|  |  |
| --- | --- |
|  | **Wireless -GRVM Application** |
|  | Mateusz Siwoski A00758640  Robin Hsieh A00657820  German Villarreal A00839611  Vincent Lau A00 |

|  |
| --- |
| **[Implementation of the (BC) Protocol]** |
|  |



Contents

[Introduction: 2](#_Toc372213496)

[Gantt Chart: 3](#_Toc372213497)

[State Diagram: 5](#_Toc372213498)

[Improved: 5](#_Toc372213499)

[Original: 6](#_Toc372213500)

[Function Prototypes: 7](#_Toc372213501)

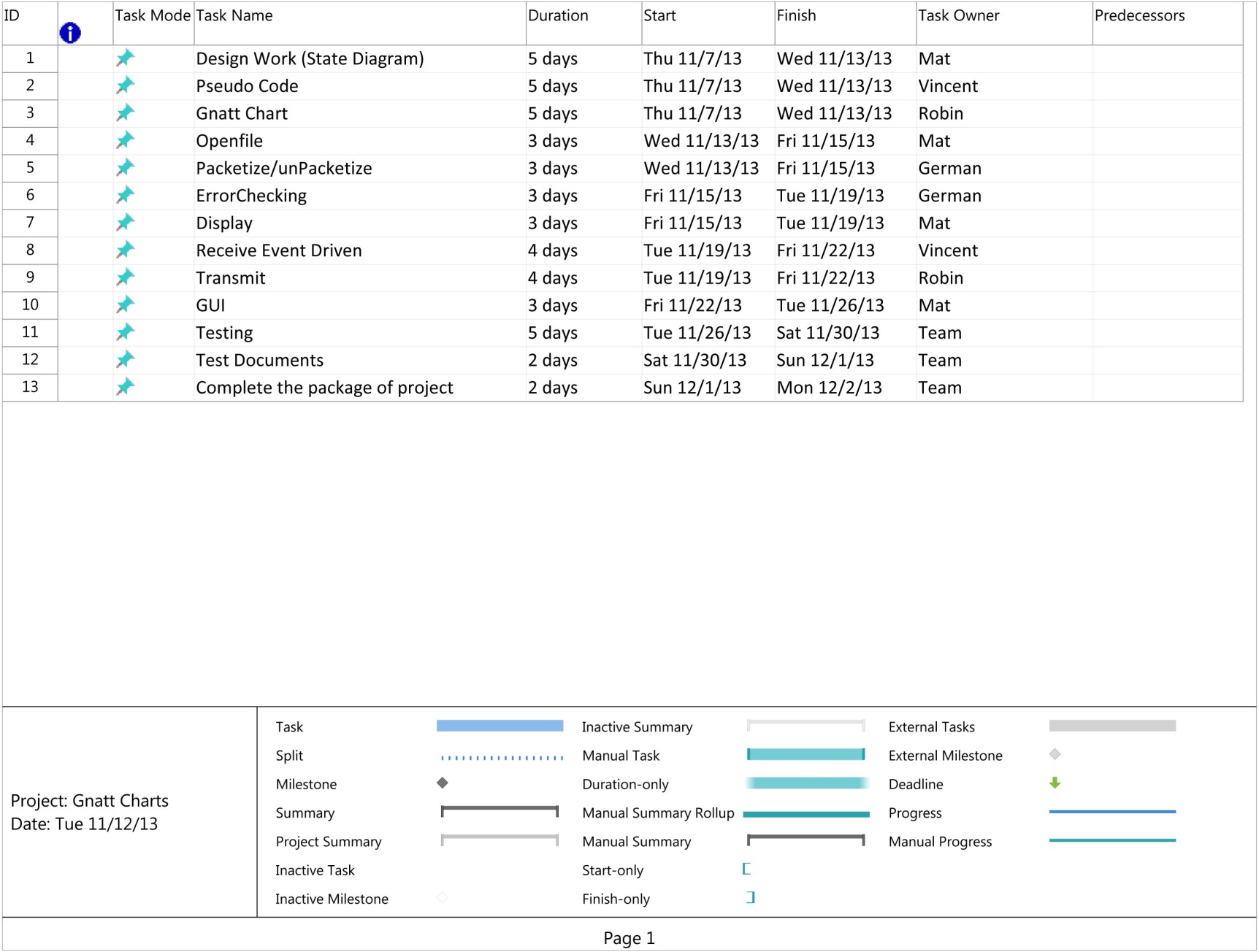
[Pseudocode: 8](#_Toc372213502)

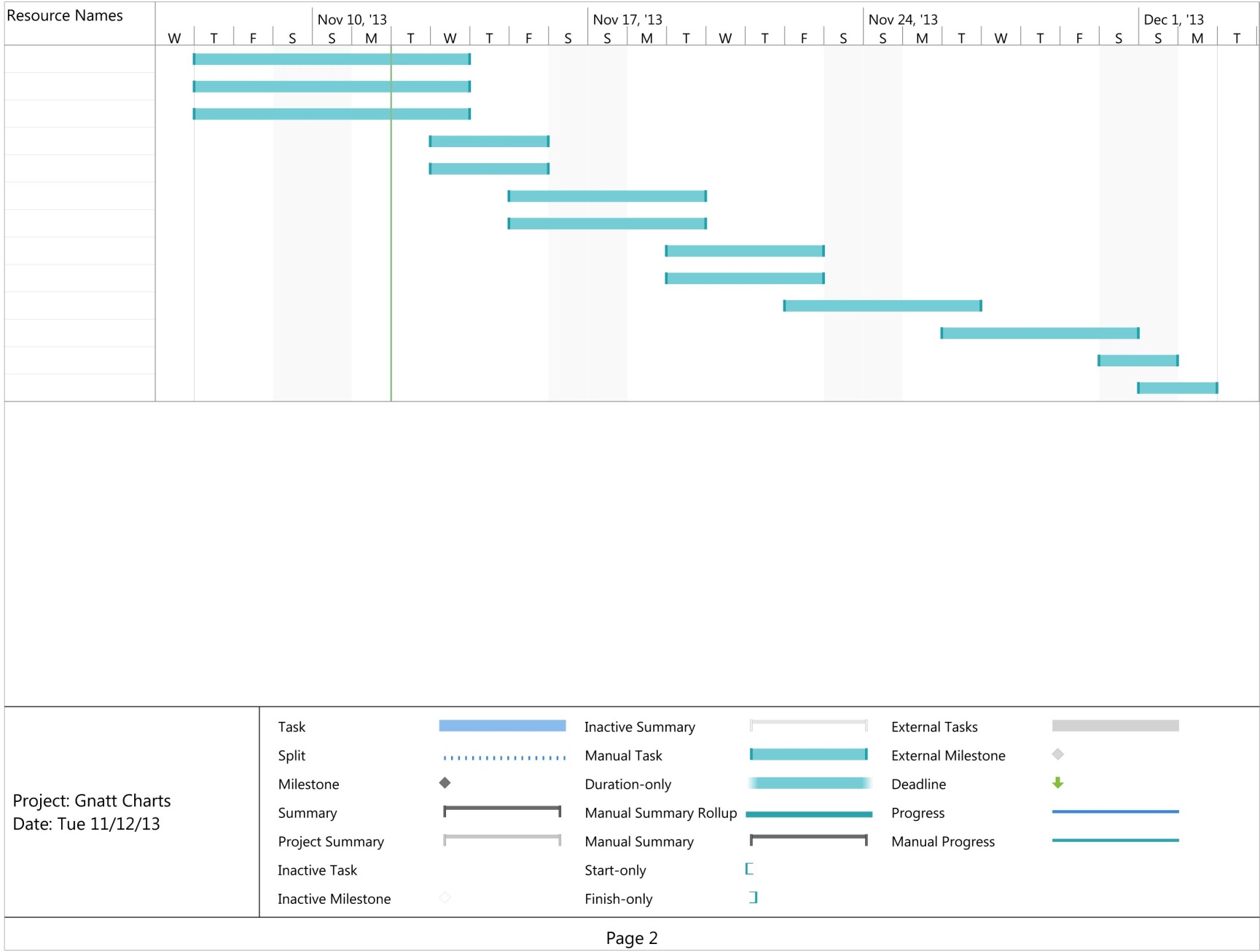
# Introduction:

This is a Win32 application that implements a wireless protocol for transmitting text characters from one device to another and displays the information on a computer monitor. This program is fully event driven and will work as a half-duplex.

The protocol used for this program is the (Be Creative) Protocol and will be run on Windows 7. The designers/programmers are Mateusz Siwoski, Robin Hsieh, German Villarreal and Vincent Lau.

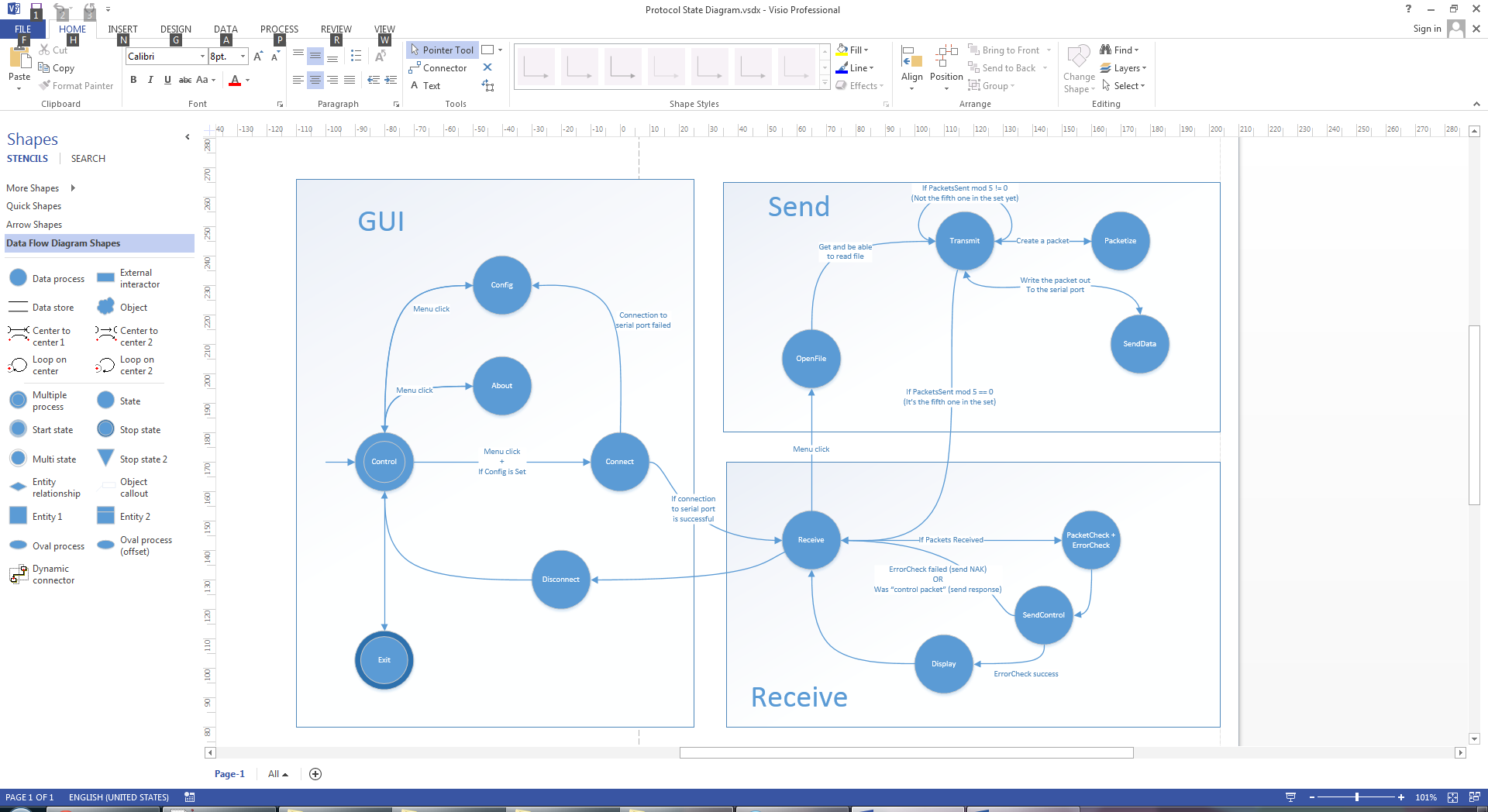
# Gantt Chart:



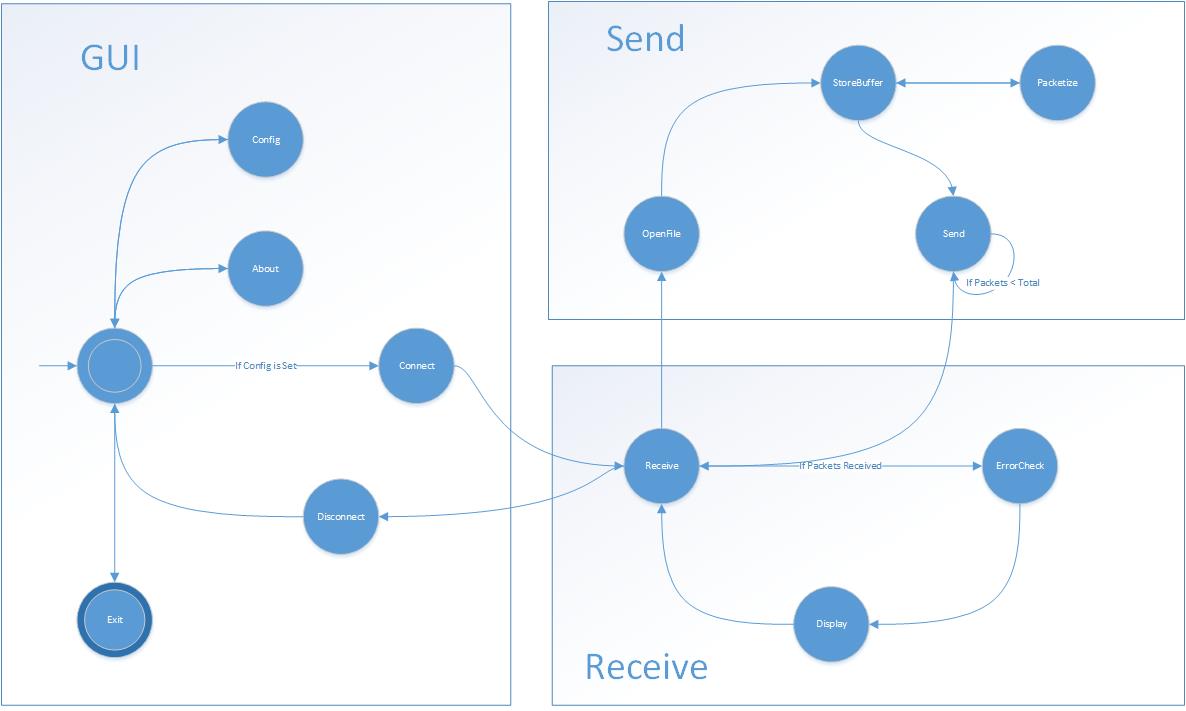


# State Diagram:

## Improved:



## Original:



Function Prototypes:  
  
INT WINAPI WinMain (HINSTANCE hInst, HINSTANCE hprevInstance, LPSTR lspszCmdParam, int nCmdShow);

BOOL Register(HINSTANCE hInst);

HWND Create(HINSTANCE hInst, int nCmdShow)  
LRESULT CALLBACK WndProc (HWND hwnd, UINT Message, WPARAM wParam, LPARAM lParam)  
BOOL Window\_OnCreate(HWND hwnd)  
void Window\_OnCommand (HWND hwnd, int id, HWND hwndCtl, UINT codeNotify)   
void Window\_OnDestroy (HWND);  
BOOL CALLBACK AboutDlgProc (HWND, UINT, WPARAM, LPARAM);  
void OpenFileInitialize(HWND )  
BOOL FileOpenDlg(HWND, PTSTR, PTSTR)  
BOOL FileRead(HWND, PTSTR)  
void OkMessage(HWND, TCHAR\*, TCHAR\*)  
BOOL disconnect(HWND hwnd);  
BOOL connect(HWND hwnd);  
DWORD WINAPI receive(LPVOID lphwnd);  
BOOL PacketCheck(HWND hwnd, char[1024] packet)  
BOOL Errorcheck(char[1024] packet);  
VOID display(HWND hwnd, char[1024] packet);  
VOID Transmit (File \*bufferWithFile) ;  
CHAR\* packetize(File \*bufferWithFile, int SentPacketCounter);  
VOID sendDataPacket (char[1022] data, char DC\_TYPE);  
VOID sendCtrlPacket (char CTRL\_TYPE);

# Pseudocode:

INT Window()

{

Register()

Create()

Set variables for commconfig

MessageLoop()

}  
BOOL Register(){

Register variables for opening main window.

}  
HWND Create{

CreateWIndow

}

LRESULT CALLBACK WndProc(){

Handle different Messages of the window

}  
BOOL Window\_OnCreate(){

OpenFileIntialize();  
 Return True

}  
void Window\_OnCommand(){

Connect()

-SendFile()

If FileOpenDialog is True

If FileRead is False

OkMessage (displays file read error)

Disconnect()

Config()

About()

Exit()

Window\_OnDestroy

}

Void Window\_OnDestroy{

CloseStream()

CloseThreads()

Exit Program

}

BOOL AboutDlgProc{

Open a DialogBox that displays information about the Program

}

void openFileInitialize{

initialize the parameters for opening a file.

}

FileOpenDialog(){

Initialize types of files to be seen

Open the file dialog

}

Void OkMessage(){  
 print the filename that failed to open

}

BOOL FileRead(){

Create a file

Get the file size

If Filesize does not equal zero

Malloc memory space for the file

Read the file and append two Null characters to the end

}

BOOL disconnect(){

CloseStream()

CloseThreads()

}

BOOL Connect ()

{

if ("configured" flag not true)

{

error message : "Port not configured, please configure"

return;

OR

if (!config())

return;

}

clear any port handles or file descriptors that may be in use

get handle to serial port

if (handle is invalid)

{

error message : "Cannot open serial port"

return false;

}

set a "want to read" flag

createReceiveThread();

return true;

}

//RECEIVE

DWORD WINAPI Receive()

{

create temporary packet buffer to save 1024 bytes (1 packet)

set our listening/read parameters for the serial port, we want CHARACTER events (eg SetCommMask)

while (we want to read)

{

if (waiting for event success)

{

if (the event triggered was a CHARACTER event)

{

read 1024 characters into temporary packet buffer

// CRIT SECTION

PacketCheck(packet buffer);

// END CRIT SECTION

}

}

}

}

BOOL PacketCheck (char[1024] packet)

{

switch (char[1])

{

case: ENQ:

send (ACK);

Set "what we're waiting for" flag to PACKET\_DC1

break;

case DC1:

if ("what we're waiting for" is a PACKET\_DC2)

{

send (NAK);

break;

}

if (!ErrorCheck(char[1022], char[1023]))

{

send (NAK);

break;

}

send (ACK);

Display();//read the remaining 1020 characters

break;

case DC2:

if ("what we're waiting for" is a PACKET\_DC1)

{

send (NAK);

break;

}

if (!ErrorCheck(char[1022], char[1023]))

{

sendControlPacket (NAK);

break;

}

send (ACK);

Display();//read the remaining 1020 characters

break;

case NAK:

Set "What we're waiting for" flag to ACK

send (previous packet); //need a way to keep that

break;

case EOT:

// GO back to IDLE state

Set "what we're waiting for" flag to ENQ

break;

}

}

BOOL ErrorCheck(char[1024] packet){

GenerateTable(){

generate a table

calculate the CRC table

}

get(begin, end){

accumulate ()

}

}

VOID display(head)

{

print data

}

BOOL openFile(){

Initialize OpenFile struct

GetFileAttributes();

if (GetOpenFileName()){

Transmitfile();

return true;

}

return false;

}

/\*if data is sent\*/

VOID Transmit (File \*bufferWithFile)

{

create sentPacketCounter = 0

create packetToSend

do

{

// giving the packetize function the sentPacketCounter allows it to skip through the file

// if necessary. It can also determine if the next data packet will be DC1 or DC2 (mod2!=0)

while (packetCounter mod 5 != 0)

{

packetToSend = packetize (File Handle, sentPacketCounter, PKT\_TYPE\_DATA)

// CRIT SECTION

sendDataPacket (packetToSend);

decrement semaphore

++packetCounter;

waitforSemaphore

set "what we're waiting for" flag to ACK

// END CRIT SECTION

}

Wait for ENQ

} while (file not done)

}

CHAR\* packetize(File \*bufferWithFile, int SentPacketCounter)

{

1020 x sentPacketCounter = startingLocation

Read 1020 chars from the file buffer, starting at startingLocation into packet string

If we encounter eof

Pad remains Bytes with null

If (sentPacketBuffer % 2 == 0)

Packet[1] = DC1

Else

Packet[1] = DC2

//create return string returnstr

//add control bytes to returnstr

//if s[i] != eof

// returnstr += s[i]

//

// while i != 1022

// returnstr += '\0'

// returnstr += trailer bytes

// return returnstr

}

VOID sendDataPacket (char[1022] data, char DC\_TYPE?)

{

char[1024] packet

set char[0] to SYN

set char[1] to DC\_TYPE // DC1 or DC2

append data and CRC

write to serial port

}

VOID sendCtrlPacket (char CTRL\_TYPE)

{

char[2] packet

set char[0] to SYN

set char[1] to CTRL\_TYPE

write to serial port

}

Test Documents