hsv_receiver.c – HSV State Machine pseudo code

---

module functions

```
// determine if were ready to transmit a message
char readyToTransmit(void);

// get the next byte to transmit
char getTransmitByte(void);

// determine if were ready to receive
char readyToReceive(void);

// determine if where to put received byte
void placeReceiveByte(char c);

// see if a full packet has been received
char fullPacketReceived(void);

// say we have processed a packet
void packetProcessed(void);

// Respond to time expired... disconnect to boat
void timeExpired(void);

// get the throttle command
unsigned char getThrottle(void);

// get the direction command
unsigned char getDirection(void);

// get whether or not to shoot
unsigned char getShoot(void);

// get whether we're paired or not
unsigned char getIsPaired(void);

// set whether its paired or not
void setIsPaired(char isPaired);

// set what mode were in (human or zombie
void setIsHuman(char isHuman);

// get what mode were in (human or zombie)
char getIsHuman(void);

// adjust throttle if a zombie
```

```
unsigned char zombieAdjustThrottle(unsigned char orig);

// adjust direction if a zombie
unsigned char zombieAdjustDirection(unsigned char orig);
```

private functions

```
// update state machine to keep track of where we are in state machine
static void updateHSVStateMachine(char packet_type);
// update state machine to keep track of packet state
static void updatePacketStateMachine(char c);
// send control request acknowledged
static void sendControlRequestAcknowledged(void);
// get the address of the controller
static void storeControllerAddress(void);
// clear the buffer and counter keeping track of the current message
static void clearPacketBuffer(void);

// transmit buffer related functions
// pack a message to transmit
static void packMessageToTransmit(void);
// calculate checksum for message to transmit
static char calculateTransmitChecksum(void);

// receive buffer related functions
// get type of packet
static PacketType_t getPacketType(void);
// calculate checksum for message that is received
static char calculateReceiveChecksum(void);
```

---

pseudocode

```
char readyToTransmit(void)
        if (transmitBufferEmpty())
                if ready to transmit
                        return 1
        end if
        return 0
end readyToTransmit
```

```
char getTransmitByte(void)
        txByte = transmitPacket[current_byte_to_transmit]
        increment current_byte_to_transmit

        if (current_byte_to_transmit > message length)
                ready to transmit = 0
                current_byte_to_transmit  = 0
        end if

        return txByte
end getTransmitByte
```

```
char readyToReceive(void)
        if (receiveBufferFull())
                return 1
        else
                return 0
        end if-else
end readyToReceive
```

```
void placeRecieveByte(char c)
        lastReceivedByte  = c
        updatePacketStateMachine(lastReceivedByte)
end placeRecieveByte
```

```
char fullPacketReceived(void)
        return possessFullPacket
end fullPacketReceived
```

```
void packetProcessed(void)
        possessFullPacket = 0
        clearPacketBuffer()→clear contents of receive packet
end packetProcessed
```

```
void timeExpired(void)
        paired = 0
        current_byte_to_transmit = 0
        current_byte_to_receive = 0
        packet_state = PACKET_EMPTY
        HSV_state = WAITING_FOR_CONTROLLER
        clearPacketBuffer() →clear contents of receive packet
        resetTimer() → that counts time between received messages
end timeExpired
```

```
char getThrottle(void)
        return throttle
end getThrottle
```

```
char getDirection(void)
        return direction
end getDirection
```

```
char getShoot (void)
        return throttle
end getShoot
```

```
char getIsPaired (void)
        return isPaired
end getIsPaired
```

```
void setIsPaired (char _isPaired)
        isPaired = _isPaired
end setIsPaired
```

```
--------------------------------------------------------------------------------
char getIsHuman (void)
        return isHuman
end getIsHuman
```

```
--------------------------------------------------------------------------------
void setIsHuman (char _isHuman)
        isHuman = _isHuman
end setIsHuman
```

```
--------------------------------------------------------------------------------
unsigned char zombieAdjustThrottle(unsigned char orig)

        if (speedEnhancement != 0)      // decrease speed by 25%
                newThrottle = 3*origThrottle/4
        end if
        else
                newThrottle = origThrottle/2
        end else

        if (switch forward and reverse)
                newThrottle = 0xFF – newThrottle
        endif

end zombieAdjustThrottle
```

```
--------------------------------------------------------------------------------
unsigned char zombieAdjustDirection(unsigned char orig)

        if (commanded to turn right)
                if (reduction in right turn rate)
                        newDirection = orig/2
                end if
        end if

        else    // if commanded left turn
                if (reduction in left turn rate)
                        newDirection = orig/2
                end if
        end else
end zombieAdjustDirection
--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------
static void updateHSVStateMachine(char packet_type)
        switch (HSV_state)
                case WAITING_FOR_CONTROLLER:
                        switch packet_type
                                case CONTROL_REQUEST
                                        storeControllerAddress()
                                        HSV_state = WAITING_FOR_COMMAND
                                        isPaired = 1
                                        send acknowledge message
                                        break;
                                default
                                        break;
                        break
                case WAITING_FOR_COMMAND:
                        switch packet_type
                                case STATUS_MESSAGE
                                        if last acknowledge not received, send another
                                        break;
                                case NECROMANCER_MESSAGE
                                        store whether or not to enhance speed
                                        store whether or not to mess up right turn
                                        store whether or not to mess up left turn
                                        store whether or not to reverse throttle, direction
                                        break;
                                case CONTROL_MESSAGE
                                        acknowledge that a message was received
                                        throttle = receivePacket[THROTTLE_BYTE]
                                        direction = receivePacket[DIRECTION_BYTE]
                                        shoot = receivePacket[SHOOT_BYTE]
                                        break;
                                default
                                        break;
                        break
                default
                        break

end updateHSVStateMachine
```

```
--------------------------------------------------------------------------------
static void updatePacketStateMachine(char c)
        switch (packetState)
                case PACKET_EMPTY
                        current_in_receive_packet = 0
                        if (c == STARTBYTE)
                                receivePacket[current_in_receive_packet] =c
```

```
                        increment current_in_receive_packet
                        packetState = PACKET_IN_PROGRESS
                        receivePacketLength = MAX_PACKET_LENGTH  (for now)
                end if
                break;
        case PACKET_IN_PROGRESS
                receivePacket[current_in_receive_packet] =c
                increment current_in_receive_packet

                if (current_in_receive_packet == 3)
                        receivePacketLength = length_msb << 8 + length_lsb
                        receivePacketLength += 4 (start + length + checksum)
                end if

                if (packet one char away from full)
                        packetState = PACKET_FULL
                end if
                break
        case PACKET_FULL
                receivePacket[current_in_receive_packet] =c
                increment current_in_receive_packet

                if (calculateReceiveChecksum() == 0xFF)
                        possessFullPacket = 1
                        packet_type = getPacketType()
                        updateHSVStateMachine(packet_type)
                end if
                else
                        clearPacketBuffer()
                end if

                packetState = PACKET_EMPTY
                resetTimer()      // timing time in between full packets
                startTimer(1 second)
                break
        default
                break
end updatePacketStateMachine
-------------------------------------------------------------------------------------


-------------------------------------------------------------------------------------
static void sendControlRequestAcknowledged(void)
        packMessageToTransmit();
        current_byte_to_transmit = 0
        ready to transmit = 1
end sendControlRequestAcknowledged
```

```
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
static void storeControllerAddress(void)
        address_lsb = receivedPacket[ADDRESS_LSB]
        address_msb = receivePacket[ADDRESS_MSB]
end storeControllerAddress
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
static void clearPacketBuffer(void)
        go through receive buffer and clear all contents
end clearPacketBuffer
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
static void packMessageToTransmit(void)
        store appropriate message in transmit buffer

        if (responseType == CONTROL_REQUEST_ACK)
                transferLength = 0x07
        else if (responseType == CONTROL_MESSAGE_ACK)
                transferLength = 0x06

        transmitBuffer[0] = 0x7e
        .
        .
        .
        transmitBuffer[2] = transferLength
        .
        .
        .

end packMessageToTransmit
--------------------------------------------------------------------------------


--------------------------------------------------------------------------------
static PacketType_t getPacketType(void)

        api indicator = receivePacket[API_INDICATOR_BYTE_NUM]

        if (api indicator == 0x81)
```

```
                message indicator = receivePacket[MESSAGE_INDICATOR_BYTE_NUM]
                if (message indicator == 0x03)
                        return CONTROL_REQUEST
                else if (message indicator == 0x01)
                        return NECROMANCER MESSAGE
                else if (message indicator == 0x02)
                        return CONTORL_MESSAGE
                else
                        return OTHER
                end if-else
        else if (api indicator == 0x89)
                return STATUS_MESSAGE
        else
                return OTHER
        end if-else
end getPacketType
```

```
static char calculateTransmitChecksum(void)
        total = sum (transmitPacket bytes 3 → transmitLength+3)
        total = 0xFF – total
        return total
end calculateTransmitChecksum
```

```
static char calculateReceiveChecksum(void)
        total = sum (receivePacket bytes 3 through receiveLength-1)
        return total
end calculateReceiveChecksum
```
