# COMP 3980 – Assignment #1 Design Implementation: "Dumb" Terminal Emulation

| State/Section | Details |
|---|---|
| **COMMAND (Start state)** | *Purpose*: Allows user to specify port number, and port parameters such as bitrate, data bits, parity, stop bits, and flow control.<br>Starts with no connection or session.<br>*Control Flow*: From here, users can move to: **Connect** \| **Exit** |
| **CONNECT** | *Purpose*: Makes the connection with the serial port to allow the transmitting and receiving of characters. Creates the thread to loop the reading of characters from keyboard to send to the serial port. Continuously monitors the arrival of characters on serial port.<br><br>Notes: Connected flag will indicate the state of the user session. In a connected state, certain UI elements will be disabled to ensure the user does not switch ports when they are still connected. In the disconnected state, UI elements will be re-enabled to allow ports to be reconfigured.<br><br>*Control Flow:* From here, users can move to: **Command** \| **Exit** |
| GET INPUT | *Purpose*: Monitors keyboard character messages.<br><br>Notes: No need to use a thread here, as the characters can be sent right away after receiving the keyboard messages. There is no benefit to using a thread for writing. |
| SEND CHAR | *Purpose*: Sends keyboard character messages retrieved by **GET INPUT** to the serial port.<br><br>Notes: Call WriteFile in overlapped mode for asynchronous I/O. |
| GET CHAR | *Purpose*: Continuously loops to read characters from the serial port.<br><br>Notes: This should be implemented as a thread. Call ReadFile in overlapped mode for asynchronous I/O. |
| DISPLAY | *Purpose*: Display the characters that are retrieved by **GET CHAR** onto the screen.<br><br>Use GetTextExtentPoint32 API to get width of individual characters being read from the buffer, in order to get better text alignment, and allow word wrapping. |
| **HELP** | *Purpose:* A simple dialog with instructions on how to use the app. Implemented as a pop-up dialog box. Click "OK" to close the dialog to return to Command or Connect. |
| **EXIT** | *Purpose*: Allow users to quit our program. Implemented as a menu option in File. |

## PSEUDOCODE:

### **Command**:

**if** user clicks Connection Parameters:

Show comm dialog, and get/set connection parameters from user (GetCommParameters function)

**if** user clicks Connect:

setup the COM port (SetupComm function)

create read thread (CreateThread function)

monitor write

**if** user clicks Disconnect:

destroy read thread

do not monitor write

### **Connect**:

*Monitor write:*

**Loop**

**Get Input:** Get char from user input (using WM_CHAR messages)

**Send Char:** Send char to serial connection (WriteToSerial

**if** Disconnect/Exit pressed, **End Loop**

*Read thread:*

Create input thread (CreateThread)

**Loop**

**Get Char:** Get char from serial connection (WriteFile function)

**Display:** Paint char to screen (PrintToScreen function)

**if** Disconnect/Exit pressed, **End loop**

### **Help**:

Display popup dialog

**if** user clicks OK on the popup dialog:

dismiss the dialog

### **Exit**:

Send exit command to program

## MENU:

***File (Dropdown menu)***

→ Connect

→ Disconnect

→ Exit

***Ports (Dropdown menu)***

→ COM1

→ COM2

→ COM3

→ COM4

→ COM5

***Communication Parameters (Dialog box popup)***

***Help (Message box popup)***

## TESTING GUIDE:

| | |
|---|---|
| **UI** | - User facing interface works (buttons, settings, dialogs)<br>- User activity flow is intuitive; Users navigate around your program effectively (User experience)<br>- Characters are displayed nicely on the screen (no overlapping characters, each character is allocated the proper width)<br>- Text properly wraps onto the next line if it extends past the window width |
| **Functionality** | - Reading from serial port works<br>  - Proper characters are displayed<br>  - Characters are displayed as soon as they are typed without delay<br>  - Characters are interpreted properly ('a' is interpreted as an 'a')<br>- Writing to serial port works<br>  - Characters typed appear on the opposing terminal program<br>  - Characters are displayed as soon as they are typed without delay<br>  - Characters are interpreted properly<br>- Able to open comm parameter dialog<br>  - Comm dialog shows current port settings<br>  - Comm dialog setting changes are persistent, and apply to the current comm connection<br>- Able to switch the ports to use<br>- Able to exit the program |
| **Robust** | - Able to make simultaneous sessions without failure (Thread stopped and restarted properly, handles re-initialized properly)<br>- Able to continuously read / write without errors (characters are not missed or delayed) |