Chương trình giao tiếp RS232 đơn giản qua Matlab

Để biết thêm chi tiết về giao tiếp với PC qua cổng RS232 qua Matlab các bạn dùng Help của Matlab và search với từ khóa: serial.

Việc giao tiếp này cũng rất dễ dàng thực hiện, không có gì là quá cao siêu cả. Mình đã giao tiếp thành công rùi. Để test nó các bạn hãy đấu tắt 2 chân 2 và 3 (TX và RX) của cổng COM lại.

Đầu tiên mình sẽ đưa ra môt chương trình thật là đơn giản, thiết lập ít tham số, còn chi tiết về thiết lập tham số nó thế nào? ý nghĩa ra sao? thì sẽ nói sau, mình cứ làm đơn giản trước rùi phức tạp -> hiểu hết về thiết lập này sau.

Bài này mình giới thiệu cách tạo đối tượng, kết nối, viêt hàm callback.

Tạo đối tượng:
Chúng ta gõ lệnh và kết quả hiện luôn (nhớ là k có dấu ; ở cuối lệnh
Code:

```
>> s = serial('COM1')


   Serial Port Object : Serial-COM1


   Communication Settings

      Port:               COM1

      BaudRate:           9600

      Terminator:         'LF'


   Communication State

      Status:             closed

      RecordStatus:       off


   Read/Write State

      TransferStatus:     idle

      BytesAvailable:     0

      ValuesReceived:     0

      ValuesSent:         0
```

Như vậy đối tượng là Serial-COM1, tốc độ 9600,..

Tiếp theo, chúng ta xem các tham số của đối tượng như thế nào bằng lệnh get(s):
Code:

```
>> get(s)

    ByteOrder = littleEndian

    BytesAvailable = 0

    BytesAvailableFcn =

    BytesAvailableFcnCount = 48

    BytesAvailableFcnMode = terminator

    BytesToOutput = 0

    ErrorFcn =

    InputBufferSize = 512

    Name = Serial-COM1

    ObjectVisibility = on

    OutputBufferSize = 512

    OutputEmptyFcn =

    RecordDetail = compact

    RecordMode = overwrite

    RecordName = record.txt

    RecordStatus = off

    Status = closed

    Tag =

    Timeout = 10

    TimerFcn =

    TimerPeriod = 1

    TransferStatus = idle

    Type = serial

    UserData = []
```

```
    ValuesReceived = 0

    ValuesSent = 0


    SERIAL specific properties:

    BaudRate = 9600

    BreakInterruptFcn =

    DataBits = 8

    DataTerminalReady = on

    FlowControl = none

    Parity = none

    PinStatus = [1x1 struct]

    PinStatusFcn =

    Port = COM1

    ReadAsyncMode = continuous

    RequestToSend = on

    StopBits = 1

    Terminator = LF
```

Các bạn thấy là có rất nhiều tham số phải không? chúng ta ở đây quan tâm đến tham số: **BytesAvailableFcn** tham số này chưa thết lập. Tham số này chính là hàm callback mà nó sẽ gọi khi có byte nhận được ở bộ đệm nhận.Vậy chúng ta viết hàm này chính là viết hàm OnComm đáp ứng sự kiện ReceiveEvent như trong MSCOMM của MS vậy.

Thiết lập này phải thực hiện trước khi mở cổng để giao tiếp, nên chúng ta sẽ viết hàm callback trước. Bạn viết 1 m-file với tên Serial_Callback.m như sau:

Code:

```
function Serial_Callback(obj,event)

    ind = fscanf(obj)
```

Cú pháp của hàm callback như trên với obj là đối tượng kiểu Serial như trên. Hàm có tác dụng đọc dữ liệu và hiển thị luôn kết quả lên command window.

Chúng ta đưa tham số tên hàm vào cho đối tượng s của ta:

Code:

```
>> s.BytesAvailableFcn = @Serial_Callback;
```

Tiếp theo chúng ta bắt đầu giao tiếp:
Code:

```
>>fopen(s);

>>fprintf(s,'chao cac ban');
```

Sau đó các bạn xem kết quả thế nào, sau đó thử truyền các kí tự khác xem bằng lệnh fprintf(s,...), hoặc thử với vi xử lý xem cho nó truyền lên các bạn sẽ thấy rất hay.

Bạn không giao tiếp nữa thì đóng cổng lại:
Code:

```
>>fclose(s);
```

Mình viết tiếp sau còn giờ đi ngủ đã.

Chương trình giao tiếp trên PC với RS232 dùng Matlab .

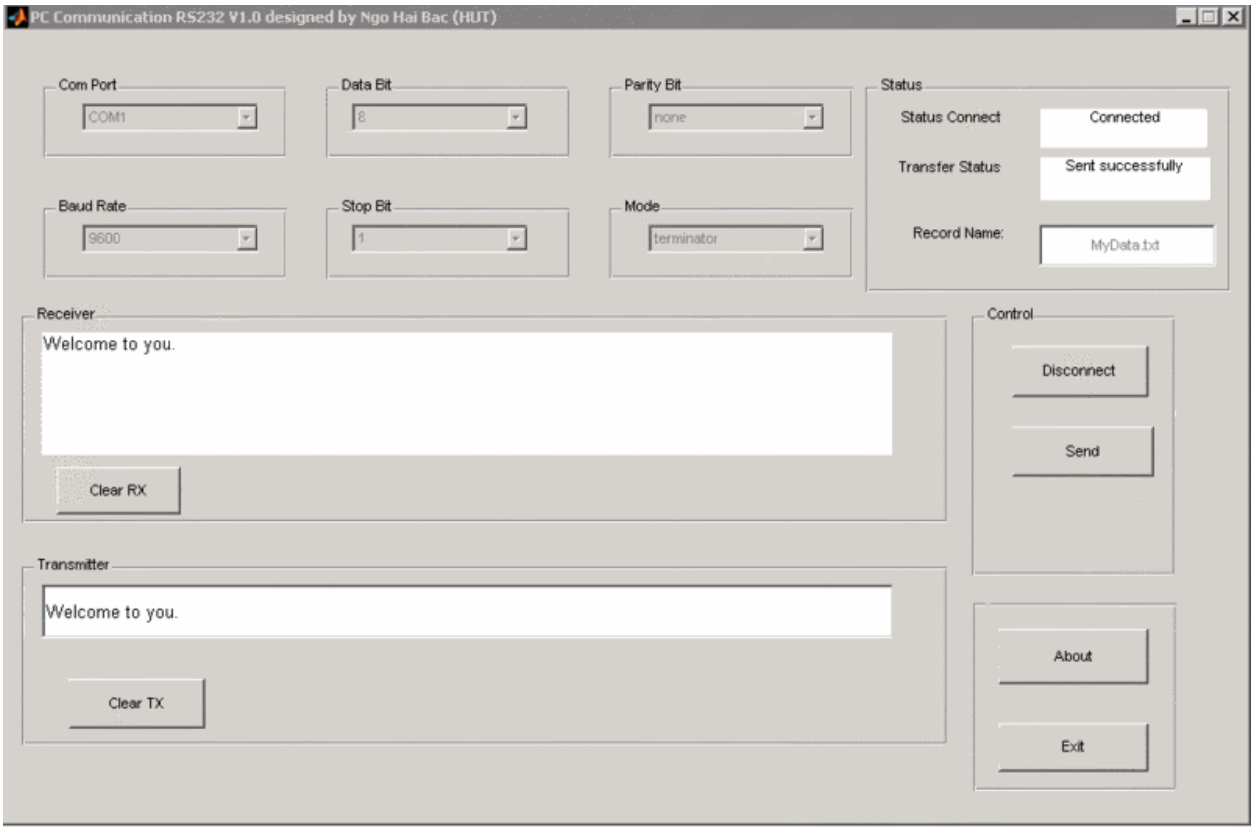
Cảm ơn bạn đã động viên.

Bài viết sau mình viết tại picvietnam.com ở luồng sau:
http://picvietnam.com/forum/showthre...=6568#post6568

Chào tất cả mọi người,

Hôm qua vừa thi xong được một môn, làm bài cũng gọi là tạm tạm không biết được mấy điểm . Sau nửa ngày hăng say lập trình , mình xin gửi mọi người một chương trình giao tiếp RS232 bằng Matlab.

Giao diện của nó như sau:



Cách dùng:

1/ Chọn tham số cho Rs232 rùi ấn nút Connect để bắt đầu kết nối với RS232 nhé.

2/ Nhập dữ liệu vào ô TX rùi nhấn nút Send để gửi dữ liệu.

3/ Để thay đổi tham số (tốc độ, ..) cho RS232 thì phải nhấn Disconnect trước rùi chỉnh tham số nhé. Sau đó quay lại bước 1.

Các bạn chú ý, đây là chương trình mình viết dưới dạng mở, tức là các bạn có thể thêm code vào các hàm để phục vụ mục đích của mình. Đó chính là các hàm ngắt nhận, ngắt gửi,...

BytesAvailableFcnCount là số byte nhận được trong bộ đệm nhận trước khi xảy ra ngắt nhận.

Các hàm đó là:

- function ByteAvailable_Callback(obj, event)
- function OutputEmpty_Callback(obj, event)
- function Error_Callback (obj, event)
- function PinStatus_Callback(obj, event)
- function Timer_Callback (obj,event)
- function BreakInterrupt_Callback(obj, event)

Chương trình này dùng GUI nên để biết thêm chi tiết về cách lập trình và chạy chương trình mời các bạn vào luồng **Lập trình GUI trên Matlab** ở www.dientuvietnam.net nhé:

http://dientuvietnam.net/forums/showthread.php?t=594

Trong chương trình, bạn có thể dùng chức năng About để xem hướng dẫn (User Manual), mình mới chỉ viết bằng tiếng Anh thôi.

Chương trình mình đã test khi nối tắt 2 chân 2 và 3 của RS232 lại với nhau. Còn với Vi điều khiển, .. mong các bạn test nhé.

Chào các bạn.

Tất nhiên là phải có Timeout rùi, trong ứng dụng của mình có tham số thiết lập timeout cho việc truyền và nhận dữ liệu đó. Theo mặc định nó = 10s.

Khi có Timeout nó sẽ gọi hàm ngắt sau:
Code:

```
function Timer_Callback (obj,event)
```

Bạn thấy rằng hàm ngắt chỉ có tham số là obj (đối tượng Serial) và event (sự kiện) thì làm sao mà tương tác với GUI được??

Để làm được điều này thì bạn phải thông qua một file mà mình đã thiết lập data.mat. Do đó trong hàm ngắt cần có 2 lệnh:

Code:

```
load data %Lấy các biến đã lưu trong data.mat

% Làm gì thì làm ở đây, hiển thị thì hiển thị :D.

save data  % Ghi dữ liệu mới của các biến thay đổi vào data.mat
```

Các bạn chú ý nhé nếu không là không tương tác được với GUI đâu. Ngoài ra bạn có thể lập trình thêm cho nó có chức năng vẽ đồ thị thời gian thực chẳng hạn, dữ liệu nhận về sau khi tính toán thì hiển thị luôn lên đồ thị.

Máy mình chậm nên chắc chẳng có thời gian thực gì cả .Các bạn có câu hỏi gì thì xin cứ trình bày mình sẽ trả lời. Còn giờ đi học thư viện đây . Chăm không??

Chúc các bạn thành công.

Chào các bạn.

Đây là một các nữa để chia sẻ dữ liệu giữa các hàm Callback.

Để lưu dữ liệu nằm trong biến X chúng ta set một trường trong cấu trúc **handles** = X và sau đó lưu trữ cấu trúc đó bằng hàm **guidata**:

Code:

```
handles.current_data = X;
```

```
guidata(hObject,handles)
```

Từ đó bạn có thể lấy dữ liệu này từ trong các hàm Callback khác bằng lệnh:
Code:

```
X = handles.current_data;
```

Chúc các bạn thành công.

RS232 Communication V2.0

Chào các bạn.

Mình đã sửa lại một tí để fix một số lỗi trong quá trình trao đổi dữ liệu giữa các hàm Callback bằng cách sử dụng cấu trúc **handles**, trao đổi giữa hàm ngắt của Serial Object bằng biến toàn cục global **hand**.

Bây giờ các hàm sẽ không phải trao đổi dữ liệu qua **data.mat** nữa, tránh được lỗi khi **save** và **load** nhiều thì file này bị hỏng.

Mình không upload được file lên site này, các bạn sang PVN down nhé:
http://www.picvietnam.com/forum//showthread.php?t=752

Chúc các bạn thành công.

Lập trình GUI trong Matlab

20-10-2005, 17:47

Các bác ạ. Em xin mạn phép nói một chút về GUI sau một số chương trình em làm được.

Một bạn điện tử đưa cho em chuyển giao diện từ Visual Basic 6.0 sang GUI(Graphic User Interface) trong Matlab. Em thấy được một số mặt như sau so với VB:

- Do Matlab là công cụ tích hợp mạnh các công cụ tính toán nên trong Gui cũng được kế thừa rất là hay như tính toán với ma trận, vecto,.. Do đó dùng GUI ta không phải viết hàm con yêu cầu tính toán nhiều như tích phân, vi phân, tính toán với ma trận(với VB thì chịu thua, viết mã dài chết).
Thứ nhất là thiết kế trực tiếp trên GUI(gõ lệnh gui trong Matlab) ta có thể nhúng các điều khiển ActiveX, biểu thị Simulink, các hệ trục,..Dùng mô phỏng như thật ấy, các bác có thể xem mấy cái Demo(nhớ cài JRE: Java Runtime
Evironment nhé down load tại java.sun.com) :

- Việc thiết kế các khung gắp thả như trong VB, lập trình cho các đối tượng cũng tương đối đơn giản với việc đặt tên cho các đối tượng trong thuộc tính Tag, tên hiển thị trên biểu mẫu với thuộc tính String, ngoài ra còn có nhiều thuộc tính cũng giống như VB như Position, Visiable,..Khi chúng ta tạo một đối tượng, thì sẽ có một hàm Callback được thêm tự động, ví dụ với đối tượng là Edit1 thì tương ứng sẽ có một hàm là Edit1_Callback được tạo ra đáp ứng sự kiện của đối tượng. Với Callback của đối tượng đó ta thao tác với đối tượng thông qua hObject và lấy thuộc tính dùng get, áp đặt thuộc tính thì dùng set:

Ví dụ:

Code:

```
  f=get(hObject,'String');% Lay sau nhap vao Edit1

   f=str2double(f);% chuyen sau thanh kieu Double

   set(hObject,'String','');
```

Còn nếu muốn lấy thuộc tính của đối tượng khác thì dùng handles.nhan
Ví dụ nếu muốn lấy thuộc tính của Edit2 trong hàm Edit1_callback thì dùng:
Code:

```
f2=get(handles.Edit2,'String');

        set(handles.Edit2,'String',' ');
```

Để lấy kết quả từ một hàm nào đó bạn có thể dùng lệnh Save để lưu vào Mfile, hoặc dùng biến toàn cục nhưng nên dùng lệnh Save, sau đó muốn láy kết quả thì dùng Load. Cách này rất tốt cho cả các form khác nhau và các ứng dụng như giữa Simulink và GUI.

Thứ nữa là nếu muốn lập trình cho menu thì tôi thấy là khi ta tạo lên một cái menu như là để mở một cái Form khác thì nếu dùng lệnh form với tên form là tên của form mà bạn cần mở ra. Nếu muốn cho form này luôn hiển thị trên các form khác thì dùng lệnh uiwait(form). Rất đơn giản phải không?Các bác có thể chạy được mấy cái đó trong môi trường Gui thôi(mở Gui và Open Browes đến file cần chạy). Hoặc là dùng lệnh trong command như trên.
Thứ hai, thiết kế bằng tay: bạn chỉ cần phải thiết kế giao diện ban đầu hơi vất, sau đó là có thể cho thuộc tính Callback gọi một số hàm dưới dạng Mfile riêng rẽ(tức là viết riêng ra). Do đó khi minh gọi dùng cho Callback của các menu thì sẽ ổn thôi. Các bác khi thiết kế thì dùng cái thuộc tính mở Gui ra mà xem, còn thiết kế thì hoàn toàn trong Mfile: ví dụ như: edi1=uicontrol('style','radiobutton',..); Viết mã này cũng hơi dài yêu cầu phải kiên trì. Rất dễ báo lỗi.

Trên đây là một số nhận xét của tôi không biết có ổn không nhưng chắc phần nào giải quyết cho các bác làm quen với Gui.

Mình đã sửa lại bài viết cho đúng với thế mạnh của nó. Nếu viết sai thì chết. Thực ra chẳng khó nếu bạn tìm được giải pháp cho đúng. Nó có thể toàn diện ngang với VB có khi còn mạnh hơn về khả năng tính toán. Còn lập trình cấp thấp thì thua là cái chắc.

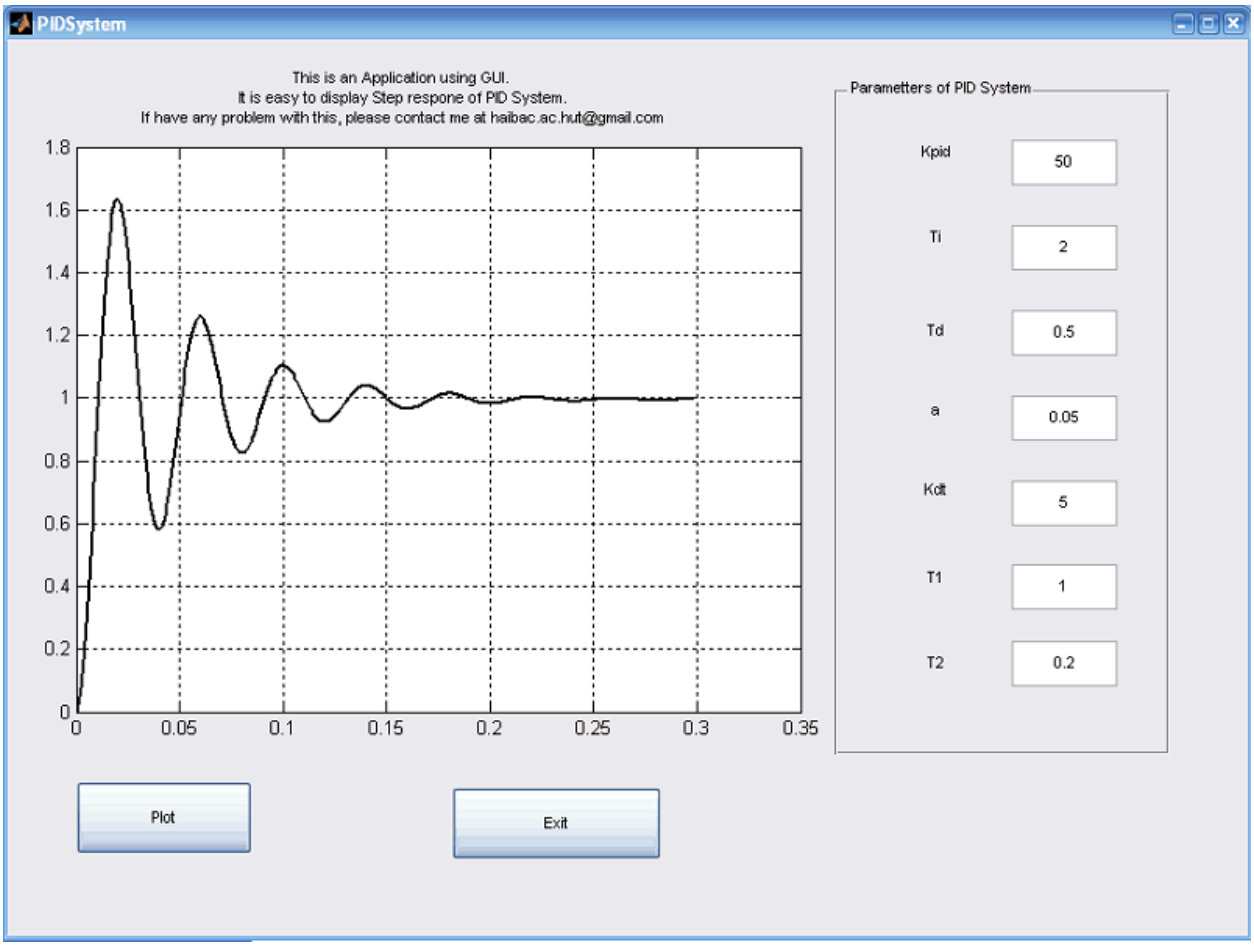
PIDSystem

Mình thấy các bạn mới học về Lý thuyết điều khiển tự động mà muốn hiểu rõ sự ảnh hưởng của các hệ số Kpid,Ti,Td với hệ thống chắc hẳn nếu dùng bằng Matlab khi thiết kế sẽ rất mất công thay thế các tham số.

Mình chọn giải pháp là dùng GUI trong Matlab để thiết kế. Khi ta thay đổi các tham số của bộ điều khiển thì ấn vào Plot thì sẽ cho ngay đặc tính của hàm quá độ để biết xem hệ có đáp ứng chỉ tiêu chất lượng như có độ quá điều chỉnh phù hợp hay không? Mình nghĩ là cái này dùng rất là trực quan. các bạn down cái này giải nén các file vào thư mục /work trong thư mục cài đặt. Vào Matlab gõ lệnh PIDSystem thì bạn sẽ xem được chương trình. Hoặc mở GUI ra sau đó Open đến filel PIDSystem.fig.

Lưu ý là tôi làm trên Matlab 7.0 nhưng mà trên bản 6.5 của bạn tôi thì không thể chạy được. Các bác cứ chạy trên bản 7 nhé.

Attached Files



Last edited by ngohaibac; 26-05-2006, 07:59.

à tôi quên là nếu mà bạn lập trình GUI thì bạn có thể chạy nó bằng của sổ command của Matlab với tên giống như tên của m file đó.

Với lại nếu mà lập trình menu thì bạn có thể dùng cái đó để lập trình tức là gọi các form khác ra.

Ảnh minh họa của bài trên tôi không up lên được mong các bác thông cảm

OK thấy các bác hưởng ứng em vui lắm. Viết bài mãi mà chẳng thấy có ai đáp lại cả(sướng ca vô loài) tưởng các bác không quan tâm đến cái này. Tôi đã giúp cho một bạn ở K46 (bạn Hiền điện tử 10.. nếu vào thì cùng trao đổi ở đây nhé.) làm đề tài về mạng viễn thông tính tổn hao đường truyền của mạng. Mình đã thêm được một số kinh nghiệm về lập trình gui. Thực ra lập trình cái này cực kì đơn giản chỉ cần biết một vài cái về nó là xong.

Tất nhiên về vấn đề bản quyền. Chương trình này phải chạy trên môi trường Matlab không dịch được ra file .exe cho nên không thể bán sản phẩm của bạn mà không có bản quyền của Matlab được(đau quá!) . Với lại nếu bạn cần phải tính toán rất nhiều liên quan đến ma trận, đồ thị thì mới nên dùng cái này (như ngành của mình chẳng hạn). Còn không thì bạn có thể dùng bất kì phần mềm nào như VB, Visual C++, .. để viết ngon lành có điều những hàm tính toán đơn giản trong Matlab chuyển qua đây cũng tương đối phức tạp, dài dòng.

Matlab là chương trình mô phỏng cực kì mạnh(Simulink,..) , có thể viết với mã C nhưng không thể giao tiếp với hệ thống được(không lập trình qua cổng Com được). Hay nói cách khác Matlab không phải là ngôn ngữ lập trình mà là một phần mềm ứng dụng cho phép ta dùng những từ đã định nghĩa sẵn trong Matlab để mô phỏng cái mà ta cần không thể lập trình được cấp thấp(low level).

Mình sẽ viết tourial về cái này cho anh em ta và đặc biệt về ứng dụng trong các ngành về điều khiển tự động

Lập trình GUI cho ứng dụng Simulink

Lần này tôi sẽ giới thiệu cho các bạn cách lập trình GUI cho ứng dụng Simulink: dùng GUI để thiết lập tham số cho các khối chức năng Function Block (FB), chạy mô phỏng hiển thị ra Scope.Lần tiếp theo tôi sẽ giới thiệu cách lấy dữ liệu đã mô phỏng sử dụng vào các mục đích khác như để ghi vào file hoặc là hiển thị ra một hệ trục tọa độ khác.

Còn đây là chi tiết về vấn đề hôm nay: Tôi dùng ví dụ điều chỉnh tốc độ của động cơ(có hàm truyền là bậc 1/bậc 2: là khâu dao động). Động cơ chính là đối tượng, còn ta sẽ bộ PID là bộ điều khiển để điều khiển tốc độ động cơ.

- Trước hết, muốn điều khiển được ứng dụng Simulink thì bạn phải tự thiết kế một modul .mdl trong Simulink bao gồm các khối chức năng, phải nối sẵn các khối đó với nhau(không cần điều chỉnh thông số cho các bộ đó). Lưu tên file vào trong thư mục Work với phần mở rộng là .mdl. Ở đây tôi chọn DCMotor. Sơ đồ các bạn có thể tham khảo ở dưới.

- Thứ hai, thiết lập trong GUI:

+ Tạo giao diện như hình dưới bằng cách kéo thả. Đặt tên cho các đối tượng: nhấp đúp chuột, hiện Properties chọn thuộc tính Tag để đặt tên(tương ứng Caption trong VB). Các bạn nên đặt tên sao cho dễ nhớ(như nút bấm thì đặt có Button ở cuối,...). Có các edittext,statictext, plushbutton : ExitButton,SimulateButton, editP, editI,editD,..

+ Viết chương trình: Các bạn lưu ý ở đây tôi dùng bộ PID rất là đơn giản đạng : P+ I/s+ D*s thôi. Các bạn có thể dùng bộ PID khác phức tạp hơn như bộ nhân,.. Khi đó các tham số P,I,D có thể thay bằng Kp,Ki,.. Bạn hoàn toàn có thể chuyển hóa các cái này với nhau bằng cách lấy thuộc tính String của nó sau đó chuyển sang dạng Double để tính toán sau đó lại truyền tham số vào các khối PID,.. dùng str2double(string) chuyển str -> double, num2str(num) chuyển num->str. Thế là vấn đề về tham số là không có vấn đề gì.

++ Với các nút bấm, edittext,.. bạn có các hàm Callback như Callback,CreateFcn,.. dùng CreateFcn để thiết lập tham số khi khởi tạo.

Mọi thứ râu ria đã xong bây giờ tôi sẽ giới thiệu cách thiết lập tham số cho bộ điều khiển. Để biết được tham số của các FB bạn vào Help theo chỉ dẫn của tôi ở dưới hình để biết chi tiết tham số. DCMotor.mdl có các khối tên là :
Sum,Scope,Transfer Fcn,PID Controller,Step.
Ở đây tôi dùng một hàm là model_open(handles) để thiết lập tham số và lấy thông tin từ các editP,editI,editD chuyển vào bộ PID, các khối khác cũng tương tự nếu cần;

Code:

```
% begin code

function model_open(handles)

     open_system('DCMotor'); % mở bộ DCMotor phải đúng tên của file

                                    %nếu không thì chẳng làm được gì đâu

     set_param('DCMotor/PID
Controller','P',get(handles.editP,'String'),'I',get(handles.editI,'String')..
.

         ,'D',get(handles.editD,'String'));

     set_param('DCMotor/Transfer Fcn','Numerator','[15]','Denominator','[1
0.5 1]');

     set_param('DCMotor/Sum','Inputs','|+-');

     set_param('DCMotor/Step','Time','0');


set_param('DCMotor/Scope','Open','on','Decimation','1','NumInputPorts','1');

    % mở scope để xem mô phỏng.
```

****
như các bạn thấy rất là đơn giản đúng không. Các khối chức năng như là một phần nhỏ của DCMotor do đó bạn dùng DCMotor/Sum để lấy tên của bộ Sum, tương tụ với các bộ khác, tham số thì tham khảo trong Help . Với khối PID lấy từ các edit.
--->thế là xong phần mở bộ điều khiển.

Phần tiếp theo là mô phỏng: khi ấn nút Simulate thì quá trình mô phỏng sẽ xảy ra. Thực hiện trong SimulateButton_Callback. có mấy cách để thực hiện có thể dùng hàm sim('DCMotor'), hoặc là

set_param('DCMotor', 'SimulationCommand', 'start') để thiết lập tham số bắt đầu 'start', 'stop'. Từ đó bạn có thể cho thêm nút dừng mô phỏng trong GUI.
:
Code:

```
function SimulateButton_Callback(hObject, eventdata, handles)

% hObject    handle to SimulateButton (see GCBO)

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)

    %Insert code

    model_open(handles);

    set_param('DCMotor', 'SimulationCommand', 'start')

   % --- Executes during object creation, after setting all properties.
```

Còn với nút Exit đơn giản chỉ cần dùng hàm Close(' tên figure');
Code:

```
close(ControlDCMotor);
```

Xong! Các bạn thấy thế nào/.
Chúc thành công.

Ở dưới tôi upload cả mấy file minh họa và file chương trình . Các bác cho giải nén cho vào thư mục Work của Matlab. Gõ lệnh ControlDCMotor trong CommandWindow

Matlab mạnh hơn VB rất nhiều về mô phỏng lập trình GUI chỉ là phần nhỏ ứng dụng các hàm tính toán đã được lập sẵn trong thư viện của nó. Nhưng bạn chẳng thể nào dịch file đó ra .exe mà chạy riêng đâu. không cần save vào Work đâu. Bạn mở Guide ra Open đến file đó là được. Rồi Run nó sẽ hỏi có muốn chuyển cái đó thành thư mục Workspace không. Ok là xong. Chạy ngon.

Vd này tạo GUI cho mô hình Simulink,nhưng khi mình chạy thử thì có khi báo lỗi,có khi lại chay được,
Các bạn thấy sao?

Bạn phải xem xét lỗi của nó trong cửa sổ Command của Matlab ấy mà sửa lỗi. Ví dụ có lỗi chỉ xẩy ra khi bạn ấn vào nút nào đó thôi. Còn chưa ấn thì vô tư..

ControlDCMotor V2.0

Các bạn thân mến hôm nay tôi sẽ giới thiệu cho các bạn cách hiển thị dữ liệu ra bằng Axes trên giao diện.

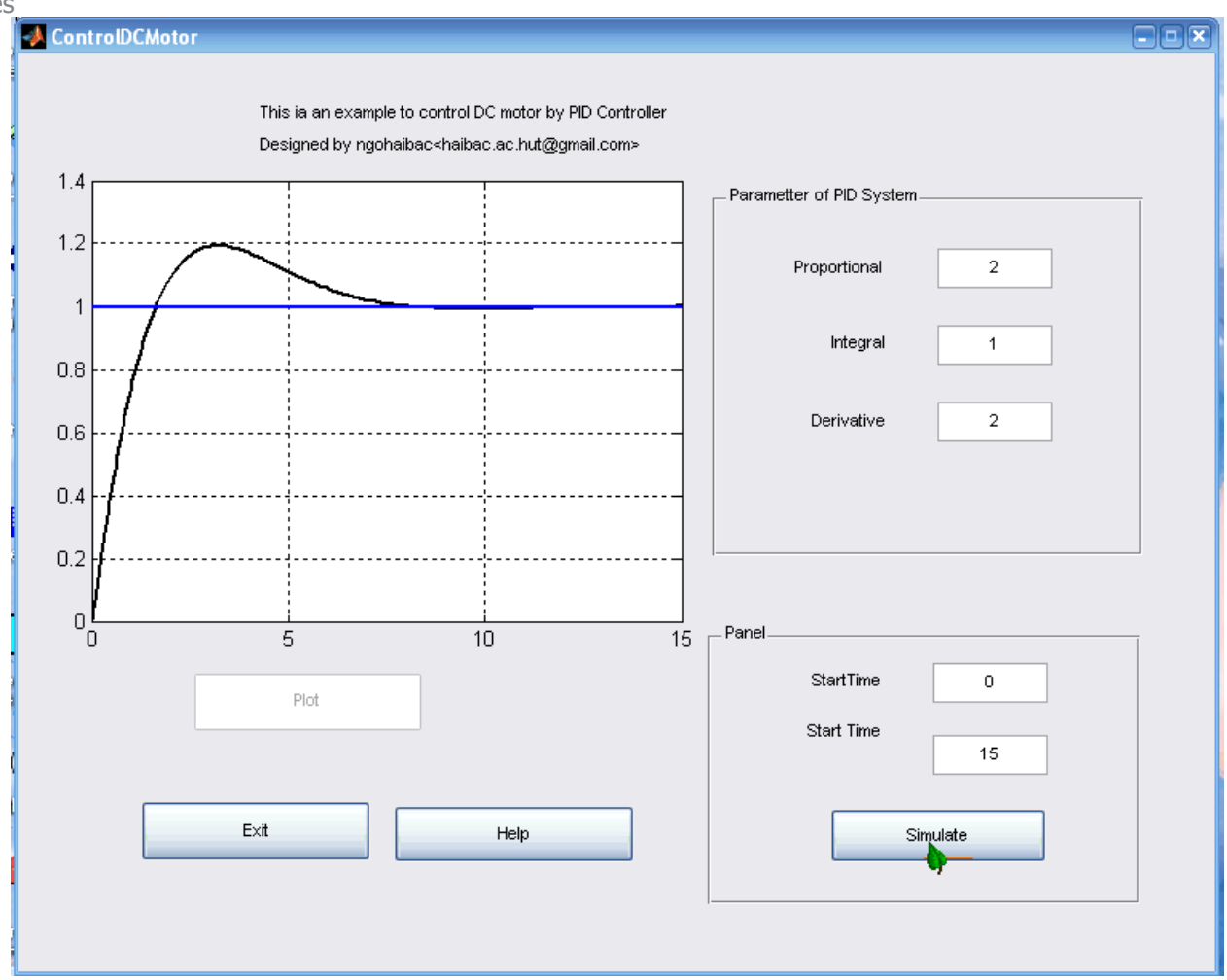
Hình minh hoạ bên dưới.

Giao diện này cho phép bạn có thể ấn vào Simulate để bắt đầu mô phỏng, ấn vào Plot để hiển thị ra đồ thị. Ấn vào Help để xem hướng dẫn. Ấn vào Exit để thoát. Nó sẽ hỏi bạn xem có muốn thoát không. Bạn đọc kĩ code để xem nhé.
Để làm được điều đó thì mình đã cho một Block: tofile vào khâu ra. Khi đó dữ liệu sẽ được ghi vào file do

mình thiết lập trong model_open, sau đó dùng file này để lấy dữ liệu hiển thị lên đồ thị(dùng Plot): dùng command: load tênfile tếnbiến. Thế là xong.

Hi vọng là thỏa mãn sự đòi hỏi của các bạn về điều khiển chạy mô phỏng Simulink với các tham số tự cho, và đưa ra đồ thị.(lưu ý: các bạn nên đánh đúng tên file để chạy ứng dụng này: giải nén hết vào thư mục Work, đánh lệnh ControlDCMotor để chạy)

Ngoài ra các bạn có thể cải tiến thêm ứng dụng này cho phù hợp với bản thân mình.
Chúc thành công
Attached Files

- 

Nhập thông số cho Transfer Fcn

Cho em hỏi bác Hải Bắc tí nha
Ở cái ControlDCmotor.fig của bác,bây giờ em muốn thêm các nút editbutton để thay đổi thông số của hàm truyền động cơ thì phải thêm những hàm gì trong file soạn thảo ControlDCmotor.m?

Bạn sẽ phải thêm các editbox đó vào trong giao diện của mình rồi đặt thuộc tính tag cho nó. Sau đó lấy dữ liệu từ nó bằng hàm str2double. Bạn sửa trong model_open(handles). Vì hàm này được gọi trong SimulateButton_Callback(..). Khi ấn vào đó thì sẽ gọi hàm này lấy tham số từ các editbox để mô phỏng. Do vậy bạn cần thêm trong hàm model_open như sau: ví dụ bạn có các editbox có thuộc tính tag là a0:

ứng với tử số, b0,b1,b2 ứng với mẫu số. Khi đó cần lấy giá trị của các cái này sau đó cho tham số vào block tên là Transfer Fcn:
Xóa dòng :
Code:

```
set_param('DCMotor/Transfer Fcn','Numerator','[15]','Denominator','[1 0.5
1]');
```

Thay bằng dòng sau:

Code:

```
a0= get(handles.a0,'String');

 b0=get(handles.b0,'String'); % tương tự với các cái khác b1,b2

set_param('DCMotor/Transfer Fcn','Numerator','[15]','Denominator',['[',b2,'
',b1,' ',b0,']']);
```

// thế là xong
Bạn xem lại cách ghép sâu. Tham số cho Nummerator là kiểu string dạng [các số] tức là có kiểu như vector nên dùng [] để liên kết dữ liệu. Nhớ trong " phải có ít nhất một khoảng trắng.
Còn nếu em muốn các đáp ứng ngõ ra của hệ thống được giữ lại trên ControlDCmotor.fig (ở cái chỗ figure mỗi khi nhấn plot thì có thể vẫn giữ lại hình ảnh đồ thị các ngõ ra của những lần trứơc,có nghĩa là khi thay đổi thông số PID thì mỗi khi plot vẫn còn lưu lại các đồ thị trên cùng 1 figure để em dễ dảng so sánh hơn) thì em phải làm sao? Có thể dùng lệnh Hold on được ko?
Mong bác giải quyết giúp em nha!!!!!!!!!!!!

Đơn giản thôi. Bạn xem trong PlotButton_Callback bạn sẽ thấy là mình có dùng hold on(giữ lại đồ thị đã vẽ lần trước), hold off (bỏ đi đồ thị đã vẽ). Do đó mình dùng hold on sau khi vẽ xong plot(x,y1) để hiển thị tiếp plot(x,y2). Trước đó mình dùng hold off để xóa đi đồ thị trước. Nên bạn muốn giữ lại tất cả thì bỏ dòng lệnh:

Code:

```
 hold off;
```

ở đầu đi.

Chúc thành công

Cảm ơn các bạn đã đọc bài viết của mình. Mình viết những bài viết này trên cơ sở tham khảo bộ Help của Matlab. Với bộ help này quả thật là chi tiết, có cả ví dụ, cách dùng như là MSDN của Microsoft vậy. Với một mục đích cụ thể cho công việc cần làm, mình nghĩ là các bạn sẽ biết cách tìm hiểu các điều mình cần trong bộ Help này. Không phải là đọc tất cả mà chỉ đọc cái mình cần, nếu không hiểu vấn đề nào thuộc những phần khác thì bạn lại quay sang đọc cái đó để hiểu rõ, rồi lại trở lại vấn đề đang đọc. Với cách đọc tài liệu như thế mình nghĩ là sẽ nhanh đạt mục đích hơn. Mình viết chương trình điều khiển động cơ trên cơ sở câu hỏi của bạn SonIC gửi đến email của mình. Các bạn có câu hỏi, mục đích gì thì cứ đưa ra diễn đàn này để chúng ta cùng giải quyết.

Mình sẽ tiếp tục giới thiệu cho các bạn một số hướng dẫn cụ thể để lập trình cho GUI liên kết với các ứng dụng khác, điển hình là bộ mô phỏng hệ thống Simulink.

Cảm ơn các bạn. Cảm ơn các bạn đã đọc bài viết của mình. Mình viết những bài viết này trên cơ sở tham khảo bộ Help của Matlab. Với bộ help này quả thật là chi tiết, có cả ví dụ, cách dùng như là MSDN của Microsoft vậy. Với một mục đích cụ thể cho công việc cần làm, mình nghĩ là các bạn sẽ biết cách tìm hiểu các điều mình cần trong bộ Help này. Không phải là đọc tất cả mà chỉ đọc cái mình cần, nếu không hiểu vấn đề nào thuộc những phần khác thì bạn lại quay sang đọc cái đó để hiểu rõ, rồi lại trở lại vấn đề đang đọc. Với cách đọc tài liệu như thế mình nghĩ là sẽ nhanh đạt mục đích hơn. Mình viết chương trình điều khiển động cơ trên cơ sở câu hỏi của bạn SonIC gửi đến email của mình. Các bạn có câu hỏi, mục đích gì thì cứ đưa ra diễn đàn này để chúng ta cùng giải quyết.

Mình sẽ tiếp tục giới thiệu cho các bạn một số hướng dẫn cụ thể để lập trình cho GUI liên kết với các ứng dụng khác, điển hình là bộ mô phỏng hệ thống Simulink.

Cảm ơn các bạn.

Động cơ một chiều

- Các bạn lưu ý: khi làm lại giao diện mà lại copy lại y nguyên mã lại không chạy sẽ báo lỗi là vì với mỗi điều khiển thì sẽ có một tên riêng trong thuộc tính tag của nó(ứng với thuộc tính Name trong VB) để các bạn làm việc với điều khiển. Nên các bạn khi download chương trình về cần đọc kĩ mã nhé. Cần phải biết hàm đó viết cho điều khiển nào, lấy thông tin từ điều khiển gì.

-Thứ hai là các bạn phải đặt tên .mdl khác với tên của cái gui tức là đuôi .m đó. Vì cả 2 file này đều có thể chạy được khi bạn gõ tên trong command của workspace nhưng file đuôi .mdl(simulink) được ưu tiên hơn nên đó là lí do tại sao mà khi đặt tên giống nhau mà lại chỉ có thể chạy được file .mdl thôi.

- Các bạn không thể chạy ứng dụng bằng cách di chuột đến file .fig trong curent directory được vì nó chạy đây đơn thuần là chạy một file hình ảnh thôi. Nó sẽ không thể liên kết được với file .m chứa mã chương trình, các hàm callback đo đó sẽ báo lỗi ngay. Các bạn có nhiều cách làm: gõ lệnh trong command trùng với tên của file .m đó, mở file .m đó rồi ấn phím tắt F5 hoặc vào Debug-> Run, ấn vào biểu tượng trên thành toolbar hình mũi tên trỏ xuống đó.

Mình xin tiếp tục hướng dẫn các bạn làm thêm một số phần trong giao diện điều khiển mô phỏng Simulink bằng GUI.

Tôi sẽ thêm vào đó popupmenu để các bạn hiển thị màu của đường đặc tính và checkbox để khi cần thì giữ lại hình để so sánh, khi không thì thôi. Thêm cả thêm thuộc tính cho các transfer fcn.

Để có thể làm việc đối với popupmenu thì bạn nên thêm sẵn cho nó trong thuộc tính String của nó ngay khi thiết kế: đánh thành nhiều dòng bao gồm:
Blue, Green,Red,...

Sau đó khi người dùng chọn một trong các màu đó thì sẽ có giá trị lấy từ thuộc tính Value của điều khiển này. Các bạn dùng thuộc tính này để quết định xem đó là màu gì bằng cách dùng Switch trong PlotButton_callback:
Code:

```
ColorIndex = get(handles.popupmenu1,'Value');

    switch ColorIndex

        case 1, a = 'b';
```

```
        case 2, a = 'g';

        case 3, a = 'r';

        case 4, a= 'c';

        case 5, a= 'm';

        case 6, a= 'y';

        case 7, a= 'k';

        otherwise,

            a= 'k';

    end
```

Nút checkbox có thuộc tính value =1 khi nó được check , =0 khi không check.
Do đó lấy giá trị này:
Code:

```
if get(handles.checkHold,'Value') == 0

        hold off;

  end
```

không cần else vì ở dưới trong khi vẽ hai hình đã có hàm hold on;
Không set nó thành off thì nó vẫn giữ nguyên.
----- Chúc các bác thành công.

ControlDCMotor V3.0

Chào các bạn, nay tôi xin giới thiệu cho các bạn một cách tạo menu và sử dụng một temp để làm trung gian để tạo dữ liệu cho đơn giản.

ở phiên bản trước chắc các bạn thấy rằng với cả 3 hàm Tranfer Fcn mà phải tạo tới 3 cái panel để điền thông số thì quả là rắc rối. Mình xin cải tiến thêm và dùng Temp là một popupmenu, thuộc tính String của nó làm nới chứa dữ liệu các tham số của hàm truyền của các hàm truyền 1,2,3. Khi bạn ấn vào nút Edit thì sẽ nhập thay đổi dữ liệu cho nó và nút biến thành Apply, bạn ấn vào nút đó để cập nhật. Thuộc tính String trong popupmenu là kiểu cell(trường) do đó để cập nhật bạn lấy thuộc tính String sau đó thay đổi từng thành phần của biến này sau đó cập nhật trở lại. Còn nếu muốn lấy dữ liệu cũng tương tự mỗi thành phần trong cell này có kiểu String, ta dùng dấu {} để xâm nhập vào từng thành phần của nó:

VD:

Code:

```
value= get(handles.Temp,'String');
```

```
    a = value{1};
```

Thuộc tính String của Temp tôi đã làm sẵn trong thiết kế. Và bị thay đổi khi ấn vào nút Apply. Do đó các điều khiển có thể dùng điều khiển Temp này.Điều khiển Temp cho thuộc tính Visible là Off để không nhìn thấy lúc thi hành mà chỉ thấy lúc thiết kế. Dữ liệu trong Temp thay đổi được trong chương trình và việc khởi tạo ban đầu có thể bằng tay hay trong hàm CreatFcn của điều khiển này.

Còn tạo menu thì các bạn vào Tools-> menu editors. Sẽ có dialog mới mở ra. Các bạn kích vào nút New menu để tạo ra menu mới, sau đó thay đổi thuộc tính Label, Tag của nó. Còn muốn tạo menu con thì bạn select vào menu muốn tạo menu con sau đó kích trái chuột vào biểu tượng <-> trên toolbar để tạo thêm menu con tương tự như trên. Còn muốn thấy được các hàm Callback của nó bạn chọn menu tương ứng kích vào View, nên để mặc định là tự động tạo hàm callback. Sau đó viết code cho các hàm callback đó. Quả thật là nếu mà viết lại mã của những callback giống của các điều khiển khác thì chỉ cần copy -> paste là xong nhưng tôi xin giới thiệu cho các bạn cách để gọi một hàm callback khác trong callback hiện tại:

Ví dụ: bạn muốn gọi hàm callback : function SimulateButton_Callback(hObject, eventdata, handles)

Thì chỉ cần thay hObject thành handles.SimulateButton là xong.

Còn tại sao mà ấn vào nút Edit lại thành Apply thì chỉ là thay đổi thuộc tính String trong callback của nó. Quan trọng là khi ấn vào nút Apply thì mới cập nhật dữ liệu. Tôi thông qua một biến Bool là value{10} trong String của Temp.
Các menu tôi mới chỉ viết code cho các menu Exit, About, Run.
Các bạn ấn vào popupmenu gần nút Edit để xem các thông số của các hàm truyền tương ứng.

Tôi sẽ hoàn thành việc phát triển cho ứng dụng này làm việc với các form khác nhau sau. Nói chung là cũng phức tạp hơn chưa các bạn. Có giống với VB chưa.

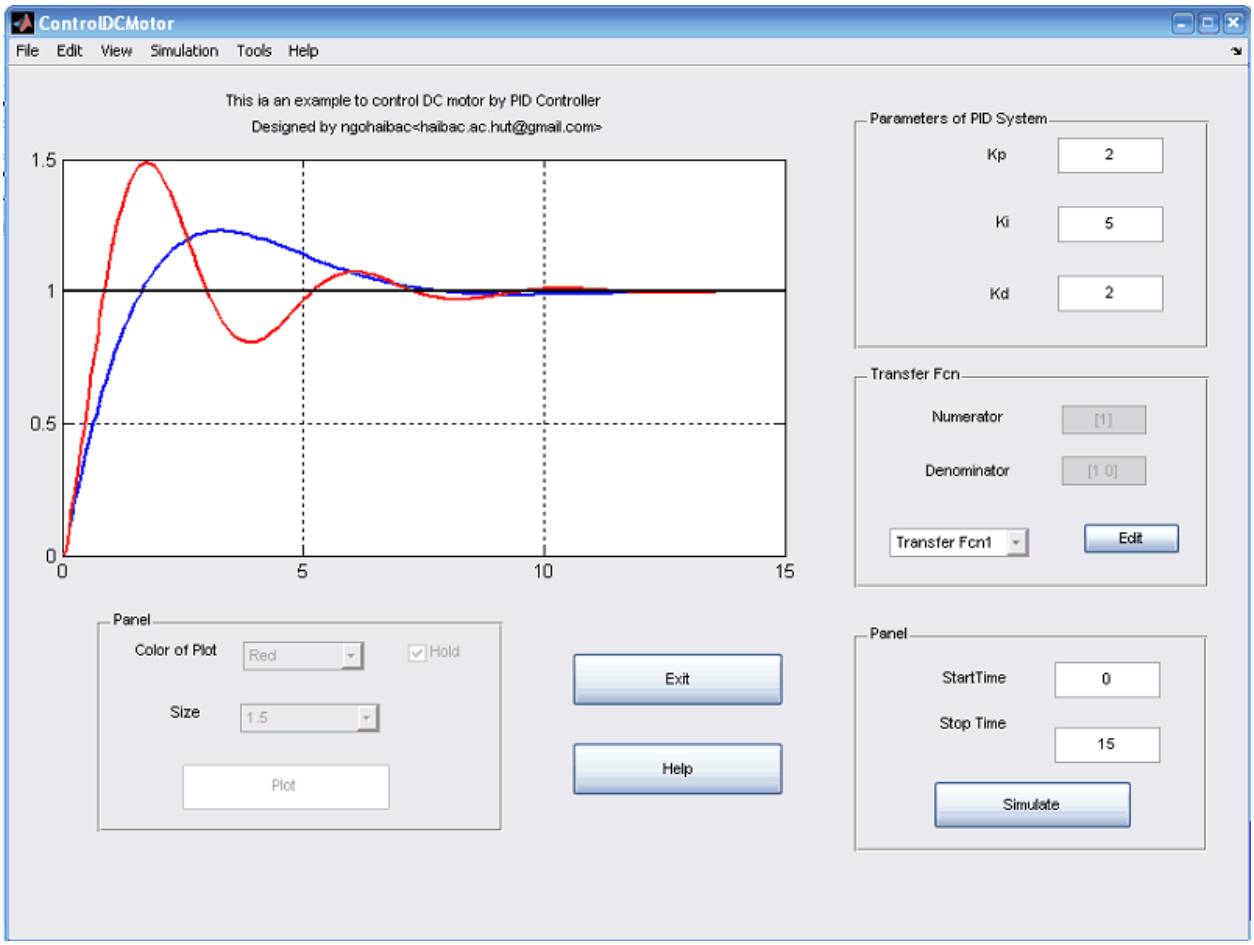
Chào các bạn. Chúc thành công.

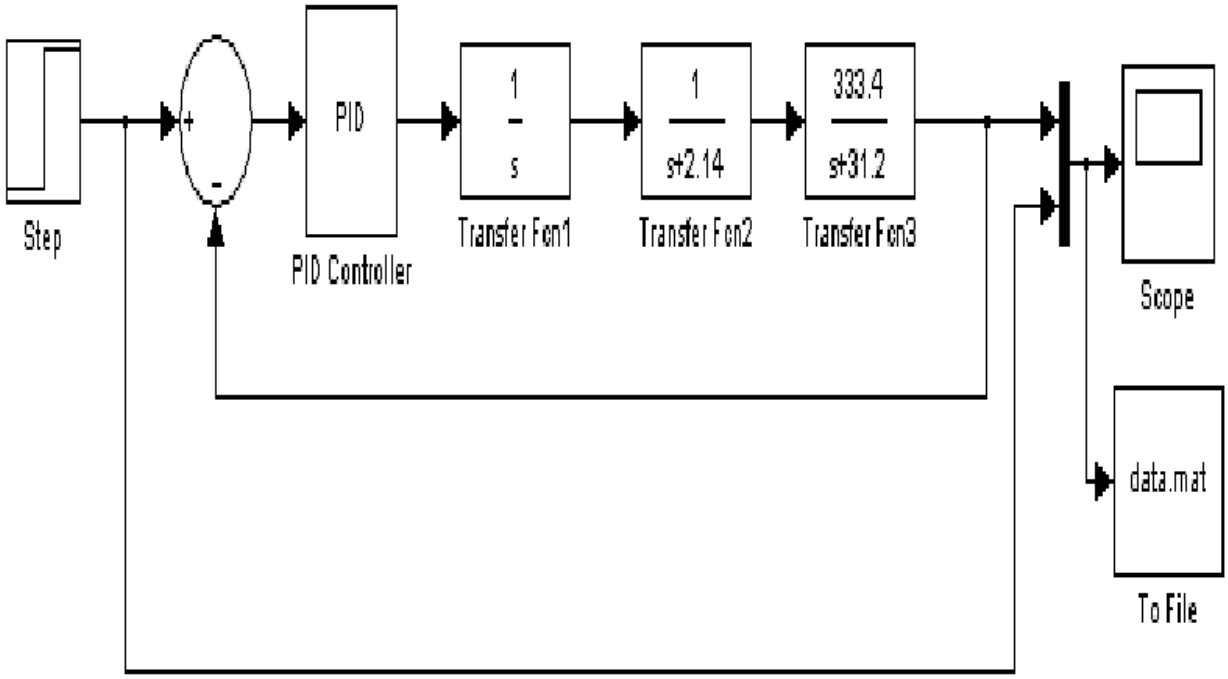
ControlDCMotor v3.1

Các bác thân.
Bận quá mới làm xong mấy cái giao diện để hiện lên khi ấn vào menu nên chưa up hoàn chỉnh được.
Nhưng cũng xin tối ưu cho chương trình trước về cách nhập hàm truyền đơn giản hơn rất nhiều.

Các bác chịu khó đọc lại mã nha. Dưới đây là hình minh họa và file đính kèm.

Chúc các bạn thành công.
Attached Files


ControlDCMotor
File Edit View Simulation Tools Help
This ia an example to control DC motor by PID Controller
Designed by ngohaibac<haibac.ac.hut@gmail.com>
1.5
1
0.5
0
0
5
10
15
Parameters of PID System
Kp
2
Ki
5
Kd
2
Transfer Fcn
Numerator
[1]
Denominator
[1 0]
Transfer Fcn1
Edit
Panel
Color of Plot
Red
Hold
Size
1.5
Plot
Exit
Help
Panel
StartTime
0
Stop Time
15
Simulate



Chào bạn.

Mình hiện giờ đang rất bận. Khi nào rảnh mình sẽ cố gắng viết Tutorial về lập trình M-file và neural toolbox cho các bạn. Đầu tiên các bạn cứ tham khảo trong Help của nó và site help online của Matlab : http://www.mathworks.com/ (tại đây có những đoạn code mẫu cực hay và nhưng ứng dụng và khả năng mà mình từng nghĩ là Matlab không có khả năng đó :d)

Chúc bạn thành công.

Bạn ngohaibac viết hay và chi tiết. Good job! Có thêm một chút bổ sung thế này:
- Chương trình viết bằng Matlab có thể dịch ra file EXE được, bằng công cụ gọi là Matlab Compiler (có sẵn trong bộ cài Matlab). Xem thêm về lệnh mc. Khi chạy chương trình đã dịch này thì không cần Matlab trong máy, nhưng phải có 1 cái kiểu như Matlab library cài trong máy (chắc cũng tầm hơn chục Mb)
- Nếu không thích dịch ra EXE thì có thể tạo ra đối tượng COM từ chương trình Matlab với công cụ Matlab COM Builder. Sau đó có thể dùng nó trong VB, Excel và các công cụ phát triển nhanh (RAD) khác
- Không phải lệnh nào, toolbox nào cũng có thể dịch được ra file EXE hay COM
- Có thể viết chương trình bằng C/C++ hay Fortran, ADA, và khi cần tính toán phức tạp thì gọi thư viện tính toán của Matlab
- Nếu lập trình bằng Java thì có thể từ Java gọi thư viện tính toán của Matlab, thậm chí control cả Matlab/Simulink. Công cụ để làm việc này tôi không nhớ tên chính xác, đại loại kiểu như Matlab Java Link, or something like that (search trên Google chắc sẽ ra ngay).

- Có 1 công cụ mà tôi nghĩ khá hay là Easy Java Simulation (http://fem.um.es/Ejs/Ejs_en/index.html). Công cụ này cho phép thiết kế các Visual Simulation trên cơ sở Java, có GUI khá bắt mắt. Một tính năng hay là nó cho phép link với Simulink, do đó mình có thể tạo GUI khá đẹp cho Simulink simulation của mình mà không mất công lập trình M-file. Hồi trước khi phát triển bộ thí nghiệm điều khiển quá trình cho sinh viên ĐKTĐ BK, tôi phải lập trình GUI bằng M-file cho Simulink rất vất vả.
- Cuối cùng, với những ai thích lập trình Matlab, có 1 blog rất hay là Loren on the Art of Matlab (http://blogs.mathworks.com/loren/?ref=fp)
--
Truong Nghiem

Chào anh HA.

Em đã đọc qua tài liệu của anh gửi và thư viện Help của Matlab 7.0 của em. Em nhận thấy là công cụ Matlab Complier thật là hay.

Vì thế sau khi em đọc tài liệu và hiểu được điều gì thì sẽ viết ngay lên cho anh và mọi người xem và chúng ta cùng nhau trao đổi về trình dịch này.

Đầu tiên nói lại về M-Complier đã (anh HA copy tiếng Anh vào có khi nhiều bạn không hiểu hết):

//==========
==========

Matlab complier có thể tạo ra các loại ứng dụng và thành phần sau:

- Ứng dụng độc lập: ứng dụng có thể chạy độc lập không yêu cầu trình Matlab được cài trên máy của người dùng.
- Các thư viện chia sẻ C, C++ (thư viện liên kết động DLL trên Microsoft Windows). Các thư viện này có thể dùng mà không cần Matlab được cài trong máy của người dùng thư viện này.
- Excel add-ins, yêu cầu phải có **MATLAB Builder for Excel**
- COM objects; yêu cầu phải có **MATLAB Builder for COM**

Matlab Complier hỗ trợ mọi hàm của Matlab bao gồm mọi chức năng của Matlabb , bao gồm cả các đối tượng. Thêm vào đó, chúng ta không cần quan tâm đến chức năng và cách thức chuyển đổi như thế nào cả, tất cả những điều đó được thực hiện bởi trình biên dịch M-Complier.

Trên đây là một vài điều ban đầu giới thiệu về M-Complier. Vấn đề này sẽ liên tục được viết tiếp vì em còn đang bận ôn thi học kì nên nghiên cứu được gì thì sẽ post lên luôn.

Chúc mọi người thành công.

Bắt đầu nhanh với trình biên dịch M-Complier

Muốn dùng được M-complier thì người dùng cần phải biết kĩ năng viết M-file.

Em đọc qua help của Matlab thì thấy phần M-complier cần nắm được các vấn đề sau và em cũng định viết theo sườn này:

- Bắt đầu nhanh (quick start) với trình dich M-complier và làm thử một vài ví dụ nho nhỏ về trình biên dịch này về dịch một ứng dụng chạy độc lập, ...
- Giới hạn và sự hạn chế của trình biên dịch đối với dịch các Toolbox, các code Matlab, các ứng dụng chạy độc lập, giải quyết vấn đề hàm Callback là mất hàm, tìm hàm bị mất trong M-file,..

- Làm việc với mcc
- Các ứng dụng độc lập
- Thư viện
- Các components Com và Exel : 3 phần này sẽ giới thiệu các dịch file và các file đích được dịch có chức năng thế nào,...

Trong này nó hướng dẫn cả về dịch ra file và các chú ý với cả 2 hệ điều hành Windows và Linux. Tuy nhiên trong các bài ở đây và dưới chúng ta chỉ quan tâm đến hệ điều hành Windows (hệ điều hành thông dụng của Microsoft - bác Bill).

Đầu tiên là bắt đầu nhanh với trình biên dịch M-Complier. Phần này sẽ giới thiệu qua cú pháp của lệnh dịch của một vài ứng dụng và thành phần. Đầu tiên không đi quá sâu vào cú pháp và ý nghĩa của tham số nhiều mà nó sẽ được viết chi tiết ở nhữn bài sau đó.

1/ Dịch để tạo một ứng dụng chạy độc lập.

To create a stand-alone application from the M-file mymfunction, use the command mcc -m mymfunction.m
This creates a stand-alone executable named mymfunction.exe on Windows and mymfunction on Linux.
Để tạo một ứng dụng từ một hàm trong M-file. Ví dụ dịch một hàm trong M-file ex1.m
Code:

```
function y = Absolute(x)

    if (x>0), y = x

    else y= -x

    end
```

Sau đó trong cửa sổ command dùng lệnh sau:
Code:

```
>> mcc -m ex1
```

Mặc dù nó có thông báo Warning nhưng kệ nó các bạn ạ. mục đích ở đây chúng ta chỉ quan tâm đến là dịch được ra file .exe còn chi tiết chúng ta sẽ xem xét sau. Tại sao lại có thông báo ,...

Các bạn mở thư mục Works trong thư mục cài đặt Matlab7 ra xem sẽ thấy fil e ex1.exe trong đó. coi như là thành công bước đầu rùi.

Đi tập thể dục cái đã  . tối viết tiếp.

chúc các bạn thành công.

Em xin tiếp tục chủ đề này.

Các bạn nên nhớ file ex1.m chỉ như là một ví dụ khi dịch có warning thì bạn cứ bỏ qua nha, chưa cần quan tâm mà chỉ quan tâm đến cách thức đã còn chi tiết bàn sau.

2/ Dịch để tạo ra thư viện từ M file ex1.m chúng ta dùng lệnh sau:
Code:

```
>>mcc -l ex1.m
```

Khi đó trình biên dịch sẽ tạo ra các file sau:

- ex1.c
- ex1.h
- ex1.dll
- ex1.lib
- ex1_main.c
- ex1_mcc_component_data.c
- ex1.ctf

Để có thể test được các component bạn phải copy thư mục **<thư mục cài Matlab7>\bin\win32** vào thư mục có chứa thư viện liên kết động của bạn

Để triển khai được các ứng dụng của bạn trong các máy PC mà không cài Matlab có version giống với máy phát triển các component đó thì bạn đóng gói các component đó và thiết lập trên máy đó như sau:

- Cài đặt **MATLAB Component Runtime**(MCR) bằng cách copy file MCRInstaller.exe trong thư mục **<thư mục cài Matlab>\toolbox\compiler\deploy\win32** vào một thư mục và cài file đó.
- Copy component đó và file .ctf của trong list file vừa được tạo vào một thư mục nào đó ví dụ: C:\approot. Rồi bạn copy thư mục **<ổ cài đặt>:\Program Files\MathWorks\MATLAB Component Runtime\v70\bin\win32**vào trong thư mục đó. Rồi test ứng dụng.

Bây giờ em xin dừng bút, tắt máy ôn thi tí đã không có điểm kém quá mama mắng chết.

Chúc mọi người thành công.

Chào bạn, việc tạo nhiều form trong GUI thì rất đơn giản mà. Mình đã làm cái này tức là giống trong Simulink chọn Setting cho chương trình ControlDCMotor, nhưng viết xong rùi bị virus xoá hết cả ổ nên không còn gì cả  , chưa có thời gian làm lại vì làm cái này hơi mất thời gian.

Bạn có thể thấy việc làm 2 form thông qua ấn nút close trong chương trình ControlDCMotor đó bạn. Việc dùng nhiều form khi lập trình thì vấn đề cần quan tâm là việc trao đổi dữ liệu giữa các form với nhau phải thông qua workspace (mình đã dùng nhiều cách rùi nhưng thấy đây là cách hiệu quả nhất) qua lệnh **load** và **save** vào file .mat.

Có thời gian mình sẽ viết bài cụ thể về vấn đề này.

Chúc các bạn thành công.

Mình đã từng thử nhiều cách để link các form với nhau nhưng thấy có thể dùng các cách sau: lưu dữ liệu vào một file .mat và dùng các lệnh **load, save** để lấy dữ liệu và lưu dữ liệu vào. Như thế các form có thể dùng chung một dữ liệu. Nhưng nếu dữ liệu của form sau được sử dụng của form trước thì có thể là khi form sau chạy xong và biến mất thì ta mới nên cho form trước active và khi đó nên có biện pháp cập nhật dữ liệu vào như là thêm nút Update có tác dụng lây dữ liệu vào chẳng hạn. Khi nào nhớ ra cách nào khác mình sẽ post sau.

Còn example code thì tớ k còn trong máy rùi. Bạn lên site mathworks.com xem nhé.

Còn viết chữ Latex vào trong Simulink thì mình chưa nghiên cứu bạn ạ nên mình không biêt.

Chúc các bạn thành công.

Chào bạn.

Trong Matlab mình chưa làm giao tiếp với cổng LPT bạn ạ. Nếu được bạn có thể cho mình xem Project của bạn được không? Mình cùng trao đổi nhé.

Theo mình nghĩ nếu muốn truyền tốc độ cao mình có thể dùng các ngôn ngữ khác như C, C++, VC++ và dùng thư viện inoutport32.dll để giao tiếp rất đơn giản bạn ạ.

Chúc bạn thành công.

Mình đã trả lời bạn vanminh_mcp qua Yahoo  , nên có lẽ bạn ấy không hỏi ở đây nữa.

Để set thuộc tính string này:

1. Cho Edit Text đó ở chế độ multi line, cho giá trị max lớn hơn 1. Chính là số dòng có thể hiển thị được trên Edit Text.

2. Thiết lập thuộc tính String, có thể có các chữ số là biến mà ta dùng, dùng lệnh **sprintf** chính là format String như trong C với lệnh printf. Dùng kí tự '\n' để xuống dòng.

Gõ: **help sprintf** trong command để biết cách dùng.

Ví dụ: str = sprintf(' %d + %d = %d',x,y,x+y);

Chúc các bạn thành công.

Giao tiếp RS232 trên PC bằng Visual Basic 6.0

05-05-2006, 00:27

Chào các bạn.

Đây là TUT về RS232 dùng VC++ để lập trình :
http://www.picvietnam.com/forum/show...=newpost&t=274

Mình xin đưa cho bạn một Code mẫu đơn giản viết băng VB và có file gửi kèm.

Code:

```
Private Sub cmdClear_Click()

    Text1.Text = ""


End Sub


Private Sub cmdExit_Click()

Unload Me

End Sub


Private Sub cmdSend_Click()

    MSComm1.Output = Text2.Text

End Sub


Private Sub Form_Load()

    cmdSend.Caption = "&Send"

    Text1.Text = ""

    Text2.Text = ""

    Text1.Enabled = False

    cmdExit.Caption = "&Exit"

    With MSComm1

        .Settings = "9600,N,8,1"

        .CommPort = 1
```

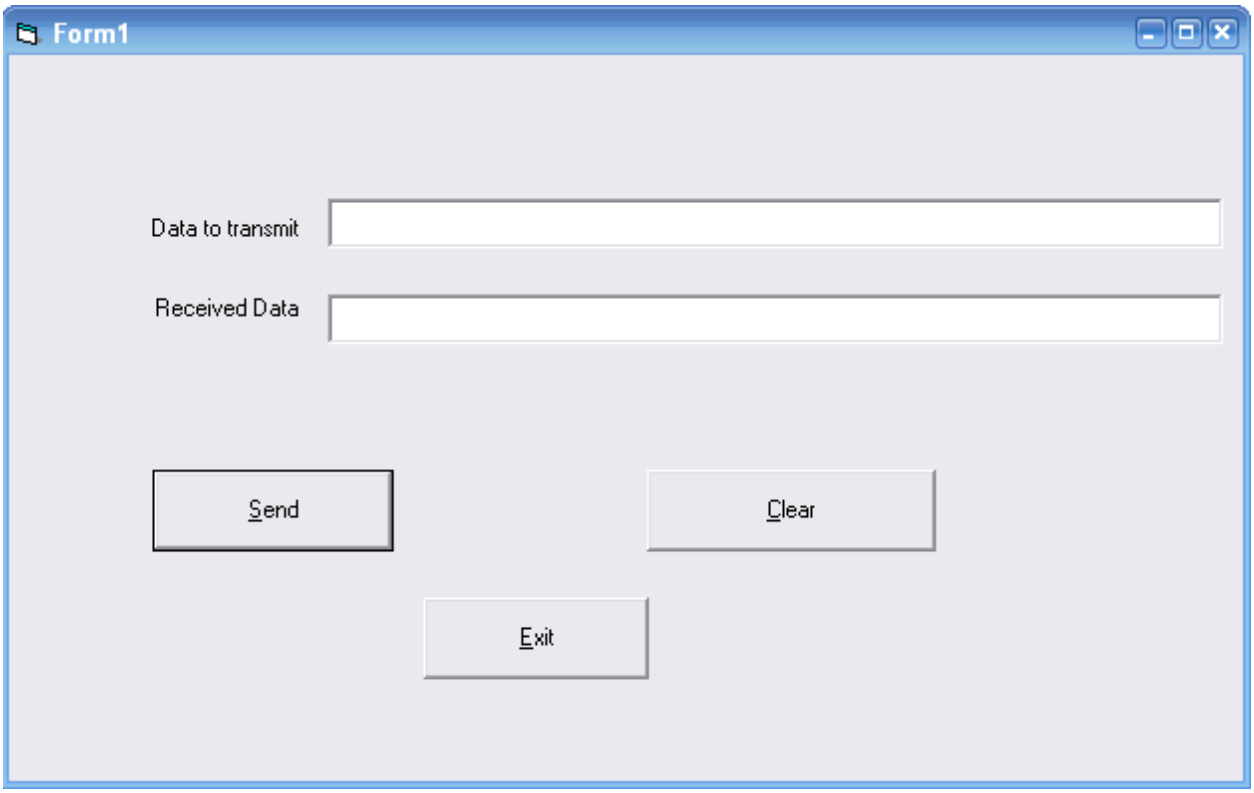
```
        .RThreshold = 1
        .SThreshold = 0
        .InputMode = comInputModeText
        .InputLen = 0
        .Handshaking = comNone
        .InBufferSize = 1024
        .OutBufferSize = 1024
        If .PortOpen = False Then
          .PortOpen = True
        End If

    End With
End Sub

Private Sub MSComm1_OnComm()
    Dim Buffer As Variant
    If MSComm1.CommEvent = comEvReceive Then
        Text1.Text = Text1.Text + MSComm1.Input
    End If
End Sub
```

Chúc bạn thành công.

Attached Files

- 

Mấy bác cho em hỏi cái này. Em truyền nhận dữ liệu giữa 16f877 và VB6 đã nhận đc nhưng chỉ với thạch anh 4000000Hz, khi thay đổi cái này thì lại không nhận được. Em làm với kit thạch anh 24Mhz ko thể nhận dữ liệu được. Mong các bác chỉ giáo giùm em với.

VB:

```
Private Sub Command1_Click()
On Error GoTo OpenFalse
If MSComm1.PortOpen = False Then
MSComm1.PortOpen = True
End If
Exit Sub
OpenFalse:
MsgBox Err.Description & vbLf + vbCr + "Loi, khong the mo cong. Hay dong cac ung dung dang su dung cong COM"

End Sub

Private Sub Command2_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
End If
End
End Sub
```

```
Private Sub Command3_Click()
On Error GoTo sendfalse
MSComm1.Output = Text1.Text
Exit Sub
sendfalse:
MsgBox Err.Description & vbLf + vbCr + "Loi, Cong COM chua duoc mo, nhan Connect"
End Sub

Private Sub Form_Load()
With MSComm1
.Settings = cmbBaudRate.Text + "N,8,1"
.RThreshold = 1
.SThreshold = 0
.InBufferSize = 1024
.OutBufferSize = 1024
.InputMode = comInputModeText
.ParityReplace = ""
.CommPort = Switch(cmbComPort.Text = "COM 1", 1, cmbComPort.Text = "COM 2", 2,
cmbComPort.Text = "COM 3", 3, cmbComPort.Text = "COM 4", 4)
End With
End Sub

Private Sub MSComm1_OnComm()
Dim StringIn As String
If MSComm1.CommEvent = comEvReceive Then
StringIn = MSComm1.Input 'Nhan chuoi du lieu to VCOM
Text2.Text = StringIn
End If
End Sub
```

PIC:

```
#include <16f877a.h>
#FUSES NOWDT, HS, NOPUT,NOPROTECT, NODEBUG, NOBROWNOUT, NOLVP, NOCPD, NOWRT
#use delay(clock=4000000)
#use rs232(baud=9600, parity=N, xmit=pin_C6, rcv=pin_C7)
#include <stdlib.h>
#byte porta = 0x05
#byte portb = 0x06
#byte portc = 0x07
#byte porte = 0x09
int8 y,i,a,b,c,d,e;
int16 x;
char string[5];
BYTE CONST maled[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x9 0};

void nhan_rs232()
```

```
{

for(i=0;i<=4;i++)
{
string[i]=getc();
}
printf("%s",string);
y=0;
x=atol(string);
a=x/10000;
x=x%10000;
b=x/1000;
x=x%1000;
c=x/100;
x=x%100;
d=x/10;
x=x%10;
e=x;

a=maled[a];
b=maled[b];
c=maled[c];
d=maled[d];
e=maled[e];

for(i=0;i<=200;i++)
{
output_A(0b01111);
output_D(a);
delay_ms(1);
output_A(0b11111);

output_A(0b10111);
output_D(b);
delay_ms(1);
output_A(0b11111);

output_A(0b11011);
output_D(c);
delay_ms(1);
output_A(0b11111);

output_A(0b11101);
output_D(d);
delay_ms(1);
output_A(0b11111);

output_A(0b11110);
output_D(e);
```

```
delay_ms(1);
output_A(0b11111);
}


}
/*void xoa()
{
for(i=0;i<=4;i++)
{
string[i]=0;
}
}*/
void main()
{
set_tris_b(0);
set_tris_e(0);
while(true)
{
if(y==0)
{
nhan_rs232();
delay_ms(100);

}
}
}
```

Em làm nhận một số 5 chữ số hiển thị led 7 đoạn

mấy anh em cho mình hỏi chút. mình đang làm đồ án nhưng tới phần giao tiếp máy tính với vđk thì nó k chạy,dưới đây là code vđk và vb của mình,mấy anh em xem giúp đỡ với:
code vđk:

```
org 00H
jmp main
org 30h
main:
mov SCON, #52h ;Port noi tiep mode 1
mov TMOD, #20h ;Timer 1 mode 2
mov TH1,#0Fdh ;9600 baud
setb TR1
loop:
jnb RI, $ ;doi nhan du lieu
clr RI
mov A, SBUF ;lay du lieu
kt_sang_trai:
cjne A, #'a', kt_sang_phai ;neu la yeu cau sang trai
MOV P0,#11001100B
CALL DELAY_50MS
```

```
MOV P0,#0
CALL DELAY_50MS
jmp main
kt_sang_phai:
cjne A, #'d', kt_tien ;Neu la yeu cau sang phai
MOV P0,#10111011B
CALL DELAY_50MS
MOV P0,#0
CALL DELAY_50MS
jmp main

kt_tien:
cjne A, #'w', kt_lui ;Neu la yeu cau tien
MOV P0,#11101110B ;CHI QUAN TAM DEN 4 BIT THAP 1000 UNG VOI ROBOT CHAY THANG
CALL DELAY_50MS
MOV P0,#0
CALL DELAY_50MS
jmp main

kt_lui:
cjne A,#'s', kt_dung ;Neu la yeu cau lui
MOV P0,#11011101B ;LUI ROBOT
CALL DELAY_50MS
MOV P0,#0
CALL DELAY_50MS
jmp main

kt_dung: ;Neu la yeu cau dung
cjne A,#'j', kt_cam_trai
MOV P0,#10101010B;DUNG ROBOT VA QUAY CAMERA
CALL DELAY_50MS
MOV P0,#0
CALL DELAY_50MS
jmp main

kt_cam_trai:
cjne A, #'k',kt_cam_phai
MOV P0,#10011001B;DUNG ROBOT VA QUAY CAMERA
CALL DELAY_50MS
MOV P0,#0
CALL DELAY_50MS ;Neu la yeu cau quay camera sang phai
jmp main
kt_cam_phai:
cjne A, #'l',kt_bao_dong
MOV P0,#10001000B;DUNG ROBOT VA QUAY CAMERA
CALL DELAY_50MS
MOV P0,#0
CALL DELAY_50MS ;Neu la yeu cau quay camera sang phai
jmp main
```

```
kt_bao_dong:
cjne A, #'h',kt_dung
MOV P0,#01110111B;bao dong
CALL DELAY_50MS
MOV P0,#0 ;Neu la yeu cau quay camera sang phai
CALL DELAY_50MS
jmp main

DELAY_50MS:
MOV TMOD,#01H
MOV R0,#50
LOOP20:
MOV TH0,#0ECH
MOV TL0,#78H
SETB TR0
JNB TF0,$
CLR TF0
CLR TR0
DJNZ R0,LOOP20
RET
end
''''''''''''''''''''
''''''''''''''''''''
```

chương trình vb:

```
Public Class Form1

Private Sub Nut_thoat_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_thoat.Click
Form2.ShowDialog()

End Sub

Private Sub HướngDẫnToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles HướngDẫnToolStripMenuItem.Click


End Sub

Private Sub ThôngTinToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ThôngTinToolStripMenuItem.Click

End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
If SerialPort1.IsOpen Then SerialPort1.Close()
SerialPort1.Open()
End Sub
```

```
Private Sub Button_phai_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_phai.Click
SerialPort1.Write("d")
Label2.Text = ("Robot đang rẽ phải")
End Sub

Private Sub Button_toi_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_toi.Click
SerialPort1.Write("w")
Label2.Text = ("Robot đang đi tới")
End Sub

Private Sub Button_trai_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_trai.Click
SerialPort1.Write("a")
Label2.Text = ("Robot đang rẽ trái")
End Sub

Private Sub Button_dung_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_dung.Click
SerialPort1.Write("j")
Label2.Text = ("Robot đã dừng")
End Sub

Private Sub Button_lui_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button_lui.Click
SerialPort1.Write("s")
Label2.Text = ("Robot đang đi lui")
End Sub

Private Sub Form1_FormClosed(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles MyBase.FormClosed
If (SerialPort1.IsOpen) Then SerialPort1.Close()
End Sub

Private Sub TớiToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TớiToolStripMenuItem.Click
SerialPort1.Write("w")
Label2.Text = ("Robot đang đi tới")

End Sub

Private Sub LuiToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles LuiToolStripMenuItem.Click
SerialPort1.Write("s")
Label2.Text = ("Robot đang đi lui")
End Sub

Private Sub PhảiToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
Handles PhảiToolStripMenuItem.Click
SerialPort1.Write("d")
Label2.Text = ("Robot đang rẽ phải")
End Sub

Private Sub TráiToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles TráiToolStripMenuItem.Click
SerialPort1.Write("a")
Label2.Text = ("Robot đang rẽ trái")
End Sub

Private Sub DừngToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles DừngToolStripMenuItem.Click
SerialPort1.Write("j")
Label2.Text = ("Robot đã dừng")
End Sub

Private Sub ThôngTinToolStripMenuItem1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ThôngTinToolStripMenuItem1.Click
MsgBox(" Chương trình điều khiển từ xa phiên bản 1.0 này là Đồ án môn học Hệ thống Cơ điện tử, do
sinh viên Nguyễn Quốc Vũ - lớp 06CDT2 - ngành Cơ Điện Tử - Trường Đại Học Bách Khoa Đà Nẵng viết.
")
End Sub

Private Sub CáchĐiềuKhiểnToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CáchĐiềuKhiểnToolStripMenuItem.Click
MsgBox(" Bạn có thể điều khiển Robot bằng bàn phím trên máy tính: Ctrl + W: đi tới ; Ctrl + S: đi lui;
Ctrl + A: sang trái ; Ctrl + D: sang phải ; Cltr + J: dừng ; Cltr + K: Camera quay trái ; Cltr + L: Camera
quay phải")
End Sub

Private Sub PhôngNềnToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles PhôngNềnToolStripMenuItem.Click
ColorDialog1.ShowDialog()
Me.BackColor = ColorDialog1.Color
End Sub

Private Sub MàuNútNhấnToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MàuNútNhấnToolStripMenuItem.Click
ColorDialog1.ShowDialog()
Button_cam_phai.BackColor = ColorDialog1.Color
Button_cam_trai.BackColor = ColorDialog1.Color
Button_trai.BackColor = ColorDialog1.Color
Button_phai.BackColor = ColorDialog1.Color
Button_toi.BackColor = ColorDialog1.Color
Button_lui.BackColor = ColorDialog1.Color
Button_dung.BackColor = ColorDialog1.Color
Button_thoat.BackColor = ColorDialog1.Color
```

```
End Sub

Private Sub Button_cam_trai_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button_cam_trai.Click
SerialPort1.Write("k")
Label2.Text = ("Camera đang quay trái")
End Sub

Private Sub Button_cam_phai_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button_cam_phai.Click
SerialPort1.Write("l")
Label2.Text = ("Camera đang quay phải")
End Sub

Private Sub CameraTráiToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CameraTráiToolStripMenuItem.Click
SerialPort1.Write("k")
Label2.Text = ("Camera đang quay trái")
End Sub

Private Sub CameraPhảiToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles CameraPhảiToolStripMenuItem.Click
SerialPort1.Write("l")
Label2.Text = ("Camera đang quay phải")
End Sub
End Class

Public Class Form2

Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Label1.Click


End Sub

Private Sub Nut_khong_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Nut_khong.Click
Me.Close()
End Sub

Private Sub Nut_co_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Nut_co.Click
End

End Sub

Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
```

```
Beep()
End Sub
End Class
```

mong mấy bác giúp đỡ dùm,mình còn mấy ngày nữa là bảo vệ r

P89V51RD2 - Giao tiếp máy tính qua RS232

13-04-2012, 02:00

1. **Tóm tắt ý tưởng thiết kế:**

   - Gửi lệnh từ PC điều khiển chip
   vd: Gửi SET+P0.0=1; -> Vđk sẽ set chân P0.0 lên mức 1, sau đó gửi chuỗi OK lên PC
   Gửi GET+P0.0=?; -> Vđk sẽ đọc chân P0.0 và gửi kết quả lên PC, sau đó gửi chuỗi OK lên PC
   Nếu lệnh gửi từ PC xuống ko nằm trong tập lệnh hoặc sai cú pháp, vđk sẽ gửi lên chuỗi Error

   - Tất cả các lệnh đều kết thúc bằng dấu chấm phẩy ";", và ko dài quá "n" ký tự, "n" sẽ được qui định trước.

   - Chỉ thực hiện một lệnh. Lệnh trước đó phải hoàn thành (hoàn tất việc gửi kết quả về PC) mới nhận lệnh tiếp theo.

   - Sau khi thực hiện xong lệnh, nếu không nhận được lệnh khác từ PC, vđk sẽ đi vào chế độ nghỉ IDL nhằm tiết kiệm năng lượng.

   - Chương trình viết bằng Keil C.

   2. Tại sao lại chọn vđk P89V51RD2 và KeilC

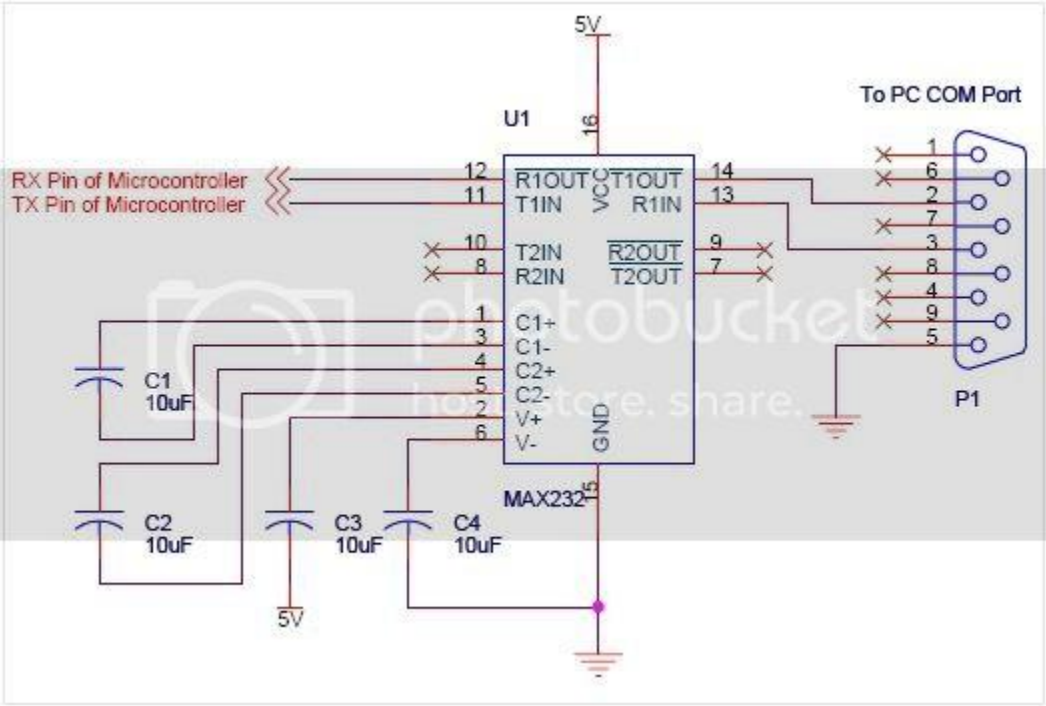
   - P89V51RD2 có bộ nhớ mở rộng: RAM là 1Kbytes và bộ nhớ code 64KBytes. Vì vậy thoải mái Code, khai báo biến xả láng, và dễ dàng mở rộng chương trình.

   - P89V51RD2 còn cho phép nạp chương trình bằng bootloader thông qua RS232 lun, vì vậy phần cứng giao tiếp máy tính của chúng ta vô tình cũng dùng để nạp chương trình lun (2 in 1).

   - KeilC là trình biên dịch thân thuộc của FAN 8051. Tối ưu hóa chương trình tốt, ít lỗi, nhưng viết code hơi cực.

   3. Phần cứng

   a. Nếu bạn nào chưa quen làm phần cứng có thể làm mạch đơn giản sau:



b. Còn đây là phần cứng mình dùng để thực hiện tut này.

***Sơ đồ nguyên lý và Mạch in:**

Mạch gồm:

- IC 7085 - Ổn áp nguồn 5V, 1A

- Vi điều khiển 8051 (AT89S52 hoặc P89V51RD2)

- MAX232 - giao tiếp RS232 và nạp đối với P89V51RD2

- Jack nạp nối tiếp SPI đối với AT89S52, và dùng lun để mở rộng ngoại vi, chẳng hạn: Mở rộng ngõ ra bằng IC dịch, và đây là các Pin giao tiếp SPI của P89V51RD2.

- DS1307 - RealTime giao tiếp I2C

- LCD 16x2, giao tiếp 4 bit, điều khiển đèn nền (bật/tắt hoặc độ sáng).

- 1 modul thu phát RF - cái này trên Nhật Tảo bán nhiều lắm (loại 4 nút, khoảng cách chừng 15m, giá khoảng 50÷60 Nghìn VNĐ)

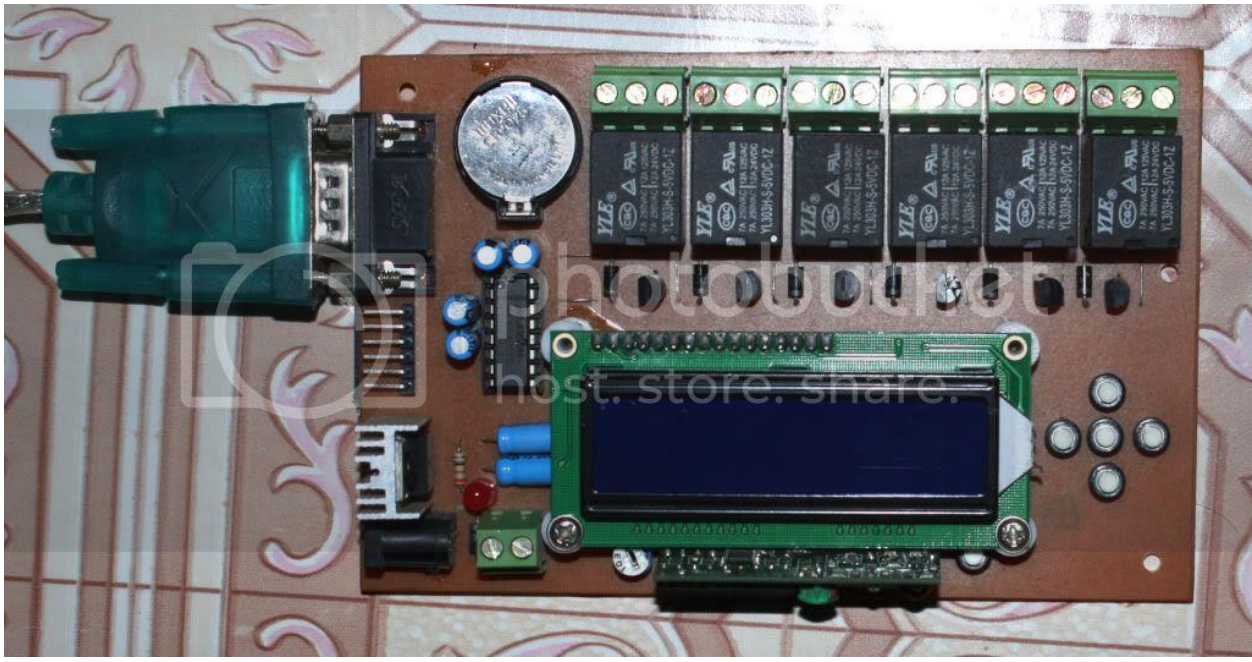
- 6 Relay 5V

- 5 nút nhấn + 1 nút Reset

- Trên board còn có một vị trí điều khiển bằng chạm tay nhưng thiết kế bị lỗi, các bạn tìm cách khắc phục giúp mình, nếu ko thì lúc thi công không cần gắn linh kiện cho khối này (khối có ngõ ra được đặt tên là "touch").

Cuối cùng, các bạn có thể down tại đây: P89V51RD2_RS232.rar

***Mạch sau khi thi công:**



4. Phần mềm

Cốt lõi của chương trình gồm:

- Chương trình main: Đầu tiên nó sẽ đi thực hiện các khai báo và lệnh khởi tạo (biến, UART), sau đó sẽ thực hiện một lệnh duy nhất trong vòng lặp while là PCON |= 0x01; để ru vđk ngủ.

- Chương trình phục vụ ngắt Nhận/Phát UART đánh thức vđk nhận lệnh, tìm lệnh, thực hiện và gửi kết quả về PC.

- Một tập lệnh xây dựng sẵn. Mình sẽ demo một số lệnh như sau:

ACK; (Acknowlegement) Nếu nhận được lệnh này vđk sẽ trả lời là ACK.

WRP+Px=0xxx; (WRite Port) Ghi một số dạng Hex ra PORT

RDP+Px=?; (ReaD Port) Đọc Port và gửi kết quả về PC

SET+Px.x=x; (SET pin) Ghi mức logic ra PIN

GET+Px.x=?; (GET pin) Đọc mức logic của PIN

WRA+0xxx=0xxx; (WRite Address) Ghi một giá trị dạng Hex xuống bộ nhớ.

RDA+0xxx=?; (ReaD Address) Đọc một giá trị của ô nhớ.

*Cần chú ý:
-Trong hai lệnh WRA và RDA: Address không phải là địa chỉ của bộ nhớ RAM, mà là chỉ số của một phần tử mảng trên bộ nhớ RAM được sử dụng để lưu các biến dùng chung giữa vđk và PC.

-Nếu dùng lệnh SET để đưa 1 trong 2 PIN của module UART (P3.0 và P3.1) về mức 0 => Không thể giao tiếp với PC được nữa.

-Trong lệnh gửi từ PC xuống phải ko chứa các khoảng trống.

*Một số ví dụ:
Send: ACK;
Nhận: ACKOK

Send: WRP+P0=0x00;
Nhận: OK

Send: RDP+P0=?;
Nhận: 0x00OK

Send: SET+P0.0=1;
Nhận: OK;

Send: GET+P0.0=?;
Nhận: 1OK;

Send: WRA+0x00=0xAA;
Nhận: OK;

Send: RDA+0x00=?;
Nhận: 0xAAOK;

Send: ABC;
Nhận: Error

Code hoàn chỉnh, các bạn có thể down tại đây:

13/04/2012: Code_P89V51RD2_RS232.rar

Chú ý các bạn Down file mới nhất, các thắc mắc mình chỉ trả lời với file mới nhất. Mình sẽ cập nhật thường xuyên để có file Code đơn giản và dễ hiểu nhất.

Mong các bạn đóng góp ý kiến để bài viết hoàn chỉnh hơn!! Cảm ơn các bạn đã đọc.

2. **Một số tính năng của P89V51RD2 được cải tiến so với 8051:**

1. Fosc lên tới 40MHz

2. Bộ chia có thể lựa chọn ở 2 mode bằng phần mềm: Fosc÷12 (bình thường - mặc định) hoặc Fosc÷6 (cải tiến)

3. Giao tiếp SPI ở chế độ Master hoặc Slave

4. Nạp chương trình bằng Bootloader

5. Vì nạp chương trình được bằng Bootloader nên mình nghĩ bộ nhớ Code cho phép ghi lúc đang chạy chương trình => có thể sử dụng để lưu thông số, mà không cần thêm EEPROM bên ngoài. Cái này mình sẽ kiểm tra sau.

Giá của nó thì hơi đắt bạn à, khoảng 70.000 VNĐ

Ngoài Keil C, mình còn dùng bộ MikroC để code cho các dòng vđk. Mình thấy MikroC hỗ trợ thư viện rất mạnh, thích hợp dùng để làm quen với phần cứng khi tìm hiểu các ngoại vi mới.

Trong MikroC mình nhận ra cái này chưa tốt, không biết có cao thủ nào khắc phục được không. Khi mình xuất chuỗi ra LCD như sau:

Lcd_Out_Cp("Hello, World!")

Biên dịch thì thấy bộ nhớ RAM đã sử dụng tăng lên đúng bằng số ký tự của chuỗi. Vì vậy mình đoán là MikroC lưu các chuỗi trên bộ nhớ RAM.

Điều này làm tiêu tốn một phần ko nhỏ bộ nhớ RAM.

erial Port - lập trình giao tiếp nối tiếp

14-12-2005, 11:18

Do mình viết bài trong "Viết chương trình = VB giao tiếp máy tính có lẽ là không tổng quát cho lắm. Tôi xin copy tất cả các bài của tôi vào box này để cho tổng quát hơn giới thiệu về giao tiếp nối tiếp qua Serial Port như RS-232, RS-485. Hi vọng là cho các bạn có kiến thức đầy đủ về lập trình nối tiếp nói chung và qua RS232, RS-485 nói riêng.

Giới thiệu giao tiếp cổng com - chapter1

Như đã hứa. Nay tôi xin giới thiệu cho các bạn chi tiết các thông tin, cách lập trình truyền thông nối tiếp dùng chuẩn RS-232 và RS-485.

Có thể có nhiều bạn đã biết rồi hoặc là chưa biết nhưng tôi mong rằng sẽ bổ sung cho các bạn một số điều cơ bản.

Tất cả mọi vấn đề tôi dịch từ cuốn Serial_complete của Jan Axelson và thực tế lập trình truyền thông giao tiếp với vdk AT89C51(một loại vi điều khiển đơn giản). Hi vọng các vấn đề tôi đưa ra sẽ làm các bạn môt phần nào hiểu biết thêm về cổng thông dụng này cổng Comm(communicactions).

Do thời gian không có nhiều vừa làm vừa ôn thi nên mỗi ngày tôi sẽ up lên từng vấn đề một. Mong các bạn góp ý thêm. Tôi làm cái này không có ý qua mặt các cao thủ, mong các cao thủ bỏ quá cho.
Các bạn có thể có thêm nhiều ví dụ khi vào trang web http://www.lvr.com

1. Thứ nhất tôi xin giới thiệu một cách khái quát về các cổng và đặc biệt là cổng RS-232 và RS-485.

Trước hết tôi xin định nghĩa liên kết: là dùng dây dẫn hoặc trung gian khác như công nghệ không dây,.. để kết nối các máy tính hoặc là kết nối các thiết bị kết nối qua PC. Khoảng cách có thể rất ngắn vài cm đến hàng ngàn km, thời gian có thể 1 giây có khi đến hàng 1 tuần. Các điểm kết nối vào mạng có thể là 2 hoặc nhiều điểm.

Đặc điểm so sánh giữa các cổng nay quên không mang bảng so sánh, mai mang sau)

- Chuẩn RS-232 chỉ có thể kết nối nhiều nhất 2 thiết bị, với khoảng cách dài nhất là 50-100 feet(12,7->25,4 m), tốc độ 20k bit/s
- Chuẩn RS-485 có thể kết nối tối đa là 32 thiết bị, khoảng cách dài hơn tối đa là 4000feet(1016 m-> hơn 1km) gấp 40 lần RS-232. Tốc độ cao 10 mega bit/s. Cả hai chuẩn này có thẻ có sẵn trong mainboard khi mua hoặc là có thể lắp thêm rất dễ dàng, giá mua rất rẻ so với các giao diện khác.Chuẩn RS-232 dùng rộng rãi, mua dễ dàng, đơn giản khi lắp thêm với nhiều cách thiết lập. Còn RS-485 dùng với khoảng cách lớn hơn, tốc độ cao hơn, nhiều đầu nối hơn.
- Chuẩn IrDA(Inared Data Asociation) dùng các UART giống nhau và định dạng dữ liệu giống như RS-232 nhưng có thêm bộ giải mã. Dữ liệu truyền từ nguồn phát hồng ngoại đến các thiết bị không giây. Giao diện này rất là có ích cho các liên kết ngắt, giữa các thiết bị mà không thể có cáp nối ở giữa(có thể là cho đẹp).
- Chuẩn MIDI(Musical instrument digital interface): giao diện số hóa các dụng cụ âm nhạc. Chuẩn này dùng dòng 5mA, tốc độ 31,5k bit/s
- Microwire, I2C,SPI là các chuẩn nối tiếp đồng bộ, dùng trong các liên kết ngắn. Nhiều vdk có 1 hoặc nhiều chuẩn này.
- USB(Universial Sẻial Bus) và Fireware(chuẩn IEEE-1384) là chuẩn mới, tốc độ cao, thông minh kết nối với PC và các PC khác, thiết bị ngoại vi. USB ra đời đã dần thay thế chuẩn RS-232 và chuẩn máy in Centronic như là một lựa chọn mới hiện đại cho các TB ngoại vi.

Fireware tốc độc truyền dữ liệu nhanh hơn nhiều và được dùng để truyền nhạc tiếng, nhạc hình, hoặc các block dung lượng lớn.

- Ethernet là các chuẩn mạng gần gũi thường dùng trong nhiều mạng. Tốc độ cao nhưng yêu cầu phần cứng và phần mềm khá phức tạp, đắt hơn nhiều so với các chuẩn khác.
- Đồng hành với chuẩn nối tiếp là chuẩn song song, có nhiều đường dữ liệu. Ví song song nên chuyển nhiều bit cùng một lúc, rất nhanh. Thường có một loạt các đường dữ liệu -> dữ liệu truyền đi theo một chiều ở một thời điểm. Nếu dùng để nối khoảng cách xa thì tiền mua quả là đắt đỏ cho việc thực hành của sinh viên cũng như các ứng dụng trong công nghiệp, ..
- Chuẩn máy in Cenntronics(IEEE-1284). Mọi pc đều có chuẩn này. Tốc độ truyền cao qua cáp. Được ứng dụng với các máy quét, các thiết bị lưu trữ mở rộng như đĩa cứng,.. và nhiều thiết bị ngoại vi đặc biệt khác. Chuẩn IEEE-488 là chuẩn song song dùng trong các ứng dụng điều khiển và trong âm nhạc.

Nói thêm về cổng RS-232 và RS-485:

Cổng nối tiếp là một phần của PC ngay khi nó mới ra đời. Với mỗi cổng Com hoặc Comm(communications) trong PC là một cổng nối tiếp không đồng bộ được điều khiển bởi cácUART. Mỗi cổng Com có thể có giao diện RS-232, RS – 485 hoặc cổng có thể để dành cho một modem trong hoặc thiết bị khác. Mỗi PC có thể có các dạng khác nhau của các cổng nối tiếp như USB, Firewire, và I2C nhưng chúng dùng các giao thức khác nhau và yêu cầu các thành phần khác nhau.

Giao diện nối tiếp mới nhất là USB và Firewire với các tình năng: nhanh, và nhiều lợi ích khác nữa. Thực tế thì nên dùng USB trong việc thay thế cổng RS-232 ở bất kì nơi nào có thể được trong các thiết kế mới, ứng dụng mới. Và với nhều thiết bị ngoại vi, giao diện mới này lại rất là phù hợp.

Nhưng RS – 232 và các giao diện giống nó vẫn sẽ tiếp tục phổ biến trong các ứng dụng như là hệ thống điều khiển, điều hành. Những giao diện này không đắt đỏ, dễ dàng lập trình, cho phép cáp dài,và dễ dùng kết hợp với các thiết bị vi điều khiển rẻ tiền, các máy tính cũ. Như cổng USB đã được sử dụng rộng rãi, các bộ chuyển đổi sẵn sàng để chuyển USB thành cổng RS-232 hoặc RS – 285. Bộ chuyển đổi sẽ kết nối với cổng USB của PC và chuyển đổi giữa USB và các giao diện khác. Thiết lập rất đơn giản để thêm một cổng RS-232 hoặc RS – 485 vào bất kỳ hệ thống nào.

Uart

Tiếp theo tôi xin giới thiệu cho các bạn UART:

Trong những máy tính IBM – PC đầu tiên, UART điều khiển cổng nối tiếp là 8250, tốc độ lớn nhất là 57.600 bit /giây. Các UART từ thời gian này đã được tiếp tục cải thiện thêm nhiều đặc điểm mới như thêm bộ đệm, tốc độ tăng lên. Ngày nay, UART trong PC là một phần không thể thiếu của chíp đa chức năng bao gồm một hay nhiều UART hỗ trọ cổng song song, thiết bị lưu trữ và các bộ phận hệ thống khác.

UART chuyển đổi giữa dữ liệu nối tiếp và song song. Một chiều, UART chuyển đổi dữ liệu song song bus hệ thống ra dữ liệu nối tiếp để truyền đi. Một chiều khác, UART chuyển đổi dữ liệu nhận được dạng dữ liệu nối tiếp thành dạng dữ liệu song song cho CPU có thể đọc vào bus hệ thống.

UART của PC hỗ trợ cả hai kiểu giao tiếp là giao tiếp đồng thời và không giao tiếp đồng thời. Giao tiếp đồng thời tức là UART có thể gửi và nhận dữ liệu vào cùng một thời điểm. Còn giao tiếp không đồng thời(không kép) là chỉ có một thiết bị có thể chuyển dữ liệu vào một thời điểm, với tín hiệu điều khiển hoặc mã sẽ quyết định bên nào có thể truyền dữ liệu. Giao tiếp không đồng thời được thực hiện khi mà

cả 2 chiều chia sẻ một đường dẫn hoặc nếu có 2 đường nhưng cả 2 thiết bị chỉ giao tiếp qua một đường ở cùng một thời điểm.
Thêm vào đường dữ liệu, UART hỗ trợ bắt tay chuẩn RS232 và tín hiêu điều khiển như RTS, CTS, DTR, DCR, RT và CD.

Để thuận tiện, các chương trình gửi và nhận dữ liệu trong định dạng không đồng bộ đơn giản hơn những gì bạn tưởng. PC và nhiều vi xủ lí khác có một bộ phận gọi là UART(universal asynchronous receiver/transmitter: truyền /nhận không đồng bộ chung) vì thế có thể vận dụng phần lớn những chi tiết truyền và nhận dữ liệu.

Trong PC, hệ điều hành và ngôn ngữ lập trình hỗ trợ cho lập trình liên kết nối tiếp mà không cần phải hiểu rõ chi tiết cấu trúc UART . Để mở liên kết, ứng dụng lựa chọn một tần số dữ liệu hoặc là thiết lập khác hoặc cho phép truyền thông tại các cổng. Để gửi 1 byte, ứng dụng ghi byte này vào bộ đệm truyền của cổng được lựa chọn, và UART gửi dữ liệu này, từng bít một, trong định dạng yêu cầu, thêm bít Start, bít Stop, bít chẵn lẻ khi cần. Trong một cách đơn giản, byte nhận được tự động được lưu trữ trong bộ đệm. UART cso thể dùng nhanh một ngắt để báo cho CPU và các ứng dụng biết dữ liệu đang nhận được và các sự kiện khác.

Một vài vi điều khiển không bao gồm UART, và thỉnh thoảng bạn cần nhiều hơn các UART mà vi xử lí có. Trong trường hợp này, có 2 lựa chọn: thêm UART ngoài, hoặc mô phỏng UART trong mã chương trình. Basic Stamp của Parallax là một ví dụ của chip với một UART bổ sung trong mã chương trình.
UART là một thiết bị đơn giản hỗ trợ tốt cả hai kiểu truyền thông đồng bộ và không đồng bộ.

Định dạng và giao thức đồng bộ và không đồng bộ

Định dạng và giao thức Format và Protocol):

Máy tính trong một mắt xích nối tiếp có thể là nhiều dạng khác nhau, nhưng tất cả chúng phải cùng đồng ý một thoả thuận và qui tắc để cho dữ liệu có thể trao đổi giữa chúng. Sự thoả thuận này phải chắc chắn rằng mọi sự chuyển dữ liệu phải tìm thấy được nơi nó cần đến, và mỗi máy tính phải hiểu thông điệp gửi tới nó.

Dưới đây giới thiệu về cách định dạng dữ liệu và giao thức dùng trong truyền thông nối tiếp. Chủ yếu là định dạng không đồng bộ dùng 2 chuẩn thông dụng là RS-232 và RS-485.

Gửi dữ liệu nối tiếp

Trong một liên kết nối tiếp, nơi gửi dữ liệu sẽ gửi từng bit một ở mỗi thời điểm nối tiếp nhau. Một liên kết nối tiếp chỉ có 2 thiết bị thì phải có đường dẫn dành cho mỗi chiều truyền hoặc là nó chỉ có 1 đường dẫn được chia sẻ bởi cả 2 thiết bị với thoả thuận của 2 thiết bị này. Khi mà có 3 hoặc nhiều thiết bị, tất cả các thiết bị này thường dùng chung một đường dẫn, và giao thức mạng quyết định xem thiết bị nào có quyền truyền nhận dữ liệu.

Một tín hiệu đòi hỏi bởi tất cả mọi liên kết nối tiếp là tín hiệu xung đồng hồ, hoặc là có sự tham khảo về thời gian để điều khiển đường truyền dữ liệu. Nơi truyền và nơi nhận dùng xung đồng hồ để quyết định khi nào gửi và khi nào đọc mỗi bít. Có hai dạng định dạng dữ liệu là đồng bộ và không đồng bộ, và mỗi định dạng này dùng các dạng xung đồng hồ khác nhau.

Định dạng đồng bộ:

Trong truyền đồng bộ, mọi thiết bị dùng một xung đồng hồ được phát ra bởi một thiết bị hoặc từ một nguồn xung ngoài. Xung đồng hồ có thể có một tần số cố định hoặc cóc thể chốt tại những khoảng thời gian không đều. Mọi bít truyền đi được đồng bộ với đồng hồ. Nói cách khác, mỗi bít được truyền đi là dựa vào sự chuyển đổi của xung(như tăng hoặc giảm của sường xung). Nơi nhận dùng sự chuyển đổi xung để quyết định khi nào đọc mỗi bít truyền tới. Từ hình vẽ các bạn cũng có thể thấy là nơi truyền sẽ truyền các bit khi mà nhận thấy sự chuyển sườn xung từ cao xuống thấp, và nơi nhận thì ngược lại phát hiện khi nào có sự chuyển sườn xung từ thấp lên cao thì đọc các bit.Chi tiết chính xác của giao thức này có thể biến đổi khác đi. Ví dụ, nơi nhận có thể chốt dữ liệu nhận trong sườn xung tăng hoặc giảm, hoặc là phát hiện mức logic ở mức cao hoặc thấp. Định dạng đồng bộ dùng các cách khác nhau để bắt đầu và kết thúc việc truyền dữ liệu, bao gồm bít Start và bít Stop và tín hiệu lựa chọn chíp.

Định dạng không đồng bộ:

Trong truyền không đồng bộ, liên kết không bao gồm đường xung đồng hồ, bởi vì mỗi điểm đầu cuối của liên kết đã có xung đồng hồ cho riêng từng cái. Mỗi điểm sẽ cần phải đồng ý cùng một tần số của đồng hồ và mọi đồng hồ chỉ khác nhau một vài %. Mỗi byte truyền đi bao gồm bít Start để đồng bộ đồng hồ và một hoặc nhiều bít Stop cho tín hiệu kết thúc việc truyền trong mỗi một từ được truyền đi. Cổng RS-232 trong PC dùng định dạng khoong đồng bộ để giao tiếp với modems(thiết bị mã hoá, giải mã dữ liệu) và các thiết bị khác. Dù RS-232 có thể truyền dữ liệu đồng bộ nhưng liên kết không đồng bộ vẫn được dùng phổ biến hơn. Phần lớn liên kết RS-485 dùng giao tiếp không đồng bộ.

Truyền không đồng bộ có thể dùng một trong vài cách định dạng phổ biến. Phổ biến nhất là kiểu 8-N-1, nơi truyền sẽ truyền mỗi byte dữ liệu một bít Start, tiếp theo là 8 bít dữ liệu bắt đầu với bít 0(bít có trọng số nhỏ nhất Least Sìgnificant Bit) và kết thúc với 1 bít Stop.

Các bạn có thể xem hình vẽ để hiểu thêm về cách định dạng này.

Chữ N trong định dạng 8-N-1 chỉ rằng truyền dữ liệu không dùng bít chẵn lẻ. Một dạng định dạng khác là bao gồm một bít chẵn lẻ giống như dạng đơn giản của kiểm soát lỗi.

Khi số các bit 1 trong byte là chẵn thì bít Odd Parity Bit = 1 và bít lẻ = 0,..
Một số dạng khác không phổ biến là dùng một số khác nhau của số bít dữ liệu. Rất nhiều cổng nối tiếp hỗ trợ mọi nơi từ 5 ->8 bít dữ liệu, cộng với bít chẵn lẻ.

Tốc độ số bít là số bít một giây được truyền đi hoặc là nhận về trong một đơn vị thời gian. Tốc độ bus là số các sự kiện hình xảy ra hoặc truyền dữ liệu trên giây. Hai giá trị này thường đồng nhất với nhau trong nhiều liên kết. Trong đường dây điện thoại, môdem tốc độ cao mã hoá nhiều bít trong mỗi chu kì dữ liệu vì thế tốc độ bus thực tế nhỏ hơn tốc độ bit(bit rate).

Mọi bít cần thiết cho truyền một giá trị từ bít Start đến bít Stop gọi là một Word. Mỗi bít trong dạng Word gọi là một Character. Trong vài liên kết, các bít là kí tự văn bản(dạng chữ hoặc số), trong khi các dạng kí tự khác lại là giá trị nhị phân. Thời gian truyền các các kí tự trong một giây bằng với tổng thời gian truyền từng bít trong word cộng lại. Thêm bít start và bít Stop làm tăng thời gian truyền mỗi byte lên 25% (vì có 10 bít cần truyền trong khi chỉ dùng có 8 bít). Với định dạng 8-N-1, một byte truyền với thời gian bằng 1/10 tần số bus: do đó 9600 bít/s truyền 960 byte/s.

Nếu nơi nhận đòi hỏi phải có một thời gian kiểm tra dữ liệu nhận được, nơi truyền sẽ kéo dài độ rộng của bít Stop ra 1,5 hoặc 2 bít.

Hôm sau tôi sẽ giới thiệu cho các bạn bài về các định dạng truyền dữ liệu kiểu nhị phân và văn bản, cách thức để chống mất dữ liệu.

Cơ chế chống mất dữ liệu

Cảm ơn sự động viên của anh Công. Sợ đụng hàng thì chết. Thế thì em sẽ tiếp tục chủ đề này.

Tôi xin giới thiệu cho các bạn cơ chế chống mất dữ liệu trong khi truyền nhận dữ liệu giữa các nút trong mạng lưới:

Phần lớn các máy tính trong mạng nối tiếp có nhiều việc phải làm bên cạnh việc chờ nhận dữ liệu. Ví dụ, mỗi đơn vị dữ liệu có thể thu thập theo chu kì và lưu trữ dữ liệu tới khi một mắt xích khác trong mạng yêu cầu dữ liệu này. Hoặc một điều khiển có thể đáp ứng các điều kiện điều khiển và điều hành, thỉnh thoảng lại nhận thông tin hoặc nhận các yêu cầu từ trong mạng.

Một máy tính muốn truyền dữ liệu trong khi một máy nhận khác đang bận với các công việc khác. Việc thiết kế mạng phải đòi hỏi rằng mỗi nơi nhận có thể biết được dữ liệu nào chuyển đến nó và tất cả mọi dữ liệu đến máy nhận phải không có lỗi.

Có nhiều cách làm để thực hiện điều đó, bao gồm bắt tay(handshaking), bộ đệm(buffering), dùng dò(polling) và ngắt(interrupts) để phát hiện dữ liệu đã đến, kiểm soát lỗi(error checking), và thừa nhận dữ liệu đã tới(acknowledging). Mỗi liên kết có thể dùng một hoặc nhiều cách trong số những cách này.

Bắt tay(handshaking):

Với tín hiệu bắt tay, máy phát có thể xác định khi nào máy tính này phải truyền dữ liệu và máy nhận có thể biết khi nào nó sẵn sàng nhận dữ liệu. Tín hiệu có thể biến đổi qua RS-232 hoặc RS-485 theo giao thức chuẩn hoặc giao thức qui ước.

Một trong những dạng bắt tay về phần cứng, nơi nhận đưa ra dòng mức cao khi sẵn sàng nhận dữ liệu, và nơi truyền chờ tín hiệu này trước khi truyền dữ liệu. Nơi nhận có thể đưa ra dòng mức thấp trong mọi thời điểm, thậm chí cả trong quá trình chờ dòng phản hồi cao trước khi kết thúc quá trình truyền nhận. Một số dạng liên kết khác hoạt động giống nguyên tắc ở trên nhưng với bắt tay bằng phần mềm, bằng cách nơi nhận gửi một mã để báo nó sẵn sàng nhận dữ liệu, và một mã khác để báo báo cho nơi truyền dừng quá trình gửi dữ liệu.

Bộ đệm(Buffer):

Bộ đệm là một dạng khác để nơi nhận có thể chắc chắn là không mất một dữ liệu nào gửi đến chúng. Bộ đệm có thể có ích cho phía truyền, nơi cho phép ứng dụng làm việc có hiệu quả bằng cách lưu trữ dữ liệu để gửi khi liên kết sẵn sàng để truyền nhận dữ liệu.

Bộ đệm có thể là bộ đệm phần cứng, phần mềm hoặc cả hai. Cổng nối tiếp dùng tất cả các dạng này nhưng máy tính cổ nhất có 16 byte bộ đệm phần cứng được tích hợp trong những UART. Trong chiều nhận, điều đó có nghĩa rằng UART có thể lưu trữ 16 byte trước khi phần mềm cần đọc chúng. Trong chiều nhận, UART có thể lưu trữ 16 byte và UART sẽ cẩn thận truyền mỗi byte theo từng bít từng bít theo giao thức lựa chọn.

Khi bộ đệm phần cứng không đủ rộng, một máy tính cá nhân có thể dùng bộ đệm phần mềm, bộ đệm này có thể lập trình được kích thước và kích thước tối đa cho phép bởi bộ nhớ hệ thống. Các thiết bị

phần mềm của cổng truyền nhận dữ liệu giữa bộ đệm phần cứng và phần mềm.

Trong các vi điều khiển, bộ đệm có xu hướng trở nên nhỏ hơn, và một số chip không có bộ đệm phần cứng. Việc làm hẹp bộ nhớ đệm điều quan trọng hơn ở đây là các chíp này dùng các công nghệ khác để chắc chắn là không dữ liệu nào bị mất.

Thăm dò và ngắt:

Sự kiện gây ra ở cổng nối tiếp bao gồm khi truyền và nhận dữ liệu, thay đổi tín hiệu bắt tay, và gửi , nhận thông điệp lỗi. Có hai cách cho ứng dụng phát hiện và gây ra những sự kiện này.

Các thứ nhất là có chương trình tự động nhảy tới các chuỗi sự kiện được sắp xếp trước(như bảng vector ngắt) khi một sự kiện xảy ra. Ứng dụng phản ứng nhanh và tự động hoạt động ở cổng mà không lãng phí thời gian kiểm tra, chỉ cần biết như không có hoạt động nào xảy ra.

Dạng lập trình này gọi là chạy đua sự kiện(event-driven) bởi vì một sự kiện bên ngoài có thể xảy ra trong bất kì thời điểm nào và chương trình chạy tới một bảng đặc biệt.

Trong VB, sự kiện OnComm của MSComm(Microsoft Communication Control 6.0 - Điều khiển ActiveX) làm công việc này. OnComm chạy đáp ứng lại ngắt phần cứng hoặc bộ đếm của bộ đệm phần mềm đạt tới giá trị xảy ra sự kiện. Nhiều bộ vi điều khiển có ngắt phần cứng dùng với mục đích này.

Cách thứ hai là thăm dò bằng cách đọc theo từng chu kì hoặc phát ra tín hiệu tìm kiếm khi nào một sự kiện xảy ra. Dạng lập trình này gọi là lập trình thủ tục, và không dùng ngắt phần cứng. Ứng dụng phải chắc chắn thăm dò cổng một cách đầy đủ để không mất bất kì một dữ liệu nào hoặc sự kiện nào. Tần số thăm dò phụ thuộc vào kích thước bộ đệm và tổng dữ liệu cần lấy(cần cho phản ứng nhanh). Ví dụ, nếu một thiết bị có 16 byte bộ đệm và dò cổng 1 lần/1 giây, thiết bị này chỉ có thể nhận không thể lớn hơn 16 byte/ 1 giây hoặc là bộ đệm sẽ bị tràn hoặc là dữ liệu sẽ bị mất.

Phương pháp thăm dò thường áp dụng cho truyền dữ liệu ngắn, đột ngột hoặc khi máy tính gửi dữ liệu và chớ đợi tín hiệu phản hồi nhanh. Một giao diện thăm dò không yêu cầu ngắt phần cứng, và bạn có thể chạy dạng lập trình này ở trên cổng mà không có đường ngắt. Nhiều giao diện thăm dò dùng ngắt timer của hệ thống để có kế hoạch đọc cổng sau một khoảng thời gian cố định.

Cơ chế chống mất dữ liệu(tiếp)

Thừa nhận(Actknowledgments):

Một vài liên kết có các nút chấp nhận mệnh lệnh mà không có một phản ứng nào, nhưng bình thường nó có ích cho nút nhận để cho bên truyền biết rằng một thông điệp
đã truyền qua, thậm chí nếu bên nhận không có một thông tin nào phản hồi. Sự thừa nhận này đặc biệt có ích trong mạng lưới, khi có nhiều nút chia sẻ cùng đường truyền thông và nơi truyền đang chuyển đổi tại thời gian không đúng có thể cản trở một thông điệp của nơi truyền khác.

Actknowledgments có thể là một byte đã được định nghĩa sẵn, như là một giá trị mà đã đồng hoá với bên nhận, hoặc là nút truyền có thể cho rằng có một nút nhận được thông điệp của nó khi nó nhận được yêu cầu dữ liệu đáp lại. Nếu nút truyền không nhận được phản ứng phản ứng mà nó yêu cầu, nút này sẽ cho rằng có một lỗi và truyền lại hoặc là làm các công việc khác.

Khi truyền tới một nút mà không có bộ đệm nhận hoặc có bộ đệm kích thước nhỏ, bên truyền có thể

dùng Actknowlegments để chắc chắn rằng sẽ có sự tham gia của nút nào đó vào quá trình truyền nhận trước khi gửi một gói dữ liệu. Nút truyền bắt đầu bằng cách gửi một byte tới tín hiệu mà nó muốn gửi dữ liệu. Khi nút nhìn thấy byte, nút này gửi một Actknowlegment và sau đó tập trung vào việc xem xét ở đầu vào nối tiếp của nó. Khi nơi nhận nhận được Actknowgment , nó biết rằng đã an toàn và yên tâm cho việc gửi dữ liệu.

Kiểm tra lỗi(Error Checking):

Bên nhận có thể dùng Error- Checking để kiểm tra rằng mọi dữ liệu đến đúng đích. Cách để kiểm tra thông điệp lỗi bao gồm gửi dữ liệu bản sao và byte kiểm tra lỗi.

Một dạng đơn giản kiểm tra lỗi đơn giản là dùng dữ liệu bản sao. Bên truyền gửi mỗi thông điệp 2 lần và bên nhận kiểm tra để xác định rằng 2 thông điệp này đều giống nhau trong cả 2 lần. Tất nhiên, điều đó có nghĩa rằng mỗi thông điệp sẽ mất gấp đôi thời gian truyền. Nó quả thật là hữu ích khi gửi một dữ liệu ngắn quan trong thỉnh thoảng, bất chợt. Nhiều điều khiển hồng ngoại dùng dạng thức này.

Mộtcách thức khác của eror-checking là gửi một byte kiểm tra lỗi cùng với dữ liệu. Checksum tính toán bằng cách thực hiện một vài phép tính toán số học hoặc logic trên byte đó trong thông điệp. Vì thế sẽ thêm một byte kiểm tra vào trong mỗi byte của thông điệp và dùng byte sau cùng để làm kết quả của sự kiểm tra.

Nơi nhận lặp đi lặp lại quá trình tính toán này, và nếu nó nhận được các kết quả khác nhau thì có nghĩa rằng nó không nhận được đúng dữ liệu đã gửi.
Một dạng khác của byte kiểm tra là CRC(cyclic redundancy code) dùng nhiều tính toán phức tạp và thực tế hơn nhiều so với checksum. Một vài giao thức phổ biến dùng trong file truyền đi là Kermit, Xmodem, Ymodem, và Zmodem.

Khi một nút phát hiện lỗi hoặc nhận được thông điệp mà nó không hiểu, nó sẽ có gắng thông báo cho nút gửi dữ liệu để báo rằng nút này có thể truyền thử lại hoặc truyền dữ liệu khác để trả lời trong hoàn cảnh này. Sau một số lần cố gắng gửi, nếu nút truyền đủ biết để giữ nút để hiển thị một thông điệp lỗi, một âm thanh báo hiệu, hoặc làm điều gì đó để cho con người biết hoạt động của lỗi và sau đó tiếp tục với trách nhiệm tốt nhất mà nó có thể làm được.

Nút nhận nên biết phải làm gì với thông điệp ngắn hơn so với mong đợi. Dù đợi mãi cho thông điệp kế thúc, nó phải có sự kiện time out và để cho nơi điều hành biết là có một lỗi. Nơi điều hành có thể gửi lại sau đó hoặc là tiếp tục. Nếu không thì mạng có thể bị treo trong sự chờ đợi kết thúc.

Giới thiệu điều khiển ActiveX MSComm

MSComm trong VB dùng điều khiển truyền thông nối tiếp. Điều khiển này có trong bản VB Proessional và Enterprise editions, nhưng không có trong phiên bản Learning editions(giá rẻ nhất). Điều khiển này dễ dàng trong lập trình và hoạt động tốt hơn các dạng truy suất cổng khác. Nếu là bản đầy đủ các bạn dễ dàng tìm được điều khiển này trong Project-> Component(Ctr-T) Chọn Microsoft Comm Control 6.0

Các đặc tính của MSComm:

Những tính chất của MSComm liên quan đến thiết lập cổng, truyền nhận dữ liệu, dùng tín hiệu bắt tay, hoặc đồng nhất các điều khiển. Các tính chất của MSComm được sắp xếp theo chức năng:

Thiết lập:

- CommID: trả lại handles đồng nhất tới thiết bị truyền thông có kiểu Long. Tính chất này không có lúc thiết kế mà chỉ có khi thi hành, thuộc tính này là ReadOnly.
- CommPort: dạng object.CommPort = value. Value là chỉ số của cổng Com có giá trị từ 1 -> 16 và mặc định có giá trị =1. Các bạn cần phải thiết lập thông số này trước khi mở cổng. Sẽ có lỗi error 68 (Device unavailable) nếu như không mở được cổng này.
- InBuferSize: thiết lập hoặc trả lại kích thước của bộ đệm nhận, tính =byte. Mặc định là 1024 byte. Các bạn không được nhầm lẫn với đặc tính InBufferCount là số byte đang chờ trong bộ đệm.
- InputLen : object.InputLen [= value] thiết lập hoặc trả lại số byte mỗi lần thuộc tính Input đọc được. Mặc định giá trị Value=0 tức là thuộc tính Input sẽ đọc hết nội dung của bộ đệm khi nó được dùng. Nếu số kí tự trong bộ đệm nhận không = InputLen thì thuộc tính Input sẽ trả lại kí tự rỗng "". Ví thế bạn cần phải chọn cách kiểm tra InBufferCount để chắc chắn số kí tự yêu cầu đã có đủ trước khi dùng lệnh .Input. Tính chất này rất là có ích khi đọc dữ liệu một máy mà dữ liệu ra được định dạng bằng các khối có chiều dài cố định.
- InputMode: object.InputMode [= value] . Value = 0 hay = comInputModeText dữ liệu nhận được dạng văn bản kiểu kí tự theo chuẩn ANSI. Dữ liệu nhận được sẽ là một sâu.
- Value=1 hay = comInputModeBinary dùng nhận mọi kiểu dữ liệu như kí tự điều khiển nhúng, kí tự NULL,.. Giá trị nhận được từ Input sẽ là một mảng kiểu Byte.
- NullDiscard: object.NullDiscard [= value] tính chất này quyết định kí tự trống có được truyền từ cổng đến bộ đệm nhận hay không. Nếu value= True kí tự này không được truyền. value = false kí tự trống sẽ được truyền. Kí tự trống được định nghía theo chuẩn ASCII là kí tự 0 – chr$(0).
- OutBuferSize: giống như InBuferSize, mặc định là 512.
- ParityReplace: thiết lập và trả lại kí tự thay thế kí tự không đúng trong lỗi giống nhau.
- PortOpen: thiết lập và trả lại tính trạng của cổng(đóng hoặc mở). object.PortOpen [= value] value = true cổng mở. =false cổng đóng và xóa toàn bộ dữ liệu trong bộ đệm nhận và truyền. Cần phải thiết lập thuộc tính CommPort đúng với tên của cổng trước khi mở cổng giao tiếp. Thêm vào đó cổng giao tiếp của thiết bị của bạn phải hỗ trợ giá trị trong thuộc tính Setting thì thiết bị của bạn mới hoạt động đúng, còn không thì nó sẽ hoạt động rất dở hơi nếu không nói là nó chạy không tốt. Đường DTR và RTS luôn giữ lại trạng thái của cổng.
- RthresHold: object.Rthreshold [= value] value kiểu số nguyên. Thiết lập số kí tự nhận được trước khi gây lên sự kiện comEvReceive. Mặc định =0 tức là không có sự kiện OnComm khi nhận được dữ liệu. Thiết lập = 1 tức là sự kiện OnComm xảy ra khi bất kì kí tự nào bị thay thế trong bộ đệm nhận.
- Settings: object.Settings [= value] thiết lập hoặc trả lại các thông số tần số baud, bít dữ liệu, bít chẵn lẻ, bít stop. Nếu Value không có giá trị khi mở sẽ gây ra lỗi 380 (Invalid property value).

  Value có dạng "BBBB,P,D,S". Trong đó, BBBB là tần số bus, P : thiết lập bít đồng bộ, D: số bít dữ liệu, S: số bít stop.
  Mặc định của nó là : "9600,N,8,1"

  Sau đây là một số tần số bus 110,300,600,1200,2400,4800,9600(mặc định), 1400,19200,28800,38400,56000,115200,128000,256000.
  Các giá trị của P: E(even), M: mark, N: none(mặc định), O: old, S: Space.
  D : có giá trị từ 4-> 8(mặc định).
  S: số bít stop có giá trị 1, 1.5, 2;
- SThreshold: thiết lập và và trả lại số kí tự nhỏ nhất được cho phép trong bộ đệm gửi để xảy ra sự kiện OnComm = comEvSend . Theo mặc định giá trị này = 0 tức là khi truyền sẽ không gây ra sự kiện OnComm. Nếu thiết lập thông số này =1 thì sự kiện OnComm xảy ra khi bộ đệm truyền rỗng. Sự kiện OnComm = comEvSend chỉ xảy ra khi mà số kí tự trong bộ đệm truyền nhỏ hơn hoặc = Sthreshold. Nếu số kí tự trong bộ đệm này luôn lớn hơn Sthreshold thì sự kiện này không thể xảy ra.

Giới thiệu điều khiển ActiveX MSComm(tiếp)

Truyền nhận dữ liệu:
+ CommEvent: trả lại phần lớn sự kiện giao tiếp hoặc có lỗi. CommEvent xảy ra khi có lỗi hoặc khi xảy ra sự kiện nào đó. Sau đây là một số hằng số lỗi:

comEventBreak 1001 A Break signal was received.
comEventFrame 1004 Framing Error. The hardware detected a framing error.
comEventOverrun 1006 Port Overrun. A character was not read from the hardware before the next character arrived and was lost.
comEventRxOver 1008 Receive Buffer Overflow. There is no room in the receive buffer.
comEventRxParity 1009 Parity Error. The hardware detected a parity error.
comEventTxFull 1010 Transmit Buffer Full. The transmit buffer was full while trying to queue a character.
comEventDCB 1011 Unexpected error retrieving Device Control Block (DCB) for the port.

Một số sự kiện :

Constant Value Description
comEvSend 1 There are fewer than Sthreshold number of characters in the transmit buffer.
comEvReceive 2 Received Rthreshold number of characters. This event is generated continuously until you use the Input property to remove the data from the receive buffer.
comEvCTS 3 Change in Clear To Send line.
comEvDSR 4 Change in Data Set Ready line. This event is only fired when DSR changes from 1 to 0.
comEvCD 5 Change in Carrier Detect line.
comEvRing 6 Ring detected. Some UARTs (universal asynchronous receiver-transmitters) may not support this event.
comEvEOF 7 End Of File (ASCII character 26) character received.

+ EOFEnable: object.EOFEnable [= value] quyết định các hành động nếu MSComm tìm thấy kí tự kết thúc file. Nếu value=true khi tìm thấy kí tự kết thúc file thì sẽ gây lên sự kiện comEvEOF trong OnCommEvent. Nếu value= false thì sẽ không gây lên sự kiện này.

+ InBufferCout: trả lại số kí tự đang có trong bộ đệm nhận Bạn có thể xoá bộ đệm nhận bằng cách đặt thuộc tính này =0 . Không nhầm với thuộc tính InBufferSize là tổng kích thước của bộ đệm nhận.

+ Input: nhận và xoá dữ liệu trong bộ đệm nhận.Nếu InputMode là comInputModeText thì giá trị trả về sẽ là một xâu tức có kiểu String , dữ liệu dạng text trong một biến kiểu Variant. Nếu InputMode = comInputModeBinary thì thuộc tính này sẽ trả lại dữ liệu dạng nhị phân dưới dạng một mảng kiểu byte trong một biến Variant.

+ OutBufferCount: trả lại số kí tự trong bộ đệm truyền.

+ Output: ghi dữ liệu vào bộ đệm truyền. có thể truyền kiểu text hoặc kiểu nhị phân. Nếu truyền bằng kiểu text thì cho một biến Variant = kiểu String, nếu truyền kiểu nhị phân thì cho cho Output= variant = một mảng kiểu Byte.
Bắt tay(handshaking):

+ Break : thiết lập hoặc xoá tín hiệu. object.Break [= value] value = true hoặc false. Khi set value= true thì thông số Break này sẽ gửi một tín hiệu break. Tín hiệu break trì hoàn việc truyền dữ liệu và đưa đường truyền vào trạng thái break tới khi mà value = false.

+ CDHolding: quết định xem sự truyền này đến đâu bằng cách truy vấn đường CD(Carrier Detect). Carrier Detect là tín hiệu gửi từ modem tới máy tính kết nối với nó thống báo rằng nó đang online. Nếu giá trị = true thì nó đường CD đang ở mức cao, nếu = false thì đường dây này đang ở mức thấp. Tính chất này không có trong lúc thiết kế chỉ có trong khi chạy chương trình.
Carrier Detect được biết như là Receive Line Signal Detect (RLSD).

+ CTSHolding: quết định khi nào bạn gửi dữ liệu bằng cách truy vấn trạng thái đường Clear To Send (CTS). Thông thường tín hiệu CTS được gửi từ modem tới máy tính kết nối với nó để báo rằng đang quá trình truyền dữ liệu. Thuộc tính Readonly chỉ xuất hiện khi chạy chương trình. Đường Clear To Send dùng trong RTS/CTS (Request To Send/Clear To Send) bắt tay phần cứng. CTSHolding cho bạn một cách để tự tay dò đường Clear To Send nếu bạn cần biết trạng thái của nó.

+ DSRHolding: biết trạng thái của đường Data Set Ready (DSR). Tín hiệu Data Set Ready truyền từ modem tới máy tính nối với nó để thông báo rằng modem đã sẵn sàng hoạt động. Tính chất này dùng khi viết Data Set Ready/Data Terminal Ready handshaking routine cho máy Data Terminal Equipment (DTE)- máy trang bị đầu cuối dữ liệu.

+ DTREnable: tính chất này quyết định khi nào cho phép đường Data Terminal Ready (DTR) trong truyền thông. Tín hiệu DTR gửi từ máy tính tới modem đẻ báo rằng máy tính sẵn sàng là nơi nhận dữ liệu. Khi DTREnable = true thì đường Data Terminal Ready set lên cao khi cổng mở, và thấp khi cổng đóng. Nếu DTREnable = false thì đường đó luôn mức thấp. Trong phần lớn trường hợp set đường Data Terminal Ready thành thấp để hang up telephone.

+ Handshaking: thiết lập và trả lại giao thức bắt tay phần cứng. object.Handshaking [= value].
Các giá trị của value:

Setting Value Description
comNone 0 (Default) No handshaking.
comXOnXOff 1 XON/XOFF handshaking.
comRTS 2 RTS/CTS (Request To Send/Clear To Send) handshaking.
comRTSXOnXOff 3 Both Request To Send and XON/XOFF handshaking.

Handshaking chỉ là giao thức truyền thông nội tại quyết định bởi dữ liệu nào được truyền từ cổng phần cứng tới bộ đệm nhận. Khi kí tự của dữ liệu tới cổng nối tiếp, thiết bị truyền thông sẽ chuyển nó vào trong bộ đệm nhận và chương trình của bạn có thể đọc chúng. Nếu không có bộ đệm dữ liệu hoặc chương trình của bạn cần đọc kí tự trực tiếp từ phần cứng , bạn có thể mất dữ liệu bởi vì kí tự từ phần cứng đến rất nhanh. Giao thức Handshaking đảm bảo dữ liệu không bị mất, khi dữ liệu đến cổng quá nhanh thì thiết bị truyền thông sẽ chuyển dữ liệu vào trong bộ đệm nhận.

+ RTSEnable: quết định khi nào cho phép đường Request To Send (RTS), Tín hiệu RTS từ máy tính tới modem để yêu cầu được tryền dữ liệu. Khi RTSEnable = true thì đường RTS mức cao khi cổng mở, tích mức thấp khi cổng đóng. Và hiển nhiên khi RTSEnable thì đường RTS luôn mức thấp.RTS dùng trong RTS/CTS hardware handshaking. RTSEnable cho phép bạn dò đường RTS khi cần biết tình trạng của đường này.

Các tính chất trên không có lúc thiết kế giao diện mà chỉ có lúc chạy chương trình (dùng trong viết code).

Ngoài ra còn có các thuộc tính khác như với các loại điều khiển khác:
Index: thiết lập và trả về một số xác định thứ tự nếu form bạn có nhiều điều khiển như thế này.
, Name: tên điều khiển, Object, Parent: trả về form hoặc đối tượng mà điều khiển này nằm trên đó,
Tag: thiết lập và trả về một biểu thức. Người dùng định nghĩa

Chắc các bạn đã có phần nào hiểu biết về điều khiển ActiveX này.

Truyền dữ liệu kiểu text và nhị phân trong VB

Chào các bạn hôm nay tôi lại tiếp tục giới thiệu cho các bạn hiểu về cách truyền, nhận dữ liệu dạng text và binary trong VB.

Trước hết, các bạn cần biết biến kiểu Variant(tra trong từ điển có nghĩa là tương tự, gần giống nhau). Do vậy mà một biến Variant có thể gán = bất kì kiểu gì cũng được.Sau đây là chi tiết về 2 vấn đề chính:

VB cho phép bạn truyền dữ liệu dạng Text hay là dạng Binary. Thuộc tính InputMode quyết định điều khiển MSComm dùng dạng nào.

1.Kiểu văn bản(Text):

Với thuộc tính InputMode = comInputModeText thì MSComm sẽ gửi và nhận dữ liệu dạng xâu theo chuẩn ANSI (không phải chuẩn ASCII). Để gửi một xâu ra port, bạn cần phải cho thuộc tính Output của MSComm = 1 xâu. Ví dụ:

Code:

```
        Dim SampleText as String

        'ví dụ bạn muốn truyền một xâu "ABC"

        SampleText = "ABC"

        ' gửi kí tự này ra cổng

        MSComm1.Output = SampleText
```

MSComm gửi một mã ANSI 8 bít cho mỗi kí tự trong xâu mà bạn gửi. Để đọc một xâu từ cổng, cần đặt một xâu = thuộc tính Input của MSComm. Ví dụ bạn muốn đọc dữ liệu từ cổng và ghi vào một biến SampleText có kiểu String:

Code:

```
Dim SampleText as String

SampleText = MSComm1.Input ' khi đó SampleText sẽ là dữ liệu đọc được

MSComm lưu trữ mỗi mã ANSI 8 bít như một kí tự văn bản.
```

Thực tế như các bạn đã biết thì giá trị truyền cho MSComm1.Output phải là kiểu Variant. Ở đây thuộc tính Output chấp nhận kiểu một biến Variant chứa một xâu kí tự và MSComm sẽ đọc xâu kí tự và gán tự động vào một biến Variant vì Variant chính là kiểu của Output. Nói cách khác ở đây có sự chuyển kiểu ngầm định giữa kiểu String sang kiểu Variant.

Ngay ở bên trong bản thân chương trình VB lại lưu trữ xâu dưới dạng mã Unicode 16 bít nhưng sự chuyển đổi giữa kiểu Unicode và kiểu xâu kí tự ANSI 8 bít của MSComm diễn ra một cách tự động.

Sự chuyển kiểu của số ASCII Hex:

Số ASCII Hex là số hexa bình thường mà ta vẫn dùng như 0xA5(trong C,C++) hoặc 0A5h(trong ASM,..) đại diện cho số 165 trong hệ Decimal(165 = 16*10+ 5).

Với các ứng dụng dùng định dạng ASCII Hex, VB có một hàm chuyển đổi giữa kiểu xâu ASCII Hex và giá trị mà nó đại diện. Toán tử Hex$ chuyển đổi một số sang dạng kí tự ASCII Hex:

Ví dụ, để kiểm tra bạn có thể dùng hàm rất đơn giản xem nó in ra thế nào :
Code:

```
debug.print Hex$(165)
```

thì kết quả sẽ hiện trên một Dialog là : A5
Toán tử Val chuyển đổi từ kiểu ASCII Hex sang kiểu giá trị của xâu đó:
Ví dụ: ta thấy 0xA5 = 165 để thử xem có đúng không dùng lệnh;
Code:

```
   debug.print Val("&h" & "A5")
```

Kết quả là 165.

Xâu đầu tiên "&h" được thêm vào để báo cho VB biết để đối xử với giá trị đưa ra sau đó như là một số hexadecimal.'

2. Kiểu nhị phân(Binary Mode):

Để truyền dữ liệu dưới dạng nhị phân, cần thiết lập thuộc tính InputMode của MSComm thành comInputModeBinary.

VB cung cấp một kiểu dữ liệu kiểu Byte để lưu trữ dữ liệu nhị phân. Các byte được ghi và đọc từ cổng nối tiếp được lưu trữ trong một biến Variant(nội dung của nó chứa một mảng mà các phần tử của mảng có kiểu Byte). Thậm chí nếu chí đọc, ghi duy nhất có 1 byte thì dữ liệu này cũng phải đặt trong một mảng byte, chứ không được lưu trữ trong một biến kiểu byte thông thường. Để ghi một mảng kiểu byte ra cổng nối tiếp gồm 2 bước.

+ Bước đầu: lưu trữ mảng kiểu byte vào một biến variant

+ Bước 2: gửi dữ liệu đi bằng cách thiết lập thông số Output của MSComm bằng biến Variant đó.

Code:

```
        Dim BytesToSend(0 to 1) as Byte ' khai báo một mảng 2 phần tủ

        Dim Buffer as Variant
```

```
        ' lưu trữ dữ liệu vào mảng kiểu byte ở trên

        BytesToSend(0) = &H4A

        BytesToSend(1) = &H23

        ' cho vào một biến Variant

        Buffer = BytesToSend()

        'ghi vào cổng nối tiếp

        MSComm1.Output = Buffer
```

Để đọc các byte tại cổng nối tiếp, bạn cũng làm tương tự như trên, đọc vào một biến Variant sau đó cho một mảng = biến đó.

Code:

```
        Dim BytesReceived()  as Byte ' khai báo một mảng động

        Dim Buffer as Variant ' khai báo biến variant

        'đọc dữ liệu từ cổng nối tiếp

        Buffer = MSComm1.Input

        ' ghi dữ liệu đọc được vào mảng động

        BytesReceived() = Buffer
```

Các ban lưu ý là phải khai báo một mảng byte động. Có 2 cách để chuyển đổi giữ mảng bytes và kiểu Variant. Bạn có thể cho một biến = một biến có số chiều đã được biết và VB làm công việc chuyển đổi này tự động:

Code:

```
Dim DimensionedArray(15) as Byte ' mảng đã khai báo số chiều =15

Dim DynamicBytesArray()  as Byte

Dim Buffer As Variant

Dim Count As Integer

' lưu trữ một mảng mảng vào một biến variant. Mảng này đã được biết số phần
tử

Buffer = DimensionedArray()



'để sao chép nội dung của một biến variant vào một mảng thì mảng này phải
khai báo là một mảng động( chưa biết số phần tử)
```

```
DynamicBytesArray() = Buffer
```

Đối với VB 6.0 bạn hoàn toàn có thể gán 2 mảng với nhau vì nó sẽ tự sao chép nội dung từ mảng nguồn sang mảng đích mà không cần phải làm bằng cách sao chép từng phần tử của 2 mảng cho nhau(như trong C thì bạn phải làm điều này rồi vì gán 2 tên mảng thực chất là bạn chỉ là cho con trỏ mảng đích trỏ vào địa chỉ của phần tử của mảng nguồn thôi, đây là sự sao chép bề mặt). Tuy nhiên bạn vẫn có thể làm điều này trong VB:

Code:

```
'lưu trữ một mảng kiểu byte trong một biến variant

Buffer = CVar(DynamicByteArray())

' CVar   -> Convert to Variant  Chuyển thành kiểu variant




'lưu nội dung của biến variant này trong một mảng kiểu byte

For Count = 0 to (LenB(Buffer)-1)

        DimmensionedArray(Count) = CByte(Buffer(count))

Next Count

' CByte -> Convert to Byte chuyển kiểu thành kiểu Byte
```

Gửi nhận dữ liệu bằng phương pháp dò

Tiếp theo tôi xin giới thiệu cho các bạn phương pháp lấy dữ liệu bằng phương pháp thăm dò(polling). Giao tiếp tại cổng bằng phương pháp dò tức là bạn chỉ đọc hoặc ghi ra cổng khi nào cần bằng cách dùng thuộc tính Input hoặc Output của MSComm.

1. Gửi dữ liệu:

Thuộc tính Output dùng để ghi dữ liệu ra cổng. Biến dùng ở bên phải cú pháp là một biến kiểu Variant. Đây là cú pháp để ghi dữ liệu:

Code:

```
        Dim DataToWrite As Variant

        MSComm1.Output = DataToWrite
```

Khi gửi một khối nhỏ dữ liệu, cần phải thiết lập thuộc tính OutBuferSize phải lớn hơn hoặc bằng số lượng lớn nhất các byte mà các bạn cần chuyển trong một lần.

Đối với việc truyền dữ liệu có tính lâu dài về thời gian dùng OutBufferCount để chắc chắn rằng bộ đệm không bị tràn. Khi bạn có nhiều dữ liệu cần gửi để tránh cho tràn bộ đệm , bạn nên đọc giá trị của OutBufferCount và so sánh với giá trị của OutBufferCount để kiểm tra xem bộ đệm còn bao nhiêu sau khi gửi dữ liệu đầu tiên. Sau đó làm đầy bộ đệm bằng cách ghi bằng đó các byte hoặc nhở hơn dữ liệu vào bộ đệm thì bộ đệm sẽ không bị tràn. Hoặc bạn có thể gửi dữ liệu đã đóng gói với kích thước xác định và

chỉ gửi các gói này được OutBufferCount chỉ rằng có đủ chỗ trống trong bộ đệm cho gói dữ liệu này. Ví dụ, OutBufferSize = 1024 và kích thước 1 gói là 512, bạn chỉ có thể gửi được gói dữ liệu này khi mà OutBuferCount <= 512.

2. Nhận dữ liệu:

Để đọc dữ liệu được truyền đến, ứng dụng đọc dữ liệu từ InBuferCount theo từng chu kì. Khi bộ đếm chỉ rằng một số các kí tự mà ứng dụng cần đã đến(như muốn lấy 5 byte chẳng hạn) thì ứn dụng sẽ đọc dữ liệu với thuộc tính Input của MSComm:

Code:

```
Dim BytesToRead As Integer

Dim DataIn As Variant



'thiết lập số byte cần đọc

NumberOfBytesToRead = 512

MSComm1.InputLen = NumberOfBytesToRead



' dò bộ đệm nhận đến khi bộ đệm có đầy đủ số byte cần đọc

Do

        DoEvents

Loop Until MSComm1.InBufferCount> NumberOfBytesToRead

' khi tổng số byte đã tới thì đọc lưu vào DataIn

DataIn = MSComm1.Input
```

Thuộc tính InBuferSize phải đủ độ rộng để cho lượng lớn nhất dữ liệu có thể tới mà không bị mất trước khi MSComm có thể đọc chúng. Nếu dữ liệu đến bằng các block với kích thước cố định thì cần thiết lập thuộc tính InBufferSize bằng bội số của kích thước 1 block.

Nếu tổng dữ liệu đến không biết kích thước thế nào, ứng dụng nên đọc bộ đệm nhận ngay khi bộ đệm chỉ nhận được 1 byte để tránh việc không kiểm soát đượcbộ đệm gây ra tràn dữ liệu. Chờ đợi nhiều byte để đọc là một việc làm không có hiệu quả bởi vì không có cách nào biết được byte nào sẽ đến cuối cùng. Nếu chờ nhiều hơn 1 byte rồi mới đọc, chương trình nên bao gồm có một "time out" chính là tổng thời gian từ lúc chờ mà tổng số byte vẫn không đến(như bạn chờ 6 byte mà mãi không đến chẳng lẽ ứng dụng chờ mấy giờ à, thế thì bạn cần phải qui định là sao bao nhiêu lâu thì đọc thôi chẳng cần chờ nữa).

Bạn có thể kết hợp phương pháp lập trình theo thủ tục và theo sự kiện bằng cách sử dụng timer để biết khi nào thì đọc cổng. Ví dụ, dùng một sự kiện Timer gây ra ở cổng đeer đọc cổng một lần / một giây.

Trên đây là cách đọc, ghi dữ liệu bằng phương pháp dò. Ngày mai tôi xin giới thiệu cho các bạn cách dùng ngắt(Interrupt), tức là dùng sự kiện OnComm của MSComm.

Phương pháp dùng ngắt (Interupt) dùng OnComm của MSComm

Tiếp theo tôi xin giới thiệu với các bạn phương pháp thứ 2 là dùng ngắt(interrupt). Ở đây dùng chương trình con OnComm của MSComm. OnComm dùng để đáp ứng lại một vài sự kiện được gây ra tại cổng. Dùng OnComm chúng ta có thể kiểm tra dữ liệu được gửi đến hoặc gửi đi. Khi một trong 17 sự kiện gây ra tại cổng, ứng dụng sẽ tự động gây ra sự kiện ComEvent hay nói cách khác là gây ra một ngắt tại cổng và nhảy đến chương trình con OnComm của điều khiển ActiveX MSComm. Cách hiệu quả nhất để biết được sự kiện nào xảy ra là dùng Switch.. Case . Để đáp ứng một sự kiện xảy ra bạn có thể thêm các đoạn mã để thực hiện đọc, ghi dữ liệu trên cổng hoặc đơn giản chẳng làm gì cả. Phần lớn sự kiện của OnComm đáp ứng trực tiếp các sự kiện phần cứng được báo hiệu bởi các bit trong UART. Các sự kiện khác gây ra bởi phần mềm.

1. Nhận dữ liệu:

Trong chiều nhận, OnComm cung cấp một cách đơn giản và hiệu quả để phát hiện khi nào dữ liệu được nhận tại cổng. Thuộc tính Rtheshold của MSComm quyết định số kí tự ít nhất được nhận để gây ra sự kiện OnComm:

Code:

```
MSComm1.Rthreshold =1
```

Khi bộ đệm nhận bao gồm 1 hoặc nhiều kí tự, ứng dụng sẽ nhẩy đến chương trình con OnComm. Bảng ngắt có trách nhiệm phát hiện sự kiện ComEvReceive xảy ra và đọc kí tự trong bộ đệm:
Code:

```
                Case ComEvReceive

                Buffer = MSComm1.Input
```

Nếu bạn thiết lập thuộc tính RThreshold >1 thì những byte đến cuối cùng hoặc có rất là nhiều byte bạn vẫn không đọc được(ví dụ bạn set =2, nếu bộ đệm nhận được 1 byte thì không có sự kiện nhận OnComm nên không chạy đoạn mã trên). Nên tốt nhất là sét thuộc tính này =1 nếu nhận được là biết ngay.

Thiết lập thuộc tính RThreshold = 0 thì sẽ vô hiệu hoá sự kiện ComEvReceive tức không dùng ngắt, và sẽ không có sự kiện nhận xảy ra.

2. Gửi dữ liệu:

Trong chiều gửi (send), bạn nên dùng sự kiện OnComm để kiểm soát dữ liệu gửi đi.

CPU có thể ghi ra cổng với tốc độ nhanh hơn tốc độ của cổng gửi dữ liệu nối tiếp ra ngoài(cho các thiết bị ngoại vi, vđk,..). Nếu tổng dung lượng của dữ liệu ghi ra cổng lớn hơn OutBufferSize thì ứng dụng không thể ghi tất cả dữ liệu vào bộ đệm một lần được, nếu cứ ghi vào thì bộ đệm sẽ bị tràn. Bạn có thể dùng OnComm để truyền đi một dung lượng lớn dữ liệu một cách có hiệu hiệu quả bằng cách luôn luôn phải đảm bảo cho bộ đệm truyền không bao giờ rỗng hoặc bị tràn.

Thuộc tính SThreshold quyết định số kí tự trong bộ đệm gửi sẽ gây ra sự kiện OnComm. Ví dụ:
Code:

```
MSComm1.SThreshold = 256
```

Cách truyền như sau: trong lần truyền đầu tiên dùng thuộc tính Output để điền đầy bộ đệm:
Code:

```
MSComm1.Output = DataToSend
```

Khi số kí tự trong bộ đệm giảm từ (SThreshold +1) xuống SThreshold thì sẽ gây ra sự kiện comEvSend và nhảy đến chương trình con OnComm. Dựa vào sự kiện comEvSend chúng ta lại truyền tiếp dữ liệu, cứ thế một dữ liệu dung lượng lớn sẽ được truyền đi.

Ví dụ chúng ta có bộ đệm truyền = 1024 byte mà chúng ta cần truyền một file có dung lượng 32k, ứng dụng có thể thiết lập SThreshold =256. Để gửi chúng ta làm như sau: lần đầu tiên ghi ra bộ đệm 1024 byte để làm đầy bộ đệm truyền và cổng bắt đầu gửi dữ liệu đi từng byte theo kiểu truyền từng bít một.Do đó số byte trong bộ đệm sẽ giảm dần. Khi trong bộ đệm vẫn còn 256 byte nữa thì sẽ gây ra sự kiện comEvSend và trong chương trình con MSComm ta có thể gửi tiếp một lượng dữ liệu mới = 1024-256 = 768 byte để làm đầy lại bộ đệm. Khi đó quá trình lại diễn ra liên tục đến khi ta gửi hết file đó thì thôi.

Nếu tổng dung lượng của thông tin cần gửi nhỏ hơn OutBufferSize thì không cần dùng OnComm trong trường hợp này. Ứng dụng có thể ghi trực tiếp dữ liệu vào bộ đệm nhận. Đó là cách đơn giản nhất nếu hệ thống có thể dành nhiều bộ nhớ cho bộ đệm. Nếu dành được nhiều thì chúng ta cứ thoải mái tăng bộ đệm lên cho đơn giản. Thiết lập SThreshold = 0 để vô hiệu hoá sự kiện comEvSend và khi đó sẽ không có sự kiện gửi xẩy ra.

Như xem ở trên các bạn tưởng rằng dùng sự kiện comEvSend là có ích, cần thiết cho bạn nhưng thực tế thì nó rất là phức tạp, và không nên dùng. Ví dụ, nếu vì một vài lí do mà CPU làm đầy bộ đệm với tốc độ chậm hơn tốc độ byte gửi đi từ cổng thì bộ đệm sẽ không bao giờ giữ được SThreshold byte trong bộ đệm và do đó sẽ không bao giờ xây ra sự kiện này vì sự kiện này chỉ xảy ra khi mà có sự thay đổi số byte trong bộ đệm giảm qua mức SThreshold byte. Trong trường hợp này thì các bạn nên dùng phương pháp thăm dò vì phương pháp thăm dò vì phương pháp thăm dò hiệu quả và thực tế hơn trong việc gửi dữ liệu.

Còn có rất nhiều sự kiện nữa như comEvCTS, comEvDSR,comEvCD, comEventRing, comEvEof, comEventBreak trong sự kiện bắt tay (Handshaking Events),Error Event nhưng tôi không nêu ra ở đây nữa khi nào dùng những sự kiện này thì tôi sẽ nói rõ hơn cho các bạn.

Trên đây là phương pháp dùng ngắt. Hẳn các bạn đã có được sự so sánh để lựa chọn cho mình đúng không. Vậy thì bây giờ tôi xin rút lại như sau: gửi dữ liệu thì nên = phương pháp dò, nhận dữ liệu có thể = ngắt hoặc = thăm dò.
Qua các bài ở trên chắc các bạn đã có một kiến thức chắc chắn hiểu về giao thức, cách truyền dữ liệu, tín hiệu bắt tay,.. trong truyền thông nối tiếp dùng cổng Com.

Còn nhiều vấn đề chúng ta cần quan tâm hơn.Tôi sẽ hướng dẫn các bạn lập trình từ những dạng rất đơn giản đến phức tạp như xác định cổng, ..Do vấn đề này rất hay và hơi dài nên tôi hẹn các bạn 2 hôm nữa(chủ nhật).

Chúc ngày cuối tuần vui vẻ.

gửi ChiBang

Thứ nhất, không hề có byte start và byte stop chỉ có bit start và stop thôi.
1-Nếu dùng ngắt để nhận dữ liệu ta đặt:MSComm1.SThreshold =1. Nêu truyền tốc độ cao rất dễ dẫn tới khi đọc thì số byte trong bộ đệm >1===> liệu có gây ra sai sót gì ko? giải pháp khắc phục.

SThreshold là tham số cho truyền dữ liệu mà bạn có liên quan với nhận đâu vì 2 bộ đệm truyền nhận là khác nhau.
Nếu bạn đánh nhầm là RThreshold = 1 .nếu thế thì bạn chưa biết về cách thức nhận dữ liệu bằng sử dụng ngắt của VB rồi. Bạn đọc lại bài ở trên về cách truyền nhận dữ liệu bằng ngắt đó. Tôi xin trả lời bạn chi tiết như sau: khi đặt SThreshold =1 tức là khi trong bộ đệm nhận có sự thay đổi số byte trong bộ đệm từ rỗng(empty) sang bắt đầu có dữ liệu (1 byte chuyển đến) và dữ liệu vẫn tiếp tục chuyển đến. Vì có sự chuyển đổi này gây ra sự kiện comEvReceive và nhảy đến thực hiện đoạn code viết trong hàm OnComm và trong hàm OnComm tất nhiên có mã để đọc nội dung của bộ đệm nhận này và làm bộ đệm lại bị rỗng. Do đó nếu lại nhận được dữ liệu nữa thì nó sẽ lại lặp lại quá trình này. Nếu khi đọc số byte trong bộ đệm >1 thì như trên tất nhiên nó cũng đọc hết và lại tiếp tục.. Bạn nên biết là tốc độ đọc của bus dữ liệu qua UART là rất lớn so với tốc độ truyền nhận của cáp qua RS232 hay RS485. Do đó chẳng có thiếu sót gì đâu .
2-Theo em:MSComm1.SThreshold =1 sẽ gây lãng phí cho thời gian của PC nhất là khi thiết lập ở tốc độ cao. Thông thường thiết bị ngoại vi ta biết trước về số byte tryền liền một lúc và giả sử = N. Nó sẽ truyền theo mô hình:
+Truyền hết N byte.
+Delay()
+Truyền tiếp N byte.
....
Giả sử delay=100 ms thì:
+Truyền hết N byte.
+Trễ 100 ms.
+Truyền tiếp N byte.
....
Vậy một giải pháp tối ưu là đặt:MSComm1.SThreshold =N. Thời gian delay của bên truyền sẽ khiến cho bên PC kịp thời xử lý. Nhưng nó sẽ có thể gặp phải một trường hợp xấu có thể xảy ra: bên phát truyền liên tục N byte nhưng bên thu chỉ nhận được <N byte(do đứt dây, do nhiễu, do thời điểm kết nối thì đúng lúc bên phát truyền dở...). Thời điểm này sẽ gây lỗi từ thời điểm đó về sau. Vậy một giải phát đưa ra dùng "timerout" giải quyết vấn đề này. Em đề cử một cách bác xem có chấp nhận được ko?
Bên PC: tại ngắt(nghĩa là nó nhận được N byte), đọc bộ đệm, khởi tạo một timer=100/2=50ms.
Hàm ngắt đọc dữ liệu viết như sau:

```
Private Sub MSComm1_OnComm()
On Error Resume Next
if(MSComm1.InBufferCount>=N) then
data = MSComm1.Input
end if
tmer1.interval=50
timer1.enable=true
End Sub
```

Hàm ngắt timer này viết như sau:

```
Private Sub Timer1_Timer()
Dim tam as string
tam=MSComm1.input 'Lam rong bo dem
timer1.enable=False 'Cam timer
End Sub
```

Giải pháp này ổn định nhưng ko vạn năng, nó chỉ áp dụng tốt khi thời gian truyền<delay(). Nên áp dụng cho trường hợp thỉnh thoảng truyền.

Vậy bác có giải pháp nào dùng ngắt và giải quyết được các nhước điểm trên ko? vì em sợ cái timer của Win lắm.

Bạn có lẽ lại đánh nhầm RThreshold thành SThreshold rồi.
Nếu thế thì Chi Bang chưa đọc kĩ bài dùng ngắt ở trên rồi. Bạn đọc lại về ý nghĩa của RThreshold khi thiết lập > 0 và khi nào xảy ra ngắt comEvSend đi và một vấn đề gặp phải khi sử dụng ngắt trong truyền dữ liệu nha. Bạn chẳng chịu đọc gì cả mà đã hỏi. Với lại sao bạn lại nghĩ đặt RThreshold = 1 là lãng phí thời gian cho hệ thống? . Nếu như tốc độ truyền của bus trong PC vẫn lớn hơn truyền qua cáp (khi không có trục trặc bất thường nào) thì chính dùng ngắt thì mới là cách tiết kiệm thời gian cho PC nhất,tránh bị tràn bộ đệm nhất đấy, còn dùng thăm dò trong trường hợp này lại gây lãng phí lớn. Nhưng đề đề phòng bất trắc thì không nên dùng ngắt mà nên dùng dò khi truyền.
Với lại bên nhận dùng ngắt có thể nhận mọi dữ liệu chẳng hề gì đâu bạn ạ. Tốc độ OK đi. Với tần số bus của hệ thống 9600 bit/s thì truyền được 96
0 byte/ s với giao thức truyền 8,N,1. Do vậy chẳng cần delay(100) gì cả.
Còn về chương trình của bạn. Khi ví dụ thiết lập Rthreshold= N rồi thì khi nào nó nhận được đủ N byte chuyển sang byte thứ N+1 thì mới có ngắt thì dùng if làm gì. Với lại bạn biết OnComm đang phục vụ ngắt nào. Có tới 14 ngắt tất cả cơ mà. Giả sử có lỗi khi truyền có ngắt khác thì sao?
Với lại ở đây khi biết thiết bị ngoại vi có N byte truyền trong mỗi lần tức là truyền theo từng block một thì xử lí lại quá là đơn giản rồi. Bạn nên hiểu Timer chỉ dùng để tạo được thời gian sau mỗi đoạn thời gian ngắn thì sẽ thăm dò bộ đệm nhận thôi. Dùng vào tạo delay phí quá!. Bạn đọc lại về cách thiết lập thời gian như thế nào để không bị mất dữ liệu, dựa vào tốc độ bus,.. Còn ưu nhược điểm của ngắt thì tôi phân tích ở trên rồi. Bạn tự đọc đi.
Chết thật, bạn chẳng chịu đọc gì cả thế?
[quote]
4-Liệu MSComm có một tính năng như sau ko?
+Có quy định được thời gian giữa 2 byte truyền? ví dụ em muốn truyền chữ chuỗi "sjahs" và mỗi byte cách nhau thời gian là 2 bit chẳng hạn? Ví dụ tốc độ 11500kbps.
Ví dụ: "s" --(delay 0.05ms) --"j"-----(delay 0.05ms)----"a"--...
[quote]
Có thể bạn ạ. Dựa vào tần số bus là có thể quyết định được tốc độ truyền nhưng giới hạn điều này do tốc độ bus còn phụ thuộc vào hệ thống của bạn, vào bắt tay giữa PC và các thiết bị khác.Khi chúng ta đã vào quá trình truyền dữ liệu, xong xuôi bắt tay,.. thì tốc độ này là cố định. Chúng phải có cùng 1 tốc độ(đọc lại về truyền nhận dữ liệu nối tiếp). Ví dụ với tốc độ 9600 bps và kiểu định dạng dữ liệu 8,N,1 thì sẽ truyền được 960byte/s từ đó bạn tỉnh ra truyền 1 bit hết bao nhiêu.9600 bit/s đấy.
5-Liệu MSComm có một tính năng như sau ko?
+Nếu mà có một sự kiện như sau thì tốt quá: ví dụ vi điều khiển truyền về PC một chuỗi theo quy luật sau: byte1-----byte2....--delay ---byteN--delay(1ms)-- byte1---byte2...--delay ---byteN...

Không có điều đó khi mà bạn truyền một lượng dữ liệu lớn, tốc độ truyền mỗi byte phụ thuộc vào hệ thống của bạn(tần số,..). Cái đó chỉ làm được khi bạn viết chương trình điều khiển bằng phần mềm đơn giản thôi là đầu tiên bạn truyền N byte sau đó delay(1ms) thì tất nhiên cái RS-232 của bạn cũng sẽ delay thời gian tương tự. Tương tự như trên sau khi thiết lập mọi thông số tín hiệu bắt tay,.. thì bạn không thể điều chỉnh cái này được.
Thế nhé. Không biết bạn hài lòng chưa.

Lập trình trên VB:

OK cảm ơn ChiBang. Thực ra tôi lập trình cũng sử dụng hàm API mà bạn.
Do hôm qua luyện chưởng sai nên không làm xong theo kế hoạch để anh em chiêm ngưỡng. Có lẽ mai hoặc ngày kia luyện công xong đã.
To CHIBANG: có thể thiết lập thời gian truyền mỗi byte và thời gian truyền mỗi bít đó bạn ạ. Với lại sau khi truyền xong một byte thì có thể dùng trễ để truyền hết byte đó cho vdk nhận xong lại truyền tiếp. ở dây dùng timeout. CHIBANG và mọi người đọc bài dưới đây nha.
Các bạn có thể biết được các hàm API về cách khai báo và một số hằng số nữa. Theo hướng dẫn hình

dưới và mở trình API Text Viewer đó lên, vào file mở load file win32api.txt rồi các bạn sẽ đọc được các kiểu khai báo ở đây.
Trước hết, trong ứng dụng này sử dụng 2 điều khiển Timer là tên là tmrTransferInterval và tmrTimeout , và 2 ActiveX là Microsoft Comm Control 6.0 và Microsoft Common Dialog 6.0 .
Trước hết tôi xin phân tích một vài điều:
* Tìm cổng:
Ứng dụng dùng MSComm không phải biết bất cứ điều gì về cổng giao tiếp mà nó sử dụng ngoại trừ tên của cổng đó, như là COM1, COM2,.. Trong nhiều trường hợp ứng dụng không phải biết địa chỉ của cổng hoặc đường IRQ bởi vì MSComm tự làm chi tiết hết cả. Nhưng ứng dụng cũng cần phải biết là có những cổng nào tồn tài trên máy tính của bạn chứ, khi đó nó sẽ cho người dùng lựa chọn cổng để giao tiếp.
Có 2 cách để tìm các cổng này:
+Hàm FindPorts phát hiện cổng = cách thử mở tất cả các cổng từ COM1 đến COM16(vì chỉ có tối đa bằng này cổng thôi). Nếu cổng đã được mở hoặc trả về lỗi "Port In Use" (lỗi 8005) thì tức là cổng này có tồn tại và thêm vào mảng CommPort.

Code:

```
Public Sub FindPorts()

'Find Comm ports by trying to open each.

'Each port must support the current settings (bit rate, etc.).

Dim Count As Integer

Dim NumberOfPorts As Integer

Dim SavedPortNumber As Integer

Dim SaveCurrentPort As Boolean

ReDim CommPorts(1 To 16)

On Error Resume Next

SaveCurrentPort = False

NumberOfPorts = 0

'If a port is already open, reopen it on exiting.

If frmMain.MSComm1.PortOpen = True Then

        frmMain.MSComm1.PortOpen = False

        SavedPortNumber = PortNumber

        SaveCurrentPort = True

End If

For Count = 1 To 16

    frmMain.MSComm1.CommPort = Count
```

```
    frmMain.MSComm1.PortOpen = True
    If Err.Number = 8005 Then
        'The port is already open
        'The port exists, so add it to the list.
        NumberOfPorts = NumberOfPorts + 1
        CommPorts(NumberOfPorts) = "COM" & CStr(Count)
    ElseIf frmMain.MSComm1.PortOpen = True Then
        'If the port opens, it exists.
        'Close it and add to the list.
        frmMain.MSComm1.PortOpen = False
        NumberOfPorts = NumberOfPorts + 1
        CommPorts(NumberOfPorts) = "COM" & CStr(Count)
    Err.Clear
    End If
Next Count
'Disable the error handler
On Error GoTo 0
ReDim Preserve CommPorts(1 To NumberOfPorts)
If SaveCurrentPort = True Then
    PortNumber = SavedPortNumber
    frmMain.MSComm1.CommPort = PortNumber
    frmMain.MSComm1.PortOpen = True
End If
End Sub
' ------------------
```

+ Hàm fncGetHeighestComPortNumber dùng theo cách khác là gọi hàm API EscapeComFunction để tìm chỉ số lớn nhất tên của cổng ở trong máy của bạn.
Code:

```
===============
```

```
Public Function fncGetHighestComPortNumber() As Integer

'Returns the number of the system's highest Com port.

'Also shows how to use the EscapeCommFunction API call.

Dim ClosePortOnExit As Boolean

Dim PortCount As Long

Dim handle As Long

'The API call requires a CommID of an open port.

If frmMain.MSComm1.PortOpen = False Then

    frmMain.MSComm1.PortOpen = True

    ClosePortOnExit = True

Else

    ClosePortOnExit = False

End If

handle = frmMain.MSComm1.CommID

PortCount = GETMAXCOM

'Add 1 because EscapeCommFunction begins counting at 0.

fncGetHighestComPortNumber = _

    EscapeCommFunction(handle, PortCount) + 1

If ClosePortOnExit = True Then

    frmMain.MSComm1.PortOpen = False

End If

End Function

-----------------------
```

* việc dùng trễ:
- Ứng dụng dùng MSComm có thể dùng nhiều hàm trễ sau để quyết định khi nào quá trình truyền nhận là kết thúc và khi nào thì thoát khỏi trạng thái chờ một đáp ứng từ một máy tính khác.
+ Hàm Delay có tham số dạng mili giây, dùng hàm API timeGetTime trong thư việc .dll. Hàm này tạo trễ có giá trị là từng mili giây do hàm timeGetTime trả về số giây mà hệ điều hành bắt đầu khởi động. Và nó bị tràn sau 24 ngày nếu các bạn cứ để PC của bạn cú chạy suốt. Nếu chạy suốt thì sau 24 ngày phải kiểm tra lại các thông số này không có sai thì chết.
Code:

```
------------------------

Public Sub Delay(DelayInMilliseconds As Single)

'Delay timer with approximately 1-msec. resolution.

'Uses the API function timeGetTime.

'Rolls over 24 days after the last Windows startup.

Dim Timeout As Single

Timeout = DelayInMilliseconds + timeGetTime()

Do Until timeGetTime() >= Timeout

    DoEvents

Loop

End Sub

-----------------------------
```

+Hàm LowResDelay dùng thời gian của hệ thống để gây trễ. Thời gian này có chu kì hoạt động là 18 chu kì/ giây nên trễ mỗi chu kì khoảng 55 mili giây. Dùng hàm Timer, hàm này trả về số mili giây tính từ nửa đêm hay nói cách khác tính như giờ hiện hành trên máy của bạn nhưng đổi ra mili giây.
Code:

```
---------------

Public Sub LowResDelay(DelayInMilliseconds As Single)

'Uses the system timer, with resolution of about 56 milliseconds.

Dim Timeout As Single

'Add the delay to the current time.

Timeout = Timer + DelayInMilliseconds / 1000

If Timeout > 86399 Then

    'If the end of the delay spans midnight,

    'subtract 24 hrs. from the Timeout count:

    Timeout = Timeout - 86399

    'and wait for midnight:

    Do Until Timer < 100

        DoEvents
```

```
    Loop

End If

'Wait for the Timeout count.

Do Until Timer >= Timeout

    DoEvents

Loop

End Sub

------------------------
```

+ Trễ 1 byte hàm fncOneByteDelay tính toán thời gian để truyền hết 1 byte. Nó có ích cho một vài liên kết kiểu halp-duplex, nơi mà máy tính phải chuyển hướng truyền sau khi gửi dữ liệu để cho máy tính khác tiếp tục truyền. Trễ này để cho chắc chắc mọi bít trong byte truyền được hết. Hàm vbSetCommTimeouts dùng hàm này.(nhân đây gửi ChiBang là câu hỏi thứ 4 của bạn có thể thực hiện được).
Code:

```
---------------------------

Public Function fncOneByteDelay(BitRate As Long) As Single

'Calculate the time in milliseconds to transmit

'8 bits + 1 Start & 1 Stop bit.

Dim DelayTime As Integer

DelayTime = 10000 / BitRate

fncOneByteDelay = DelayTime

End Function

---------------------------
```

- Timeout: hàm api SetCommTimeouts quyết định thời gian thiết bị giao tiếp của Windows hoàn thành việc truyền nhận dữ liệu. Giá trị mặc định là 5 giây với mỗi việc này. Khi có 1 timeout xảy ra thì thì công việc sẽ bị huỷ bỏ, cái này xảy ra khi ứng dụng có thể chưa hoàn thành công việc nào cả(truyền hoặc nhận) hoặc chưa biết lỗi xảy ra ở đâu trong việc truyền, nhận. Dữ liệu nhận vẫn còn trong bộ đệm cho đến khi ứng dụng đọc nó nhưng bên truyền bộ đệm bị xoá khi xảy ra timeouts.
Để chống lại timeout dùng SetCommTimeouts đê thiết lập thông số cho thời gian ứng dụng hoàn thành việc truyền nhận dữ liệu.
Bắt tay phần cứng có thể chậm hơn bên truyền đữ liệu. Ví dụ liên kết với vđk thì cần phải có thời gian để cho vdk nhận biết được tín hiệu bắt tay do đó cần phải thêm thời gian này vào timeout.
Khi dùng thuộc tính Input hoặc Output để đọc, truyền dữ liệu thì nếu ứng dụng truyền một block nhở và bên nhận không cần trễ để bắt tay dài thì giá trị timeout mặc định là 5 giây sẽ không gây ra lỗi gì cả.
Trong chiều nhận, bạn có thể chống lại timeout bằng cách kiểm tra InBufferCount và dùng Input để đọc

đữ liệu ngay khi dữ liệu bắt đầu đến. Đó là cách rất hay vì VB sẽ trả lại một lỗi khi mà ứng dụng đọc bộ đệm rỗng.
Để vô hiệu hoá timeóut , thì thiết lập mọi giá trị = 0.

Mình đã tìm được một chương trình của bọn lvr.com
Đây là chương trình phải nói là rất là hay dùng chuẩn RS-485 với 1 master là PC và các có thêm 7 node khác là slave. Nó cũng cho phép hiển thị tất cả các cổng com có trong PC của bạn. Do tôi không có phần cứng thử nghiệm nên không test được.
Các bạn hoàn toàn có thể download về và lấy một phần đoạn mã để làm với ứng dụng của mình.
Tôi sẽ hoàn thành viết cho RS-232 sau.
Chương trình có hướng dẫn cụ thể có lẽ tôi chẳng cần phải hướng dẫn gì đâu. bạn nào có thắc mắc thì xin cho ý kiến. Tourial này có thể nói là đã gần hoàn thành rồi các bạn ạ. Dựa vào các kiến thức tôi đã đưa ra ở trên việc hiểu chương trình không còn là vấn đề gì nữa.
Chương trình này có dùng một số delay. Các bạn đọc kỹ nhé.
Hi vọng chương trình này sẽ làm cho các bạn cảm thấy hứng thú.

Phân tích code

Có lẽ mọi người đã có được chương trình rồi. Việc làm tiếp trên RS232 chẳng có gì là khó cả bạn ạ.
Do tôi đang rất bận, làm nhiều thứ. Nên tôi chỉ phân tích các hàm cho các bạn để các bạn dễ hiểu có thể làm cho mình những ứng dụng thực tế khác.
Còn về chương trình qua RS232 xin khất các bạn sau. Tôi đang phải làm viết cái đó trên VC++.
Trên đây tôi đã giới thiệu một vài hàm. Bây giờ tôi xin giới thiệu chi tiết một các hàm, các điều khiển dùng trong mỗi form:
- Trong file serport.bas : file thư viện
+ Gồm các hằng số lấy từ file win32api.txt như ở trên bao gồm: Parity, Stop bits, Error, CommEventMark bit, Escape Comm Function value,Bit rates, DTR Control Flow Values, RTS Control flow values, DCB Bits values. Chi tiết các bạn xem trong file gửi ở dưới. Tôi lấy một vài ví dụ các hằng số này như số bít stop:
Code:

```
'Stop bits

Global Const ONESTOPBIT = 0

Global Const ONE5STOPBITS = 1

Global Const TWOSTOPBITS = 2

… …
```

Các hằng số này có trong file win32api.txt.
+ Định nghia các kiểu dữ liệu làm tham số cho các hàm api: dcbType, COMMTIMEOUTS:
Code:

```
'define type COMMTIMEOUTS

Public Type COMMTIMEOUTS

     ReadIntervalTimeout As Long

    ReadTotalTimoutMultiplier As Long
```

```
    ReadTotalTimeoutConstant As Long
    WriteTotalTimeoutMultiplier As Long
    WriteTotalTimeoutConstant As Long
End Type


'define type dcbType
Public Type dcbType
        DCBlength As Long
        BaudRate As Long
        Bits1 As Long
        wReserved As Integer
        XonLim As Integer
        XoffLim As Integer
        ByteSize As Byte
        Parity As Byte
        StopBits As Byte
        XonChar As Byte
        XoffChar As Byte
        ErrorChar As Byte
        EofChar As Byte
        EvtChar As Byte
        wReserved2 As Integer
End Type
```

+ Các biến và hằng toàn cục dùng chung cho ứng dụng(rất quan trong xuyên suốt cả ứng dụng này)
Code:

```
Public Const ProjectName = "SerialPortComplete"


Public BitRate As Long  ' tần số bus
```

```
Public Buffer As Variant  ' bộ đệm
Public CommDCB As dcbType  '
Public CommPorts() As String  ' mảng để lưu trữ các cổng có trong PC
Public OneByteDelay As Single ' thời gian trễ khi truyền 1 byte
Public PortExists As Boolean ' Port tồn tại
Public PortInUse As Boolean ' port đang sử dụng
Public PortNumber As Integer ' số thứ tự của cổng
Public PortOpen As Boolean '
Public SaveDataInFile As Boolean '
Public TimedOut As Boolean  ' Có xảy ra timeouts hay không
Public ValidPort As Boolean ' Cổng có thực
```

+ Tiếp theo là khai báo một số hàm api và ý nghĩa của nó tôi sẽ nói trong khi dùng nó để các bạn biết rõ:

Code:

```
Public Declare Function apiGetCommState _
    Lib "kernel32"   Alias "GetCommState"   (ByVal nCid As Long, _
    lpDCB As dcbType)   As Long
Public Declare Function apiSetCommState    Lib "kernel32"
    Alias "SetCommState"   (ByVal hCommDev As Long, _
    lpDCB As dcbType)   As Long
Public Declare Function EscapeCommFunction  Lib "kernel32" _
    (ByVal nCid As Long,   ByVal nFunc As Long)   As Long
Public Declare Function GetCommTimeouts    Lib "kernel32" _
    (ByVal hFile As Long, lpCommTimeouts As COMMTIMEOUTS) _
    As Long
Public Declare Function SetCommTimeouts  Lib "kernel32" _
    (ByVal hFile As Long, _
    lpCommTimeouts As COMMTIMEOUTS)   As Long
Public Declare Function timeGetTime  Lib "winmm.dll" () As Long
```

```
Public Declare Function TransmitCommChar  Lib "kernel32" _

    (ByVal nCid As Long,    ByVal cChar As Byte)    As Long
```

+ Việc khai báo các hàm API đã xong bây giờ phải viết một vài chương trình chung cho cả ứng dụng:
++ Hàm fncByteToAsciiHex: chuyển đổi một byte thành 2 xâu bao gồm 2 kí tự ASCII HEX như 165 -> "A5"
Code:

```
Public Function fncByteToAsciiHex _

    (ByteToConvert As Byte) _

    As String

'Converts a byte to a 2-character ASCII Hex string

Dim AsciiHex As String

AsciiHex = Hex$(ByteToConvert) ' dùng hàm Hex$ để chuyển đổi.

If Len(AsciiHex) = 1 Then ' nếu chỉ là một kí tự thì thêm '0' vào trước

    AsciiHex = "0" & AsciiHex

End If

fncByteToAsciiHex = AsciiHex ' return

End Function
```

++ Hàm fncDisplayDataAndFile không tham số dùng chuyển đổi giờ hệ thống ra thành kiểu dạng : ngày – thòi gian
Code:

```
Public Function fncDisplayDateAndTime() As String

'Date and time formatting.

fncDisplayDateAndTime = _

    CStr(Format(Date, "General Date")) & ", " & _

    (Format(Time, "Long Time"))

End Function

------------------------
```

Phân tích code(tiếp)

++ hàm fncGetHighestComPortNumber
Code:

```
Public Function fncGetHighestComPortNumber() As Integer

'trả về chỉ số lớn nhất của cổng trong hệ thống

‘Dùng hàm API: EscapeCommFunction cần dùng 2 tham số là CommID của đối tượng MSComm và hằng số dùng cho hàm ở trên GETMAXCOM = 9.

Dim ClosePortOnExit As Boolean ‘ biến này dùng để biết là cổng giao tiếp đã mở chưa, hay đóng: true : cổng chưa mở, false: đã mở

Dim PortCount As Long

Dim handle As Long

'The API call requires a CommID of an open port.

‘ nếu chưa mở thì mở cổng và lưu trữ trạng thái trong ClosePortOnExit

If frmMain.MSComm1.PortOpen = False Then

    frmMain.MSComm1.PortOpen = True

    ClosePortOnExit = True

Else

    ClosePortOnExit = False

End If

‘ phải mở cổng thì mới lấy được CommID

handle = frmMain.MSComm1.CommID ‘ lấy CommID của điều khiển ‘MSComm.

PortCount = GETMAXCOM

'Thêm vào 1 bởi vì hàm EscapeCommFunction bắt đầu tính từ 0.

fncGetHighestComPortNumber = _

    EscapeCommFunction(handle, PortCount) + 1

‘ nếu cổng vừa mở thì đóng lại để trả lại trạng thái ban đầu.

If ClosePortOnExit = True Then

    frmMain.MSComm1.PortOpen = False

End If

End Function

------------------------
```

++ Hàm fncVerifyChecksum:
Code:

```
Public Function fncVerifyChecksum(UserString As String) As Boolean

'Kiểm tra dữ liệu bằng cách so sánh checksum nhận được với giá trị được tính
toán

'UserString là một loạt các byte kiểu Ascii Hex format,

'Kết thúc cho vào cuối  a checksum.

Dim Count As Integer

Dim Sum As Long

Dim Checksum As Byte

Dim ChecksumAsAsciiHex As String

‘Sum là tổng của các cắp số trong UserString.

For Count = 1 To Len(UserString) - 3 Step 2

    Sum = Sum + Val("&h" & Mid(UserString, Count, 2))

Next Count

'checksum byte thấp của Sum

Checksum = Sum - (CInt(Sum / 256)) * 256

‘chuyển sang kiểu xâu

ChecksumAsAsciiHex = fncByteToAsciiHex(Checksum)

'So sánh với checksum vừa tính toàn với checksum nhận được: 2 byte cuối của
UserSetting

If Checksum = Val("&h" & (Right(UserString, 2))) Then

    fncVerifyChecksum = True ‘ nếu đúng thì trả lại True và ngược lại

Else

    fncVerifyChecksum = False

End If

End Function

-------------
```

++ Hàm trễ với thời gian cho vào là kiểu mili giây
Code:

```
Public Sub Delay(DelayInMilliseconds As Single)

'hàm trễ từng mili giây một

' dùng hàm API timeGetTime.

'sẽ bị tràn sau 24 ngày sau khi hệ thống bắt đầu khởi động.

Dim Timeout As Single

Timeout = DelayInMilliseconds + timeGetTime() ' Timeout = thời gian hiện tại
+ thời gian trễ

' trễ nếu thời gian của hệ thống không nhỏ hơn thì không làm gì cả

Do Until timeGetTime() >= Timeout

    DoEvents ' chẳng làm gì cả, trạng thái nghỉ.

Loop

End Sub

--------------
```

++ hàm EditDCB :

Code:

```
Public Sub EditDCB()

'Enables changes to a port's DCB.

'The port must be open.

Dim Success As Boolean

Dim PortID As Long

PortID = frmMain.MSComm1.CommID

Success = apiGetCommState(PortID, CommDCB)



'To change a value, uncomment and revise the appropriate line:

'CommDCB.BaudRate = 2400

'CommDCB.Bits1 = &H11

'CommDCB.XonLim = 64

'CommDCB.XoffLim = 64

'CommDCB.ByteSize = 8
```

```
'CommDCB.Parity = 0
'CommDCB.StopBits = 0
'CommDCB.XonChar = &H12
'CommDCB.XoffChar = &H13
'CommDCB.ErrorChar = 0
'CommDCB.EofChar = &H1A
'CommDCB.EvtChar = 0

'Write the values to the DCB.
Success = apiSetCommState(PortID, CommDCB)

'Read the values back to verify changes.
Success = apiGetCommState(PortID, CommDCB)

Debug.Print "DCBlength: ", Hex$(CommDCB.DCBlength)
Debug.Print "BaudRate: ", CommDCB.BaudRate
Debug.Print "Bits1: ", Hex$(CommDCB.Bits1); "h"
Debug.Print "wReserved: ", Hex$(CommDCB.wReserved)
Debug.Print "XonLim: ", CommDCB.XonLim
Debug.Print "XoffLim: ", CommDCB.XoffLim
Debug.Print "ByteSize: ", CommDCB.ByteSize
Debug.Print "Parity: ", CommDCB.Parity
Debug.Print "StopBits: ", CommDCB.StopBits
Debug.Print "XonChar: ", Hex$(CommDCB.XonChar); "h"
Debug.Print "XoffChar: ", Hex$(CommDCB.XoffChar); "h"
Debug.Print "ErrorChar: ", Hex$(CommDCB.ErrorChar); "h"
Debug.Print "EofChar: ", Hex$(CommDCB.EofChar); "h"
Debug.Print "EvtChar: ", Hex$(CommDCB.EvtChar); "h"
```

```
Debug.Print "wReserved2: ", Hex$(CommDCB.wReserved2)



End Sub
```

++ Hàm findPorts: giới thiệu rồi
++ Lấy thiết lập mới từ lựa chọn của người dùng từ frmPortSetting. Để hiểu nó các bạn đọc tiếp các phàn sau:
Code:

```
Public Sub GetNewSettings()

'Read and store user changes in the Setup menu.

BitRate = Val(frmPortSettings.cboBitRate.Text)

PortNumber = Val(Right(frmPortSettings.cboPort.Text, 1))

‘ gọi hàm khởi tạo cho cổng

Call frmMain.fncInitializeComPort(BitRate, PortNumber)

End Sub

++  Lấy thông số ban đầu:

Public Sub GetSettings()

'Get user settings from last time.

BitRate = GetSetting(ProjectName, "Startup", "BitRate", 1200)

PortNumber = GetSetting(ProjectName, "Startup", "PortNumber", 1)

'Defaults in case values retrieved are invalid:

If BitRate < 300 Then BitRate = 9600

If PortNumber < 1 Then PortNumber = 1

End Sub
```

++ ImmediateTransmit truyền trực tiếp các byte.
Code:

```
Sub ImmediateTransmit(ByteToSend As Byte)

'Places a byte at the top of the transmit buffer

'for immediate sending.

Dim Success As Boolean

Success = TransmitCommChar(frmMain.MSComm1.CommID, ByteToSend)
```

```
End Sub
```

Lập trình giao tiếp nối tiếp bằng Visual C++

Chào các bạn.

Trong thời gian qua tôi hơi bận nên chưa tiếp tục chủ đề này được. Có lẽ phần mềm Visual Basic mọi người khá thông thạo thì không phải bàn cãi gì nữa. Tôi xin gửi cho mọi người chương trình khá đơn giản để gửi kí tự mình đánh từ bàn phím qua RS232 đến vi điều khiển:
http://luckytoki.com/haibac/download...SerialPort.rar

Tôi xin tiếp tục hướng dẫn các bạn làm việc với Visual C++ , công cụ khá mạnh mà nếu các bạn học qua C,C++ thì làm việc với Visual C++ lại khá là đơn giản.

Ở đây tôi cũng dùng điều khiển ActiveX là Microsoft Communication 6.0 của bọn Microsoft. Và thiết lập thông số hoàn toàn tương tự với dùng với VB. Các bạn nên đọc qua các bài viết của tôi ở trên để nắm được hướng làm việc, cách thiết lập thông số.

Giới thiệu thế thôi. Mai viết tiếp.

Giới thiệu điều khiển MSComm trong VC++

Chào các bạn. Cách lập trình giao tiếp cổng Com có hơi khác với dùng VB một chút thôi. Chỉ cần các bạn có kiến thức đầy đủ về C,C++ lập trình hướng đối tượng như lớp Class, Thừa kế(lớp cơ sở, lớp dẫn suất),.. là các bạn có thể hoàn toàn có thể lập trình được ngon lành rồi.

Trước hết tôi xin giới thiệu các bạn cách lấy điều khiển MSComm cho vào list của bạn để dùng:

Bước 1:

Bạn vào Project -> Add to Project ->Components Control...

Bước 2:
Tiếp theo các bạn chọn như hình vẽ:

Khi đó nó sẽ định nghĩa cho chúng ta một lớp có tên là MSComm được định nghĩa trong 2 file :
- Header file: MSComm.h
- Source file: MSComm.c

Khi đó bạn gắp thả vào trong điều khiển của mình bình thường như các điều khiển khác:

Kích chuột phải chọn Properties khi đó sẽ hiện lên một số các thuộc tính của điều khiển như tên ID, Setting ban đầu. Những cái này các bạn xem qua chắc hẳn thấy rất giống với trong VB đúng không? Những cái này hoàn toàn lập trình được chúng ta sẽ làm sau:

Để có thể lập trình với điều khiển này. Các bạn ấn Ctr- W để sau đó thêm một biến điều khiển để lập trình. Biến này là biến Control , kiểu CMSComm.

Quên không mang theo hình vẽ nên không cho các bạn xem được.

Giới thiệu lớp MSComm

Các bạn mở file MSComm.h thì các bạn sẽ biết được các hàm được định nghĩa đối với lớp MSComm. Còn chi tiết nó thực hiện như thế nào thì xem trong file MSComm.c

Tất cả các hàm này đều không được nói rõ trong MSDN có lẽ là do sự tương tự với bản thân nó trong VB.

Tôi xin giới thiệu cho các bạn lớp này: Các điều khiển trong VC toàn thừa kế từ lớp CWnd do đó các bạn có thể dùng các hàm trong lớp CWnd như GetDlgItemText(..) , GetDlgItem(....),...

Code:

```
class CMSComm : public CWnd

{

protected:

        DECLARE_DYNCREATE(CMSComm)

public:

        CLSID const& GetClsid()

        {

                static CLSID const clsid
```

```
                        = { 0x648a5600, 0x2c6e, 0x101b, { 0x82, 0xb6, 0x0, 0x0,
0x0, 0x0, 0x0, 0x14 } };

                return clsid;

        }

        virtual BOOL Create(LPCTSTR lpszClassName,

                LPCTSTR lpszWindowName, DWORD dwStyle,

                const RECT& rect,

                CWnd* pParentWnd, UINT nID,

                CCreateContext* pContext = NULL)

        { return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect,
pParentWnd, nID); }



    BOOL Create(LPCTSTR lpszWindowName, DWORD dwStyle,

                const RECT& rect, CWnd* pParentWnd, UINT nID,

                CFile* pPersist = NULL, BOOL bStorage = FALSE,

                BSTR bstrLicKey = NULL)

        { return CreateControl(GetClsid(), lpszWindowName, dwStyle, rect,
pParentWnd, nID,

                pPersist, bStorage, bstrLicKey); }



// Attributes

public:



// Operations

public:

        void SetCDHolding(BOOL bNewValue);

        BOOL GetCDHolding();

        void SetCommID(long nNewValue);

        long GetCommID();
```

```
void SetCommPort(short nNewValue);
short GetCommPort();
void SetCTSHolding(BOOL bNewValue);
BOOL GetCTSHolding();
void SetDSRHolding(BOOL bNewValue);
BOOL GetDSRHolding();
void SetDTREnable(BOOL bNewValue);
BOOL GetDTREnable();
void SetHandshaking(long nNewValue);
long GetHandshaking();
void SetInBufferSize(short nNewValue);
short GetInBufferSize();
void SetInBufferCount(short nNewValue);
short GetInBufferCount();
void SetBreak(BOOL bNewValue);
BOOL GetBreak();
void SetInputLen(short nNewValue);
short GetInputLen();
void SetNullDiscard(BOOL bNewValue);
BOOL GetNullDiscard();
void SetOutBufferSize(short nNewValue);
short GetOutBufferSize();
void SetOutBufferCount(short nNewValue);
short GetOutBufferCount();
void SetParityReplace(LPCTSTR lpszNewValue);
CString GetParityReplace();
void SetPortOpen(BOOL bNewValue);
BOOL GetPortOpen();
```

```
        void SetRThreshold(short nNewValue);
        short GetRThreshold();
        void SetRTSEnable(BOOL bNewValue);
        BOOL GetRTSEnable();
        void SetSettings(LPCTSTR lpszNewValue);
        CString GetSettings();
        void SetSThreshold(short nNewValue);
        short GetSThreshold();
        void SetOutput(const VARIANT& newValue);
        VARIANT GetOutput();
        void SetInput(const VARIANT& newValue);
        VARIANT GetInput();
        void SetCommEvent(short nNewValue);
        short GetCommEvent();
        void SetEOFEnable(BOOL bNewValue);
        BOOL GetEOFEnable();
        void SetInputMode(long nNewValue);
        long GetInputMode();
};
```

Khi đó chắc bạn đã biết về cách thiết lập trong VC đúng không?

Sự tương tự là với một thuộc tính nào đó như InputLen thì nếu lấy giá trị này thì gọi :
Code:

```
CMSComm a;
int b = a.GetInputLen();    // lấy giá trị
a.SetInputLen(1); //  thiết lập thuộc tính
```

Các thiết lập khác cũng tương tự.

Các bạn lại nhìn lại các hàm SetOutput:

Code:
```
void SetOutput(const VARIANT& newValue);
```

Biến ở đây là biến hằng tham chiếu đến kiểu VARIANT. Đây là một cấu trúc do đó bạn không thể nào trực tiếp chuyển đổi kiểu nó từ kiểu CString mà bạn phải làm như sau: Thông qua một lớp COleVariant
Code:
```
if (m_mscomm1.GetPortOpen())

  m_mscomm1.SetOutput(COleVariant(m_dToSend));
```

// còn chiều nhận lại trả về kiểu VARIANT bạn phải làm sao để chuyển sang kiểu CString để dùng.

Code:
```
VARIANT data = m_mscomm1.GetInput();

  m_dReceive = (CString) data.bstrVal; // ép chuyển đổi kiểu
```

Chúc các bạn thành công.

Dùng ngắt OnComm

Các bạn đã biết cho điều khiển activeX vào rồi. Hẳn các bạn cũng tự hỏi là mình sẽ lập trình với điều khiển này thế nào để nó làm theo ý muốn như trong VB. Ví dụ như hàm ngắt OnComm , dùng thế nào,làm sao biết ngắt nào là ngắt nhận, truyền,.. Tôi sẽ giới thiệu cho các bạn từ từ.

Trước hết tôi xin đưa ra các tham số cần thiết lập ban đầu cho điều khiển MSComm của các bạn. Tôi cho đây là tối ưu nhất cho việc nhận dữ liệu và truyền dữ liệu. Các tham số này trong ứng dụng đơn giản bạn nên đặt vào trong hàm OnInitDialog() (các bạn làm việc với VC++ đều biết cái này nằm ở đâu nên tôi không cần nói nữa).

Tôi ví dụ bạn đã đặt tên cho biến điều khiển của MSComm1 là m_mscomm1.

Code:
```
        // Thiết lập cho mscomm

        m_mscomm1.SetCommPort(1);// chọn cổng COM1

        m_mscomm1.SetHandshaking(0); // không chọn bắt tay

         m_mscomm1.SetSettings("9600,N,8,1");



        // thiết lập cho bên truyền

        m_mscomm1.SetInBufferSize(1024); // bộ đệm nhận

        m_mscomm1.SetInputLen(1);  // mỗi lần đọc 1 kí tự
```

```
m_mscomm1.SetRThreshold(1); // dùng sự kiện comEvReceive

m_mscomm1.SetInputMode(0); // Text Mode

m_mscomm1.SetOutBufferSize(512);

//m_mscomm1.SetSThreshold(0);// don't use comEvSend
```

Thế là các bạn đã thiết lập xong.

Bây giờ bạn kích đúp chuột vào điều khiển MSComm1 nó sẽ gợi ý cho bạn đặt tên hàm là : OnOnCommMscomm1 . Bạn đồng ý luôn.

Khi bạn dùng ngắt thì giá trị trả về của m_mscomm1.GetCommEvent() cho ta biết là đang có ngắt nào. Giá trị đó chính là số thứ tự của các ngắt trong thuộc tính ComEvent của VB tôi đã giới thiệu trên (trong MSDN).

Do đó hàm đơn giản như sau(trong OnOnCommMscomm1()):

Code:

```
UpdateData(true);

switch( m_mscomm1.GetCommEvent()){

   case 1: // comEvSend

            // làm gì thì làm

           break;

   case 2://  comEvReceive

           VARIANT data = m_mscomm1.GetInput();

            CString dataReceive = (CString) data.bstrVal;

           break;

   case 3:// comEvCTS

           break;

   case 4://comEvDSR

           break;

   case 5: //comEvCD

           break;

   case 6://comEvRing
```

```
                break;
        case 7: //comEvEOF
                break;
        };
    UpdateData(false);
```

Hi vọng các bạn làm việc với điều khiển ngon lành.

Coi như đây là quà đầu năm mới chúc mọi người sức khỏe tôt(để còn cầy).

Chúc bạn, bạn của bạn, người yêu của bạn sức khỏe tốt, hạnh phúc,.. thích gì được lấy.

Cảm ơn bạn. Có lẽ vấn đề giao tiếp qua cổng Com là vấn đề mà nhiều bạn hỏi rất là nhiều nên tôi chỉ muốn xây dựng một luồng này như là nơi để các bạn giải đáp được những thắc mắc của mình về vấn đề giao tiếp qua cổng Com thôi.

Bài viết ở trên tôi chưa thiết lập mở cổng. Để mở cổng giao tiếp thì các bạn phải mở cổng giao tiếp trong một khâu nào đó trước khi mà các bạn gửi dữ liệu ra ngoài trong một hàm nào đó.

Cú pháp của mở cổng như sau:

Code:

```
m_mscomm1.SetPortOpen(true); // mở cổng

m_mscomm1.SetPortOpen(false);  // đóng cổng


bool a = m_mscomm1.GetPortOpen();  // lấy trạng thái của cổng
```

Các bạn lưu ý là trước khi các bạn gửi dữ liệu thì phải chắc chắn là cổng đã mở. Do đó các bạn nên lấy thông tin về trạng thái của cổng như trên trước khi gửi nha.

Chúc thành công. Tôi sẽ viết tiếp vào ngày mai.

Chương trình mẫu giao tiếp RS232 trên PC

Đây là bài tôi viết trên http://picvietnam.com tại luồng http://picvietnam.com/showthread.php?p=1347#post1347

Chào các bạn tôi xin đưa cho các bạn chương trình mẫu để giao tiếp với RS232 trên PC. Chương trình này tôi viết bằng Visual C++ trong bộ Microsoft Studio 6.0 của Microsoft, các bạn dịch lại ra file .exe để chạy. Phần hướng dẫn chi tiết tôi sẽ gửi sau khi được kiểm duyệt kĩ lưỡng. Chương trình này còn đơn giản. Sẽ có nhiều phiên bản sau ra đời.

Giao diện chương trình như sau:

Hình 1: Giao diện chương trình

Chương trình có chức năng sau:

- Nhập kí tự hoặc xâu kí tự vào EditBox Transfer, điều chỉnh tham số giao tiếp trên các ComboBox. Nhấn nút Send để gửi dữ liệu ra cổng COM.

- Đồng thời với nó nếu có dữ liệu truyền về cổng Com thì dữ liệu sẽ được hiển thị lên EditBox Receive. Khi bạn nhấn vào Clear thì sẽ xoá dữ liệu hiển thị trên EditBox này

Chú ý:

Để có thể test luôn chương trình các bạn nối tắt chân 2 và chân 3 của RS232 lại với nhau chính là nối chân RxD và TxD để chúng ta truyền dữ liệu ra RS232 sau đó nhận dữ liệu luôn.

Hình 2: Sơ đồ đấu chân của RS232

Chúc các bạn thành công.

Anh Công quá khen rồi.

Mình hi vọng là TUT này sẽ hướng dẫn các bạn cơ bản về cách lập trình ứng dụng này.

Xin giới thiệu lại TUT vậy.

Chào các bạn, sau một số ngày cung phu làm TUT này, cuối cùng TUT cũng được anh Falleaf phê duyệt.

TUT này là về cách lập trình giao tiếp RS232 trên PC bằng công cụ Visual C++ 6.0 trong bộ Visual Studio 6.0 của Microsoft. Tài liệu có nói chi tiết về các thuộc tính của MSComm và cũng hướng dẫn rất chi tiết về cách lập trình.

Tài liệu này bao gồm 78 trang(hơi cung phu) có hình vẽ minh hoạ đầy đủ từng bước đảm bảo các bạn làm theo là được ngay.

Đây là một hình minh họa trong TUT của tôi.

Thứ nhất, kiểm tra các thuộc tính của cổng COM

Thứ hai,tạo dự án

Thứ ba,thêm thư viện MSComm:

Thứ tư,tạo giao diện:

Thứ năm,đặt các thuộc tính cho điều khiển:

Thứ sáu.thêm các biến điều khiển:

Thứ bảy,viết mã cho chương trình:

Chúc các bạn thành công.

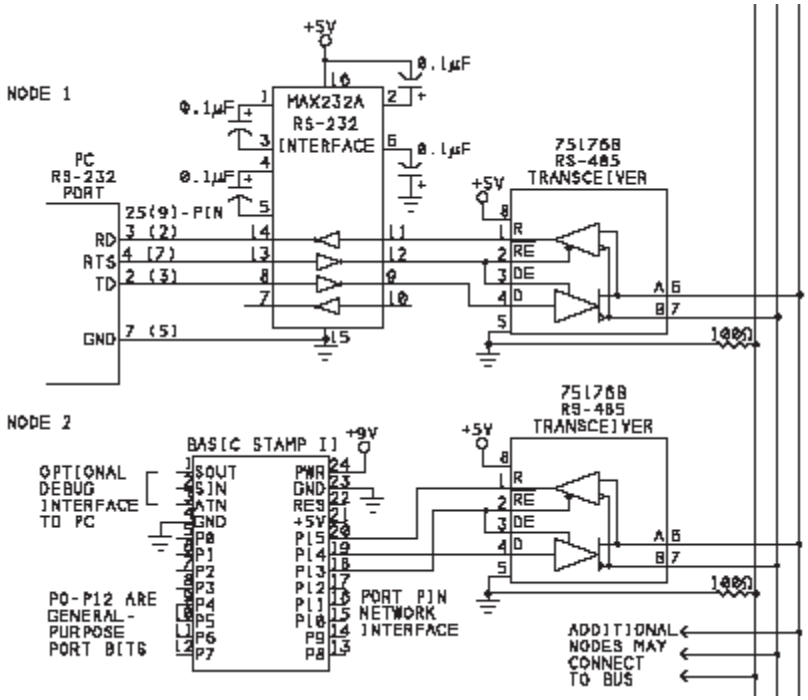
Nếu nói về chuẩn hoặc xây dựng cả một chuẩn riêng thì cả một vấn đề. Và vấn đề ở đây mình cũng chưa thấy bạn nào phân biệt một cách rõ ràng chuẩn và giao thức. Qua quá trình tìm hiểu ngắn thì theo mình biết chuẩn là những quy chuẩn quy định về mặt vật lý (mức tín hiệu, đường truyền, ...). Còn giao thức (protocol) là những quy định để cho phép các phần tử truyền thông trong mạng hiểu và nhận đúng dữ liệu, tránh sự sai lạc trong quá trình truyền. Đơn giản nhất nếu bạn tìm hiểu về truyền thông RS232 thì về chuẩn truyền thông quy định mức tín hiệu logic -12V và +12V và đi kèm với nó là các giao thức bắt tay phần cứng (sử dụng các chân CTS, RTS, DTR,...), giao thức bắt tay mềm (X-ON, X-OFF) hoặc không sử dụng giao thức bắt tay. Hay một ví dụ đơn giản nữa , nếu bạn nào đã từng truyền thông với PLC của Mitshubishi mà cụ thể là họ FX thì các bạn cũng sẽ hiểu. Công lập trình của FX sử dụng chuẩn RS422 nên khi muốn truyền thông với máy tính phải sử dụng bộ chuyển đổi RS422 sang RS232. Tuy nhiên nếu chỉ thực hiện chuyển đổi đơn thuần (đồng mức vật lý) thì không thể truyền thông với FX được và vấn đề chúng ta cần phải quan tâm nữa là giao thức. Với các thiết bị phức tạp như PLC thì có rất nhiều vùng dữ liệu với các địa chỉ khác nhau đồng thời phải thực hiện quá trình kiểm tra lỗi dữ liệu, FX thực hiện truyền thông theo các Frame và thực tế nếu ngoài dữ liệu chính trong Frame còn có các thông tin như các ký tự điều khiển (stx,...), địa chỉ dữ liệu, dữ liệu, mã kiểm tra SUMCHECK CODE,... Vì vậy mong các bạn hãy cùng mình phân biệt một cách cụ thể, rõ ràng hơn về chuẩn và giao thức.

Máy tính, giao tiếp với VDK qua mạng 485

Trong Max485 có hai chân RE, DE. Em đã thử nếu như nối chân DE với nguồn, RE với đất, để luôn luôn cho phép truyền và cho phép nhận. Tuy nhiên em nhận thấy nếu để như vậy thì không thể truyền tín hiệu được. Mà phải nối với nhau và được điều khiển bằng 1 chân. Nếu muốn truyền thì set bit = 1, sau khi truyền xong mà muốn nhận thì clear bit = 0. Ở đây em có đưa lên mạch mà em đã sử dụng để truyền thông giữa máy tính và vi điều khiển qua mạng 485 (có sử dụng bộ chuyển đổi 232 -485) . việc điều khiển 2 chân DE, RE thông qua chân RTS. Giả sử muốn truyền từ máy tính xuống một dữ liệu và muốn vi điều khiển phải trả lời thì phải làm như sau: từ máy tính cho RTS = 1, truyền dữ liệu, cho **delay** để truyền hết dữ liệu, [/B]cho RTS = 0 để cho phép máy tính nhận từ nhận từ vi điều khiển

...Với cách làm như trên sẽ làm cho 1 vòng quét chiếm mất nhiều thời gian, đặc biệt nếu sử dụng nhiều vi điều khiển (máy tính là master, VDK là slave). Không biết có cách nào để biết là máy tính đã truyền hết dữ liệu xuống vi điều khiển hay chưa, Hình như phải kiểm tra một thanh ghi điều khiển gì đó. Ở đây em dùng ngôn ngữ C# cho việc điều khiển trên máy tính. Kính mong Cao Thủ chỉ giúp cho em.

Attached Files



Về mạch chuyển RS485 sang RS232 và ngược lại trên mạng cũng rất nhiều. Theo hình vẽ của Picachu thì dùng IC 75176, và cổng COM phải dùng thêm chân RTS--> điều này là không cần thiết: sửa lại như sau--> nối chân 2 và 3 của IC75176 vào nhau. Chân DI (chân 4 của IC75176) cho qua 2 con NOT(dùng CD4069- 1 chíp có 6 bộ NOT) sau đó cho qua mạch R1//R2//Diode (220K//33K//diode)-> nối đến 1 chân tụ 102 (chân còn lại của tụ nối mát)--> đi đến chân 2 và 3 IC 75176. -----> thế là OK, trên phần mềm không phải điều khiển chân RTS nữa, cứ truyền cổng COM bình thường là bên nhận sẽ nhận đúng. Mạch này tôi đã sản xuất vài chục mạch -> chạy rất tốt.
Đã áp dụng làm mạch lặp RS485: chỉ việc ghép 2 mạch trên lại nhưng bỏ IC max232 đi là xong.

m đã làm đúng như anh hoctro89xxx nói nhưng mạch không chạy, Nhưng khi chỉ sử dụng 1 con NOT rồi sau đó cho qua mạch R1//R2//Diode (220K//33K//diode)... như anh hoctro89xxx nói thì mạch lại chạy bình thường . Mà thậm chí rằng bỏ cả R1,R2,Diode mạch vẫn chạy.
Tức là mạch bây giờ đơn giản chỉ là Chân DI (chân 4 của MAX485) qua 1 con NOT sau đó đi đến chân 2,3 của Max485. Như vậy là chạy bình thường. Không hiểu tại sao lại như vậy. Liệu như vậy thì mạch có chạy ổn định không. Hiện tại thì em chưa phát hiện ra lỗi. Rất có thể sau này sẽ có lỗi. Anh hoctro89xxx có thể chỉ giúp cho em được không.

Hic, định post lại là chỉ cần 1 NOT (hay 3 NOT). mạch gin của nó còn nhiều thứ khác nhưng qua thực tế đã lược bớt như thế. với loại chỉ dùng 1 NOT và bỏ hết trở tụ... Tôi chỉ làm 1 mạch và hiện vẫn chạy--> chưa vấn đề gì.
còn mạch đủ cả tụ, trở và 1 NOT tôi đã sản xuất 50 chiếc, đã cho chạy liên tục cả 50 chiếc 24h/24h/3 tháng, cả 50 chiếc này luôn có tín hiệu data qua lại-----> chưa thấy lỗi gì cả-----> thật tuyệt vời

Mắc diode quay Anode vào NOT.

Không có gì!
Ngày xưa mới tìm hiểu, thấy sơ đồ nó bắt viết phần mềm điều khiển chân 2 và 3 IC75176--> ngại --> bỏ. thời gian sau kiếm được cái mạch này--> quá hay, chỉ việc truyền RS232 như bình thường.
Sắp tới sẽ cho ra mắt sản phẩm:
1.RS232 <-> RS485
2.RS485 <-> RS485
HI vọng sẽ bán được nhiều!

Mình đã VẼ ĐƯỢC ĐỒ THỊ bằng Teechart trong VB , với dữ liệu từ cổng COM rồi, bạn nào quan tâm , xin vào Thread này để xem nghen (thread này cũng do mình lập ra):

http://dientuvietnam.net/forums/show...t=11088&page=2

Link Ve DO THi ne:

http://dientuvietnam.net/forums/showthread.php?t=11088