Wireless - GRVM Application

Mateusz Siwoski A00758640
Robin Hsieh A00657820
German Villarreal A00839611
Vincent Lau A00758190

[IMPLEMENTATION OF THE (BC) PROTOCOL]

Abstract

The Wireless-GRVM application is a wireless protocol that functions on serial. Using serial communication to adhere to the protocol introduces issues such as no actual established connection as well as security concerns. The results were as unexpected as usual, as the wireless medium has a lot of interference which brought about complications. Testing occurred on both wired and wireless and, as expected, this application is more reliable when it comes to wired connections.

Contents

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 application will have two states for transmission, Send and Receive. The program will be sending and receiving text files and will display the file upon receiving 5 bytes of data. To ensure correct information is being sent, the program uses the CRCITT

The designers/programmers are Mateusz Siwoski, Robin Hsieh, German Villarreal and Vincent Lau.

Description

This program is implemented to communicate with other computers using a newly specified wireless protocol. By transmitting data through the serial port to a wireless modem; this program is able to send files to other connected computers. Computers connect to each other and will start listening for another computer; once another computer is found a connection is established and they are able to transmit files.

Users must connect to each other and specify the file to send. If a proper connection was established and the other computer is also strictly following our protocol the file will transmit flawlessly.

The protocol used to transmit files is more thoroughly covered in the *Features* section.

Features

The features of this program are heavily based on the wireless protocol we have designed. Our program strictly follows the protocol and thus allows us to communicate with others through a half-duplex wireless connection. After establishing a connection and bidding for the line, the file to send is packetized and sent; up to a maximum of 5 packets are sent, the line is dropped by this sender, allowing the other computer to bid for the line and send its 5 packets. This pattern continues until they are both sending.

Our program handles all possible erroneous cases and treats them accordingly depending on the communication medium. When both computers try to send at the same time they are able to properly communicate and let one of them proceed to send first. Sent packets are also verified by the receiver using CRCITT to ensure no packets have arrived with no corrupt bits or out of order. If an error has occurred the erroneous packet must be resent.

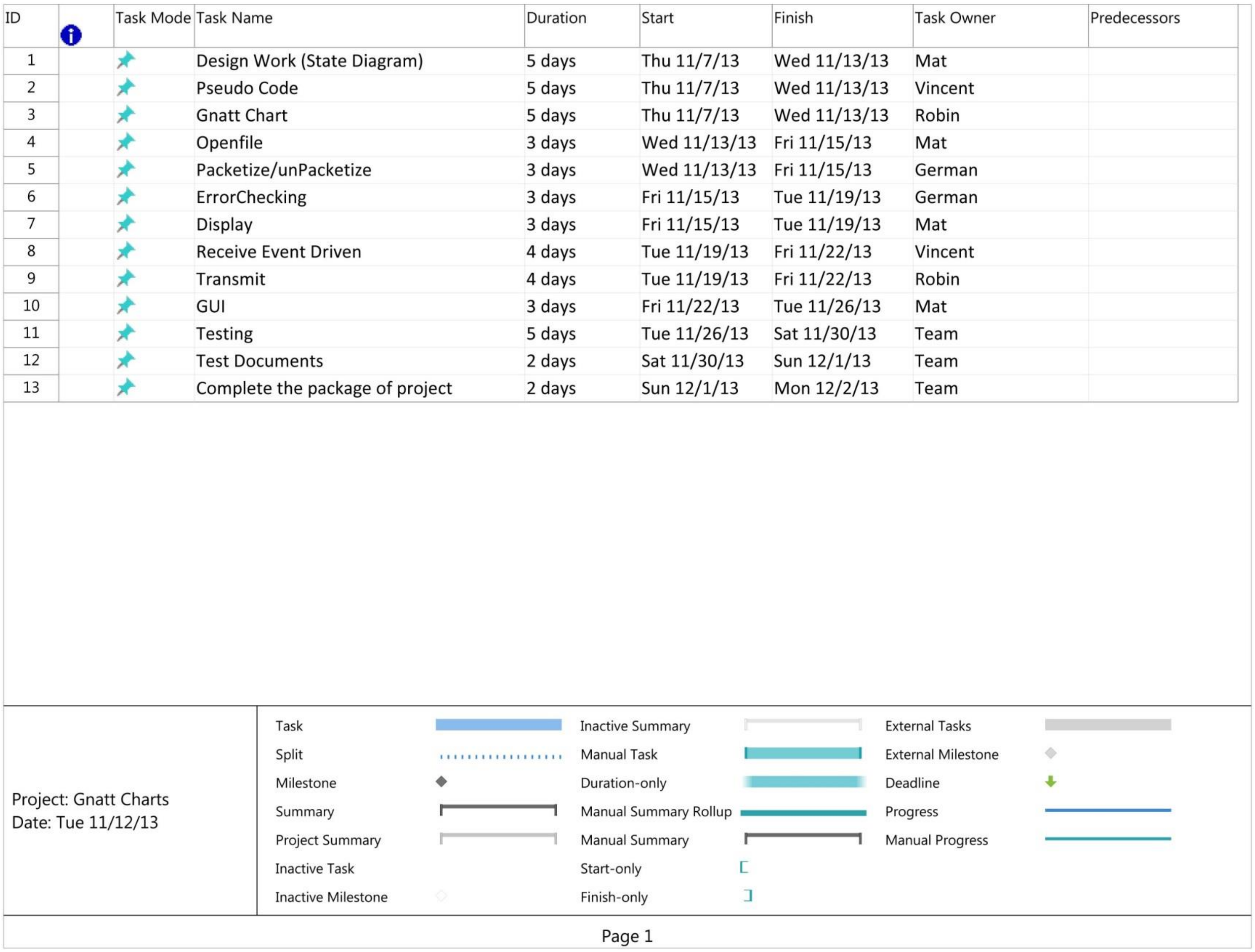
Issues Experienced

The major issue we have encountered has been corruption over the wireless connection. Packets do not seem to be fully sent and often arrive in with errors due to high interference. When testing through a wireless medium the throughput drops drastically; however, when two computers are connected via serial cable all packets arrive timely and with precision providing optimal throughput and the maximum efficiency while strictly adhering to the established protocol.

We encountered some problems as well when attempting to keep the file to send active in memory. When data was to be sent out on the line there were instances that created problems in attempting to packetize the proceeding data; and then also adjusting for when the CRC failed on the receiver's side and we were forced to resend the data.

Determining the logic in the receive thread also caused many problems with the sequence in which different packets were expected to arrive and the actual sequence in which they arrived. Sometimes corrupt data was kept in the serial port buffer which conflicted with what was expected causing repetitive NAK's to be sent.

Gantt Chart

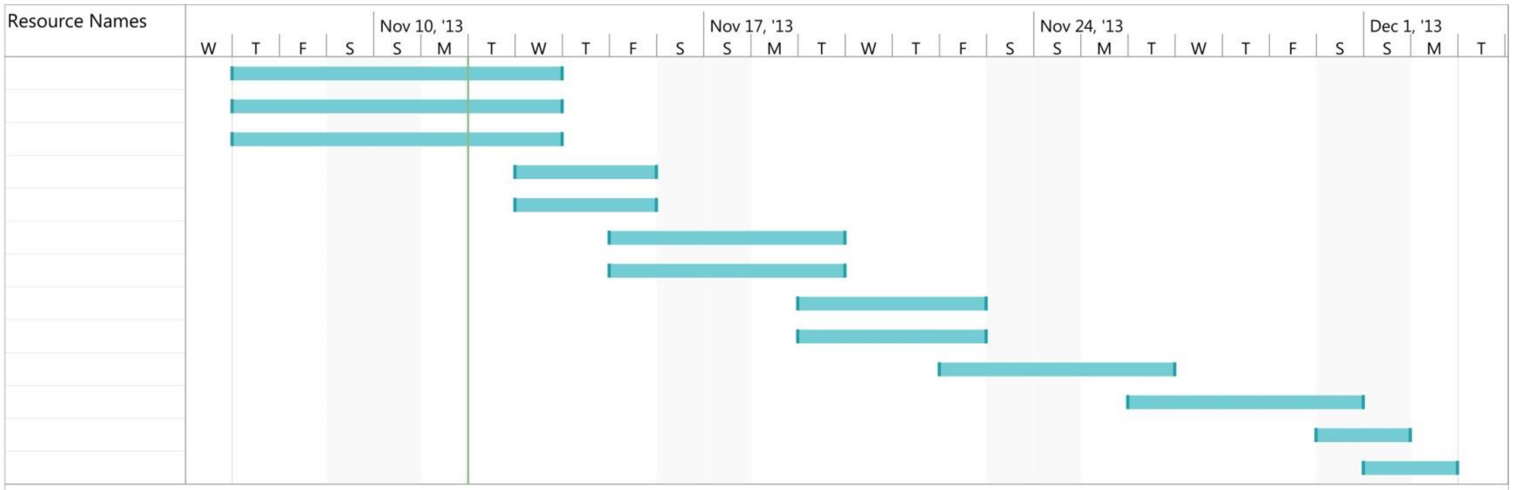


ID	ℹ	Task Mode	Task Name	Duration	Start	Finish	Task Owner	Predecessors
1		📌	Design Work (State Diagram)	5 days	Thu 11/7/13	Wed 11/13/13	Mat	
2		📌	Pseudo Code	5 days	Thu 11/7/13	Wed 11/13/13	Vincent	
3		📌	Gnatt Chart	5 days	Thu 11/7/13	Wed 11/13/13	Robin	
4		📌	Openfile	3 days	Wed 11/13/13	Fri 11/15/13	Mat	
5		📌	Packetize/unPacketize	3 days	Wed 11/13/13	Fri 11/15/13	German	
6		📌	ErrorChecking	3 days	Fri 11/15/13	Tue 11/19/13	German	
7		📌	Display	3 days	Fri 11/15/13	Tue 11/19/13	Mat	
8		📌	Receive Event Driven	4 days	Tue 11/19/13	Fri 11/22/13	Vincent	
9		📌	Transmit	4 days	Tue 11/19/13	Fri 11/22/13	Robin	
10		📌	GUI	3 days	Fri 11/22/13	Tue 11/26/13	Mat	
11		📌	Testing	5 days	Tue 11/26/13	Sat 11/30/13	Team	
12		📌	Test Documents	2 days	Sat 11/30/13	Sun 12/1/13	Team	
13		📌	Complete the package of project	2 days	Sun 12/1/13	Mon 12/2/13	Team	

Project: Gnatt Charts
Date: Tue 11/12/13

Task	▬	Inactive Summary	▭	External Tasks	▬
Split	┄	Manual Task	▬	External Milestone	◆
Milestone	◆	Duration-only	▬	Deadline	⬇
Summary	┌┐	Manual Summary Rollup	▬	Progress	─
Project Summary	┌┐	Manual Summary	┌┐	Manual Progress	─
Inactive Task		Start-only	[		
Inactive Milestone	◇	Finish-only	]		

Page 1

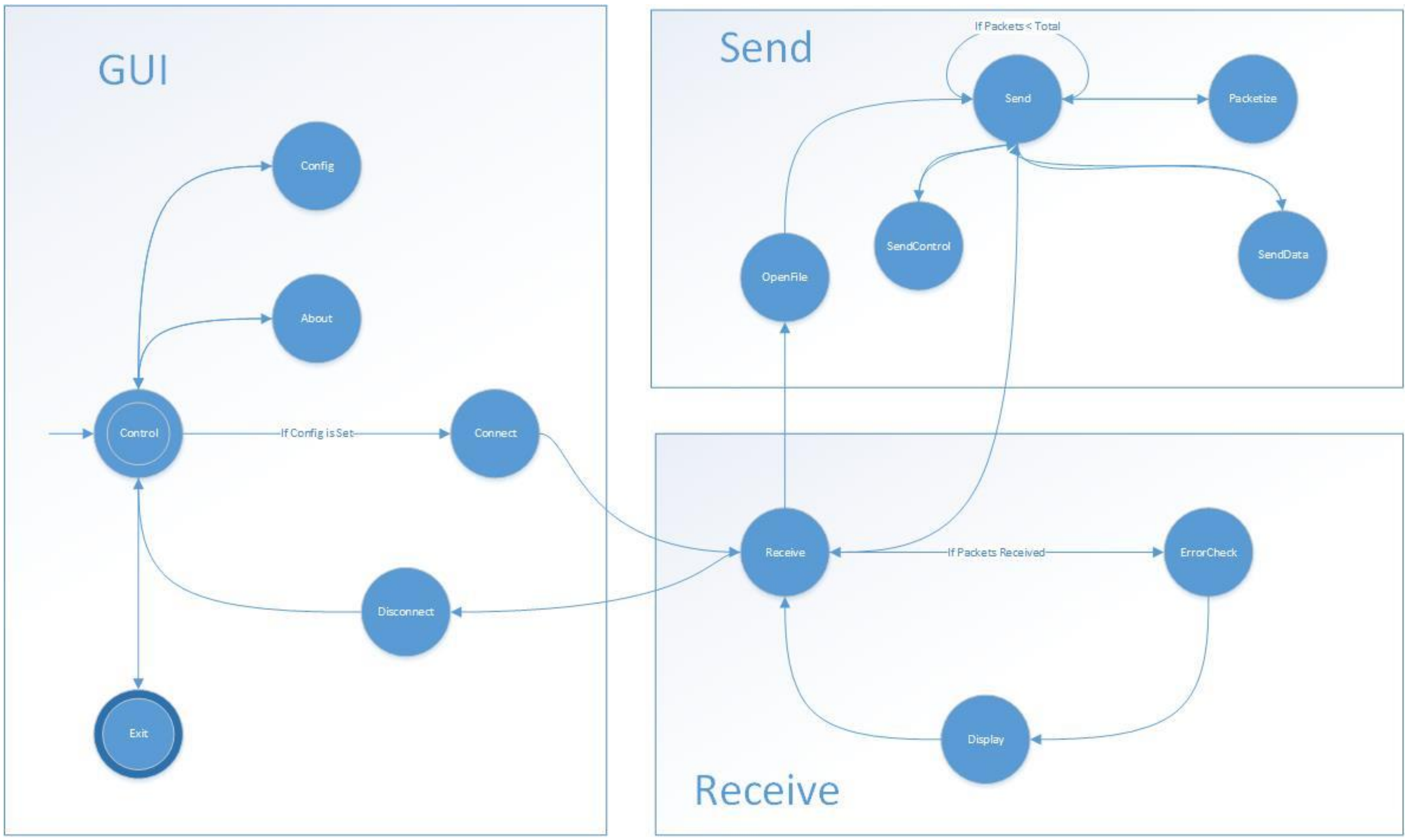


Project: Gnatt Charts Date: Tue 11/12/13	Task	▬	Inactive Summary	▭	External Tasks	▬
	Split	┈	Manual Task	▬	External Milestone	◆
	Milestone	◆	Duration-only	▬	Deadline	⬇
	Summary	┌┐	Manual Summary Rollup	▬	Progress	─
	Project Summary	┌┐	Manual Summary	┌┐	Manual Progress	─
	Inactive Task		Start-only	[		
	Inactive Milestone	◇	Finish-only	]		

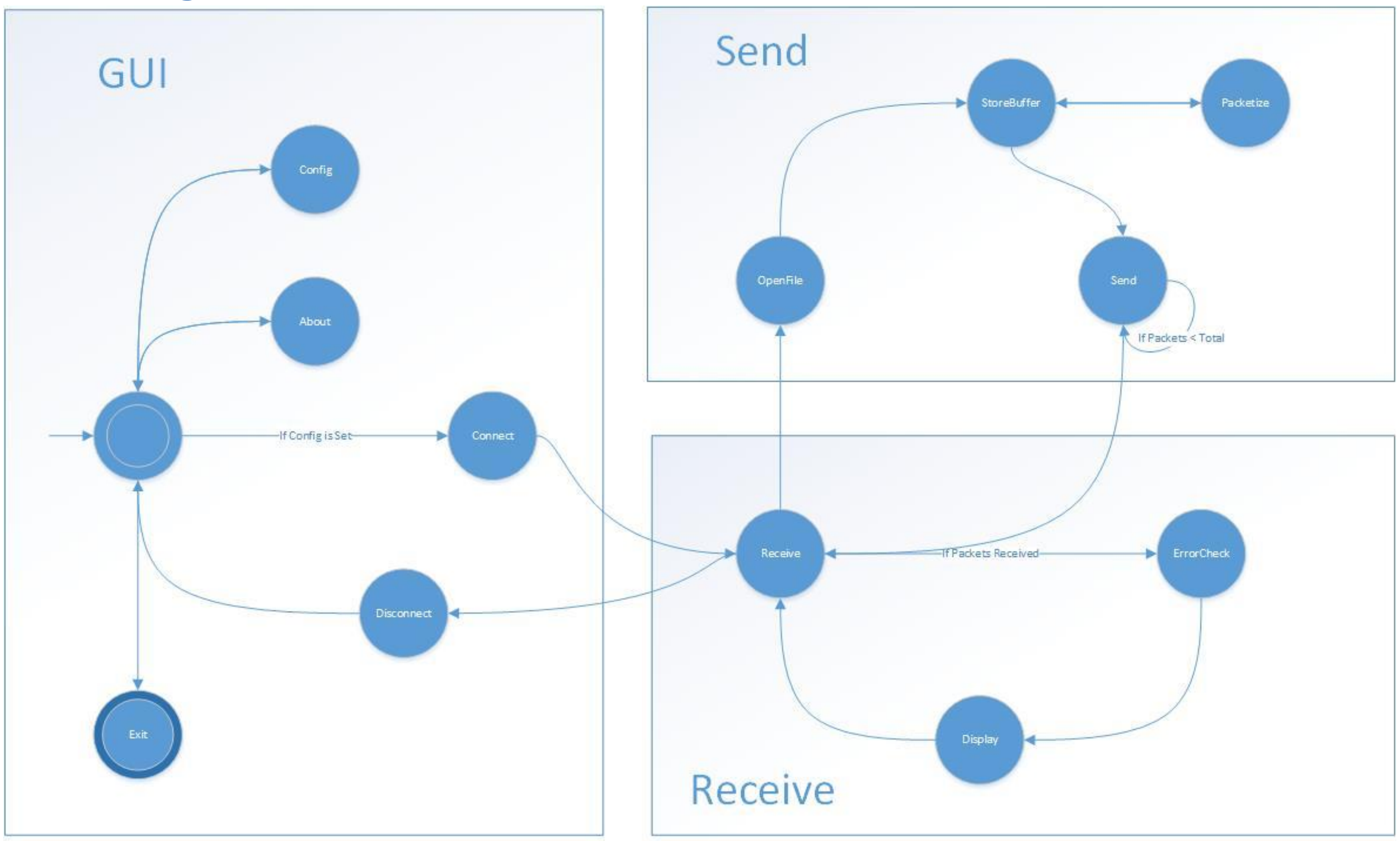
Page 2

State Diagram

Improved:



Original:


GUI
Config
About
If Config is Set
Connect
Disconnect
Exit
Send
StoreBuffer
Packetize
OpenFile
Send
If Packets < Total
Receive
If Packets Received
ErrorCheck
Display
Receive

Function Prototypes

ErrorCheck

```
unsigned short crc16(char, unsigned short);
void GenerateCRC(char*, char*);
BOOL ErrorCheck(char*);
```

Main

```
LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
BOOL Register(HINSTANCE);
HWND Create(HINSTANCE, int);
BOOL Window_OnCreate(HWND, LPCREATESTRUCT);
void Window_OnCommand (HWND, int, HWND, UINT);
void Window_OnDestroy (HWND);
BOOL CALLBACK AboutDlgProc (HWND, UINT, WPARAM, LPARAM);
void OpenFileInitialize(HWND);
BOOL FileOpenDlg (HWND, PTSTR, LPCSTR);
BOOL FileRead(HWND, const LPCSTR);
void OkMessage(HWND, TCHAR*, TCHAR*);
BOOL ErrorCheck(char*);
void GenerateCRC(char*, char*);
void DisplayText(HWND, LPCSTR);
void Window_OnVScroll(HWND, HWND, UINT, int);
void Window_OnPaint(HWND);
void Window_OnSize(HWND, UINT, int, int);
BOOL FileSave(HWND, LPCTSTR);
BOOL FileSaveDlg (HWND, PTSTR, PTSTR);
```

Packet

```
BOOL Packetize(CHAR*, int);
BOOL PacketCheck(HWND, CHAR*);
BOOL PacketCheckControl(HWND hwnd, CHAR* packet);
void GetData(CHAR*, CHAR*);
```

Physical

```
BOOL SendControl(HANDLE, int);
LONG_PTR SendData(HANDLE hComm, char* packetToSend);
BOOL ReadSerialPortControl(HANDLE hComm, char* packetBuffer, DWORD dwBytesToRead, LPDWORD
lpdwBytesRead);
BOOL ReadSerialPortData(HANDLE hComm, char* packetBuffer, DWORD dwBytesToRead, LPDWORD
lpdwBytesRead);
```

Presentation

```
BOOL AddToBuffer(const char*);
```

Session

```
BOOL SetupPort (LPTSTR);
BOOL ConfPort (HWND*, LPTSTR);
```

Transport

```
DWORD WINAPI TransmitThread(LPVOID);
DWORD WINAPI ReceiveThread  (LPVOID);
```

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()
        If Com Port was OK and not in use
            -Open Receive Thread
    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 Config ()
{
  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)
         {
           If there is 1024 chars to read
                 read 1024 characters into temporary packet buffer
                 packetcheck()
           If there is 2 chars to read
                 read 2 characters into temporary packet buffer
                 packetcheck()
         }
      }
   }
}

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)
      sendDataPacket (packetToSend);
        decrement semaphore
      ++packetCounter;
      set "what we're waiting for" flag to ACK
      waitforSemaphore
        if semaphore timed out
                resend Packet
    }
   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
}
```

Conclusion

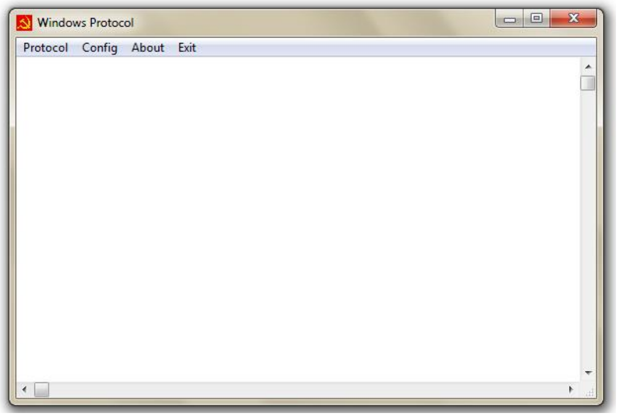
During our hours testing we have concluded that sending over a wireless medium is far less reliable for the receiver to receive the appropriate data; in our tests, retransmission rates were extremely high due to wireless interference and data corruption. A circular buffer could have helped with our results by more appropriately controlling the transmitting data.

Test Cases

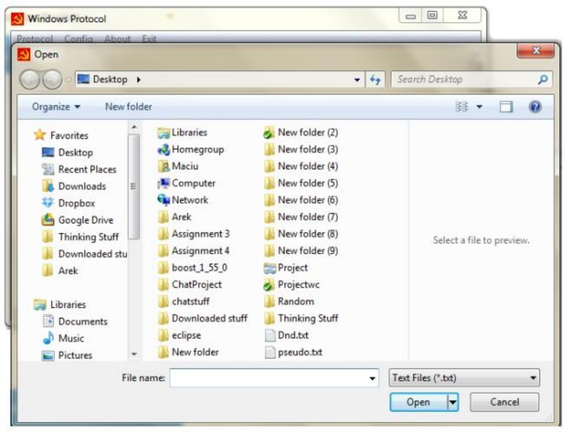
Test	Tests Description	Tools Used	Expected Result	Pass/Fail	Notes
1	Opening the GUI	Wireless-GRVM	The program should open with correct menu options	Pass	See Figure 1
2	Open a File/Display File	Wireless-GRVM	The program should be able to open a .txt and Display the Text	Pass	See Figure 2
3	Threading/CPU usage of program	Wireless-GRVM/Resource Monitor	The program should not use a lot of CPU resources	Fail	See Figure 3
4	Threading/CPU usage of program	Wireless-GRVM/Resource Monitor	The program should not use a lot of CPU resources	Pass	See Figure 4
5	Packet Padding at EOF Successful	Wireless-GRVM/HyperTerminal	The W's display the padding after the EOF was reached	Pass	See Figure 5
6	Packet Sending	Wireless-GRVM/HyperTerminal	Displays an error sent by GRVM to HyperTerminal	Fail	See Figure 6
7	Packet Sending/Receiving	Wireless-GRVM	Received packet is displayed	Pass	See Figure 7

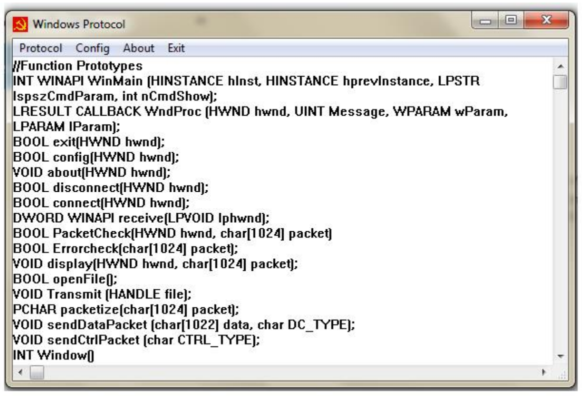
Figures

Test 1: Initial Start Up (PASS)



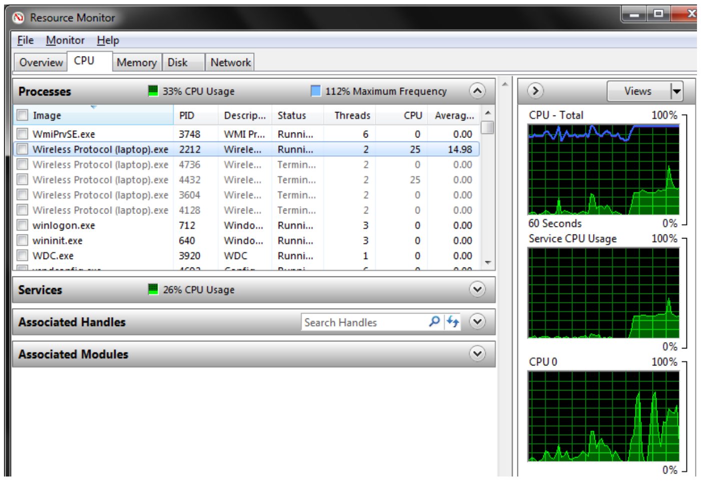
Test 2: Opening and Displaying a File (PASS)





```
//Function Prototypes
INT WINAPI WinMain (HINSTANCE hInst, HINSTANCE hprevInstance, LPSTR
lspszCmdParam, int nCmdShow);
LRESULT CALLBACK WndProc (HWND hwnd, UINT Message, WPARAM wParam,
LPARAM lParam);
BOOL exit(HWND hwnd);
BOOL config(HWND hwnd);
VOID about(HWND hwnd);
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);
BOOL openFile();
VOID Transmit (HANDLE file);
PCHAR packetize(char[1024] packet);
VOID sendDataPacket (char[1022] data, char DC_TYPE);
VOID sendCtrlPacket (char CTRL_TYPE);
INT Window()
```

Test 3: Threading (FAIL)



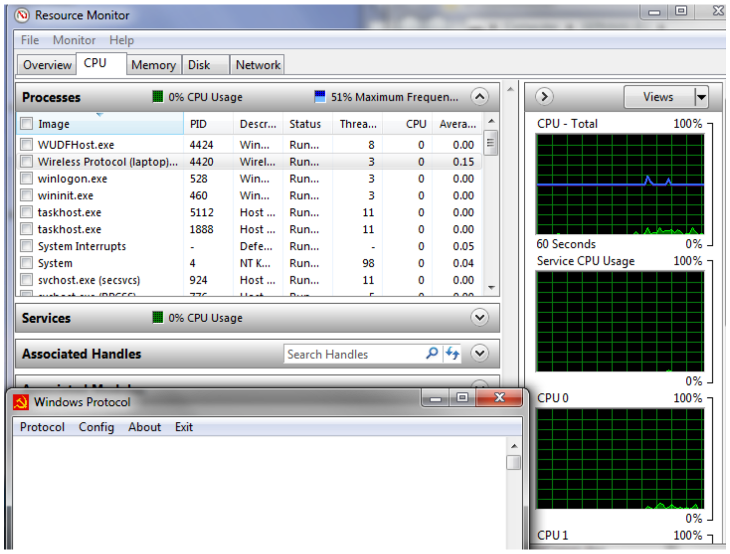
Processes 🟩 33% CPU Usage 🟦 112% Maximum Frequency

Image	PID	Descrip...	Status	Threads	CPU	Averag...
WmiPrvSE.exe	3748	WMI Pr...	Runni...	6	0	0.00
Wireless Protocol (laptop).exe	2212	Wirele...	Runni...	2	25	14.98
Wireless Protocol (laptop).exe	4736	Wirele...	Termin...	2	0	0.00
Wireless Protocol (laptop).exe	4432	Wirele...	Termin...	2	25	0.00
Wireless Protocol (laptop).exe	3604	Wirele...	Termin...	2	0	0.00
Wireless Protocol (laptop).exe	4128	Wirele...	Termin...	2	0	0.00
winlogon.exe	712	Windo...	Runni...	3	0	0.00
wininit.exe	640	Windo...	Runni...	3	0	0.00
WDC.exe	3920	WDC	Runni...	1	0	0.00

Services 🟩 26% CPU Usage

Associated Handles Search Handles

Test 4: Threading (PASS)



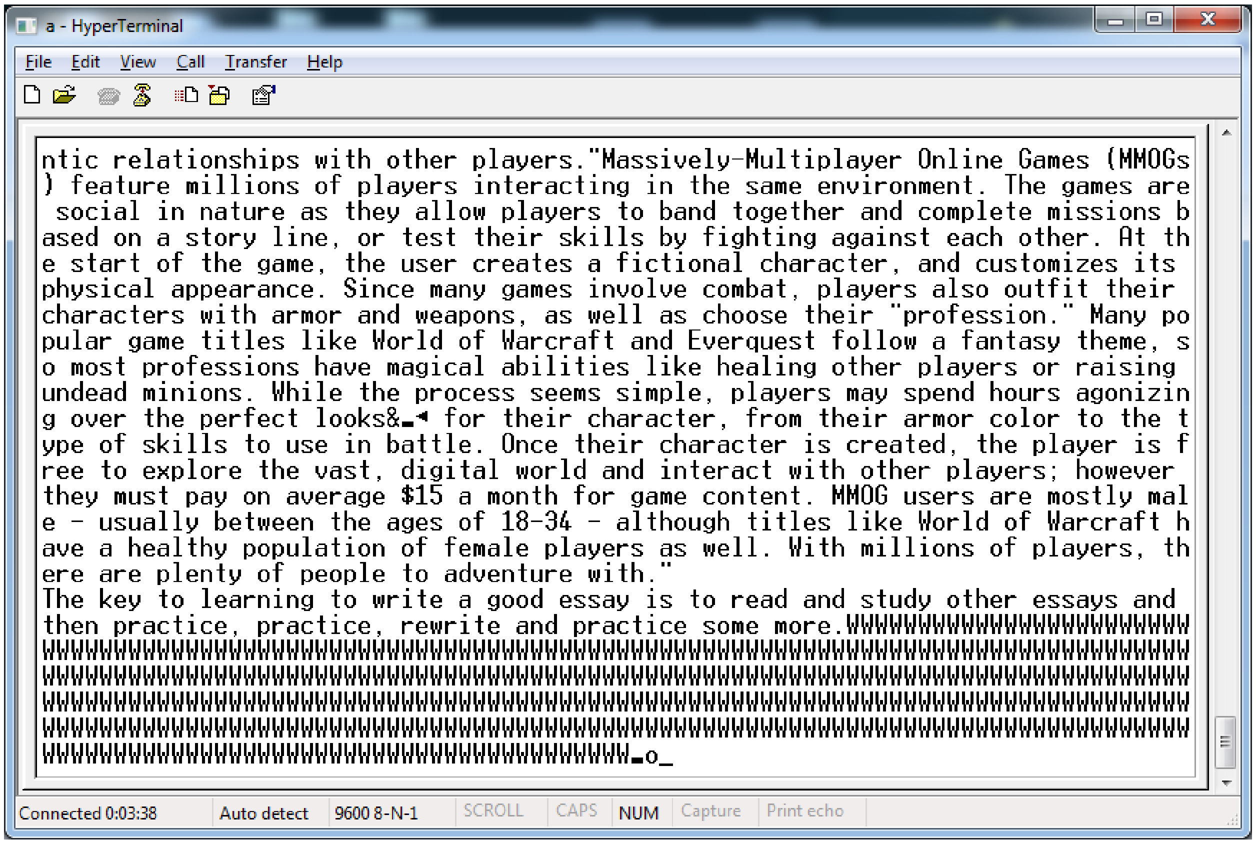
Processes 🟩 0% CPU Usage 🟦 51% Maximum Frequen...

Image	PID	Descr...	Status	Threa...	CPU	Avera...
WUDFHost.exe	4424	Win...	Run...	8	0	0.00
Wireless Protocol (laptop)...	4420	Wirel...	Run...	3	0	0.15
winlogon.exe	528	Win...	Run...	3	0	0.00
wininit.exe	460	Win...	Run...	3	0	0.00
taskhost.exe	5112	Host ...	Run...	11	0	0.00
taskhost.exe	1888	Host ...	Run...	11	0	0.00
System Interrupts	-	Defe...	Run...	-	0	0.05
System	4	NT K...	Run...	98	0	0.04
svchost.exe (secsvcs)	924	Host ...	Run...	11	0	0.00

Services 🟩 0% CPU Usage

Associated Handles Search Handles

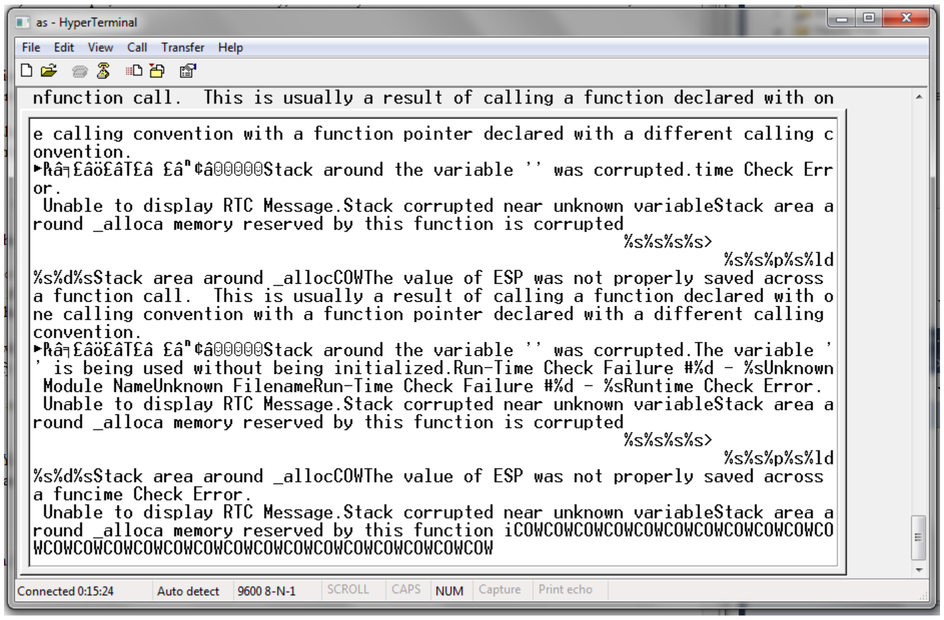
Test 5: (PASS)



```
ntic relationships with other players."Massively-Multiplayer Online Games (MMOGs
) feature millions of players interacting in the same environment. The games are
 social in nature as they allow players to band together and complete missions b
ased on a story line, or test their skills by fighting against each other. At th
e start of the game, the user creates a fictional character, and customizes its
physical appearance. Since many games involve combat, players also outfit their
characters with armor and weapons, as well as choose their "profession." Many po
pular game titles like World of Warcraft and Everquest follow a fantasy theme, s
o most professions have magical abilities like healing other players or raising
undead minions. While the process seems simple, players may spend hours agonizin
g over the perfect looks&▬◄ for their character, from their armor color to the t
ype of skills to use in battle. Once their character is created, the player is f
ree to explore the vast, digital world and interact with other players; however
they must pay on average $15 a month for game content. MMOG users are mostly mal
e - usually between the ages of 18-34 - although titles like World of Warcraft h
ave a healthy population of female players as well. With millions of players, th
ere are plenty of people to adventure with."
The key to learning to write a good essay is to read and study other essays and
then practice, practice, rewrite and practice some more.WWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW
WWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWWW▬o_
```

Connected 0:03:38 | Auto detect | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

Test 6: (FAIL)

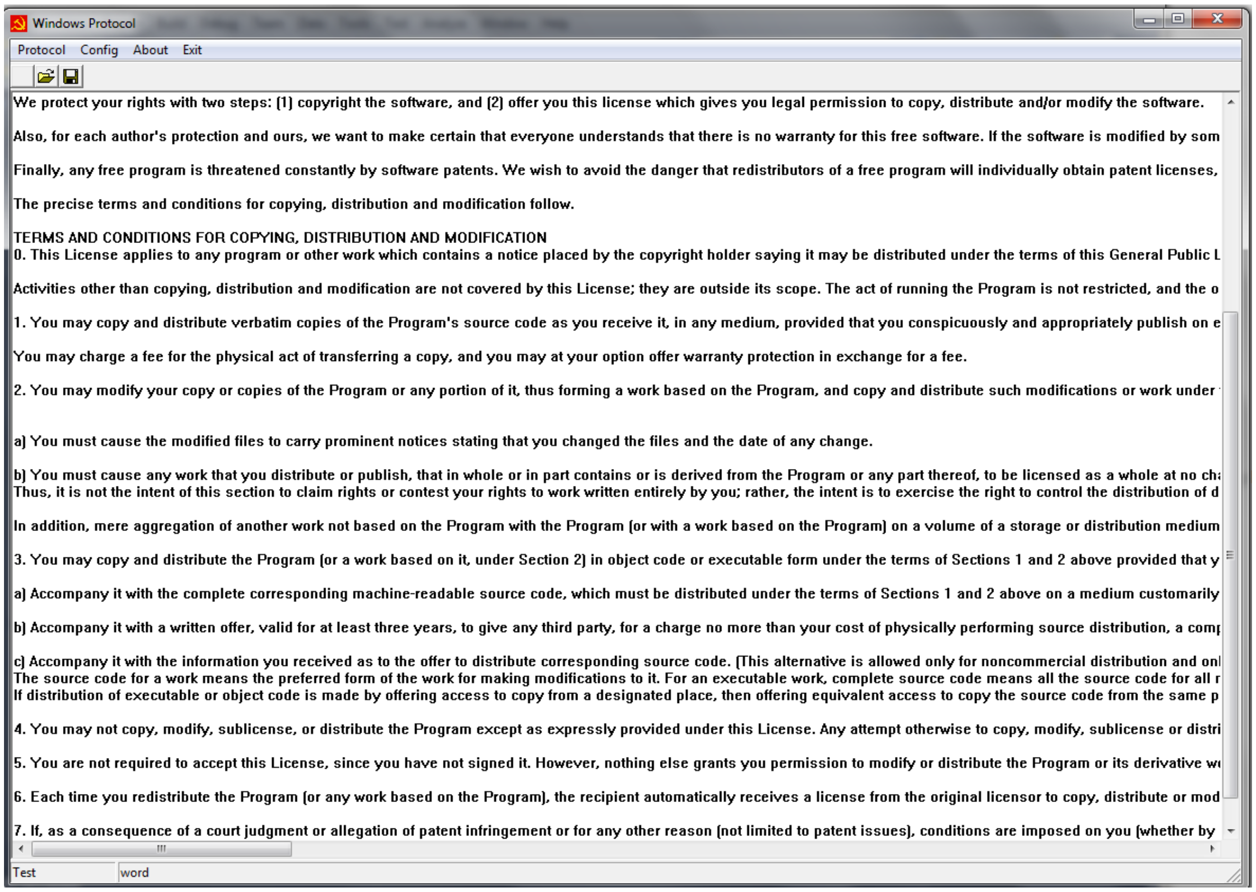


```
nfunction call.  This is usually a result of calling a function declared with on

e calling convention with a function pointer declared with a different calling c
onvention.
►Ŗâ┐£âö£âT£â £â"¢â000000Stack around the variable '' was corrupted.time Check Err
or.
 Unable to display RTC Message.Stack corrupted near unknown variableStack area a
round _alloca memory reserved by this function is corrupted
                                                            %s%s%s%s>
                                                                       %s%s%p%s%ld
%s%d%sStack area around _allocCOWThe value of ESP was not properly saved across
a function call.  This is usually a result of calling a function declared with o
ne calling convention with a function pointer declared with a different calling
convention.
►Ŗâ┐£âö£âT£â £â"¢â000000Stack around the variable '' was corrupted.The variable '
' is being used without being initialized.Run-Time Check Failure #%d - %sUnknown
 Module NameUnknown FilenameRun-Time Check Failure #%d - %sRuntime Check Error.
 Unable to display RTC Message.Stack corrupted near unknown variableStack area a
round _alloca memory reserved by this function is corrupted
                                                            %s%s%s%s>
                                                                       %s%s%p%s%ld
%s%d%sStack area around _allocCOWThe value of ESP was not properly saved across
a funcime Check Error.
 Unable to display RTC Message.Stack corrupted near unknown variableStack area a
round _alloca memory reserved by this function iCOWCOWCOWCOWCOWCOWCOWCOWCOWCOWCO
WCOWCOWCOWCOWCOWCOWCOWCOWCOWCOWCOWCOWCOWCOW
```

Connected 0:15:24 | Auto detect | 9600 8-N-1 | SCROLL | CAPS | NUM | Capture | Print echo

Test 7: (PASS)



We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by som

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses,

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION
0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public L

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the o

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on e

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no ch
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of d

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that y

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a com

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and onl
The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all r
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same p

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distri

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative w

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or mod

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by