11/18/2018

Final Assignment

COMP 3980

Version 2.0

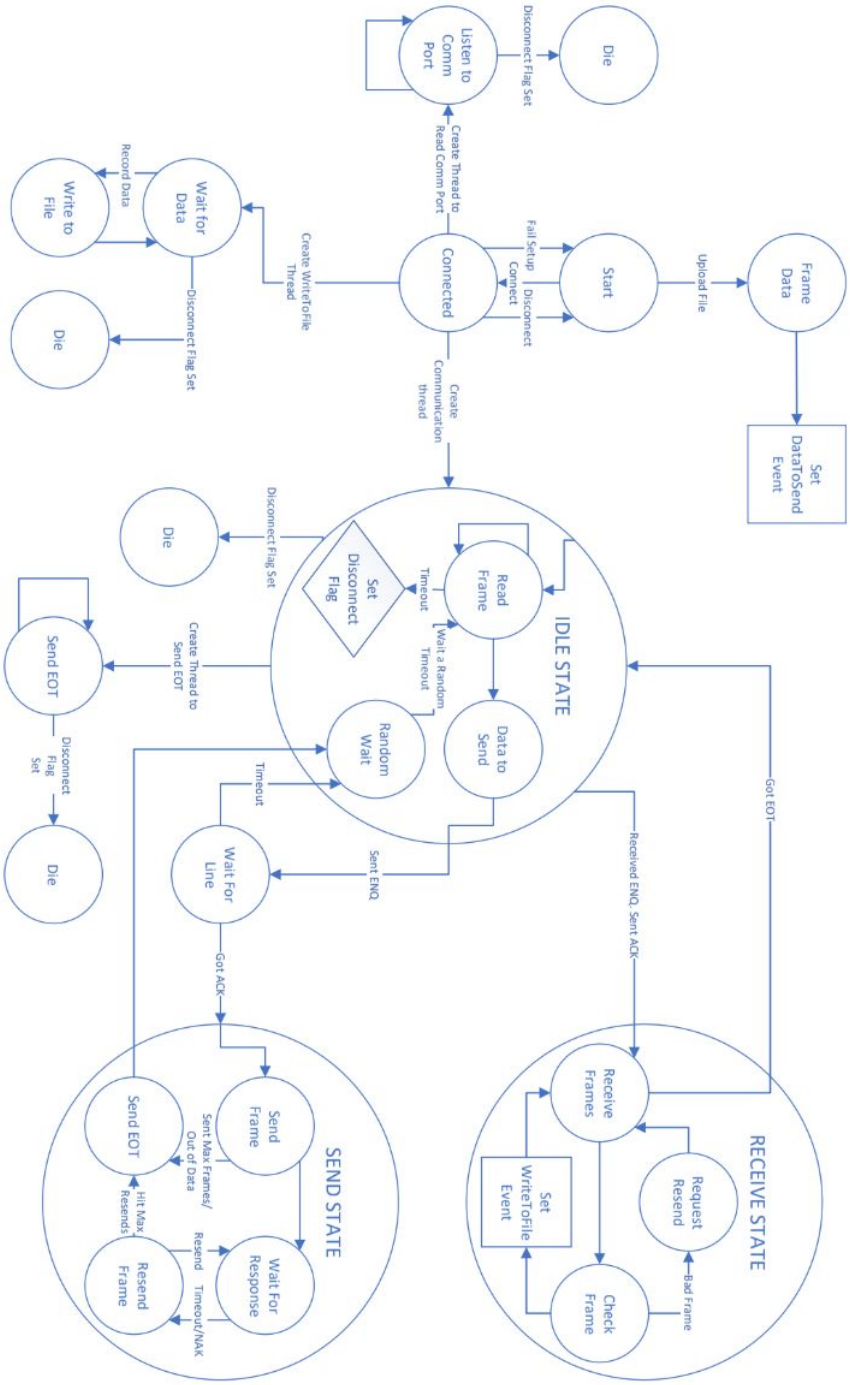
Phat Le, Jacky Li, Simon Wu, Cameron Roberts
TEAM 3

Table of Contents

1.0 Project Design

This section explains the design of the application. There will be three total threads, the main UI thread, writeToFile thread and Communication Thread.

1.1 State Diagram



1.2 Pseudocode

i. Main Thread:

Start

Initialize control characters constexpr (Refer to Protocol Spec Page 7 Frame Makeup)
Initialize the timeouts
Refer to Design Spec Page 18
Initialize Structs containing events for each state (Refer to State Diagram)
ComInputStruct
Port Handle
vector<string> frames
Initialize hFrameReceived Event
bool* connected
dataEventStruct
vector<string> data
initialize hEvent
bool* connected
idleStruct
Port Handle
Initialize hInIdle Event
bool* connected
ProtocolStruct
dataEventStruct frameData
dataEventStruct fileData
comInputStruct comInput
idleStruct is
bool connected
make connected pointers in the structs point to connected in this struct
Create window GUI
Connect Button
Upload Button
Help Button
Exit Button

Connected

Open Port connection with user specified settings
Set Flag saying we are connected in **ProtocolStruct**
Create Read Comm Port Thread
Create Communication Thread
Create WriteToFile Thread
Receiving ESC character will close port and return to Start

Frame Data

Initialize Struct to store all the framed data
Read the entire uploaded file and frame all the data (Refer to Protocol Spec Page 7 Frame Makeup)
Set **DataToSend Event**

ii. Read Comm Port Thread:

Listen to Comm Port

Receive Comm Port Handle
While connected
Wait for comm port input
Stored received data into the **ComInputStruct**
Set **hFrameReceived event**
Go to Die

Die

Close the thread

iii. WriteToFile Thread:

Wait for Data

While connected
Wait until **receivedDataEvent** is set to write frames to file
Create a new file to write data
if file currently exist, increment file number (ie. file2, file3, file4 … fileN)
Call WriteToFile when event is triggered
Reset **receivedDataEvent**
Go to Die

Write to File

Read data from the received frame
Print the data into the file
Go back to Wait for Data

Die

Close the thread

iv. Communication Thread:

Die (IDLE STATE, SEND STATE, RECEIVED STATE)

Close the thread

Read Frame

Wait for **hFrameReceived Event**
Read data from **ComInputStruct**
If the frame is received
Return the frame
If timeout (Refer to Design Spec Page 18)
Return 0

IDLE STATE

Create Send EOT Thread

While connected
Set **hInIdle event**
Idle Read

Idle Read
Read Frame
If frame is EOT
Reset Timeout (Refer to Design Spec Page 18)

If ENQ is received
Reset **hInIdle event**
Send ACK and go to RECEIVE STATE
If you uploaded a file
Reset **hInIdle event**
go to Data to Send
If Timeout is reached (Refer to Design Spec Page 18)
Go to Die

Data to Send
Send ENQ
Go to Wait For Line

Random Wait
Initialize a random number generator (1ms to 10000ms)
Wait for a random time while reading for ENQ
If ENQ is received
Go to RECEIVE STATE
Return to Idle Read

Wait For Line
Read Frame
If Frame is ACK
Go to SEND STATE
If Timeout (Refer to Design Spec Page 18)
return to Random Wait

SEND STATE
Initialize FrameSent counter to 0
Initialize ResentFrame counter to 0

Send Frame
If FrameSent < MaxSend (10)
Read from the **DataEventStruct** and send a frame
Increment FrameSent counter
Go to Wait For Response
else
Go to Send EOT

Wait for Response
Read Frame (Refer to Design Spec Page 18)
If the frame received is ACK
Reset ResentFrame to 0
Go to Send Frame
If the frame received is EOT
Go to Send EOT
If the frame received is NAK
Go to Resend Frame
If the frame received is anything else
Do nothing
Continue to timeout
If Timeout (Refer to Design Spec Page 18)
Go to Resend Frame

Resend Frame
If ResendFrames < MaxResendFrames (3)
Increment ResendFrames
Resend the Frame
Go to Wait For Response

Send EOT
Send EOT
Go back to IDLE STATE

RECEIVED STATE
Receive Frames
Read Frame
If a Data Frame is received
Go to Check Frame
If the frame received is EOT
Go to IDLE STATE
If the frame is corrupted
Do nothing and wait for timeout
When timeout (Refer to Design Spec Page 18)
Send EOT and go to IDLE STATE

Check Frame
Check for CRC in the data frame
If CRC passed
Set WriteToFile Event
Go back to Received Frames
Else
Go to Request Resend

Request Resend
Send NAK

Go to Receive Frames

v. Send EOT Thread

Send EOT

While connected, loop
Wait for **hInIdle event** (Timeout 30sec)
If successful
Send EOT control frame
Sleep 500ms
If timeout
Do nothing
When loop is over, go to Die

Die

Close the thread

2.0 Work Timeline

Team Member	Requires	Task Title	Description	Due Date
Phat		**Create the START state**	- Create Window program - UI properly displayed	**Nov 19**
Phat		**Create port connections**	**-** Read com port settings **-** Configure the com port with the setting **-** Connect to the com port	**Nov 19**
Cameron		**Get IDLE to send EOT and receive EOT**	- Create the loop - Send EOT in the loop **-** Able to receive EOT	**Nov 23**
Simon Wu		**IDLE state receive data**	**-** Able to receive and process other data frame **-** Read and process control data frame	**Nov 23**
Cameron	Send/Receive Frames in IDLE state	**Finish IDLE STATE**	- Finish IDLE state with proper logic flow - Timeouts are implemented - Complete ENQ logic	**Nov 26**

Jacky		**Create Frame Data state**	- Package the data into a frame - Proper headers are placed	**Nov 26**
Phat **Simon Wu**	Frame state completed	**Create SEND state**	**-** Process Control frames **-** Frames are being sent out **-** Resend frame works **-** Timeout logic implemented	**Nov 28**
Cameron **Jacky**	Send State	**Create STATE to receive data frame**	**-** Read data time **-** Send data to write thread	**Dec 2**
Simon Wu		**Writing thread**	**-** Create file to write into **-** Read data from buffer	**Dec 3**
Team	All states	**Integration of code**	**-** Integrate the code to complete the final product	**Dec 3**
Team	Complete product	**Bug test/fixing**	**-** Check for logic errors **-** Edge case testing **-** Fix bugs	**Dec 3**
Phat		**Document**	- Finish all documents needed	**Dec 3**