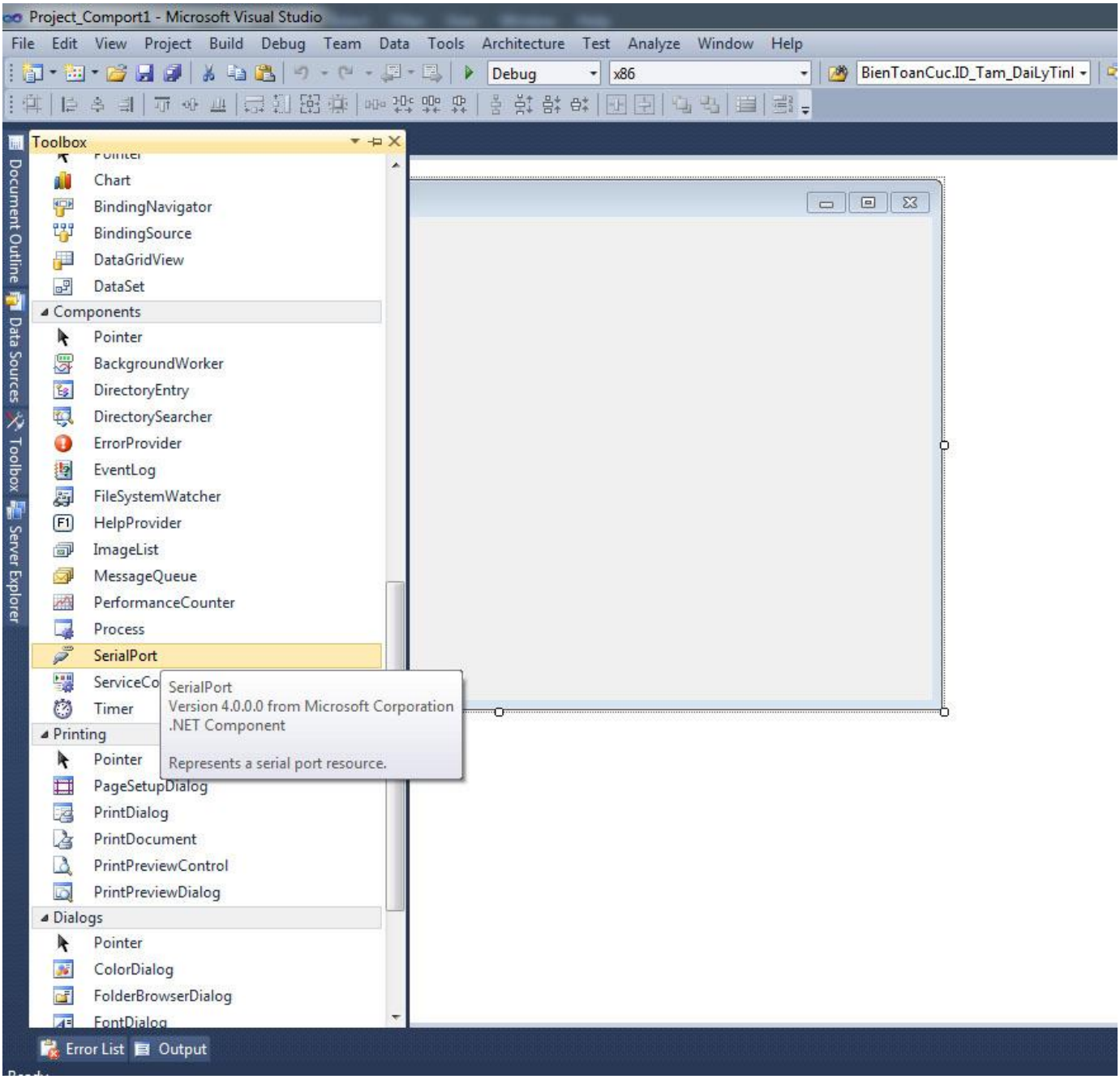
Đầu tiên ta tạo mới một project: **New -> Project**



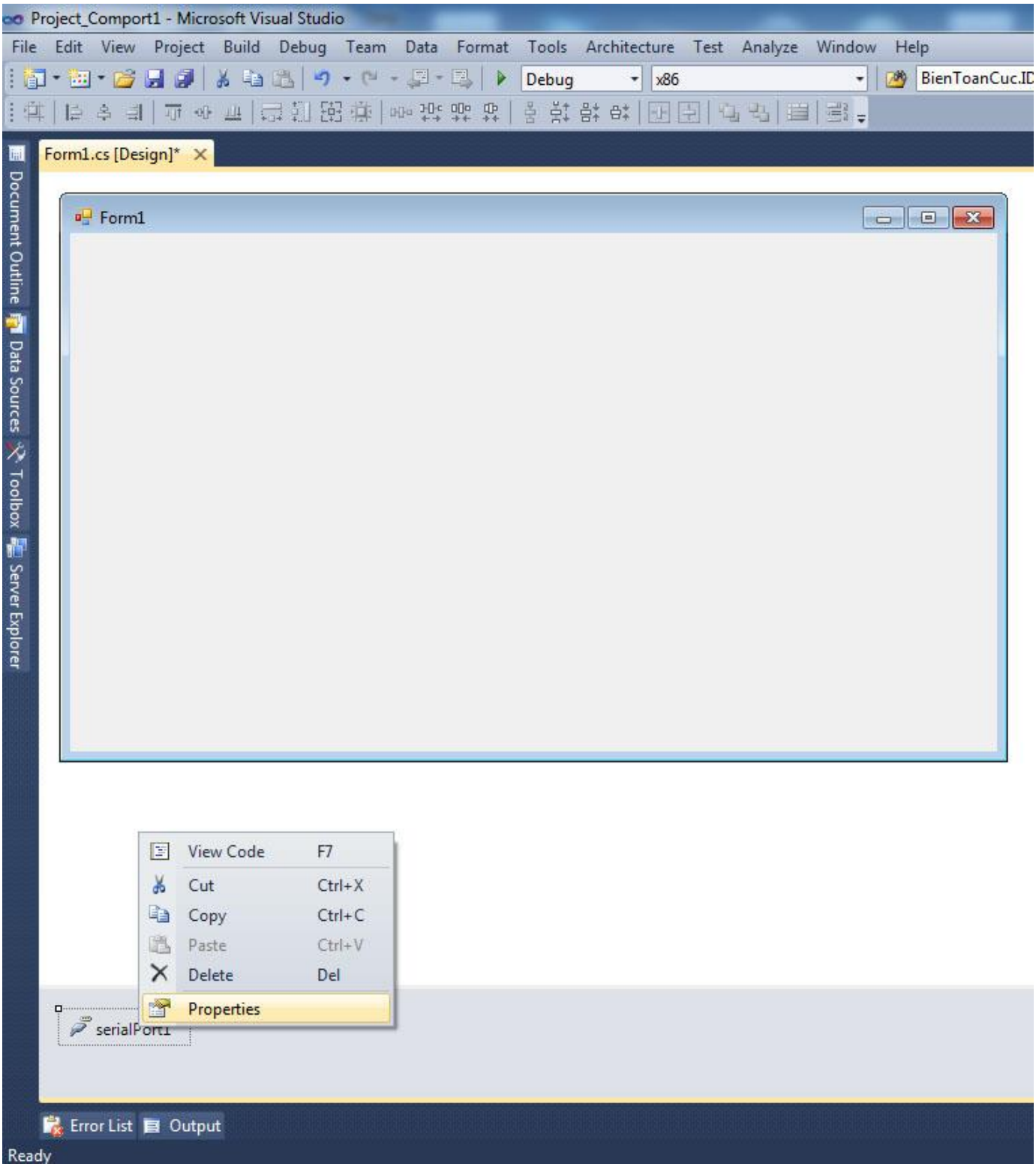
Chọn **Windows Forms Application** và đặt tên cho chúng. Ở ví dụ dưới đặt là Project_comport1



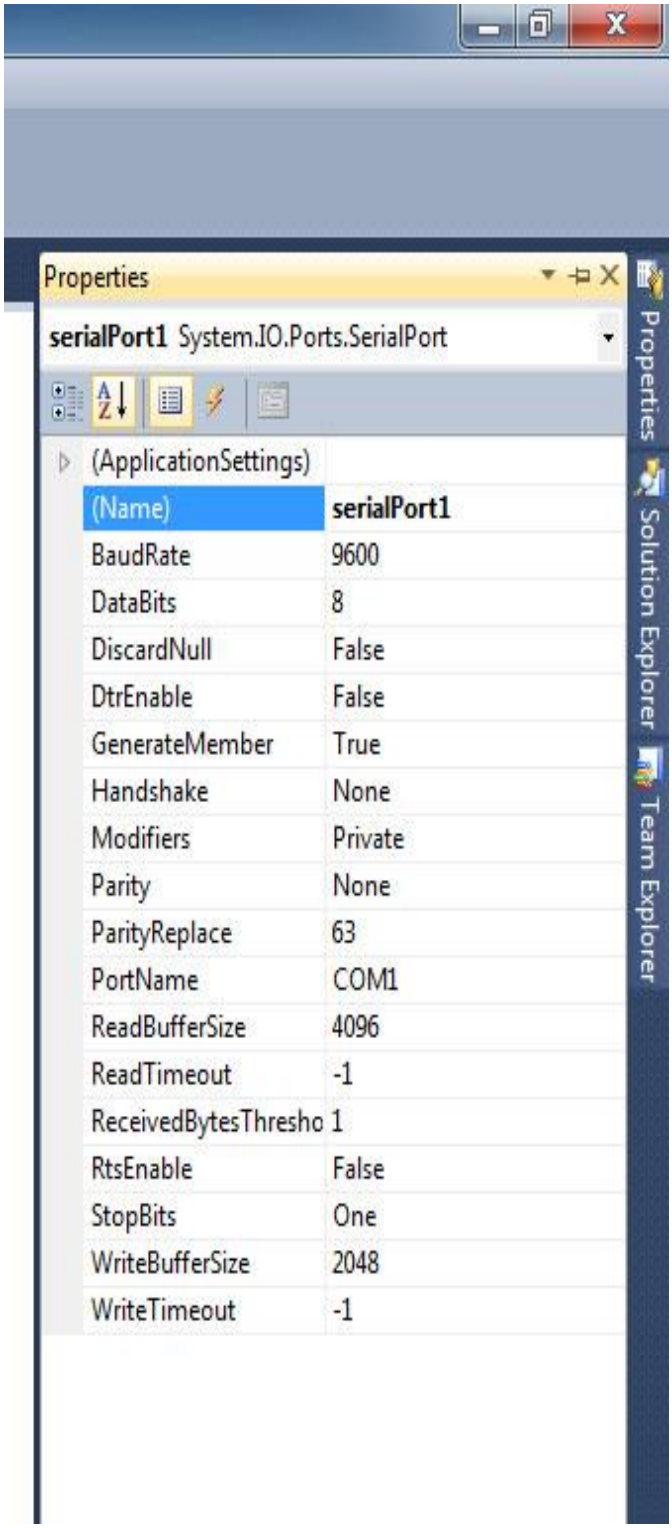
Kéo lớp **SerialPort** ở cửa sổ toolbox vào form mới vừa được tạo (Nếu ta không thấy cửa sổ toolbox ta vào menu View -> Toolbox).



Nhấp chuộc phải lên control vừa mới kéo thả , chọn **Properties** như hình dưới:



Hình dưới là cửa sổ của Properties của serialPort1



Ở đây ta quan tâm đến các properties sau:

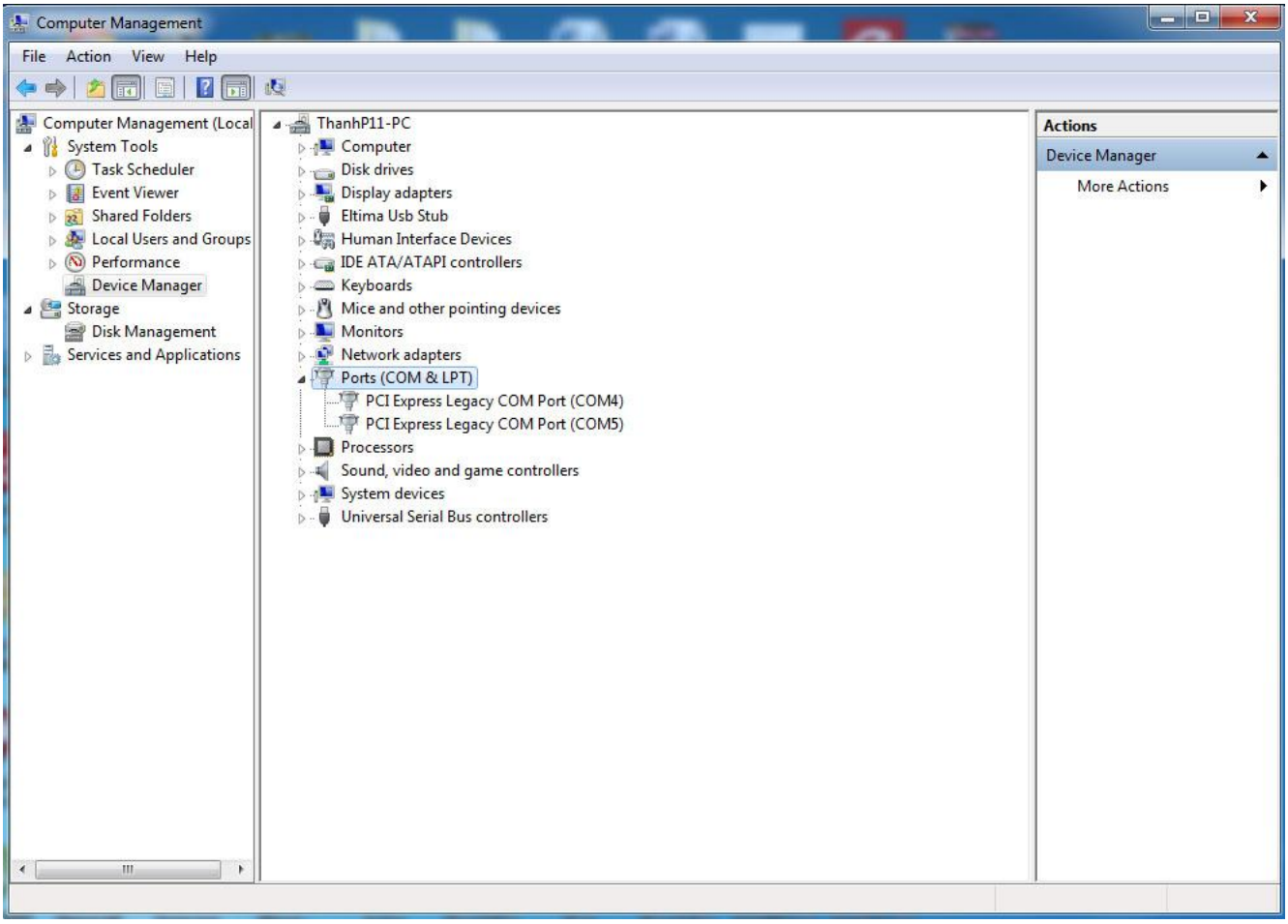
BaudRate : Chỉ định tốc độ baudRate (ta có thể chọn 1 trong các giá trị: 19200, 9600, 4800, 2400, 1800, 1200, 600, 300, 150, 110)

DataBits : Số bit dữ liệu (ta có thể chọn 1 trong các giá trị: 6, 7, 8)

Parity : Kiểm tra chẵn lẻ (ta có thể chọn 1 trong các giá trị: Odd, None, Even)

StopBits : số bit stops (ta có thể chọn 1 trong các giá trị: 1, 1.5, 2)

PortName: tên cổng vật lý mà ta muốn kết nối đến vi điều khiển, muốn biết máy tính có bao nhiêu cổng Com ta có thể Device Manager bằng cách: Nhấp chuột phải lên My computer - > manage - >Device Manager -> Ports, hình bên dưới là các cổng Com trên máy tính mình đang có đó là COM4 và COM5



Ta thiết lập lại các thông số của properties **serialPort1** như sau:

BaudRate : 9600

DataBits : 8

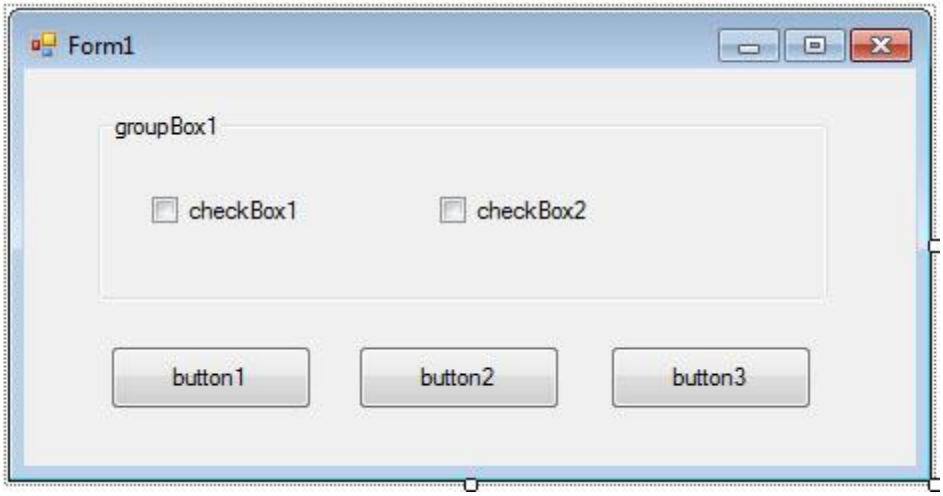
Parity : None

StopBits : 1

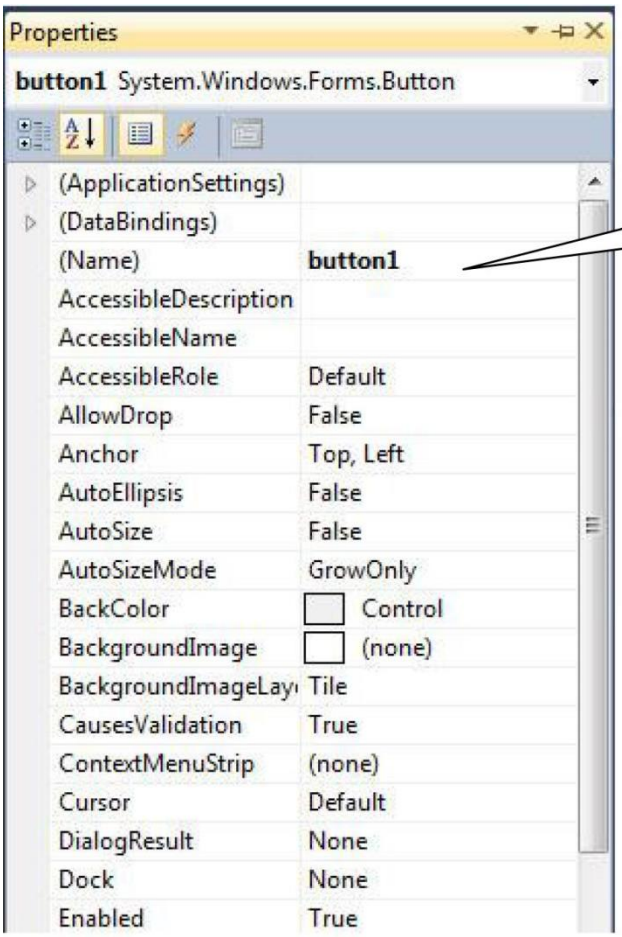
PortName: COM4 (Đối với máy tính của bạn phải thiết lập theo tên cổng COM mà bạn đang muốn kết nối đến vi xữ lý).

Các thông số còn lại để theo mặc định.

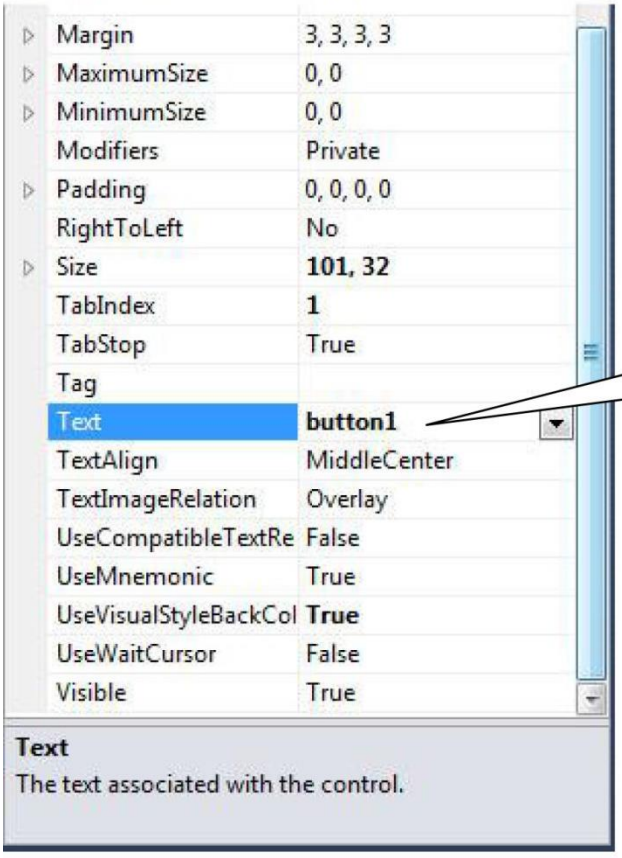
Thêm vào các control trong cửa sổ toolbox để được như hình bên dưới:



Ta tiến hành đặt tên lại tên (Name) và nhãn hiển thị (Text) cho mỗi control bằng cách nhấn chuộc phải lên từng control (Ví dụ nút nhấn) và chọn Properties

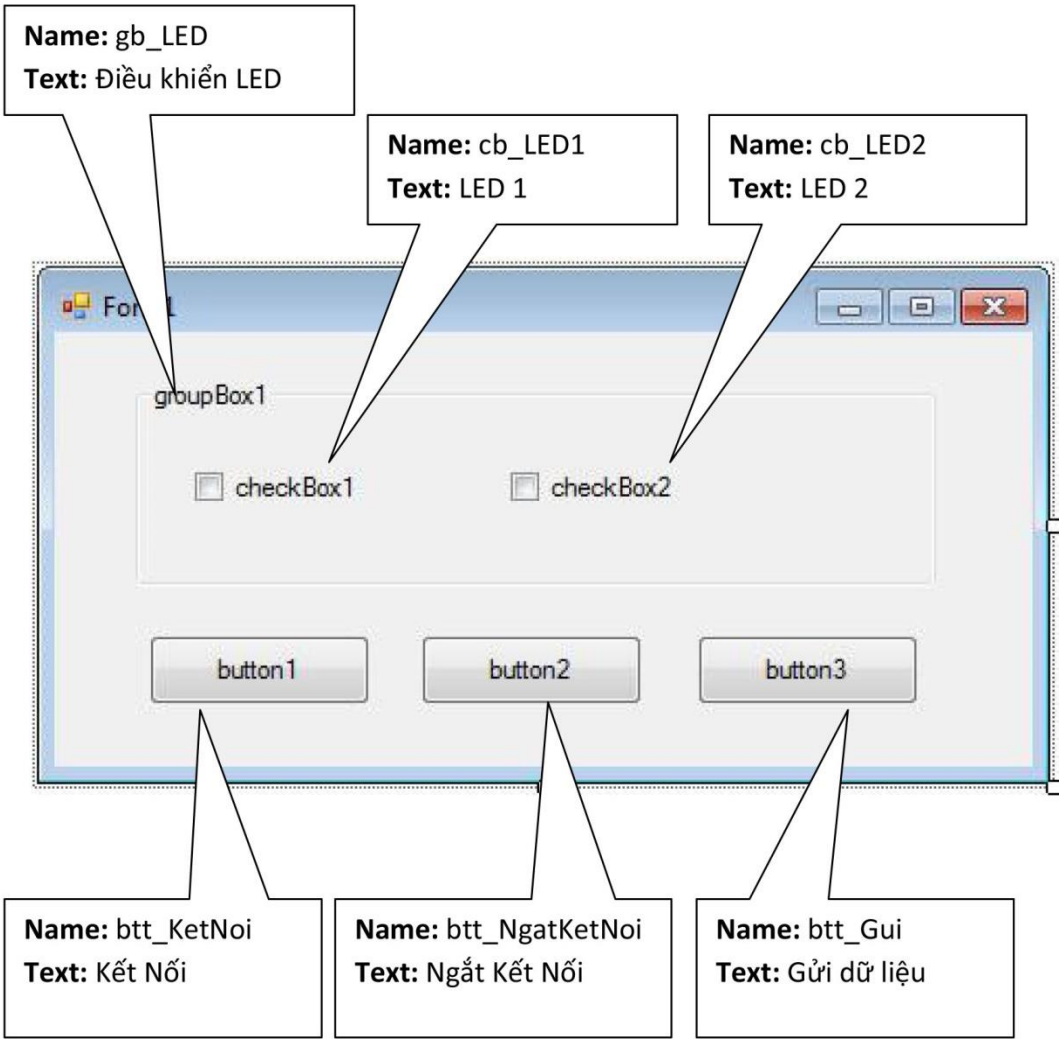


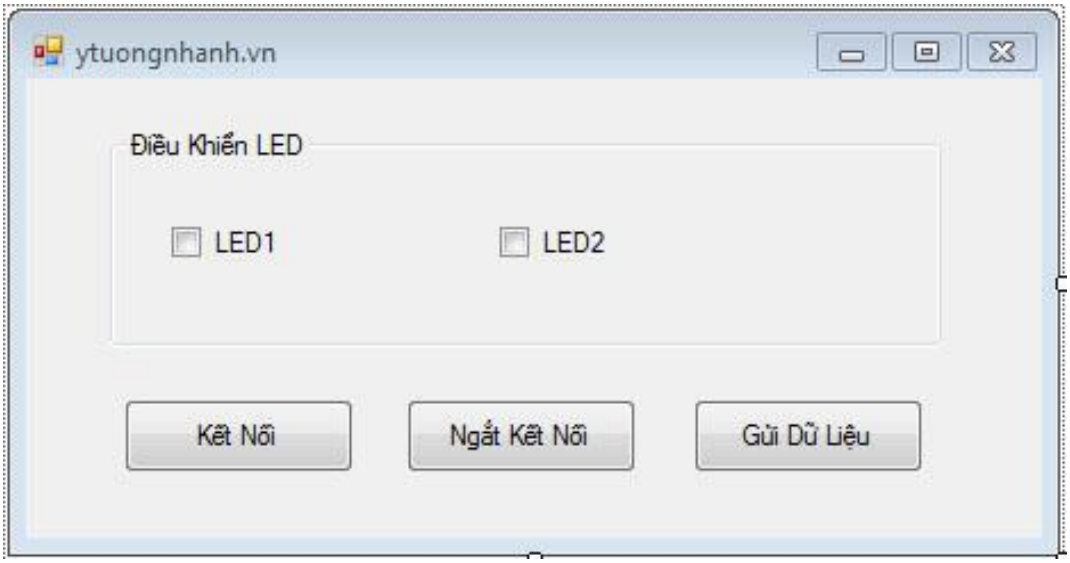
Tên Control



Tên nhãn của control

Ta tiến hành đặt tên lần lượt từng control như hình bên dưới:





Click đúp chuột lên nút **Kết Nối** để mở cửa sổ soạn thảo code, chèn đoạn code sau đây vào :

```
        if (serialPort1.IsOpen == false)
        {
            serialPort1.Open();
            MessageBox.Show("Bạn đã mở thành công");
        }
        else
        {

            MessageBox.Show("Cổng COM đang mở");
        }

/* Lệnh trên kiểm tra cổng serialPort đã mở chưa, nếu chưa mở thì mở bằng lệnh serialPort1.Open()
, và hiển thông điệp "Bạn đã mở thành công" . Ngược lại báo "Cổng COM đang mở" */
```

Làm tương tự với nút **Ngắt Kết Nối**

```
                    if (serialPort1.IsOpen == true)
                    {
                        serialPort1.Close();
                    }

            /* Lệnh  serialPort1.Close() ngắt kết nối */
```

Nút **Gửi Dữ Liệu**:

```
        if (serialPort1.IsOpen == true)
        {
            if (cb_LED1.Checked == false && cb_LED2.Checked == false)
            {
                serialPort1.Write("0");
            }
```

```
            if (cb_LED1.Checked == true && cb_LED2.Checked == false)
            {
                serialPort1.Write("1");
            }
            if (cb_LED1.Checked == false && cb_LED2.Checked == true)
            {
                serialPort1.Write("2");
            }
            if (cb_LED1.Checked == true && cb_LED2.Checked == true)
            {
                serialPort1.Write("3");
            }
        }
        else
        {
            MessageBox.Show("Bạn chưa mở cổng COM");
        }
```

Mô phỏng giao tiếp RS232 giữa máy tính và vi điều khiển PIC bằng Proteus

Nếu bạn không có phần cứng để kiểm tra mạch ta có thể dùng phần mềm để mô phỏng, bài này sẽ hướng dẫn bạn cách để mô phỏng

Các phần mềm cần có:

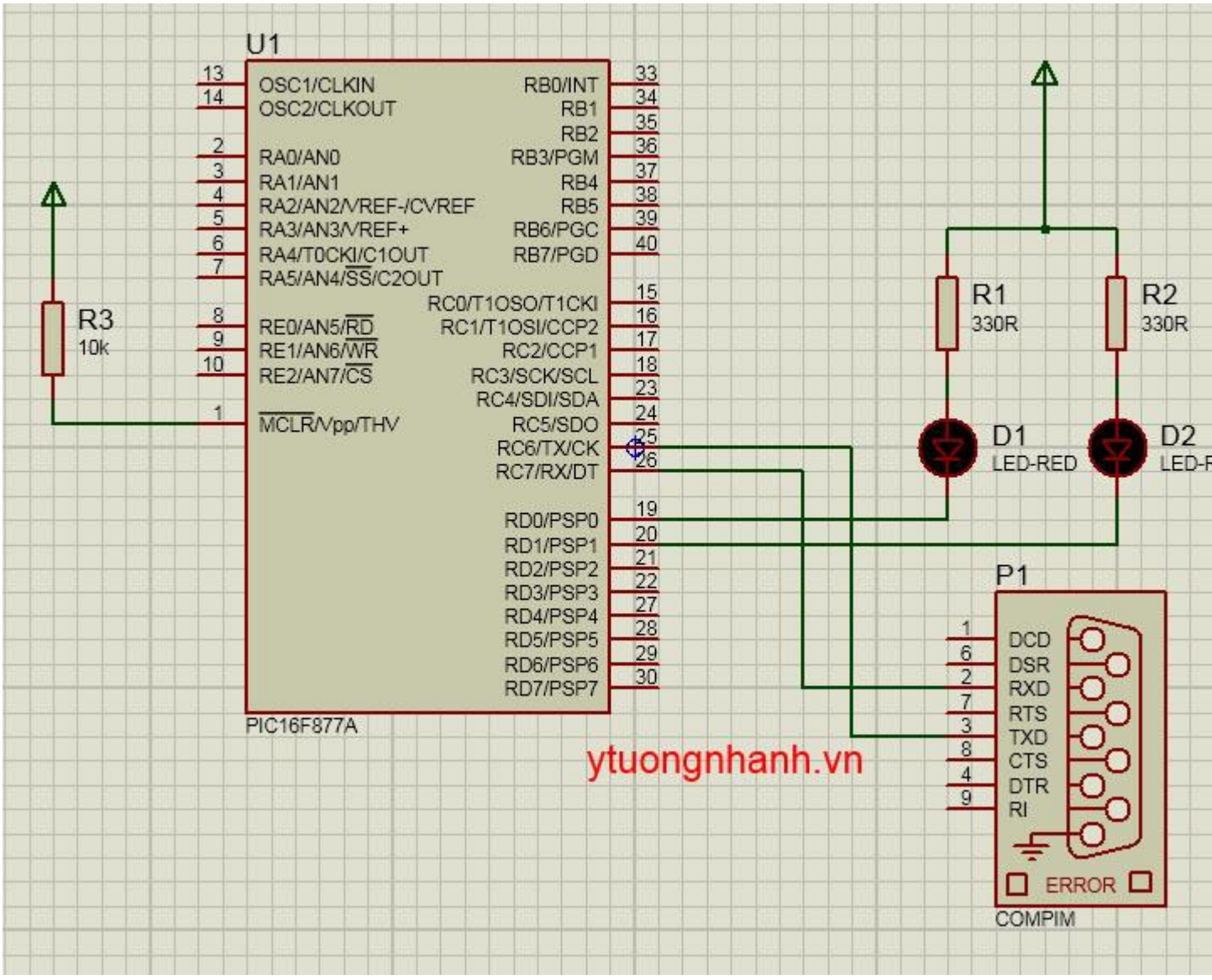
Virtual Serial Port: Trước tiên bạn tải về phần mềm **virtual serial port** tại địa chỉ : http://www.eltima.com/products/vspdxp/, phần mềm có tác dụng tạo kết nối ảo giữa mộ cặp cổng COM, trong bài này ta cần tạo kết nối giữa cổng COM trong máy tính với Cổng co

ảo của phần mềm mô phỏng proteus. Link download bảng Virtual serial Port Full

Phần mềm mô phỏng mạch Proteus : Trong bài này mình sử dụng Proteus 8.0

Bước 1: Sau khi cài đặt phần mềm xong ta mở phần mềm proteus và vẽ mạch như hình dưới:

*Trong proteus ta tìm cổng **COM** với tên là **COMPIM***



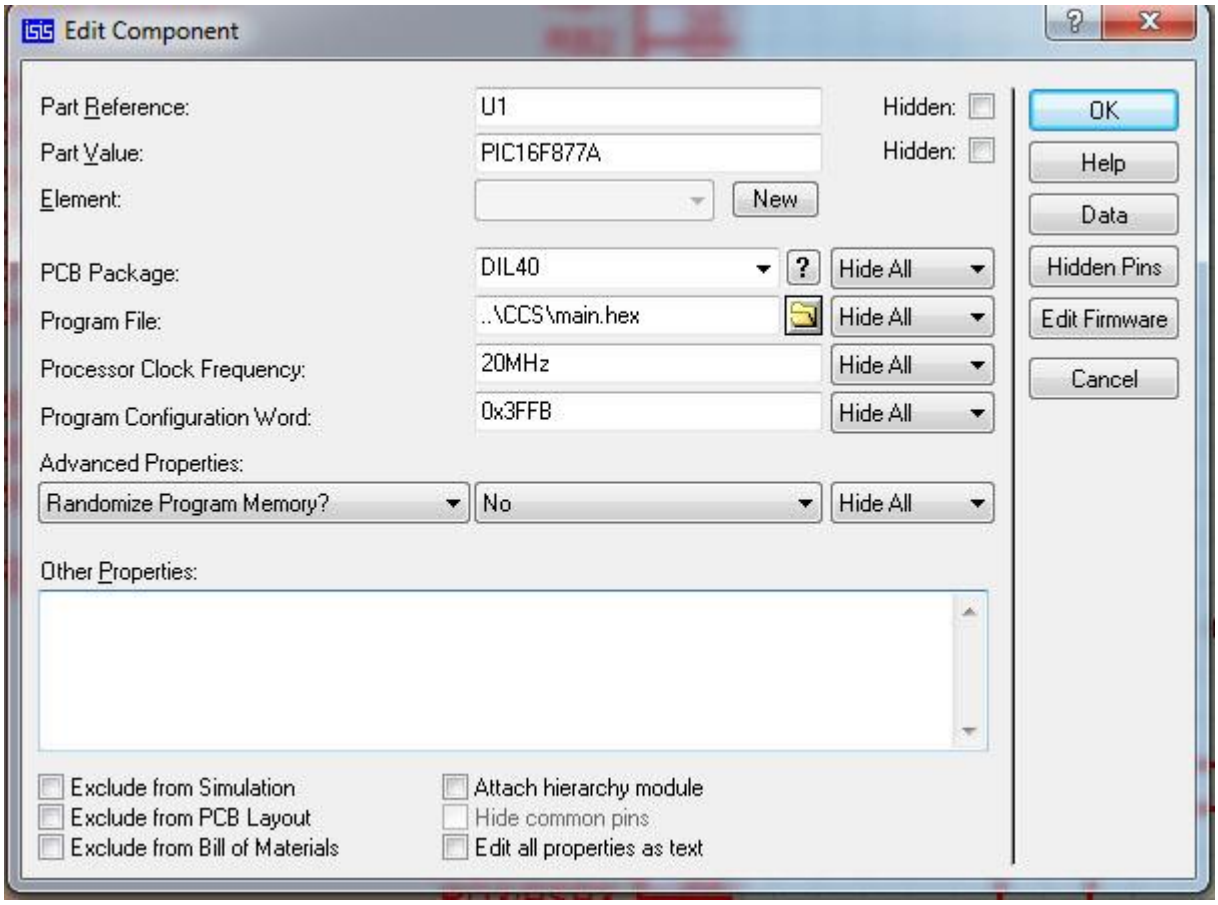
Bước 2: Tiếp theo ta thiết lập các thông số cho mạch

- Kích đúp chuột lên PIC16f877a , một cửa sổ Edit Component cho vi điều khiển được bật lên , ta chú ý hai thông số:

Program File : chứa đường dẫn đến file hex ta biên dịch bằng chương trình CCS ở phần đầu.

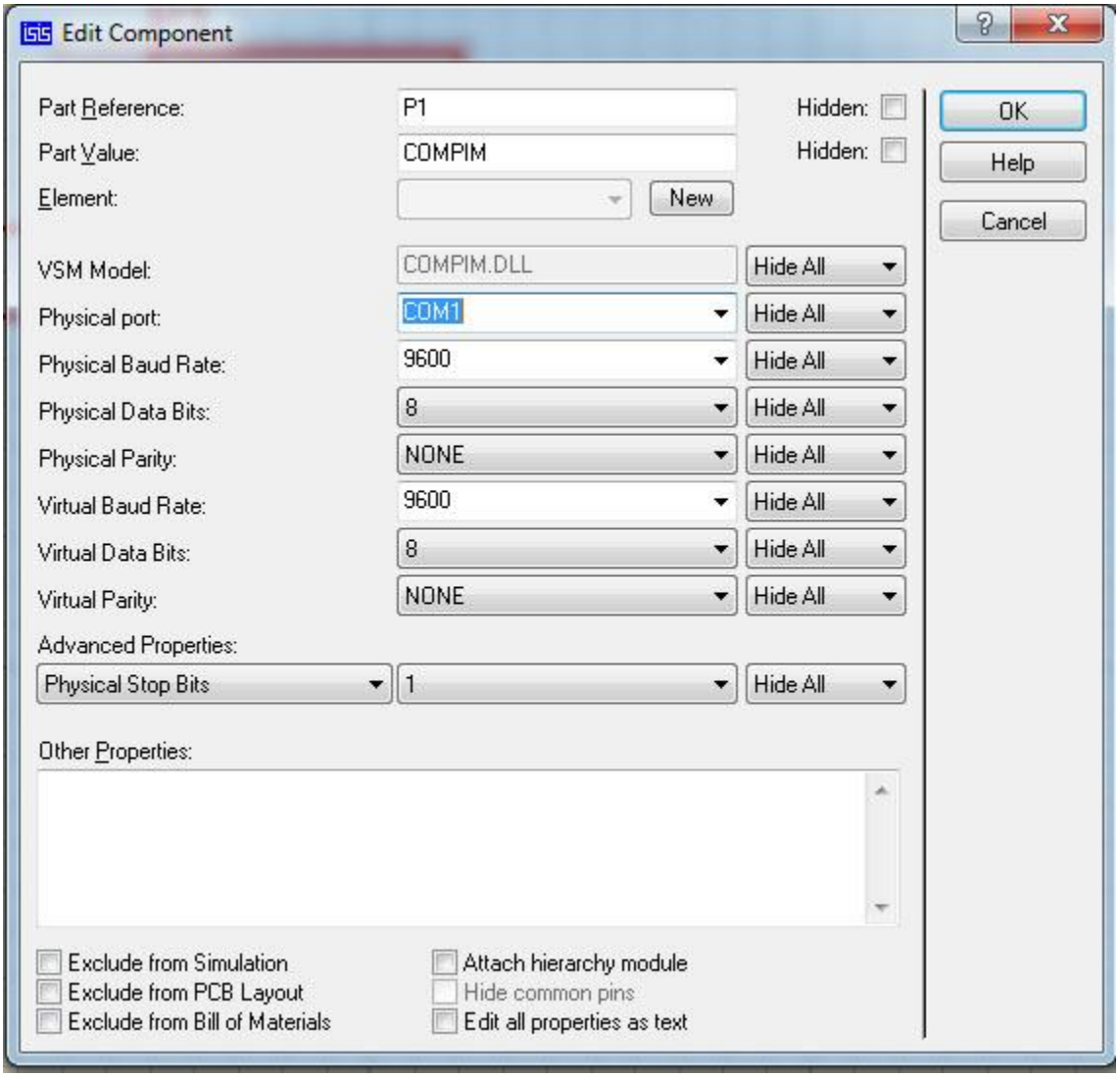
Processor Clock Frequency: Khai báo tần số thạch anh cấp cho vi điều khiển, ở đây ta chọn 20 Mhz

Nhấn OK để hoàn tất thiết lập

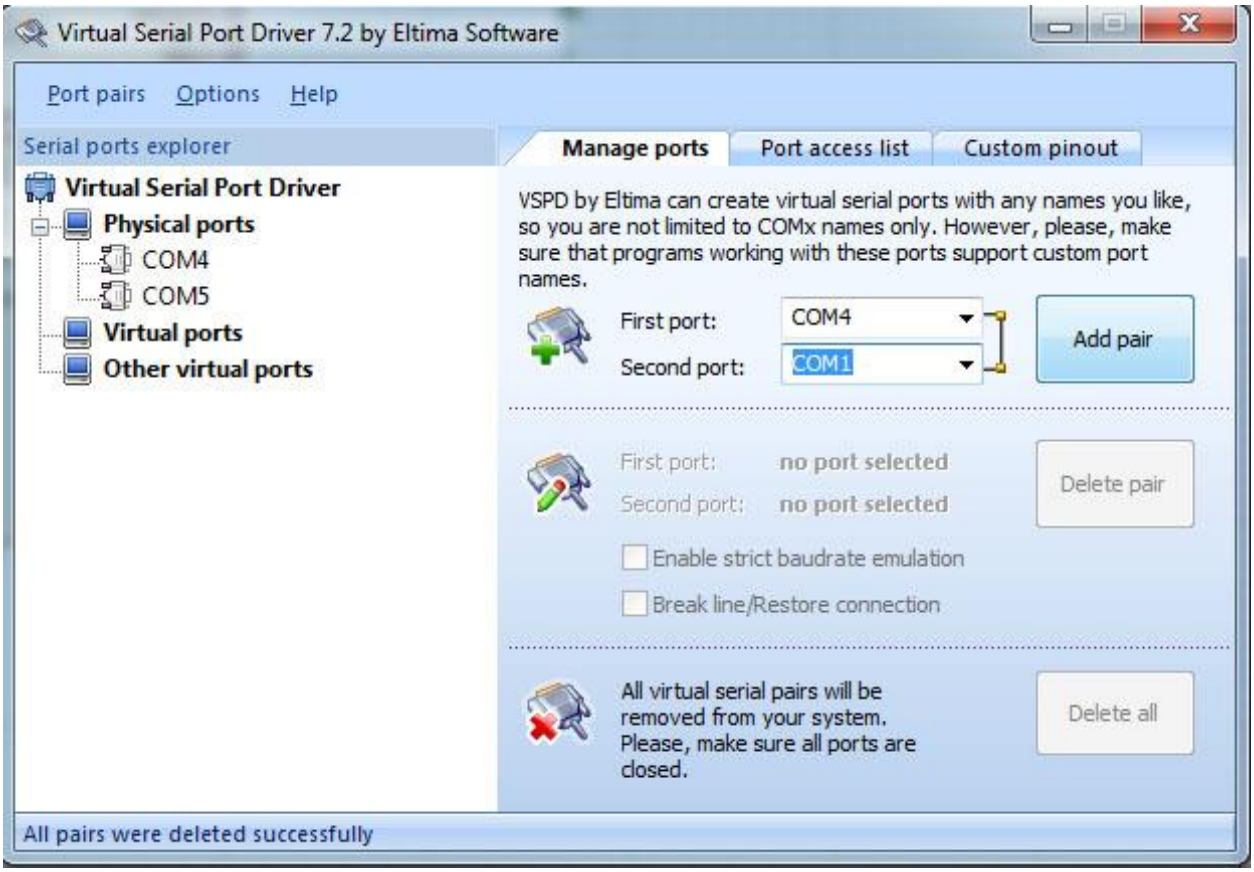


Hình : Cửa sổ Edit Component của vi điều khiển PIC16F877A.

Ta kích đúp chuột lên cổng COM để mở cửa sổ Edit Component, ta thiết lập các thông số như hình bên dưới:



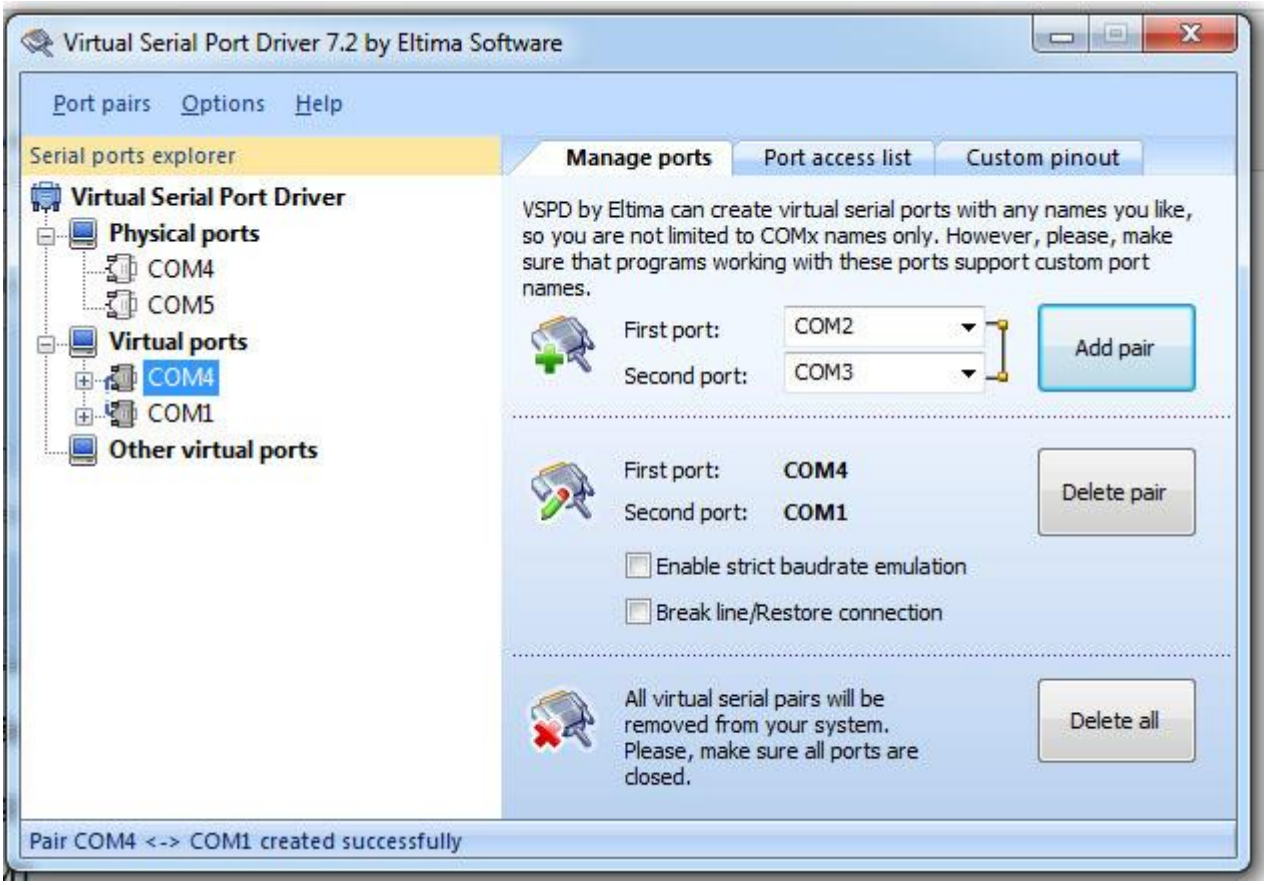
<u>Bước 2:</u> Ta mở phần mềm **Virtual Serial Port Driver** lên:



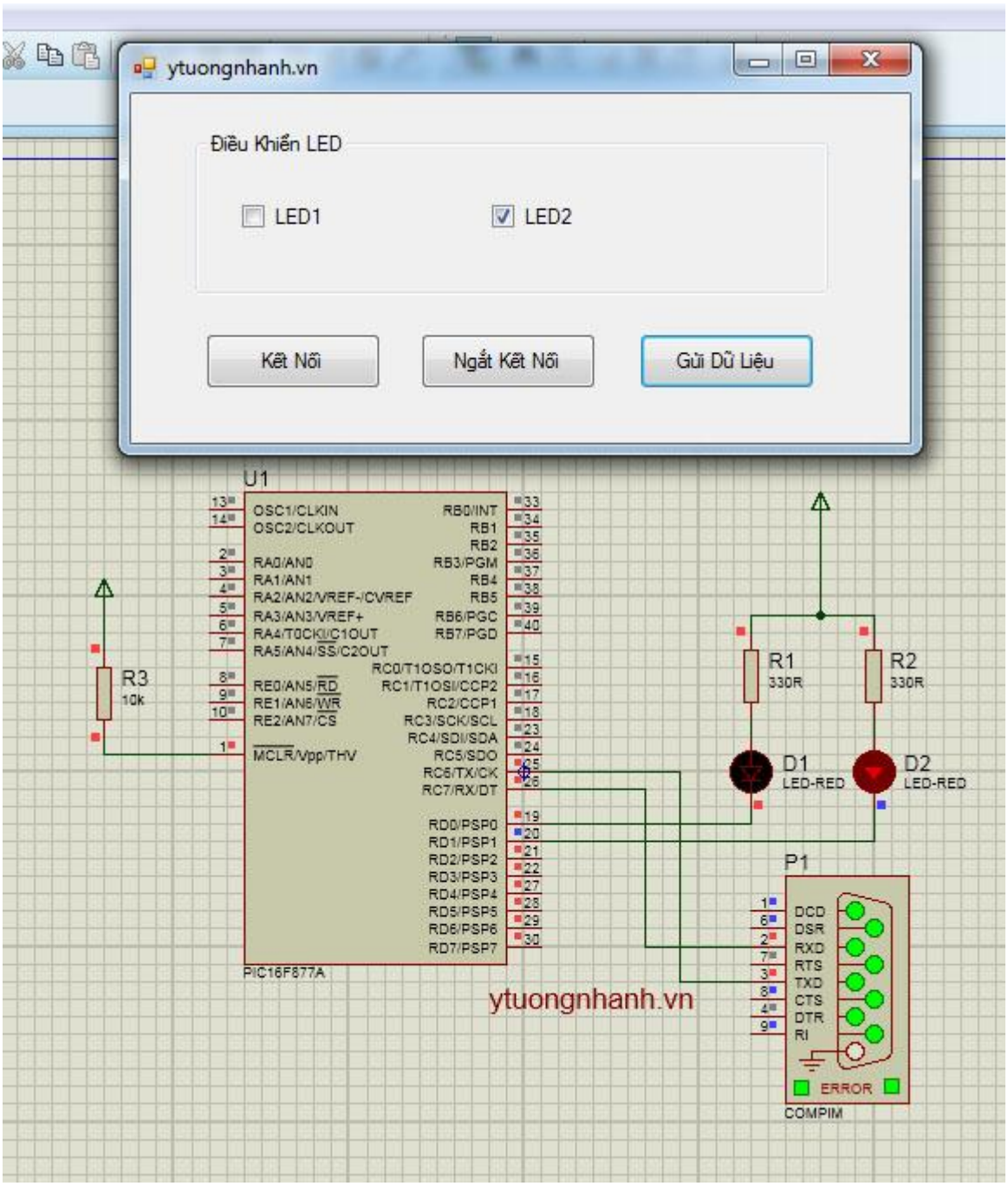
Chọn **First port** là tên cổng COM vật lý ta muốn kết nối đến board vi điều khiển PIC thực tế, do chương trình viết bằng C# lúc đầu mình chọn là COM4 nên ở đây chọn là **COM4**

Second port là tên cổng COM của phần mềm Proteus, ở trên mình chọn cổng COM1 , nên Second port mình chọn là COM1

Sau đó nhấn nút **Add pair ,** vậy là ta đã tạo một kết nối ảo giữa COM4 và COM1.



<u>Bước 3:</u> Cuối cùng, mở phần mềm viết bằng C# lên, nhấn nút **Kết Nối ,** chọn Led muốn điều khiển và nhấn nút **Gửi Dữ Liệu** để kiểm tra



Ví dụ 2: Trong chương trình ở ví dụ 1 vi điều khiển chờ nhận dữ liệu từ cổng COM trong một vòng lặp vô tận, vì vậy vi điều khiển sử dụng gần như toàn bộ thời gian để chờ dữ liệu gửi từ máy tính xuống và xữ lý. Ở ví dụ này ta sử dụng ngắt nhận dữ liệu, khi có dữ liệu trong bộ đệm nhận vi điều khiển tạo ra một ngắt , trong chương trình ngắt vi điều khiển đọc dữ liệu từ bộ đệm nhận và xữ lý dữ liệu nhận được. Còn nếu không có dữ liệu trong bộ đệm nhân, vi

điều khiển được giải phóng để thực thi các công việc khác trong chương trình chính.

Chương trình bài 1 được viết lại như sau:

```
#include <16f877a.h>
#fuses HS,NOWDT,NOPROTECT,BROWNOUT,PUT,NOLVP
#use delay(clock=20000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

//============================
#INT_RDA
void NgatNhan_RS232()
{
   char c;
   c = getc();
    if(c=='0')
   {
   output_d(0b11111111);
   }
   if(c=='1')
   {
   output_d(0b11111110);
   }
   if(c=='2')
   {
   output_d(0b11111101);
   }
   if(c=='3')
   {
   output_d(0b11111100);
   }
}
void main()
{

enable_interrupts(GLOBAL); // Cho phép ngắt toàn cục
enable_interrupts(INT_RDA); // cho phép ngắt nhận
Set_tris_D(0x00); // thie^'t la^.p co^?ng I/O cu?a port D là co^?ng xua^'t
Output_D(0xFF); // ta('t ta^'t ca? các LED ke^'t no^'i ?e^'n port D
while(1)
  {

  }
}
```