Opener - Embedded Software Skills Assignment

Instructions

In this assignment, you are asked to write a serial message parser for the message format described in this document. The program will loop indefinitely and continue to parse incoming messages on the serial port.

This assignment is to be completed in either C or C++. Helper functions are provided to receive data from the serial port and to indicate when a complete message has been received.

Notes:

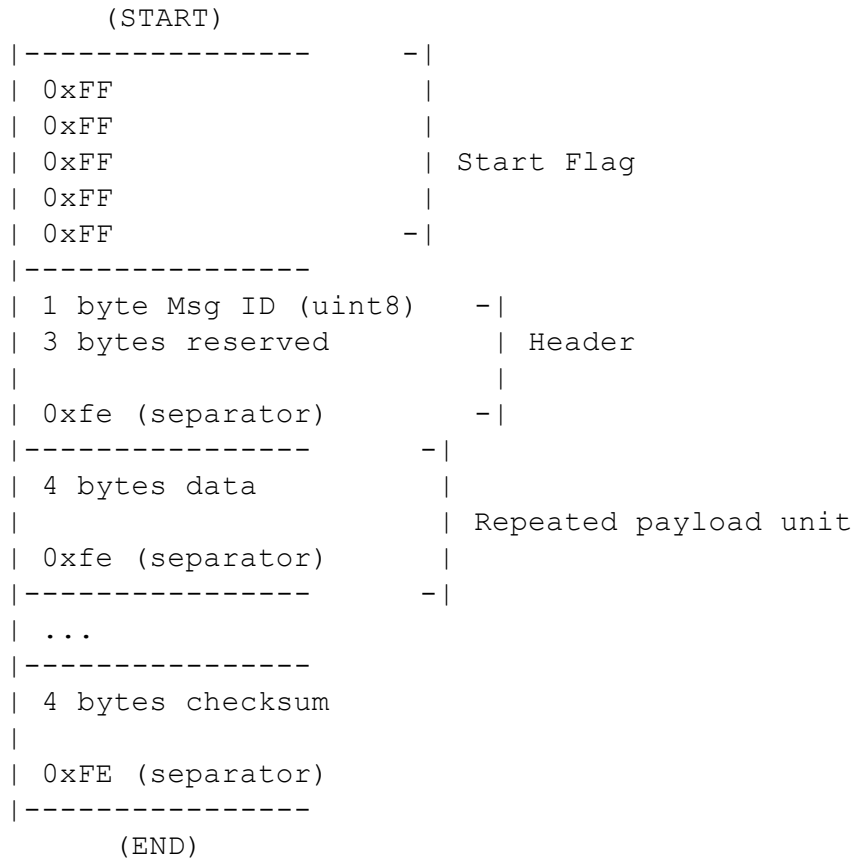
- Discard any data received outside of messages.
- Assume that garbage data between messages will not form a start flag.
- Once a start flag is received, a full message will follow.
- Discard any messages that do not correspond with their checksum.
- Data is sent to the serial port asynchronously. Data for an entire message may not be available immediately.
- Do your best to write this as you would production code.
- Assume the serial port is already opened and configured. The serial port can only be accessed through the `read()` helper function.
- You may use standard libraries.
- `process_message()` should only be called with a buffer that contains a complete message with a valid checksum.
- Assume data is sent in little-endian order.

Please complete this assignment and email your solution back to us by the specified time. You may use internet resources for reference only (e.g. standard library syntax). The submitted solution should be your own work. Please avoid discussing the problem or soliciting help from others.

Message Format

All messages contain (in this order):

- 5 start bytes (0xFF)
- 1 byte representing the message ID
- 3 reserved bytes
- 1 separator byte (0xFE)
- Some number of repeated payload units which contain 4-bytes of data followed by a separator byte (0xFE)
- 4 byte checksum
- 1 separator byte (0xFE)



The number of repeated payload units is fixed for a specific message ID. The message length for each message ID is known and provided through a helper function. The maximum possible size of a message (excluding the start flag) is 100 bytes.

Checksum

Every message contains a 32-bit checksum just before the final separator byte. The checksum is a 32-bit bitwise XOR of the message header and all payload units. The start bytes (0xFF) and separator bytes (0xFE) are not included in the checksum. Messages with invalid checksums should be discarded.

Helper Functions

Assume that the following functions have already been implemented and can be used in your program.

```
/**
 * Attempts to read up to count number of bytes from the serial
 * port into the buffer.
 *
 * Returns the number of bytes actually read.
 */
int read(char *buffer, unsigned int count);


/**
 * This function should only be called when a complete message
 * has been parsed. The complete message should be at the start
 * of the input buffer.
 */
void process_message(char *buffer, unsigned int message_id);


/**
 * Returns the size (in bytes) of the message with the specified
 * message_id. The message size includes the separator bytes
 * (0xFE) and the checksum but does not include the start flag.
 */
int get_message_size_from_message_id(int message_id);
```