**ME218C Final Project, 2016**
**Team 6**
**Comm SM Pseudocode (Lobbyist Side)**

**Lobbyist State Machine**
Module Variables: myPriority, txData, rxData, EncKey, txPacketLen, rxPacketLen, currentTxByte, RecState, txAddrMSB, txAddrLSB, curEncByte, lastEncChecksum

**InitCommSM**
Takes nothing, returns nothing

```
Initialize UART
Set priority

End InitCommSM
```

**PostCommSM**
Takes parameter ES_Event ThisEvent, returns true/false Returns ES_PostToService with this service's priority and event

```
End PostCommSM
```

**RunCommSM**
Takes parameter ES_Event ThisEvent, returns ES_Event ReturnEvent

```
Comm state machine
      Waiting4Start state
            If you receive start delimiter, go to waiting for MSB state
      Waiting4MSB state
            if you receive empty byte, go to waiting for LSB state
      Waiting4LSB state
            If you get a byte
                  go to receiving data state
                  set packet len variable
      Receiving data state
            if you're at packet length
                  check checksum
                        if good, post event to appropriate SM
                        else throw away packet
                  go back to waiting2start state
            if not yet at length
                  store byte in data array
                  add new byte to checksum

End CommSM
```

**sendPairPacket**
Takes lobbyist Id, returns nothing

```
set packet len var
load pair packet values into txData array
Calculate Checksum
Send it!

End sendPairPacket
```

**sendStatusPacket**
Takes paired and decryption error bool, returns nothing

```
setPacketLen
set paired bit
load packet values into txData array
load last encrypted checksum
Calculate Checksum
Send it!

End sendStatusPacket
```

**getNextByte**
Takes nothing, returns uint8

```
return next byte from txData array

End getNextByte
```

**isTxComplete**
Takes nothing, returns bool

```
check if we've transmitted the entire packet

End isTxComplete
```

**calcChecksum**
Takes nothing, returns uint8

```
add all the packet bytes
subtract from 0xFF

End calcChecksum
```

**getRxDataAPIId**
Takes nothing, returns uint8

```
return the last rxDataAPIID

End getRxDataAPIId
```

**getPacketType**
Takes nothing, returns uint8

```
if last packet was a receive
      return the protocol packet type

End getPacketType
```

**getPairData**
Takes nothing, returns uint8

```
if last packet was pair request
      return data byte

End getPairData
```

**setTxAddrToLastSender**
Takes nothing, returns nothing

```
if last packet was receive
      store address
```

End setTxAddrToLastSender

**setEncKey**
Takes nothing, returns nothing

```
if last packet was encryption
      store it
```

End setEncKey

**decryptCtrl**
Takes nothing, returns bool

```
store last checksum
Use encryption key to decrypt
check if packet type is CTRL and checksum matches
```

End decryptCtrl

**confirmMSGOrigin**
Takes nothing, returns bool

check message origin against stored value

End confirmMSGOrigin

**checkUnpair**
Takes nothing, returns bool

return true if bit is set

End checkUnpair

**checkBrake**
Takes nothing, returns bool

if brake bit is set return true

End checkBrake

**lastCMDChecksumEqual**
Takes nothing, returns bool

check if the last received packet was a duplicate

End lastCMDChecksumEqual

**resendLastPacket**
Takes nothing, returns nothing

resent counter and resend the last packet

End resendLastPacket

**updatePropulsion**
Takes nothing, returns nothing

set Propulsion to the CTRL byte values

End updatePropulsion