COMP370 Lab 7: A “Chat” Application using MFC

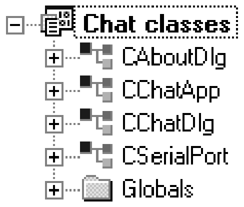
In this experiment you’ll use MFC to create a simple terminal-emulation program. This program will communicate through the serial port on the PC to another computer. With two programs running together (or EZ-Terminal on the other computer) you can “chat” point-to-point!

Instructions:

1. Create a new dialog based application called "Chat".

2. Download the files "SerialPort.cpp" and "SerialPort.h" from the instructor's web site. Place these into the project folder, and add the two files to your project. (Project->Add To Project->Files...) The CSerialPort object encapsulates the functionality of a serial communications port on your computer. You'll use it to communicate with the other computer.

3. Compile the project with F7. No errors should occur, and the class view should look like this:

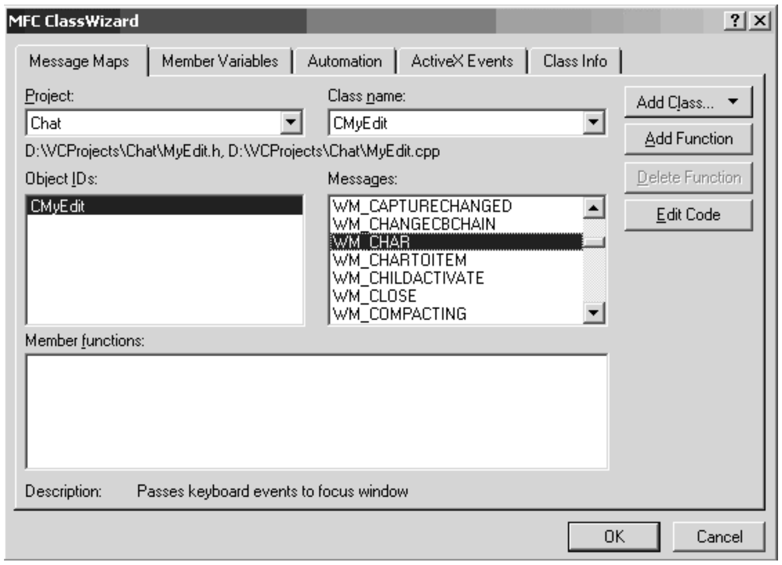


4. Now we're going to add some functionality to the code. In order to intercept keystrokes, we're going to derive a new class from CEdit. Add the new class CMyEdit as follows:

   a) Right-click on "Chat classes" in ClassView and select "New Class."
   b) Set the Class type as MFC Class, and name it CMyEdit. Set the BASE CLASS as CEdit. Leave Automation as "None."
   c) Click OK to create the new class.

5. We're going to intercept the message "WM_CHAR" in the CMyEdit class as follows:

a) Click on "CMyEdit" in class view.
b) Type ^W to bring up the ClassWizard. Make sure CMyEdit is selected under Object IDs, and select the message "WM_CHAR" in the Messages window. It should look like this:



Project: Chat

Class name: CMyEdit

D:\VCProjects\Chat\MyEdit.h, D:\VCProjects\Chat\MyEdit.cpp

Object IDs: CMyEdit

Messages: WM_CAPTURECHANGED, WM_CHANGECBCHAIN, WM_CHAR, WM_CHARTOITEM, WM_CHILDACTIVATE, WM_CLOSE, WM_COMPACTING

Member functions:

Description: Passes keyboard events to focus window

c) Click "Add Function" to add the function "OnChar()" to the class.
d) Add a PUBLIC member variable to class CMyEdit:

```
public:

CSerialPort* m_pPort;
```

You can do this by right-clicking on the class name "CMyEdit" in the class pane, and choose "Add Member Variable" from the context menu. Fill in the blanks appropriately. This variable will be used to access the serial port object.

e) The OnChar() handler function should be added to intercept the incoming keyboard characters:

```
void CMyEdit::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
if (m_pPort)
    {
    m_pPort->WriteByte( (char) nChar ); // write char to serial port
    }

CEdit::OnChar( nChar,  nRepCnt,  nFlags); // Echo typed character
}
```

f) You'll need to include "SerialPort.h" at the top of "MyEdit.h" so that the compiler knows about the class CSerialPort:

```
// MyEdit.h : header file
//

#include "SerialPort.h"

/////////////////////////////////////////////////////////////////////////
  ///
// CMyEdit window
```

g) Your code should still compile properly at this point!

6. Now we need to initialize the CSerialPort object in the main dialog, and open up a CMyEdit window. Add the following variables with PRIVATE access to the CChatDlg class (right-click on "CChatDlg" and select "Add Member Variable" from the context menu.)

```
   CMyEdit* m_pEditWnd; // Pointer to edit window which will be created
   CSerialPort m_Port;  // One instance of the serial port control
```

At the top of ChatDlg.h, you'll need to add the following:

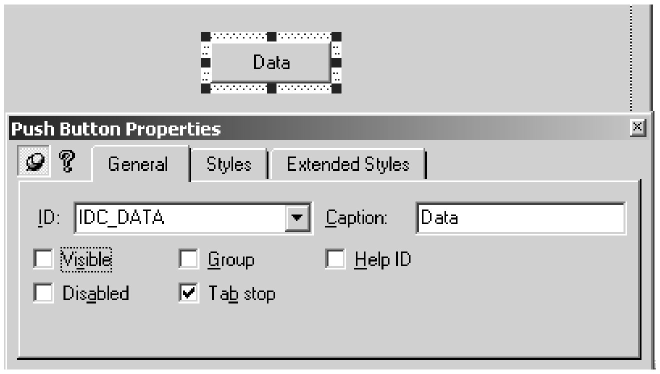
```
#include "SerialPort.h" // Added by ClassView
#include "MyEdit.h"
```

7. In the class view, expand [+] the view of the CChatDlg class if it isn't already expanded. Double-click on the "OnInitDialog()" function in the CChatDlg class view. We're going to add the following code to the *end* of this function to initalize the serial port, create the MyEdit window, and place the window into the dialog box:

```
// TODO: Add extra initialization here

m_Port.Open( 1, 9600);  // COM1 , 9600 bps: Change as needed for YOUR
  workstation

m_pEditWnd = new CMyEdit;     // Construct a CMyEdit object

m_pEditWnd->m_pPort = &m_Port;// Tell it where our serial port is...

m_pEditWnd->Create( ES_MULTILINE | ES_AUTOHSCROLL |
  ES_AUTOVSCROLL | ES_WANTRETURN | WS_DLGFRAME
  | WS_VSCROLL ,        // Creation style (dlg frame, multiline + Vscroll)
  CRect(10,10,380,300), // Size & location of Edit Window
  this,                 // Owner of the control (US!)
  1001);                // ID to assign to the control (start at 1000)


m_pEditWnd->ShowWindow(SW_SHOWNORMAL);
m_pEditWnd->SetFocus();
               // The following statement was provided by the framework.
               // We must modify it to return FALSE since we're setting
return FALSE;  // focus to a specific control (the Edit Window)
```

8. Your code should now compile and run. You should see the edit window with a flashing cursor when you run the program. Anything you type should appear in the edit window. *If this works, you're now transmitting data to the serial port!* All we need to do now is add the logic to process incoming characters. We're going to use a second (background) thread for this job, which is automatically provided by the CSerialPort object. The serial port will send us a WM_COMMAND message when it gets incoming data, and we'll simply intercept the message with a "dummy" button handler.

   a) Switch to the dialog in the resource editor, and add a button called "Data" to the dialog box. Make sure it has the id "IDC_DATA" and the "Visible" attribute unchecked. This is a "dummy" button that creates a message map entry for the event handler.



ID: IDC_DATA Caption: Data
☐ Visible ☐ Group ☐ Help ID
☐ Disabled ☑ Tab stop

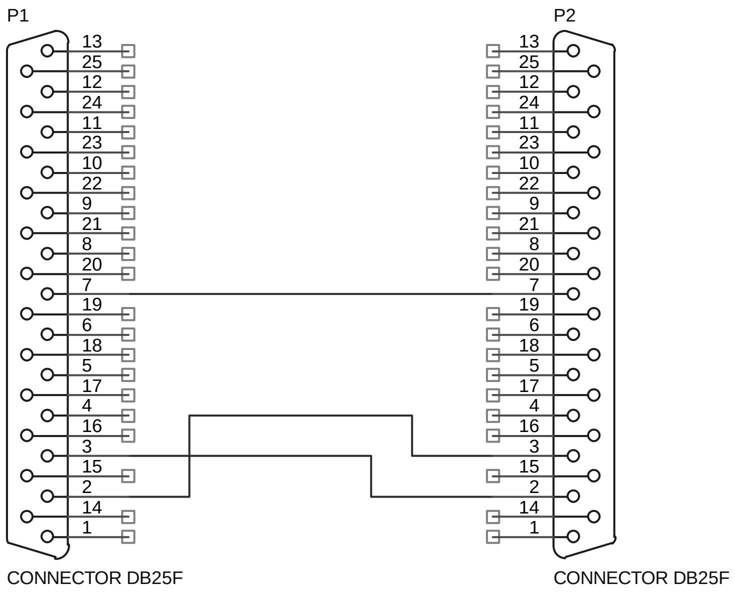
b) At the end of the “ OnInitDialog()” function in CChatDlg (just after the code you added above, but BEFORE the RETURN statement), add the following code the activate the subthread:

```
m_Port.StartReceiverThread( this, IDC_DATA); //trigger on any incoming
                                             //data

               // The following statement was provided by the framework.
               // We must modify it to return FALSE since we’re setting
return FALSE;  // focus to a specific control (the Edit Window)
```

c) In the ClassWizard, create a message map entry for "IDC_DATA" and add a message handler function for "BN_CLICKED." (The CSerialPort object sends this type of message when data comes in; it "pretends" to be a control button.) The following code is the handler:

```
void CChatDlg::OnData()
{
CString aString;
char szIn[32];
int nResult;

m_pEditWnd->GetWindowText(aString); // Grab ALL text in Edit Window

// We got data, so ask the serial port (who originated this “event”)
// for the data that came in…

nResult = m_Port.ReadString( szIn, 31 );

// The incoming string is binary, so must provide zero-terminator for
// interpretation as a text string

szIn[nResult] = 0;      // Terminate string

// If there’s any data, add it to the Edit Window text, then push it back
// out to the Edit Window. (Crude, but works!)

if (nResult)
  {
  aString += szIn;      // Add text to string buffer
  m_pEditWnd->SetWindowText(aString); // update the edit window with new
  text

  // Force caret to end of edit window…

  m_pEditWnd->SetSel( aString.GetLength(), aString.GetLength(), FALSE);
  }

}
```

9. That's it! Build the application, connect your computer to another workstation using a crossover cable, and launch the program. Whatever you type into the Edit window should show up on the other side of the connection! Here's how to build a suitable crossover cable (or use a break-out box, available in the crib):


P1
P2
13
25
12
24
11
23
10
22
9
21
8
20
7
19
6
18
5
17
4
16
3
15
2
14
1
CONNECTOR DB25F
CONNECTOR DB25F