# **Driver Pseudocode**

### **BuildWindow – WinMain**

Initialize window

Set window dimensions

Populate window with the several items:

File loader open button

Send button

Save Button

Priority Heiarchy Checkbox

Editable text areas for files to be sent

Non-editable text areas for files received

List view box that displays files received

Progress bar to keep track of the files received

Non-editable text area for statistics of each file

Define file opener structure

Assign file opener structure a parent window

**Transition** to **Enter Comm Param**

### **Enter Comm Param**

Bring up comm dialog to allow user to set parameters

When user clicks OK

Get the parameters from the comm dialog

Set the parameters for the wireless modem

if read thread not started

**transition** to **Engine Read Thread Start**

else

**transition** to **Command Mode**

### **Command Mode – WndProc**

### Waits for window input, such as button presses.

If user presses **Open** b

**transition** to **File Loader**

If user presses **Send**

if user has selected a file to send

**transition** to **Packetize Data**

else

prompt user to select a file first

### **File Loader**

User can select a file to send

if user selects and file and presses OK

update and display content in GUI

**transition** to **Command Mode**

### **Packetize Data**

Check the size of the file that was passed in

Set buffer size to 516 bytes

**Loop:** Iterate through file to packetize:

Set first byte in buffer to **SOH**(**0x01**)

Set second byte in buffer to **0 (0x30)** or **1 (0x31)** (alternating

sync bit)

Set 2-byte checksum to third and fourth byte in buffer

if remaining file data is larger than 512 bytes

loop from buffer index 0+4 to 516

add each char to buffer

else

loop from buffer index 0+4 to size of data

add each char to buffer

add **EOT** (**0x04**) after the last char in buffer

store packet into buffer

**Transition** to **Engine Write Thread Start**

### **Engine Write Thread Start**

Write thread is initialized

**Transition** into a continues **Reading** state in the protocol engine

### 

### **Engine Read Thread Start**

Read thread is initialized

**Transition** into **Sending** state in the protocol engine until file transfer complete

### **Idle**

Loops forever

If ENQ or DC2 is received

**transition to Acknowledge Line**

If user input detected

**transition to confirm line**

# **READ – Forever loop**

### **Acknowledge Line**

Set Listener for ENQ and DC2

Wait for event

if ENQ OR DC2 is received

If DC2 received

set **senderHasPriority** to true

if we want priority (based on user input)

transmit DC1

if we don’t want priority

transmit ACK1

set **reading** to true

**transition** to **Wait For Data** state

### **Wait for data**

while reading is true

If timeout

set reading to false

break

go to **Packet Check** state

### **Packet Check**

if **SOH** is detected

if sync byte is alternating

perform **checksum** and if **checksum** passes

send **ACK** or **DC1**

loop through packet message

if we want priority (based on user input)

transmit DC1

if we don’t want priority

transmit ACK1

if **EOT** found

set **reading** to false

**Transition** to **Receiver side: Check**

**Priority State**

else sync byte is not alternating

drop packet

**Transition** to **Wait for Data**

### **Acknowledge Package**

Send ACK or DC1

transition to **Wait For Data**

### **Receiver side: Check Priority State**

if we don’t have priority and they want priority

**transition** to **wait** state

else

**transition** to **idle** state

# 

# **SEND – Ends when file transfer complete**

### **Confirm Line**

if **number of tries >= 5**

**transition** to **Wait**

if we want priority (based on user input)

transmit **DC2**

if we don’t want priority

transmit **ENQ**

**transition** to **Wait for ACK**

### **Wait for ACK**

set listener for ACK and DC1

Wait for Event

Set **timeout** to 5ms

if **timeout** > 5ms

increment **number of tries**

**transition** to **Confirm Line**

if **get ACK** or **DC1**

**transition** to **Send Data**

### **Send Data**

Transmit next packet

**transition** to **Wait for Packet ACK**

### **Wait for Packet ACK**

set listener for ACK and DC1

Wait for Event

Set **timeout** to 5ms

if **timeout** > 5ms

increment **number of tries**

**transition** to **Send Data**

if **get ACK** or **DC1** (packet transfer was successful)

flip header sync bit (1->0, or 0->1)

**transition** to **Sender side: Check Priority State**

### **Sender side: Check Priority State**

if we have priority and they don’t have priority

**transition** to **Idle** state

else

**transition** to **Wait** state